cadence™

Conformal®Constraint Designer Rule Reference

Product Version 23.2
October 2023

© 2004-2023 Cadence Design Systems, Inc. All rights reserved.
Printed in the United States of America.

Cadence Design Systems, Inc., 2655 Seely Avenue, San Jose, CA 95134, USA

Trademarks: Trademarks and service marks of Cadence Design Systems, Inc. (Cadence) contained in this document are attributed to Cadence with the appropriate symbol. For queries regarding Cadence's trademarks, contact the corporate legal department at the address shown above or call 800.862.4522.

Open SystemC, Open SystemC Initiative, OSCI, SystemC, and SystemC Initiative are trademarks or registered trademarks of Open SystemC Initiative, Inc. in the United States and other countries and are used with permission.

All other trademarks are the property of their respective holders.

Restricted Print Permission: This publication is protected by copyright and any unauthorized use of this publication may violate copyright, trademark, and other laws. Except as specified in this permission statement, this publication may not be copied, reproduced, modified, published, uploaded, posted, transmitted, or distributed in any way, without prior written permission from Cadence. This statement grants you permission to print one (1) hard copy of this publication subject to the following conditions:

1. The publication may be used solely for personal, informational, and noncommercial purposes;
2. The publication may not be modified in any way;
3. Any copy of the publication or portion thereof must include all original copyright, trademark, and other proprietary notices and this permission statement; and
4. Cadence reserves the right to revoke this authorization at any time, and any such use shall be discontinued immediately upon written notice from Cadence.

Disclaimer: Information in this publication is subject to change without notice and does not represent a commitment on the part of Cadence. The information contained herein is the proprietary and confidential information of Cadence or its licensors, and is supplied subject to, and may be used only by Cadence's customer in accordance with, a written agreement between Cadence and its customer. Except as may be explicitly set forth in such agreement, Cadence does not make, and expressly disclaims, any representations or warranties as to the completeness, accuracy or usefulness of the information contained in this document. Cadence does not warrant that use of such information will not infringe any third party rights, nor does Cadence assume any liability for damages or costs of any kind that may result from use of such information.

Cadence is committed to using respectful language in our code and communications. We are also active in the removal and/or replacement of inappropriate language from existing content. This product documentation may however contain material that is no longer considered appropriate but still reflects long-standing industry terminology. Such content will be addressed at a time when the related software can be updated without end-user impact.

Restricted Rights: Use, duplication, or disclosure by the Government is subject to restrictions as set forth in FAR52.227-14 and DFAR252.227-7013 et seq. or its successor.

Contents

© 1999-2023 All Rights Reserved.

© 1999-2023 All Rights Reserved.

© 1999-2023
All Rights Reserved.

© 1999-2023
All Rights Reserved.

© 1999-2023
All Rights Reserved.

© 1999-2023
All Rights Reserved.

© 1999-2023 All Rights Reserved.

© 1999-2023 All Rights Reserved.

© 1999-2023
All Rights Reserved.

© 1999-2023 All Rights Reserved.

© 1999-2023 All Rights Reserved.

© 1999-2023 All Rights Reserved.

1

About This Manual

This manual documents rule checks for the Encounter® Conformal® Constraint Designer software.

Audience

This manual is written for experienced designers of digital integrated circuits who must be familiar with RTL, synthesis, and design verification; as well as having a solid understanding of UNIX and Tcl/Tk programming.

UNIX Commands

To execute a UNIX command from within the Conformal Constraint Designer or a Conformal Constraint Designer command script, start the line with an exclamation point "!" or with the SYSTEM command. When you execute commands in this way, they display to the standard output, and the Conformal Constraint Designer records them in a log file, if one is active.

Pseudo-UNIX Commands

The Conformal Constraint Designer supports the following list of UNIX-like commands:

```
cd
date
echo
hostname
ls
printenv
pwd
```


© 1999-2023 All Rights Reserved.

```
setenv
```

These pseudo-UNIX commands do actions that are similar to their UNIX operating system counterparts, but do not support all command options. Additionally, their behavior can differ from the behavior found in UNIX command descriptions. The advantage of using these commands is that results will not vary from one UNIX operating system to another.

Conventions

Abbreviations commonly found in this manual are as follows:

Abbreviation	Definition
FP	False path
MCP	Multi-cycle path
SDC	Synopsys Design Constraints

The command syntax structure is as follows:

Convention	Definition
`Bold Case`	Indicates the command name.
`UPPERCASE`	Indicates the required minimum character entry.
`< >`	Indicates required arguments. Do not type the angle brackets.
`[ ]`	Indicates optional arguments. Do not type the square brackets.
|	Indicates a choice among alternatives. Do not type the vertical bar.
`\`	The backslash character (\) at the end of a line indicates that the command you are typing continues on the next line.
`...`	Indicates multiple entries of an argument.
`*`	Indicates that the entry can use the wildcard (*) to represent zero or more characters.

© 1999-2023 All Rights Reserved.

2

Related Documents

The following lists the documents related to Conformal Constraint Designer:

Document	Description
Conformal Constraint Designer User Guide	Describes how to use theConformal Constraint Designer solution.
Conformal Constraint Designer Command Reference	Describes the commands for Conformal Constraint Designer.
Conformal Constraint Designer Database Access Object and Attribute Reference	Describes the database access objects and attributes.
Conformal Constraint Designer Rule Check Reference	Describes the modeling messages, policy rule checks lint rule checks, the CDC rule checks, and the atomic checks.
Conformal HDL Rule Check Reference	Describes the HDL rule checks that apply to all Conformal tools.

© 1999-2023 All Rights Reserved.

© 1999-2023
All Rights Reserved.

3

Modeling Rule Checks

This chapter contains the modeling messages for the Conformal Constraint Designer software.

Modeling rules are not checked by default. To enable modeling rules, you must add them to a *rule set* or read in the `cfm_default_modeling_ruleset.vpx` default rule set using the `ADD RULE SET` command..

Definitions for Modeling Rules are explained in the following sections with demonstration examples. This chapter includes the following rule categories:

© 1999-2023 All Rights Reserved.

Structural Rules

This category of rules checks for design structural errors. The following lists the Structural (STRC) rule checks:

© 1999-2023 All Rights Reserved.

STRC1.1

Message

```
Strong combinational loop detected
```

Default Severity

Warning

Description

The design includes a combinational loop that does not contain any weak devices. This could cause signals to oscillate and reflects a potential error in the design.

If this reflects design intent, you can ignore it. It will not affect verification. If this does not reflect design intent, correct the design. First, generate a report on the combinational loop using the `REPORT PATH` command:

```
report path -loop
```

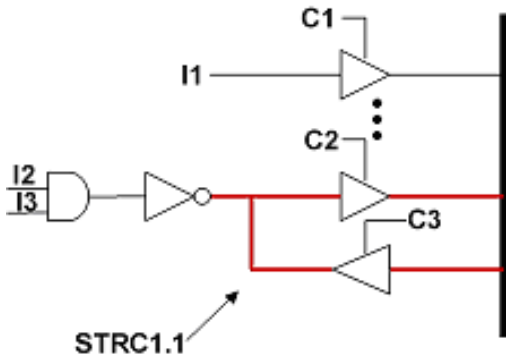
The presence of this violation does not automatically indicate there is a problem in the design. For example, a combinational loop might be in a false path.

Note: Any proofs containing a functional loop may generate inconsistent results if the loop oscillates.

© 1999-2023 All Rights Reserved.

Example

In the following figure, a strong combinational feedback loop exists if `C2` and `C3` can be active at the same time.



© 1999-2023
All Rights Reserved.

STRC1.2

Message

`Weak combinational loop detected`

Default Severity

Warning

Description

The design includes a combinational loop that contains at least one weak device. This could cause signals to oscillate and reflects a potential error in the design.

If this reflects design intent, you can ignore it. It will not affect verification. If this does not reflect design intent, correct the design. First, generate a report on the combinational loop using the `REPORT PATH` command:

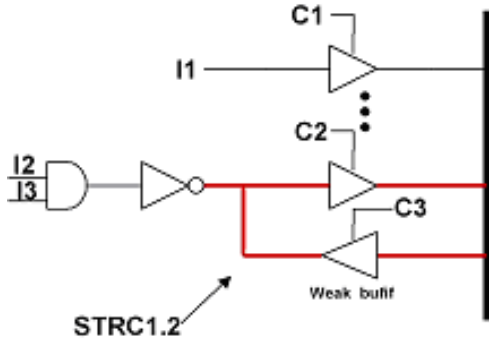
```
report path -loop
```

The presence of this violation does not automatically indicate there is a problem in the design. For example, a combinational loop may be in a false path.

Note: Any proofs containing a functional loop may generate inconsistent results if the loop oscillates.

Example

In the following figure, a strong combinational feedback loop exists if `C2` and `C3` can be active at the same time.



© 1999-2023
All Rights Reserved.

STRC2

Message

`Undriven fanin detected`

Default Severity

Warning

Description

The design includes undriven pins. They are the subset of the floating pins of the design that are actually driving some logic.

Learn more about where these violations occurred by viewing the Modeling Rule Manager in the GUI mode or list them with one of the following `REPORT RULE CHECK` commands:

- `report rule check STRC2 -summary`
- `report rule check STRC2 -verbose`

Use the "Browse Gate" (schematic view) in the GUI mode to graphically pinpoint the source of the violation and the Source Code Browser to examine the origin. Modify the design accordingly.

An undriven fan-in remains undriven even if you SET it to a value with the `SET UNDRIVEN SIGNAL` command. Thus, the STRC2 message remains. However, an undriven fan-in becomes driven when you TIE it to a value with the `ADD TIED SIGNAL` command. Thus, the Conformal Constraint Designer will not generate the STRC2 message.

The presence of an undriven fan-in may not indicate problems with the design; but this warning points to *potential* problem areas.

Important

You must resolve undriven signals for assertions or clocks. Floating signals will not allow proper sequential verification. They generate a vacuous proof.

© 1999-2023 All Rights Reserved.

Example

The following example shows a Verilog code with a floating signal. Note that `x4` is not being driven by anything, but it is driving the D input of a flip flop. (See line 20, in bold.) This generates an STRC2 message as well as an STRC3 message.

```
1.  module test (in1, in2, in3,
2.                reset_bar, clk,
3.                en1, en2, en3,
4.                out1);
5.
6.  input in1, in2, in3;
7.  input en1, en2, en3;
8.  input reset_bar, clk;
9.  output out1;
10.
11. wire x1, x2, x3;
12. wire dq1, dq2, dq3;
13.
14. IV U_inv1 (.Z(x1), .A(en1));
15. IV U_inv2 (.Z(x2), .A(en2));
16. IV U_inv3 (.Z(x3), .A(en3));
17.
18. FD2 U_dff1 (.Q(dq1), .D(x1), .CP(clk), CD(reset_bar));
19. FD2 U_dff2 (.Q(dq2), .D(x2), .CP(clk),  .CD(reset_bar));
20. FD2 U_dff3 (.Q(dq3), .D(x4), .CP(clk),  .CD(reset_bar));
21.
22. BTS4 U_trist1 (.Z(out1), .A(in1), .E(dq1));
23. BTS4 U_trist2 (.Z(out1), .A(in2), .E(dq2));
24. BTS4 U_trist3 (.Z(out1), .A(in3), .E(dq3));
```


© 1999-2023 All Rights Reserved.

```
1.  module test (in1, in2, in3,
25.
26. endmodule
```


© 1999-2023
All Rights Reserved.

STRC3

Message

```
Input to DFF/DLAT is always high-impedance (Z)
```

Default Severity

Error

Description

The design includes signals at the inputs of the flip-flop or latch that are always high-impedance (Z). These may be undriven (that is, disconnected) or connected to logic such as a constantly disabled buffer. You must resolve the situation prior to verification.

Learn more about where these violations occurred by viewing the Modeling Rule Manager in the GUI mode or list them with one of the following `REPORT RULE CHECK` commands:

- `report rule check STRC3 -summary`
- `report rule check STRC3 -verbose`

Use the "Browse Gate" (schematic view) in the GUI mode to graphically pinpoint the source of the violation and the Source Code Browser to examine the origin. Modify the design accordingly.

This message can be caused by an undriven input to a flip-flop or latch. Thus, the Conformal Constraint Designer also reports STRC2. Or it can be caused by an input that is tied to Z (for example, tied to the output of a bufif1 when enable is 0.) In this case, Conformal Constraint Designer does not report STRC2.

To remove this message, either physically connect the input to the DFF/DLAT to a non-Z value, or use the `ADD TIED SIGNAL` or `SET UNDRIVEN SIGNAL` (non-Z) command.

Important

You must resolve undriven DFFs and DLATs prior to verification.

© 1999-2023 All Rights Reserved.

Example

The following example shows a Verilog code with a floating signal. Note that `x4` is not being driven by anything, but it is driving the D input of a flip flop. (See line 20, in bold.) This generates an STRC2 message as well as an STRC3 message.

```
1.  module test (in1, in2, in3,
2.                reset_bar, clk,
3.                en1, en2, en3,
4.                out1);
5.
6.  input in1, in2, in3;
7.  input en1, en2, en3;
8.  input reset_bar, clk;
9.  output out1;
10.
11. wire x1, x2, x3;
12. wire dq1, dq2, dq3;
13.
14. IV U_inv1 (.Z(x1), .A(en1));
15. IV U_inv2 (.Z(x2), .A(en2));
16. IV U_inv3 (.Z(x3), .A(en3));
17.
18. FD2 U_dff1 (.Q(dq1), .D(x1), .CP(clk), CD(reset_bar));
19. FD2 U_dff2 (.Q(dq2), .D(x2), .CP(clk),  .CD(reset_bar));
20. FD2 U_dff3 (.Q(dq3), .D(x4), .CP(clk),  .CD(reset_bar));
21.
22. BTS4 U_trist1 (.Z(out1), .A(in1), .E(dq1));
23. BTS4 U_trist2 (.Z(out1), .A(in2), .E(dq2));
24. BTS4 U_trist3 (.Z(out1), .A(in3), .E(dq3));
```


© 1999-2023 All Rights Reserved.

```
1.  module test (in1, in2, in3,
25.
26.  endmodule
```


© 1999-2023
All Rights Reserved.

STRC4

Message

`Set/Reset port of the DFF/DLAT is always enabled`

Default Severity

Warning

Description

The design includes floating signals at the inputs of DFFs and DLATs.

Learn more about where these violations occurred by viewing the Modeling Rule Manager in the GUI mode or list them with one of the following `REPORT RULE CHECK` commands:

- `report rule check STRC4 -summary`
- `report rule check STRC4 -verbose`

© 1999-2023 All Rights Reserved.

STRC5.1

Message

```
Clock port of the DFF is tied
```

Default Severity

Warning

Description

The design includes a clock signal that is tied to low or high logic.

Learn more about where these violations occurred by viewing the Modeling Rule Manager in the GUI mode or list them with one of the following `REPORT RULE CHECK` commands:

- `report rule check STRC5.1 -summary`
- `report rule check STRC5.1 -verbose`

© 1999-2023
All Rights Reserved.

STRC5.2

Message

```
Clock port of the DLAT is always disabled
```

Default Severity

Warning

Description

The design includes a clock signal of a DLAT that is tied to ground.

Learn more about where these violations occurred by viewing the Modeling Rule Manager in the GUI mode or list them with the `REPORT RULE CHECK` command:

- `report rule check STRC5.2 -summary`
- `report rule check STRC5.2 -verbose`

© 1999-2023 All Rights Reserved.

STRC5.3

Message

```
Clock port of the DLAT is always enabled (transparent DLAT)
```

Default Severity

Note

Description

The design includes a clock signal of a DLAT that is tied to high with disabled Set and Reset.

Learn more about where these violations occurred by viewing the Modeling Rule Manager in the GUI mode or list them with one of the following `REPORT RULE CHECK` commands:

- `report rule check STRC5.3 -summary`
- `report rule check STRC5.3 -verbose`

© 1999-2023 All Rights Reserved.

STRC6.1

Message

```
Data port of the DFF is tied
```

Default Severity

Warning

Description

The design includes a DFF whose data port is tied to a 1 or 0.

© 1999-2023 All Rights Reserved.

STRC6.2

Message

```
Data port of the DLAT is tied
```

Default Severity

Warning

Description

The design includes a DLAT whose data port is tied to a 1 or 0.

© 1999-2023 All Rights Reserved.

STRC7

Message

`Set/Reset port of DFF/DLAT is driven by a multi-input logic gate`

Default Severity

Warning

Description

The tool will trace backward from the asynchronous set or reset port of DFF or DLAT and flag the violation if it reach a multi-input logic gate that has more than one possible sensitization input.

© 1999-2023 All Rights Reserved.

Initialization Rules

In order to properly initialize a design and perform a valid verification, the Conformal Constraint Designer uses the initialization rules to check for certain conditions in the design. The following lists the Initialization (INIT) rule checks:

© 1999-2023 All Rights Reserved.

INIT1

Message

```
Inconsistent initial state setting
```

Default Severity

Warning

Description

An INIT1 violation occurs when any part of circuit has an inconsistent initial state setting. An inconsistent initial state can occur in one of two ways:

1. Two different initialization commands try to initialize a state element to different values. In this case, the latter value overrides the previous, and Conformal Constraint Designer issues the INIT1 warning.
2. After applying initialization commands, the Conformal Constraint Designer performs combinational propagation to determine if previously initialized values (initialized by the initialization commands) are changed by the combinational propagation. If so, the new value overrides the old, and the Conformal Constraint Designer issues the INIT1 warning message.

Inconsistencies in initial value settings should be resolved as they might cause the Conformal Constraint Designer to produce incorrect results. To resolve this problem, return to the Setup system mode and delete the initialization command that caused the violation.

Note: Resolve any CNST1 violations before trying to resolve INIT1 and INIT2 violations. Resolving CNST1 violations first will help resolve any relevant INIT1/INIT2 violations.

© 1999-2023 All Rights Reserved.

INIT2

Message

```
Initial states are unjustifiable
```

Default Severity

Error

Description

The design includes initial output values of flip-flops or latches that are not attainable based on the initial input values.

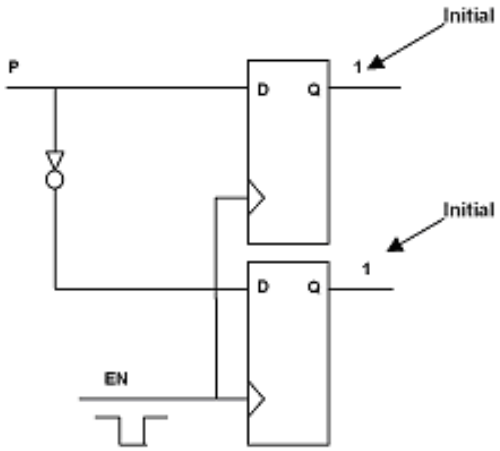
INIT2 is usually caused by a combination of conflicting initialization commands and/or constraints. Thus, during diagnosis with the GUI, the INIT2 schematic contains all the gates directly involved in the warning message. The schematic also displays the fan-ins of those gates (two to three levels deep).

Note: Resolve any CNST1 violations before trying to resolve INIT1 and INIT2 violations. Resolving CNST1 violations first will help resolve any relevant INIT1/INIT2 violations.

Example

© 1999-2023 All Rights Reserved.

The following figure shows an example of an `INIT2` rule check violation.



© 1999-2023
All Rights Reserved.

INIT3

Message

`Uninitialized DFF/DLAT`

Default Severity

Warning

Description

The design includes a flip-flop or latch that is not initialized. In this case, the output port of the flip-flop or latch takes the assumed "X" value.

Use the `REPORT GATE` command to report which flip-flops and latches were not initialized correctly.

Note: This automatically assigns ID numbers. They can differ from one version to another. Always use the full path in dofiles and any time you rerun a design with a different version.

© 1999-2023 All Rights Reserved.

INIT4

Message

`Unused initial state value from VCD`

Default Severity

Warning

Description

There is no corresponding hierarchical pattern in the design matching the extracted hierarchical pattern from the VCD.

You can find out more about where these violations occurred by viewing the Modeling Rule Manager in the GUI mode or list occurrences (in detail) with the `REPORT RULE CHECK` command:

```
report rule check INIT4 -verbose
```


© 1999-2023 All Rights Reserved.

Constraints

The Conformal Constraint Designer looks for design over-constrained errors. The following lists the rule check:

- CNST1 on page 44

© 1999-2023 All Rights Reserved.

CNST1

Message

```
System is over-constrained
```

Default Severity

Error

Description

The design includes too many constraints to a circuit.

It is possible for a designer to insert either too few or too many constraints to a circuit. In the case of the latter, overconstrained specifications occur when two conflicting assignments to the same variable are present during the same time frame, or when contradicting constraints have been assigned. In this case, the proof yields meaningless results because certain conditions would never occur due to the specified constraints.

Remove the improper pin constraint with the `DELETE PIN CONSTRAINTS` command:

Note: CNST1 violations must be resolved before you proceed to proving the circuit's properties.

Example

The following SDC commands contradict each other, causing the design to become over-constrained (assuming `or1` and `nor1` share the same input signals):

```
set_case_analysis 1 [get_pins or1/Y]
set_case_analysis 1 [get_pins nor1/Y]
```


© 1999-2023 All Rights Reserved.

4

SDC Lint Rule Checks

This chapter contains the SDC lint rule checks that the Conformal Constraint Designer uses to verify SDC data.

SDC lint rule checks are built-in quality rule checks. You can configure these rules, but you cannot disable them.

Unlike RTL lint rules, you can run SDC lint rules to check that your SDC files comply with the SDC language syntax. RTL lint rules cannot be configured in this way.

SDC lint rules ensure that the SDC is free of syntax violations.

Rule Categories:

© 1999-2023 All Rights Reserved.

SDC_LINT_CMD*

This section describes the SDC rule checks that relate to command usage.

© 1999-2023
All Rights Reserved.

SDC_LINT_CMD1

Message

`Unknown command used`

Default Severity

Error

Description

Indicates that there is a command in an SDC file that the software does not recognize.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes a rule check violation because the software does not recognize the `create_virtual_clock` command:

```
create_virtual_clock -name clka -period 10
```


© 1999-2023 All Rights Reserved.

SDC_LINT_CMD2

Message

`CCD unsupported command used`

Default Severity

Error

Description

Indicates that an unsupported command is used in an SDC file. The software recognizes this command, but ignores it.

Commands can also be configured as unsupported by setting the `usage` command parameter to a value of `unsupported`.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes a rule check violation because the software does not recognize the `read_verilog` command:

```
read_verilog
```


© 1999-2023 All Rights Reserved.

SDC_LINT_CMD3

Message

`User defined forbidden command used`

Default Severity

Warning

Description

Indicates that a forbidden command is used in an SDC file.

To configure a command as forbidden, use the `configure_lint_check` command for that command. This command has a parameter called `usage`, where you can specify the command as `forbidden`.

Example

Suppose you want to forbid the use of the `set_dont_touch` command in your SDC files. To configure `set_dont_touch` as forbidden, use the following command in a lint configuration file:

```
configure_lint_check set_dont_touch { command { usage forbidden } }
```

After reading that configuration file, the following command causes a rule check violation because the `set_dont_touch` command is used:

```
set_dont_touch [get_nets n1]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_CMD4

Message

`Non-SDC command used`

Default Severity

Warning

Description

Indicates that a command that does not comply with the currently selected version of SDC is used. For a list of all the SDC-compliant commands and options:

```
<release>/share/cfm/ccd/gui/tcl/sdccmd.<sdc_version>
```

To select an SDC version, set the `sdc_version` Tcl variable (see example). To check the latest supported version, use the following commands:

```
tclmode
puts $sdc::sdc_version
```

This is checked when you read in your SDC files with the `READ SDC` command.

To configure a command as non-SDC compliant, use the `configure_lint_check` command for that command. This command has a `is_sdc_compliant` parameter that you can set to `no`.

Note: If a command is not SDC compliant, it can still be among the commands/options supported by the tool. For a list of the non-SDC commands that the Conformal software supports, refer to the Appendix called "Supported Non-SDC Commands" in the *Conformal Constraint Designer User Guide*.

Example

The following commands cause a rule check violation because `set_ideal_net` is not compliant with SDC version 1.4:

```
set sdc_version 1.4
set_ideal_net [get_nets n1]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_CMD5

Message

`Command not compliant with Tcl syntax`

Default Severity

Error

Description

Indicates a syntax error in an SDC file.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes a rule check violation because there is a missing close bracket after `U1`:

```
set_false_path -to [get_cells U1
```


© 1999-2023 All Rights Reserved.

SDC_LINT_CMD6

Message

`Command overwritten`

Default Severity

Warning

Description

Indicates that a command specifies constraints that duplicate or overwrite existing constraints from a previous command. For timing exceptions, only commands with the same path specifications are reported. For other commands, this rule will report an SDC command even if only part of the new attributes duplicate or overwrite existing ones. In many cases, one of the two commands reported by this rule could be removed from the SDC file without changing its meaning.

This rule is flagged when the `configure_lint_check`'s `warn_if_overwritten` parameter for `command` is set to `yes`. By default, this attribute is set to `yes`.

This is checked when you read in your SDC files with the `READ SDC` command.

The following commands are subject to this check:

- `create_clock`
- `create_generated_clock`
- `set_annotated_transition`
- `set_case_analysis`
- `set_clock_gating_check`
- `set_clock_latency`
- `set_clock_transition`
- `set_clock_uncertainty`
- `set_disable_timing`
- `set_drive`

© 1999-2023 All Rights Reserved.

- set_false_path
- set_fanout_load
- set_ideal_transition
- set_input_delay
- set_input_transition
- set_load
- set_max_delay
- set_max_fanout
- set_max_time_borrow
- set_max_transition
- set_min_delay
- set_multicycle_path
- set_output_delay
- set_propagated_clock
- set_scan_signal

Examples

- In the following pair of commands, the second set false path command duplicates the false path exception already defined on timing paths starting from CLK1 and CLK2. However, it can define new false paths if there are other clocks in addition to CLK1 and CLK2.

```
set_false_path -from [get_clocks {CLK1 CLK2}]
set_false_path -from CLK*
```

- In the following pair of commands, the second set multicycle path command overwrites the multi-cycle multiplier already defined on timing paths starting from CLK1 and CLK2. However, it can define new multi-cycle paths if there are other clocks in addition to CLK1 and CLK2.

```
set_multicycle_path 2 -from [get_clocks {CLK1 CLK2}]
set_multicycle_path 3 -hold -from CLK*
```


© 1999-2023 All Rights Reserved.

SDC_LINT_CMD7

Message

`Cannot overwrite existing command`

Default Severity

Error

Description

Indicates that an SDC command attempted to overwrite a previous command, but the tool does not support the overwrite.

This is checked when you read in your SDC files using the `READ SDC` command.

Examples

The following commands fail this check, because they would require deleting the clock defined by the first command. Such deletion is not supported in this version of the tool.

```
# attempt to define virtual clock CLK1:
create_clock -period 20 -name CLK1
# attempt to define generated clock CLK1:
create_generated_clock -source clk1 -divide_by 2 -name CLK1 [get_pins div2/Q]
# attempt to define real clock on same pin with different name, replacing CLK1:
create_clock -period 30 -name CLK2 [get_ports clk]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_CMD8

Message

`User defined unhandled command used`

Default Severity

Warning

Description

Indicates that a user-defined unhandled command is used in an SDC file.

To configure a command as unhandled, use the `configure_lint_check` command for that command. This command has a `usage` parameter that you can set to a value of `unhandled`.

Examples

Suppose you want to specify the `set_dont_touch` command as *unhandled* in your SDC files. To configure `set_dont_touch` as unhandled, use the following command in a lint configuration file:

```
configure_lint_check set_dont_touch { command { usage unhandled } }
```

After reading that configuration file, the following command causes a rule check violation because the `set_dont_touch` command is used:

```
set_dont_touch [get_nets n1]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_CMD9

Message

`Command replaced`

Default Severity

Error

Description

Indicates that a command has been invalidated by a different command that replaces it.

The SDC statement object of the replaced command has attribute is_overwritten == 1 and its overwritten_history attribute lists the replacing commands.

Examples

In the following pair of commands, the `set_clock_groups` command replaces the `set_false_path` exception already defined on timing paths from `CLK1` to `CLK2`.

```
set_false_path -from [get_clocks CLK1] -to [get_clocks CLK2]
set_clock_groups -asynchronous -name CLK1_CLK2 -group [get_clocks CLK1] -group
[get_clocks CLK2]
```

Note: The order of these two commands does not affect this check.

© 1999-2023 All Rights Reserved.

SDC_LINT_OPT*

This section describes the SDC rule checks that relate to option usage.

© 1999-2023
All Rights Reserved.

SDC_LINT_OPT1

Message

`Unknown option used`

Default Severity

Error

Description

Indicates that a command option that the software does not recognize is used in an SDC file.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes a rule check violation because the software does not recognize the option named `-periode`:

```
create_clock -name CLK3 -periode 10
```


© 1999-2023 All Rights Reserved.

SDC_LINT_OPT2

Message

`CCD unsupported option used`

Default Severity

Warning

Description

Indicates that an unsupported command option is used in an SDC file. The software recognizes this option, but ignores it.

To configure an option as unsupported, use the `configure_lint_check` command. This command has a `usage` parameter that you can set to a value of `unsupported`.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes a rule check violation because the software does not support the `-exact` option in the `get_nets` command:

```
get_nets -exact clka
```


© 1999-2023 All Rights Reserved.

SDC_LINT_OPT3

Message

`User defined forbidden option used`

Default Severity

Warning

Description

Indicates that a forbidden command option is used in an SDC file.

To configure an option as forbidden, use the `configure_lint_check` command for that option. This command has a parameter called `option`, which you can set to a value of `forbidden`.

Example

To forbid the using the `-hsc` option in the `get_cells` command, use the following command in a lint configuration file:

```
configure_lint_check get_cells { -hsc { usage forbidden } }
```

After reading that configuration file, the following command causes a rule check violation because the `-hsc` option is used:

```
get_cells -hsc @ mod1/a_reg[1]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_OPT4

Message

`Non-SDC option used`

Default Severity

Warning

Description

Indicates that you have used a command option that does not comply with the currently selected version of SDC. For a list of all the supported SDC commands and options, see:

```
<release>/lib/gui/tcl/sdccmd.<sdc version>.
```

To select an SDC version, set the `sdc_version` Tcl variable (see example). To check the latest supported version, use the following commands:

```
tclmode
puts $sdc::sdc_version
```

This is checked when you read in your SDC files with the `READ SDC` command.

To configure an option as non-SDC compliant, use the `configure_lint_check` command for that option. This command has a parameter called `is_sdc_compliant`, which you can set to a value of `no`.

Note: If a command is not SDC compliant, it can still be among the commands/options supported by the tool. For a list of the non-SDC commands that the Conformal software supports, refer to the Appendix called "Supported Non-SDC Commands" in the *Conformal Constraint Designer User Guide*.

Example

The following commands cause a rule check violation because the `-rise_from` option does not comply with SDC version 1.6:

```
set sdc_version 1.6
set_false_path -rise_from [get_clocks {CLK1 CLK2}]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_OPT5

Message

`Missing required option`

Default Severity

Error

Description

Indicates that a command option has been used without one of its required options.

To configure an option as required, use the `configure_lint_check` command for that option. This command has a parameter called `required_options`, where you can specify a list of required options.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes a rule check violation because using the `-clock_fall` option requires the presence of the `-clock` option:

```
set_input_delay 2 -clock_fall [get_ports IN1]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_OPT6

Message

`Missing mandatory option`

Default Severity

Error

Description

Indicates that a mandatory option of a command is not specified. This also applies to instances where one of several options is mandatory.

To configure an option as mandatory, use the `configure_lint_check` command for that option. This command has a parameter called `usage`, which you can set to `mandatory`.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes a rule check violation because using the `-clock_fall` option requires the presence of the `-clock` option:

```
set_input_delay 2 -clock_fall [get_ports IN1]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_OPT7

Message

`Missing recommended option`

Default Severity

Warning

Description

Indicates that a recommended option of a command is not specified.

To configure an option as recommended, use the `configure_lint_check` command for that option. This command has a parameter called `usage`, which you can set to a value of `recommended`.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

To recommend using the `-waveform` option in the `create_clock` command, use the following command in a lint configuration file:

```
configure_lint_check create_clock { -waveform { usage recommended } }
```

After reading that configuration file, the following command causes a rule check violation because it is missing the recommended `-waveform` option.

```
create_clock -name CLK -period 10
```


© 1999-2023 All Rights Reserved.

SDC_LINT_OPT8

Message

```
Using mutually exclusive options
```

Default Severity

Error

Description

Indicates that options that cannot appear together are used in the same command.

To configure options as mutually exclusive, use the `configure_lint_check` command. This command has a parameter called `mutex_options`, where you can specify a list of options that are mutually exclusive.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes a rule check violation because the `-edges` and `-duty` cycle cannot appear together in the same create generate clock command:

```
create_generated_clock -edges {1 2 3} -duty_cycle 50 -source \
[get_ports clk] [get_pins div_reg/Q]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_OPT9

Message

`Option appears more than once`

Default Severity

Error

Description

Indicates that a command option is used more than once in a command.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes a rule check violation because the `-period` option appears more than once:

```
create_clock -period 10 -period 8 -waveform {0 5} [get_ports clk]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_OPT10

Message

`User defined unhandled option used`

Default Severity

Warning

Description

Indicates that a unhandled command option is used in an SDC file.

To configure an option as unhandled, use the `configure_lint_check` command for that option. This command has a parameter called `usage`, which you can set to a value of `unhandled`.

Example

To set `-hsc` option in the `get_cells` command as unhandled, use the following command in a lint configuration file:

```
configure_lint_check get_cells { -hsc { usage unhandled } }
```

After reading that configuration file, the following command causes a rule check violation because the `-hsc` option is used:

```
get_cells -hsc @ mod1/a_reg[1]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_VAL*

This section describes the SDC rule checks that relate to values.

© 1999-2023 All Rights Reserved.

SDC_LINT_VAL1

Message

`Missing required value`

Default Severity

Error

Description

Indicates that a required value for a command option is not specified.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes a rule check violation because no value is specified for the `-period` option:

```
create_clock -name CLK -waveform {0 5} -period
```


© 1999-2023 All Rights Reserved.

SDC_LINT_VAL2

Message

`Invalid value type specified`

Default Severity

Error

Description

Indicates that a value of the wrong type is specified to a command option.

To configure an option's valid values, use the `configure_lint_check` command for that option. This command has a parameter called `valid_types`, where you can specify a list of valid types.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes a rule check violation because the value for the path multiplier must be an integer, not a float (`1.5`):

```
set_multicycle_path 1.5 -from A -to B
```


© 1999-2023 All Rights Reserved.

SDC_LINT_VAL3

Message

```
Invalid value specified
```

Default Severity

Error

Description

Indicates that the value specified for a command or option is invalid. This usually happens in two cases: the given value does not match one of the expected enumerated types, or the value does not match one of the patterns specified as valid values.

To configure the valid values for an option, use the `configure_lint_check` command for that option. This command has a parameter called `valid_values`, where you can specify a list of valid values for the option.

This is checked when you read in your SDC files with the `READ SDC` command.

Examples

- The following command causes a rule check violation because `bc_wk` is not a valid value for the `-analysis_type` option of the `set_operating_conditions` command:

  ```
  set_operating_conditions -analysis_type bc_wk
  ```

- The following command causes a rule check violation because `block` is not a valid mode name for the `set_wire_load_mode` command:

  ```
  set_wire_load_mode block
  ```


© 1999-2023 All Rights Reserved.

SDC_LINT_VAL4

Message

`Value is out of range`

Default Severity

Error

Description

Indicates that a value specified to a command option is out of range. That is, it does not satisfy the range expression specified in the SDC language or in the lint configuration.

To configure the valid range for an option value, use the `configure_lint_check` command for that option. This command has a parameter called `valid_range`, where you can specify valid range for the option.

A valid range for an option value can be configured with the `valid_range` option attribute.

Example

The following command causes a rule check violation because the value specified for minimum porosity is outside the valid range of 0-90:

```
set_min_porosity 90.001 [current_design]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_VAL5

Message

```
Value is an empty list
```

Default Severity

Error

Description

Indicates that a value is expected, but an empty list is found instead.

Example

The following command causes a rule check violation because the value specified for the period option is an empty list:

```
create_clock -name CLK20 -waveform {0 5} -period {}
```


© 1999-2023 All Rights Reserved.

SDC_LINT_REF*

This section describes the SDC rule checks that relate to references.

© 1999-2023 All Rights Reserved.

SDC_LINT_REF1

Message

```
Object cannot be found
```

Default Severity

Error

Description

Indicates that no object of the expected type matches the given name in a command. This can happen in get* commands, or in commands that use implicit references.

Note: For objects that are inside a blackbox, see rule SDC_LINT_REF2.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes a rule check violation if clock `CLK1` does not exist:

```
set_multicycle_path 3 -from [get_clocks CLK1]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_REF2

Message

```
The specified object name(s) is inside a black-box
```

Default Severity

Error

Description

Indicates that there is an occurrence where no object of the expected type was found because the given name leads into a blackbox. This can happen in get* commands, or in commands that use implicit references.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes a rule check violation if the `mod1` module is a blackbox:

```
get_cells mod1/a_reg[1]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_REF3

Message

`Logical pin name used`

Default Severity

Warning

Description

Indicates that a logical pin name is used in a command.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes a rule check violation if `testkey/test_clk` is a logical pin:

```
create_generated_clock -edges {1 2 3} \
   -source [get_ports clka] [get_pins testkey/test_clk]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_REF4

Message

`Invalid object specified`

Default Severity

Error

Description

Indicates that an object specified in a command argument has the wrong object type, is not a valid reference for the specific argument, or is an invalid object by itself (for example, a generated clock that has no master clock).

The type of an object referenced by a SDC command is one of the following:

- `clock`
- `cell`
- `design`
- `lib`
- `lib_cell`
- `lib_pin`
- `net`
- `pin`
- `port`

This is checked when you read in your SDC files with the `READ SDC` command.

To configure a list of valid object types for an option, use the `configure_lint_check` command for that option. This command has a parameter called `valid_object_types`, where you can specify a list of valid object types.

© 1999-2023 All Rights Reserved.

Example

The following command causes a rule check violation because the object list cannot be of type `clock`:

```
set_disable_timing [get_clocks CLK1]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_REF5

Message

`Implicit object specified`

Default Severity

Warning

Description

Indicates that a potentially ambiguous implicit object reference is used: the appropriate `get *` command has not been used, and there is more than one allowed object type for this argument.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes a rule check violation because the `-to` option allows a `clock`, `cell`, `port`, or `pin` argument:

```
set_false_path -to CLK1
```


© 1999-2023 All Rights Reserved.

SDC_LINT_REF6

Message

`Reference is an empty list`

Default Severity

Error

Description

Indicates that an object list specified to a command option is empty. This usually happens when a nested get* command fails. This relates to rule SDC_LINT_REF1.

This is checked when you read in your SDC files with the `READ SDC` command.

Example

The following command causes rule check violation if creation fails for clock `CLK1`:

```
set_false_path -from [get_clocks CLK1]
```


© 1999-2023 All Rights Reserved.

SDC_LINT_REF7

Message

`Not recommended object type specified`

Default Severity

Warning

Description

Indicates that an object that was used in the specified command is not recommended.

To configure a list of object types that are not recommended for an option, use the `configure_lint_check` command for that option. This command has a parameter called `flag_on_object_types`, where you can specify a list of object types that are not recommended.

© 1999-2023 All Rights Reserved.

SDC_LINT_REF8

Message

```
Incorrect clocks propagate to reference object
```

Default Severity

Error

Description

Indicates that a statement has failed (and no SDC object was created for it), due to how clocks propagate or do not propagate, to one or more objects referenced in the command.

Note: This rule is checked during `set system mode verify`.

Example

These are the cases reported in this rule:

- For a generated clock with `-master_clock`, the specified master clock does not propagate to the `-source` pin.
- For a generated clock, the only clock propagating to the `-source` pin is invalid (has `SDC_LINT_REF8` reported itself).
- For a generated clock, the clock that propagates to the `-source` pin is inactive.
- For a generated clock without `-master_clock`, multiple clocks propagate to the `-source pin`, so a master clock cannot be determined.
- A generated clock has no root master clock (non-generated ancestor). In other words, following the master clock chain, a circularity is found. One clock in that circularity is shown in the rule occurrence.
- For a `set_input_delay` or `set_output_delay` with `-reference_pin` and `-clock`, the specified pin is not in the network of the specified clock.
- For a `set_input_delay` or `set_output_delay` with `-reference_pin` and without `-clock`, the specified pin is not in the network of a clock.

© 1999-2023 All Rights Reserved.

- For a `set_clock_sense` with `-clock`, a specified clock does not propagate to a reference object.
- For a `set_clock_sense` without `-clock`, no clock propagates to a reference object.

© 1999-2023 All Rights Reserved.

SDC_LINT_REF9

Message

`Multiple master clocks for generated clock`

Default Severity

Warning

Description

Indicates that, for a generated clock without `-master_clock`, multiple clocks propagate to the `-source` pin. A master clock will be determined as follows:

- If only one active clock propagates to the `-source` pin, the active clock is the master clock.
- If more than one active clocks propagate to the `-source` pin, the master clock is the first one in the list sorted by clock name.

Example

The following commands cause only one active clock to propagate to the `-source` pin of generated clock, so the active clock clk1 is used as the master clock

```
create_clock -name clk0 -period 2 d -add
create_clock -name clk1 -period 2 d -add
set_active_clocks {clk1}
create_generated_clock -name gclk -source d -divide_by 1 q
```

The following commands cause more than one active clock to propagate to the `-source` pin of generated clock, so clk1 is used as the master clock:

```
create_clock -name clka -period 2 d -add
create_clock -name clk1 -period 2 d -add
create_generated_clock -name gclk -source d -divide_by 1 q
```


© 1999-2023 All Rights Reserved.

© 1999-2023 All Rights Reserved.

5

Clock Domain Crossing Rule Checks

This section describes the clock domain crossing rule set (`cdc_def_rs`), which is a default rule set included with Conformal Constraint Designer.

For information on CDC checks, refer to the "Clock Domain Crossing Flow" in the *Conformal Constraint Designer User Guid*e.

© 1999-2023 All Rights Reserved.

CDC Rule Check Quick Reference

Table 5-1 Quick Tasks

Task	Command Example
Add CDC rule set	`add rule set -file ccd_default_cdc_ruleset.tcl`
Report all the CDC rule groups	`report rule group cdc_def_rs/*`
Report CDC rule source	`report rule source cdc_*` `report rule source cdc_datactrl_sync_rule`
Report rule instance	`report rule instance cdc_def_rs/cdc_checks/clocked->unclocked` `report rule instance cdc_def_rs/sr_sync_crossing_checks/ sr_sync_clkA->clkA`

Table 5-2 CDC Rule Set (cdc_def_rs) Quick Reference

Rule Group	Rule Instance Base Name	Rule Source
`cdc_checks`	`cdc_<clk_name>->cdc_<clk_name>`	cdc_datactrl_sync_rule
	`clocked->unclocked`	
	`unclocked -> clocked`	
	`unclocked -> unclocked`	
`conv_checks`	`convergence_in_<clk_name>`	cdc_conv_check_rule
	`convergence_in_clkA`	
	`convergence_in_clkB`	
	`convergence_in_u_plldiv_by_2`	
`set_rst_checks`	`set_reset_for_<clk_name>`	cdc_setreset_sync_rule
	`set_reset_for_clkA`	
	`set_reset_for_clkB`	
	`set_reset_for_plldiv_by_2`	
	`set_reset_for_unclocked`	
`sr_sync_crossing_check`	`sr_sync_<clk_name>-><clk_name>`	cdc_sr_sync_crossing_rule

© 1999-2023
All Rights Reserved.

	`sr_sync_clkA->clkA`
	`sr_sync_clkB-> clkB`
	`sr_sync_u_plldiv_by_2-> u_plldiv_by_2`

Note: The Rule Instance names provided are base names, the full path of the rule instance would contain its rule set and rule group. For example, here are some full rule instance names:

```
cdc_def_rs/sr_sync_crossing_checks/sr_sync_u_plldiv_by_2-> u_plldiv_by_2
cdc_def_rs/set_rst_checks/set_reset_for_unclocked
cdc_def_rs/cdc_checks/unclocked -> unclocked
```


© 1999-2023 All Rights Reserved.

cdc_datactrl_sync_rule

```
cdc_datactrl_sync_rule
     analysis_mode <structural | functional>
     source_clock <clock_object>
     destination_clock <clock_object>
     from <start_object>
     to <end_object>
     modules <list_of_modules>
     consider_clock_group <yes | no>
     consider_clock_phase <yes | no>
     expand_same_clock_group <no | yes>
     sync_chain_logic {dff <wire | buffer | logic> mux <wire | buffer | logic>}
     sync_chain_fanout {dff <single | multiple> mux <single | multiple>}
     cdc_path_logic {dff <wire  | buffer | logic> mux <wire | buffer | logic>
                     user <wire | buffer | logic>}
     cdc_path_fanout {dff <single | multiple> mux <single | multiple>
                     user <single | multiple>}
     allow_sync_scheme <all | dff | mux | user>
     dff_sync_scheme {min <2> max <infinite> first <dff | latch>}
     mux_sync_scheme {min 0 max 0 first <dff | latch>}
     monitor_cycles <1 | 2>
     exclude_modules <list_of_modules>
     filter_paths {from <object> to <object>} ...
     exclude_atomic_checks <list_of_atomic_checks>
     suppress_options_in_header <list_of_options>
```

Structurally and functionally verifies that each crossing between the source and destination clocks are correctly synchronized. If the synchronization scheme does not adhere to the specified criteria, the tool will report a violation of this rule check for the crossing.

Rule Attributes

Criteria for the synchronization is specified through attributes. To view the attributes from the tool, use the following command:

```
report rule source "cdc_datactrl_sync_rule" -verbose
```

The following details the supported attributes for this check. Some of these attributes are proved by atomic checks (described in "Atomic Checks" on page 533).

analysis_mode

Description	Specifies whether to perform structural or functional checks.
Possible Value	`<structural | functional>`

© 1999-2023 All Rights Reserved.

Example	`set_attribute $rule_instance analysis_mode functional`

source_clock

Description	Specifies the source clock. Must be an SDC object of a clock.
Possible Value	*`clock_object`*

Example	You can also use the keywords `clocked` and `unclocked` to specify a source clock. However, you cannot combine these keywords with SDC objects (for example, you cannot designate a source clock as `clocked`/`unclocked`, and then designate an SDC object for the destination clock); both must be SDC objects or both must be keywords.
	For example:
	`set_attribute $rule_instance source_clock \`
	`  [find -sdcobj <clk1>] \`
	`set_attribute $rule_instance source_clock clocked`
	`set_attribute $rule_instance source_clock \`
	`  [find -sdcobj <clk1>]`

destination_clock

Description	Specifies destination clock, must be an SDC object of a clock.
Possible Value	*`clock_object`*

Example	`set_attribute $rule_instance destination_clock \`
	`  [find -sdcobj <clk2>]`
	You can also use the keywords `clocked` and `unclocked` to specify a destination clock. However, you cannot combine these keywords with SDC objects (for example, you cannot designate a source clock as `clocked`/`unclocked`, and then designate an SDC object for the destination clock); both must be SDC objects or both must be keywords.
	For example:
	`set_attribute $rule_instance destination_clock \`
	`  [find -sdcobj <clk2>]`
	`set_attribute $rule_instance destination_clock unclocked`

© 1999-2023 All Rights Reserved.

from

Description	Specifies the start point of a crossing; must be a design object. (optional)
Possible Value	`start_object`
Example	`set_attribute $rule_instance from [find -instance \ <top/sub/abc_reg>]`

Refer to Figure 5-11 on page 106 and Figure 5-12 on page 107 for an example.

to

Description	Specifies the end point of a crossing; must be a design object. (optional)
Possible Value	`end_object`
Example	`set_attribute $rule_instance to \ [find -instance <top/sub/xyz_reg>]`

modules

Description	Constrains the rule instance of a CDC structural check such that it is applied to the CDC paths of the specified modules.
Possible Value	`<list_of_modules>`
Example	`set_attribute $rule_instance module \ [ find -design modue_b ]`

© 1999-2023 All Rights Reserved.

consider_clock_group

Description	**Please note that this option is no longer supported because clock groups are not considered for synchronizer detection.** Specifies whether to take `clock_groups` into account when considering a CDC. Note: Source and destination clocks are treated as asynchronous even if they are in the same clock group.
Possible Value	<yes | no> Default is yes.
Example	`set_attribute $rule_instance consider_clock_groups no`

consider_clock_phase

Description	Specifies whether to take `clock_phase` into account when considering a CDC. Yes is the default. Note: Source and destination clocks are treated as asynchronous even if they are in the same clock group.
Possible Value	<yes | no> Default is yes.
Example	`set_attribute $rule_instance consider_clock_phase no`

© 1999-2023
All Rights Reserved.

Figure 5-1 consider_clock_phase==yes

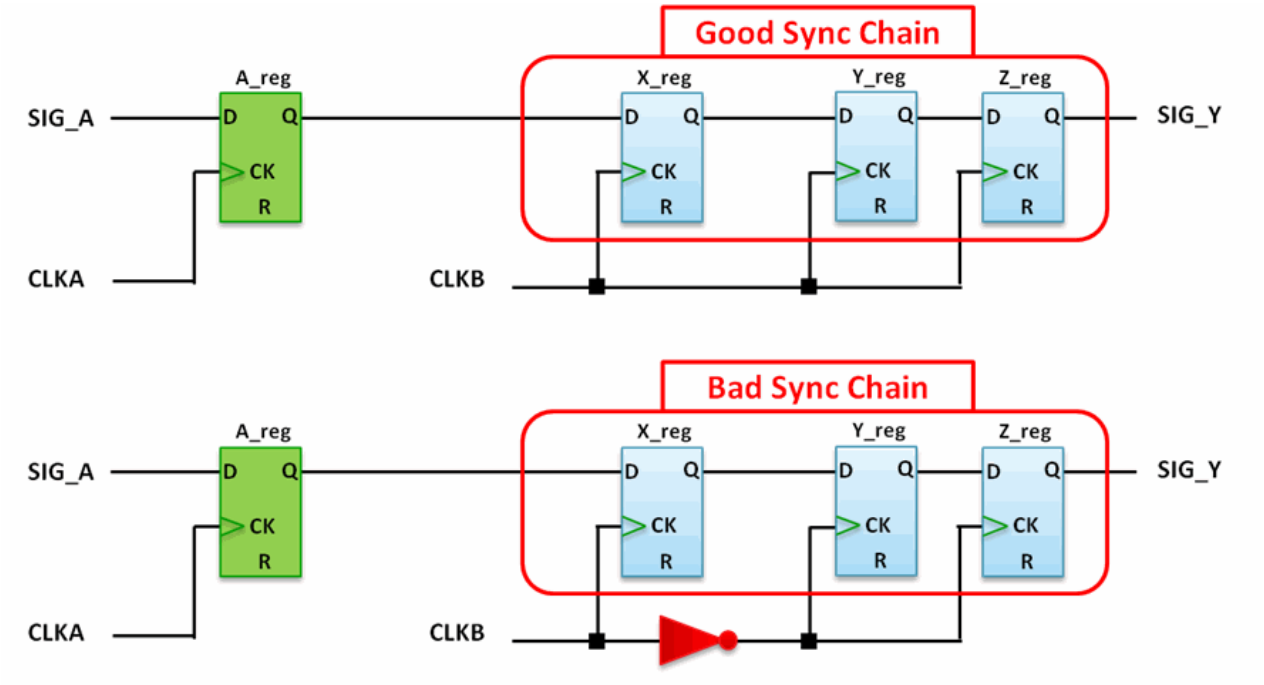
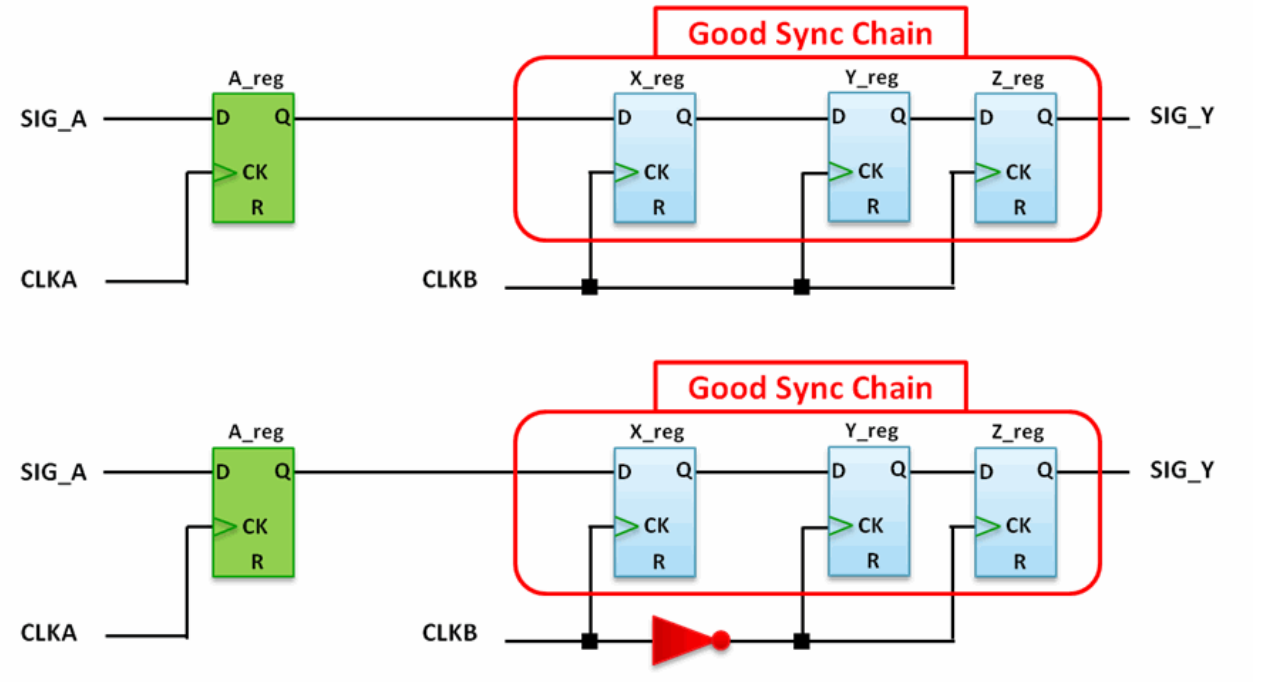


Figure 5-2 consider_clock_phase==no



© 1999-2023
All Rights Reserved.

expand_same_clock_group

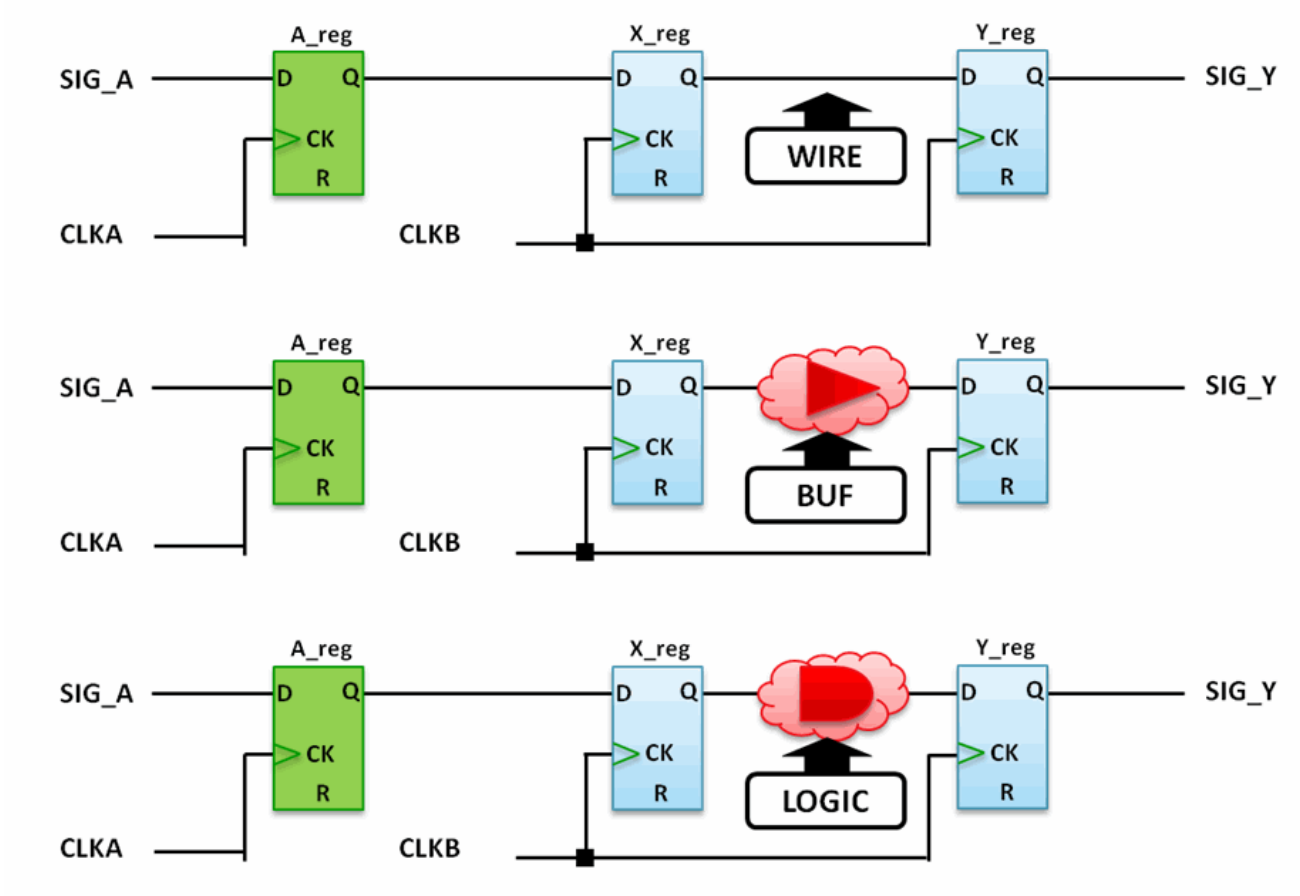
Description	Specifies whether to expand the paths between the source and destination clocks within the same clock group.
Possible Value	<no | yes> Default is `no`.
Example	`set_attribute $rule_instance expand_same_clock_group yes`

sync_chain_logic

Description	Specifies the sync chain logic.
Possible Value	`{dff <wire | buffer | logic> mux <wire | buffer | logic>}` Default value is `dff wire mux wire`. `dff` — For DFF synchronization schemes `mux` — For MUX synchronization schemes `wire` — No logic is allowed `buffer` — Allows only buffers/inverters inside the synchronization chain `logic` — Allows any logic inside the synchronization chain
Atomic Checks	`cdc_ctrl_logic_type_check` checks logic type in the data path (for DFF crossings). `cdc_data_logic_type_check` checks logic type in the data path (for MUX crossings).
Example	`set_attribute $rule_instance sync_chain_logic dff wire` `set_attribute $rule_instance sync_chain_logic mux buffer`

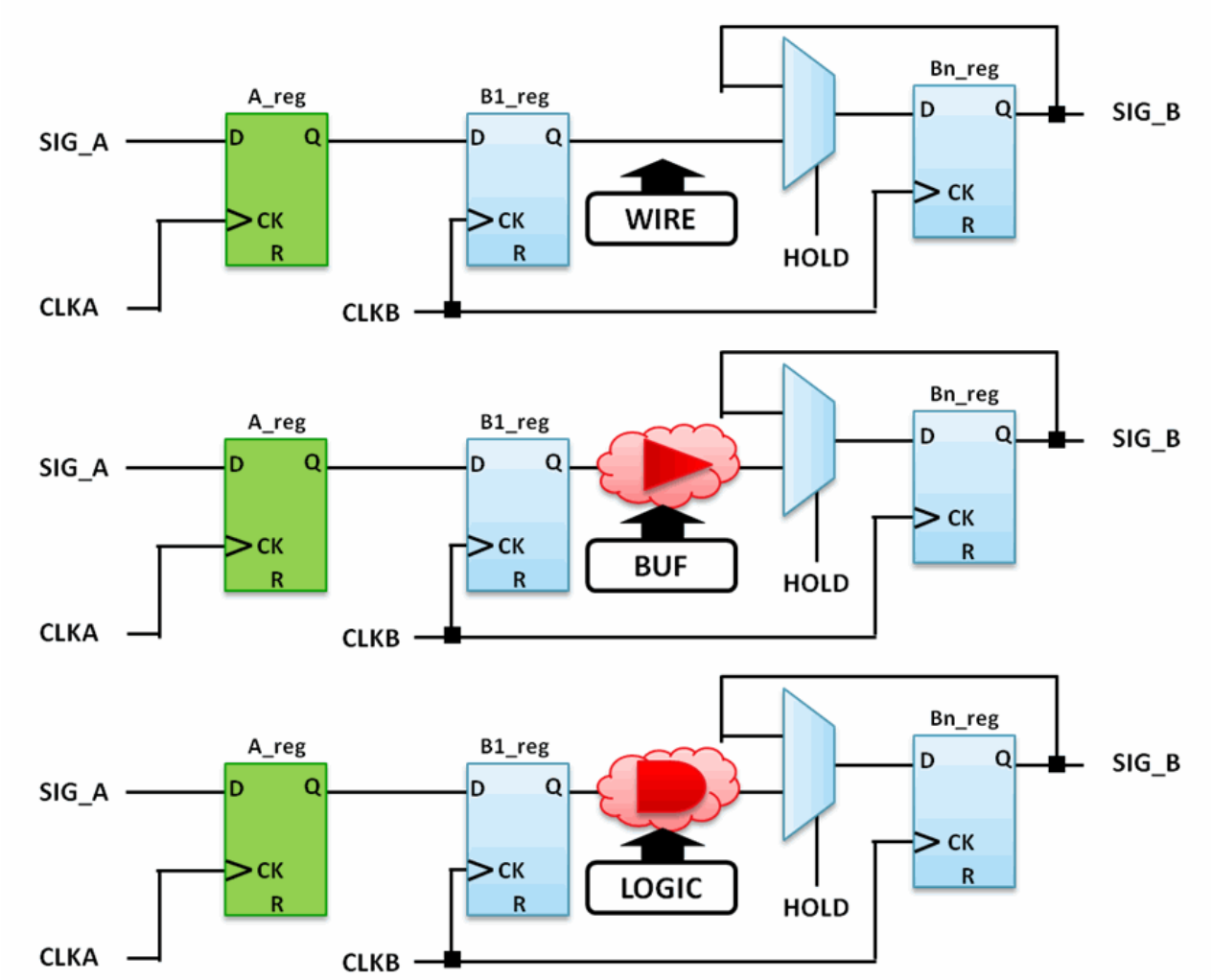
© 1999-2023 All Rights Reserved.

Figure 5-3 sync_chain_logic (for D Flip Flop)



© 1999-2023
All Rights Reserved.

Figure 5-4 sync_chain_logic (for MUX)

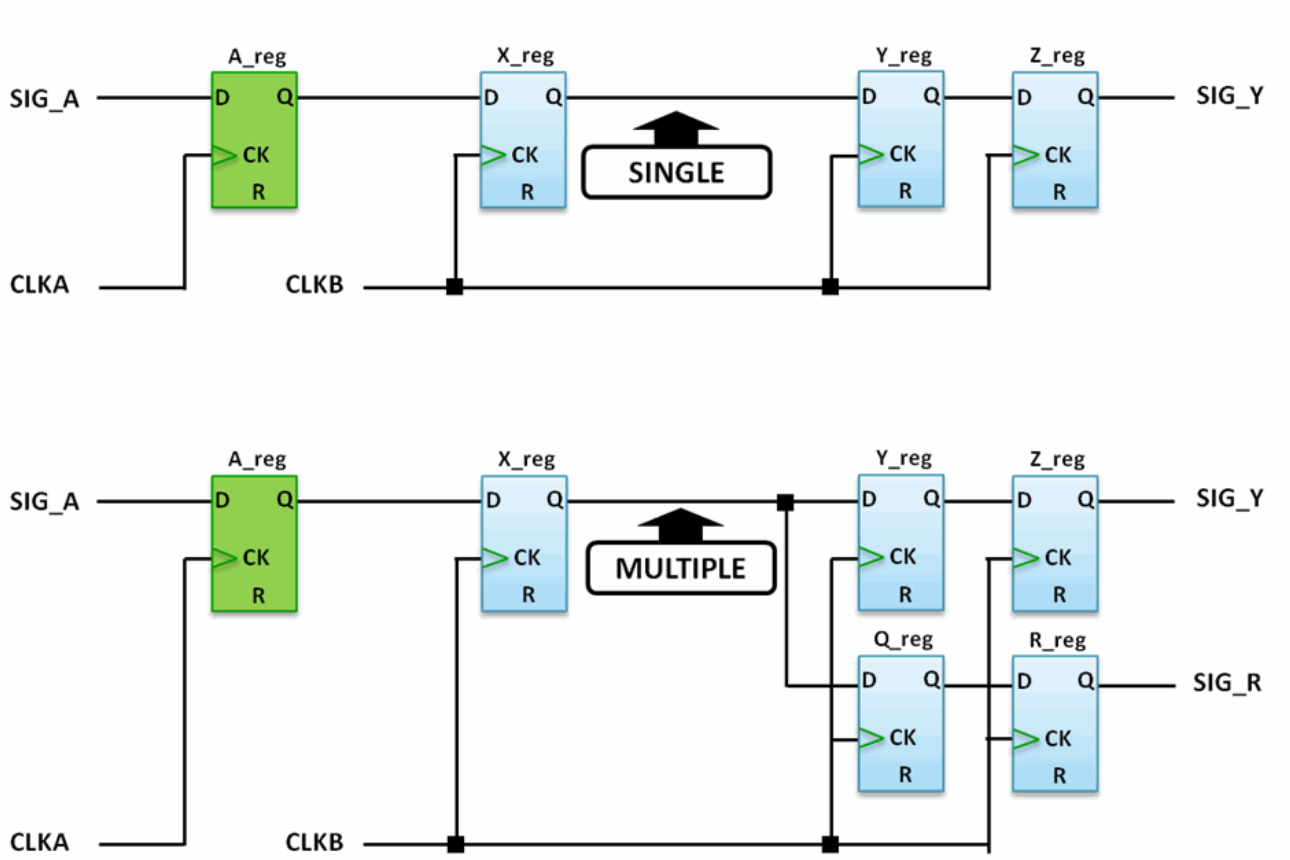


sync_chain_fanout

Description	Specifies whether fanouts are allowed in the sync chain.	
Possible Value	`{dff <single | multiple> mux <single | multiple>}`	
	`dff`	For DFF synchronization schemes
	`mux`	For MUX synchronization schemes

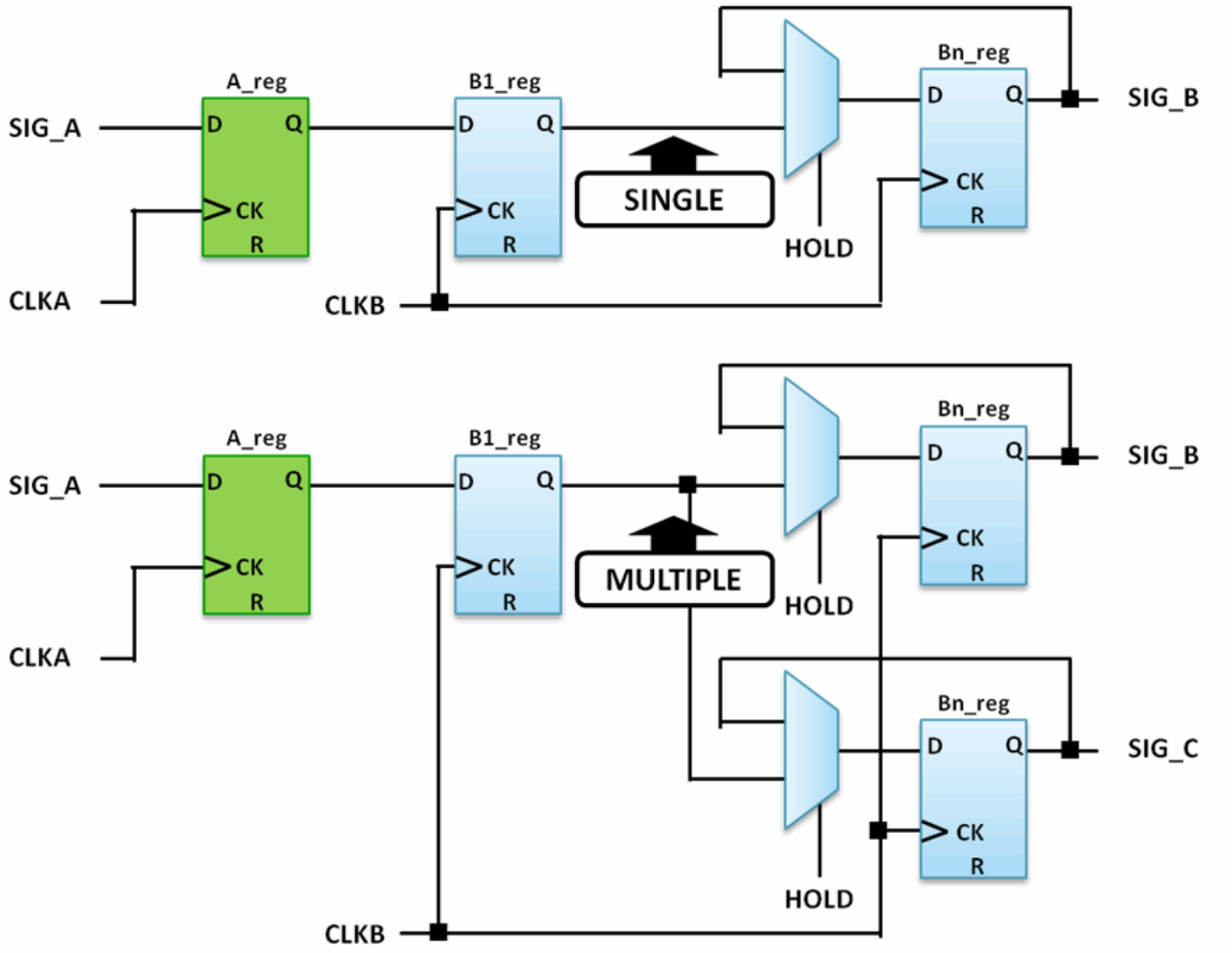
	`single`	Fanouts are not allowed
	`multiple`	Fanouts are allowed
Atomic Checks	`cdc_ctrl_multi_chain_check` checks if sync-chain fanouts to multiple destinations (for DFF).	
	`cdc_data_multi_chain_check` checks if sync-chain fanouts to multiple destinations (for MUX).	
Example	`set_attribute $rule_instance sync_chain_fanout \ [list dff multiple mux multiple]`	

Figure 5-5 sync_chain_fanout (for D Flip Flop)



© 1999-2023
All Rights Reserved.

Figure 5-6 sync_chain_fanout (for MUX)



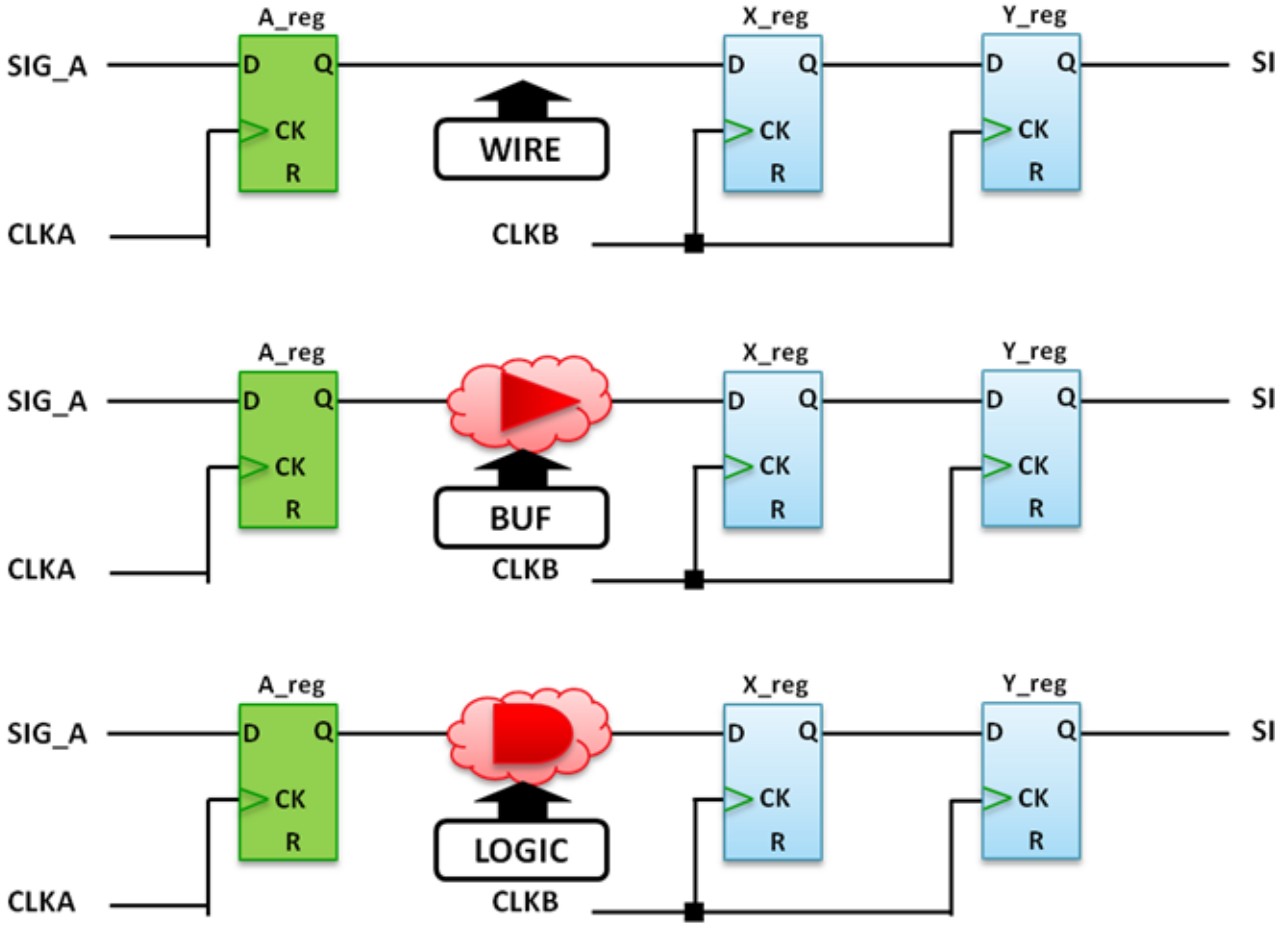
cdc_path_logic

Description	Specifies the logic type in the CDC path. Default value is `dff wire mux wire user wire`
Possible Value	`{dff <wire | buffer | logic> mux <wire | buffer | logic> user <wire | buffer | logic>}`
	`dff` For DFF synchronization schemes
	`mux` For MUX synchronization schemes

© 1999-2023
All Rights Reserved.

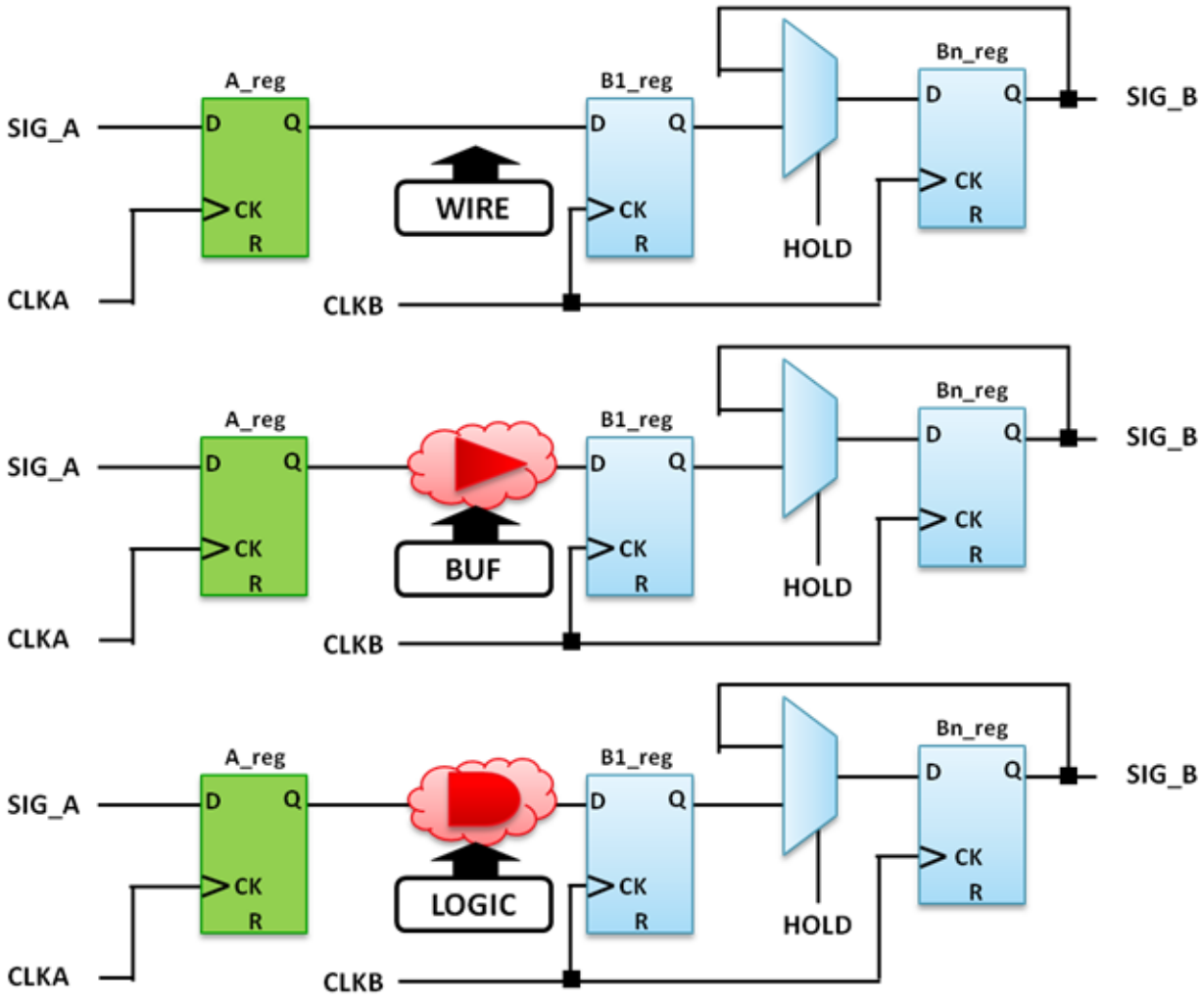
	`user`	For user synchronization scheme
	`wire`	No logic is allowed
	`buffer`	Allows only buffers/inverters inside the synchronization chain
	`logic`	Allows any logic inside the synchronization chain
Atomic Checks	`cdc_path_logic_type_check` checks logic type in the CDC path.	
Example	`set_attribute $rule_instance cdc_path_logic \ [list dff wire mux wire user logic]`	

Figure 5-7 cdc_path_logic for DFF



© 1999-2023
All Rights Reserved.

Figure 5-8 cdc_path_logic for MUX



cdc_path_fanout

Description	Specifies whether a CDC path can fanout to multiple destinations.
Possible Value	`{dff <single | multiple> mux <single | multiple> user <single | multiple>}`
	`dff` For DFF synchronization schemes

© 1999-2023
All Rights Reserved.

	`mux`	For MUX synchronization schemes
	`user`	For user synchronization schemes
	`single`	Fanouts are not allowed
	`multiple`	Fanouts are allowed
Atomic Checks	`cdc_path_destination_check` checks for multiple destinations (fanouts) in the CDC path.	
Example	`set_attribute $rule_instance cdc_path_fanout \ [list dff multiple mux multiple user single]`	

allow_sync_scheme

Description	Specifies the allowed types of sync schemes for the CDC path being validated:	
Possible Value	`<all | dff | mux | user>`	
	`all`	Allows all clock schemes
	`dff`	Only DFF type of sync scheme is allowed
	`mux`	Only MUX type of sync scheme is allowed
	`user`	Only user specified sync module is allowed

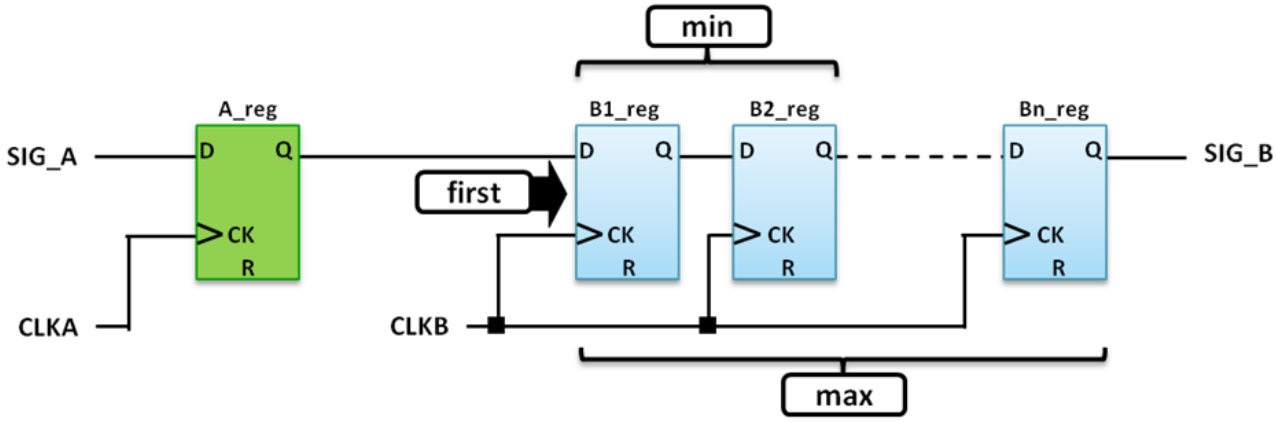
dff_sync_scheme

Description	Specifies the minimum/maximum number of DFFs allowed in the sync chain and whether the first destination should be a latch or a DFF. Default value is `min 2 max infinite first dff`.	
Possible Value	`{min <2> max <infinite> first <dff | latch>}`	
	`min`	Specifies the minimum number of DFFs (must be 2 or more)
	`max`	Specifies the maximum number of DFFs (must be greater than or equal to the minimum number)

© 1999-2023 All Rights Reserved.

	`first`	Specifies whether the first destination is a latch or DFF
Atomic Checks	`cdc_ctrl_min_sync_chain_check` checks if the minimum number of flops is met.	
	`cdc_data_max_sync_chain_check` checks if the maximum number of flops is met	
	`cdc_ctrl_first_dlatch_check` Checks if the first sequential element in the destination is a latch.	
Example	`set_attribute $rule_instance dff_sync_scheme \ [list min 2 max 3 first latch]`	

Figure 5-9 dff_sync_scheme



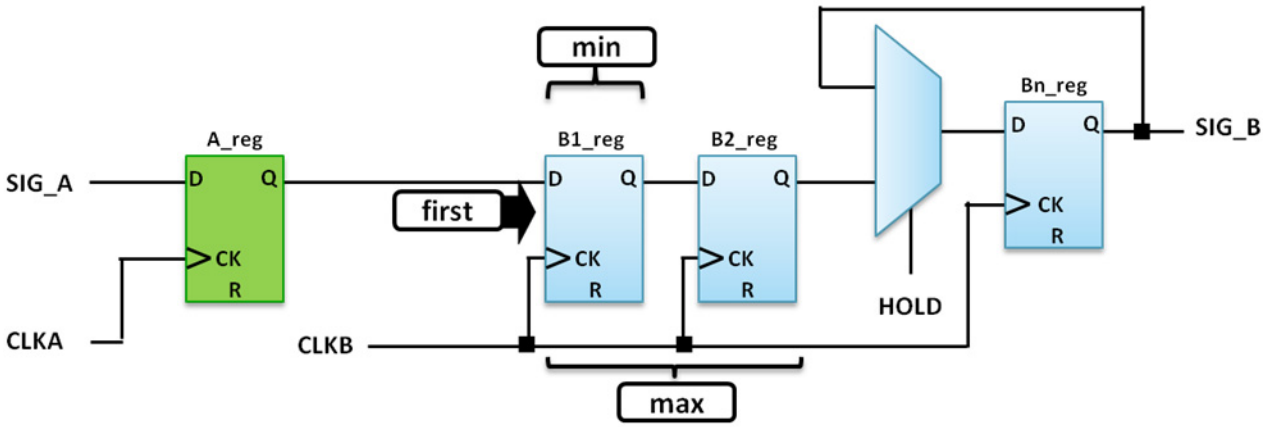
mux_sync_scheme

Description	For MUX synchronizers, specifies the minimum/maximum number of flops and the logic type of the first destination.	
Possible Value	`{min 0 max 0 first <dff | latch>}`	
	Default value is `min 0 max 0 first dff`.	
	`min`	For MUX synchronizers, specifies the minimum number of flops (must be 0 or more)

All Rights Reserved.

	`max`	For MUX synchronizers, specifies the maximum number of flops (must be greater than or equal to the minimum number)
	`first`	Specifies whether the first destination is a latch or DFF
Atomic Checks	`cdc_data_first_dlatch_check` checks if the first sequential element in the destination is a latch or a DFF. `cdc_data_max_sync_chain_check` checks if the maximum number of flops is met. `cdc_data_min_sync_chain_check` checks if the minimum number of flops has been met.	
Example	`set_attribute $rule_instance mux_sync_scheme \` `  [list min 1 max 3 first latch]`	

Figure 5-10 mux_sync_scheme



© 1999-2023
All Rights Reserved.

monitor_cycles

Description	Specifies the number of clock cycles used to monitor the signals for CDC functional checking purposes. When the source and destination clocks have the same clock periods, then one monitor cycle equals the clock period multiplied by 1. When the clock periods are different, then one monitor cycle equals the least common multiplier (LCM) multiplied by one. For example, if `monitor_cycles` changes to 2, then it would be the clock period multiplied by 2 or the LCM multiplied by 2.
Possible Value	`<1 | 2>` Default value is `1` clock cycle.
Example	`set_attribute $rule_instance monitor_cycles 2`

exclude_modules

Description	Constrains the rule instance of a CDC structural check such that it is not applied to the CDC paths of the specified modules.
Possible Value	`<list_of_modules>`
Example	`set_attribute $rule_instance exclude_module \` `[ find -design mod_a ]`

filter_paths

Description	A list of path specifications to filter out.
Possible Value	`{from <object> to <object>} ...`

© 1999-2023
All Rights Reserved.

Example

For a single destination:

```
set_attribute $rule_instance filter_paths [list \
        from [find -instance A_reg] \
        to [find -instance B1_reg]]
```

This would filter `A_reg->B1_reg`.

For multiple destinations, use the following format. If you use the `set_attribute filter_paths` command more than once, the last command overrides all previous commands.

For multiple destinations:

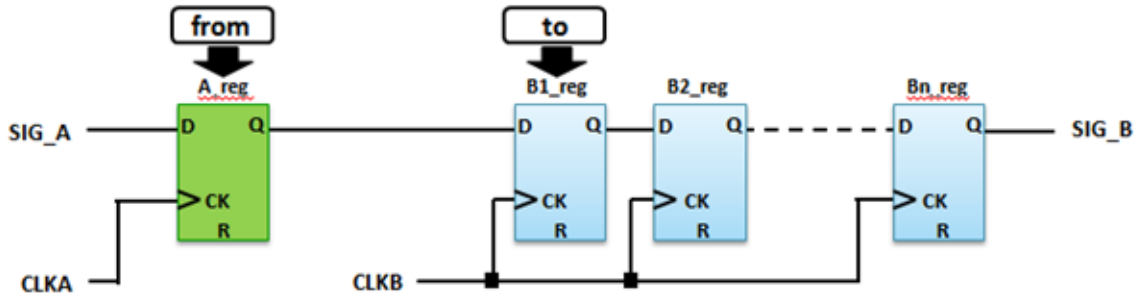
```
set_attribute $rule_instance filter_paths \
      [list from [find -instance A_reg] to \
      [find -instance {B1_reg C1_reg}] ]
```

This would filter `A_reg->B1_reg` and `A_reg->C1_reg`.

```
set_attribute $rule_instance filter_paths [list \
       [ list from [find -instance {a_reg b_reg} ]
       to [ find -instance { c_reg d_reg }]] \
       [ list from [ find -instance f_reg ] to \
       [ find -instance g_reg ]] ]
```

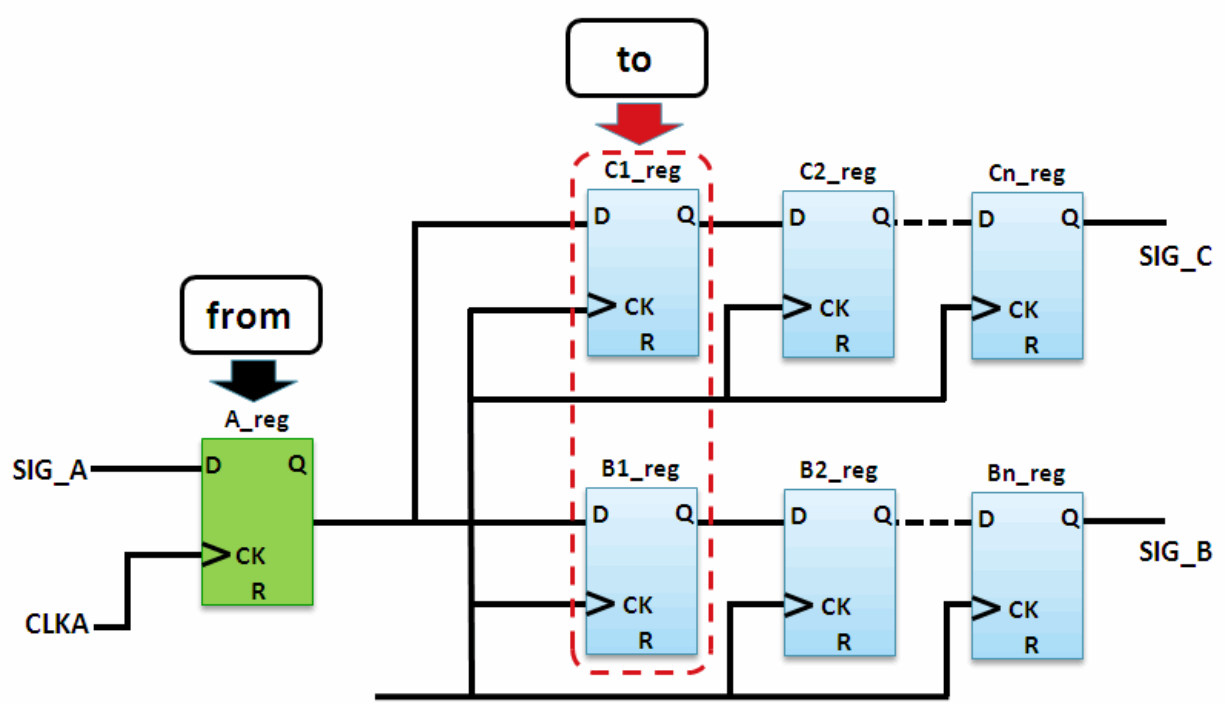
This would filter `a_reg->c_reg`, `a_reg->d_reg`, `b_reg->c_reg`, `b_reg->d_reg` paths as well as `f_reg->g_reg`.

Figure 5-11 filter_paths for datactrl



© 1999-2023
All Rights Reserved.

Figure 5-12 filter_paths for datactrl (multiple destinations)



exclude_atomic_checks

Description	Excludes/disables atomic checks for the specified rule instance.
Possible Value	`<list_of_atomic_checks>`

Example	`set_attribute $rule_instance exclude_atomic_checks \`
	`  [list cdc_path_logic_type_check \`
	`  cdc_path_destination_check ]`

suppress_options_in_header

Description	Excludes the specified rule options from the `REPORT RULE CHECK` header.
Possible Value	`<list_of_instances_to_exclude>`

All Rights Reserved.

Example	For example, to exclude the paths specified by the `filter_paths` attribute from the `REPORT RULE CHECK` header, use the `set_attribute` command on each rule instance: `set_attribute $rule_instance suppress_options_in_header \ [ list filter_paths]`

© 1999-2023 All Rights Reserved.

cdc_conv_check_rule

```
cdc_conv_check_rule
     analysis_mode <both | structural | functional>
     destination_clock <clock_object>
     from <start_object>
     to <end_object>
     check_conv_type <same_clock_groups_conv | diff_clock_groups_conv | all>
     check_vector_conv <all | invector | outvector>
     filter_paths {from <convergence_source_list> end_point
     <convergence_end_point>} ...
     suppress_options_in_header <list_of_options>
```

Structurally and functionally verifies that the CDCs are not converging at the destination domain. The convergence type is specified through the rule attributes listed below. If the convergence does not adhere to the criteria specified, the tool will report a violation of this rule for each convergence point.

Rule Attributes

Criteria for the rule is specified through rule attributes. To view the rule attributes from the tool, use the following command:

```
report rule source "cdc_conv_check_rule" -verbose
```

The following table details the rule attributes for this check. Some of these rule attributes are proved by the atomic checks described in (described in "Atomic Checks" on page 533).

analysis_mode

Description	Specifies whether to perform structural and/or functional checks.
Possible Value	`<both | structural | functional>`
Example	`set_attribute $rule_instance analysis_mode functional`

destination_clock

Description	Specifies destination clock, must be an SDC object of a clock.
Possible Value	*`clock_object`*

© 1999-2023 All Rights Reserved.

Example	`set_attribute $rule_instance destination_clock \ [find -sdcobj <clk2>]`
	
	You can also use the keywords `clocked` and `unclocked` to specify a destination clock. However, you cannot combine these keywords with SDC objects (for example, you cannot designate a source clock as `clocked`/`unclocked`, and then designate an SDC object for the destination clock); both must be SDC objects or both must be keywords.
	
	For example:
	
	`set_attribute $rule_instance destination_clock \ [find -sdcobj <clk2>]`
	`set_attribute $rule_instance destination_clock unclocked`

from

Description	Specifies the start point of a crossing; must be a design object. (optional)
Possible Value	`start_object`
Example	`set_attribute $rule_instance from [find -instance \ <top/sub/abc_reg>]`

to

Description	Specifies the end point of a crossing; must be a design object. (optional)
Possible Value	`end_object`
Example	`set_attribute $rule_instance to \ [find -instance <top/sub/xyz_reg>]`

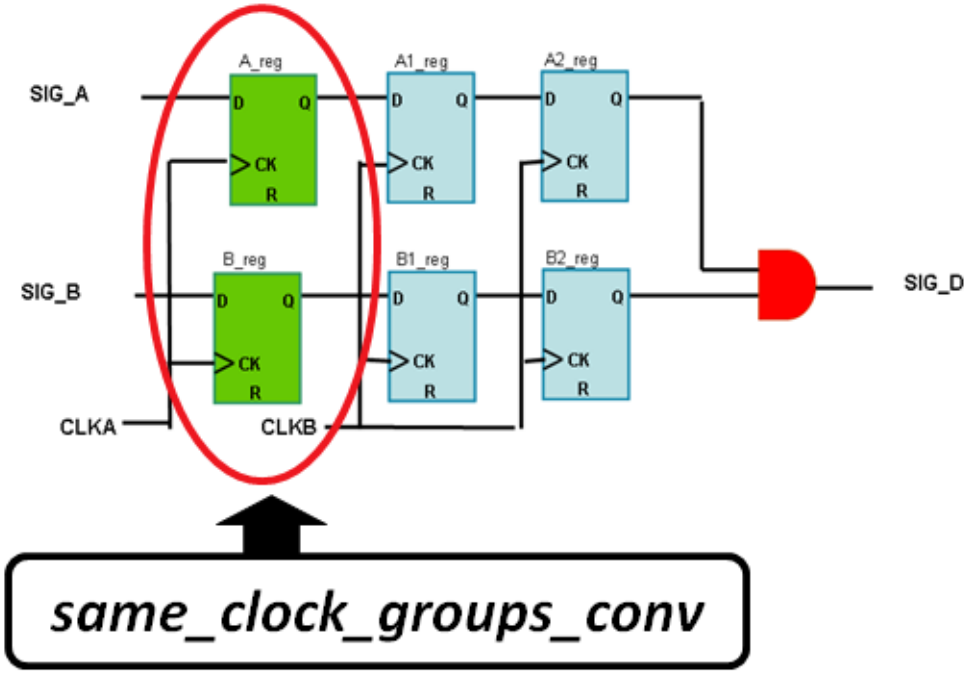
check_conv_type

Description	Specifies whether the convergence is from the same source domain or from difference source domains. Default is `same_clock_groups_conv`.

© 1999-2023 All Rights Reserved.

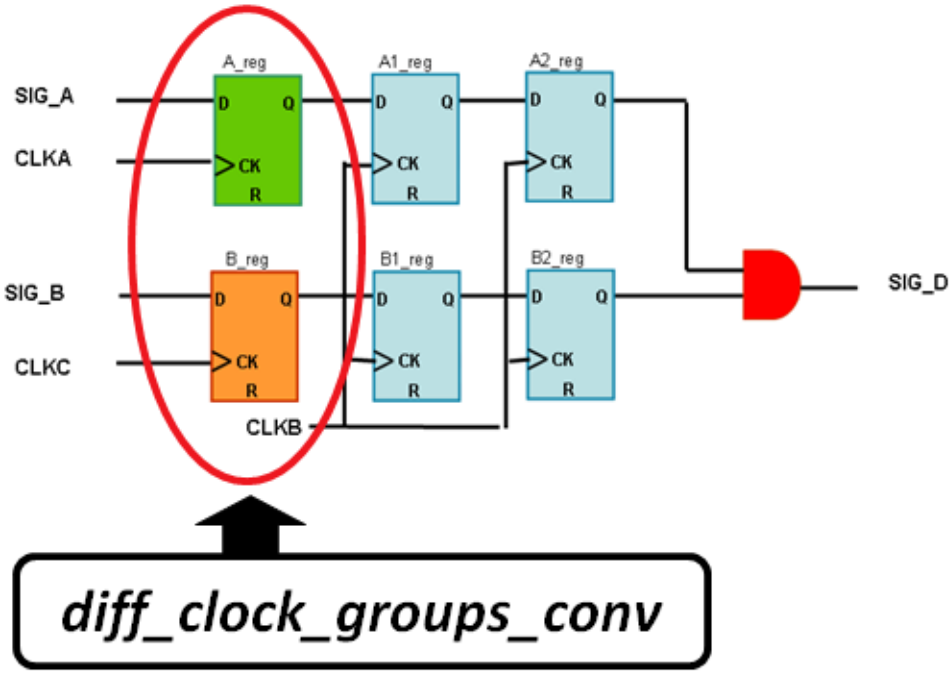
Possible Value	`<same_clock_groups_conv | diff_clock_groups_conv | all>`	
	`same_clock_groups_conv`	Convergence between signals is from same source clock
	`diff_clock_groups_conv`	Convergence between the signals is from different source clocks
	`all`	Convergence can be from the same source clock, or from different source clocks.
Atomic Checks	`cdc_conv_same_domain_check` checks for convergence from the same source domain.	
	`cdc_conv_diff_domain_check` checks for convergence from different source domains.	
Example	`set_attribute $rule_instance check_conv_type \ same_clock_domain`	

Figure 5-13 check_conv_type (from same source clock)



© 1999-2023
All Rights Reserved.

Figure 5-14 check_conv_type (from different source clock)



check_vector_conv

Description	Specifies whether the convergence is from the same vectors (`invector`) or from different vectors (`outvector`).	
Possible Value	<all | invector | outvector>	
	Default is `all`.	
	`all`	Check both invector and outvector convergence
	`invector`	Check for *invector convergence*, where convergence occurs amongst `from_instance` of CDC paths that belong to the same group of multibit registers (such as arrays).
	`outvector`	Check for *outvector convergence*, where the convergence occurs amongst `from_instance` of CDC paths that belong to different groups of single-bit or multi-bit registers.

© 1999-2023
All Rights Reserved.

Atomic Checks	`cdc_conv_in_vector_check` checks whether convergence is from the same vector. `cdc_conv_out_vector_check` checks whether convergence is from different vectors.
Example	`set_attribute $rule_instance check_vector_conv invector`

Figure 5-15 check_vector_conv (invector convergence)

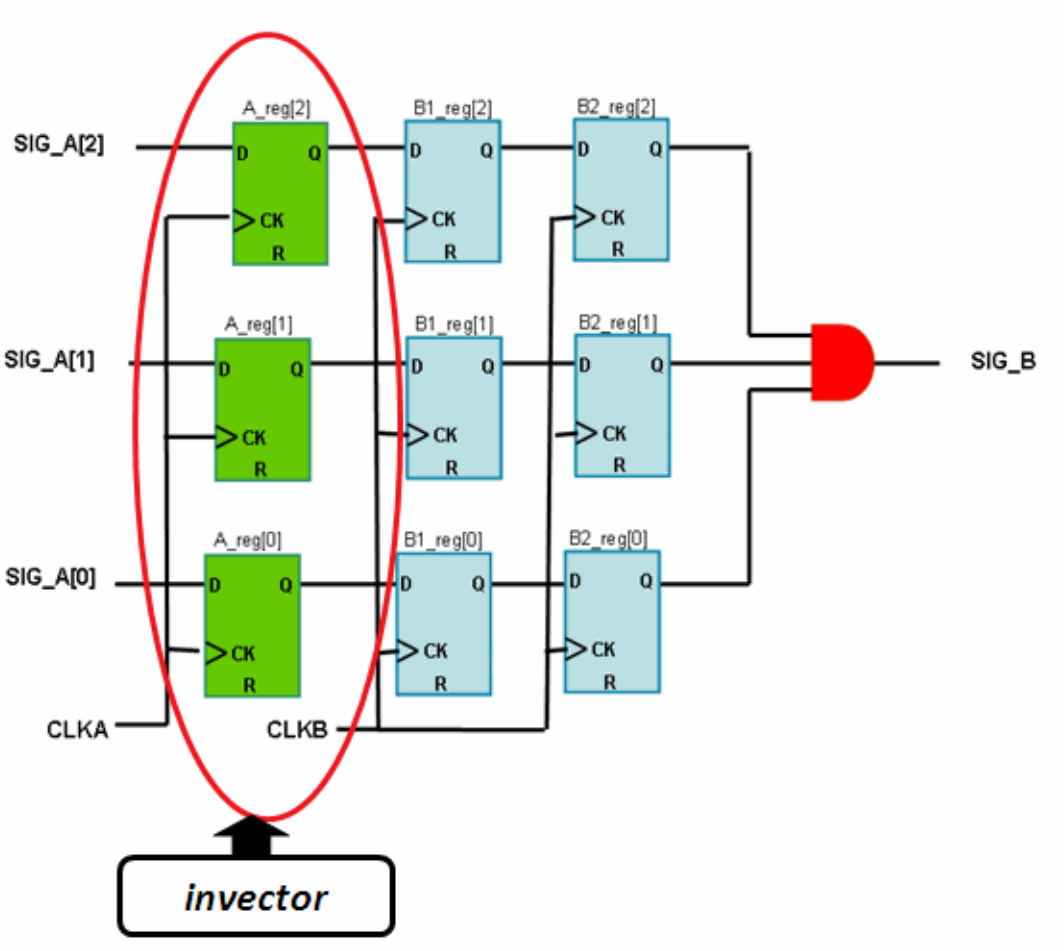
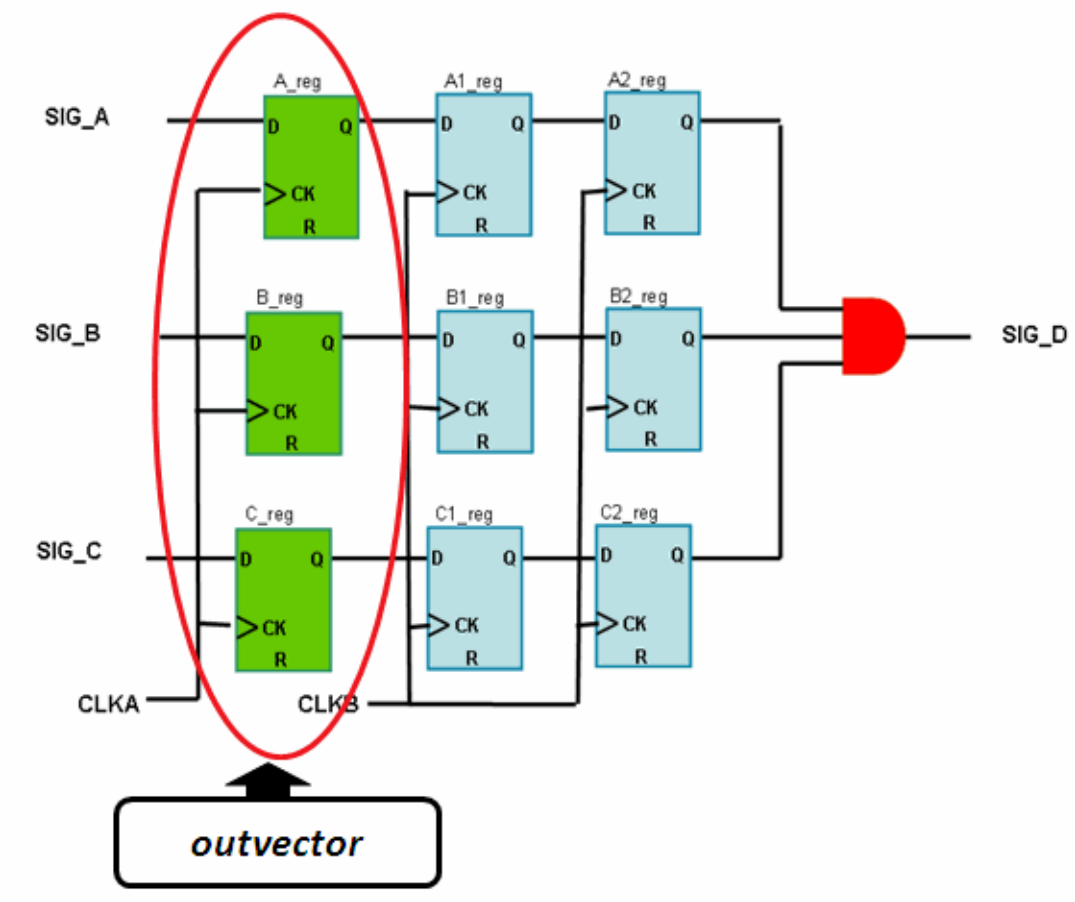


Figure 5-16 check_vector_conv (outvector convergence)



filter_paths

Description	Specifies the convergence paths to filter using a list of convergence sources and their corresponding convergence points.
Possible Value	`{from <object> end_point <object>} ...`
Atomic Checks	`cdc_conv_source_filtered` indicates that the specified convergence path has been filtered.

Example	`set_attribute $rule_instance filter_paths [list \`
	`  from [find -instance A_reg] \`
	`  end_point [find -instance B1_reg]]`

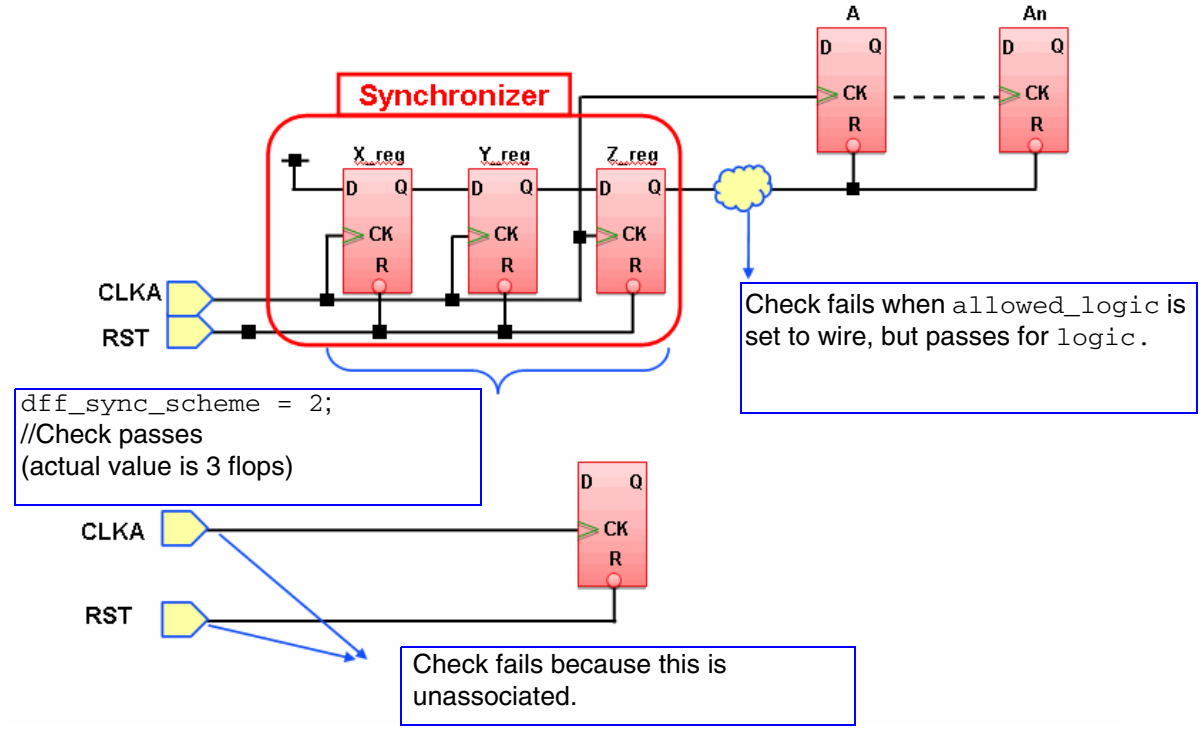
All Rights Reserved.

cdc_setreset_sync_rule

```
cdc_setreset_sync_rule
     analysis_mode <structural | functional>
     destination_clock <clock_object>
     consider_clock_phase <yes | no>
     consider_clock_group <yes | no>
     allow_syncchain_with_no_sr <no | yes>
     dff_sync_scheme {min <2>}
     allowed_logic {logic | inv | wire}
     check_type <both | set | reset>
     set_reset_source <list_of_objects>
     filter_paths {driver {object} from <object> to <object>} ...
     exclude_atomic_checks <list_of_atomic_checks>
     suppress_options_in_header <list_of_options>
```

Structurally verifies that the set/reset drivers are synchronized to the destination domain. Criteria for the synchronization is specified through the rule attributes listed below. If the synchronization scheme does nreot adhere to the specified criteria, the tool will report a violation of this rule. If there are multiple control paths to setreset driver, violation for each unique path to the setreset driver will be reported.

Figure 5-17 cdc_setreset_sync_rule



© 1999-2023
All Rights Reserved.

Rule Attributes

Criteria for the rule is specified through rule attributes. To view the rule attributes from the tool, use the following command:

```
report rule source "cdc_setreset_sync_rule" -verbose
```

The following table details the rule attributes for this check. Some of these rule attributes are provied by the atomic checks described in (described in "Atomic Checks" on page 533).

analysis_mode

Description	Specifies whether to perform structural or functional checks.
Possible Value	`<structural | functional>`
Example	`set_attribute $rule_instance analysis_mode functional`

destination_clock

Description	Specifies destination clock. This must be an SDC object of a clock.
Possible Value	*`clock_object`*

Example	`set_attribute $rule_instance destination_clock \ [find -sdcobj <clk2>]`
	You can also use the keywords `clocked` and `unclocked` to specify a destination clock. However, you cannot combine these keywords with SDC objects (for example, you cannot designate a source clock as `clocked`/`unclocked`, and then designate an SDC object for the destination clock); both must be SDC objects or both must be keywords.
	For example:
	`set_attribute $rule_instance destination_clock \ [find -sdcobj <clk2>]`
	`set_attribute $rule_instance destination_clock unclocked`

© 1999-2023 All Rights Reserved.

consider_clock_phase

Description	Specifies whether to check `clock_phase` in the synch chain.
Possible Value	<yes | no> `Default is yes.`
Example	`set_attribute $rule_instance consider_clock_phase no`

consider_clock_group

Description	Specifies whether to take `clock_groups` into account when considering a CDC. Note: Source and destination clocks are treated as asynchronous even if they are in the same clock group.
Possible Value	<yes | no> Default is yes.
Example	`set_attribute $rule_instance consider_clock_groups no`

allow_syncchain_with_no_sr

Description	Specifies whether synch chains with no set/reset signals are allowed
Possible Value	<no | yes> Default is no.
Example	`set_attribute $rule_instance allow_syncchain_with_no_sr yes`

dff_sync_scheme

Description	Specifies the minimum/maximum number of flops and whether the first destination is a latch or DFF.

© 1999-2023
All Rights Reserved.

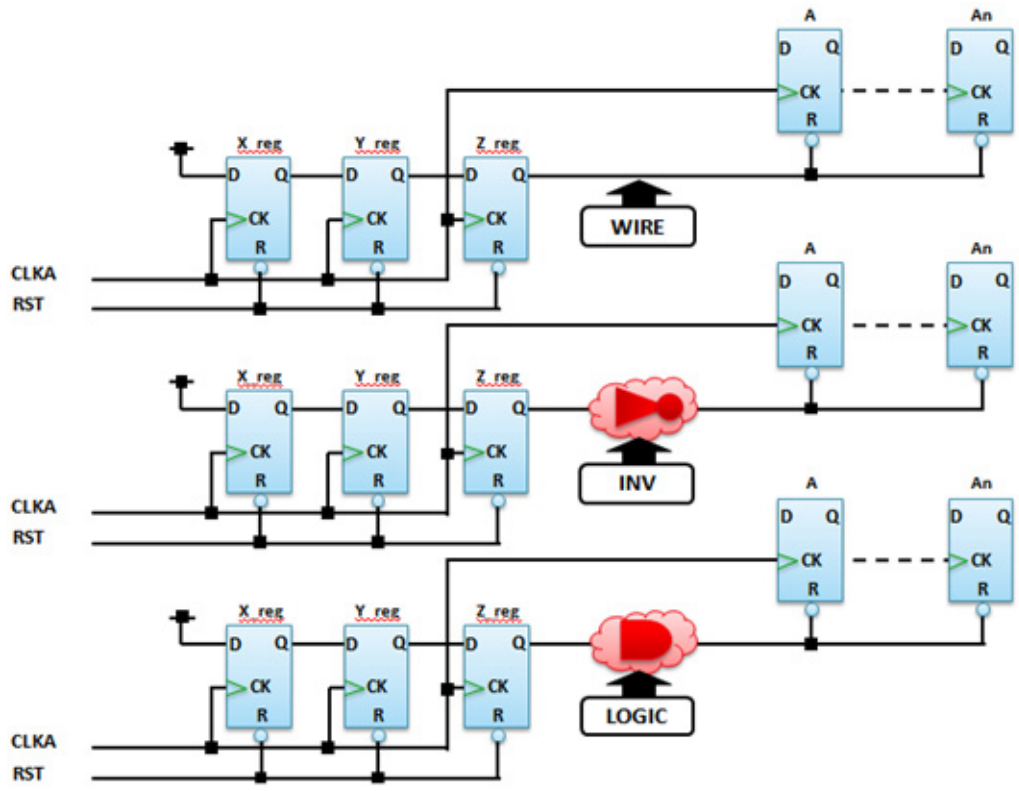
Possible Value	{min <2>} Default is `min 2`. `min` — Specifies the minimum number of DFFs (must be 2 or more) `max` — Specifies the maximum number of DFFs (must be greater than or equal to the minimum number) `first` — Specifies whether the first element in the destination is a latch or a DFF
Atomic Checks	`cdc_setreset_min_dff_check` checks if the minimum number of flops has been met.
Example	`set_attribute $rule_instance dff_sync_scheme "min 2"`

allowed_logic

Description	Specifies the allowed logic type for the sync chain.
Possible Value	`{logic | inv | wire}` Default value is `logic`. `logic` — Any logic is allowed as a set-reset driver `wire` — Glue logic is not allowed as a set-reset driver `inv` — Only buffers/invertors are allowed as a set-reset driver
Atomic Checks	`cdc_setreset_logic_type_check` checks the logic type in the sync chain (from driver to the first key point).
Example	`set_attribute $rule_instance allowed_logic logic`

© 1999-2023 All Rights Reserved.

Figure 5-18 allowed_logic



check_type

Description	Specifies whether the port of a flip flop is `set` or `reset`.
Possible Value	`<both | set | reset>`
	Default value is `both`.
	`both` Perform both set and reset checks
	`set` Perform only set checks
	`reset` Perform only reset checks
Example	`set_attribute $rule_instance check_type set`

All Rights Reserved.

set_reset_source

Description	Specifies the source for the set/reset. When you specify this rule attribute, the tool will perform additional checks to ensure that the specified list of objects are in the fan-in cone of the set/reset driver.
Possible Value	`<list_of_objects>`
Example	`set_attribute $rule_instance set_reset_source \` `  [find -port resetn]`

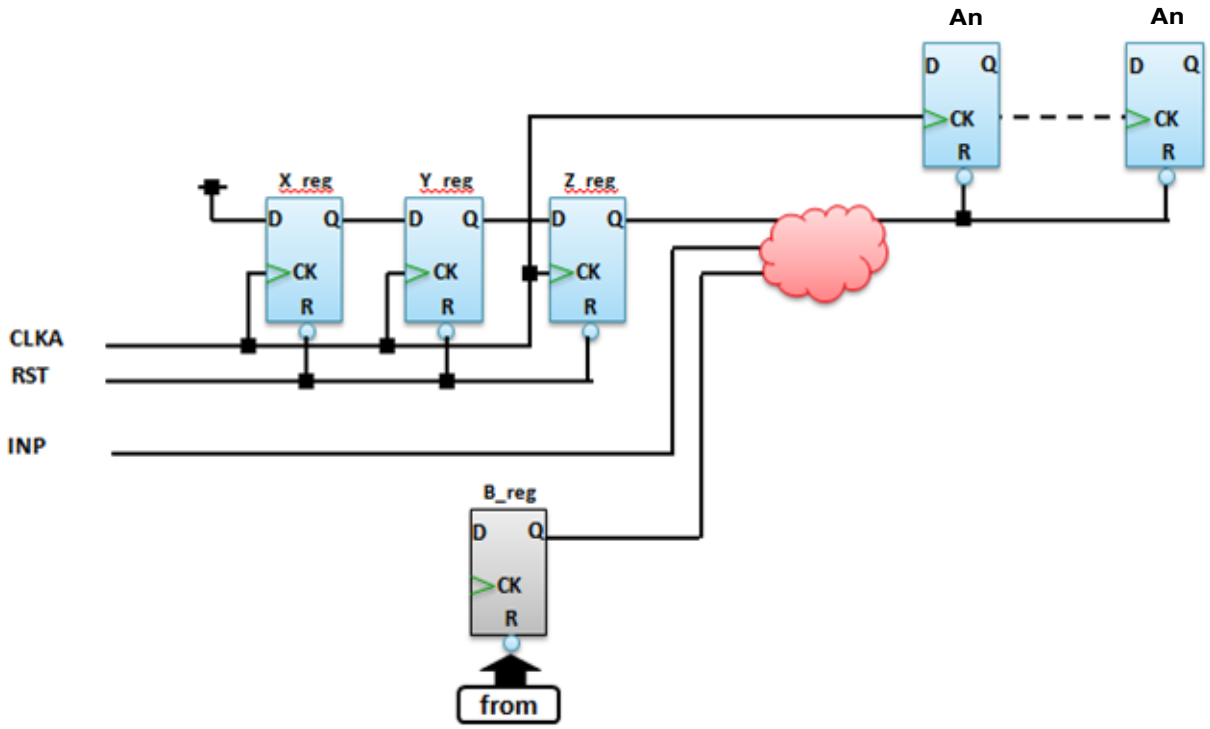
filter_paths

Description	Filters out paths with the specified `from`/`to` or with the specified driver object.
Possible Value	`{driver {object} from <object> to <object>} ...`

Example	`set_attribute $rule_instance filter_paths [list \`
	`  from [find -instance A_reg] \`
	`  to [find -instance B1_reg]]`

© 1999-2023 All Rights Reserved.

Figure 5-19 filter_paths for setreset



exclude_atomic_checks

Description	Excludes/disables atomic checks for the specified rule instance.
Possible Value	`<list_of_atomic_checks>`
Example	`set_attribute $rule_instance exclude_atomic_checks \ [list cdc_setreset_target_clock_sync_check \ cdc_setreset_min_dff_check ]`

suppress_options_in_header

Description	Excludes the specified rule options from the `REPORT RULE CHECK` header.
Possible Value	`<list_of_instances_to_exclude>`

© 1999-2023
All Rights Reserved.

Example	For example, to exclude the paths specified by the `filter_paths` attributes from the `REPORT RULE CHECK` header, use the `set_attribute` command on each rule instance: `set_attribute $rule_instance suppress_options_in_header \ [ list filter_paths]`

© 1999-2023 All Rights Reserved.

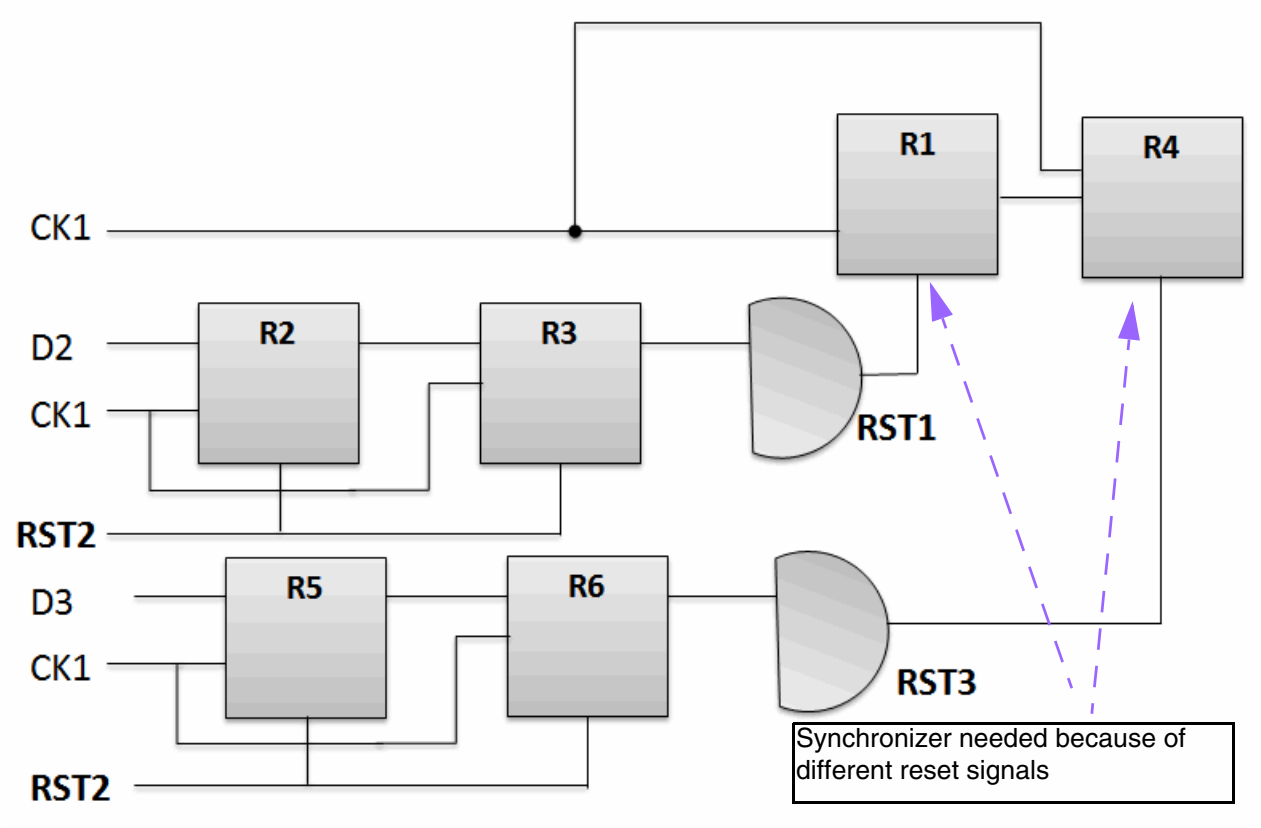
cdc_sr_sync_crossing_rule

```
cdc_sr_sync_crossing_rule
     analysis_mode <structural | functional>
     source_clock <clock_object>
     destination_clock <clock_object>
     from <start_object>
     to <end_object>
     modules <list_of_modules>
     consider_clock_group <yes | no>
     consider_clock_phase <yes | no>
     expand_same_clock_group <no | yes>
     consider_same_sr_source <no | yes>
     expand_same_sr_source <no | yes>
     sync_chain_logic {dff <wire | buffer | logic> mux <wire | buffer | logic>}
     sync_chain_fanout {dff <single | multiple> mux <single | multiple>}
     cdc_path_logic {dff <wire  | buffer | logic> mux <wire | buffer | logic>
                     user <wire | buffer | logic>}
     cdc_path_fanout {dff <single | multiple> mux <single | multiple>
                     user <single | multiple>}
     allow_sync_scheme <all | dff | mux | user>
     dff_sync_scheme {min <2> max <infinite> first <dff | latch>}
     mux_sync_scheme {min 0 max 0 first <dff | latch>}
     monitor_cycles <1 | 2>
     exclude_modules <list_of_modules>
     filter_paths {from <object> to <object>} ...
     exclude_atomic_checks <list_of_atomic_checks>
     suppress_options_in_header <list_of_options>
```

Structurally and functionally verifies each set/reset crossing. This check can identify meta-stabilities caused by timing uncertainty between the deactivation of a set/reset signal and a flip-flop's clock triggering edge.

© 1999-2023 All Rights Reserved.

Figure 5-20 cdc_sr_sync_crossing_rule



analysis_mode

Description	Specifies whether to perform structural or functional checks.
Possible Value	`<structural | functional>`
Example	`set_attribute $rule_instance analysis_mode functional`

destination_clock

Description	Specifies destination clock, must be an SDC object of a clock.
Possible Value	*`clock_object`*

© 1999-2023
All Rights Reserved.

Example	`set_attribute $rule_instance destination_clock \ [find -sdcobj <clk2>]`
	
	You can also use the keywords `clocked` and `unclocked` to specify a destination clock. However, you cannot combine these keywords with SDC objects (for example, you cannot designate a source clock as `clocked`/`unclocked`, and then designate an SDC object for the destination clock); both must be SDC objects or both must be keywords.
	
	For example:
	
	`set_attribute $rule_instance destination_clock \ [find -sdcobj <clk2>]`
	`set_attribute $rule_instance destination_clock unclocked`

from

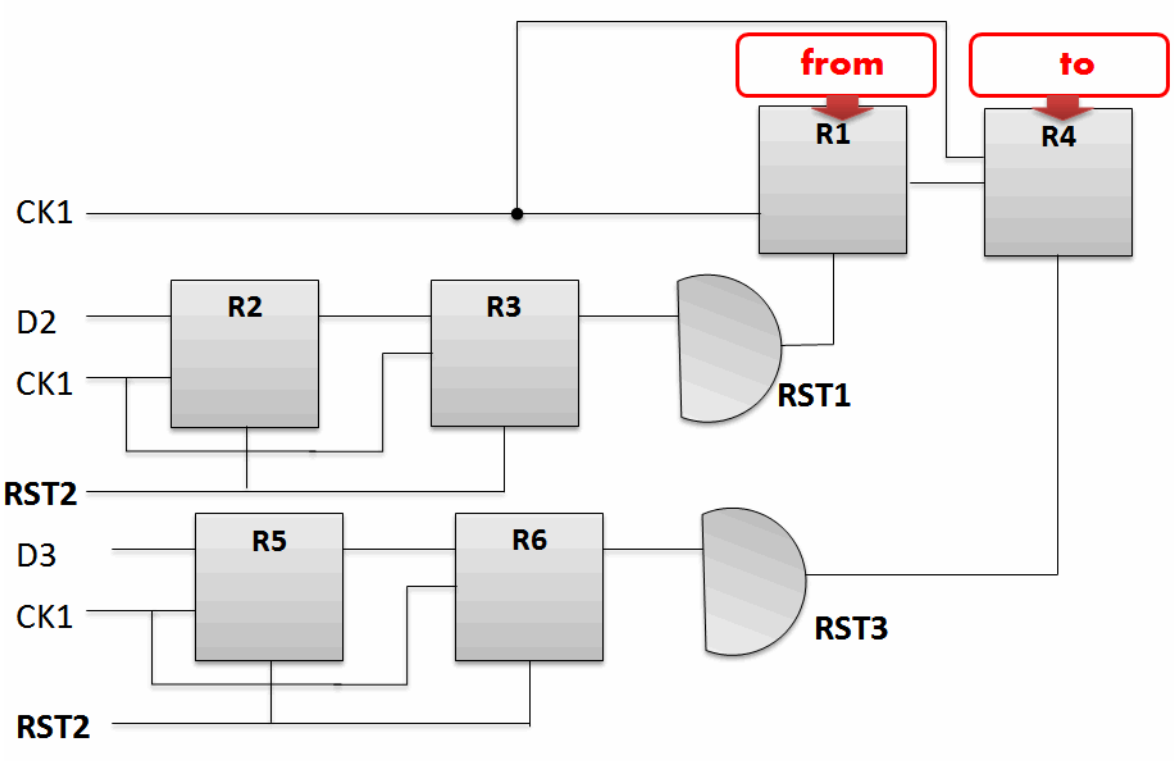
Description	Specifies the start point of a crossing; must be a design object. (optional)
Possible Value	`start_object`
Example	`set_attribute $rule_instance from [find -instance \ <top/sub/abc_reg>]`

to

Description	Specifies the end point of a crossing; must be a design object. (optional)
Possible Value	`end_object`
Example	`set_attribute $rule_instance to \ [find -instance <top/sub/xyz_reg>]`

© 1999-2023 All Rights Reserved.

Figure 5-21 cdc_sr_sync_crossing_rule from and to



modules

Description	Constrains the rule instance of a CDC structural check such that it is applied to the CDC paths of the specified modules.
Possible Value	`<list_of_modules>`
Example	`set_attribute $rule_instance module \` `[ find -design modue_b ]`

consider_clock_group

Description	Specifies whether to take `clock_groups` into account when considering a CDC. Note: Source and destination clocks are treated as asynchronous even if they are in the same clock group.

© 1999-2023
All Rights Reserved.

Possible Value	<yes | no> Default is yes.
Example	`set_attribute $rule_instance consider_clock_groups no`

consider_clock_phase

Description	Specifies whether to take `clock_phase` into account when considering a CDC. Yes is the default. Note: Source and destination clocks are treated as asynchronous even if they are in the same clock group.
Possible Value	<yes | no> Default is yes.
Example	`set_attribute $rule_instance consider_clock_phase no`

expand_same_clock_group

Description	Specifies whether to expand the paths between the source and destination clocks within the same clock group.
Possible Value	<no | yes> Default is `no`.
Example	`set_attribute $rule_instance expand_same_clock_group yes`

consider_same_sr_source

Description	Specifies whether to consider same set/reset source to determine set/reset synchronization crossing
Possible Value	<no | yes> Default is `no`.
Example	`set_attribute $rule_instance consider_same_sr_source yes`

© 1999-2023 All Rights Reserved.

expand_same_sr_source

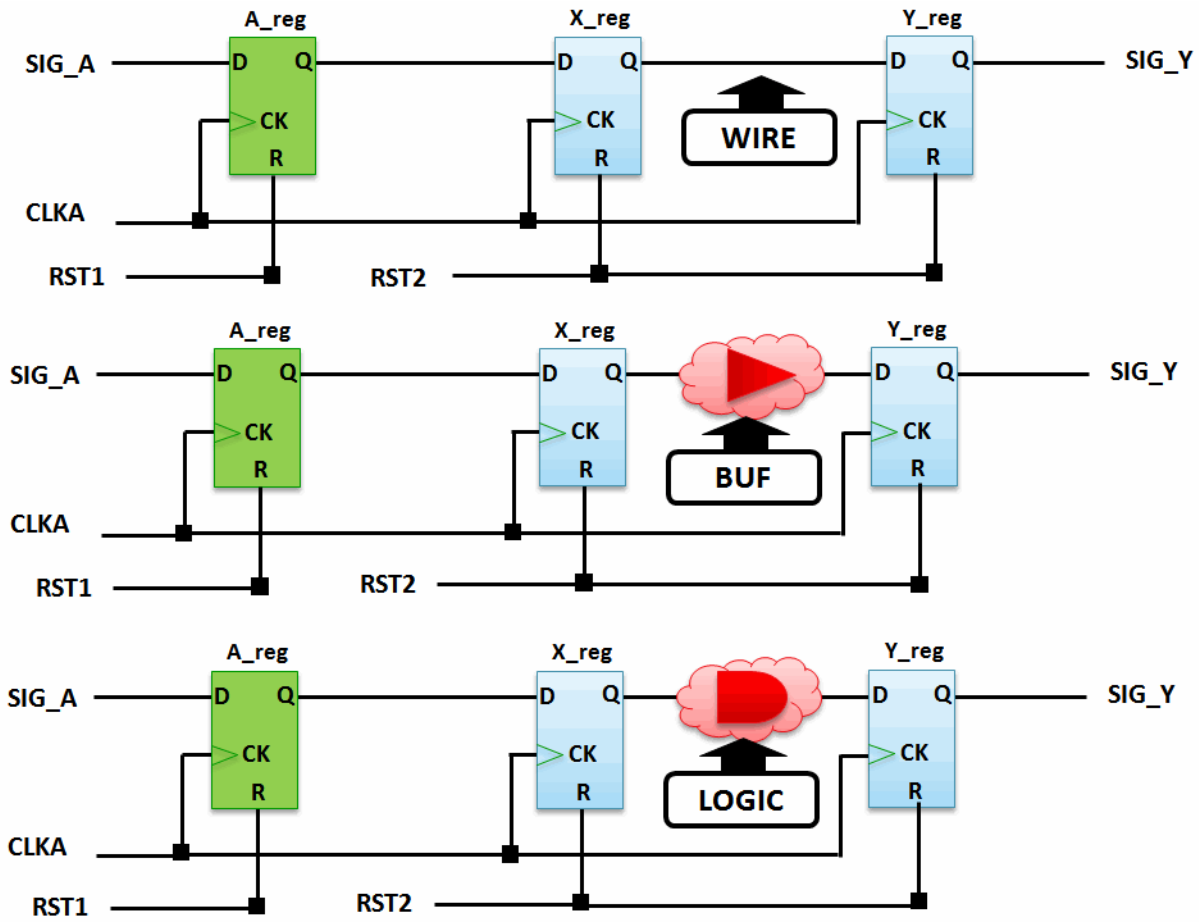
Description	Specifies whether to expand set/reset synchronization crossing paths that have the same set/reset source.
Possible Value	<no | yes> Default is `no`.
Example	`set_attribute $rule_instance expand_same_sr_source yes`

sync_chain_logic

Description	Specifies the sync chain logic.
Possible Value	`{dff <wire | buffer | logic> mux <wire | buffer | logic>}` Default value is `dff wire mux wire`. `dff` — For DFF synchronization schemes `mux` — For MUX synchronization schemes `wire` — No logic is allowed `buffer` — Allows only buffers/inverters inside the synchronization chain `logic` — Allows any logic inside the synchronization chain
Atomic Checks	`cdc_ctrl_logic_type_check` checks logic type in the data path (for DFF crossings). `cdc_data_logic_type_check` checks logic type in the data path (for MUX crossings).
Example	`set_attribute $rule_instance sync_chain_logic dff wire` `set_attribute $rule_instance sync_chain_logic mux buffer`

© 1999-2023 All Rights Reserved.

Figure 5-22 synch_chain_logic (for D Flip Flop)



sync_chain_fanout

Description	Specifies whether fanouts are allowed in the sync chain.	
Possible Value	`{dff <single | multiple> mux <single | multiple>}`	
	`dff`	For DFF synchronization schemes
	`mux`	For MUX synchronization schemes
	`single`	Fanouts are not allowed
	`multiple`	Fanouts are allowed

© 1999-2023
All Rights Reserved.

Atomic Checks	`cdc_ctrl_multi_chain_check` checks if sync-chain fanouts to multiple destinations (for DFF). `cdc_data_multi_chain_check` checks if sync-chain fanouts to multiple destinations (for MUX).
Example	`set_attribute $rule_instance sync_chain_fanout \` `  [list dff multiple mux multiple]`

Refer to Figure 5-6 on page 99 for an example.

cdc_path_logic

Description	Specifies the logic type in the CDC path. Default value is `dff wire mux wire user wire`	
Possible Value	`{dff <wire | buffer | logic> mux <wire | buffer | logic> user <wire | buffer | logic>}`	
	`dff`	For DFF synchronization schemes
	`mux`	For MUX synchronization schemes
	`user`	For user synchronization scheme
	`wire`	No logic is allowed
	`buf`	Allows only buffers/inverters inside the synchronization chain
	`logic`	Allows any logic inside the synchronization chain
Atomic Checks	`cdc_data_logic_type_check` checks logic type in the data path.	
Example	`set_attribute $rule_instance cdc_path_logic \` `  [list dff wire mux wire user logic]`	

Refer to Figure 5-7 on page 100 and Figure 5-8 on page 101 for an example.

cdc_path_fanout

Description	Specifies whether a CDC path can fanout to multiple destinations.

© 1999-2023 All Rights Reserved.

Possible Value	`{dff <single | multiple> mux <single | multiple> user <single | multiple>}`
	`dff` For DFF synchronization schemes
	`mux` For MUX synchronization schemes
	`user` For user synchronization schemes
	`single` Fanouts are not allowed
	`multiple` Fanouts are allowed
Atomic Checks	`cdc_path_destination_check` checks for multiple destinations (fanouts) in the CDC path.
Example	`set_attribute $rule_instance cdc_path_fanout \ [list dff multiple mux multiple user single]`

allow_sync_scheme

Description	Specifies the allowed types of sync schemes for the CDC path being validated:
Possible Value	`<all | dff | mux | user>`
	`all` Allows all clock schemes
	`dff` Only DFF type of sync scheme is allowed
	`mux` Only MUX type of sync scheme is allowed
	`user` Only user specified sync module is allowed

dff_sync_scheme

Description	Specifies the minimum/maximum number of DFFs allowed in the sync chain and whether the first destination should be a latch or a DFF. Default value is `min 2 max infinite first dff`.
Possible Value	`{min <2> max <infinite> first <dff | latch>}`

© 1999-2023 All Rights Reserved.

	`min`	Specifies the minimum number of DFFs (must be 2 or more)
	`max`	Specifies the maximum number of DFFs (must be greater than or equal to the minimum number)
	`first`	Specifies whether the first destination is a latch or DFF
Atomic Checks	`cdc_ctrl_min_sync_chain_check` checks if the maximum number of flops is met.	
	`cdc_data_max_sync_chain_check` checks if the minimum number of flops is met	
	`cdc_ctrl_first_dlatch_check` Checks if the first sequential element in the destination is a latch.	
Example	`set_attribute $rule_instance dff_sync_scheme \ [list min 2 max 3 first latch]`	

Refer to Figure 5-9 on page 103 for an example.

mux_sync_scheme

Description	For MUX synchronizers, specifies the minimum/maximum number of flops and the logic type of the first destination.	
Possible Value	`{min 0 max 0 first <dff | latch>}`	
	Default value is `min 0 max 0 first dff`.	
	`min`	For MUX synchronizers, specifies the minimum number of flops (must be 0 or more)
	`max`	For MUX synchronizers, specifies the maximum number of flops (must be greater than or equal to the minimum number)
	`first`	Specifies whether the first destination is a latch or DFF

© 1999-2023 All Rights Reserved.

Atomic Checks	`cdc_data_first_dlatch_check` checks if the first sequential element in the destination is a latch or a DFF. `cdc_data_max_sync_chain_check` checks if the maximum number of flops is met. `cdc_data_min_sync_chain_check` checks if the minimum number of flops has been met.
Example	`set_attribute $rule_instance mux_sync_scheme \` `  [list min 1 max 3 first latch]`

Refer to Figure 5-10 on page 104 for an example.

monitor_cycles

Description	Specifies the number of clock cycles used to monitor the signals for CDC functional checking purposes. When the source and destination clocks have the same clock periods, then one monitor cycle equals the clock period multiplied by 1. When the clock periods are different, then one monitor cycle equals the least common multiplier (LCM) multiplied by one. For example, if `monitor_cycles` changes to 2, then it would be the clock period multiplied by 2 or the LCM multiplied by 2.
Possible Value	`<1 | 2>` Default value is `1` clock cycle.
Example	`set_attribute $rule_instance monitor_cycles 2`

exclude_modules

Description	Constrains the rule instance of a CDC structural check such that it is not applied to the CDC paths of the specified modules.
Possible Value	`<list_of_modules>`
Example	`set_attribute $rule_instance exclude_module \` `[ find -design mod_a ]`

© 1999-2023 All Rights Reserved.

filter_paths

Description	A list of path specifications to filter out.
Possible Value	`{from <object> to <object>} ...`

Example	**For a single destination:**
	`set_attribute $rule_instance filter_paths [list \`
	`  from [find -instance A_reg] \`
	`  to [find -instance B1_reg]]`
	This would filter `A_reg->B1_reg`.
	For multiple destinations, use the following format. If you use the `set_attribute filter_paths` command more than once, the last command overrides all previous commands.
	For multiple destinations:
	`set_attribute $rule_instance filter_paths \`
	`  [list from [find -instance A_reg] to \`
	`  [find -instance {B1_reg C1_reg}] ]`
	This would filter `A_reg->B1_reg` and `A_reg->C1_reg`.
	`set_attribute $rule_instance filter_paths [list \`
	`  [ list from [find -instance {a_reg b_reg} ]`
	`  to [ find -instance { c_reg d_reg }]] \`
	`  [ list from [ find -instance f_reg ] to \`
	`  [ find -instance g_reg ]] ]`
	This would filter `a_reg->c_reg`, `a_reg->d_reg`, `b_reg->c_reg`, `b_reg->d_reg` paths as well as `f_reg->g_reg`.

exclude_atomic_checks

Description	Excludes/disables atomic checks for the specified rule instance.
Possible Value	`<list_of_atomic_checks>`

Example	`set_attribute $rule_instance exclude_atomic_checks \`
	`  [list cdc_path_logic_type_check \`
	`  cdc_path_destination_check ]`

© 1999-2023 All Rights Reserved.

suppress_options_in_header

Description	Excludes the specified rule options from the `REPORT RULE CHECK` header.
Possible Value	`<list_of_instances_to_exclude>`
Example	For example, to exclude the paths specified by the `filter_paths` attributes from the `REPORT RULE CHECK` header, use the `set_attribute` command on each rule instance: `set_attribute $rule_instance suppress_options_in_header \ [ list filter_paths]`

© 1999-2023 All Rights Reserved.

© 1999-2023 All Rights Reserved.

6

Policy Rule Checks

A *policy rule* is a check performed on a design, with or without SDC, that reports whether a given criterion is met. Check levels can vary, from basic structural analysis to complex timing graph analysis—or a combination of both.

You can customize policy rules, write your own rules (for example, reports can be turned into policy rules), and perform Tcl commands on policy rules.

Policy rules are not checked by default. To enable policy rules, you must add them to a *rule set* or read in the `ccd_default_cdc_ruleset.tcl` (or, `ccd_default_cdc_ruleset.ntcl` if you are in Tcl mode) default set (see ADD RULE SET). For more information on working with rule sets, refer to About Rule Checks in the *Conformal Constraint Designer User Guide*.

© 1999-2023 All Rights Reserved.

Overview of Policy Rule Checks

The following table describes some of the terms that you will need to know in order to understand the list of rule checks.

Note: Some rules depend on user-defined parameters. Use the `SET CCD PARAMETER` command to assign parameter values and `REPORT CCD PARAMETER` to view them.

Term	Description
real clock	Refers to clocks that are created using `create_clock` with `source_objects`.
virtual clock	Refers to clocks that are created using `create_clock` without `source_objects`.

© 1999-2023 All Rights Reserved.

Term	Description
generated clock	Refers to clocks that are created using `create_generated_clock` and that always have `source_objects`.
logical pins	Refers to pins of hierarchical (non-leaf) instances, which may disappear when flattening the design.
clock tree	Refers to all the nets in the combinational fan-out of a clock, after constant propagation has occurred. That is, gates with tied values do not propagate a clock's tree.
root clocks	Refers to the source objects of real clocks in a clock group. This also includes generated clocks whose master clock does not belong to the same group. **Note:** You can use the `ADD CLOCK GROUP` command to define groups of synchronous clocks.

Rule Classification

The rule checks that are performed on SDC data are classified into the following categories:

- Syntax errors
- Syntax rules
- Usage rules
- Structural rules

As implied by their names, categories 2–4 use the rule mechanism to report violations, which means you can specify how many errors you want to check for. When the Conformal Constraint Designer reaches the specified number of errors, it stops parsing the SDC files.

Rules are checked at different times during a session, depending on their category. Syntax and usage rules are checked during Setup mode; structural rules are checked when you switch from *Setup* mode to *Verify* mode, or when you run the `run rule check` command.

Syntax Errors

When the SDC parser encounters syntax errors, it prints a detailed message and stops. You cannot change the severity level of syntax errors. Syntax errors are:

© 1999-2023 All Rights Reserved.

- Tcl syntax errors
- Unknown command(s)

  This error covers any command that is not registered with the Tcl interpreter. You can use the `sdc_add_unhandled` Tcl command, before you read in your SDC file, to declare these unhandled commands.

  Note: This message might also indicate that you have misspelled a command. Before using the `sdc_add_unhandled` Tcl command, ensure that you have spelled the command correctly.

Syntax Rules

The Conformal Constraint Designer checks syntax rules for violations of the command syntax. When the Conformal Constraint Designer encounters these violations, it allows parsing to continue. Here are some examples of syntax violations:

- Known but unsupported command. (Its arguments will not be checked.)
- Unknown option, flag, or argument in a supported command.
- Any known option of a supported SDC command that is not supported (generally, those that are not part of the SDC standard).
- Missing required argument, for example, `create_clock` without `-period`, or without either `-name` or an object list.
- Missing required sub-argument, for example, `set_false_path -from -to CLK1.`
- Wrong argument type (Tcl types: list, string, float, integer, and so forth).
- Numerical argument out of range (for example, negative clock period value).
- Mutually exclusive arguments.

Usage Rules

The Conformal Constraint Designer flags usage rule violations for interactions between otherwise well-built arguments, checks done on specified objects, and desired usage enforcement. The Conformal Constraint Designer runs these checks while parsing each command.

Here are some examples of usage violations:

© 1999-2023 All Rights Reserved.

- Missing objects (wildcard expression that does not match or an empty list returned by a `get_*` command)
- Objects of the wrong type, for example, a cell where a clock is expected
- Use of unusual or undesired commands or command arguments
- Missing optional but necessary parameters
- Other style issues (wildcard use, implicit object referencing, and so forth)

Structural Rules

The Conformal Constraint Designer checks structural rules for the violations listed below when going from Setup to Verify mode, or when you run the `run rule check` command. It reports violations in Verify mode.

Structural rule violations include:

- Connectivity issues
- Suggested constraints or timing exception commands that are missing
- Unused, redundant, overlapping, conflicting, or undesired constraints
- High count of certain constraints or exceptions that are above the user-defined threshold

Reducing Rule Checks

You can use the `min_set.do` file, which enables high priority rules and disables other rules to reduce the amount of data. Source this file before reading in SDC constraints, by executing the following command:

This file is located at:

```
SETUP> dofile install_directory/share/cfm/ccd/tool_kit/min_set.do
```


© 1999-2023 All Rights Reserved.

CCD_SDC_STR*

This section describes the rule checks that relate to the SDC file structure.

© 1999-2023 All Rights Reserved.

CCD_SDC_STR7

Message

`Conflicting constraints`

Default Severity

Warning

Description

Indicates that you have specified conflicting constraints. The following can scenarios trigger this rule check:

- When you have more than one clock defined on a single pin, unless the second invocation of `create_clock` has the `-add` option.
- When you have more than one of the following commands used for any port: `set_input_transition`, `set_driving_cell`, `set_clock_transition`

This is checked when you read in your SDC files using the `READ SDC` command.

Example

1. The following causes a warning, because `CLK1` and `CLK2` share the `clk` pin. To avoid this warning, the `CLK2` definition should have an `-add` option.

```
create_clock -name CLK1 [get_ports {clk}] -period 10 -waveform {0 4}
create_clock -name CLK2 [get_ports {clk}] -period 17 -waveform {0 4}
```

2. The following causes a warning, because `set_input_transition` and `set_driving_cell` are both used on port `pi[1]`.

```
set_input_transition 0.00 [get_ports {pi[1]}]
set_driving_cell [get_ports {pi[1]}]
```


© 1999-2023 All Rights Reserved.

CCD_SDC_STR10

Message

`set_case_analysis conflict`

Default Severity

Warning

Description

Indicates that a `set_case_analysis` command imposes a constant value on a pin that has a conflicting tied logic value, or that two `set_case_analysis` constants cannot be justified because that would require conflicting tied logic values at a given object in their fan-in. This can help diagnose many cases of design over-constraining that are reported by modeling rule `CNST1`.

Note: The conflicting tied value can originate from another `set_case_analysis` command, a pin constraint added in the dofile, or even a constant assignment in the design itself.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Examples

If the DFF '`a_reg`' has its output pin Q connected to the S pin of the MUX '`select`', the following commands will cause a conflict reported by this rule:

```
set_case_analysis 0 [get_pins a_reg/Q]
set_case_analysis 1 [get_pins select/S]
```

If all scannable DFFs have their SE pins connected to the primary input `SCAN_EN` (perhaps through buffers or an even number of inverters), the following commands will cause a conflict reported by this rule:

© 1999-2023 All Rights Reserved.

```
set_case_analysis 0 [get_pins a_reg/SE]
set_case_analysis 1 [get_pins b_reg/SE]
```


© 1999-2023 All Rights Reserved.

CCD_SDC_STR11

Message

```
Object with set_case_analysis is undriven or has an undriven pin in its effective
     fanin
```

Default Severity

Warning

Description

Indicates that a `set_case_analysis` command is specified on a port or pin that either is undriven, or has some object in its effective fanin cone that has some undriven input. This can be a reason for the modeling rule CNST1 to be reported, in which case functional checks (e.g. those used for False Path validation and generation) cannot be performed.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

If the input `X` of gate `I1` is left floating, then this rule will report a command like:

```
set_case_analysis 0 [get_pins I1/X]
```

The same would happen if `I1/X` is driven by an AND gate with one undriven input pin, as long as the other input pin is not tied to `0`.

© 1999-2023 All Rights Reserved.

CCD_SDC_HIER*

This section describes the rule checks that relate to SDC hierarchy.

© 1999-2023
All Rights Reserved.

CCD_SDC_HIER1

Message

`Inconsistent set_dont_touch at the top-level vs. block`

Default Severity

Warning

Description

Indicates that `set_dont_touch` is set at the block, but not the top level—or vice versa. If `set_dont_touch` is set in both, this message indicates that they do not have the same value.

When you switch from Setup to Verify mode, this rule will be checked automatically unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

`Block_A.sdc` has this line:

```
set_dont_touch [get_nets n1]
set_dont_touch [get_nets n2] false
set_dont_touch [get_nets clk]]
```

But it does not have the command:

```
set_dont_touch [get_nets n3]
```

`Top.sdc` has this line:

```
set_dont_touch [get_nets Block_A/n*]
```


© 1999-2023 All Rights Reserved.

But it does not have any of these commands:

```
set_dont_touch [get_nets Block_A/clk]
set_dont_touch [get_nets clk]
```

The Conformal Constraint Designer issues a warning because:

- For `n2`, `set_dont_touch` is set to different values in the block and top level.
- For `n3`, `set_dont_touch` is not set at the block level
- For `clk`, `set_dont_touch` is not set at the top level

© 1999-2023 All Rights Reserved.

CCD_SDC_HIER2

Message

`Inconsistent set_dont_touch_network at the top-level vs. block`

Default Severity

Warning

Description

Indicates that `set_dont_touch_network` is set on a block, but not at the top level—or vice versa. The Conformal Constraint Designer performs this check on a block's ports and pins. The attribute is propagated through the clock tree, which means the top level can set it on a port that drives a block's port, instead of directly on the block's port. If this attribute is set on a clock object, the Conformal Constraint Designer treats it as if it was set on all of its source objects.

When you switch from Setup to Verify mode, this rule will be checked automatically unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

1. `Block_A.sdc` has these lines:

   ```
   create_clock -name CLK1 -period 10 [get_ports clk]
   set_dont_touch_network [get_clocks CLK1]
   ```

   The Conformal Constraint Designer issues a warning if `Top.sdc` does not have `set_dont_touch_network` specified on pin `Block_A/clk` or any clock source in its fan-in.

2. `Top.sdc` has this line:

© 1999-2023 All Rights Reserved.

```
set_dont_touch_network [get_clocks clk]
```

where `Block_A/clk` is in the clock tree of `clk`.

The Conformal Constraint Designer issues a warning if `Block_A.sdc` does not have `set_dont_touch_network` specified for `clk`.

© 1999-2023 All Rights Reserved.

CCD_SDC_HIER3

Message

`Inconsistent set_case_analysis at the top-level vs. block`

Default Severity

Warning

Description

Indicates that there is a `set_case_analysis` on a block object that does not have a tied value at the top level.

Every block object with `set_case_analysis` at the top level, and every block port with a tied value at the top level, should have a consistent `set_case_analysis` at the block.

When you switch from Setup to Verify mode. This rule will be checked automatically unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

Note: Specify the current SDC design using the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

1. `Block_A.sdc` has this line:

   ```
   set_case_analysis 1 [get_pin internal_sel]
   ```

   The Conformal Constraint Designer issues a warning if `Top.sdc` does not have a `set_case_analysis` specified that causes pin `Block_A/internal_sel` to be tied to 1.

2. `Block_A.sdc` has this line:

   ```
   set_case_analysis 0 [get_ports mode]
   ```

   `Top.sdc` has this line:

© 1999-2023 All Rights Reserved.

```
set_case_analysis 0 [get_ports mode_n]
```

where `Block_A/mode` gets the inverted value of `mode_n`.

The Conformal Constraint Designer issues a warning because `Block_A/mode` is tied to 1 at the top level.

© 1999-2023
All Rights Reserved.

CCD_SDC_INT*

This section describes rule checks that relate to SDC integration.

© 1999-2023 All Rights Reserved.

CCD_SDC_INT1

Message

`SDC integration: set_case_analysis`

Default Severity

Warning

Description

Configures the integration of the `set_case_analysis` command and reports the messages generated during this integration process.

This is checked when you run the `INTEGRATE` command.

Example

For example, you are using the default SDC integration configuration (`add rule instance -default`) and have a block called `BLK` that has an input `P` that is driven by a propagated constant value 0 (either from `set_case_analysis` at the full-chip level, or from tied signals in the design). If `BLK`.sdc contains the following command

```
set_case_analysis 0 [get_ports P]
```

you will get the following message when you try to integrate this block constraint into a full-chip SDC file:

```
CCD_SDC_INT1: SDC integration: set_case_analysis
  Severity: Warning Occurrence 1
1: [blk_sca2] In line 1, file BLK.sdc (set_case_analysis): Not promoted because it
is defined on a pin with a consistent tied value
```


© 1999-2023 All Rights Reserved.

CCD_DGN_PRT*

This section describes the rule checks that relate to design partitioning violations.

© 1999-2023 All Rights Reserved.

CCD_DGN_PRT4

Message

`Output not registered`

Default Severity

Warning

Description

Indicates that you have an unregistered output. Every output port must be driven by a latch or a flip-flop.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

© 1999-2023 All Rights Reserved.

CCD_DGN_PRT6

Message

`Unbuffered net connection to output port (post-layout)`

Default Severity

Warning

Description

Indicates that there is a net without dedicated buffering, and it is connected to at least one output port that is reused internally (output port fan-out > 1).

Note: This rule check applies to the block level only.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Note: This rule is not checked for top-level SDC designs when there are blocks defined.

© 1999-2023 All Rights Reserved.

CCD_DGN_RST*

This section describes the rule checks that relate to reset signals.

© 1999-2023 All Rights Reserved.

CCD_DGN_RST1

Message

A reset signal whose multiple fan-outs converge

Default Severity

Warning

Description

Indicates that a reset signal diverges and then reconverges in its combinational fan-outs. If there is another reset signal defined in its fan-out, then that path is blocked and no re-convergence will be considered with that path.

Note: This check will be run only when reset signals have been defined using the `add_reset` command.

This is checked when you run the `RUN RULE CHECK` command.

Example

For example, only one reset signal defined at RST port:

```
add_reset -name rst1 [find -port RST]
```

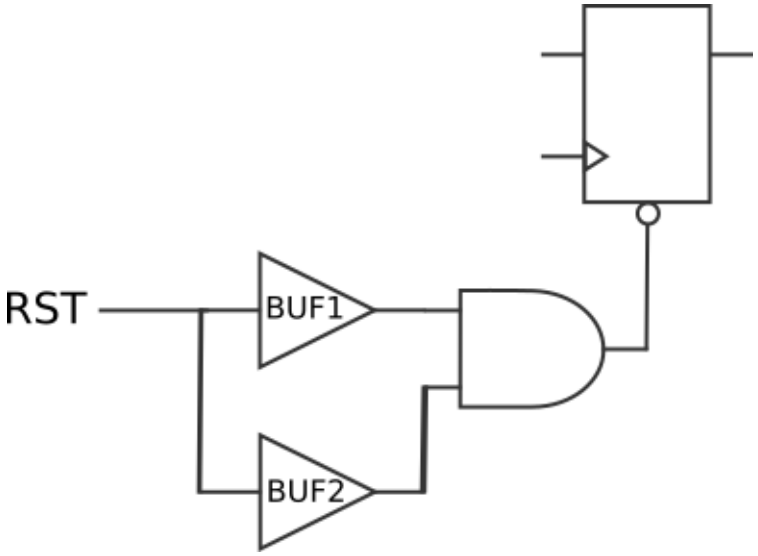
Then reset signal re-converges at the output pin of AND gate and a violation will be reported. However, if there is another reset signal defined at the output of BUF2:

```
add_reset -name rst2 [find -pin BUF2/Y]
```

Then there will be no re-convergence.

© 1999-2023 All Rights Reserved.

Example



© 1999-2023
All Rights Reserved.

CCD_DGN_RST2

Message

`Sequential elements which are not asserted by reset signal`

Default Severity

Warning

Description

Indicates that a sequential element is not asserted by any user-defined reset signals.

Note: This check will be run only when user defines reset signals using the `add_reset` command.

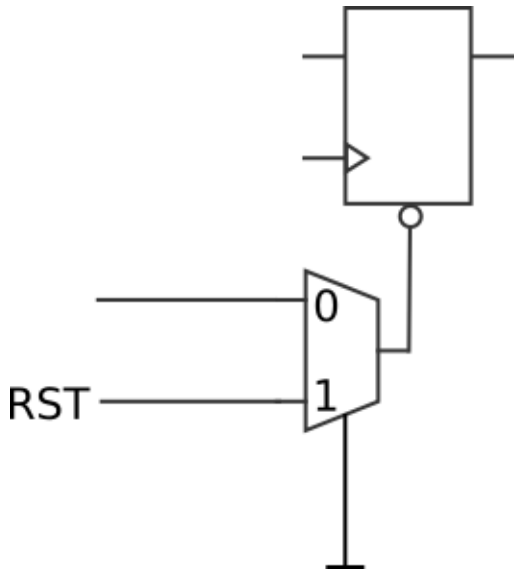
This is checked when you run the `RUN RULE CHECK` command.

Example

The reset signal is defined at RST but MUX select is tied to 0.

© 1999-2023 All Rights Reserved.

Example



© 1999-2023
All Rights Reserved.

CCD_DGN_RST3

Message

`Synchronous reset signal is used as both active-high and active-low`

Default Severity

Warning

Description

Indicates that the synchronous reset signal is used as both active-high and active-low. A reset signal is considered as synchronous if the assertion is associated with the clock input of sequential element.

Note: This check will be run only when reset signals have been defined using the `add_reset` command.

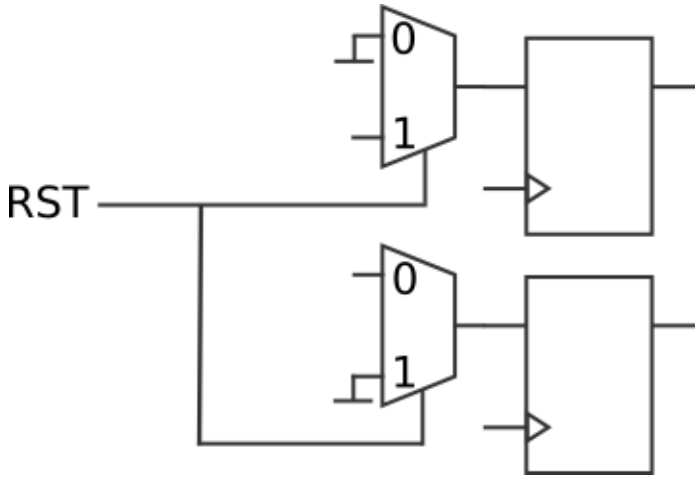
This is checked when you run the `RUN RULE CHECK` command.

Example

The RST is used as active-low at the top DFF and active-high at the bottom DFF.

© 1999-2023 All Rights Reserved.

Example



© 1999-2023
All Rights Reserved.

CCD_DGN_RST4

Message

`Asynchronous reset signal is used as both active-high and active-low`

Default Severity

Warning

Description

Indicates that the asynchronous reset signal is used as both active-high and active-low. A reset signal is considered as asynchronous if the assertion is independent of the clock input of sequential element.

Note: This check will be run only when reset signals have been defined using the `add_reset` command.

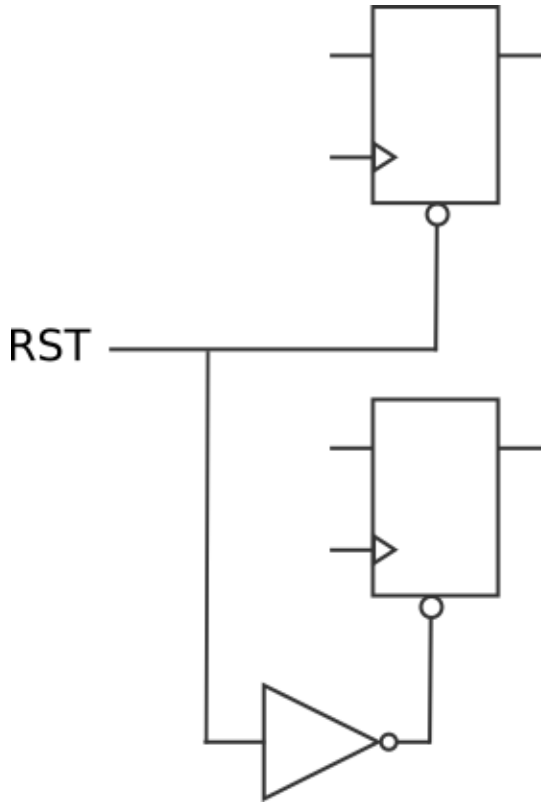
This is checked when you run the `RUN RULE CHECK` command.

Example

The RST is used as active-low at the top DFF and active-high at the bottom DFF.

© 1999-2023 All Rights Reserved.

Example



© 1999-2023
All Rights Reserved.

CCD_DGN_CMB*

This section describes the rule checks that relate to combinational path violations.

© 1999-2023 All Rights Reserved.

CCD_DGN_CMB1

Message

```
Unconstrained input/output delay value for combinational path
```

Default Severity

Warning

Description

Indicates that you have an unconstrained input/output delay value for a combinational path. Every pair of primary inputs and outputs that are connected to a combinational path must have a `set_min_delay`/`set_max_delay` specified, or a complete set of `input`/`output` delays with respect to the same clock (or, independent of any clock).

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

This rule will not be reported for an input/output port pair if all the combinational paths connecting them are matched by one or more `set_false_path` commands that have no clocks in their path specification.

Example

In the following example, a combinational path exists between `IN1` and `OUT2`, and a maximum delay is defined:

```
set_max_delay 3 -from IN1 -to OUT2
```

The Conformal Constraint Designer issues a warning because `set_min_delay` is missing.

© 1999-2023 All Rights Reserved.

CCD_DGN_CMB2

Message

`Inconsistent set_min_delay/set_max_delay for combinational path`

Default Severity

Warning

Description

Indicates that you have an inconsistent specification for `set_min_delay` and `set_max_delay`. The rule check detects whether `min` > `max` for each pair of `set_min_delay` and `set_max_delay` with the same path specification.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

In the following example, the Conformal Constraint Designer issues a warning because the minimum delay is greater than the maximum delay for the same combinational path:

```
set_max_delay 3 -through U1/Z -to OUT2
set_min_dealy 3.5 -through U1/Z -to OUT2
```


© 1999-2023 All Rights Reserved.

CCD_DGN_CMB3

Message

```
Inconsistent input/output delay value vs. clock period for combinational path, or
     -min > -max
```

Default Severity

Warning

Description

Indicates that you have a combinational path with inconsistencies in input and output delay values versus clock periods. That is, you get a rule check violation if the expression defined in the CCD parameter `SDC_CMB3_CHECK` is not satisfied.

`SDC_CMB3_CHECK`'s default setting is:

`input_delay` + *`output_delay`* >= *`clock_period`*

Also, this rule checks whether `min > max` for all input and output delays—even for delays without clocks.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

This rule will not be reported for an input/output port pair if all the combinational paths connecting them are matched by one or more `set_false_path` commands.

Example

The following example defines a virtual clock `vCLK3` with a period of 8, and input and output delays of 5.00:

```
create_clock -period 8 -name vCLK3
set_input_delay 5.00 {IN1, IN2} -clock vCLK3
set_output_delay 5.00 {OUT1, OUT2} -clock vCLK3
```

In the dofile, `SDC_CMB3_CHECK` is defined as:

© 1999-2023 All Rights Reserved.

```
set ccd parameter SDC_CMB3_CHECK "input_delay + output_delay < period * 0.9"
```

The Conformal Constraint Designer issues a warning because the sum of the input and output delays is greater than 90% of the clock period.

© 1999-2023 All Rights Reserved.

CCD_DGN_CMB5

Message

`Combinational constraints present on elements belonging to internal sequential path`

Default Severity

Warning

Description

Indicates that you have combinational constraints on elements that belong to internal, sequential paths. This rule checks whether a `-from` or `-to` point of a `set_min_delay` or `set_max_delay` is in a path between sequential elements.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

The Conformal Constraint Designer issues this message if you specify the delay for a sequential path, because it can compute this delay automatically.

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF*

These rule checks are for violations that relate to clock definitions.

© 1999-2023 All Rights Reserved.

© 1999-2023
All Rights Reserved.

CCD_CLK_DEF1

Message

```
Unconstrained clock: A clock pin does not belong to any clock tree (missing
     create_clock or create_generated_clock command)
```

Default Severity

Warning

Description

Indicates that there is an unconstrained clock pin on a sequential element. That is, this rule checks for clocks that do not belong to any clock tree. This can happen if there is a missing `create_clock` or `create_generated_clock` command, or if there is an incorrect `set_case_analysis`.

Note: Each clock tree driver cell is reported once, together with one sequential cell driven by it, and the number of additional such sequential cells. To help the debugging process concentrate on the more important cases first, the detailed messages that appear in the verbose report and in the GUI are sorted by the number of driven sequential cells, in descending order.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

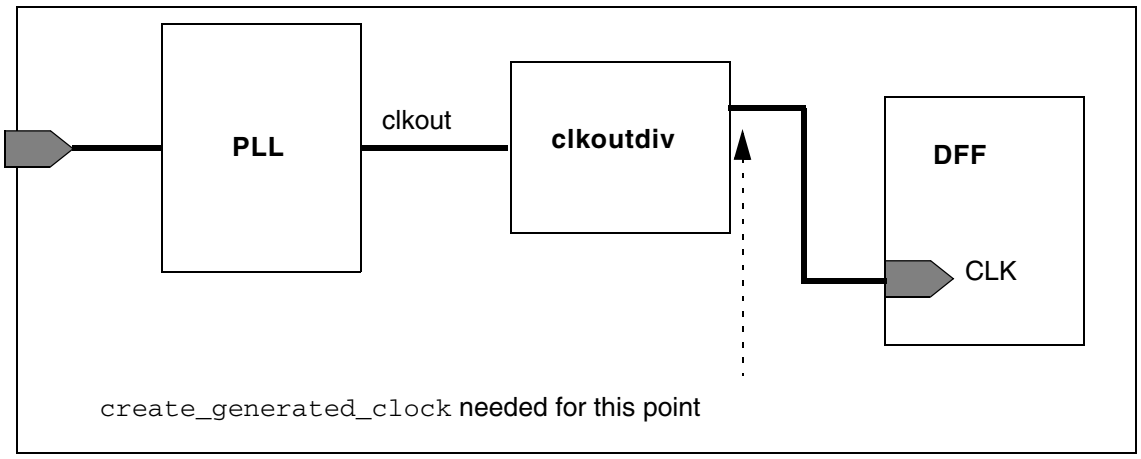
When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

The following generates a warning because the SDC is missing a `create_generated_clock`:

```
create_generate_clock -name div_by_2 -source [get_ports PLL/clkout]
```


© 1999-2023 All Rights Reserved.



© 1999-2023
All Rights Reserved.

CCD_CLK_DEF2

Message

`Unused clock constraint`

Default Severity

Warning

Description

Indicates that you have an SDC statement that defines a clock, but that clock is not used in the design.

A real or generated clock is considered used when:

- The port or pin that it drives is used to drive the clock port of a DLAT or DFF.
- It is the master clock of a generated clock.

A virtual clock is used when either of the SDC commands `set_input_delay` or `set_output_delay` is used to define a delay between the virtual clock and any port or pin in the design.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

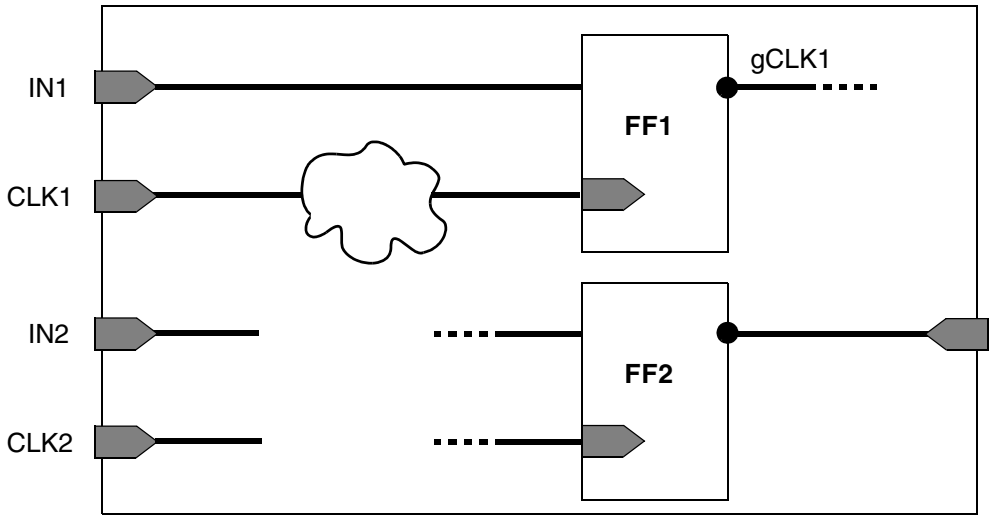
Example

The following example causes a violation, because `CLK2` and `gCLK2` are not used within the block.

```
create_clock -period 8 CLK1
create_clock -period 10 CLK2
```


© 1999-2023 All Rights Reserved.

```
create_generated_clock -name gCLK1 -source CLK1 -divide_by 2 [get_pins FF1/Q]
create_generated_clock -name gCLK2 -source CLK2 -divide_by 4 [get_pins FF2/Q]
```



© 1999-2023
All Rights Reserved.

CCD_CLK_DEF3

Message

```
A source pin (one of the source_objects) of a generated clock is not in the
     transitive fan-out of its 'master clock'
```

Default Severity

Warning

Description

Indicates that you have a source object of a generated clock that is not in the transitive fanout of any of the source objects of its master clock. (Source object does not refer to the pin specified in the `-source` argument.)

When checking if a clock is defined in the fanout of another one, Conformal Constraint Designer takes into account constant propagation. Also, a clock's fanout propagation is interrupted where other clocks are defined. For example, if a clock `GCLK` is defined in the fanout of `MCLK`, then the logic in the fanout of `GCLK` is NOT considered to be in the fanout of `MCLK` unless it can be reached from `MCLK` without traversing the definition point of `GCLK`.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

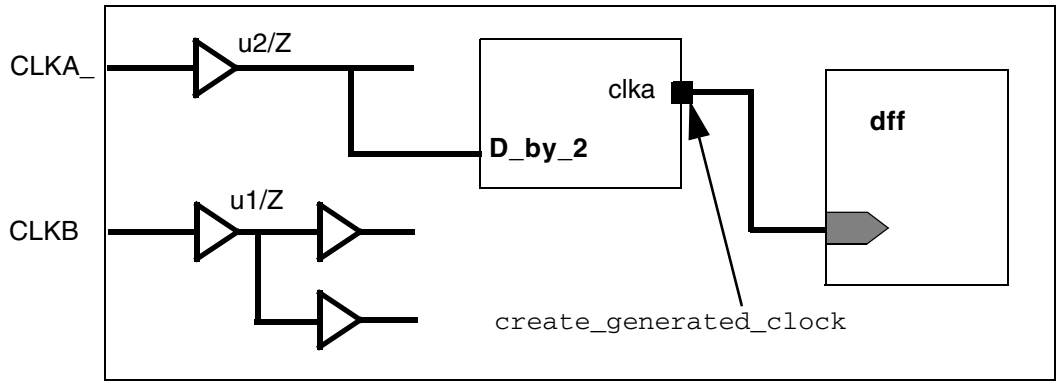
The following generates a warning because D_by_2/clka is not in the fanout of the port CLKA (the source object of the master clock):

```
create_generated_clock -source [get_pins u1/Z] [get_pins D_by_2/clka]
```


© 1999-2023 All Rights Reserved.

The correct command would be the following, because D_by_2/clka is in the fanout of CLKA.

```
create_generated_clock -source [get_pins u2/Z] [get_pins D_by_2/clka]
```



© 1999-2023
All Rights Reserved.

CCD_CLK_DEF4

Message

`The '-source master_pin' of a generated clock is not the source of another clock`

Default Severity

Warning

Description

Indicates that there is a master pin (specified with `-source`) of a generated clock that is not the source of another clock.

This rule does not point to an incorrect command, but you can use this to enforce a methodology that requires specifying master clock source objects as the generated clock's source pins.

This rule is reported even if a clock propagates to the pin specified in `-source`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

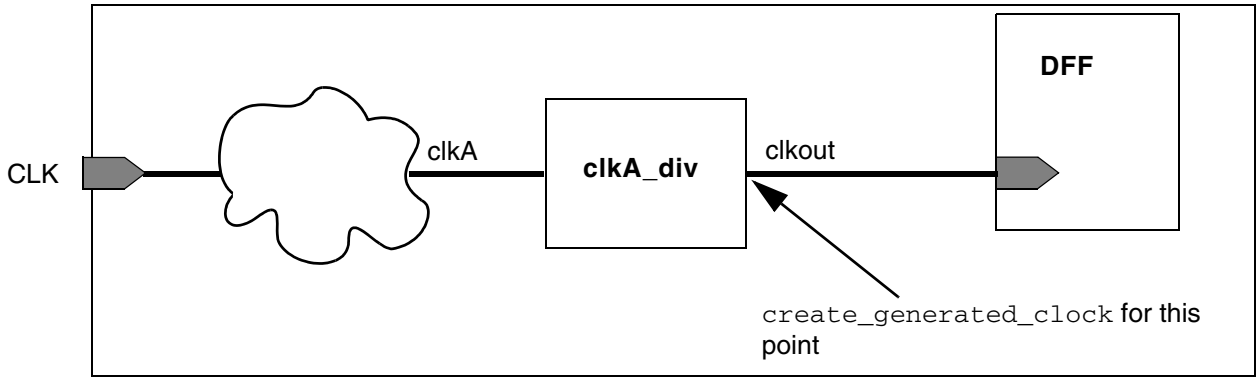
The following generates a warning as the `-source <pin>` as indicated by the `create_generated_clock` command points to `clkA` which is not the source of another clock. The `-source` should instead point to `CLK`.

```
create_clock -name MCLK [get_ports CLK]
create_generate_clock -name GCLK -source clkA -divide_by 2
    [get_pins clkA_div/clkout]
```

This rule will not be reported if the command is changed to:

```
create_generate_clock -name GCLK -source CLK -divide_by 2
    [get_pins clkA_div/clkout]
```


© 1999-2023 All Rights Reserved.



© 1999-2023
All Rights Reserved.

CCD_CLK_DEF5

Message

`A clock group has more than one root clock source`

Default Severity

Warning

Description

Indicates that you have an instance where a given clock group has more than one root clock source object. A clock group's root clock is any real clock or any generated clock whose master clock does not belong to the same group.

This rule check ensures that clock tree synthesis can align all the registers in a clock group at the block level.

If you encounter a violation for this rule check, run the `REPORT CLOCK GROUP` command to display your clock groups.

This is checked when you run the `run rule check` command.

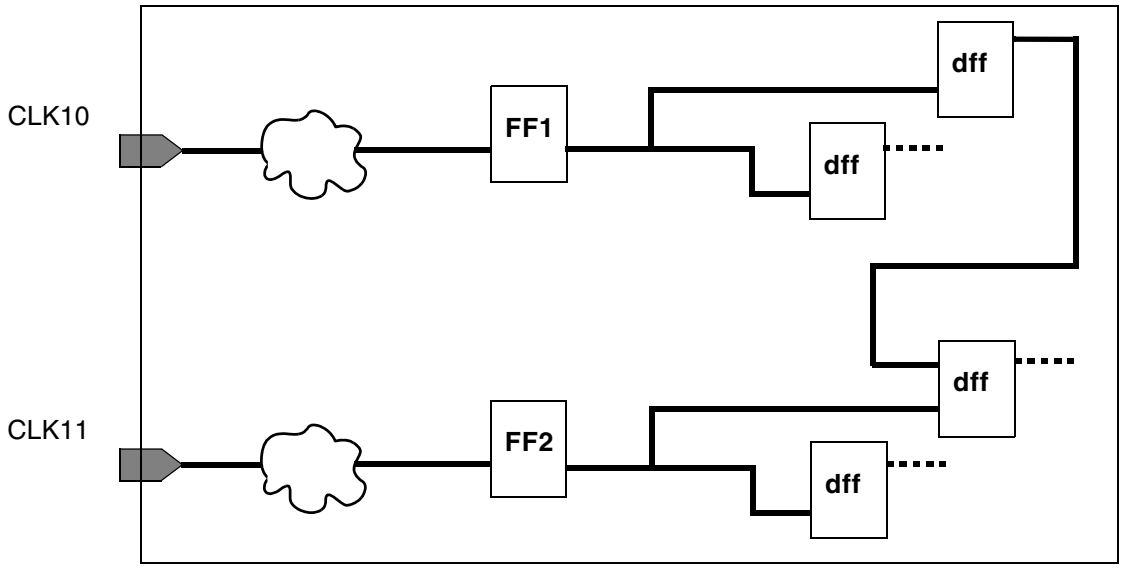
Example

The following generates this warning because a clock group has more than one root clock source:

```
create_clock -name CLK_1 -period 5.0 -waveform {0.0 2.5} [get_ports CLK10]
```


© 1999-2023 All Rights Reserved.

```
create_clock -name CLK_2 -period 5.0 -waveform {0.0 2.5} [get_ports CLK11]
```



© 1999-2023
All Rights Reserved.

CCD_CLK_DEF6

Message

`Detected overlapping clock trees`

Default Severity

Warning

Description

Indicates that two clock trees propagate to the same pin.

Note: Only clock tree overlaps on pins in the fanin of the clock pin of sequential elements are reported.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

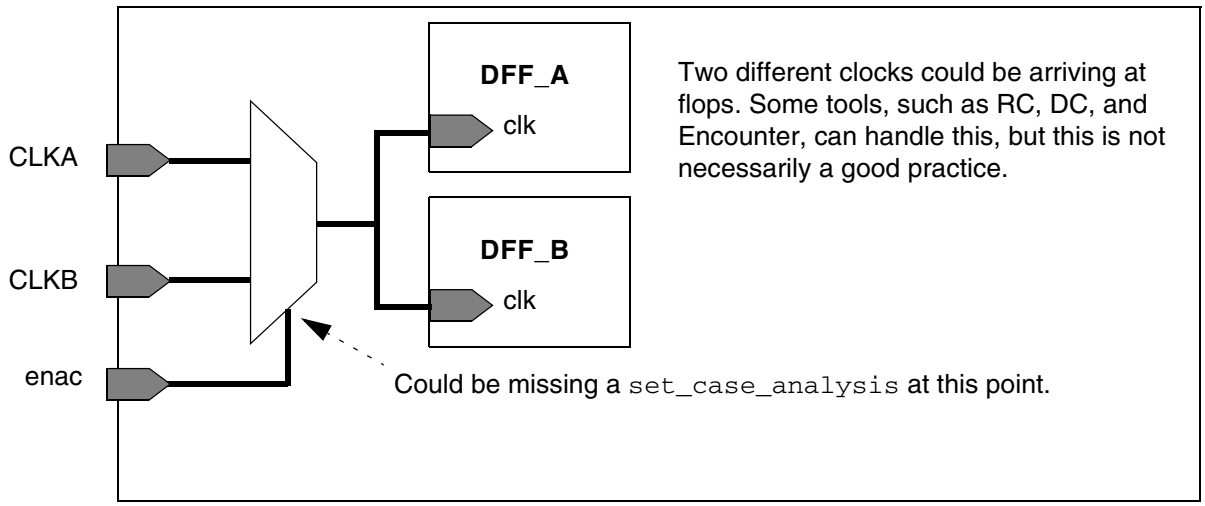
Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

The following generates a warning because the Conformal software detects overlapping clocks. A possible cause could be a missing `set_case_analysis` statement.

```
create_clock -name MCLK [get_ports CLKA]
create_clock -name TCLK [get_ports CLKB]
```


© 1999-2023 All Rights Reserved.



© 1999-2023
All Rights Reserved.

CCD_CLK_DEF7

Message

`Unbuffered clock tree nets connected to output ports`

Default Severity

Warning

Description

Indicates that there is an output port in a clock tree driven by a combinational gate with more than one fan-out. The Conformal Constraint Designer checks for every output port in a clock tree that it is driven by any combinational gate with a single fan-out.

When you do not have a buffer that connects a clock tree net and an output port, the slopes and the delays after the clock tree net have to depend heavily on the load capacitance of the port.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

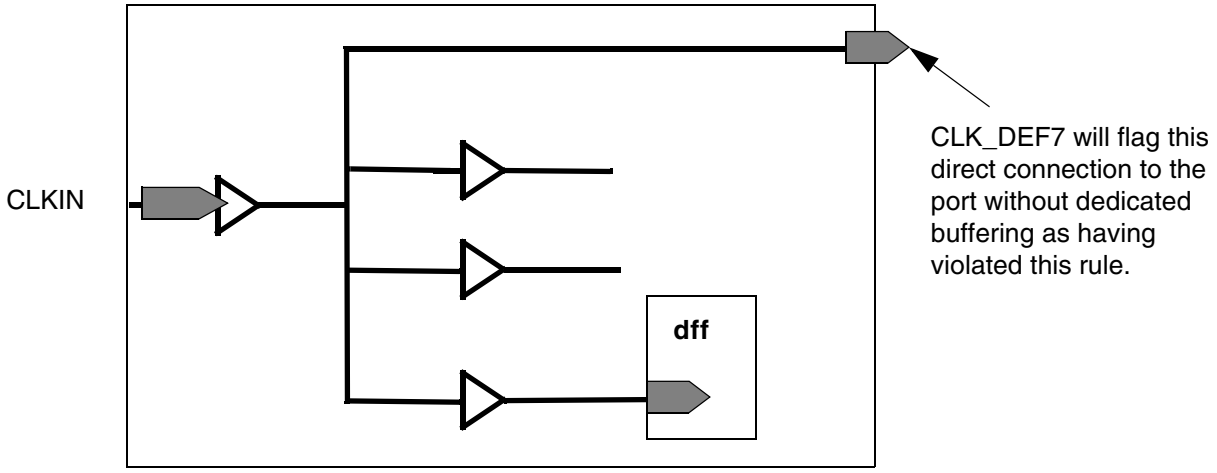
When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

© 1999-2023 All Rights Reserved.

Example

Any unbuffered direct connection to an output port will cause this warning.



© 1999-2023
All Rights Reserved.

CCD_CLK_DEF8

Message

`Source object of a generated clock is a port`

Default Severity

Warning

Description

Indicates that you have a generated clock defined on a port.

This is checked when you read in your SDC files using the `READ SDC` command.

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF9

Message

```
Source pin of a generated clock is in the fan-out of the source pin of another
     clock, but is not generated by the latter
```

Default Severity

Warning

Description

Indicates that there is a generated clock's source pin in the fan-out of the source pin of another clock that is not its master clock.

Note: A generated clock will be not be reported by this rule if it is defined in the fanout of its master clock.

When checking if a clock is defined in the fanout of another one, Conformal Constraint Designer takes into account constant propagation. Also, a clock's fanout propagation is interrupted where other clocks are defined. For example, if a clock `GCLK` is defined in the fanout of `MCLK`, then the logic in the fanout of `GCLK` is NOT considered to be in the fanout of `MCLK` unless it can be reached from `MCLK` without traversing the definition point of `GCLK`.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

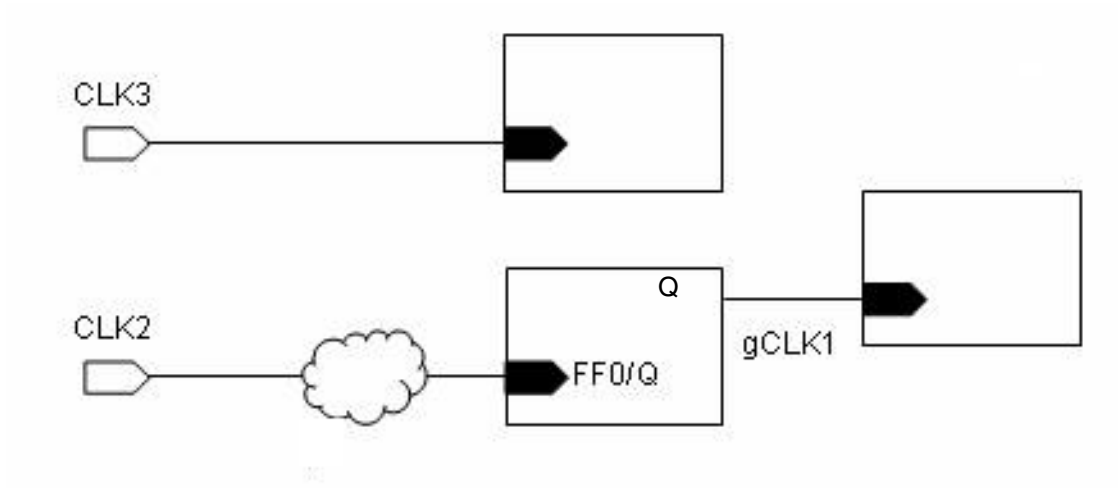
Examples

The following generates a warning because `gCLK1`'s source pin `FF0/Q` is in the fan-out of `CLK2`, but `gCLK1` was not defined as generated from that clock, and it is not in the fanout of its master clock `CLK3`.

```
create_clock -period 8.0 [get_ports CLK2]
create_clock -period 5.0 [get_ports CLK3]
```

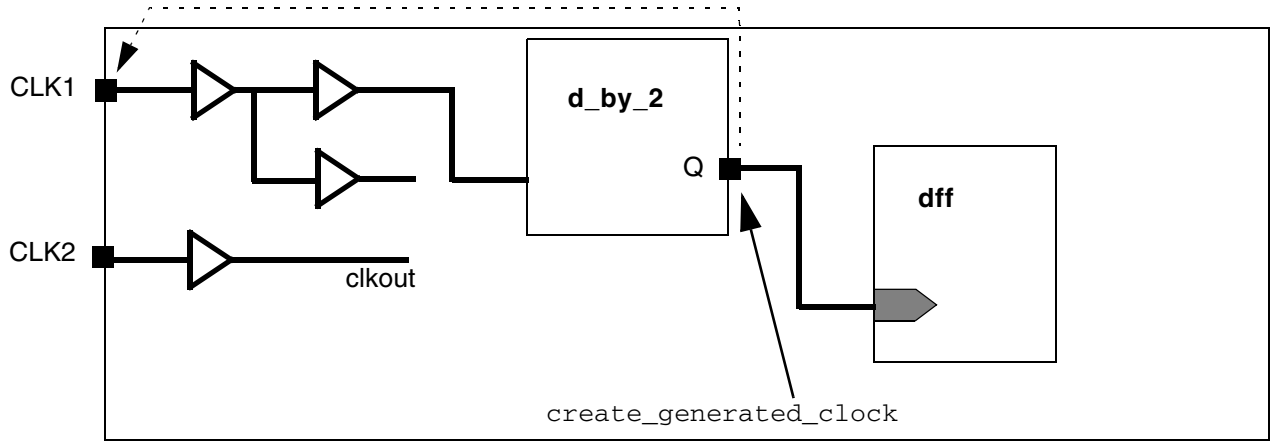

© 1999-2023 All Rights Reserved.

```
create_generated_clock -name gCLK1 -source [get_ports CLK3] \
    -divide_by 2 [get_pins FF0/Q]
```



The following is another example:

```
create_clock -name MCLK -period 5.0 [get_ports CLK1]
create_generated_clock -name GCLK -source [get_pins clkout] \
  -divide_by 2 [get_pins d_by_2/Q]  ---(Incorrect usage: flags "CCD_CLKDEF9")
create_generated_clock -name GCLK -source [get_ports CLK1] \
   -divide_by 2 [get_pins d_by_2/Q]  ------ (Correct Usage)
```



© 1999-2023
All Rights Reserved.

CCD_CLK_DEF10

Message

`Clock not propagated (missing set_propagated_clock on a created clock)`

Default Severity

Warning

Description

Indicates that there is a real or generated clock without a `set_propagated_clock` command.

This is checked when you read in your SDC files using the `READ SDC` command.

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF11

Message

`Clock (real or generated) not ideal, i.e., propagated.`

Default Severity

Warning

Description

Indicates that a real or generated clock has `set_propagated_clock` specified.

This is checked when you read in your SDC files using the `READ SDC` command.

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF12

Message

`Clock defined on logical pins (not ports or leaf cell pins)`

Default Severity

Warning

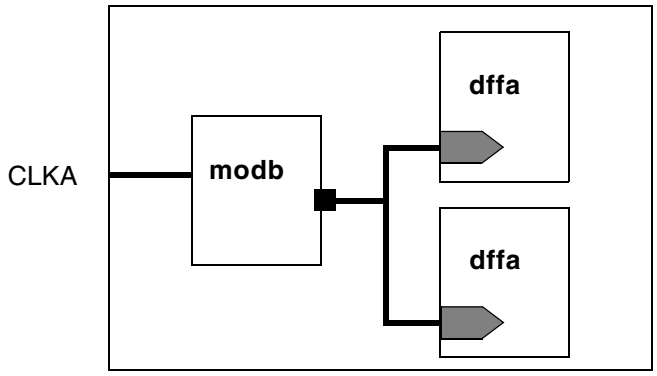
Description

Indicates that a real or generated clock is defined on hierarchical pins that might get lost after synthesis. Ports and leaf cell pins are not considered *logical pins*.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

`create_clock` is defined on the output of `modb` instead of the source CLKA. Correct usage is to define `create_clock` on `CLKA`.



© 1999-2023
All Rights Reserved.

CCD_CLK_DEF13

Message

`Virtual clock has no corresponding real clock with the same waveform and period`

Default Severity

Warning

Description

Indicates that there is a virtual clock without a corresponding real clock with the same period and waveform.

Note: This rule check applies to the block level only.

This is checked when you read in your SDC files using the `READ SDC` command. This rule is not checked for top-level SDC designs when there are blocks defined.

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF14

Message

`Incomplete clock definition: missing -waveform`

Default Severity

Warning

Description

Indicates that there is a clock (real or virtual) definition that does not have the `-waveform` argument.

This is checked when you read in your SDC files using the `READ SDC` command.

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF16

Message

`Divided/multiplied clock`

Default Severity

Note

Description

Indicates that you have a divided clock. Specifically, the Conformal Constraint Designer checks for generated clocks that were created using the `-divide_by` or `-multiply_by` options.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

For example, the Conformal Constraint Designer flags the following code because `gCLK1` is a divided clock:

```
create_generated_clock -name gCLK1 -source CLK1 -divide_by 2 [get_pins ff1/Q]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_DEF17

Message

`Edge-derived clock`

Default Severity

Note

Description

Indicates that you have a generated clock that was created with the `-edges` or `-edge_shift` options.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following command causes a rule check violation:

```
create_generated_clock -edges {1 3 7}
    -edge_shift {2 2 2} -source SYSCLK {get_pins DIVIDE}
```


© 1999-2023 All Rights Reserved.

CCD_CLK_DEF18

Message

`Unusual generated clock option (-add, -master_clock)`

Default Severity

Note

Description

Indicates that you have a `create_generated_clock` command with either the `-add` or `-master_clock` option.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

For example, the Conformal Constraint Designer flags the following command because of the `-add` option.

```
create_generated_clock -name gCLK1 -source CLK1 -divide_by 2 -add [get_pins FF1/Q]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_DEF20

Message

`Generated clock and its master clock defined asynchronous by add/delete clock group`

Default Severity

Warning

Description

Indicates that a generated clock and its master clock were defined as asynchronous using the `ADD CLOCK GROUP` or `DELETE CLOCK GROUP` command.

These are the recommended actions to take when this rule is reported:

1. Use the `REPORT CLOCK GROUP` command to display the clock groups.
2. If the clock groups are incorrect, use the `ADD CLOCK GROUP` or `DELETE CLOCK GROUP` command to define the clock groups.

This is checkedin Verify mode during the `RUN RULE CHECK` command. Prior to running the `RUN RULE CHECK` command, you must run the `COMMIT CLOCK` command (in Verify mode) for this rule to be checked.

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF21

Message

A clock does not belong to any of the user-defined clock groups

Default Severity

Warning

Description

Indicates that there is a clock that does not belong to any of the clock groups created by a ADD CLOCK GROUP command. A clock can become part of a user-defined clock group even if its name is not mentioned explicitly. Only clocks that remain unrelated to any user-defined groups will be reported.

Note: If there are no user-defined clock groups, this rule will not be reported for any clock.

These are the recommended actions to take when this rule is reported:

1. Use the REPORT CLOCK GROUP command to display the clock groups.
2. If the clock groups are incorrect, use the ADD CLOCK GROUP or DELETE CLOCK GROUP command to define the clock groups.

This is checked in Verify mode during the RUN RULE CHECK command. Prior to running the RUN RULE CHECK command, you must run the COMMIT CLOCK command (in Verify mode) for this rule to be checked.

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF22

Message

`Existing clock eliminated by the creation of a new clock with the same name and type`

Default Severity

Warning

Description

Reports existing clocks that were eliminated by the creation of a new clock with the same name and type (real, generated, or virtual).

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following commands cause a rule check violation because the second command creates a clock that eliminates the one defined in the first command:

```
create_clock -period 10 [get_ports clk]
create_clock -name clk -period 5 [get_ports {c1 c2 clk}]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_DEF23

Message

```
Clock redefined: used create_clock on an object that is in the transitive fanout
     of another clock
```

Default Severity

Warning

Description

Indicates that there is a redefined clock. This happens when you use `create_clock` on an object that is in another clock's transitive fanout.

When checking if a clock is defined in the fanout of another one, Conformal Constraint Designer takes into account constant propagation. Also, a clock's fanout propagation is interrupted where other clocks are defined. For example, if a clock `GCLK` is defined in the fanout of `MCLK`, then the logic in the fanout of `GCLK` is NOT considered to be in the fanout of `MCLK` unless it can be reached from `MCLK` without traversing the definition point of `GCLK`.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF26

Message

`Clock source is tied to a constant value`

Default Severity

Warning

Description

Indicates that a port or pin where a clock is defined has a constant logic value. Usually, such a constant value is set by a `set_case_analysis` SDC command. In some cases, clocks defined on pins inside the design can be tied to a constant value in the design, or can have a constant value propagated from another pin or port.

Both ports and pins can have locally defined constant values, as well as propagated constant values. These can be found in port/pin attributes `constant_value` and `p_constant_value`. For example:

- To find the locally defined constant value of a port and its source, use the Tcl command:

  ```
  get_attribute [find -port <port>] constant_value
  ```

- To find the propagated constant value of a pin and its source, use the Tcl command:

  ```
  get_attribute [find -pin <some/pin>] p_constant_value
  ```

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

The following commands cause this rule check violation because the second command ties the clock source defined in the first command to a constant.

```
create_clock -name clk -period 2 [get_ports clk]
set_case_analysis 0 [get_ports clk]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_DEF27

Message

`Clock source is not an input port or an output pin of a sequential element`

Default Severity

Warning

Description

Indicates that there is a clock source that is neither an input port nor an output pin of a sequential element.

This is checked when you read in your SDC files using the `READ SDC` command.

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF32

Message

`A clock tree drives a non-clock pin of a sequential cell`

Default Severity

Warning

Description

Indicates that there is a real or generated clock whose clock tree reaches a pin of a sequential cell, that is not a clock pin. This is normally considered a bad design practice.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command. This displays a path from the clock source to the reported pin.

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF33

Message

`Input/output delay set on port/pin on which a clock is defined`

Default Severity

Warning

Description

Indicates that a `set_input_delay` or `set_output_delay` has been applied to a port or pin on which a clock is defined.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following commands will cause this rule to be reported for port `CLK`:

```
set_input_delay 1 [all_inputs]
create_clock -period 10 [get_ports CLK]
```

The following commands will cause this rule to be reported for pin `B/OUT`:

```
create_generated_clock -source [get_pins B/MCLK] -divide_by 2 [get_pins B/GCLK]
set_output_delay 2 [get_pins B/GCLK].
```


© 1999-2023 All Rights Reserved.

CCD_CLK_DEF34

Message

`Clock tree propagates to output port`

Default Severity

Warning

Description

Indicates that there is an output port connected to a clock tree net. Output ports that are defined as source objects of clocks are not reported by this rule.

Note: See also SDC rule CCD_CLK_DEF7.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF35

Message

`Clock tree XOR gate has non-clock input that is not tied to a constant value`

Default Severity

Warning

Description

Indicates that one or more clock trees propagate to an exclusive-or (XOR) gate and that gate has a side input that is not on any clock's tree and is not set to a constant value, directly or through constant propagation.

This is not always a problem in the SDC or the design. It can be due to a missing `set_case_analysis` propagating to the side input being reported (use the `REPORT RULE CHECK -verbose` command to get more details.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

If the RTL has the assignment:

```
assign clk_or_clkbar = clk ^ negate_clk;
```

And the SDC file has this command:

```
create_clock -name CLK -period 10 [get_ports clk]
```

but not this command:

```
set_case_analysis 1 [get_ports negate_clk]
```

Then the XOR gate will be reported by this rule.

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF36

Message

`Slower clock selected in functional mode`

Default Severity

Warning

Description

Indicates that a clock tree propagates to a cell where its propagation is interrupted in favor of another clock with a bigger period. This check is performed only when the values of the `SDC_MODE` and `SDC_SCAN_SHIFT_MODE` CCD parameters are different, which means that the current mode is a functional one.

This rule is mainly intended to report `set_case_analysis` selecting the wrong clock at a multiplexer, a problem that would otherwise be only detected much later in the flow.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

If the RTL has this instance:

```
MYMUX mux_i(.Z(theclk), .I0(funclk), .I1(scanclk), .SEL(scan_mode));
```

And the SDC has the following commands:

```
create_clock -period 5 [get_ports funclk]
create_clock -period 20 [get_ports scanclk]
set_case_analysis 1 [get_ports scan_mode]
```

Then then this rule will report that `scanclk` is selected over `funclk` in `mux_i` if the `SDC_MODE` CCD parameter has not been assigned the same value as `SDC_SCAN_SHIFT_MODE`.

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF37

Message

`Multiple clocks are defined on the same pin`

Default Severity

Warning

Description

Indicates that more than one real or generated clock was defined on the same pin, using the `-add` option of the `create_clock` or `create_generated_clock` commands. In the verbose report, the clocks are listed and are annotated as "active" or "not active."

In clock waveform modeling for sequential validation, the inactive clocks are ignored.

By default, a clock being defined on a pin that is already a clock source, is marked "inactive." It is possible to define certain clocks as active by setting the CCD parameter `SDC_ACTIVE_CLOCKS`.

Note: This rule does not necessarily point to a problem in the constraints, so it can be safely disabled.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following commands will cause this rule to be reported for port `CLK`:

```
create_clock -period 10 -name fastclk [get_ports CLK]
create_clock -period 20 -name slowclk [get_ports CLK] -add
```

In this case, `fastclk` will be reported as "active" and `slowclk` as "NOT active."

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF38

Message

`set_disable_timing on clock tree`

Default Severity

Warning

Description

Indicates that a `set_disable_timing` command has been specified on a clock tree.

Note: This rule flags any set disable timing on clock tree, even if it does not affect any clock pin of a sequential element or if the object on the clock tree is in the fanout of another object in the same clock tree with set disable timing on it

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

If the constraint file contains the following command:

```
create_clock -name CLK1 -period 5000 -waveform {0 2500} [get_ports clk]
```

then this rule would report a warning for the following SDC command:

```
set_disable_timing [get_ports clk]
```

or for any object on the clock tree of `CLK1` with set disable timing.

© 1999-2023 All Rights Reserved.

CCD_CLK_DEF39

Message

`Generated clock is incorrectly defined.`

Default Severity

Warning

Description

Indicates that a generated clock is incorrectly defined. Tool performs the following structural checks on the source latency paths (from `-source` to `source_object`) of a generated clock to ensure a correct definition.

- A generated clock defined with `-divide_by 1`, `-multiply_by 1` or `-combinational` should have at least one source latency path that is a combinational path.
- A generated clock defined with `-divide_by 2` should have at least one source latency path passing through a flop.
- A generated clock defined with `-divide_by 1`, `-multiply_by 1`, `-combinational` or `-edges` should have at least one source latency path conforming to an edge-to-edge relationship between master clock source (`-source`) and generated clock source (`source_object`).

This rule is checked when running rule check with `run_rule_check` command in Verify mode and can only be checked after committing clocks with `commit_clock` command.

This rule is disabled by default. Enable using the `add_rule_instance` command.

Review generated clock definition and source latency path. Correct `create_generated_clock` timing constraint or design structure.

Example

In the case input port 'clk' is directly connected to output ports 'gclk0' and gclk1'. This case reports this rule because:

© 1999-2023 All Rights Reserved.

- The source latency path of generated clock gclk0, from input port 'clk' to output port 'gclk0', does not pass through any flop.
- The source latency path of generated clock gclk1, from 'rise' edge at input port 'clk' to 'fall' edge at output port 'gclk1', is not found.

```
create_clock -name clk -period 2 [get_ports clk]
create_generated_clock -name gclk0 -source [get_ports clk] -divide_by 2 \
[get_ports gclk0]
create_generated_clock -name gclk1 -source [get_ports clk] -edges {1 3 5} \
[get_ports gclk1]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_DEF40

Message

`Generated clock source latency path is not found.`

Default Severity

Warning

Description

This rule indicates that the source latency path of a generated clock is not found. The following are possible causes of this rule violation.

- A source latency path does not exist.
- A source latency path is broken by a real clock (through the `create_clock` command).
- A source latency path is broken by disabled timing arc (through the `set_disable_timing` command).
- A source latency path is broken by a constant (through the `set_case_analysis` command or a hard constant).
- A source latency path is broken by a clock stop (through the `set_clock_sense` command or the `set_sense -stop_propagation` command).

This rule is also a complement to CCD_CLK_DEF39 rule, which checks the source latency path.

This rule is checked when running rule check with `run_rule_check` command in Verify mode and can only be checked after committing clocks with `commit_clock` command.

This rule is disabled by default. Eable using the `add_rule_instance` command.

Review the generated clock definition and source latency path. Correct the `create_generated_clock` timing constraint or design structure.

© 1999-2023 All Rights Reserved.

Example

In the following case there are no connections between input port 'clk' and output port 'gclk'. This rule is reported because the source latency path of generated clock 'gclk' does not exist.

```
create_clock -name clk -period 2 [get_ports clk]
create_generated_clock -name gclk -source [get_ports clk] -divide_by 2 [get_ports
gclk]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_GRP*

This section describes rule checks that relate to clock group violations. To review current clock groupings, use the `REPORT CLOCK GROUP` command.

© 1999-2023 All Rights Reserved.

CCD_CLK_GRP1

Message

```
Asynchronous or physically exclusive clock group is defined among master clock and
     its generated clocks.
```

Default severity

Warning

Description

Master clock and its generated clocks are usually considered to be correlated. This rule indicates that an asynchronous or physically exclusive clock group is defined among a master clock and its generated clocks, making them no longer correlated.

Review the clock groups using the `REPORT CLOCK GROUP` command. Correct the clock groups of the '`set_clock_groups`' timing constraint command, by removing any extra clock group definitions.

This rule is disabled by default. To enable, use the `ADD RULE INSTANCE` command. It is run when the `RULE RULE CHECK` command is used in Verify mode and can only be checked after the `COMMIT CLOCK` command.

Example

The following case reports this rule because generated clock 'gclk1' is asynchronous to master clock 'clk' and correlated generated clock 'gclk0'.

```
create_clock -name clk -period 2 [get_ports clk]
create_generated_clock -name gclk0 -source [get_ports clk] -divide_by 1 [get_ports
gclk0]
create_generated_clock -name gclk1 -source [get_ports clk] -divide_by 1 [get_ports
gclk1]
set_clock_groups -name async_cg0 -asynchronous -group [get_clocks "clk gclk0"] -
group [get_clocks gclk1]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_GRP2

Message

```
Asynchronous or physically exclusive clock group is not defined for the groups of
     clocks in the transitive fanout of asynchronous or physically exclusive source
     clocks.
```

Default severity

Warning

Description

Generated clocks in the transitive fanout of uncorrelated source clocks are usually considered to be uncorrelated as well. This rule indicates that a clock group inconsistency is found between generated clocks in the transitive fanout of asynchronous or physically exclusive source clocks.

Review the clock groups using the `REPORT CLOCK GROUP` command. Correct the clock groups of the 'set_clock_groups' timing constraint command, by adding any missing clock group definitions.

This rule is disabled by default. To enable, use the `ADD RULE INSTANCE` command. It is run when the `RULE RULE CHECK` command is used in Verify mode and can only be checked after the `COMMIT CLOCK` command.

Example

The following case reports this rule because source clock 'clk0' and source clock 'clk1' are asynchronous, but their generated clock 'gclk0' and 'gclk1' are synchronous.

```
create_clock -name clk0 -period 2 [get_ports clk0]
create_clock -name clk1 -period 2 [get_ports clk1]
create_generated_clock -name gclk0 -source [get_ports clk0] -divide_by 1 [get_ports
gclk0]
create_generated_clock -name gclk1 -source [get_ports clk1] -divide_by 1 [get_ports
gclk1]
set_clock_groups -name async_cg0 -asynchronous -group [get_clocks clk0] -group
[get_clocks clk1]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_GRP3

Message

`The harmonic cycle of a set of correlated clocks exceeds the threshold`

Default Severity

Warning

Description

Indicates the number of harmonic cycles of a set of correlated clocks exceeds the threshold. The number of cycles is calculated based on the clock with the highest frequency amongst its correlated clocks. The threshold can be changed by the `set_clkgrp_harmony_threshold` command, where the default is 100 cycles.

Note: This check will be run only after '`COMMIT CLOCK`' is run and is checked when you run the `RUN RULE CHECK` command.

Example

If there are three clocks in the same clock group specified by the following SDC statements:

```
create_clock -name clk1 -period 5 clk1
create_clock -name clk2 -period 15 clk2
create_clock -name gclk -source CK -master_clock clk2 -divide_by 2 FF1/Q
set_clock_groups -name grp1 -asynchronous -group {clk1 clk2 gclk}
```

The periods of clk1, clk2, and gclk are 5, 15, and 30 respectively. So the harmonic cycle will be 6 because the LCM of their periods is 30 and it is 6 cycles of clk1, which is the clock with highest frequency among correlated clocks. Therefore, this number of cycles, 6, will be checked against the threshold.

© 1999-2023 All Rights Reserved.

CCD_CLK_GRP5

Message

`Generated clocks are asynchronous but their master clocks are synchronous.`

Default Severity

Warning

Description

Generated clocks of synchronous master clocks are usually synchronous as well. This rule indicates that a clock group inconsistency is found between master clocks and their generated clocks.

This rule is checked when running rule check with '`run_rule_check`' command in Verify mode and can only be checked after committing clocks with the '`commit_clock`' command.

Review clock groups using the '`report_clock_group`' command and correct clock groups using the '`set_clock_groups`' timing constraint by adding missing clock group definitions.

This rule is disabled by default. Use '`add_rule_instance`' command to enable.

Example

The following example results in this warning because master clock 'clk0' and master clock 'clk1' are synchronous, but their generated clock 'gclk0' and 'gclk1' are asynchronous.

```
create_clock -name clk0 -period 2 [get_ports clk0]
create_clock -name clk1 -period 2 [get_ports clk1]
create_generated_clock -name gclk0 -source [get_ports clk0] -divide_by 1 \
 [get_ports gclk0]
create_generated_clock -name gclk1 -source [get_ports clk1] -divide_by 1 \
 [get_ports gclk1]
set_clock_groups -name async_cg0 -asynchronous -group [get_clocks gclk0] -group
[get_clocks gclk1]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_LAT*

This section describes rule checks that relate to clock latency violations. *Source clock latency* is the time it takes a clock signal to propagate from its the actual waveform origin point to the clock-definition point in the design.

© 1999-2023 All Rights Reserved.

CCD_CLK_LAT1

Message

`Undefined clock latency for real clocks`

Default Severity

Warning

Description

Indicates that a real clock has undefined clock latency. *Clock latency* is the time it takes a clock signal to propagate from the clock definition to the register clock pin.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

1. If you have this line: `set_clock_latency -source 0.5 CLK3`
   But, you do not have this line: `set_clock_latency <value> CLK3`

   the Conformal Constraint Designer issues a rule check violation if you created `CLK3` using `create_clock` and `CLK3` is not a virtual clock.

2. You have this line: `set_clock_latency 1.0 CLK3`

   The Conformal Constraint Designer does not issue a rule check violation if `CLK3` is a virtual clock.

© 1999-2023 All Rights Reserved.

CCD_CLK_LAT2

Message

`Clock latency is set on an object that is not a clock (pre-layout)`

Default Severity

Warning

Description

Indicates that clock latency is set on an object that is not a clock (such as a port or a pin).

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following code causes a violation because clock latency should be set on `CLK2_w`—not `CLK2`.

```
create_clock -name CLK2_w -period 10 CLK2
set_clock_latency 2.0 -source CLK2
```

The following is the correct version:

```
set_clock_latency 2.0 -source [get_clocks CLK2_w]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_LAT3

Message

```
Source latency for a generated clock is less than or equal to the source latency
     of its master clock
```

Default Severity

Warning

Description

Indicates that the source latency on a generated clock is less than or equal to the latency of its master clock.

Note: The Conformal Constraint Designer checks every combination of `-rise/-fall`, `-min/-max`, and `-early/-late` separately.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

For example, you have the following lines:

```
create_clock -period <value> clk2
create_generated_clock -name gCLK1 -source clk2 -divide_by 2 [get_pins FF1/Q]
create_generated_clock -name gCLK2 -source [get_pins FF1/Q] -divide_by 2 \
       [get_pins FF2/Q]
set_clock_latency <value0> -source clk2
set_clock_latency <value1> -source gCLK1
set_clock_latency <value2> -source gCLK2
```

The Conformal Constraint Designer issues a rule check violation if:

- `clk2` generates `gCLK1`, and `value0` >= `value1`, or
- `gCLK1` generates `gCLK2`, and `value1 >= value2`

© 1999-2023 All Rights Reserved.

CCD_CLK_LAT4

Message

`Clocks in the same clock group have different (latency + source latency)`

Default Severity

Warning

Description

Indicates that clocks in the same group have different values for:

`latency` + *`source_latency`*

This ensures that all the register clock pins in a group can see the simultaneous arrival of the clock signals.

Note: For `set_clock_latency`, the Conformal Constraint Designer checks every combination of `-rise/-fall`, `-min/-max`, and `-early/-late` separately.

This is checked when you run the `run rule check` command.

Example

For example, you have the following lines:

```
create_clock -period 10 clk2
create_generated_clock -name gCLK1 -source clk2 -divide_by 2 [get_pins FF1/Q]
create_generated_clock -name gCLK2 -source [get_pins FF1/Q] -divide_by 2 \
     [get_pins FF2/Q]
set_clock_latency 2.5 clk2
set_clock_latency 2.0 gCLK1
set_clock_latency 0.5 -source gCLK1
set_clock_latency 1.5 gCLK2
set_clock_latency 1.0 -source gCLK2
```

In this example there is one group that contains three clocks: {`clk2, gCLK1, gCLK2`}

This does not cause a rule check violation because the sum of *`latency`* + *`source_latency`* is always 2.5:

```
clk2  = 2.5
gCLK1 = 2.0+0.5
```


© 1999-2023 All Rights Reserved.

```
gCLK2 = 1.5+1.0
```


© 1999-2023
All Rights Reserved.

CCD_CLK_LAT5

Message

```
Detected set_clock_latency (in post-layout stage)
```

Default Severity

Warning

Description

Indicates that there is clock network latency (`set_clock_latency` without `-source`) defined on a non-virtual clock, during post-layout

This is checked when you read in your SDC files using the `READ SDC` command.

Example

In the following example, the Conformal Constraint Designer issues a warning because the clock should not have latency in the post-layout stage:

```
set_clock_latency 1.5 -rise CLK1
```

The following commands will not be reported since they specify source latency or latency on virtual clock:

```
set_clock_latency 2.0 -source CLK2
set_clock_latency 1.0 -min CLK2_vir
```


© 1999-2023 All Rights Reserved.

CCD_CLK_LAT6

Message

`Undefined source clock latency for generated clocks`

Default Severity

Warning

Description

Indicates that there is a generated clock with undefined source clock latency.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

For example, you have the following lines:

```
create_clock -period 10 clk2
create_generated_clock -name gCLK1 -source clk2 -divide_by 2 [get_pins FF1/Q]
create_generated_clock -name gCLK2 -source [get_pins FF1/Q] -divide_by 2 \
   [get_pins FF2/Q]
set_clock_latency 2.5 clk2
set_clock_latency 2.0 gCLK1
set_clock_latency 1.5 gCLK2
set_clock_latency 1.0 -source gCLK2
```

This causes a rule check violation because gCLK1 does not have `-source` defined.

© 1999-2023 All Rights Reserved.

CCD_CLK_LAT7

Message

`Incomplete clock latency options`

Default Severity

Warning

Description

Indicates that you have incomplete clock latency options. Where `set_clock_latency` needs to have both or none of the options in these pairs: `-rise/-fall`, `-min/-max`, and `-early/-late`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following commands cause a rule check violation because the clock latency for `-rise` and `-late` are missing.

```
set_clock_latency -rise -early 1.3 -source gCLK1
set_clock_latency -fall 1.5  -source gCLK1
```


© 1999-2023 All Rights Reserved.

CCD_CLK_LAT8

Message

`Inconsistent clock latency option values`

Default Severity

Warning

Description

Indicates that there are inconsistent clock latency option values. The Conformal Constraint Designer checks `set_clock_latency` for instances where `min > max` and `early > late`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following causes a rule check violation because `-min > -max` for `CLK1`

```
set_clock_latency 1.5 -max CLK1
set_clock_latency  2.0 -min CLK1
```


© 1999-2023 All Rights Reserved.

CCD_CLK_LAT9

Message

`Negative clock latency value`

Default Severity

Warning

Description

Indicates that there is a negative value for `set_clock_latency`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following command causes a violation because `-min` is set to -2.0:

```
set_clock_latency -2.0 -min CLK1
```


© 1999-2023 All Rights Reserved.

CCD_CLK_LAT10

Message

```
In set_clock_latency, -clock does not affect clocks in the object_list
```

Default Severity

Warning

Description

The `-clock` option is used to specify latencies on objects that are not clocks, when applying different latency values to paths that are constrained by different clocks. When the latency is specified on a clock object, the `-clock` option has no effect.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The Conformal Constraint Designer issues a warning for lines such as:

```
set_clock_latency 2.0 -clock CK1 -rise [get_clocks CK2]
set_clock_latency 1.0 -clock CK2 [list [get_clocks CK3] [get_ports clk4]]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_LAT11

Message

```
A port/pin with set_clock_latency defined for a specific clock is not in that
     clock's tree
```

Default Severity

Warning

Description

The `-clock` option is used to specify latencies on objects that are not clocks, when applying different latency values to paths that are constrained by different clocks. It is expected that if the latency is applied to a port/pin, using the `-clock` option, then the specified clock's tree should propagate to that port/pin.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

The Conformal Constraint Designer issues a warning if pin `clkgate/Z` is not in the tree of clock CK1 and this command is used:

```
set_clock_latency 2.0 -clock CK1 [get_pins {clkgate/Z}]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_LAT12

Message

`Undefined source clock latency for virtual clock`

Default Severity

Warning

Description

Indicates that a virtual clock does not have any source latency defined. Source clock latency is defined with the `set_clock_latency -source` command.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following lines cause a rule check violation for `vclk3`:

```
create_clock -period 10 -name vclk1
create_clock -period 20 -name vclk2
create_clock -period 30 -name vclk3
set_clock_latency 2.5 -source vclk1
set_clock_latency 0.0 -source vclk2
#set_clock_latency 1.0 -source vclk3
```


© 1999-2023 All Rights Reserved.

CCD_CLK_UNC*

This section describes rule checks for violations that relate to clock uncertainty.

© 1999-2023 All Rights Reserved.

CCD_CLK_UNC1

Message

`Undefined clock uncertainty`

Default Severity

Warning

Description

Indicates that a real or generated clock has an undefined clock uncertainty.

Note: This checks only per-clock uncertainty.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

In the following example, the Conformal Constraint Designer issues a warning if a command like the following is missing for clock `CLK1`:

```
set_clock_uncertainty 1.0 [get_clocks CLK1]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_UNC2

Message

`Clock uncertainty is set for an object that is not a clock`

Default Severity

Warning

Description

Indicates that clock uncertainty is set for an object that is not a clock.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

In the following example, the Conformal Constraint Designer issues a warning because `IN1` is not a clock:

```
set_clock_uncertainty 2.5 -setup IN1
```


© 1999-2023 All Rights Reserved.

CCD_CLK_UNC3

Message

`Inconsistent clock uncertainty vs. clock period`

Default Severity

Warning

Description

Indicates that a clock uncertainty value is bigger than the clock period of the relevant clock. For inter-clock uncertainty, the uncertainty value is compared with the clock period of the `-to` clock.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

The following commands will cause this rule to be reported twice:

```
create_clock -period  5 -name clk11 [get_ports clk1]
create_clock -period 10 -name clk22 [get_ports clk2]
set_clock_uncertainty 8 [get_clocks clk11]
set_clock_uncertainty 9 -from [get_clocks clk22] -to [get_clocks clk11]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_UNC4

Message

`Incomplete clock uncertainty options for clock objects`

Default Severity

Warning

Description

Indicates that there is a clock object with incomplete clock-uncertainty options. You must have both or none of the options for the following pairs: `-setup/-hold` and `-rise/-fall` (for inter-clocks).

This is checked when you read in your SDC files using the `READ SDC` command.

Example

In the following example, the Conformal Constraint Designer issues a warning because the `-hold` uncertainty for `CLK1` is missing:

```
set_clock_uncertainty 1.5 -setup CLK1
```

In the following example, the `-fall` inter-clock uncertainty from `CLK2` to `CLK3` is missing:

```
set_clock_uncertainty 2.0 -rise -from CLK2 -to CLK3
```


© 1999-2023 All Rights Reserved.

CCD_CLK_UNC5

Message

`Negative clock uncertainty value`

Default Severity

Warning

Description

Indicates that there is a negative value for clock uncertainty.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following example generates a warning because it specifies a negative clock uncertainty:

```
set_clock_uncertainty -1.5 -setup CLK1
```


© 1999-2023 All Rights Reserved.

CCD_CLK_UNC6

Message

```
Inter-clock uncertainty not defined between synchronous clocks that drive connected
     sequential elements
```

Default Severity

Warning

Description

Indicates that you did not define inter-clock uncertainty between synchronous clocks (real or generated) that drive connected, sequential elements.

This rule will not report missing inter-clock uncertainties to a clock that has *simple* clock uncertainty defined because it affects all the paths ending in that clock. However, if a clock has no uncertainties defined, only `CCD_CLK_UNC1` will be reported to avoid repetition. In summary, this rule reports only clocks that do not have *simple* uncertainty defined, and have inter-clock uncertainty defined from at least one clock.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

© 1999-2023 All Rights Reserved.

CCD_CLK_UNC7

Message

`Inter-clock uncertainty defined between clocks that are not connected`

Default Severity

Warning

Description

Indicates that for two clocks (for example, `CLK1` and `CLK2`), there is a command:

```
set_clock_uncertainty -from CLK1 -to CLK2 ...
```

but there is not path from clock groups containing `CLK1` to the clock groups containing `CLK3`.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

© 1999-2023 All Rights Reserved.

CCD_CLK_UNC9

Message

```
Inter-clock uncertainty not defined between a virtual clock and a connected non-
     virtual clock with the same waveform and period
```

Default Severity

Warning

Description

Indicates that you did not define inter-clock uncertainty between a virtual clock and a non-virtual clock with the same period and waveform, that are connected (constrain a timing path). This rule will not report missing inter-clock uncertainties to a clock that has simple clock uncertainty defined because it affects all the paths ending in that clock.

Note: Clock pairs that are not related (for example, asynchronous clocks) will not be reported.

This is checked when you run the `run rule check` command and can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

If there is a path from `DATA_IN` to `MY_REG/D`, the clock tree of `RCLK` reaches `MY_REG/CK`, and virtual clock `VCLK` and real clock `RCLK` are synchronous, then this rule will be reported if the following SDC commands are given:

```
create_clock -name VCLK -period 10.0 -waveform {0.0 5.0}
create_clock -name RCLK -period 10.0 -waveform {0.0 5.0} [get_ports clk]
set_input_delay -clock VCLK 1.0 [get_ports DATA_IN]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_UNC10

Message

Inter-clock uncertainty is defined between asynchronous or mutually exclusive clocks.

Default severity

Warning

Description

The paths between asynchronous or mutually exclusive clocks are not considered during timing analysis (that is, they are considered false paths). This rule indicates that unnecessary interclock uncertainty is defined between asynchronous or mutually exclusive clocks. Remove the inter-clock uncertainty defined between the aschronous or mutually exclusive clocks to improve the timing constraint quality.

This rule is disabled by default. To enable, use the `ADD RULE INSTANCE` command. It is run when the `RULE RULE CHECK` command is used in Verify mode and can only be checked after the `COMMIT CLOCK` command.

Example

The case reports this rule because inter-clock uncertainty is defined from clock 'clk0' to clock 'clk1', which are asynchronous.

```
create_clock -name clk0 -period 2 [get_ports clk0]
create_clock -name clk1 -period 3 [get_ports clk1]
set_clock_groups -name async_cg0 -asynchronous -group [get_clocks clk0] -group
[get_clocks clk1]
set_clock_uncertainty 1 -from [get_clocks clk0] -to [get_clocks clk1]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_CTR*

This section describes rule checks that relate to clock transitions.

© 1999-2023 All Rights Reserved.

CCD_CLK_CTR1

Message

`Undefined clock transition`

Default Severity

Warning

Description

Indicates that you did not define `set_clock_transition` for a real or generated clock.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

If you have three clocks `(CLK1, sysCLK, Hclk)` and the following line:

```
set_clock_transition  0.75 -rise  CLK1
```

the Conformal Constraint Designer issues a violation because you did not define a clock transition for `sysClk` and `Hclk`.

© 1999-2023 All Rights Reserved.

CCD_CLK_CTR2

Message

`Clock transition is set on a virtual clock`

Default Severity

Warning

Description

Indicates that you have `set_clock_transition` set on a virtual clock.

Note: `SDC_LINT_REF4` reports objects of the wrong type.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The Conformal Constraint Designer issues a violation because the clock transition is set on a virtual clock.

```
create_clock -period 10 -name CLK1_w
set_clock_transition  0.75 -rise CLK1_w
```


© 1999-2023 All Rights Reserved.

CCD_CLK_CTR3

Message

`Clock transition value outside the characterization range (pre-layout)`

Default Severity

Warning

Description

Indicates that the value for `set_clock_transition` is outside the characterization range. If the ports of the clock specified as an argument to `set_clock_transition` drive a pin of a liberty cell with `min_transition` or `max_transition` defined, the characterization range is defined by these values. If not, the characterization range is defined by the `SDC_MIN_CLK_TRANSITION` and `SDC_MAX_CLK_TRANSITION` parameters.

For a list of CCD parameters and definitions, see `SET CCD PARAMETER` in the *Conformal Constraint Designer Reference Manual*.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

If you set the `SDC_MIN_CLK_TRANSITION` parameter to `0.1` and the `SDC_MAX_CLK_TRANSITION` parameter to `0.5`, then the following command causes a warning because 0.75 is outside the characterization range.

```
set_clock_transition 0.75 -min CLK1
```


© 1999-2023 All Rights Reserved.

CCD_CLK_CTR4

Message

`set_annotated_transition not defined on the clock pin of a leaf cell`

Default Severity

Warning

Description

Indicates that a pin of a liberty cell that has the clock attribute, does not have a transition defined using the `set_annotated_transition` command.

Note: The `set_annotated_transition` command is not included in SDC versions 1.6 or earlier.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The Conformal Constraint Designer issues a violation of this rule if there is an instance (`a_reg`) of a sequential cell whose clock pin is called `CP`, and the constraints file has no command like the following:

```
set_annotated_transition 0.1 [get_pins {a_reg/CP}]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_CTR5

Message

`set_annotated_transition outside the characterization range`

Default Severity

Warning

Description

Indicates that the value for `set_annotated_transition` is outside the characterization range. The characterization range is defined by the `SDC_MIN_CLK_TRANSITION` and `SDC_MAX_CLK_TRANSITION` parameters.

For a list of CCD parameters and definitions, see `SET CCD PARAMETER` in the *Conformal Constraint Designer Reference Manual*.

This is checked when you read in your SDC files using the `READ SDC` command.

© 1999-2023 All Rights Reserved.

CCD_CLK_CTR6

Message

```
Undefined clock transition on real clock: Source input port without
     set_input_transition
```

Default Severity

Warning

Description

Indicates that a real clock has an undefined clock transition. That is, PI/POs that are the source of a clock do not have a defined `set_input_transition`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The Conformal Constraint Designer issues a violation if the second line is missing, where `CLK1` is the port name—not the clock name:

```
create_clock -period 10 -name CLK1_w CLK1
#set_input_transition  0.75 CLK1
```


© 1999-2023 All Rights Reserved.

CCD_CLK_CTR7

Message

```
Clock transition value outside the characterization range (post-layout)
```

Default Severity

Warning

Description

Indicates that the value for `set_input_transition` is outside the characterization range. If the port specified as argument to `set_input_transition` drives a pin of a liberty cell with `min_transition` or `max_transition` defined, the characterization range is defined by these values. If not, the characterization range is defined by the `SDC_MIN_CLK_TRANSITION` and `SDC_MAX_CLK_TRANSITION` parameters.

For a list of CCD parameters and definitions, see `SET CCD PARAMETER` in the *Conformal Constraint Designer Reference Manual*.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

If you have the following commands in your dofile:

```
set ccd parameter SDC_MIN_CLK_TRANSITION 0
set ccd parameter SDC_MAX_CLK_TRANSITION 0.50
```

then, the following SDC command generates a warning because the clock transition time is not within the specified range:

```
set_input_transition 0.75 -min CLK1
```


© 1999-2023 All Rights Reserved.

CCD_CLK_CTR8

Message

`set_clock_transition is set on a clock in post-layout`

Default Severity

Warning

Description

Indicates that `set_clock_transition` is set on a clock in post-layout.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following generates an error only when in post-layout:

```
set_clock_transition 0.75 -rise CLK1
```


© 1999-2023 All Rights Reserved.

CCD_CLK_CTR9

Message

```
set_input_transition and set_driving_cell on clock ports are not recommended in
     pre-layout
```

Default Severity

Warning

Description

Indicates that you have `set_input_transition` or `set_driving_cell` on a clock port. These options are not recommended in pre-layout.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following command causes a rule check violation because `set_input_transition` on a clock port.

```
create_clock -name clk1 -period 10 [get_ports clock]
set_input_transition 1 [get_ports clock]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_CTR10

Message

`set_driving_cell on clock ports is not recommended in post-layout`

Default Severity

Warning

Description

Indicates that you have `set_driving_cell` defined on a clock port. This is not recommended in post-layout.

This is checked when you read in your SDC files using the `READ SDC` command.

© 1999-2023 All Rights Reserved.

CCD_CLK_CTR11

Message

`Incomplete clock transition options (pre-layout)`

Default Severity

Warning

Description

Indicates that `set_clock_transition` has incomplete clock transition options. You need to have values for both or none of the following pairs of options: `-rise`/`-fall` and `-min`/`-max`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following produces a warning:

```
set_clock_transition 0.75 -rise CLK1
# Missing a command: set_clock_transition ... -fall CLK1
```


© 1999-2023 All Rights Reserved.

CCD_CLK_CTR12

Message

`Incomplete clock transition options (post-layout)`

Default Severity

Warning

Description

Indicates that `set_input_transition` has incomplete clock transition options. You need to have values for both or none of the following pairs of options: `-rise`/`-fall` and `-min`/`-max`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following produces a warning:

```
set_input_transition 0.75 -rise CLK1
# Missing a command: set_input_transition ... -fall CLK1
```


© 1999-2023 All Rights Reserved.

CCD_CLK_CTR13

Message

`Inconsistent clock transition option values`

Default Severity

Warning

Description

Indicates that for some `set_input_transition` or `set_clock_transition`, the `-min` value is greater than the corresponding `-max` value.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

```
set_clock_transition -min 1 [get_clocks CLK1]
set_clock_transition -max 0.5 [get_clocks CLK1]
```


© 1999-2023 All Rights Reserved.

CCD_CLK_CTR14

Message

`Unusual clock transition option`

Default Severity

Warning

Description

Indicates that `set_input_transition` has either `-clock` or `-clock_fall` defined.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following command causes a rule check violation because of the `-clock` option.

```
set_input_transition -rise -min 0.75 -clock CLK2 IN1
```


© 1999-2023 All Rights Reserved.

CCD_CLK_CTR15

Message

`Negative clock transition value`

Default Severity

Warning

Description

Indicates that you have a negative clock transition value.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following produces a warning because the input transition is a negative number:

```
set_input_transition -0.75 -fall CLK1
```


© 1999-2023 All Rights Reserved.

CCD_CLK_HIER*

This section describes rule checks that relate to hierarchical clocks.

© 1999-2023 All Rights Reserved.

CCD_CLK_HIER1

Message

`Multiple top-level clocks drive a block's clock`

Default Severity

Warning

Description

Indicates that a block's clock definition has more than one top-level root clock in its fan-in cone.

This rule is checked at the top level. When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Note: Specify the current SDC design using the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Example

If `Block_A.sdc` has this line:

```
create_clock -period 8 CLK1A
```

`Top.sdc` has these lines:

```
create_clock -period 10 CLK3
create_clock -period 10 CLK1
```

The Conformal software issues a warning because `Block_A` has two top-level root clocks in its fan-in cone.

© 1999-2023 All Rights Reserved.

CCD_CLK_HIER2a

Message

`A block's clock is slower than a related top-level clock`

Default Severity

Warning

Description

Indicates that the top-level clock's period is not greater than or equal to the period of the corresponding clock definition at the block.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Example

If `Top.sdc` has the following line:

```
create_clock -period 3 [get_ports CLK1]
```

and `Block_A.sdc` has the following line:

```
create_clock -period 5 [get_ports CLK1A]
```

The Conformal software issues a warning because the clock's period at the top level is less than that at the block.

© 1999-2023 All Rights Reserved.

CCD_CLK_HIER2b

Message

`A block's clock is faster than a related top-level clock`

Default Severity

Warning

Description

Indicates that the top-level clock's period is not less than or equal to the period of the corresponding clock definition at the block.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Example

If `Top.sdc` has the following line:

```
create_clock -period 5 [get_ports CLK1]
```

and `Block_A.sdc` has the following line:

```
create_clock -period 3 [get_ports CLK1A]
```

The Conformal software issues a warning because the clock's period at the top level is greater than that at the block.

© 1999-2023 All Rights Reserved.

CCD_CLK_HIER2c

Message

```
A block's clock and a related top-level clock have different duty-cycle
```

Default Severity

Warning

Description

Indicates that a top-level clock's period is equal to the period of the corresponding clock definition at the block, but these two clocks have different duty cycle (`(falling edge - rising edge)/period`).

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Example

If `Top.sdc` has the following line:

```
create_clock -period 10 [get_ports CLK1]
```

and `Block_A.sdc` has the following line:

```
create_clock -period 10 [get_ports CLK1A] -waveform {0 4}
```

The Conformal software issues a warning because the clock's period at the top level is equal to that at the block, but their duty cycles are 50% and 40%, respectively.

© 1999-2023 All Rights Reserved.

CCD_CLK_HIER3

Message

`Missing top-level clock driving a block's clock source`

Default Severity

Warning

Description

Indicates that the top level does not have a `create_clock` statement that defines a clock in the fan-in of the block clock's source object.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Example

If `Block_A.sdc` has this line:

```
create_clock -period 8 CLK1A
```

The Conformal software issues a warning if a line similar to the following, which specifies a clock that drives `CLK1A`, is missing from `Top.sdc`:

```
create_clock -period 10 CLK1
```


© 1999-2023 All Rights Reserved.

CCD_CLK_HIER4

Message

```
Missing block clock on block port/pin connected to a top-level clock source
```

Default Severity

Warning

Description

Indicates that a block's constraint file does not define a clock on a port or pin that is in the clock tree, or is a source, of a top-level clock object. In the case that such a block clock is defined, this rule reports if the block clock and the top-level clock are not defined as equivalent.

Note: To see the clock equivalences (as computed automatically or derived from the `SDC_HIER_CLOCK_EQUIV` CCD parameter), use the `REPORT CLOCK -hier_equiv` command in Verify mode.

For more information, see "Using SDC_HIER_CLOCK_EQUIV" in the *Conformal Constraint Designer User Guide*.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

If the top-level port `clk` is connected to `myblock/clk`, the SDC file of block `myblock` contains:

© 1999-2023 All Rights Reserved.

```
create_clock -period 10 [get_ports clk] -name CLK1
```

and the top-level SDC contains:

```
create_clock -period 10 [get_ports clk] -name CLK
```

This rule will be reported, unless the clock `CLK` is defined as the top-level equivalent of `CLK1`. To do this, add this command to the dofile:

```
set ccd parameter SDC_HIER_CLOCK_EQUIV "CLK CLK1, ... "
```


© 1999-2023 All Rights Reserved.

CCD_CLK_HIER5

Message

```
Inconsistent clock groups between block-level and top-level clocks
```

Default Severity

Warning

Description

Indicates that the relationship between the block-level clock and its corresponding top-level clock is inconsistent. The relationships being checked are: synchronous/asynchronous and mutually exclusive.

To see the clock relationships (as computed automatically or derived from the `SDC_HIER_CLOCK_EQUIV CCD` parameter), use the `REPORT CLOCK -hier_equiv` command in Verify mode. For more information, see "Using SDC_HIER_CLOCK_EQUIV" in the *Conformal Constraint Designer User Guide*.

This rule is checked at the top level. The current SDC design is specified with the `SET SDC DESIGN` command. Subdesigns are specified using the `ADD SDC DESIGN` command or when the `READ HIERARCHICAL SDC` command is used; the root design becomes a top-level design when you specify a subdesign.

This rule check is performed when you use the `RUN RULE CHECK` command at the top level. The check requires that you are in Verify mode and that you have committed the clocks. Also, for each block that is being checked, the rule requires that you have committed the clocks when entering the Verify mode for the block (manually using `COMMIT CLOCK` command, or automatically when using `READ HIERARCHICAL SDC`).

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

© 1999-2023 All Rights Reserved.

Rule Attributes

This policy rule check is customizable; its properties are specified through the attributes listed below. To view all the attributes for this rule check from the tool, use the following command:

```
report rule source "CCD_CLK_HIER5" -verbose
```

Table 6-1 CCD_CLK_HIER5 Attributes

Attribute	Description
check_type	Specifies the types of clock relationships to be checked Possible Value: <all | asynchronous | logically_exclusive | physically_exclusive> Where: all—Check the consistency of all types of clock relationships in block versus top. asynchronous—Checks the consistency of only the synchronous/asynchronous clock relationship in block versus top. logically_exclusive—Checks the consistency of logically mutually exclusive clocks in block versus top. physically_exclusive—Checks the consistency of physically mutually exclusive clocks in block versus top.
check_connectivity	Specifies whether to check the clock connectivity before reporting an occurrence of this rule. This attribute will affect only the asynchronous check type. Possible Value: <no | yes > Where: no—Clock connectivity is not checked. yes—Clock connectivity is checked.

© 1999-2023
All Rights Reserved.

CCD_CLK_HIER6

Message

```
Missing virtual clock at block level
```

Default Severity

Warning

Description

Indicates that a block's constraint file does not define a virtual clock for a top-level equivalent clock which launches or captures timing path data through a block port.

Note: To see the clock equivalences (as computed automatically or derived from the CCD parameter `SDC_HIER_CLOCK_EQUIV`), use the `report clock -hier_equiv` command in Verify mode.

For more information, see "Using SDC_HIER_CLOCK_EQUIV" in the *Conformal Constraint Designer User Guide*.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command

When you run the run rule check command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the -force option.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

Top-level port 'clk' drives the clock pin of top-level flop 'myreg'. The flop 'myreg' is connected to block pin 'myblock/din' and the SDC file of block 'myblock' is missing a command such as:

```
create_clock -name BCK -period 4.0 -waveform {0.0 2.0}
```


© 1999-2023 All Rights Reserved.

The top-level SDC contains:

```
create_clock -name TCK -add -period 4.0 -waveform {0.0 2.0} [get_ports clk]
```

This rule will be reported, unless the clock TCK is defined as the top-level equivalent of BCK. To do this, add this command to the dofile:

```
set ccd parameter SDC_HIER_CLOCK_EQUIV "TCK BCK, ... "
```


© 1999-2023 All Rights Reserved.

CCD_CLK_INT*

This section describes rule checks that relate to SDC integration for clocks and their attributes.

© 1999-2023 All Rights Reserved.

CCD_CLK_INT1

Message

`SDC integration: clocks`

Default Severity

Warning

Description

Configures the integration of `create_clock` and `create_generated_clock` commands, and reports the messages generated during this integration process.

This is checked when you run the `INTEGRATE` command.

Example

For example, if you are using the default SDC integration configuration (`add rule instance -default`), and you have a block `BLK` with an input `CLK1` that is connected to a top-level net `clk_net` that is driven by selector MUX. If `BLK.sdc` contains the following command:

```
create_clock ?name CLK1 ?period 10 [get_ports CLK1]
```

you will get the following message when you try to integrate this block constraint into a full-chip SDC file:

```
CCD_CLK_INT1: SDC integration: clocks
Severity: Warning    Occurrence: 1
1: [blk_clk_on_blk_input_prop_to_internal_okay] Propagation of clock backwards from BLK/CLK1 was stopped at clk_net. Candidate clock sources in its fanin: clk1, clk2
```


© 1999-2023 All Rights Reserved.

CCD_IO_IDL*

This section describes rule checks that relate to input delay violations.

© 1999-2023 All Rights Reserved.

CCD_IO_IDL1

Message

`Unconstrained input`

Default Severity

Warning

Description

Indicates that you have an unconstrained input. Flags input ports that do not have `set_input_delay` and annotates the ports that are in `set_dont_touch` and `set_dont_touch_network`.

Only input ports that are connected to some constrained path endpoint (a clocked register or a pin with `set_output_delay`) are considered for this rule. For example, a clock source will not be checked in this rule if it only drives clock pins of sequential elements.

This rule applies only to ports that are start points of a timing path that does not match any `set_false_path`.

Note: `set_false_path` should have both `-setup` and `-hold` specified, or neither. Otherwise, the port is still subject to this rule check.

This is checked when you run the `run rule check` command.

Example

If your design has three input pins—`IN1`, `IN2`, and `IN3`—the following produces a warning because the input delay for `IN1` is missing:

```
set_input_delay 0.75 -rise -clock CLK1 {IN2}
set_input_delay 0.75 -rise -clock CLK1 {IN3}
```


© 1999-2023 All Rights Reserved.

CCD_IO_IDL2

Message

```
Incorrect input delay constraint vs. several clocks (missing -add_delay)
```

Default Severity

Warning

Description

Indicates that, for an input port that is driven by more than one clock, the `set_input_delay` command is missing the `-add_delay` option, which is required for each additional clock that drives the input port.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following causes a rule check violation because there is a missing `-add_delay` option for `CLK1`.

```
set_input_delay 2.0 -clock CLK0 {IN1}
set_input_delay 3.0 -clock CLK1 {IN1}
```


© 1999-2023 All Rights Reserved.

CCD_IO_IDL4

Message

`Incomplete input delay options`

Default Severity

Warning

Description

Indicates that you have incomplete input delay options. The Conformal Constraint Designer checks `set_input_delay` to ensure that there are values for both or none of the following option combinations: `-rise`/`-fall` and `-min`/`-max`. This applies to every specified clock and for delays that are specified without a clock.

This rule does not apply to ports that are specified in `set_false_path` and that have a single path specification switch, which is either `-from`, `-to`, or a single `-through`.

Note: `set_false_path` should have both `-setup` and `-hold` specified, or neither. Otherwise, the port is still subject to this rule check.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following command causes a rule check violation because it is missing the `-fall` option.

```
set_input_delay 0.75 -rise -clock CLK1 {IN2}
```


© 1999-2023 All Rights Reserved.

CCD_IO_IDL5

Message

`Inconsistent input delay vs. clock period, or min>max`

Default Severity

Warning

Description

Indicates one of the following conditions for `set_input_delay` values:

- `-min delay > -max delay`
- a delay value does not satisfy the formula specified in CCD parameter `SDC_INPUT_DELAY_CHECK`

Note: The default formula is '`delay<ratio*period`,' where '`delay`' indicates `-min/-max` delay, '`ratio`' indicates the value of CCD parameter `SDC_INPUT_DELAY_RATIO`, and '`period`' indicates the clock period. `SDC_INPUT_DELAY_RATIO` has a default value of 1.0. To change the value of `SDC_INPUT_DELAY_RATIO` or `SDC_INPUT_DELAY_CHECK`, use the `SET CCD PARAMETER` command.

Note: If there is a `set_multicycle_path` N with a single `-from` or `-through` list, in which the port is included, then the clock period is multiplied by the multi-cycle path value.

Note: This rule does not apply to ports that are specified in `set_false_path` and that have a single path specification switch, which is either `-from` or a single `-through`. `set_false_path` should have both `-setup` and `-hold` specified, or neither. Otherwise, the port is still subject to this rule check.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

© 1999-2023 All Rights Reserved.

Examples

- In the following command, a violation occurs because 2.75 (minimum value) > 1.75 (maximum value).

```
create_clock -period 10 -name CLK1_w CLK1
set_input_delay 1.75 -max -clock CLK1_w {IN1}
set_input_delay 2.75 -min -clock CLK1_w {IN1}
```

- In the following command, the Conformal Constraint Designer issues a warning because the input delay does not satisfy the formula '`delay<ratio*period`,' where `SDC_INPUT_DELAY_RATIO` has its default value 1.0:

```
create_clock -period 8.0 -name vCLK3
set_input_delay 10.0 [get_ports {IN1, IN2}] -clock vCLK3
```


© 1999-2023 All Rights Reserved.

CCD_IO_IDL6

Message

`Input delay constrained vs. wrong clock`

Default Severity

Warning

Description

For input ports that have input delay defined with respect to a clock, and that are connected to some constrained endpoint, this rule indicates that none of those endpoints is constrained by a clock that is synchronous to the one for which the input delay was defined.

This rule does not apply to ports that are specified in `set_false_path` and that have a single path specification switch, which is either `-from`, `-to`, or a single `-through`.

Note: `set_false_path` should have both `-setup` and `-hold` specified, or neither. Otherwise, the port is still subject to this rule check.

This is checked when you run the `run rule check` command.

Example

In the following example, if there is no path from `IN1` to a sequential element constrained by `CLK2`, the Conformal Constraint Designer issues a warning because `set_input_delay` has specified the wrong clock:

```
set_input_delay 2.75 -min -clock CLK2 {IN1}
```


© 1999-2023 All Rights Reserved.

CCD_IO_IDL7

Message

```
Detected input delay vs. clock that is not virtual
```

Default Severity

Warning

Description

Indicates that there is an input delay specified for a clock that is not virtual.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following produces a warning message if `CLK1` is a real clock (rather than a virtual clock):

```
set_input_delay 2.75 -min -clock CLK1 {IN1}
```


© 1999-2023 All Rights Reserved.

CCD_IO_IDL9

Message

`Detected level sensitive input`

Default Severity

Warning

Description

Indicates that is a level-sensitive input port. Specifically, you have a `set_input_delay` command that uses `-level_sensitive`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following command causes a warning because it uses a level-sensitive delay.

```
set_input_delay 2.0 -clock CLK1 -level_sensitive {IN1}
```


© 1999-2023 All Rights Reserved.

CCD_IO_IDL10

Message

```
Detected edge sensitive input relative to clock falling edge
```

Default Severity

Warning

Description

Indicates that there is an edge-sensitive input in relation to the clock falling edge. Specifically, you have a `set_input_delay` command that uses `-clk_fall`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following command causes a warning because its edge-sensitive delay input uses a clock falling edge.

```
set_input_delay 2.0 -clock CLK1 -clock_fall {IN1}
```


© 1999-2023 All Rights Reserved.

CCD_IO_IDL11

Message

`Unusual input delay option`

Default Severity

Warning

Description

Indicates that you have a `set_input_delay` command that uses either `-source_latency_included` or `-network_latency_included`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following command causes a violation because it uses `-source_latency_included`.

```
set_input_delay 2.0 -clock CLK1 -source_latency_included {IN1}
```


© 1999-2023 All Rights Reserved.

CCD_IO_IDL12

Message

`Negative input delay value`

Default Severity

Warning

Description

Indicates that you have a `set_input_delay` command with a negative value.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following code causes a violation because it has a negative delay value.

```
set_input_delay -2.0 -clock CLK1 {IN1}
```


© 1999-2023 All Rights Reserved.

CCD_IO_IDL13

Message

```
An input port is in a timing path within a clock group for which it does not have
     input delay defined
```

Default Severity

Warning

Description

Indicates that you have an input port in whose fanout there are sequential elements or ports in a given clock group, but this input port has no input delay with respect to any clock in that clock group.

Note: This rule applies only to ports that are start points of a timing path that does not match any `set_false_path`.

`set_false_path` should have both `-setup` and `-hold` specified, or neither. Otherwise, the port is still subject to this rule check.

This rule does not check inputs that are clock sources.

This is checked when you run the `run rule check` command.

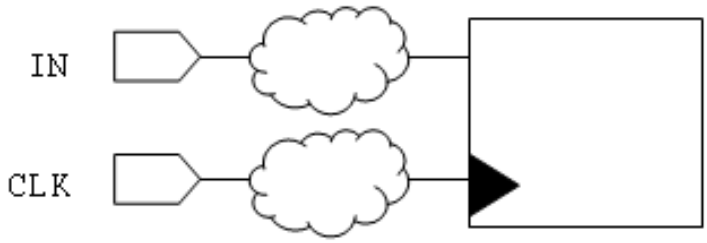
Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

With the following command, the Conformal Constraint Designer checks that `IN` has an input delay with respect to `CLK` or any other clock in the same group.

© 1999-2023 All Rights Reserved.

```
set_input_delay 1 IN
```



This example satisfies `CCD_IO_IDL1`; however, `CCD_IO_IDL13` issues a warning because it requires a delay with respect to `CLK` or any clock synchronous to it.

© 1999-2023
All Rights Reserved.

CCD_IO_IDL15

Message

```
An input delay is defined vs. a virtual clock on a port/pin, but no sequential
     element in its fanout is driven by a real/generated clock with the same
     waveform
```

Default Severity

Warning

Description

Indicates that, for input ports or internal pins that have input delay defined with respect to a virtual clock, there are sequential elements in the fanout, but their clock pins are driven by real or generated clocks that have a different period or waveform edges than the virtual clock.

This rule does not apply to ports that are specified in `set_false_path` and that have a single path specification switch, which is either `-from`, `-to`, or a single `-through`.

Note: `set_false_path` should have both `-setup` and `-hold` specified, or neither. Otherwise, the port is still subject to this rule check.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

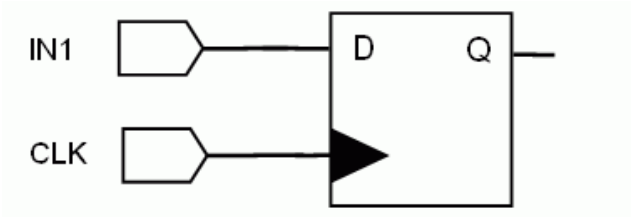
Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

In the following example, the Conformal Constraint Designer issues a warning because `set_input_delay` has specified a virtual clock whose frequency does not match that of any real clock driving a sequential element in the fanout of `IN1`.

© 1999-2023 All Rights Reserved.

```
create_clock -name RCLK -period 10 -waveform {0 5} [get_ports CLK1]
create_clock -name VCLK -period 20 -waveform {0 10}
set_input_delay 2.75 -clock VCLK {IN1}
```



© 1999-2023
All Rights Reserved.

CCD_IO_IDL16

Message

`set_input_delay defined on an internal pin`

Default Severity

Warning

Description

Indicates that the SDC file specifies `set_input_delay` on a pin internal to the current design (not a port). In some cases this can cause timing path splitting.

This rule does not necessarily indicate a problem, depending on the methodology used.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

```
set_input_delay 1 [get_pins blk/d1]
```


© 1999-2023 All Rights Reserved.

CCD_IO_ITR*

This section describes rule checks that relate to input transitions.

© 1999-2023 All Rights Reserved.

CCD_IO_ITR1

Message

`Undefined input transition`

Default Severity

Warning

Description

Indicates that an input or inout port does not have `set_input_transition`, `set_driving_cell`, `set_drive`, or `set_load` specified.

This rule check does not apply to ports that have `set_dont_touch_network` specified.

This is checked when you read in your SDC files using the `READ SDC` command.

© 1999-2023 All Rights Reserved.

CCD_IO_ITR2

Message

`Incomplete input transition options`

Default Severity

Warning

Description

Indicates that there are incomplete input transition options. You must have values for both or none of the following option combinations: `-rise`/`-fall` and `-min`/`-max`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following produces a warning because it specifies `-min`, but not `-max`:

```
set_input_transition 2.74 -min {IN1}
```

You can specify the `-min` and `-max` values in the same or separate commands.

© 1999-2023 All Rights Reserved.

CCD_IO_ITR3

Message

`Inconsistent input transition option values`

Default Severity

Warning

Description

Indicates that there are `set_input_transition` or `set_drive` options with inconsistent values. Specifically, the Conformal Constraint Designer checks for instances where `min > max`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following generates a warning because `-min > -max` for `IN1`.

```
set_input_transition -rise -min 0.75 IN1
set_input_transition -rise -max 0.50 IN1
```


© 1999-2023 All Rights Reserved.

CCD_IO_ITR4

Message

`Input transition value outside characterization range`

Default Severity

Warning

Description

Indicates that the input transition value is outside the characterization range. If the port specified as an argument to `set_input_transition` drives a pin of a liberty cell with `min_transition` or `max_transition` defined, the characterization range is defined by these values. If not, the characterization range is defined by the `SDC_MIN_INPUT_TRANSITION` and `SDC_MAX_INPUT_TRANSITION` parameters.

For more information on how to set these parameters, see `SET CCD PARAMETER` in the *Conformal Constraint Designer Reference Manual*.

This is checked when you read in your SDC files using the `READ SDC` command.

© 1999-2023 All Rights Reserved.

CCD_IO_ITR5

Message

```
Inconsistent input transition value vs. associated clock transition value (pre-
     layout)
```

Default Severity

Warning

Description

Indicates that the input transition value is inconsistent versus the associated clock transition value. Specifically, the `-min` value of `set_clock_transition` should be greater than or equal to the `-min` value of `set_input_transition`, and the `-max` value of `set_clock_transition` should be less than or equal to the `-max` value of `set_input_transition`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following generates a warning because the transition value in `set_input_transition` is greater than the transition value in `set_clock_transition` in the `-min` case:

```
create_clock -period 10 -name CLK1
set_clock_transition 0.75 -rise CLK1
set_input_transition -rise -clock CLK1 1.0 IN1A
```


© 1999-2023 All Rights Reserved.

CCD_IO_ITR6

Message

```
Inconsistent input transition value vs. associated clock transition value (post-
     layout)
```

Default Severity

Warning

Description

Indicates that the input transition value is inconsistent versus the associated clock transition value. Specifically, the `-min` value of `set_clock_transition` should be greater than or equal to the `-min` value of `set_input_transition`, and the `-max` value of `set_clock_transition` should be less than or equal to the `-max` value of `set_input_transition`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following generates a warning because the transition value in `set_input_transition` is greater than the transition value in `set_clock_transition` in the `-min` case:

```
create_clock -period 10 -name CLK1
set_clock_transition 0.75 -rise CLK1
set_input_transition -rise -clock CLK1 1.0 IN1A
```


© 1999-2023 All Rights Reserved.

CCD_IO_ITR7

Message

`Unusual input transition option`

Default Severity

Warning

Description

Indicates that you have instances of `set_input_transition -clock/-clock_fall` and `set_driving_cell -min/-max/-clock/-clock_fall`. The Conformal Constraint Designer issues a warning when it detects these options, because they are not present in all versions of SDC.

This is checked when you read in your SDC files using the `READ SDC` command.

© 1999-2023 All Rights Reserved.

CCD_IO_ITR8

Message

`set_drive is not recommended`

Default Severity

Warning

Description

Indicates that the `set_drive` command has been used. This command works with a linear delay model. It is inaccurate when using non-linear delay models. This command has been superseded by `set_driving_cell`, which works with all delay models accurately.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

```
set_drive -rise 0.75 [get_ports {IN1}]
```


© 1999-2023 All Rights Reserved.

CCD_IO_ITR9

Message

`Detected driving cell that is not recommended`

Default Severity

Warning

Description

Indicates that a `set_driving_cell` command has specified a library cell that is not one of the cells listed in the CCD parameter `SDC_DRIVING_CELLS`. If the value of this parameter is empty (the default), then all library cells are allowed.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

If the dofile contains this command:

```
set_ccd_parameter SDC_DRIVING_CELLS "BFLLX16 BFLLX12"
```

then this rule would report a warning for the following SDC command:

```
set_driving_cell -lib_cell [get_lib_cells OR2LLX05] [get_ports {in1}]
```


© 1999-2023 All Rights Reserved.

CCD_IO_ITR10

Message

`set_load and set_driving_cell/set_drive not used together`

Default Severity

Warning

Description

Indicates that a port has a driving cell (or a `set_drive`) but no load value, or a load value but no driving cell (or `set_drive`).

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following design has 4 input ports named `pi0`, `pi1`, `pi2`, and `pi3`.

```
set_load 0.5 [get_ports {pi0 pi1}]
set_driving_cell $CELL_NAME [get_ports {pi0 pi2}]
```

`pi0` has both `set_load` and `set_driving_cell`. It does not flag `ITR10`.

`pi1` has only `set_load`. This port flags `ITR10`

`pi2` has `set_driving_cell`, but lacks `set_load`. This port flags `ITR10`.

`pi3` has neither `set_driving_cell` or `set_load`. It does not flag `ITR10`.

© 1999-2023 All Rights Reserved.

CCD_IO_ODL*

This section describes rule checks that relate to output delays. Output delays help model the external delays that arrive at the output port of a constrained module.

© 1999-2023 All Rights Reserved.

CCD_IO_ODL1

Message

`Unconstrained output`

Default Severity

Warning

Description

Indicates that you have an unconstrained output. Specifically, the Conformal Constraint Designer checks for output ports without `set_output_delay` and annotates the ones that in `set_dont_touch` and `set_dont_touch_network`.

Only output ports that are connected to some constrained path startpoint (a clocked register or a pin with `set_input_delay`) are considered for this rule. For example, an undriven output port will not be checked in this rule.

This rule applies only to ports that are end points of a timing path that does not match any `set_false_path`.

Note: `set_false_path` should have both `-setup` and `-hold` specified, or neither. Otherwise, the port is still subject to this rule check.

This is checked when you run the `run rule check` command.

Example

If a design contains three output ports—`OUT1`, `OUT2`, and `OUT3`—the following produces a warning because the output delay for `OUT1` is missing:

```
set_output_delay 0.75 -rise -clock CLK1 {OUT2}
set_output_delay 0.75 -rise -clock CLK1 {OUT3}
```


© 1999-2023 All Rights Reserved.

CCD_IO_ODL2

Message

```
Incorrect output delay constraint vs. several clocks (missing -add_delay)
```

Default Severity

Warning

Description

Indicates that there is an incorrect output delay constraint versus multiple clocks. Specifically, the Conformal Constraint Designer checks output ports that are driven by more than one clock; `set_output_delay` commands must have a `-add_delay` option defined for each additional clock that drives the output port.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following command causes a rule check violation because there is a missing `-add_delay` option for `CLK2`.

```
set_output_delay 2.75 -clock CLK2 {OUT2}
set_output_delay 3.0 -clock CLK1 {OUT2}
```


© 1999-2023 All Rights Reserved.

CCD_IO_ODL4

Message

```
Incomplete output delay options
```

Default Severity

Warning

Description

Indicates that you have incomplete output delay options. You must have values for both or none of the options in the following combinations: `-rise`/`-fall` and `-min`/`-max`. This applies for every clock specified and for delays specified without a clock.

This rule does not apply to ports that are specified in `set_false_path` and that have a single path specification switch, which is either `-from`, `-to`, or a single `-through`.

Note: `set_false_path` should have both `-setup` and `-hold` specified, or neither. Otherwise, the port is still subject to this rule check.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following generates a warning:

```
set_output_delay 0.75 -rise -clock CLK1 {OUT1}
set_output_delay 0.75 -max -clock CLK1 {OUT1}
```

because there is no corresponding `set_output_delay` for `-fall`/`-min`.

© 1999-2023 All Rights Reserved.

CCD_IO_ODL5

Message

`Inconsistent output delay vs. clock period, or min>max`

Default Severity

Warning

Description

Indicates one of the following conditions for `set_output_delay` values:

- `-min delay > -max delay`
- a delay value does not satisfy the formula specified in CCD parameter `SDC_OUTPUT_DELAY_CHECK`

Note: The default formula is '`delay<ratio*period`,' where '`delay`' indicates `-min/-max` delay, '`ratio`' indicates the value of CCD parameter `SDC_OUTPUT_DELAY_RATIO`, and '`period`' indicates the clock period. `SDC_OUTPUT_DELAY_RATIO` has a default value of 1.0. To change the value of `SDC_OUTPUT_DELAY_RATIO` or `SDC_OUTPUT_DELAY_CHECK`, use the `SET CCD PARAMETER` command.

Note: If there is a `set_multicycle_path` N with a single `-to` or `-through` list, in which the port is included, then the clock period is multiplied by the multi-cycle path value.

Note: This rule does not apply to ports that are specified in `set_false_path` and that have a single path specification switch, which is either `-to` or a single `-through`. `set_false_path` should have both `-setup` and `-hold` specified, or neither. Otherwise, the port is still subject to this rule check.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

© 1999-2023 All Rights Reserved.

Examples

- In the following command, the Conformal Constraint Designer issues a warning because the minimum delay is greater than the maximum delay:

```
set_output_delay 1.74 -min -clock CLK1_w {OUT1}
set_output_delay 0.74 -max -clock CLK1_w {OUT1}
```

- In the following command, the Conformal Constraint Designer issues a warning because the output delay does not satisfy the formula `delay<ratio*period`, where `SDC_OUTPUT_DELAY_RATIO` has its default value 1.0:

```
create_clock -period 8.0 -name vCLK3
set_output_delay 10.0 [get_ports {IN1, IN2}] -clock vCLK3
```


© 1999-2023 All Rights Reserved.

CCD_IO_ODL6

Message

`Output delay constrained vs. wrong clock`

Default Severity

Warning

Description

For output ports that have output delay defined with respect to a clock, and that are connected to some constrained startpoint, this rule indicates that none of those start points is constrained by a clock that is synchronous to the one for which the output delay was defined.

This rule does not apply to ports that are specified in `set_false_path` and that have a single path specification switch, which is either `-from`, `-to`, or a single `-through`.

Note: `set_false_path` should have both `-setup` and `-hold` specified, or neither. Otherwise, the port is still subject to this rule check.

This is checked when you run the `run rule check` command.

Example

In the following example, if there no path from a sequential element constrained by `CLK2` to `OUT1`, the Conformal Constraint Designer issues a warning because `set_output_delay` has specified the wrong clock:

```
set_output_delay 2.75 -min -clock CLK2 {OUT1}
```


© 1999-2023 All Rights Reserved.

CCD_IO_ODL7

Message

`Detected output delay vs. clock that is not virtual`

Default Severity

Warning

Description

Indicates that there is an output delay specified versus a clock that is not virtual.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following produces a warning because `set_output_delay` should specify the virtual clock, `vCLK3`, rather than the real clock, `CLK3`:

```
create_clock -period 10 CLK3
create_clock -period 10 -name vCLK3
set_output_delay 0.75 -min -clock CLK3 {OUT1}
```


© 1999-2023 All Rights Reserved.

CCD_IO_ODL9

Message

`Detected level sensitive output`

Default Severity

Warning

Description

Indicates level-sensitive outputs. Specifically, there is a `set_output_delay` command that uses `-level_sensitive`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following command causes a warning because it uses a level-sensitive output delay.

```
set_output_delay 2.0 -clock CLK1 -level_sensitive {OUT1A}
```


© 1999-2023 All Rights Reserved.

CCD_IO_ODL10

Message

`Detected edge sensitive output relative to clock falling edge`

Default Severity

Warning

Description

Indicates that there is an edge-sensitive output in relation to a clock-falling edge. Specifically, there is a `set_output_delay` command that uses `-clock_fall`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following command causes a warning because its edge-sensitive output delay uses a clock falling edge.

```
set_output_delay 2.0 -clock CLK1 -clock_fall {OUT1A}
```


© 1999-2023 All Rights Reserved.

CCD_IO_ODL11

Message

`Unusual output delay option`

Default Severity

Warning

Description

Indicates that you have a `set_output_delay` command that uses either `-source_latency_included` or `-network_latency_included`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following command causes a violation because it uses `-source_latency_included`.

```
set_output_delay 2.0 -clock CLK1 -source_latency_included {OUT1A}
```


© 1999-2023 All Rights Reserved.

CCD_IO_ODL12

Message

`Negative -max output delay value`

Default Severity

Warning

Description

Indicates that you have a negative maximum output delay value. Specifically, this rule check looks for `set_output_delay` commands that have a negative value, except for those where `-min` is specified and `-max` is not.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following command causes a violation because it has a negative delay value.

```
set_output_delay -2.0 -clock CLK1 {OUT1A}
```


© 1999-2023 All Rights Reserved.

CCD_IO_ODL13

Message

```
An output port is in a timing path within a clock group for which it does not have
     output delay defined
```

Default Severity

Warning

Description

Indicates that you have an output port in whose fan-in there are sequential elements or ports in a given clock group, but this output port has no output delay with respect to any clock in that clock group.

Note: This rule applies only to ports that are end points of a timing path that does not match any `set_false_path`.

`set_false_path` should have both `-setup` and `-hold` specified, or neither. Otherwise, the port is still subject to this rule check.

This rule does not check outputs that are in clock trees.

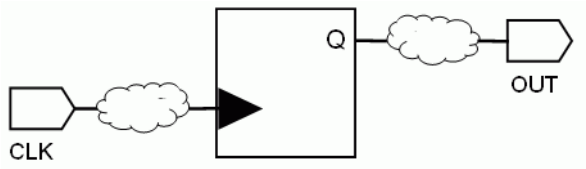
This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

In the following figure, the Conformal Constraint Designer checks that `OUT` has an output delay with respect to `CLK` or any other clock in the same group.

```
set_output_delay 1 OUT
```



© 1999-2023
All Rights Reserved.

This example satisfies `CCD_IO_ODL1`; however, `CCD_IO_ODL13` issues a warning because it requires a delay with respect to `CLK` or any clock synchronous to it.

© 1999-2023 All Rights Reserved.

CCD_IO_ODL14

Message

`Positive/zero -min output delay value`

Default Severity

Warning

Description

Indicates that you have a positive/zero minimum delay value.

Note: In a design methodology where there is no glue logic between blocks, and all block inputs are registered, the delay from a block's output to a register in another block is zero. Because the register is outside the block, the hold check is done versus the ideal clock edge. Therefore, to model the real hold check, the `output_delay` must be negative, and its absolute value must equal the expected hold time requirement. This is the reason rule `CCD_IO_ODL14` reports non-negative `-min` output delays.

In a design methodology using glue logic, or where output delays are set on the top-level output ports, the external delay from the output to the capturing register could be positive and bigger than the hold requirement, in which case the correct output delay value would be positive. For this methodology, rule `CCD_IO_ODL14` is not as useful and might be disabled.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following command causes a violation because it does not have a negative delay value.

```
set_output_delay -min 0 -clock CLK1 {OUT1A}
```


© 1999-2023 All Rights Reserved.

CCD_IO_ODL15

Message

```
An output delay is defined vs. a virtual clock on a port/pin, but no sequential
     element in its fanin is driven by a real/generated clock with the same waveform
```

Default Severity

Warning

Description

Indicates that, for output ports or internal pins that have output delay defined with respect to a virtual clock, there are sequential elements in the fanin, but their clock pins are driven by real or generated clocks that have a different period or waveform edges than the virtual clock.

This rule does not apply to ports that are specified in `set_false_path` and that have a single path specification switch, which is either `-from`, `-to`, or a single `-through`.

Note: `set_false_path` should have both `-setup` and `-hold` specified, or neither. Otherwise, the port is still subject to this rule check.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

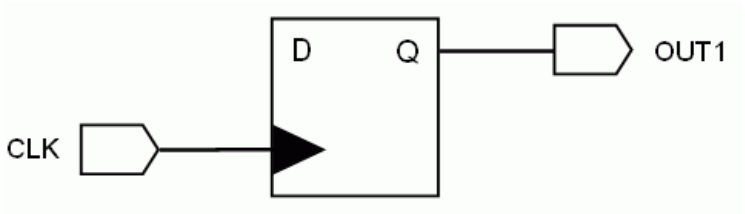
Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

In the following example, the Conformal Constraint Designer issues a warning because `set_output_delay` has specified a virtual clock whose frequency does not match that of any real clock driving a sequential element in the fan-in of `OUT1`.

© 1999-2023 All Rights Reserved.

```
create_clock -name RCLK -period 10 -waveform {0 5} [get_ports CLK1]
create_clock -name VCLK -period 20 -waveform {0 10}
set_output_delay 2.75 -clock VCLK {OUT1}
```



© 1999-2023
All Rights Reserved.

CCD_IO_ODL16

Message

`set_output_delay defined on an internal pin`

Default Severity

Warning

Description

Indicates that the SDC file specifies `set_output_delay` on a pin internal to the current design (not a port). In some cases this can cause timing path splitting.

This rule does not necessarily indicate a problem, depending on the methodology used.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

```
set_output_delay 1 [get_pins blk/d1]
```


© 1999-2023 All Rights Reserved.

CCD_IO_OLD*

This section describes rule checks that relate to output load violations.

© 1999-2023 All Rights Reserved.

CCD_IO_OLD1

Message

`Undefined load on output`

Default Severity

Warning

Description

Indicates that you have an undefined load on outputs (PO and PIO), which is set using the `set_load` command. This rule does not trigger when you specify only a `-min` or only a `-max`.

This rule check does not apply to ports that have `set_dont_touch_network` specified.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

If a design has three outputs—`OUT1`, `OUT2`, and `OUT3`—and you have the following command:

```
set_load 5.36 {OUT3, OUT2}
```

You would get a warning message if you do not have a command that specifies the load for `OUT1`.

© 1999-2023 All Rights Reserved.

CCD_IO_OLD2

Message

`Load value outside characterization range`

Default Severity

Warning

Description

Indicates that the load value is outside the characterization range. If the port specified as an argument to `set_load` is driven by a pin of a liberty cell with `min_capacitance` or `max_capacitance` defined, the characterization range is defined by these values. If not, the characterization range is defined by the `SDC_MIN_CAPACITANCE` and `SDC_MAX_CAPACITANCE` parameters.

For a list of CCD parameters and definitions, see `SET CCD PARAMETER` in the *Conformal Constraint Designer Reference Manual*.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

If you set the `SDC_MIN_CAPACITANCE` parameter to `0.1` and the `SDC_MAX_CAPACITANCE` parameter to `0.5`, then the following command causes a warning because 0.75 is outside the characterization range.

```
set_load 0.75 {OUT1}
```


© 1999-2023 All Rights Reserved.

CCD_IO_OLD3

Message

```
Inconsistent load values (-min > -max)
```

Default Severity

Warning

Description

Indicates that there is a port or net with a load value for `-min` that is greater than the load value for `-max`. For ports, `-pin_load` and `-wire_load` values are checked independently.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The following two commands set inconsistent constraints on the port `OUT`:

```
set_load 0.5  [all_outputs]
set_load 0.75 -min [get_ports OUT] -pin_load
```


© 1999-2023 All Rights Reserved.

CCD_IO_OLD4

Message

`Incomplete load options`

Default Severity

Warning

Description

Indicates that there is a port or net with a load value for `-min` but not for `-max`, or vice-versa. For ports, `-pin_load` and `-wire_load` values are checked independently.

Note: This rule is only checked in min-max mode (as set by the `set_operating_conditions` command)

This is checked when you read in your SDC files using the `READ SDC` command.

Example

An occurrence of this rule would be reported if these commands are specified because the `-max` wire load would be missing for port `OUT`:

```
set_load 0.75 -min [get_ports OUT] -pin_load
set_load 1.0 -min [get_ports OUT] -wire_load
set_load 2.0 -max [get_ports OUT]
```


© 1999-2023 All Rights Reserved.

CCD_IO_OLD5

Message

```
Used set_load -min/-max when not in min-max mode
```

Default Severity

Warning

Description

Indicates that you have used `set_load`'s `-min`/`-max` options but the design has not been previously set to be timed in min-max mode.

To set min-max mode, use the `set_operating_condition` SDC command with the `-analysis_type <bc_wc | on_chip_variation>` option, or with `-min` and `-max`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

```
set_operating_condition -analysis_type single some_opcon
set_load -min -pin_load 0.25 [get_ports Z]
```


© 1999-2023 All Rights Reserved.

CCD_IO_DRC*

This section describes rule checks that relate to port capacitance.

© 1999-2023 All Rights Reserved.

CCD_IO_DRC1

Message

`Undefined set_max_capacitance on input or inout port`

Default Severity

Warning

Description

Indicates that an input or inout port has not set `set_max_capacitance`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

If you do not have the following set for input port `in_port`, the Conformal Constraint Designer issues a warning:

```
set_max_capacitance 0.5 [get_ports {in_port}]
```


© 1999-2023 All Rights Reserved.

CCD_IO_DRC2

Message

`Undefined set_max_capacitance on output or inout port`

Default Severity

Warning

Description

Indicates that an output or inout port does not `set_max_capacitance` set.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

If you do not have the following set for output port `out_port`, the Conformal Constraint Designer issues a warning:

```
set_max_capacitance 0.5 [get_ports {out_port}]
```


© 1999-2023 All Rights Reserved.

CCD_IO_HIER*

This section describes rules that perform checks between the constraints placed on the inputs or outputs of blocks (such as on SDC designs other than the top level) and the environment for which the blocks are placed (such as the top-level design and its own constraints).

© 1999-2023 All Rights Reserved.

CCD_IO_HIER1

Message

```
Block input port with delay vs. a clock is not in a timing path within the same
     clock group at the top-level
```

Default Severity

Warning

Description

For every block input port `IN` with input delay versus clock `CLK`, Conformal Constraint Designer finds the top-level clock groups of `CLK`'s top-level equivalent clock, and checks that there is a path between timing points in those clock groups that traverse `IN`.

This is checked when you run the `run rule check` command.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

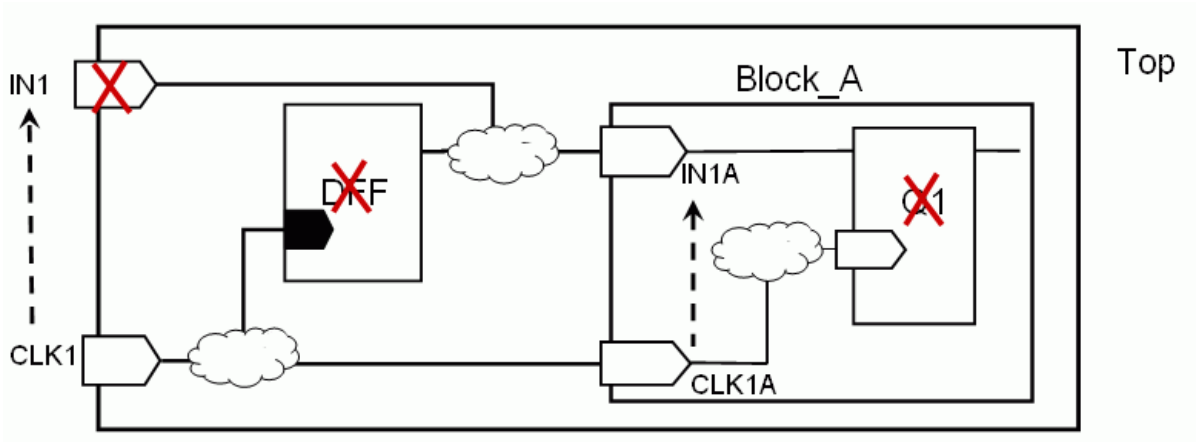
Example

The design in the following figure (where `IN1A` has an input delay versus `CLK1A`) would cause a rule check violation if the following were not present in the combinational fan-in of `IN1A`:

- A DFF that is clocked by the top-level clock `CLK1`
- A primary input IN1 that has an input delay versus `CLK1`.

© 1999-2023 All Rights Reserved.

This rule would also be flagged if no DFF/port in the same clock group can be found in the fan-out of `IN1A`.



© 1999-2023
All Rights Reserved.

CCD_IO_HIER2

Message

```
Block input port is in a timing path within a clock group for which it does not
     have input delay defined
```

Default Severity

Warning

Description

For every block input port `IN` that has a timing startpoint in its fan-in and a timing endpoint in its fanout, such as both belong to the same top-level clock group, Conformal Constraint Designer checks that `IN` has input delay defined versus a block clock whose top-level equivalent clock is in the same clock group.

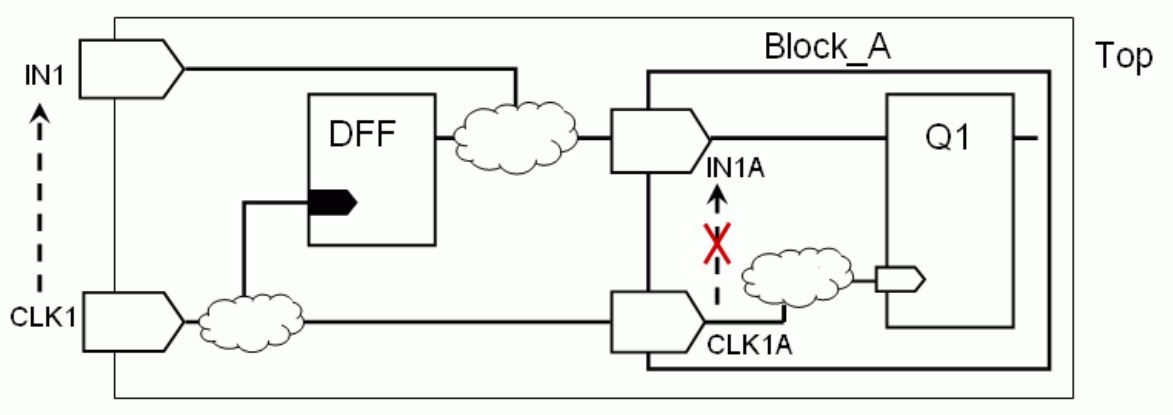
This is checked when you run the `run rule check` command.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

© 1999-2023 All Rights Reserved.

Example

The design in the following figure would cause a rule check violation, if `IN1A` did not have an input delay versus `CLK1A`. `CLK1A` is related to the top-level clock `CLK1`, whose domain drives `IN1A`.



© 1999-2023
All Rights Reserved.

CCD_IO_HIER3

Message

```
Block output port with delay vs. a clock is not in a timing path within the same
     clock group at the top-level
```

Default Severity

Warning

Description

For every block output port `OUT` with output delay versus clock `CLK`, Conformal Constraint Designer finds the top-level clock group of `CLK`'s top-level equivalent clock, and checks that there is a path between timing points in that clock group that traverses `OUT`.

This is checked when you run the `run rule check` command.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

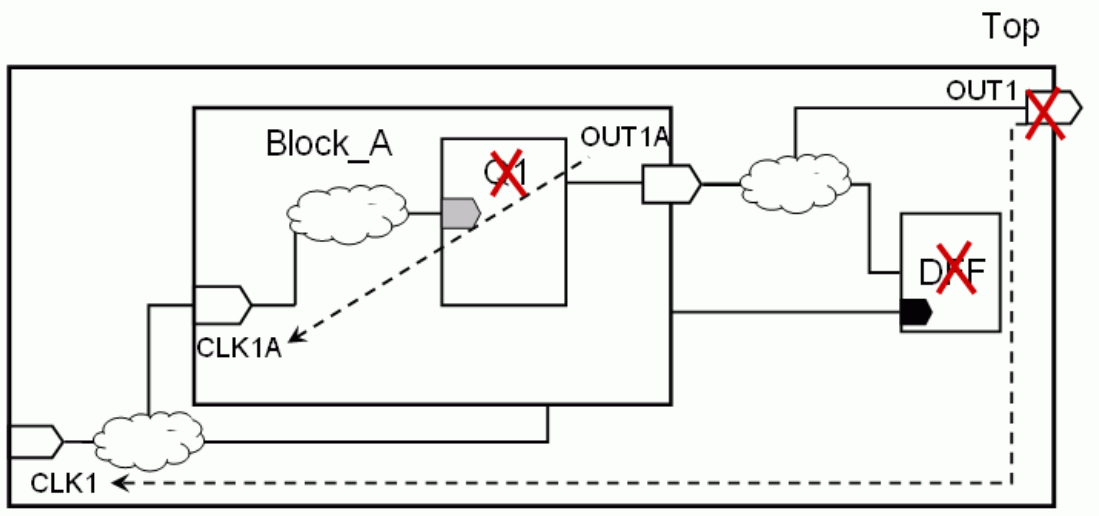
Example

The design in the following figure (where `OUT1A` has a delay versus `CLK1A`) would cause a rule check violation if none of the following were present in the combinational fan-out of `OUT1A`:

- A DFF that is clocked by the top-level clock `CLK1`
- A pin or primary output `OUT1` that has an output delay versus `CLK1`.

© 1999-2023 All Rights Reserved.

This rule would also be flagged if no DFF/port in the same clock group can be found in the fan-in of `OUT1A`.



© 1999-2023
All Rights Reserved.

CCD_IO_HIER4

Message

```
Block output port is in a timing path within a clock group for which it does not
     have output delay defined
```

Default Severity

Warning

Description

For every block output port `OUT` that has a timing startpoint in its fan-in and a timing endpoint in its fanout, such as both belong to the same top-level clock group, Conformal Constraint Designer checks that `OUT` has output delay defined versus a block clock whose top-level equivalent clock is in the same clock group.

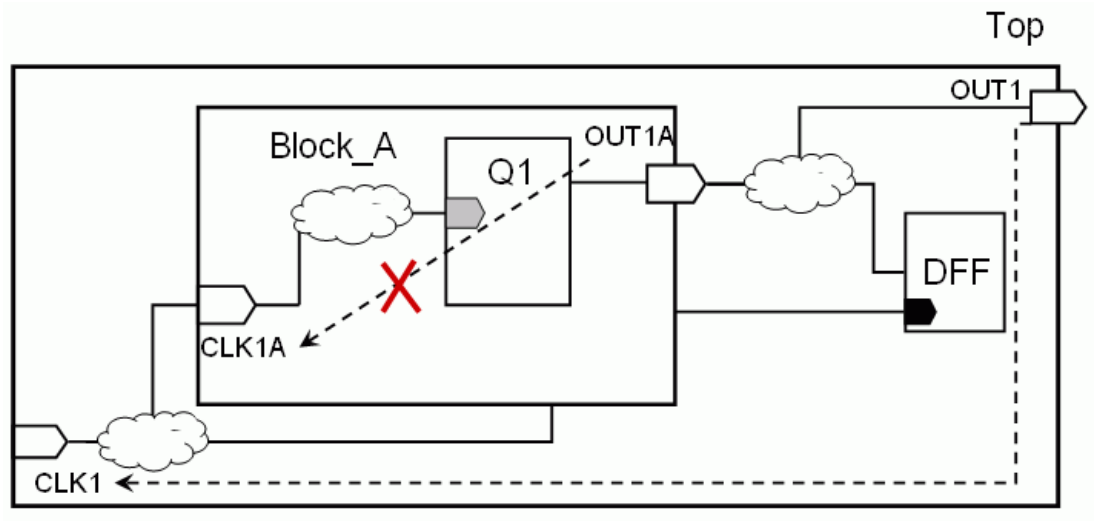
This is checked when you run the `run rule check` command.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

© 1999-2023 All Rights Reserved.

Example

The design in the following figure would cause a rule check violation, if `OUT1A` did not have an output delay versus `CLK1A`.



© 1999-2023
All Rights Reserved.

CCD_IO_HIER5i

Message

```
Missing set_input_delay in full-chip SDC
```

Default Severity

Warning

Description

Indicates that a `set_input_delay` in a sub-design (for example, a block or '`||`') has no corresponding `set_input_delay` at the full-chip (SDC design '`/`').

Only delays with `-clock` are considered. The corresponding delays are determined based on the clock equivalences (see the `SET CLOCK EQUIVALENCE` command) and the presence or absence of the `-clock_fall` option.

This is checked when you run the `run rule check` command.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Example

If a port `IN` has no input delay vs. top-level clock `CLK`, or if it is defined with `-clock_fall`, this rule will be reported if `IN` is connected to block port `BLK/IN`, the block SDC contains the command:

```
set_input_delay 1 -clock BCLK [get_ports IN]
```

and the dofile contains:

```
set clock equivalence CLK -sdc_design BLK BCLK
```


© 1999-2023 All Rights Reserved.

CCD_IO_HIER5o

Message

```
Missing set_output_delay in full-chip SDC
```

Default Severity

Warning

Description

Indicates that a `set_output_delay` in a sub-design (for example, a block or '| |') has no corresponding `set_output_delay` at the full-chip (SDC design '`/`').

Only delays with `-clock` are considered. The corresponding delays are determined based on the clock equivalences (see the `SET CLOCK EQUIVALENCE` command) and the presence or absence of the `-clock_fall` option.

This is checked when you run the `run rule check` command.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Example

If a port `OUT` has no output delay vs. top-level clock `CLK`, or if it is defined with `-clock_fall`, this rule will be reported if `OUT` is connected to block port `BLK/OUT`, the block SDC contains the command:

```
set_output_delay 1 -clock BCLK [get_ports OUT]
```

and the dofile contains:

```
set clock equivalence CLK -sdc_design BLK BCLK
```


© 1999-2023 All Rights Reserved.

CCD_IO_HIER6i

Message

```
Missing set_input_delay in block/glue SDC
```

Default Severity

Warning

Description

Indicates that a `set_input_delay` in the full-chip SDC has no corresponding `set_input_delay` in a sub-design (for example, a block or '||').

Only delays with `-clock` are considered. The corresponding delays are determined based on the clock equivalences (see the `SET CLOCK EQUIVALENCE` command) and the presence or absence of the `-clock_fall` option.

This is checked when you run the `run rule check` command.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Example

If a port `IN` has `set_input_delay` vs. top-level clock CLK, this rule will be reported if `IN` is connected to block port `BLK/IN`, and the block SDC does not contain the command:

```
set_input_delay 1 -clock BCLK [get_ports IN]
```

for any block clock `BCLK` such that the dofile contains:

```
set clock equivalence CLK -sdc_design BLK BCLK
```


© 1999-2023 All Rights Reserved.

CCD_IO_HIER6o

Message

```
Missing set_output_delay in block/glue SDC
```

Default Severity

Warning

Description

Indicates that a `set_output_delay` in the full-chip SDC has no corresponding `set_output_delay` in a sub-design (for example, a block or '| |').

Only delays with `-clock` are considered. The corresponding delays are determined based on the clock equivalences (see the `SET CLOCK EQUIVALENCE` command) and the presence or absence of the `-clock_fall` option.

This is checked when you run the `run rule check` command.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Example

If a port `OUT` has `set_output_delay` vs. top-level clock CLK, this rule will be reported if `OUT` is connected to block port `BLK/OUT`, and the block SDC does not contain the command:

```
set_output_delay 1 -clock BCLK [get_ports OUT]
```

for any block clock `BCLK` such that the dofile contains:

```
set clock equivalence CLK -sdc_design BLK BCLK
```


© 1999-2023 All Rights Reserved.

CCD_IO_HIER7i

Message

```
Block/glue set_input_delay has smaller value than in full-chip
```

Default Severity

Warning

Description

Indicates that a `set_input_delay` in a sub-design (for example, a block or '||') has a corresponding `set_input_delay` at the full-chip (SDC design '/'), which has a bigger value.

Only delays with `-clock` are considered. The corresponding delays are determined based on the clock equivalences (see the `SET CLOCK EQUIVALENCE` command) and the presence or absence of the `-clock_fall` option.

This is checked when you run the `run rule check` command.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Example

This rule will be reported if port `IN` is connected to block port `BLK/IN`, the full-chip SDC file contains the command:

```
set_input_delay 2 -clock CLK -clock_fall
```

the block SDC file contains the command:

```
set_input_delay 1 -clock BCLK [get_ports IN] -clock_fall
```

and the dofile contains

```
set clock equivalence CLK -sdc_design BLK BCLK
```


© 1999-2023 All Rights Reserved.

CCD_IO_HIER7o

Message

```
Block/glue set_output_delay has smaller value than in full-chip
```

Default Severity

Warning

Description

Indicates that a `set_output_delay` in a sub-design (for example, a block or '| |') has a corresponding `set_output_delay` at the full-chip (SDC design '`/`'), which has a bigger value.

Only delays with `-clock` are considered. The corresponding delays are determined based on the clock equivalences (see the `SET CLOCK EQUIVALENCE` command) and the presence or absence of the `-clock_fall` option.

This is checked when you run the `run rule check` command.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Example

This rule will be reported if port `OUT` is connected to block port `BLK/OUT`, the full-chip SDC file contains the command:

```
set_output_delay 2 -clock CLK -clock_fall
```

the block SDC file contains the command:

```
set_output_delay 1 -clock BCLK [get_ports OUT] -clock_fall
```

and the dofile contains

```
set clock equivalence CLK -sdc_design BLK BCLK
```


© 1999-2023 All Rights Reserved.

CCD_IO_HIER8i

Message

```
Block/glue set_input_delay has greater value than in full-chip
```

Default Severity

Warning

Description

Indicates that a `set_input_delay` in a sub-design (for example, a block or '||') has a corresponding `set_input_delay` at the full-chip (SDC design '/'), which has a smaller value.

Only delays with `-clock` are considered. The corresponding delays are determined based on the clock equivalences (see the `SET CLOCK EQUIVALENCE` command) and the presence or absence of the `-clock_fall` option.

Note: When this rule is reported for blocks, it does not necessarily mean there is an error, unless the blocks' ports are connected directly to top-level ports. This rule is provided for completeness, and can be most useful when comparing two SDC files for the same design using hierarchical checks between SDC designs '/' and 'll'.

This is checked when you run the `run rule check` command.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Example

This rule will be reported if port `IN` is connected to block port `BLK/IN`, the full-chip SDC file contains the command:

```
set_input_delay 1 -clock CLK -clock_fall
```

the block SDC file contains the command:

```
set_input_delay 2 -clock BCLK [get_ports IN] -clock_fall
```

and the dofile contains

© 1999-2023 All Rights Reserved.

```
set clock equivalence CLK -sdc_design BLK BCLK
```


© 1999-2023
All Rights Reserved.

CCD_IO_HIER8o

Message

```
Block/glue set_output_delay has greater value than in full-chip
```

Default Severity

Warning

Description

Indicates that a `set_output_delay` in a sub-design (for example, a block or '| |') has a corresponding `set_output_delay` at the full-chip (SDC design '`/`'), which has a smaller value.

Only delays with `-clock` are considered. The corresponding delays are determined based on the clock equivalences (see the `SET CLOCK EQUIVALENCE` command) and the presence or absence of the `-clock_fall` option.

This is checked when you run the `run rule check` command.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Example

This rule will be reported if port `OUT` is connected to block port `BLK/OUT`, the full-chip SDC file contains the command:

```
set_output_delay 1 -clock CLK -clock_fall
```

the block SDC file contains the command:

```
set_output_delay 2 -clock BCLK [get_ports OUT] -clock_fall
```

and the dofile contains

```
set clock equivalence CLK -sdc_design BLK BCLK
```


© 1999-2023 All Rights Reserved.

CCD_IO_HIER9

Message

```
Inter-block connection output/input delays under-constrain the path
```

Default Severity

Warning

Description

Indicates that a timing path crossing between two SDC designs is budgeted more delay than the total clock period.

At each point where the path crosses from one SDC design to another one, the output and input delays at both sides represent the delay allocated to the other side of the crossing. If these add up to less than the clock period, then the path is under-constrained.

The input and output delays to be correlated must be defined versus block clocks that have a common shared top-level equivalent clock, whose period will be used for the check. Therefore, this check requires the top-level SDC design to have clocks defined, and the block-top equivalences to be correctly extracted by the tool, or specified in the dofile using the `SET CLOCK EQUIVALENCE` command. To report clock equivalences, use `REPORT CLOCK -hier_equiv`.

This rule is checked when you run the `RUN RULE CHECK` command and is checked at the top-level. To specify the current SDC design, use the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command or when reading timing constraints with `READ HIERARCHICAL SDC`.

Note: Currently, only block-to-block crossings are checked. Zero-delay interconnect is assumed, and timing exceptions are not considered.

Example

For example:

- There is a connection from `B1/OUT` to `B2/IN`
- `B1.sdc` has: `set_output_delay 1 -clock bclk1 OUT`

© 1999-2023 All Rights Reserved.

- `B2.sdc` has: `set_input_delay 2 -clock bclk2 IN`
- `TOP.sdc` has: `create_clock tclk -period 4`
- `tclk` is the top-level equivalent clock of `B1/bclk1` and of B2/bclk2

In this case, the first part of the path can have delay 3, and the second part can have delay 2, for a total of 5 (more than the clock period). Since the sum of the input and output delay is less than 4, CCD_IO_HIER9 will be reported.

© 1999-2023 All Rights Reserved.

CCD_IO_HIER10

Message

```
Inter-block connection output/input delays over-constrain the path
```

Default Severity

Warning

Description

Indicates that a timing path crossing between two SDC designs is budgeted less delay than the total clock period.

At each point where the path crosses from one SDC design to another one, the output and input delays at both sides represent the delay allocated to the other side of the crossing. If these add up to more than the clock period, then the path is under-constrained.

The input and output delays to be correlated must be defined versus block clocks that have a common shared top-level equivalent clock, whose period will be used for the check. Therefore, this check requires the top-level SDC design to have clocks defined, and the block-top equivalences to be correctly extracted by the tool, or specified in the dofile using the `SET CLOCK EQUIVALENCE` command. To report clock equivalences, use `REPORT CLOCK -hier_equiv`.

This is checked when you run the `RUN RULE CHECK` command and at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command or when reading timing constraints with `READ HIERARCHICAL SDC`.

Note: Currently, only block-to-block crossings are checked. Zero-delay interconnect is assumed, and timing exceptions are not considered.

Example

For example:

- There is a connection from `B1/OUT` to `B2/IN`
- B1.sdc has: `set_output_delay 3 -clock bclk1 OUT`

© 1999-2023 All Rights Reserved.

- B2.sdc has: `set_input_delay 2 -clock bclk2 IN`
- TOP.sdc has: `create_clock tclk -period 4`
- `tclk` is top-level equivalent clock of `B1/bclk1` and of `B2/bclk2`

In this case, the first part of the path can have delay 1, and the second part can have delay 2, for a total of 3 (less than the clock period).

Since the sum of the input and output delay is more than 4, CCD_IO_HIER10 will be reported.

© 1999-2023 All Rights Reserved.

CCD_IO_INT*

This section describes rule checks that relate to SDC integration for input/output constraints.

© 1999-2023 All Rights Reserved.

CCD_IO_INT1

Message

`SDC integration: Input/output delays`

Default Severity

Warning

Description

Configures the integration of `set_input_delay` and `set_output_delay` commands, and reports the messages generated during this integration process.

This is checked when you run the `INTEGRATE` command.

© 1999-2023 All Rights Reserved.

CCD_EXC_FLP*

This section describes rule checks that relate to violations for false path exceptions.

© 1999-2023 All Rights Reserved.

CCD_EXC_FLP1

Message

`False path exception does not match any timing path`

Default Severity

Warning

Description

Indicates that a `set_false_path` command does not match any timing path in the design, i.e., a connected path that is not interrupted by any constant value that propagates to a design object on the path.

Note: One of the cases in which a false-path exception can be reported by this rule is when there is no physical connection among the specified points. In this case, the exception will appear as "No-Path" in the Exception Validation reports. On the other hand, if this rule reports an exception because it does not match any statically sensitizable path due to propagated constants, the Exception Validation report will validate it and show it as Pass.

This is checked when you run the `run rule check` command.

Examples

If the only path from register A to register B is defined by the assignment `A = B & IN`, and the constraint `set_case_analysis 0 [get_ports IN]` is specified, the Conformal Constraint Designer generates a warning for the following command:

```
set_false_path -from A -to B
```

If there is no path from `pi[0]` to `a_reg[1]` (e.g., if there is a typo in second command because it was supposed to affect paths from `pi[1]` to `a_reg[1]`), this rule will be flagged.

```
set_false_path -from [get_ports {pi[0]}] -to [get_cells {a_reg[0]}]
set_false_path -from [get_ports {pi[0]}] -to [get_cells {a_reg[1]}]
```


© 1999-2023 All Rights Reserved.

CCD_EXC_FLP2

Message

`Logical pin referenced in false path`

Default Severity

Warning

Description

Indicates that you referenced a logical pin in a false path.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

In the following example, the Conformal Constraint Designer issues a warning if `mod` is a hierarchical module instance:

```
set_false_path -setup -from IN1 -to mod/sel
```


© 1999-2023 All Rights Reserved.

CCD_EXC_FLP3

Message

`False path does not have both -from and -to`

Default Severity

Warning

Description

Indicates a `set_false_path` command that does not have both `-from` and `-to` arguments.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

```
set_false_path -from [get_cells conf_reg_1]
```


© 1999-2023 All Rights Reserved.

CCD_EXC_FLP6

Message

```
Connected asynchronous clocks have not been declared as set_false_path -from clock
     -to clock
```

Default Severity

Warning

Description

Indicates that for pairs of clocks that do not belong to the same clock group, you did not specify `set_false_path -from` *`clock`* `-to clock`.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

If `CLK1`, `CLK2`, and `CLK3` are declared as asynchronous domains, and you have the following commands:

```
set_false_path -from CLK1 -to CLK2
set_false_path -from CLK2 -to CLK1
```

You would get an warning if you do not set a false path for `CLK3`.

© 1999-2023 All Rights Reserved.

CCD_EXC_FLP7

Message

```
Clocks declared as set_false_path -from clock -to clock (in both directions) are
     in the same clock group
```

Default Severity

Warning

Description

Indicates that for a pair of clocks (C1, C2) that belong to the same clock group, you specified both `set_false_path -from C1 -to C2` and `set_false_path -from C2 -to C1`.

This rule will not be flagged if any combination of `-setup/-hold` with `-rise/-fall` is not covered by the `set_false_path` commands, or if they have `-through` options.

This is checked when you run the `run rule check` command.

Example

The `dofile` contains:

```
add clock group -single
```

The `sdc` file contains:

```
set_false_path -from CLK1 -to CLK2
set_false_path -from CLK2 -to CLK1
set_false_path -from CLK1 -to CLK3
set_false_path -hold -from CLK2 -to CLK3
set_false_path -from CLK3 -to CLK2
```

This rule will report the clock pair (`CLK1`, `CLK2`), but it will not report the pairs (`CLK1`, `CLK3`) or (`CLK2`, `CLK3`).

© 1999-2023 All Rights Reserved.

CCD_EXC_MCP*

This section describes rule checks that relate to violations for multi-path exceptions.

© 1999-2023 All Rights Reserved.

CCD_EXC_MCP1

Message

`Multi-cycle path exception does not match any timing path`

Default Severity

Warning

Description

Indicates that a `set_multicycle_path` command does not match any timing path in the design, i.e., a connected path that is not interrupted by any constant value that propagates to a design object on the path.

This is checked when you run the `run rule check` command.

Example

The Conformal Constraint Designer generates a warning if the paths from `din` into `mem` are disabled by a constant 0 at `write_en`:

```
set_case_analysis 0 write_en
set_multicycle_path 2 -from [get_ports {din[*]}] -to [get_cells {mem[*]}]
```

If there is no path from `pi[0]` to `a_reg[1]` (e.g., if there is a typo in second command because it was supposed to affect paths from `pi[1]` to `a_reg[1]`), this rule will be flagged.

```
set_multicycle_path 2 -from [get_ports {pi[0]}] -to [get_cells {a_reg[0]}]
set_multicycle_path 2 -from [get_ports {pi[0]}] -to [get_cells {a_reg[1]}]
```


© 1999-2023 All Rights Reserved.

CCD_EXC_MCP2

Message

`Logical pin referenced in multi-cycle path`

Default Severity

Warning

Description

Indicates that you referenced logical pins in a multi-cycle path.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The `set_multicycle_path` command must reference ports, registers, leaf cells, or clocks. In the following example, the Conformal Constraint Designer issues a warning if `mod/out` is a is a hierarchical module instance:

```
set_multicycle_path 2 -setup -through mod/out
```


© 1999-2023 All Rights Reserved.

CCD_EXC_MCP3

Message

`Multi-cycle path does not have both -from and -to`

Default Severity

Warning

Description

Indicates a `set_multicycle_path` command that does not have both `-from` and `-to` arguments.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

```
set_multicycle_path 2 -through [get_pins mul/and2/Z]
```


© 1999-2023 All Rights Reserved.

CCD_EXC_MCP5

Message

```
Incomplete multi-cycle path options
```

Default Severity

Warning

Description

Indicates that `set_multicycle_path` has incomplete options. You need to have values for both or none of the following pairs of options: `-rise`/`-fall` and `-setup`/`-hold`.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

```
set_multicycle_path 2 -setup -from IN1 -to -FF1
set_multicycle_path 2 -setup -from FF1 -to OUT1
set_multicycle_path 2 -hold -from FF1 -to OUT1
```

The Conformal Constraint Designer generates a warning, if the following command is missing:

```
set_multicycle_path 1 -hold -from IN1 -to FF1
```


© 1999-2023 All Rights Reserved.

CCD_EXC_MCP6

Message

```
Multi-cycle path setup/hold multiplier is over/under defined.
```

Default severity

Warning

Description

The setup multiplier of a multi-cycle path should always at least one greater than that of a hold multiplier. This rule indicates a multiplier inconsistency of a multi-cycle path between setup and hold corners.

Correct the multiplier of the `set_multicycle_paths` timing constraint.

This rule is disabled by default. To enable, use the `ADD RULE INSTANCE` command. It is run when a timing constraint file is read in using the `READ SDC` command.

Example

The case reports this rule because multicycle path from input port 'in' to output port 'out' has setup multiplier 4 which is not at least one greater than hold multiplier 4.

```
set_multicycle_path -setup 4 -from [get_ports in] -to [get_ports out]
set_multicycle_path -hold 4 -from [get_ports in] -to [get_ports out]
```


© 1999-2023 All Rights Reserved.

CCD_EXC_SMD*

This section describes rule checks that relate to setting min/max delay exceptions.

© 1999-2023 All Rights Reserved.

CCD_EXC_SMD1

Message

`set_max_delay exception does not match any timing path`

Default Severity

Warning

Description

Indicates that a `set_max_delay` command does not match any timing path in the design, i.e., a connected path that is not interrupted by any constant value that propagates to a design object on the path.

This is checked when you run the `run rule check` command.

Example

The Conformal Constraint Designer generates a warning if the paths from `din` into `mem` are disabled by a constant `0` at `write_en`:

```
set_case_analysis 0 write_en
set_max_delay 2.0 -from [get_ports {din[*]}] -to [get_cells {mem[*]}]
```

If there is no path from `pi[0]` to `a_reg[1]` (e.g., if there is a typo in second command because it was supposed to affect paths from `pi[1]` to `a_reg[1]`), this rule will be flagged.

```
set_max_delay 2.0 -from [get_ports {pi[0]}] -to [get_cells {a_reg[0]}]
set_max_delay 2.0 -from [get_ports {pi[0]}] -to [get_cells {a_reg[1]}]
```


© 1999-2023 All Rights Reserved.

CCD_EXC_SMD2

Message

`set_min_delay exception does not match any timing path`

Default Severity

Warning

Description

Indicates that a `set_min_delay` command does not match any timing path in the design, i.e., a connected path that is not interrupted by any constant value that propagates to a design object on the path.

This is checked when you run the `run rule check` command.

Example

The Conformal Constraint Designer generates a warning if the paths from `din` into `mem` are disabled by a constant `0` at `write_en`:

```
set_case_analysis 0 write_en
set_min_delay 2.0 -from [get_ports {din[*]}] -to [get_cells {mem[*]}]
```

If there is no path from `pi[0]` to `a_reg[1]` (e.g., if there is a typo in second command because it was supposed to affect paths from `pi[1]` to `a_reg[1]`), this rule will be flagged.

```
set_min_delay 2.0 -from [get_ports {pi[0]}] -to [get_cells {a_reg[0]}]
set_min_delay 2.0 -from [get_ports {pi[0]}] -to [get_cells {a_reg[1]}]
```


© 1999-2023 All Rights Reserved.

CCD_EXC_SMD3

Message

```
Incomplete set_min/max_delay options
```

Default Severity

Warning

Description

Indicates that a `set_max_delay` or `set_min_delay` has been specified for `-rise` but not for `-fall`, or vice versa.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

For the following:

```
set_max_delay 5 -rise -from [get_ports in1] -to [get_ports out1]
set_min_delay 6 -fall -from [get_ports in2]
set_min_delay 5 -rise  -to [get_ports out2]
```

The Conformal Constraint Designer generates warnings if the following commands are missing:

```
set_max_delay <value> -fall -from [get_ports in1] -to [get_ports out1]
set_min_delay <value> -rise -from [get_ports in2]
set_min_delay <value> -fall  -to [get_ports out2]
```


© 1999-2023 All Rights Reserved.

CCD_EXC_SDT*

This section describes rule checks that relate to disable timing settings.

© 1999-2023 All Rights Reserved.

CCD_EXC_SDT1

Message

`A timing loop is not cut by any set_disable_timing`

Default Severity

Warning

Description

Indicates that, after all the `set_disable_timing` from the SDC files have been applied, there is a timing loop that is not broken by any of them.

Note: Conformal Constraint Designer will break the loop at an arbitrary place. This is also done by timing engines before delay propagation, and it can cause the wrong paths to be disabled. You should add `set_disable_timing` commands as needed to remove all timing loops.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command. The schematic displays the path leading to the loop, the loop itself, and the part of the loop that is being disabled.

Example

If there is a path A-->B-->C-->D-->E-->F-->C, and there is no `set_disable_timing` command that disables any of the pins or cells in the loop (C,D,E,F,C), the Conformal Constraint Designer will report this loop.

In the diagnosis schematic window, the parts of the path from C to F and from F to C will appear in different colors, showing which parts of the path are in a loop and which one is disabled.

© 1999-2023 All Rights Reserved.

CCD_EXC_OLP*

This section describes rule checks that relate to overlapping timing exceptions.

© 1999-2023 All Rights Reserved.

CCD_EXC_OLP1a

Message

```
A set_false_path and a set_multicycle_path command match the same timing path
```

Default Severity

Warning

Description

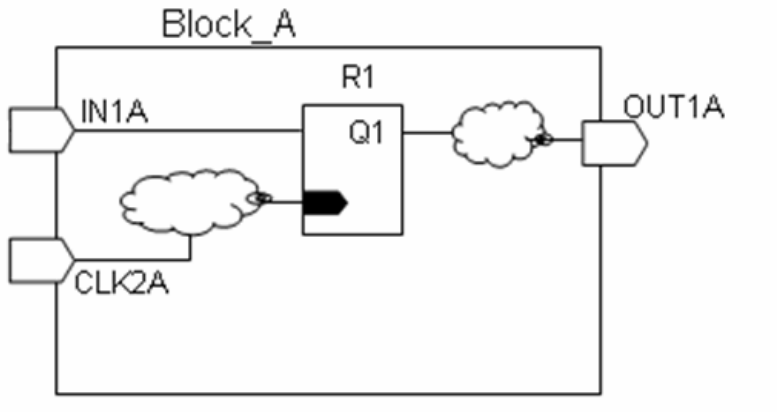
Reports any two timing exception commands, one `set_false_path` and one `set_multicycle_path`, that apply to a common timing path.

Note: This message does not imply that any of the two reported exceptions are necessarily redundant.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example



The following commands cause a rule check violation.

```
set_false_path -from [get_ports IN1A]
set_multicycle_path 2 -to [get_cells R1]
```


© 1999-2023
All Rights Reserved.

CCD_EXC_OLP1b

Message

```
A set_false_path and a set_max/min_delay command match the same timing path
```

Default Severity

Warning

Description

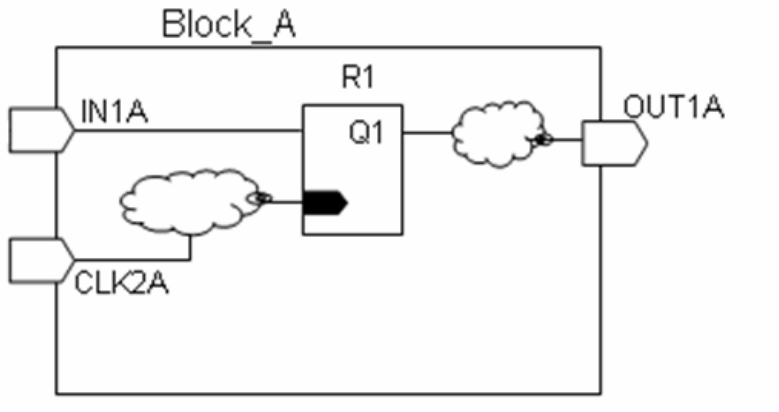
Reports any two timing exception commands, one `set_false_path` and one `set_max_delay`/`set_min_delay`, that apply to a common timing path.

Note: This message does not imply that any of the two reported exceptions are necessarily redundant.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example



The following commands cause a rule check violation.

```
set_false_path -from [get_ports IN1A]
set_max_delay 2 -to [get_cells R1]
```


© 1999-2023
All Rights Reserved.

CCD_EXC_OLP1c

Message

```
A set_max/min_delay and a set_multicycle_path command match the same timing path
```

Default Severity

Warning

Description

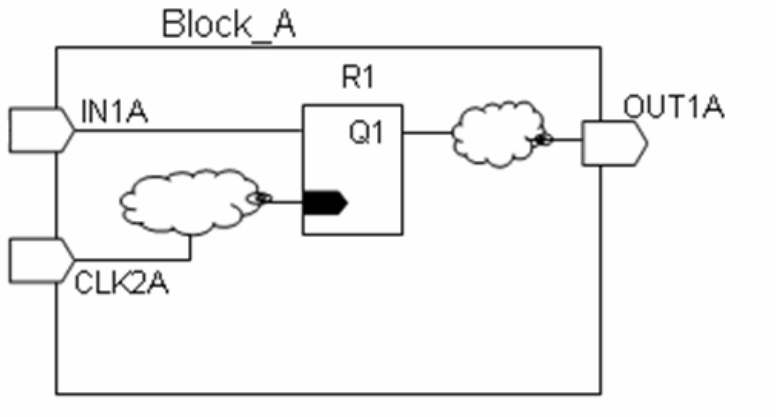
Reports any two timing exception commands, one `set_multicycle_path` and one `set_max_delay`/`set_min_delay`, that apply to a common timing path.

Note: This message does not imply that any of the two reported exceptions are necessarily redundant.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example



The following commands cause a rule check violation.

```
set_multicycle_path 3 -from [get_ports IN1A]
set_max_delay 2 -to [get_cells R1]
```


© 1999-2023
All Rights Reserved.

CCD_EXC_OLP1d

Message

`Two set_false_path commands match the same timing path`

Default Severity

Warning

Description

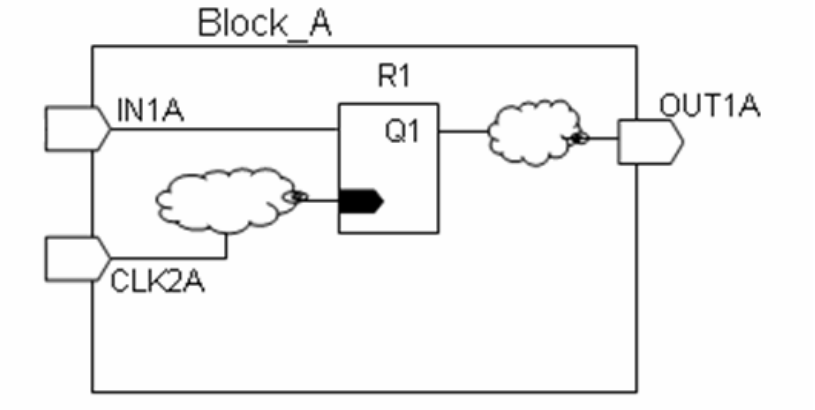
Reports any two `set_false_path` commands that apply to a common timing path.

Note: This message does not imply that any of the two reported exceptions are necessarily redundant.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example



The following commands cause a rule check violation.

```
set_false_path -from [get_ports IN1A]
set_false_path -to [get_cells R1]
```


© 1999-2023
All Rights Reserved.

CCD_EXC_OLP1e

Message

```
Two set_multicycle_path commands match the same timing path
```

Default Severity

Warning

Description

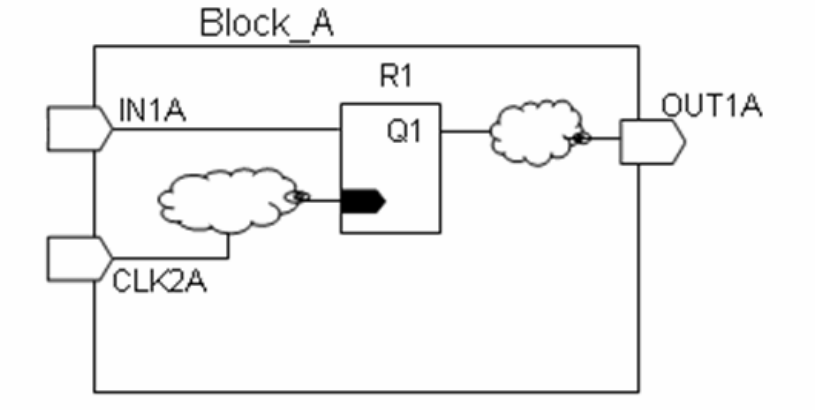
Reports any two `set_multicycle_path` commands that apply to a common timing path.

Note: This message does not imply that any of the two reported exceptions are necessarily redundant.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example



The following commands cause a rule check violation.

```
set_multicycle_path 3 -from [get_ports IN1A]
set_multicycle_path 2 -to [get_cells R1]
```


© 1999-2023
All Rights Reserved.

CCD_EXC_OLP1f

Message

`Two set_max/min_delay commands match the same timing path`

Default Severity

Warning

Description

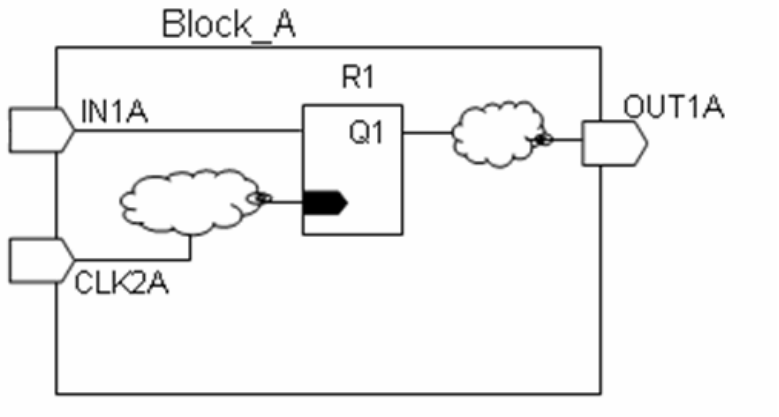
Reports any two `set_max_delay`/`set_min_delay` commands that apply to a common timing path.

Note: This message does not imply that any of the two reported exceptions are necessarily redundant.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example



The following commands cause a rule check violation.

```
set_max_delay 3 -from [get_ports IN1A]
set_max_delay 2 -to [get_cells R1]
```


© 1999-2023 All Rights Reserved.

CCD_EXC_OLP2

Message

`A timing exception is overridden on all paths by exceptions of higher priority`

Default Severity

Warning

Description

Reports any timing exception command such that for every matching timing path that is not tied to a constant value, there is a different timing exception with higher priority that matches the same path.

Note: This message is a strong indicator that the reported exception could be redundant. However, the results of this check are correct only if the design's timing paths are constrained properly, and if the check runs to completion. You should exercise care when deciding to remove a timing exception reported by this rule.

This is checked when you run the `run rule check` command.

Note: This rule cannot be diagnosed using the `DIAGNOSE RULE CHECK` command. However, in the GUI, the SDC Command Browser will display the related occurrences of `CCD_EXC_OLP1`, which can help diagnose this rule.

Example

If there is a path from `r_c_reg` to `x_reg` that traverses the net `mul`, this rule reports the first of the following commands:

```
set_multicycle_path -through [get_nets mul] -setup 2
set_false_path -from [get_cells r_c_reg]
set_false_path -to [get_cells x_reg]
```

Note: The two last commands have the same meaning for the path mentioned above, so none of them will be reported by this rule.

© 1999-2023 All Rights Reserved.

CCD_EXC_HIER*

This section describes rules that perform consistency checks between timing exceptions that are applied to blocks and timing exceptions that are defined for the top-level design.

Note: An occurrence of these rules does not necessarily indicate an error. For these rules, each message shows an instance where a timing path within a block corresponds to a top-level timing path (or a part of a top-level timing path), and the two paths are not affected by the same timing exception.

© 1999-2023 All Rights Reserved.

CCD_EXC_HIER1

Message

`A block path without set_false_path is part of a top-level false path`

Default Severity

Warning

Description

Indicates that a block path, which was not declared as false at the block level, has been affected by a top-level FP specification.

This is checked when you run the `run rule check` command.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

If `Block_A.sdc` has this line:

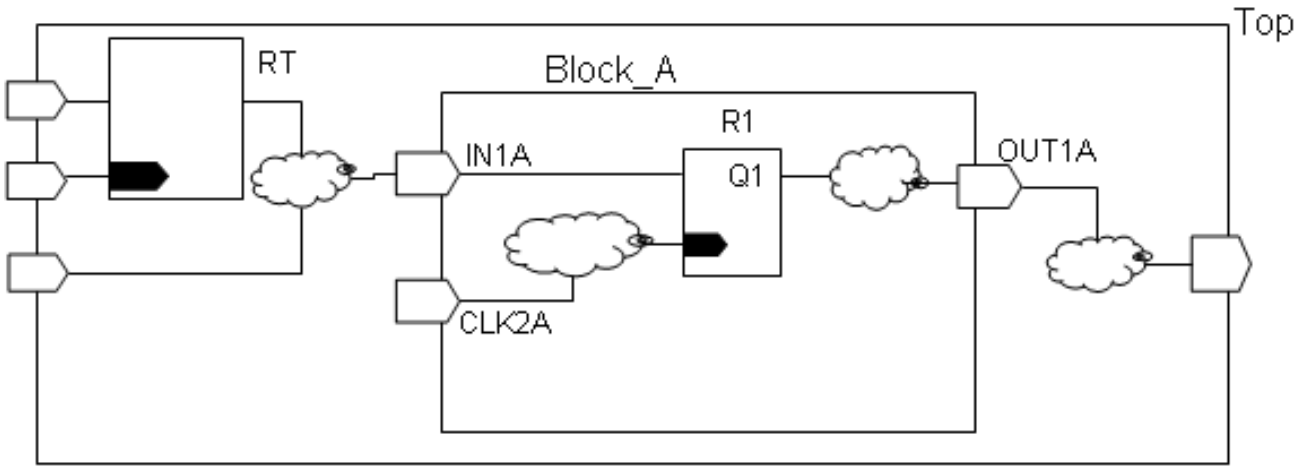
```
set_false_path -from R1
```

and `Top.sdc` has this line:

```
set_false_path -through Block_A/IN1A
```


© 1999-2023 All Rights Reserved.

Conformal CD issues a warning because the block-level did not consider the path from `IN1A` to `R1` as false, even if there is an FP specification for the top level.



© 1999-2023
All Rights Reserved.

CCD_EXC_HIER2

Message

`A block's false path is part of a top-level path without set_false_path`

Default Severity

Warning

Description

Indicates that a block-level FP is not declared as an FP at the top level.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

If `Block_A.sdc` has this line:

```
set_false_path -from R1
```

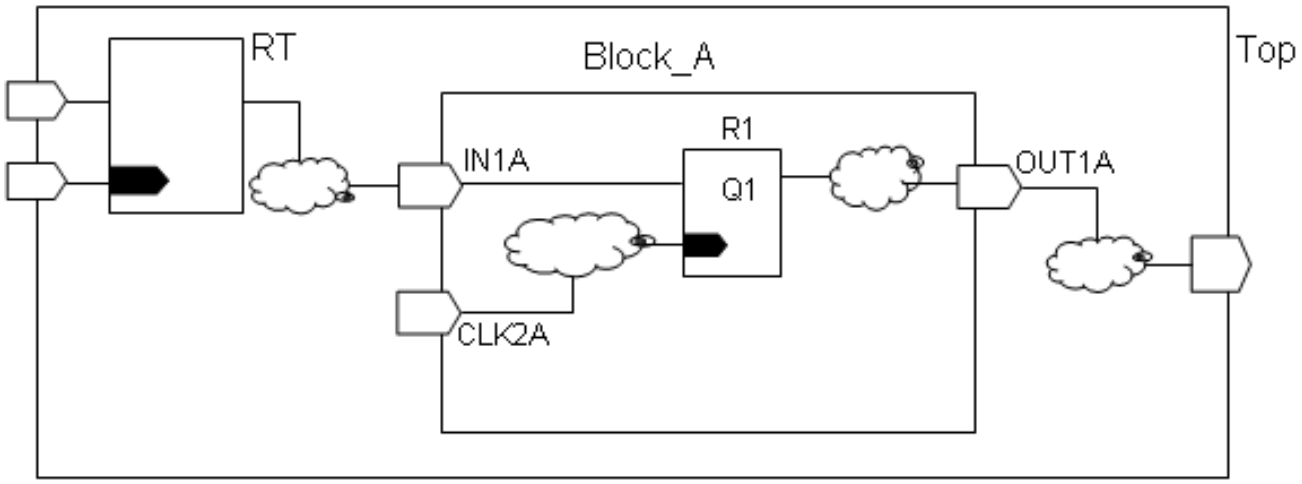
and `Top.sdc` has this line:

```
set_false_path -from RT
```

If the following line is missing from `Top.sdc`, then Conformal CD issues a warning because the block level would consider the path from `R1` to `OUT1A` as false, but the top level does not specify this as false:

© 1999-2023 All Rights Reserved.

```
set_false_path -from BlockA/R1
```



© 1999-2023
All Rights Reserved.

CCD_EXC_HIER3

Message

`A block's false path is part of a top-level multicycle path`

Default Severity

Warning

Description

Indicates that an FP at the block level is declared as an MCP at the top level.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

If `Block_A.sdc` has this line:

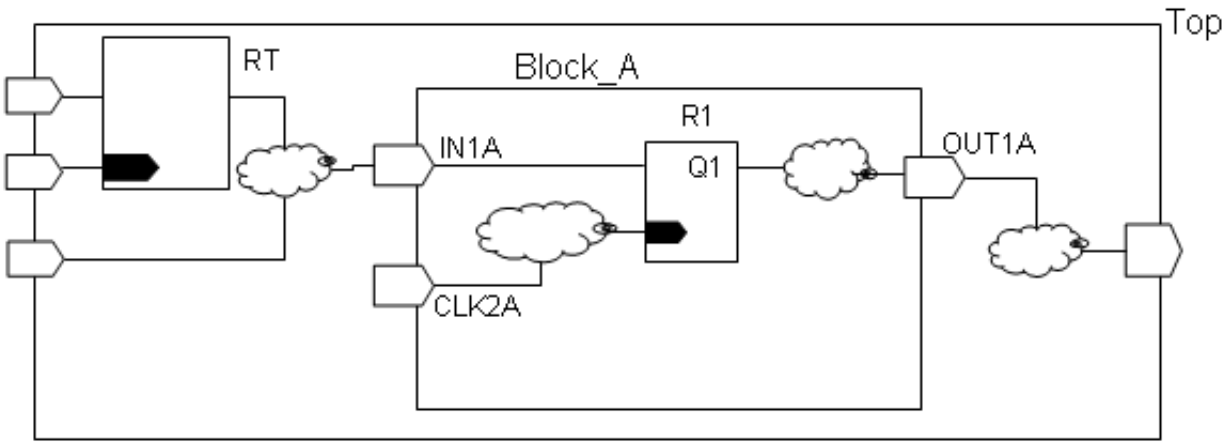
```
set_false_path -through IN1A
```

and `Top.sdc` has this line:

```
set_multicycle_path 2 -through Mul
```


© 1999-2023 All Rights Reserved.

Conformal CD issues a warning because the FP from port `IN1A` is now considered an MCP because of the `-through Mul` specified in `Top.sdc`.



© 1999-2023
All Rights Reserved.

CCD_EXC_HIER4

Message

`A block's multicycle path is part of a top-level false path`

Default Severity

Warning

Description

Indicates that a block's MCP is declared as an FP at the top level.

Note: Checks for each combination of `-setup`/`-hold` and `-rise`/`-fall` separately.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

If `Block_A.sdc` has this line:

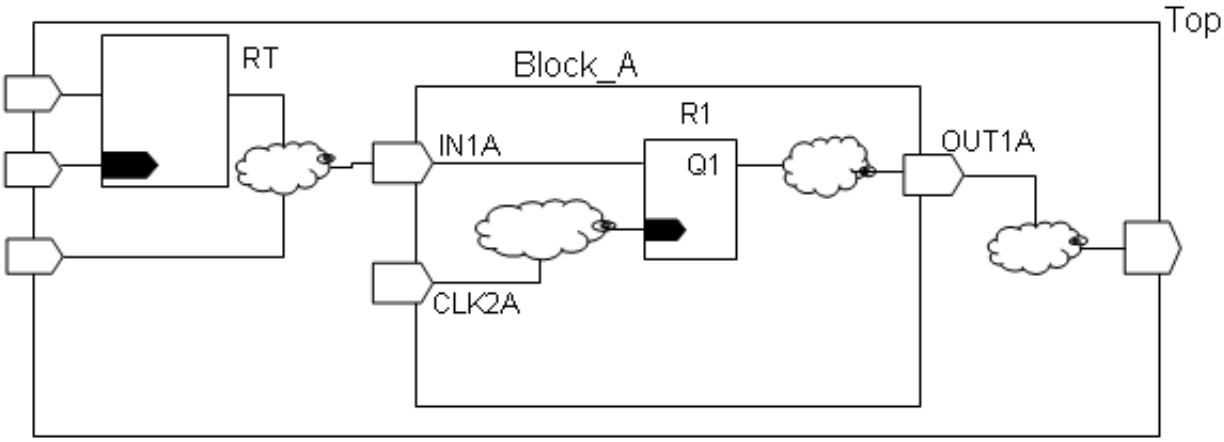
```
set_multicycle_path 2 -through IN1A
```

and `Top.sdc` has this line:

```
set_false_path -through Mul
```


© 1999-2023 All Rights Reserved.

Conformal CD issues a warning because the MCP from port `IN1A` is now considered an FP because of the `-through Mul` specified in `Top.sdc`.



© 1999-2023
All Rights Reserved.

CCD_EXC_HIER5

Message

`A block's single cycle path is part of a top-level multicycle path`

Default Severity

Warning

Description

Indicates that a block that is not declared as an MCP, is part of a top-level path that is declared as an MCP at the top-level SDC file.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

If `Block_A.sdc` does not have this line:

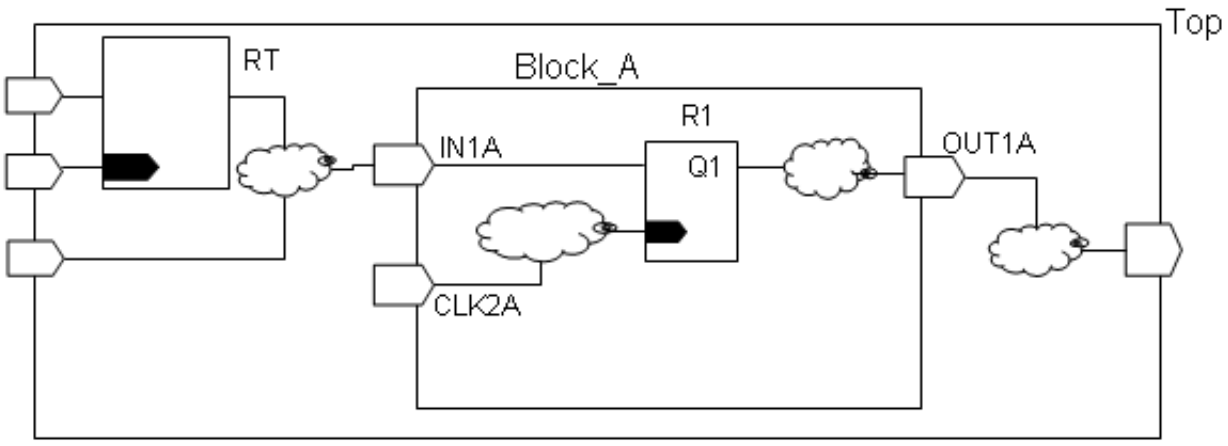
```
set_multicycle_path 2 -through IN1A
```

`Top.sdc` has this line:

```
set_multicycle_path 2 -through Mul
```


© 1999-2023 All Rights Reserved.

Conformal CD issues a warning because the block-level path from port `IN1A` to `R1` is now considered an MCP path because of the `-through Mul` specified in `Top.sdc`.



© 1999-2023
All Rights Reserved.

CCD_EXC_HIER6

Message

```
A block's multicycle path is part of a top-level single cycle path
```

Default Severity

Warning

Description

Indicates that an MCP in the block is not declared as an MCP at the top level.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

If `Block_A.sdc` has this line:

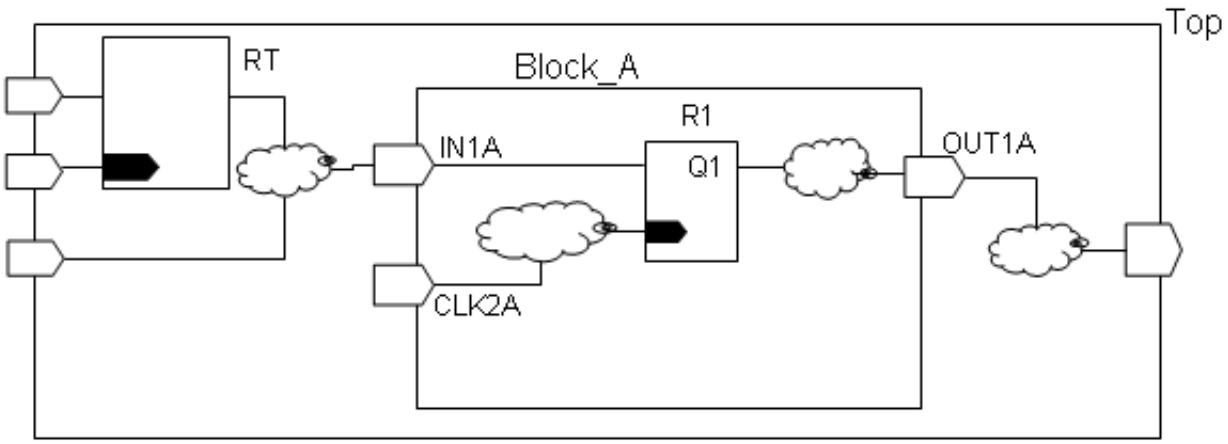
```
set_multicycle_path 2 -through IN1A
```

`Top.sdc` does not have this line:

```
set_multicycle_path 2 -from Block_A/IN1A
```


© 1999-2023 All Rights Reserved.

Conformal CD issues a warning because the block-level MCP path from port `IN1A` to `R1` is now considered a single-cycle path.



© 1999-2023
All Rights Reserved.

CCD_EXC_HIER7

Message

```
A block's multicycle path is part of a top-level multicycle path with a different
     cycle count
```

Default Severity

Warning

Description

Indicates that there are conflicting cycle counts between the block and top-level MCPs.

This is checked when you run the RUN RULE CHECK command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

If `Block_A.sdc` has this line:

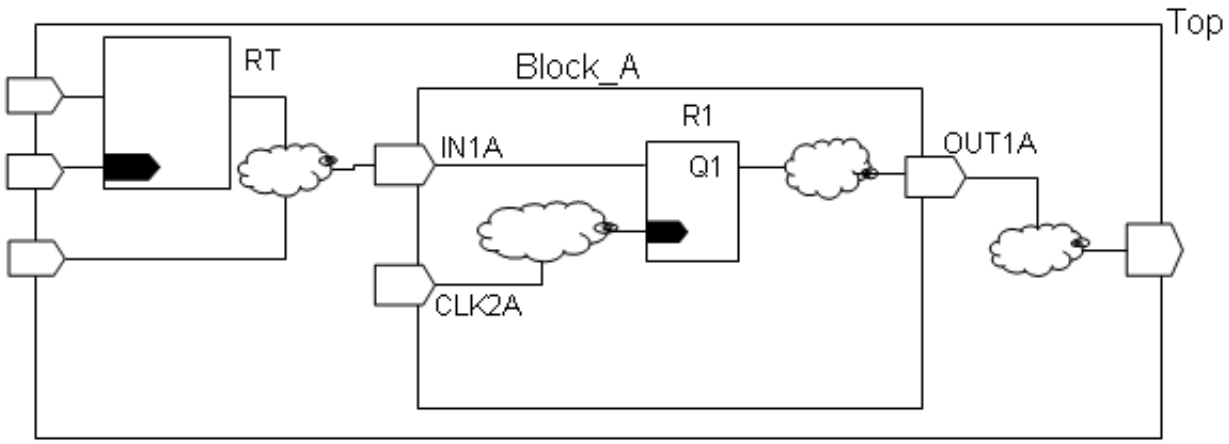
```
set_multicycle_path 2 -setup -from Q1 -to OUT1A
```

`Top.sdc` has this line:

```
set_multicycle_path 3 -setup -from Block_A/Q1 -to Block_A/OUT1A
```


© 1999-2023 All Rights Reserved.

Conformal CD issues a warning because of conflicting cycle counts between `Block_A.sdc` and `Top.sdc`.



© 1999-2023
All Rights Reserved.

CCD_EXC_HIER8

Message

`Inconsistent set_disable_timing at the top-level vs. block`

Default Severity

Warning

Description

Indicates that `set_disable_timing` is used at the block, but not at the top level, or vice versa. For `set_disable_timing` on library cell or library pin objects, the check is done only if the block instantiates the library cell.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Examples

1. `Block_A.sdc` has this line:

   ```
   set_disable_timing [get_cells and234]
   ```

   but top.sdc does not have this line:

   ```
   set_disable_timing [get_cells Block_A/and234]
   ```

   The Conformal software issues a warning because `set_disable_timing` is specified at the block, but not at the top level.

2. `top.sdc` has this line:

   ```
   set_disable_timing [get_lib_pins MYLIB/MYDFF/SEN]
   ```

   but `Block_A.sdc` does not have the same line.

© 1999-2023 All Rights Reserved.

The Conformal software issues a warning only if `BlockA` has an instance of cell `MYDFF` from library `MYLIB`.

© 1999-2023 All Rights Reserved.

CCD_EXC_HIER9

Message

```
A fragment of block's timing path cannot be matched to a corresponding fragment of
     top-level timing path
```

Default Severity

Warning

Description

Indicates that the software was not able to identify a fragment of the path in the top-level that includes, or is identical to, the fragment of the path in the block-level.

This rule can be reported as a consequence of inconsistent `set_disable_timing` between the chip level and the sub-designs. That problem would be reported by rule `CCD_EXC_HIER8`. It can also be caused by an inconsistency in the automatic loop-cutting done at the block and chip level, as reported by `CCD_EXC_SDT1` in each SDC design.

In both cases, make sure that there are sufficient `set_disable_timing` commands at both levels that are consistent, eliminating all occurrences of `CCD_EXC_SDT1` (at each SDC design) and `CCD_EXC_HIER8`.

If there are still occurrences of `CCD_EXC_HIER9`, this could be because of an incorrect definition of clock equivalences (see `SET CLOCK EQUIVALENCE` and `REPORT CLOCK -hier`), or inconsistent application of SDC constraints that cause path splitting (such as `set_input_delay`) between the block and the chip level.

The occurrence of `CCD_EXC_HIER9` points to an inconsistency between the timing graphs being compared, which can negatively affect the quality of results of the hierarchical checks.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

1. A block path ends in an output port of that block, which is not connected to any timing endpoint in the top-level design.

© 1999-2023 All Rights Reserved.

2. The clock driving the startpoint of a block path is not mapped to any top-level clock.
3. There is a `set_disable_timing` only in the top-level SDC, affecting paths through a block.

© 1999-2023 All Rights Reserved.

CCD_EXC_HIER10

Message

```
A block path without set_max_delay/set_min_delay is part of a top-level path with
     max/min delay settings
```

Default Severity

Warning

Description

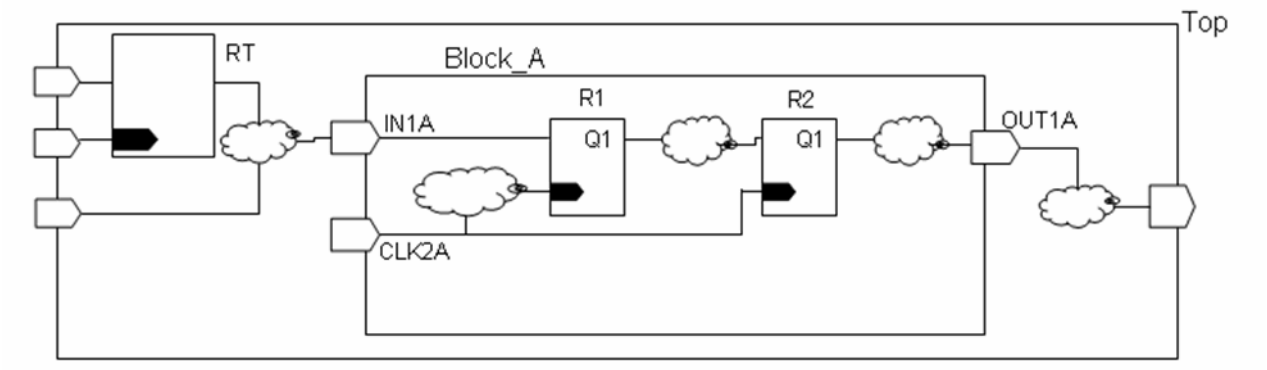
Indicates that a block path, which was not set any maximum or minimum delay at the block level, has been set by a top-level maximum or minimum delay.

This is checked when you run the `run rule check` command.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `ADD SDC DESIGN` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example



The following command in the `Top.sdc` causes a rule check violation.

```
set_max_delay 4 -from [get_cells Block_A/R1] -to [get_cells Block_A/R2]
```


© 1999-2023
All Rights Reserved.

To fix this, add the following command to the `Block_A`.sdc file:

```
set_max_delay 4 -from [get_cells R1] -to [get_cells R2]
```


© 1999-2023 All Rights Reserved.

CCD_EXC_HIER11

Message

```
A block's path with max/min delay settings is part of a top-level path without
     set_max_delay/set_min_delay
```

Default Severity

Warning

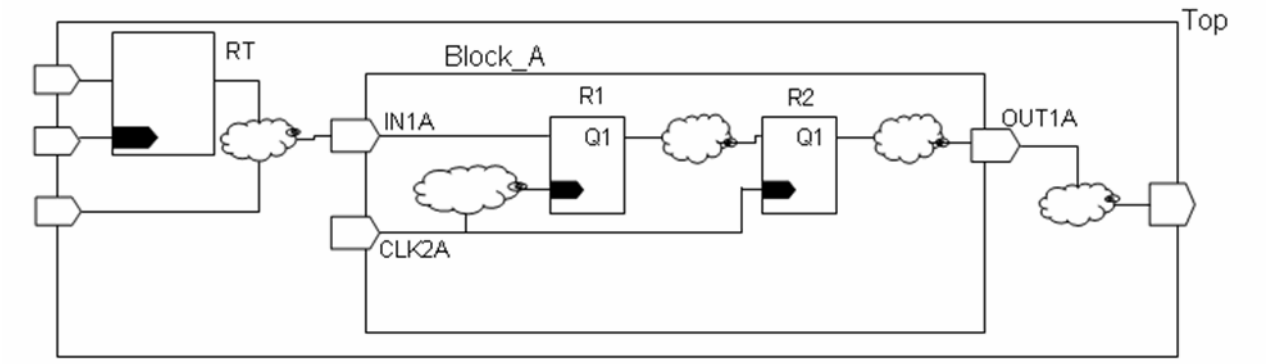
Description

Indicates that a block path, which was set a maximum or minimum delay at the block level, has not been set with any top-level maximum or minimum delay.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example



The following command in the `Block_A.sdc` causes a rule check violation:

```
set_max_delay 4 -from [get_cells R1] -to [get_cells R2]
```

To fix this, add the following command to the `Top.sdc` file:

```
set_max_delay 4 -from [get_cells Block_A/R1] -to [get_cells Block_A/R2]
```


© 1999-2023
All Rights Reserved.

CCD_EXC_HIER12

Message

```
A block's false path is part of a top-level path with max/min delay settings
```

Default Severity

Warning

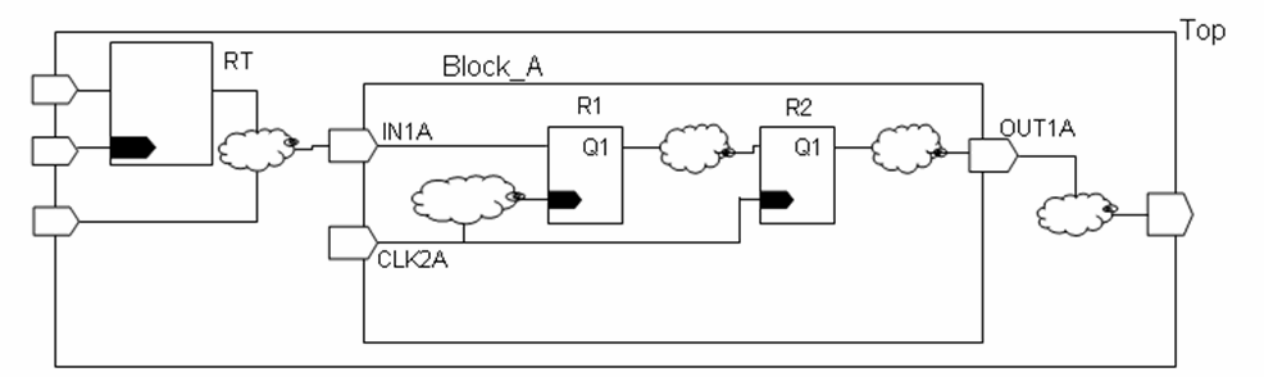
Description

Indicates that a block path, which was declared as false at the block level, has been set by a top-level maximum or minimum delay.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example



The following commands cause a rule check violation:

(in `Top.sdc`)

```
set_max_delay 4 -from [get_cells Block_A/R1] -to [get_cells Block_A/R2]
```

(in `Block_A.sdc`)

© 1999-2023
All Rights Reserved.

```
set_false_path -from [get_cells R1] -to [get_cells R2]
```


© 1999-2023
All Rights Reserved.

CCD_EXC_HIER13

Message

```
A block's path with max/min delay settings is part of a top-level false path
```

Default Severity

Warning

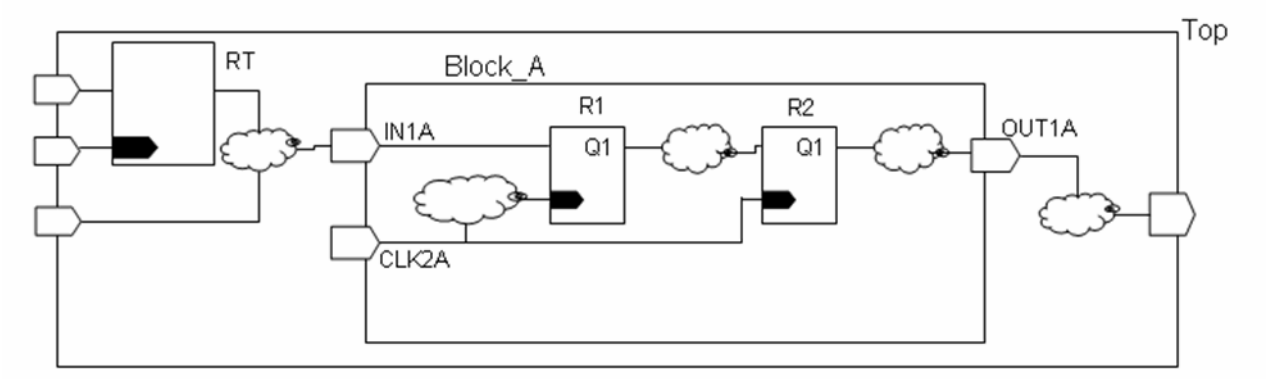
Description

Indicates that a block path, which was set a maximum or minimum delay at the block level, has been affected by a false-path specification.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example



The following commands cause a rule check violation:

(in `Top.sdc`)

```
set_false_path -from [get_cells Block_A/R1] -to [get_cells Block_A/R2]
```

(in `Block_A.sdc`)

© 1999-2023
All Rights Reserved.

```
set_max_delay 4 -from [get_cells R1] -to [get_cells R2]
```


© 1999-2023
All Rights Reserved.

CCD_EXC_HIER14

Message

```
A block's multicycle path is part of a top-level path with max/min delay settings
```

Default Severity

Warning

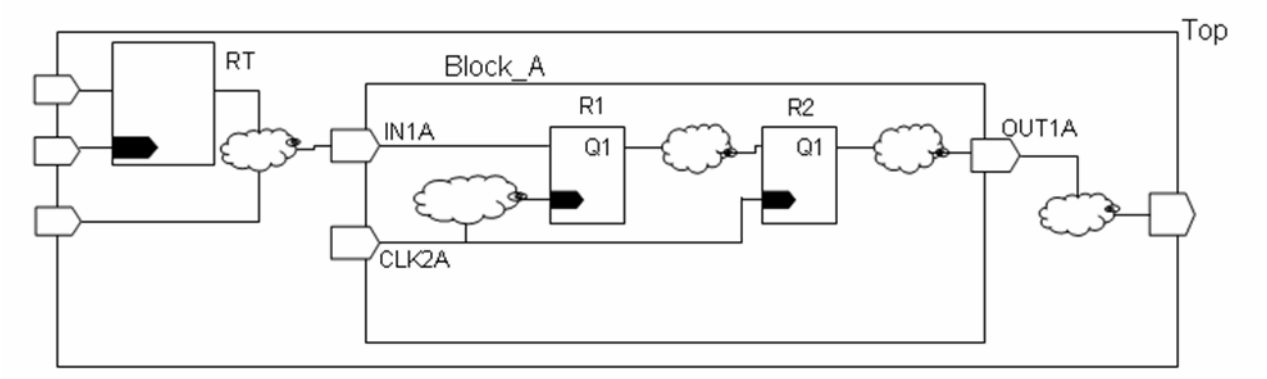
Description

Indicates that a block path, which was declared as multi-cycle at the block level, has been set by a top-level maximum or minimum delay.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example



The following commands cause a rule check violation:

(in `Top.sdc`)

```
set_max_delay 4 -from [get_cells BlkB/R1] -to [get_cells BlkB/R2]
```

(in `Block_A.sdc`)

© 1999-2023
All Rights Reserved.

```
set_multicycle_path 4 -from [get_cells R1] -to [get_cells R2]
```


© 1999-2023
All Rights Reserved.

CCD_EXC_HIER15

Message

```
A block's path with max/min delay settings is part of a top-level multicycle path
```

Default Severity

Warning

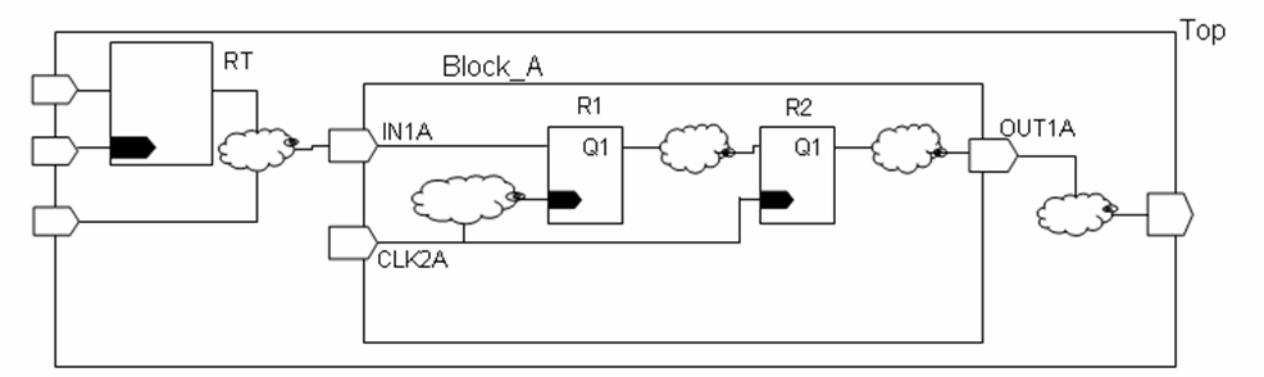
Description

Indicates that a block path, which was set a maximum or minimum delay at the block level, has been affected by a multi-cycle path specification.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example



The following commands cause a rule check violation:

(in `Top.sdc`)

```
set_multicycle_path 3 -from [get_cells BlkB/R1] -to [get_cells BlkB/R2]
```

(in `Block_A.sdc`)

© 1999-2023
All Rights Reserved.

```
set_max_delay 4 -from [get_cells R1] -to [get_cells R2]
```


© 1999-2023
All Rights Reserved.

CCD_EXC_HIER16

Message

```
A block's path is part of a top-level path with different max/min delay settings
```

Default Severity

Warning

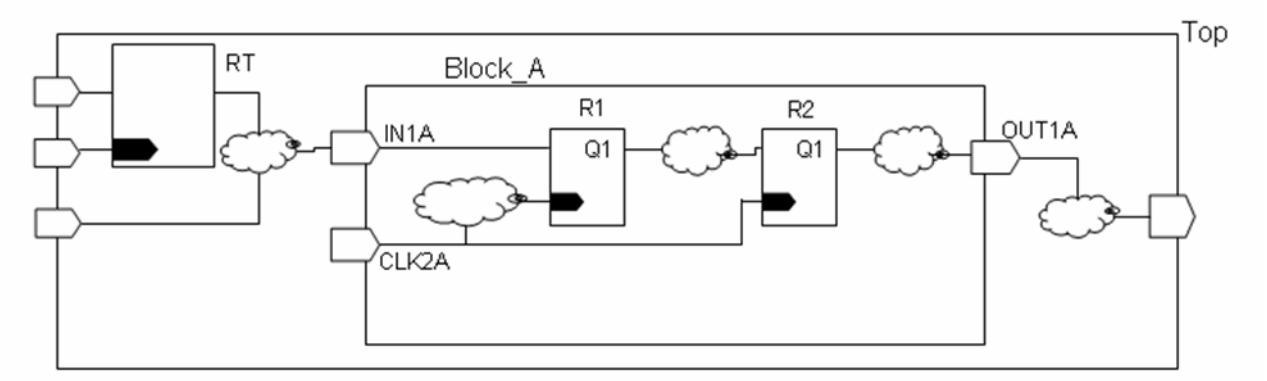
Description

Indicates that a block path is specified a minimum or maximum delay with a different value than that specified for a corresponding top-level path.

This is checked when you run the `run rule check` command.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example



The following commands cause a rule check violation:

(in `Top.sdc`)

```
set_max_delay 8 -from [get_cells BlkB/R1] -to [get_cells BlkB/R2]
```

(in `Block_A.sdc`)

© 1999-2023
All Rights Reserved.

```
set_max_delay 4 -from [get_cells R1] -to [get_cells R2]
```


© 1999-2023
All Rights Reserved.

CCD_EXC_HIER17

Message

```
A block path without set_false_path is part of top-level paths and all such top-
     level paths are defined as false paths
```

Default Severity

Warning

Description

Indicates that a block path was not declared as false at the block level and all top paths containing this block path have been affected by a top-level false-path specification.

This is checked when you run the `run rule check` command.

This rule is checked at the top level. You can specify the current SDC design with the `SET SDC DESIGN` command. The root design becomes a top-level design when you specify a sub-design using the `SET SDC DESIGN` or `ADD SDC DESIGN` commands.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

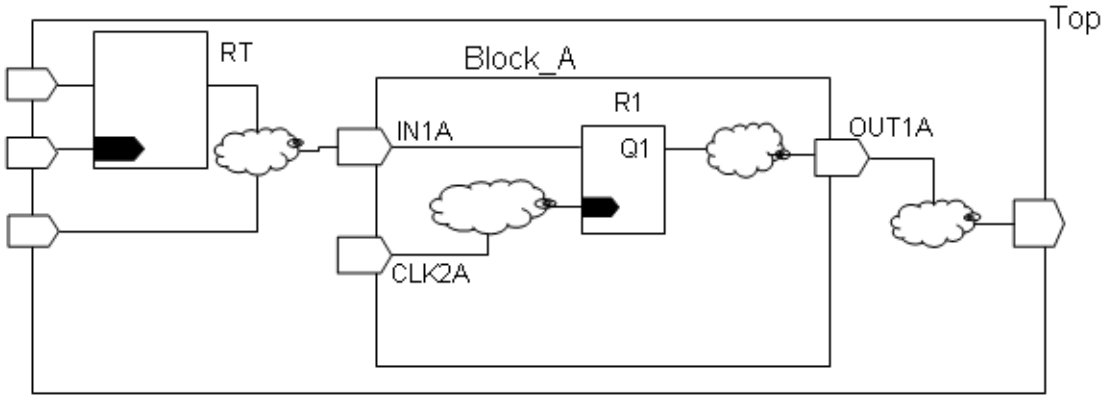
If `Block_A.sdc` has no `set_false_path` and `Top.sdc` has these lines:

```
set_false_path -through RT
set_false_path -through Block_A/OUT1A
```

The software issues a warning because the block-level did not consider the path from `R1` to `OUT1A` as false, even if all top paths containing this block path were affected by a false-path specification. However, the path from `IN1A` to `D1` was not reported by `CCD_EXC_HIER17` because the path from `P2` through `IN1A` to `D1` was not affected by any false-path

© 1999-2023 All Rights Reserved.

specification. However, such path will cause `CCD_EXC_HIER1` to be reported for the first top-level exception.



© 1999-2023
All Rights Reserved.

CCD_EXC_INT*

This section describes rule checks that relate to SDC integration for timing exceptions and similar constraints.

© 1999-2023 All Rights Reserved.

CCD_EXC_INT1

Message

`SDC integration: timing exceptions`

Default Severity

Warning

Description

Configures the integration of `set_false_path`, `set_multicycle_path`, `set_min_delay`, and `set_max_delay` commands, and reports the messages generated during this integration process.

This is checked when you run the `INTEGRATE` command.

© 1999-2023 All Rights Reserved.

CCD_MISC*

This section describes miscellaneous rule checks.

© 1999-2023 All Rights Reserved.

- CCD_MISC_INT1 on page 449

© 1999-2023
All Rights Reserved.

CCD_MISC_HFN1

Message

```
Clock is not set as dont_touch_network (pre-layout)
```

Default Severity

Warning

Description

Indicates that a real or generated clock does not have a `set_dont_touch_network`, and it is also missing from one of its source ports/pins.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

To fix this problem, use the `set_dont_touch_network` command. For example:

```
set_dont_touch_network CLK3
```


© 1999-2023 All Rights Reserved.

CCD_MISC_HFN1b

Message

`Set/reset is not set as dont_touch_network`

Default Severity

Warning

Description

Indicates that a port or pin that drives sequential element set or reset pins does not have a `set_dont_touch_network` command.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

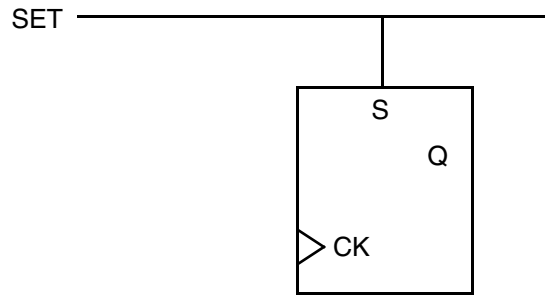
When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command.

Example

This rule will be reported if the SDC file does not contain a command like:

```
set_dont_touch_network [get_ports SET]
```



© 1999-2023
All Rights Reserved.

CCD_MISC_HFN2

Message

```
Object with set_dont_touch_network is not a clock, set or reset
```

Default Severity

Warning

Description

Indicates that a port or pin specified in `set_dont_touch_network` does not drive a sequential element clock, set or reset pins.

Note: When checking if the specified port or pin drives a clock, set or reset pin, propagated constants are taking into account for paths leading to clock pins, and they are ignored for paths leading to set or reset pins.

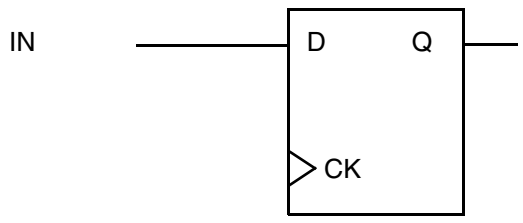
When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

This rule will be reported if the SDC file contains a command like:

```
set_dont_touch_network [get_ports IN]
```



© 1999-2023
All Rights Reserved.

CCD_MISC_HFN6

Message

`Inconsistent ideal transition values`

Default Severity

Warning

Description

Indicates that an ideal transition value specified for maximum delay analysis is smaller than the corresponding value specified for minimum delay analysis.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

In the following, the Conformal Constraint Designer will report the two first commands as inconsistent for the `-fall` case:

```
set_ideal_transition 1.00 -max [get_ports {pi[3]}]
set_ideal_transition 2.00 -min -fall [get_ports {pi[3]}]
set_ideal_transition 0.50 -min -rise [get_ports {pi[3]}]
```


© 1999-2023 All Rights Reserved.

CCD_MISC_HFN10

Message

`Invalid constraint in current context (post-layout)`

Default Severity

Warning

Description

Indicates that you have an invalid constraint in the current context. Specifically, the Conformal Constraint Designer flags `set_dont_touch`, `set_dont_touch_network`, `set_ideal_net`, `set_ideal_network`, `set_ideal_latency`, and `set_ideal_transition`.

This is checked when you read in your SDC files using the `READ SDC` command.

© 1999-2023 All Rights Reserved.

CCD_MISC_HFN13

Message

`set_dont_touch should not be used on library cells`

Default Severity

Warning

Description

Indicates that the `set_dont_touch` command was specified for a libcell object.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

This rule will be reported in a case like the following:

```
set_dont_touch [get_lib_cells {slow/OR2LLX05}]
```


© 1999-2023 All Rights Reserved.

CCD_MISC_NAM1

Message

`Clock has same name as port or pin`

Default Severity

Warning

Description

Indicates that you have a clock with the same name as a port or a pin.

This is checked when you read in your SDC files using the `READ SDC` command. This can happen when the `create_clock` or `create_generated_clock` command is missing the -name option. It is not recommended to define clocks whose names coincide with ports or pins, because implicit references to such objects (that is, commands that refer to them without an appropriate `get_clocks`/`get_ports`/`get_pins` command) would be ambiguous.

Example

The first command produces a warning because the `-name` argument is missing, which causes the new clock to have the same name as the port on which it is defined:

```
create_clock -period 10 -waveform {0 5} clk1
# Where clk1 is a port
```


© 1999-2023 All Rights Reserved.

CCD_MISC_NAM2

Message

`Virtual clock name does not match the defined standard`

Default Severity

Warning

Description

Indicates that you have a virtual clock name that does not match any of the wildcard patterns defined by the `SDC_VCLK_NAMES` parameter. Specifically, the Conformal Constraint Designer checks the `-name` value of the `create_clock` commands.

Note: A clock is *virtual* when you use `create_clock` without a defined source list.

For more information on CCD parameters, see `SET CCD PARAMETER` in the *Conformal Constraint Designer Reference Manual*.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

If you set the `SDC_VCLK_NAMES` parameter to "`*_v`", then the following command causes a warning:

```
create_clock -name vir_clk1 -period 10.0
```


© 1999-2023 All Rights Reserved.

CCD_MISC_DFT5

Message

`Scan mode is not enabled on scan enable pins in test mode constraint files`

Default Severity

Warning

Description

Indicates that you have an instance whose scan mode is not enabled in the test mode constraint files. For example, after propagating `set_case_analysis`, the scan pin of a sequential cell is not tied to the active value (`1` for `test_scan_enable`, `0` for `test_scan_enable_inverted`). For cells where the liberty annotation of scan enable pins is not available, the pin name is defined by the `SDC_SCAN_ENABLE_PIN` parameter.

Note: The Conformal Constraint Designer determines whether a constraint file is for test mode by checking if the CCD parameter `SDC_MODE` matches any of the modes specified in the parameter `SDC_SCAN_SHIFT_MODE`.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

For example, the dofile contains these commands:

```
set ccd parameter SDC_MODE shift2
set ccd parameter SDC_SCAN_SHIFT_MODE shift1,shift2
```

Assume that the liberty cell `SCANDFF` has a pin called `TE` with `test_scan_enable`, or that the dofile contains the following command:

```
set ccd parameter SDC_SCAN_ENABLE_PIN TE
```


© 1999-2023 All Rights Reserved.

This rule will be reported if the TE pin of any instance of SCANDFF is not set to the active value. To fix this, assuming SCANEN is a port that drives the TE pins directly, you can add the following command to the SDC file:

```
set_case_analysis 1 SCANEN
```


© 1999-2023 All Rights Reserved.

CCD_MISC_DFT6

Message

`Scan mode is not disabled on scan enable pins in functional mode constraint files`

Default Severity

Warning

Description

Indicates that you have an instance whose scan mode is not disabled in the functional constraint files. This rule check looks for instances where, after propagating `set_case_analysis`, the scan pin of a sequential cell is not tied to the inactive value (`0` for `test_scan_enable`, `1` for `test_scan_enable_inverted`). For cells where the liberty annotation of scan enable pins is not available, the pin name is defined by the `SDC_SCAN_ENABLE_PIN` parameter.

Note: The Conformal Constraint Designer determines whether a constraint file is for functional mode by checking if the CCD parameter `SDC_MODE` does not match any of the modes specified in the parameter `SDC_SCAN_SHIFT_MODE`.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

For example, the dofile contains these commands:

```
set ccd parameter SDC_MODE normal
set ccd parameter SDC_SCAN_SHIFT_MODE shift1,shift2
```

Assume that the liberty cell `SCANDFF` has a pin called `TE` with `test_scan_enable`, or that the dofile contains the following command:

```
set ccd parameter SDC_SCAN_ENABLE_PIN TE
```


© 1999-2023 All Rights Reserved.

This rule will be reported if the TE pin of any instance of SCANDFF is not set to the active value. To fix this, assuming `SCANEN` is a port that drives the TE pins directly, you can add the following command to the SDC file:

```
set_case_analysis 0 SCANEN
```


© 1999-2023 All Rights Reserved.

CCD_MISC_DFT7

Message

`Scan mode is not enabled on scan enable ports in test mode constraint files`

Default Severity

Warning

Description

Indicates that a scan enable port is not set to the active value, to enable scan mode, in the test mode constraint files. Specifically, this rule check looks for scan enable ports that are not tied to 1, or to 0 when inverted.

The scan enable ports are defined by the following SDC command:

```
set_scan_signal test_scan_enable[_inverted] -port port_list
```

Note: The Conformal Constraint Designer determines whether a constraint file is for test mode by checking if the CCD parameter `SDC_MODE` matches any of the modes specified in the parameter `SDC_SCAN_SHIFT_MODE`.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

If the dofile contains:

```
set ccd parameter SDC_MODE shift2
set ccd parameter SDC_SCAN_SHIFT_MODE shift1,shift2
```

And the SDC file contains:

```
set_case_analysis 1 SCANEN
set_scan_signal test_scan_enable_inverted -port SCANEN
```


© 1999-2023 All Rights Reserved.

the rule will report that an inverted scan enable port is not active. To fix this, change the `set_case_analysis` value to `0`.

© 1999-2023 All Rights Reserved.

CCD_MISC_DFT8

Message

`Scan mode is not disabled on scan enable ports in functional mode constraint files`

Default Severity

Warning

Description

Indicates that a scan enable port is not set to the inactive value, to disable scan mode, in the functional mode constraint files.

The scan enable ports are defined by the following SDC command:

```
set_scan_signal test_scan_enable[_inverted] -port port_list
```

Note: The Conformal Constraint Designer determines whether a constraint file is for functional mode by checking if the CCD parameter `SDC_MODE` does not match any of the modes specified in the parameter `SDC_SCAN_SHIFT_MODE`.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

If the dofile contains:

```
set ccd parameter SDC_MODE normal
set ccd parameter SDC_SCAN_SHIFT_MODE shift1,shift2
```

And the SDC file contains:

```
set_scan_signal test_scan_enable_inverted -port SCANEN
```

the rule will report that an inverted scan enable port is not set to the inactive value.

To fix this, add to the SDC file the command:

© 1999-2023 All Rights Reserved.

```
set_case_analysis 1 SCANEN
```


© 1999-2023
All Rights Reserved.

CCD_MISC_POW1

Message

`Undefined maximum dynamic power constraint`

Default Severity

Warning

Description

Indicates that no `set_max_dynamic_power` constraint has been defined.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

This rule will report if no `set_max_dynamic_power` SDC command is specified, but it could also be reported in a case like the following:

```
set_max_dynamic_power -2.0
```

The command fails because the value specified was negative, and this is reported by `CCD_SDC_SYN1`.

In this example, because there is no `set_max_dynamic_power` command that runs successfully, there is no maximum dynamic power constraint. This is reported by `CCD_MISC_POW1`.

© 1999-2023 All Rights Reserved.

CCD_MISC_POW2

Message

`Undefined maximum leakage power constraint`

Default Severity

Warning

Description

Indicates that no `set_max_leakage_power` constraint has been defined.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

This rule will report if no `set_max_leakage_power` SDC command is specified, but it could also be reported in a case like the following:

```
set_max_leakage_power -2.0
```

The command fails because the value specified was negative, and this is reported by `CCD_SDC_SYN1`.

In this example, because there is no `set_max_leakage_power` command that runs successfully, there is no maximum leakage power constraint. This is reported by `CCD_MISC_POW2`.

© 1999-2023 All Rights Reserved.

CCD_MISC_MSC2

Message

`Use of set_logic_* command (gate-level)`

Default Severity

Warning

Description

Indicates that you used `set_logic_*` commands. Specifically, the Conformal Constraint Designer flags `set_logic_dc`, `set_logic_one`, and `set_logic_zero` because `set_case_analysis` is the recommended command.

This is checked when you read in your SDC files using the `READ SDC` command.

© 1999-2023 All Rights Reserved.

CCD_MISC_MSC3

Message

`Use of set_logic_* command (pre-layout)`

Default Severity

Warning

Description

Indicates that you used `set_logic_*` commands. Specifically, the Conformal Constraint Designer flags `set_logic_dc`, `set_logic_one`, and `set_logic_zero` because `set_case_analysis` is the recommended command.

This is checked when you read in your SDC files using the `READ SDC` command.

Note: Avoid using logical pins in a constraints file at early stages of the flow, because hierarchical boundaries can be dissolved by implementation tools during flattening.

Example

The following command causes a rule check violation because it uses a ogical pin name:

```
set_case_analysis 0 [get_pins sub0/dataIn]
```


© 1999-2023 All Rights Reserved.

CCD_MISC_MSC5

Message

```
A cell specified in set_clock_gating_check is not on a clock tree
```

Default Severity

Warning

Description

Indicates that the `set_clock_gating_check` command has been applied to a cell, and that none of that cell's inputs is reached by any clock.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

If the instance `u_clock_gater` of cell `clock_gater` is not reached by the clock tree of any clock, then this rule will be reported if a command like this is found:

```
set_clock_gating_check -setup 0.4 [get_cells u_clock_gater}]
```


© 1999-2023 All Rights Reserved.

CCD_MISC_MSC6

Message

```
Total fanout load on a net that drives an output port exceeds the driver max_fanout
     attribute
```

Default Severity

Warning

Description

Indicates that an output (or inout) port with `set_fanout_load` is driven by a port or pin whose `max_fanout` t is smaller than the sum of the `fanout_load` attributes for all the loads connected to it.

The `max_fanout` attribute of the driver is the smallest (more restrictive) of any values specified in any of the following ways:

1. The SDC command `set_max_fanout` (on an input or inout port and/or on the current design)
2. If the driver is a cell pin, the `max_fanout` specified in the Liberty file for that library cell pin. If not specified, the `default_max_fanout` attribute of that cell's library is used.
3. If the driver is an input port and `set_driving_cell` is specified for that port, then (2) applies to the specified driving cell pin.
4. If none of the above is available, a default value of 0.0 is assumed.

The `fanout_load` of a load is determined as follows:

1. If it is an output or inout port, a `set_fanout_load` command from the SDC file.
2. If it is an input of a cell, the `fanout_load` attribute specified for that cell in the Liberty file for that library cell pin. If not specified, the `default_fanout_load` attribute of that cell's library is used.
3. If none of the above is available, a default value of 0.0 is assumed.

This is checked when you read in your SDC files using the `READ SDC` command.

© 1999-2023 All Rights Reserved.

Note: This rule can be diagnosed using the `DIAGNOSE RULE CHECK` command. The complete list of fanouts and their `fanout_load` attributes are displayed. A flattened schematic window shows the driver cell. To view all the loads, expand the displayed cell's fanout.

Example

A cell pin `I1/Z` drives both the ports OUT and the cell pin `I2/A`.

`I1` is an instance of cell `C1`, whose Liberty description defines `max_fanout` of pin `Z` as 10.0

`I2` is an instance of cell `C2`, that has no `max_fanout` defined in Liberty, but the library contains a `default_fanout_load` of 1.0.

If the SDC file contains the command:

```
set_fanout_load 10.0 [get_ports OUT]
```

then this rule will be reported because the total fanout load is 11.0.

© 1999-2023 All Rights Reserved.

CCD_MISC_MSC7

Message

`set_port_fanout_number present in post-layout`

Default Severity

Warning

Description

Indicates that the `set_port_fanout_number` command has been used in the G-post design stage.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

The dofile contains:

```
read sdc cst.sdc
```

And `cst.sdc` contains:

```
set_port_fanout_number 3 [get_ports din]
```


© 1999-2023 All Rights Reserved.

CCD_MISC_MSC10

Message

`Incomplete clock gating check`

Default Severity

Warning

Description

Indicates that there is an object with clock gating check specified, but one or more of the combinations of the options `-setup/-hold` and `-rise/-fall` are missing.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

In the following example, the Conformal Constraint Designer issues a warning because the `-fall` clock gating check was not specified:

```
set_clock_gating_check -setup 1.0 -hold 0.5 -rise [get_cells mygate]
```


© 1999-2023 All Rights Reserved.

CCD_MISC_MSC11

Message

```
set_max_time_borrow is not set on a latch or on a data/enable pin of a latch, or
     its value is bigger than half the clock period
```

Default Severity

Warning

Description

The `set_max_time_borrow` command can be applied to clocks, cells or pins. For other object types, rule `SDC_LINT_REF4` is reported. This rule is reported when the command is applied to clock objects, cells that are not level-sensitive, or pins that are not data or enable pins of level-sensitive sequential cells. It is also reported if the affected object is a level-sensitive cell or its data or enable pin, but the value specified is bigger than half the clock period of any clock propagating to the cell's enable pin.

When you switch from Setup to Verify mode, this rule will be checked automatically, unless its current severity is lower than the severity specified in the CCD parameter `SDC_AUTO_CHECK_SEVERITY`.

When you run the `run rule check` command (in Verify mode), this rule will be checked if it is still unchecked, or if you use the `-force` option.

Example

If the enable pin of latch `L1` is reached by clock `CLK` whose period is `10`, this rule will be reported for the following commands:

```
set_max_time_borrow 7 [get_cells L1]
set_max_time_borrow 7 [get_pins L1/D]
set_max_time_borrow 3 [get_clocks CLK]
set_max_time_borrow 3 [get_cells MUX1]
set_max_time_borrow 3 [get_pins L1/Q]
```


© 1999-2023 All Rights Reserved.

CCD_MISC_MSC12

Message

```
set_max_time_borrow set on a cell that is not a latch, or on a pin other than a
     data/enable pin of a latch
```

Default Severity

Warning

Description

The `set_max_time_borrow` command can be applied to clocks, cells or pins. For other object types, rule `SDC_LINT_REF4` is reported. This rule is reported when the command is applied to cells that are not level-sensitive, or pins that are not data or enable pins of level-sensitive sequential cells.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

If cell `L1` is a latch, this rule will be reported for the following command:

```
set_max_time_borrow 5 [get_cells MUX1]
set_max_time_borrow 5 [get_pins L1/Q]
```


© 1999-2023 All Rights Reserved.

CCD_MISC_MSC13

Message

`set_clock_gating_check applied to an object that is not a cell or a pin`

Default Severity

Warning

Description

The `set_clock_gating_check` command can be applied to clocks, ports, pins, or cells. This rule is reported when the object is a clock or a port. For other objects that are not pins or cells, rule `SDC_LINT_REF4` is reported. This rule is useful for certain methodologies and can be disabled safely.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

This rule will be reported for the following commands:

```
set_clock_gating_check [get_clocks CLK]
set_clock_gating_check [get_ports clk1]
```


© 1999-2023 All Rights Reserved.

CCD_MISC_MSC16

Message

`Incomplete set_data_check`

Default Severity

Warning

Description

Indicates that a `set_data_check` has been specified for `-setup` but not for `-hold`, or vice versa.

This is checked when you read in your SDC files using the `READ SDC` command.

Example

If the following constraints are specified:

```
(1) set_data_check 1 -setup -from A -to B -clock CLK1
(2) set_data_check 2 -hold -rise_from C -to D
(3) set_data_check 3 -setup -from C -rise_to D
```

this rule will report that:

- for (1), `set_data_check -hold` is missing
- for (2), `set_data_check -setup` is missing for `-rise_from/-fall_to`
- for (3), `set_data_check -hold` is missing for `-fall_from/-rise_to`

© 1999-2023 All Rights Reserved.

CCD_MISC_INT1

Message

`SDC integration: Miscellaneous commands`

Default Severity

Warning

Description

Configures the integration of arbitrary SDC commands and reports the messages generated during this integration process. This rule does not perform any checks that are command-specific. It acts on all the SDC commands that were parsed successfully, based on their textual representation only.

This is checked when you run the `INTEGRATE` command.

© 1999-2023 All Rights Reserved.

CCD_MMC*

This section describes rule checks that apply to different SDC modes.

© 1999-2023 All Rights Reserved.

CCD_MMC_CONFLCST

Message

`Conflicting constraints in two modes`

Default Severity

Warning

Description

Indicates that different constraints in two SDC modes are in conflict. For example, a `set_input_delay` with `-min` has a bigger value than a `set_input_delay` with `-max` in a different mode.

This is checked in Verify mode when you run the `run rule check` command if there are SDC modes defined (see the `ADD SDC MODE` command).

Note: In the current release, the path specification options `rise`/`fall_from`/`through`/`to` are not supported for this check.

© 1999-2023 All Rights Reserved.

CCD_MMC_INCONCST

Message

`Inconsistent constraints in two modes`

Default Severity

Warning

Description

Indicates that related constraints in two SDC modes have different values.

This is checked in Verify mode when you run the `run rule check` command if there are SDC modes defined (see the `ADD SDC MODE` command).

Note: In the current release, the path specification options `rise`/`fall_from`/`through`/`to` are not supported for this check.

© 1999-2023 All Rights Reserved.

CCD_MMC_UNCONSTR

Message

`Object not constrained in any mode`

Default Severity

Warning.

Description

Reports design objects that are not constrained in any SDC mode in which they are start points or endpoints of timing paths that are not affected by a `set_false_path`.

This is checked in Verify mode when you run the `run rule check` command if there are SDC modes defined (see the `ADD SDC MODE` command).

Note: In the current release, the path specification options `rise`/`fall_from`/`through`/`to` are not supported for this check.

© 1999-2023 All Rights Reserved.

© 1999-2023
All Rights Reserved.

7

Clock Tree Reporting Rules

This section describes the rule source for the clock tree rule set (`sdc_report_clock_tree`):

© 1999-2023 All Rights Reserved.

Clock Tree Reporting Quick Reference

Table 7-1 Quick Tasks for Clock Tree Reporting Rule Checks

Task	Command Example
Add rule set	`add rule set -file ccd_report_clock_tree_ruleset.tcl`
Report all rule groups	`report rule group sdc_report_clock_tree/*`
Report rule source	`report rule source sdc_report_clock_tree_*` `report rule source sdc_report_clock_tree_for_clock_pins`
Report rule instance	`report rule instance sdc_report_clock_tree/blocked_clocks/blocked` `report rule instance sdc_report_clock_tree/clock_pins/no_clocks`

Table 7-2 Clock Tree Reporting Rule Set (sdc_report_clock_tree) Quick Reference

Rule Group	Rule Instance	Rule Source
blocked_clocks	blocked	sdc_report_clock_tree_blocked
clock_convergence_reconvergence		
	convergence	sdc_report_clock_tree_conv
	reconvergence	
clock_pins	multiple_clocks	sdc_report_clock_tree_for_clock_pins
	no_clocks	
	single_clock	
clock_tree_xor	xor_with_untied_input	sdc_report_clock_tree_xor
missing_clocks	missing	sdc_report_clock_tree_missing
non_clock_pins	floating_pins	sdc_report_clock_tree_for_non_clock_pins
	non_clk_seq_pins	
	POs	
sfp_and_clock_overlap	sfp_overlap	sdc_report_clock_tree_sfp_overlap

© 1999-2023 All Rights Reserved.

Note: The Rule Instance names provided are base names, the full path of the rule instance would contain its rule set and rule group. For example, here are some full rule instance names:

```
sdc_report_clock_tree/blocked_clocks/blocked
sdc_report_clock_tree/non_clock_pins/POs
sdc_report_clock_tree/clock_tree_xor/xor_with_untied_input
```


© 1999-2023 All Rights Reserved.

sdc_report_clock_tree_blocked

```
sdc_report_clock_tree_blocked
     clocks <list_of_clocks>
     file_path <relative_path | full_path>
```

Reports objects where clock propagation has stopped or is blocked due to constraints (hard constraints or SDC constraints like `set_case_analysis`, `set_disable_timing`, or `set_clock_sense`). The report will include the clock name and the reason why the propagation stopped.

Rule Attributes

Criteria for the rule is specified through attributes. The following describes the supported attributes for this check. Some of these attributes are proved by atomic checks (described in "Atomic Checks" on page 533).

clocks

Description	Specifies the clocks to check. By default, all clocks will be checked.
Possible Value	*list_of_clocks*
Example	`set_attribute $rule_instance clocks \` `[find -sdcobj "<clk1> <clk2>"]`

file_path

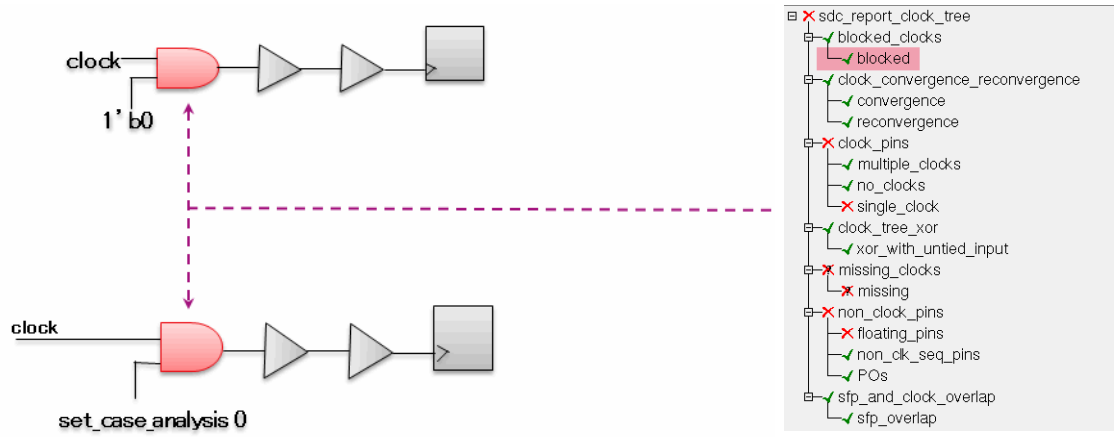
Description	Report either the full or relative file path for the location. By default, the relative path is reported.
Possible Value	`relative_path | full_path`
Example	`set_attribute $rule_instance file_path full_path`

© 1999-2023 All Rights Reserved.

Example

The following diagram illustrates the situations that can be reported by this check. The purple dotted arrows point to the objects that trigger the highlighted rule instance in the Rule Manager.

Figure 7-1 Objects/Locations Where Clock Propagation has Stopped



All Rights Reserved.

sdc_report_clock_tree_conv

```
sdc_report_clock_tree_conv
     clocks <list_of_clocks>
     check_type <single_clock | multiple_clock | no_clocks>
     file_path <relative_path | full_path>
```

Reports the points where multiple clocks converge, or where a given clock reconverges.

Rule Attributes

Criteria for the rule is specified through attributes. The following describes the supported attributes for this check. Some of these attributes are proved by atomic checks (described in "Atomic Checks" on page 533).

clocks

Description	Specifies the clocks to check. By default, all clocks will be checked.
Possible Value	*list_of_clocks*
Example	`set_attribute $rule_instance clocks [find -sdcobj \ "<clk1> <clk2>"]`

check_type

Description	Specifies the type of check. Convergence is checked by default.	
Possible Value	<convergence | reconvergence>	
	convergence	Check for points where multiple clocks converge
	reconvergence	Check for points where the given clock reconverges
Example	`set_attribute $rule_instance check_type reconvergence`	

© 1999-2023 All Rights Reserved.

file_path

Description	Report either the full or relative file path for the location. By default, the relative path is reported.
Possible Value	`relative_path | full_path`
Example	`set_attribute $rule_instance file_path full_path`

Example

The following diagrams illustrate the situations that can be reported by this check. The purple dotted arrow points to the object that triggers the highlighted rule instance in the Rule Manager.

Figure 7-2 Check Type: Convergence

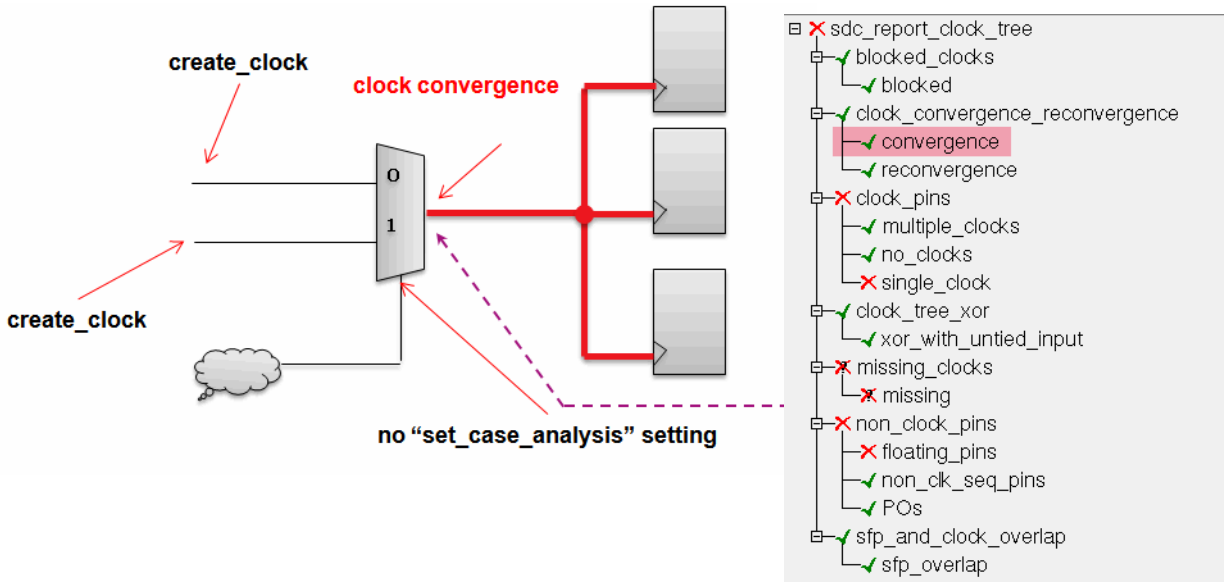
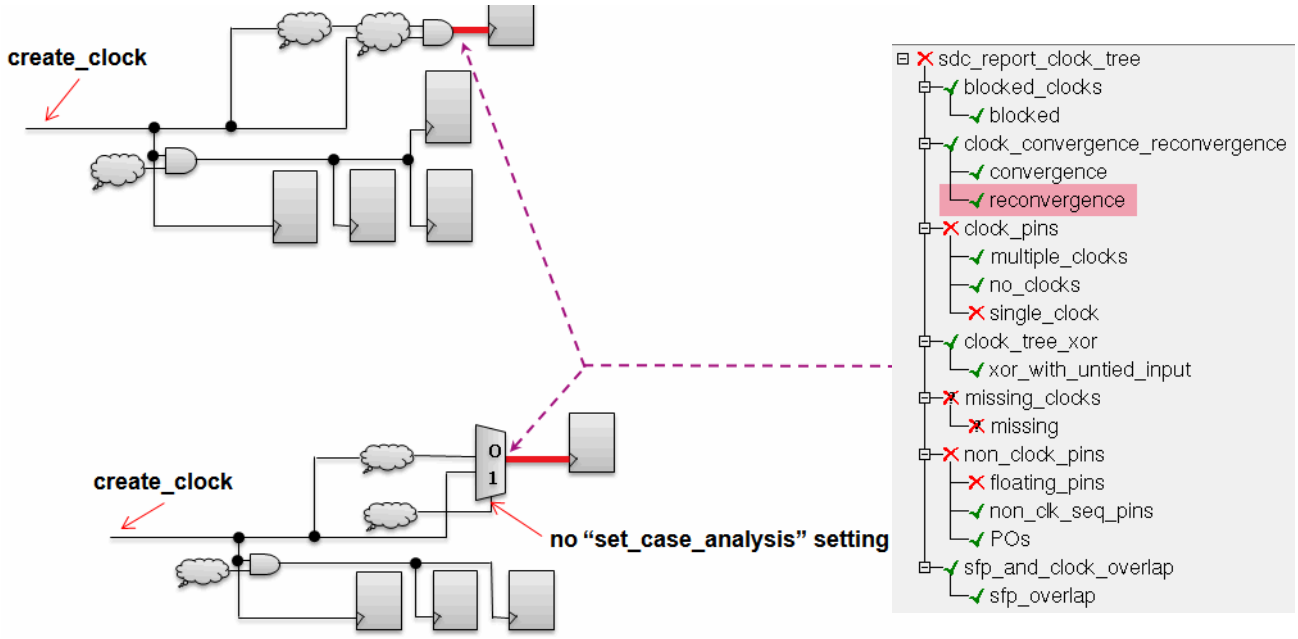


Figure 7-3 Check Type: Reconvergence



© 1999-2023
All Rights Reserved.

sdc_report_clock_tree_for_clock_pins

```
sdc_report_clock_tree_for_clock_pins
     clocks <list_of_clocks>
     check_type <single_clock | multiple_clock | no_clocks>
     file_path <relative_path | full_path>
```

Reports the clock pins that are reached by the specified clocks.

Rule Attributes

Criteria for the rule is specified through attributes. The following describes the supported attributes for this check. Some of these attributes are proved by atomic checks (described in "Atomic Checks" on page 533).

clocks

Description	Specifies the clocks to check. By default, all clocks will be checked.
Possible Value	*list_of_clocks*
Example	`set_attribute $rule_instance clocks \` `[find -sdcobj "<clk1> <clk2>"]`

check_type

Description	Specifies the type of pins to report. By default, pins that are reached by a single clock are reported.	
Possible Value	`<single_clock | multiple_clock | no_clocks>`	
	`single_clock`	Report clock pins that are reached by a single clock.
	`multiple_clock`	Report clock pins that are reached by multiple clocks.
	`no_clocks`	Report clock pins that are not reached by any clock.
Example	`set_attribute $rule_instance check_type no_clocks`	

© 1999-2023 All Rights Reserved.

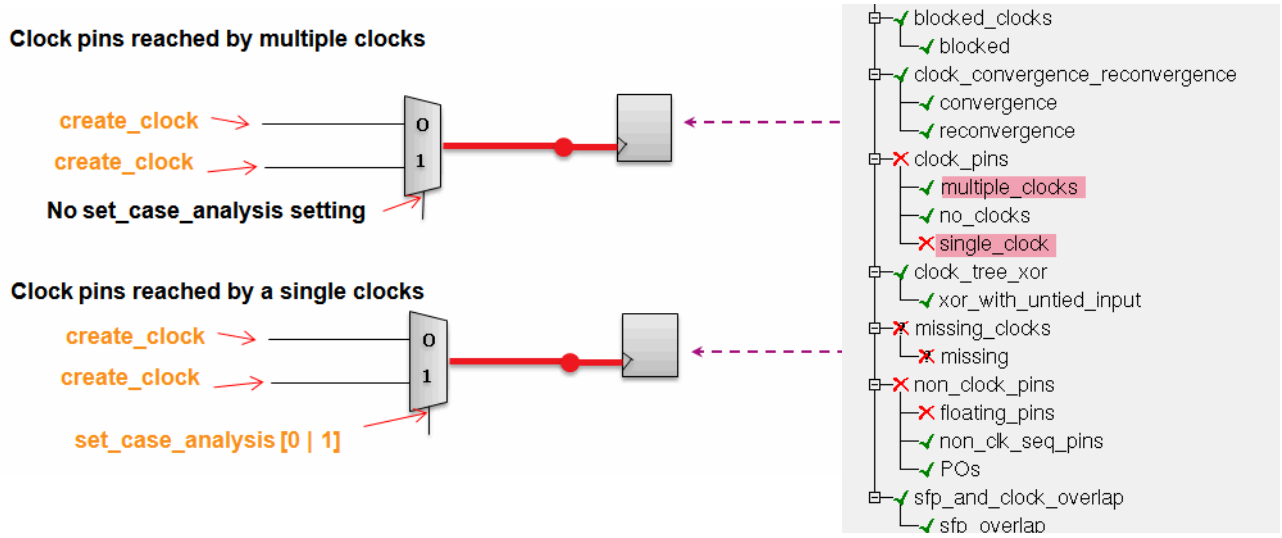
file_path

Description	Report either the full or relative file path for the location. By default, the relative path is reported.
Possible Value	`relative_path | full_path`
Example	`set_attribute $rule_instance file_path full_path`

Example

The following diagrams illustrate the situations that can be reported by this check. The purple dotted arrow points to the object that triggers the highlighted rule instance in the Rule Manager.

Figure 7-4 Check Types: Single Clock and Multiple Clock



All Rights Reserved.

sdc_report_clock_tree_xor

```
sdc_report_clock_tree_xor
     clocks <list_of_clocks>
     file_path <relative_path | full_path>
```

Report XOR/XNOR instances in the clock tree that have a non-clock input that is not tied to a constant.

Rule Attributes

Criteria for the rule is specified through attributes. The following describes the supported attributes for this check. Some of these attributes are proved by atomic checks (described in "Atomic Checks" on page 533).

clocks

Description	Specifies the clocks to check. By default, all clocks will be checked.
Possible Value	*list_of_clocks*
Example	`set_attribute $rule_instance clocks \` `[find -sdcobj "<clk1> <clk2>"]`

file_path

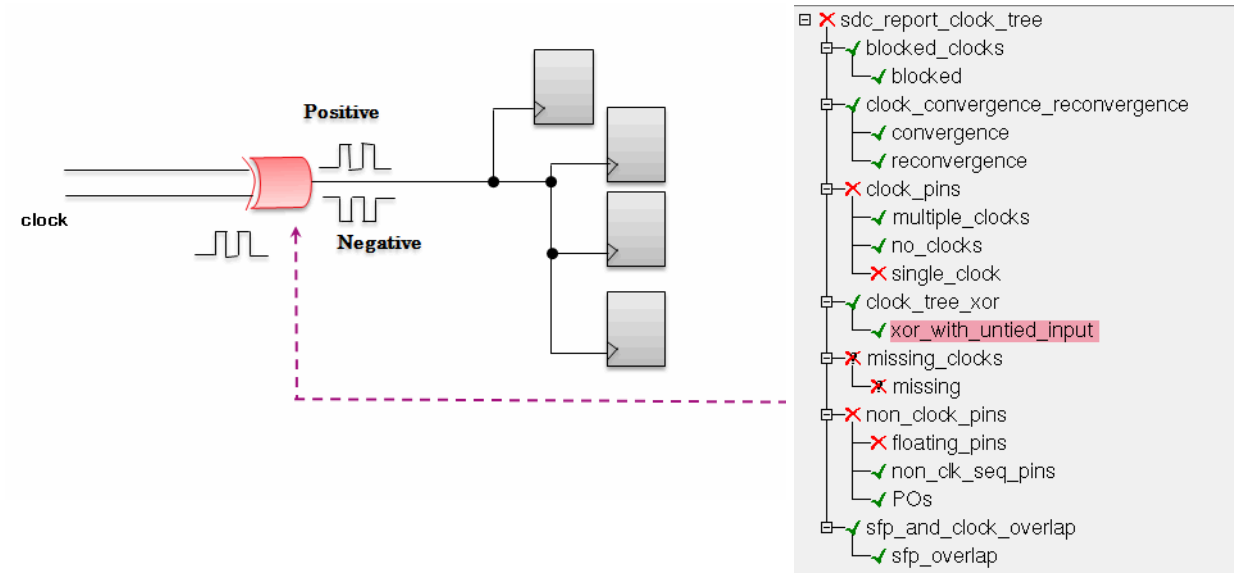
Description	Report either the full or relative file path for the location. By default, the relative path is reported.
Possible Value	`relative_path | full_path`
Example	`set_attribute $rule_instance file_path full_path`

© 1999-2023 All Rights Reserved.

Example

The following diagram illustrates a situation that can be reported by this check. The purple dotted arrow points to the object that triggers the highlighted rule instance in the Rule Manager.

Figure 7-5 XOR/XNOR Instance with a Non-Clock Input Not Tied to a Constant



© 1999-2023
All Rights Reserved.

sdc_report_clock_tree_missing

```
sdc_report_clock_tree_missing
     file_path <full_path | relative_path>
```

Reports clock pins to which no clocks propagate because of missing clock constraints.

Rule Attributes

Criteria for the rule is specified through attributes. The following describes the supported attributes for this check. Some of these attributes are proved by atomic checks (described in "Atomic Checks" on page 533).

file_path

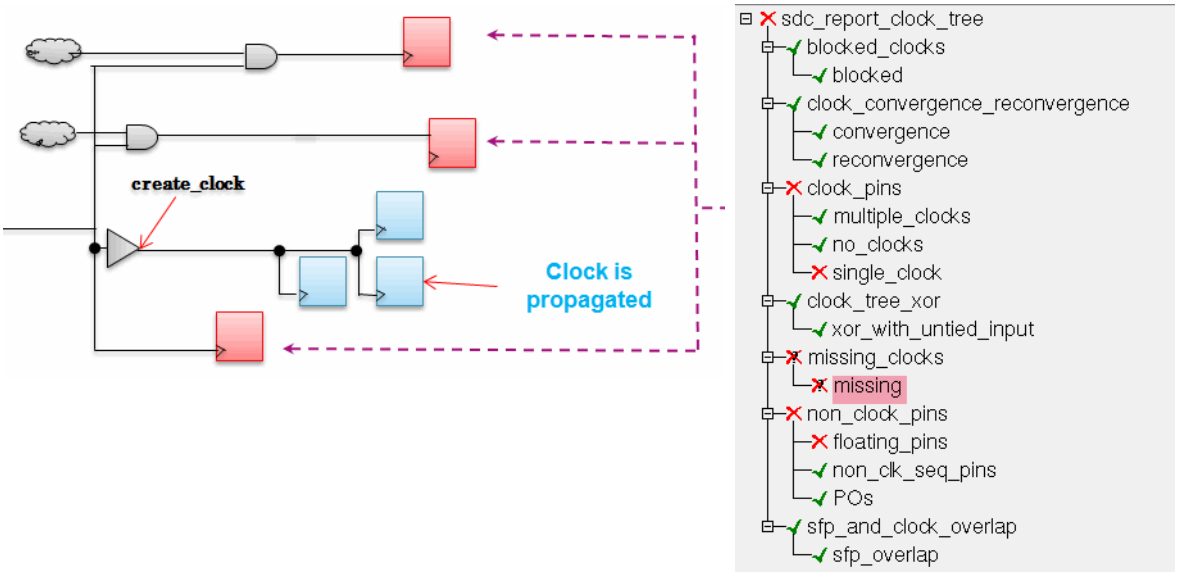
Description	Report either the full or relative file path for the location. By default, the relative path is reported.
Possible Value	`relative_path | full_path`
Example	`set_attribute $rule_instance file_path full_path`

© 1999-2023 All Rights Reserved.

Example

The following diagram illustrates a situation that can be reported by this check. The purple dotted arrow points to the object that triggers the highlighted rule instance in the Rule Manager.

Figure 7-6 Clock Pins to Which No Clocks Propagate due to Missing Clock Constraints



© 1999-2023
All Rights Reserved.

sdc_report_clock_tree_for_non_clock_pins

```
sdc_report_clock_tree_for_non_clock_pins
     clocks <list_of_clocks>
     pin_type <non_clk_seq_pin | po | floating>
     file_path <relative_path | full_path>
```

Reports non-clock pins reached by the given clocks.

Rule Attributes

Criteria for the rule is specified through attributes. The following describes the supported attributes for this check. Some of these attributes are proved by atomic checks (described in "Atomic Checks" on page 533).

clocks

Description	Specifies the clocks to check. By default, all clocks will be checked.
Possible Value	*list_of_clocks*
Example	`set_attribute $rule_instance clocks \` `[find -sdcobj "<clk1> <clk2>"]`

pin_type

Description	Specifies the type of pins to report. Default is `non_clk_seq_pin`.	
Possible Value	`<non_clk_seq_pin | po | floating>`	
	`non_clk_seq_pin`	Report non-clock pins of sequential instances that are reached by any clock.
	`po`	Report primary outputs that are reached by any clock.
	`floating`	Report floating pins that are reached by any clocks.
Example	`set_attribute $rule_instance pin_type floating`	

© 1999-2023 All Rights Reserved.

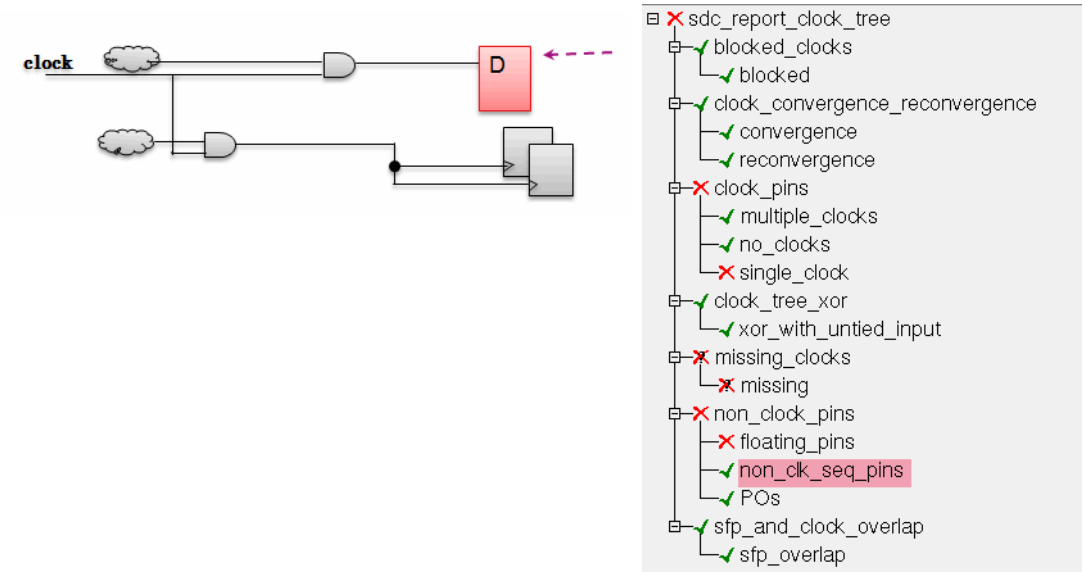
file_path

Description	Report either the full or relative file path for the location. By default, the relative path is reported.
Possible Value	`relative_path | full_path`
Example	`set_attribute $rule_instance file_path full_path`

Example

The following diagrams illustrate situations that can be reported by this check. The purple dotted arrow points to the object that triggers the highlighted rule instance in the Rule Manager.

Figure 7-7 Non-Clock Pins of Sequential Instances Reached by any Clock



© 1999-2023
All Rights Reserved.

Figure 7-8 Primary Outputs Reached by any Clock

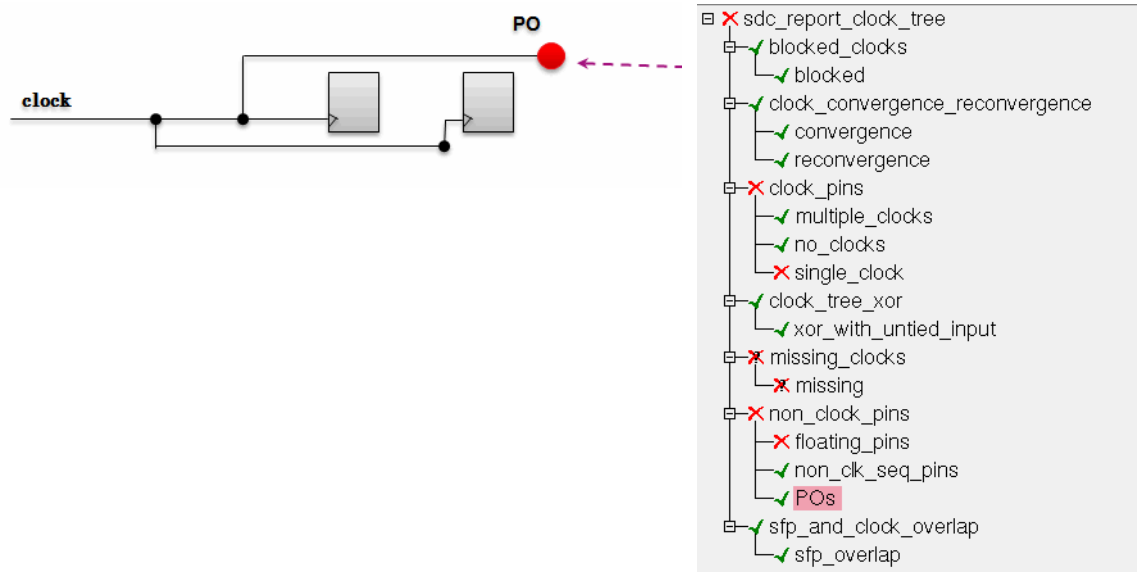
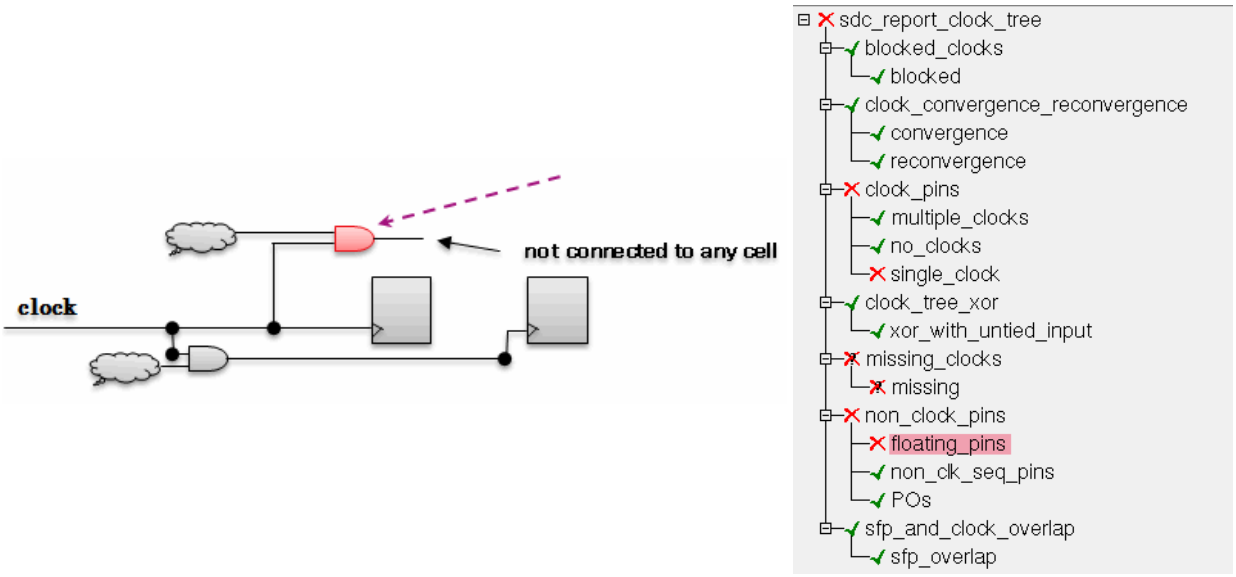


Figure 7-9 Floating Pins Reached by any Clock



All Rights Reserved.

sdc_report_clock_tree_sfp_overlap

```
sdc_report_clock_tree_sfp_overlap
     clocks <list_of_clocks>
     file_path <relative_path | full_path>
```

Report overlap between clock tree defined by `create_clock`/`create_generated_clock` and `set_false_path` without clocks specified.

Rule Attributes

Criteria for the rule is specified through attributes. The following describes the supported attributes for this check. Some of these attributes are proved by atomic checks (described in "Atomic Checks" on page 533).

clocks

Description	Specifies the clocks to check. By default, all clocks will be checked.
Possible Value	*list_of_clocks*
Example	`set_attribute $rule_instance clocks \` `[find -sdcobj "<clk1> <clk2>"]`

file_path

Description	Report either the full or relative file path for the location. By default, the relative path is reported.
Possible Value	`relative_path | full_path`
Example	`set_attribute $rule_instance file_path full_path`

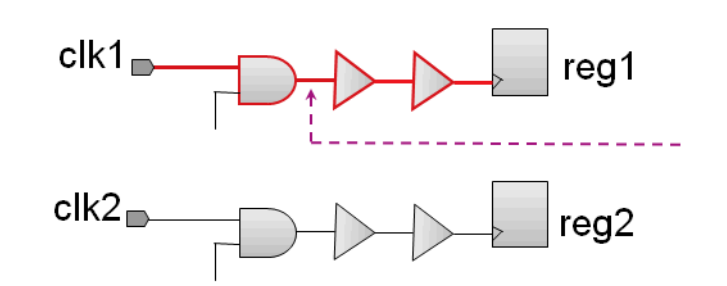
© 1999-2023 All Rights Reserved.

Example

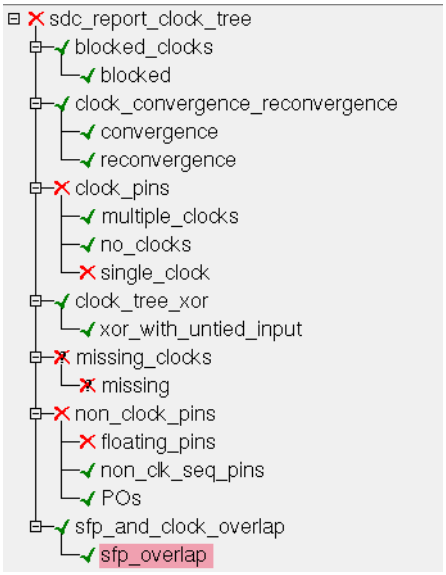
The following diagram illustrates a situation that can be reported by this check. The purple dotted arrow points to the object that triggers the highlighted rule instance in the Rule Manager.

In the following figure, it is reported as an overlap because the `set_false_path` path covers part of the clock tree for `CLK`.

Figure 7-10 Overlap Between Clock Definition and set_false_path




```
create_clock -name CLK [get_ports {clk1 clk2}]
set_false_path -from [get_ports clk1]
```



© 1999-2023
All Rights Reserved.

© 1999-2023 All Rights Reserved.

A

Atomic Checks

This section describes the atomic checks related to FIFO rule checks and CDC rule checks. This section also provides supplementary figures to help describe the various rule attributes and atomic checks.

© 1999-2023 All Rights Reserved.

CDC Atomic Checks

Clock Domain Crossing Rule Checks have rule attributes that are proved by CDC atomic checks. The following lists the CDC atomic checks, their applicable synchronization type, and a link to a supplementary figure that illustrates the portion of the synchronizer that is covered by the atomic check.

For more information on the crossing types, refer to the "Running Clock Domain Crossing Checks" chapter of the *Conformal Constraint Designer User Guide*.

Structural Checks	Synchronization Type				FIGURE
	DFF	MUX	CONV	SR	
cdc_path_logic_type_check	Y	Y			A-1, A-2
cdc_path_destination_check	Y	Y			A-3, A-4
cdc_data_holding_check		Y			A-5
cdc_data_other_domain_check		Y			A-6
cdc_data_holding_sync_check		Y			A-7
cdc_data_holding_unknown		Y			A-8
cdc_data_logic_type_check		Y			A-9
cdc_data_min_sync_chain_check		Y			A-10
cdc_data_max_sync_chain_check		Y			A-10
cdc_data_multi_chain_check		Y			A-11
cdc_data_mixed_domain_check		Y			A-12
cdc_data_first_dlatch_check		Y			A-10
cdc_ctrl_logic_type_check	Y				A-13
cdc_ctrl_multi_chain_check	Y				A-14
cdc_ctrl_min_sync_chain_check	Y				A-15
cdc_ctrl_max_sync_chain_check	Y				A-15
cdc_ctrl_mixed_domain_check	Y				A-16
cdc_ctrl_first_dlatch_check	Y				A-15
cdc_conv_in_vector_check			Y		A-17

© 1999-2023 All Rights Reserved.

Structural Checks	Synchronization Type				FIGURE
	DFF	MUX	CONV	SR	
cdc_conv_out_vector_check			Y		A-18
cdc_conv_same_domain_check			Y		A-19
cdc_conv_diff_domain_check			Y		A-20
cdc_conv_source_filtered			Y		NA
cdc_setreset_deassertion_check				Y	NA
cdc_setreset_min_dff_check				Y	NA
cdc_setreset_mixed_domain_check				Y	
cdc_setreset_pi_association_check				Y	NA
cdc_setreset_sync_chain_mix_sr_check				Y	NA
cdc_setreset_sync_chain_nosr_check				Y	NA
cdc_setreset_logic_type_check				Y	NA
cdc_setreset_source_driver_check				Y	NA
cdc_setreset_target_clock_sync_check				Y	NA
cdc_user_sync_module_check	Y	Y			NA
cdc_inactive_path_check	Y	Y	Y	Y	NA
cdc_fifo_crossing_check	Y	Y	Y	Y	NA
cdc_clock_group_check	Y	Y		Y	NA
cdc_path_not_processed	Y	Y	Y	Y	NA
Functional Checks	**Applicable Crossings**				**FIG.**
	DFF	**MUX**	**CONV**	**SR**	
cdc_source_stability	Y	Y			A-21
cdc_destination_stability	Y	Y			A-22
cdc_mux_enable_stability		Y			A-23
cdc_single_bit_change	Y		Y		A-24

© 1999-2023
All Rights Reserved.

CDC Atomic Checks by Rule Attribute

The following lists the CDC rule attributes and the atomic checks that prove them.

Rule Name	Rule Attribute	Atomic Checks
cdc_datactrl_sync_rule	sync_chain_logic	cdc_data_logic_type_check
		cdc_ctrl_logic_type_check
	sync_chain_fanout	cdc_data_multi_chain_check
		cdc_ctrl_multi_chain_check
	cdc_path_logic	cdc_path_logic_type_check
	cdc_path_fanout	cdc_path_destination_check
	dff_sync_scheme	cdc_ctrl_min_sync_chain_check
		cdc_ctrl_max_sync_chain_check
		cdc_ctrl_first_dlatch_check
	mux_sync_scheme	cdc_data_min_sync_chain_check
		cdc_data_max_sync_chain_check
		cdc_data_first_dlatch_check
cdc_conv_check_rule	check_conv_type	cdc_conv_same_domain_check
		cdc_conv_diff_domain_check
	check_vector_conv	cdc_conv_in_vector_check
		cdc_conv_out_vector_check
cdc_setreset_sync_rule	dff_sync_scheme	cdc_setreset_min_dff_check
	allowed_logic	cdc_setreset_logic_type_check

© 1999-2023 All Rights Reserved.

cdc_path_logic_type_check

Checks that the logic type in the CDC path matches the type specified by the `cdc_path_logic` attribute of the `cdc_datactrl_sync_rule` check.

Synchronization Scheme

DFF and MUX

Example

In the following example, the `cdc_path_logic_type_check` fails because there is a "logic" AND gate in the design (line 9) when the `cdc_path_logic` attribute is set to "wire".

```
VERIFY> report rule check cdc_def_rs/cdc* -verbose -status fail

|  cdc_path_logic                : dff wire mux wire user wire
...
...
|         structural : cdc_path_logic_type_check           : LOGIC          : Fail
```

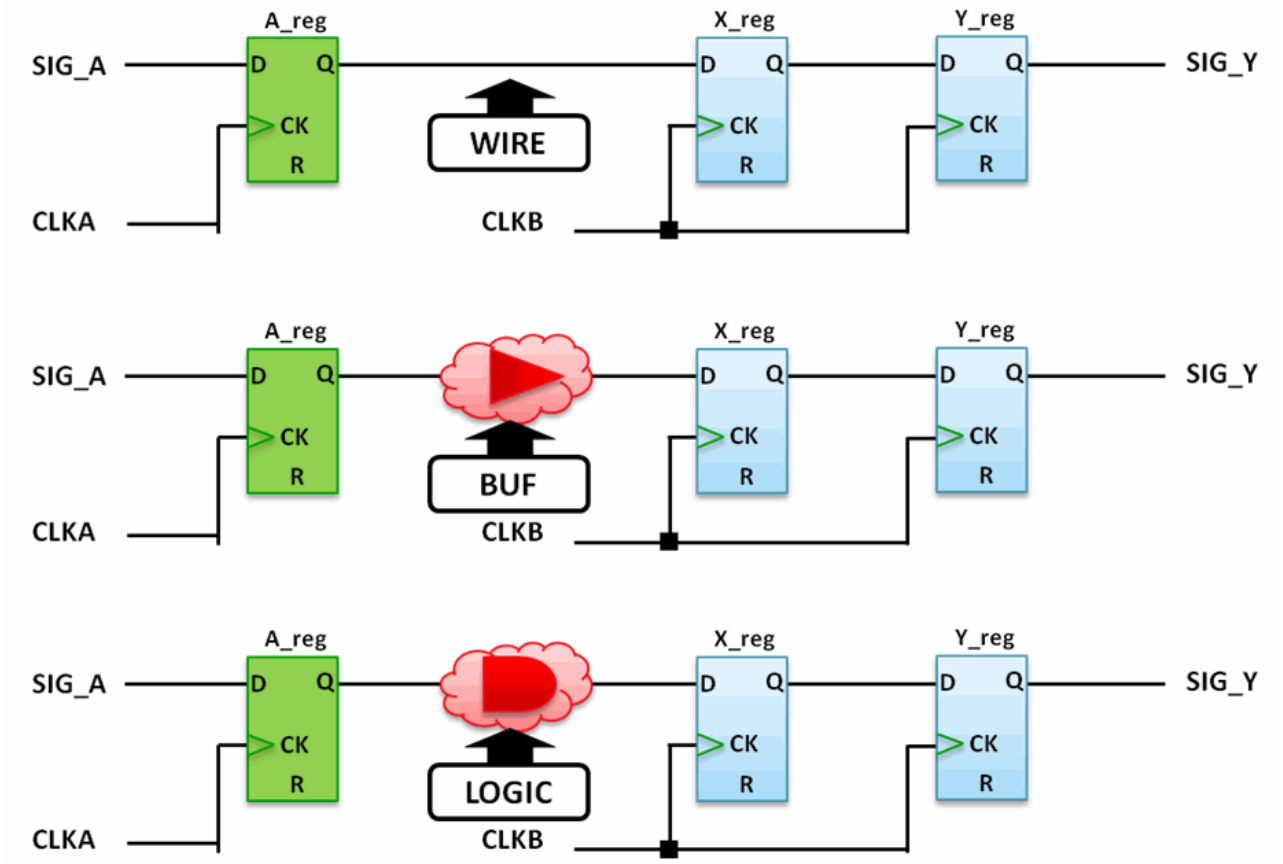
```
module test(data, clka, clkb, out);
input [1:0] data;
input  clka, clkb;
output out;
reg out, din1,dand;
reg [1:0] din;
always @(posedge clka)
  din <= data;
assign dand = din[1] & din[0];
always @(posedge clkb)
begin
  din1 <= dand;
  out <= din1;
end
endmodule
```

To resolve the mismatch flagged by the atomic check (granted the "logic" type is expected; otherwise, this is a true bug and the design that needs to be fixed):

```
set_attribute [ find -ruleinst cdc_def_rs/cdc*] cdc_path_logic "dff logic"
```

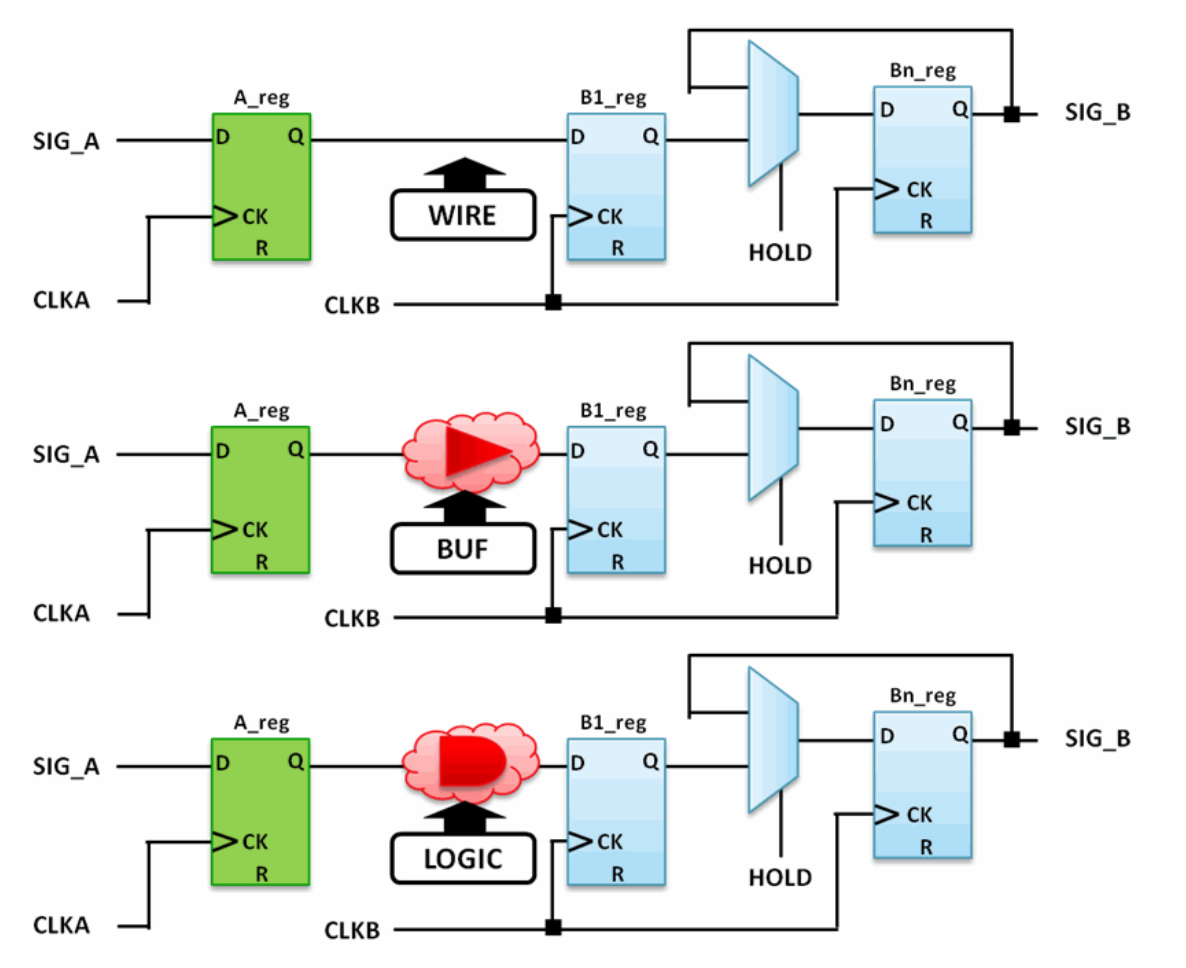

© 1999-2023 All Rights Reserved.

Figure A-1 cdc_path_logic_type_check (for DFF)



© 1999-2023
All Rights Reserved.

Figure A-2 cdc_path_logic_type_check (for MUX)



cdc_path_destination_check

Checks that the destination (single or multiple) in the CDC path matches the type specified by the `cdc_path_fanout` attribute of the `cdc_datactrl_sync_rule` check.

Synchronization Scheme

DFF and MUX

Example

In the following example, `cdc_path_destination_check` fails because there are multiple fanouts in the design (line 11) when the `cdc_path_fanout` attribute is set to "single".

```
VERIFY> report rule check cdc_def_rs/cdc* -verbose -status fail
|  cdc_path_fanout               : dff single mux single user single
...
...
|         structural : cdc_path_destination_check          : MULTIPLE      : Fail
```

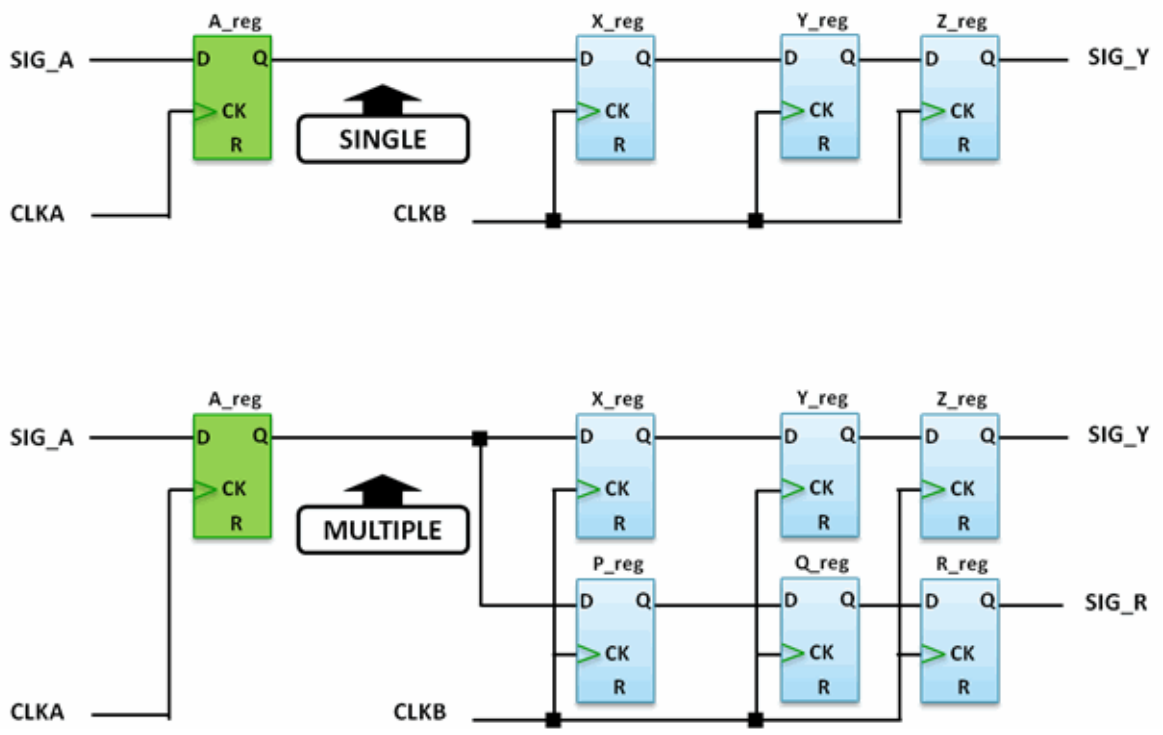
```
module test(data, clka, clkb, out);
input data;
input  clka, clkb;
output [1:0] out;
reg din;
reg [1:0] out, din1;
always @(posedge clka)
  din <= data;
always @(posedge clkb)
begin
  din1 <= {din,din};
  out <= din1;
end
endmodule
```

To resolve the mismatch flagged by the atomic check (granted multiple fanouts are expected in the design; otherwise, this is a true bug and the design that needs to be fixed):

```
set_attribute [ find -ruleinst cdc_def_rs/cdc*] cdc_path_fanout "dff multiple"
```

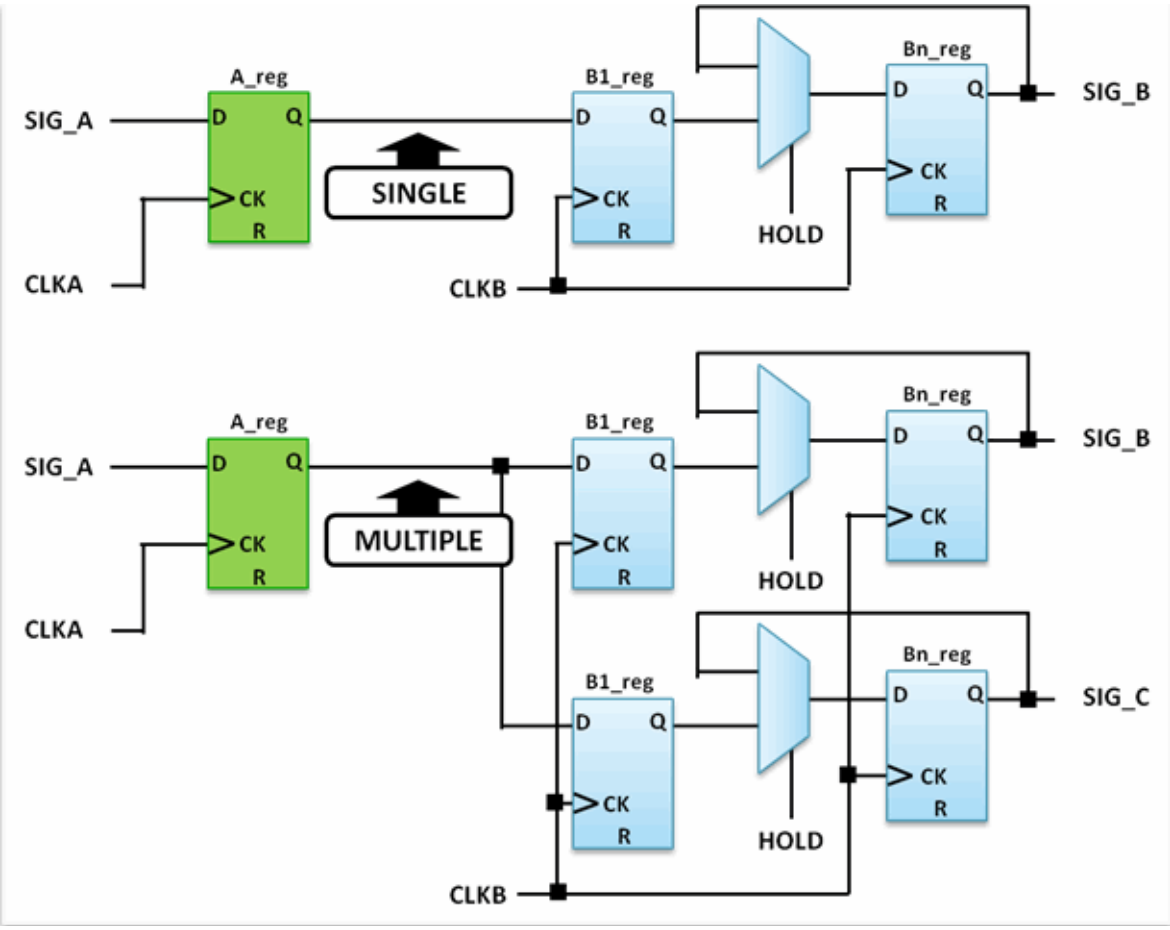

© 1999-2023 All Rights Reserved.

Figure A-3 cdc_path_destination_check (for DFF)



© 1999-2023
All Rights Reserved.

Figure A-4 cdc_path_destination_check (for MUX)



© 1999-2023
All Rights Reserved.

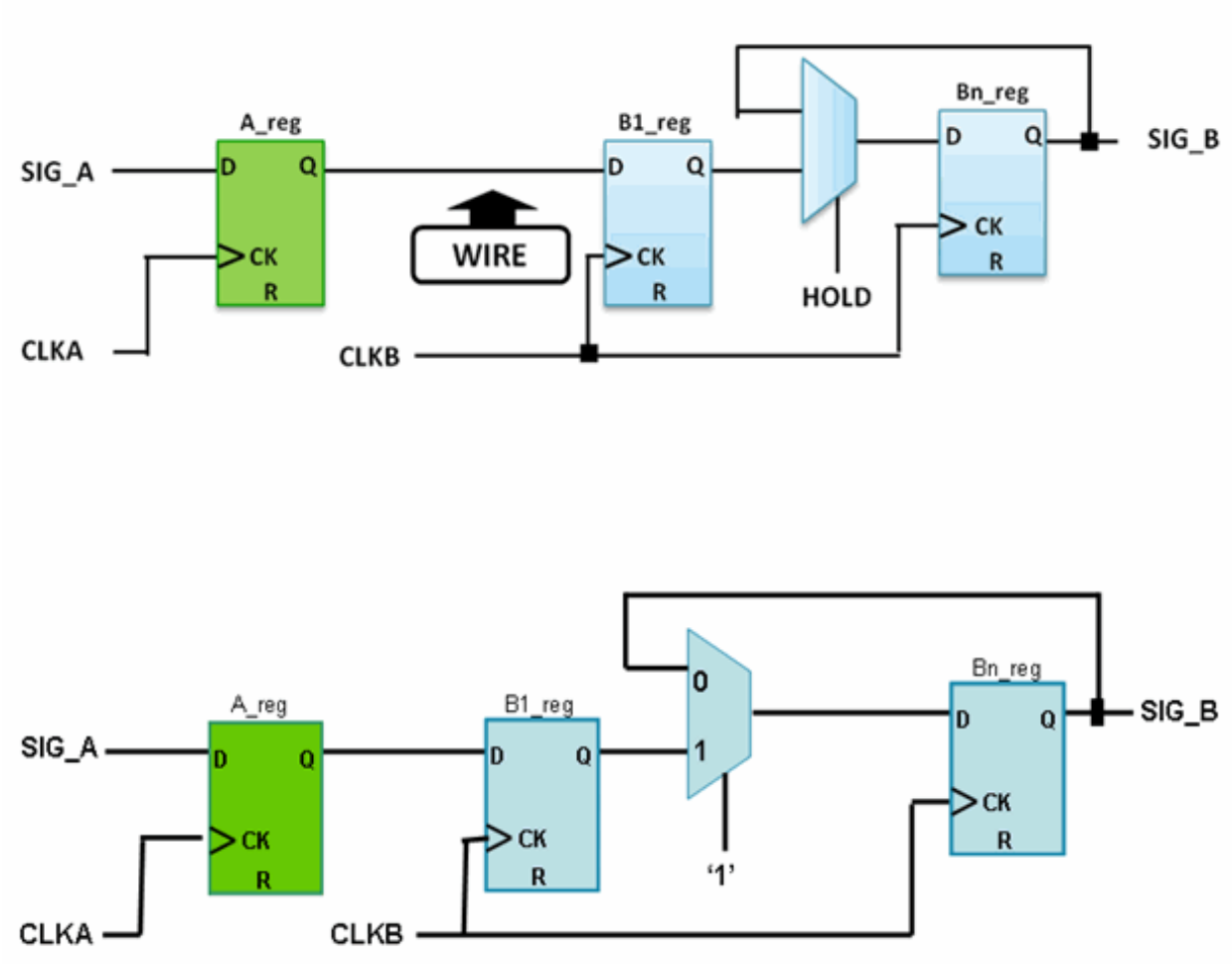
cdc_data_holding_check

Checks for the data holding condition.

Synchronization Scheme

MUX

Figure A-5 cdc_data_holding_check (for MUX)



© 1999-2023
All Rights Reserved.

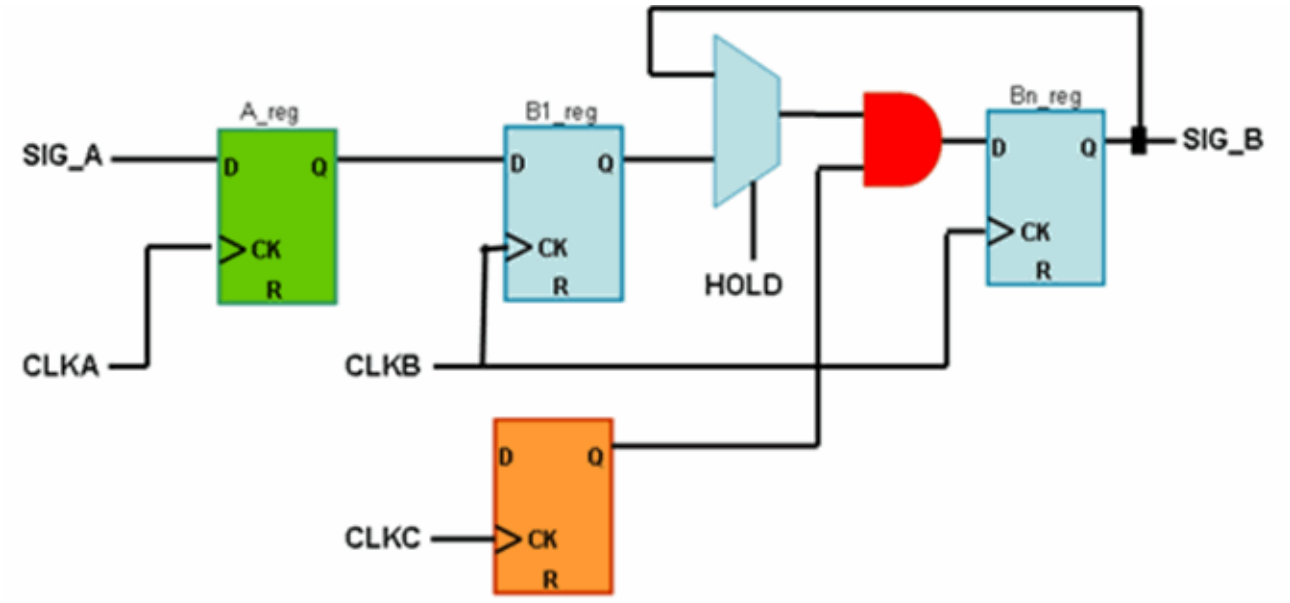
cdc_data_other_domain_check

Checks if the data path contains domains other than the source and destination domains.

Synchronization Scheme

MUX

Figure A-6 cdc_data_other_domain_check



© 1999-2023
All Rights Reserved.

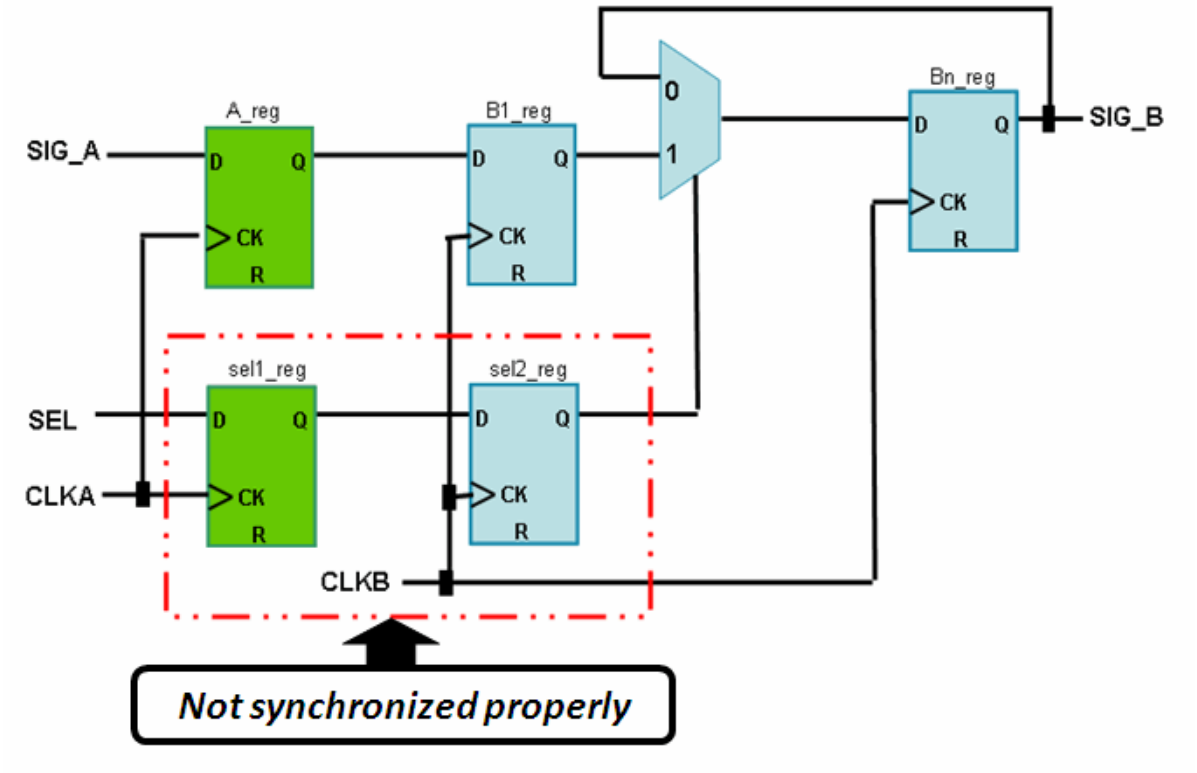
cdc_data_holding_sync_check

Checks that the holding logic for the data path is correctly synchronized.

Synchronization Scheme

MUX

Figure A-7 cdc_data_holding_sync_check (for MUX)



© 1999-2023
All Rights Reserved.

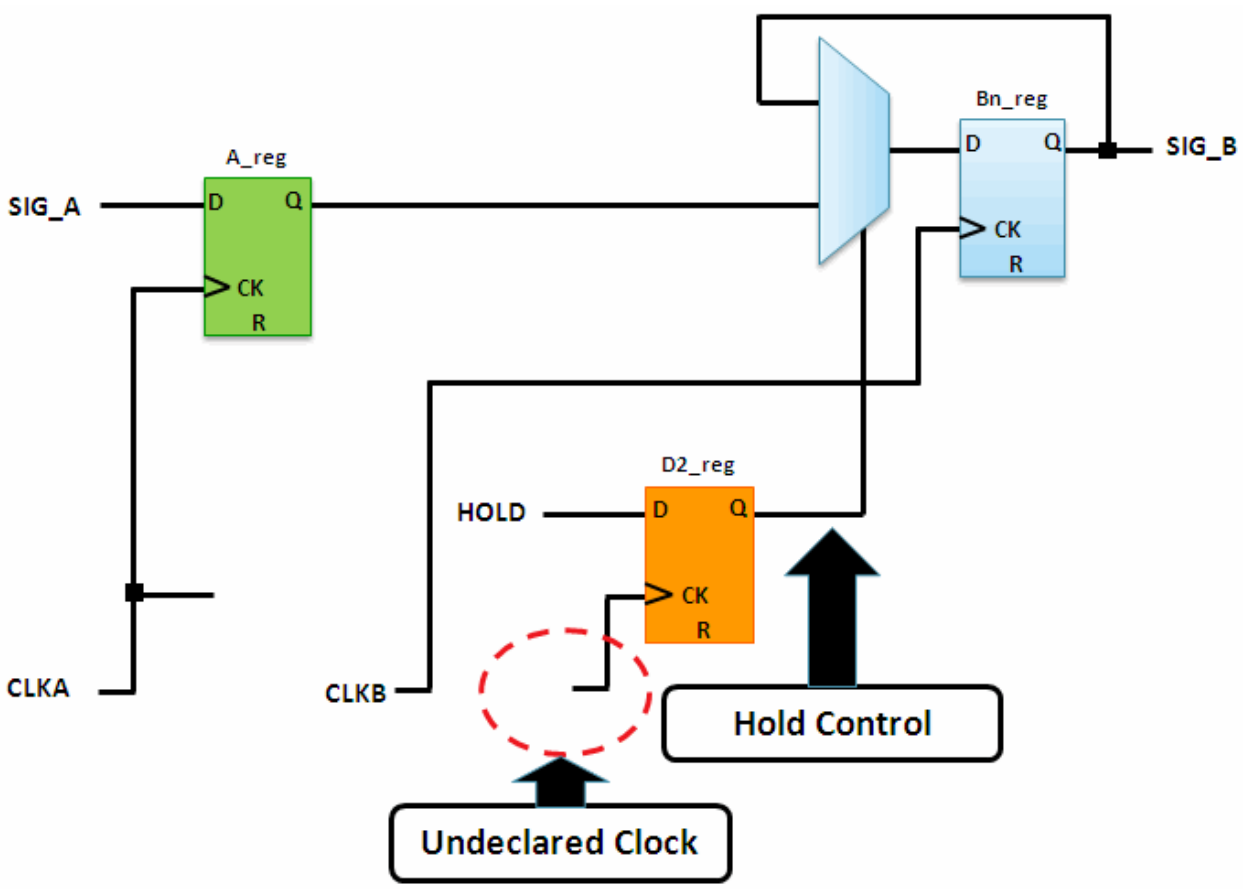
cdc_data_holding_unknown

Checks if unknown domains are propagated to hold control.

Synchronization Scheme

MUX

Figure A-8 cdc_data_holding_unknown



© 1999-2023
All Rights Reserved.

cdc_data_logic_type_check

Checks that the sync chain logic type in the CDC path matches the type specified by the `sync_chain_logic` attribute of the `cdc_datactrl_sync_rule` check.

Synchronization Scheme

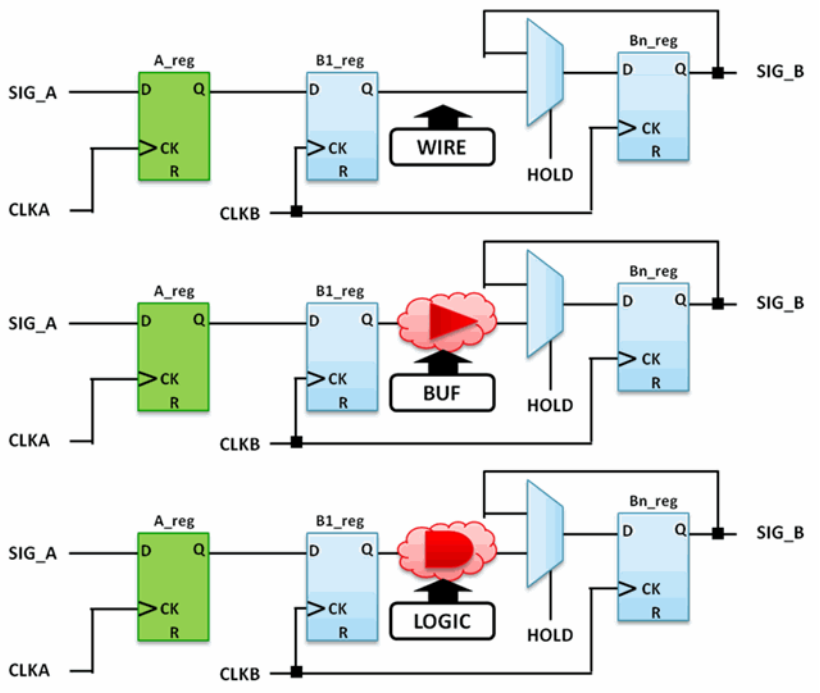
MUX

Example

For example, to change the logic type to buf:

```
set_attribute [find -ruleinst cdc_def_rs/cdc*] sync_chain_logic \
  "mux buffer"
```

Figure A-9 cdc_data_logic_type_check (for MUX)



© 1999-2023
All Rights Reserved.

cdc_data_min_sync_chain_check

Checks whether the sync chain has enough flops (as specified in the `mux_sync_scheme` attribute of the `cdc_datactrl_sync_rule` check).

Synchronization Scheme

MUX

Example

For example, to change the minimum required number of flops to 4:

```
set_attribute [find -ruleinst cdc_def_rs/cdc*] \
   mux_sync_scheme "min 4 max 4"
```


© 1999-2023 All Rights Reserved.

cdc_data_max_sync_chain_check

Checks whether the sync chain exceeds the maximum number of flops allowed in the sync chain (as specified in the `mux_sync_scheme` attribute of the `cdc_datactrl_sync_rule` check).

Synchronization Scheme

MUX

Example

For example, to change the maximum number of flops allowed to 6:

```
set_attribute [find -ruleinst cdc_def_rs/cdc*] mux_sync_scheme \
  "max 6"
```


© 1999-2023 All Rights Reserved.

cdc_data_first_dlatch_check

Checks whether the first sequential element in the destination matches the type specified in the `mux_sync_scheme` attribute of the `cdc_datactrl_sync_rule` check (DFF or a latch).

Synchronization Scheme

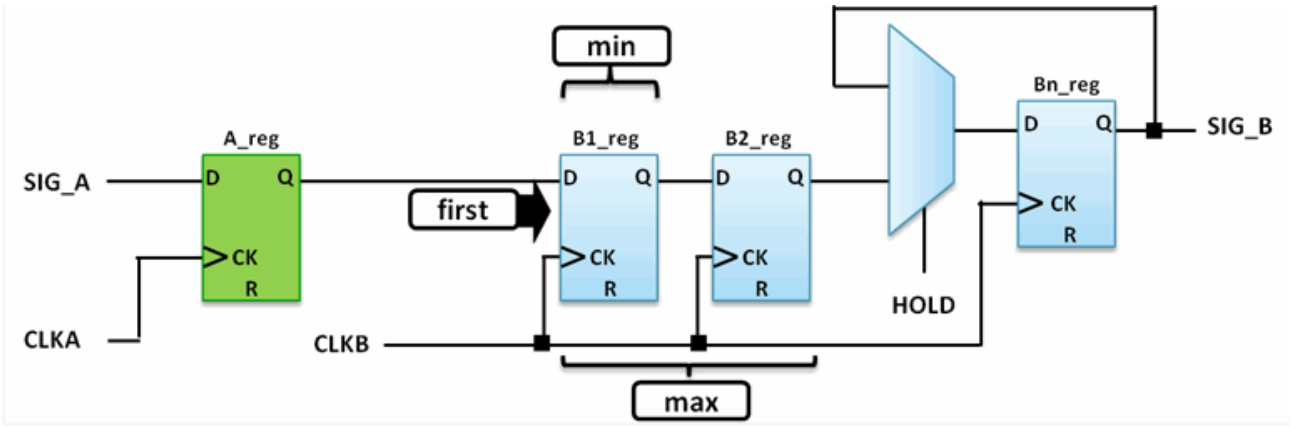
MUX

Example

To change the logic type:

```
set_attribute [find -ruleinst cdc_def_rs/cdc*] \
   mux_sync_scheme "first latch"
```

Figure A-10 cdc_data_min_sync_chain_check, cdc_data_max_sync_chain_check, and cdc_data_first_dlatch_check (for MUX)



© 1999-2023
All Rights Reserved.

cdc_data_multi_chain_check

Checks if the sync chain fanouts to multiple destinations.

Synchronization Scheme

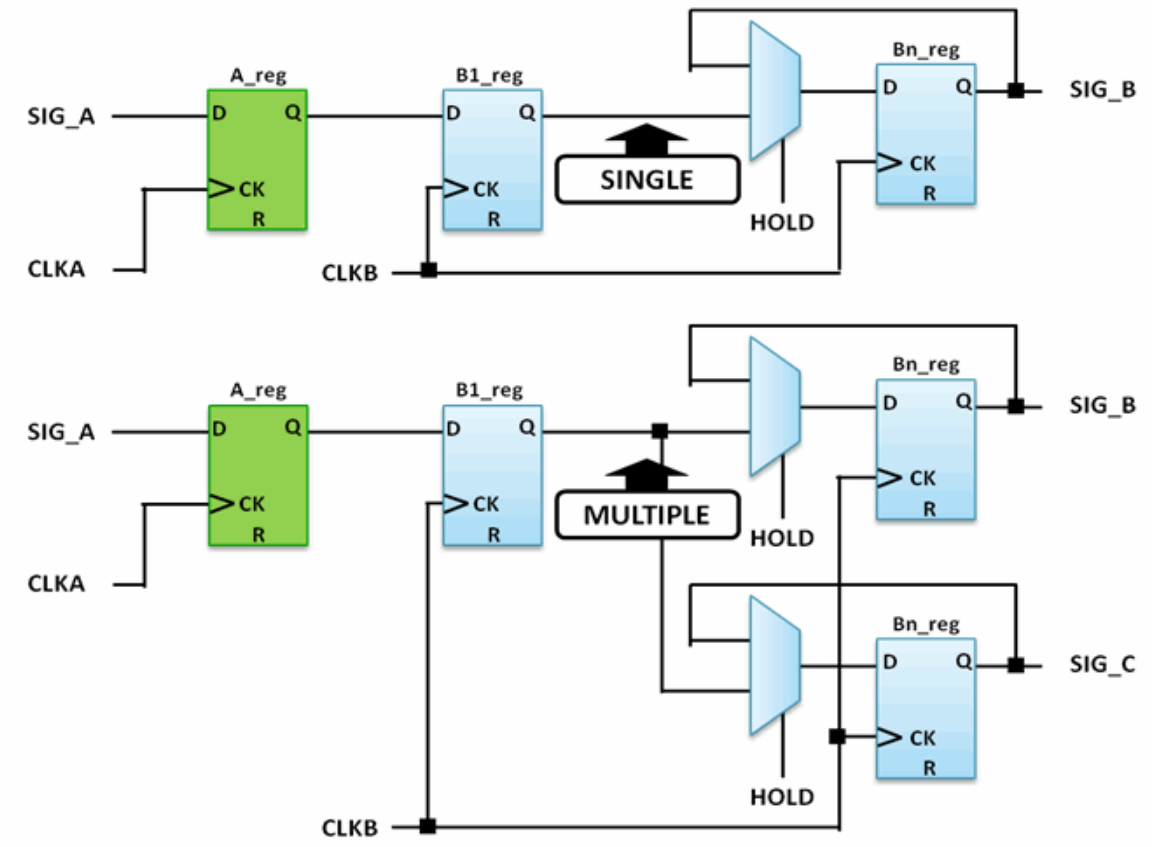
MUX

Example

To specify whether fanouts are allowed in the sync chain, use the `sync_chain_fanout` attribute of the `cdc_datactrl_sync_rule` rule check. For example:

```
set_attribute [find -ruleinst cdc_def_rs/cdc*] \
   sync_chain_fanout "mux multiple"
```

Figure A-11 cdc_data_multi_chain_check (for MUX)



© 1999-2023
All Rights Reserved.

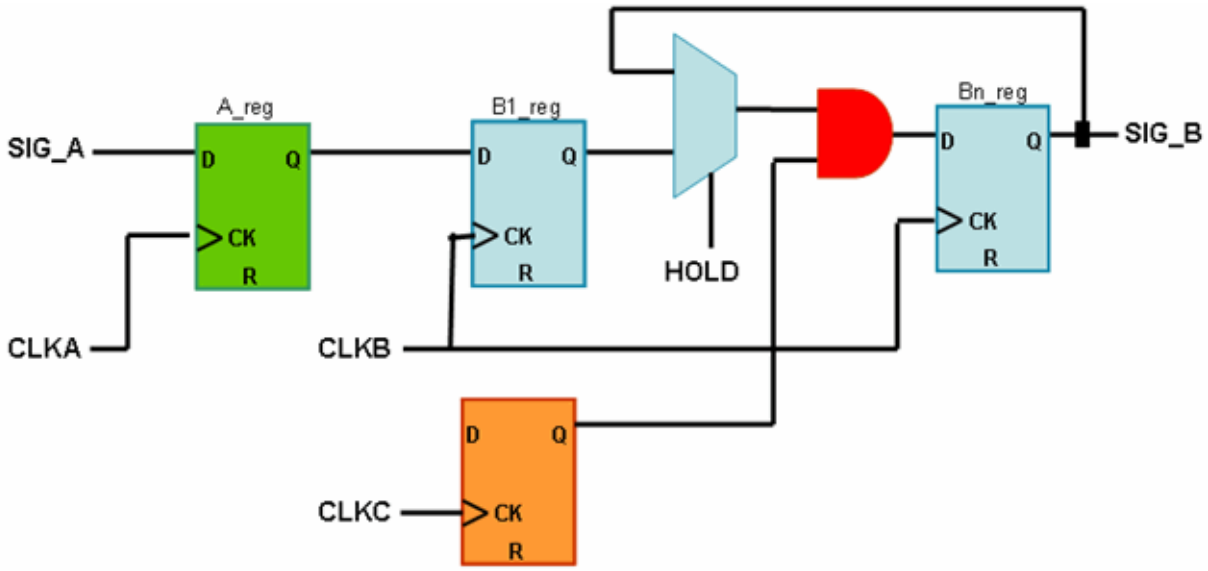
cdc_data_mixed_domain_check

Checks if all the signals in the sync-chain logic come from the destination clock domain of the crossing.

Synchronization Scheme

MUX

Figure A-12 cdc_data_mixed_domain_check (for MUX)



© 1999-2023
All Rights Reserved.

cdc_ctrl_logic_type_check

Checks that the logic type in the sync chain of the DFF synchronizer matches the type specified by the `sync_chain_logic` attribute of the `cdc_datactrl_sync_rule` check.

Synchronization Scheme

DFF

Example

In the following example, the `cdc_ctrl_logic_type_check` fails because there is a "`buffer`" in the design (line 9) when the `sync_chain_logic` attribute is set to "`wire`".

```
VERIFY> report rule check cdc_def_rs/cdc* -verbose -status fail
|  sync_chain_logic              : dff wire mux wire
...
...
|         structural : cdc_ctrl_logic_type_check           : BUFFER        : Fail
```

```
module test(data, clka, clkb, out);
input data;
input  clka, clkb;
output out;
reg out, din, din1;
wire din2;
always @(posedge clka)
  din <= data;
buf u0(din2,din1);
always @(posedge clkb)
begin
  din1 <= din;
  out <= din2;
end
endmodule
```

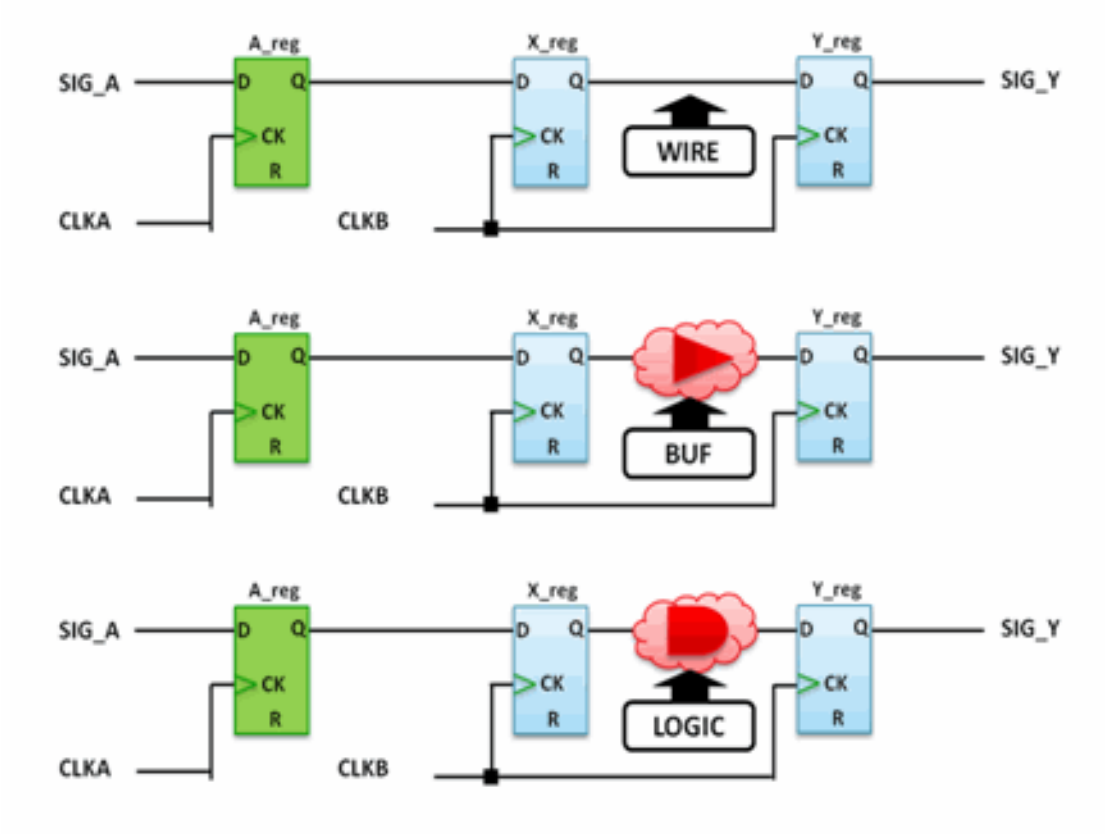
To resolve the mismatch flagged by the atomic check, use the following (if a buffer is allowed in the design):

```
set_attribute [ find -ruleinst cdc_def_rs/cdc*] sync_chain_logic "dff buffer"
```

If a buffer is not allowed, then this is a true bug and the design that needs to be fixed.

© 1999-2023 All Rights Reserved.

Figure A-13 cdc_ctrl_logic_type_check



© 1999-2023
All Rights Reserved.

cdc_ctrl_multi_chain_check

Checks that the destination (single or multiple) in the sync chain matches the type specified by the `sync_chain_fanout` attribute of the `cdc_datactrl_sync_rule` check.

Synchronization Scheme

DFF

Example

In the following example, the `cdc_ctrl_multi_chain_check` fails because the sync chain in the design has multiple fanouts (line 12) when the `sync_chain_fanout` attribute is set to "single".

```
VERIFY> report rule check cdc_def_rs/cdc* -verbose -status fail
|  sync_chain_fanout             : dff single mux single
...
...
|         structural : cdc_ctrl_multi_chain_check          : MULTIPLE      : Fail
```

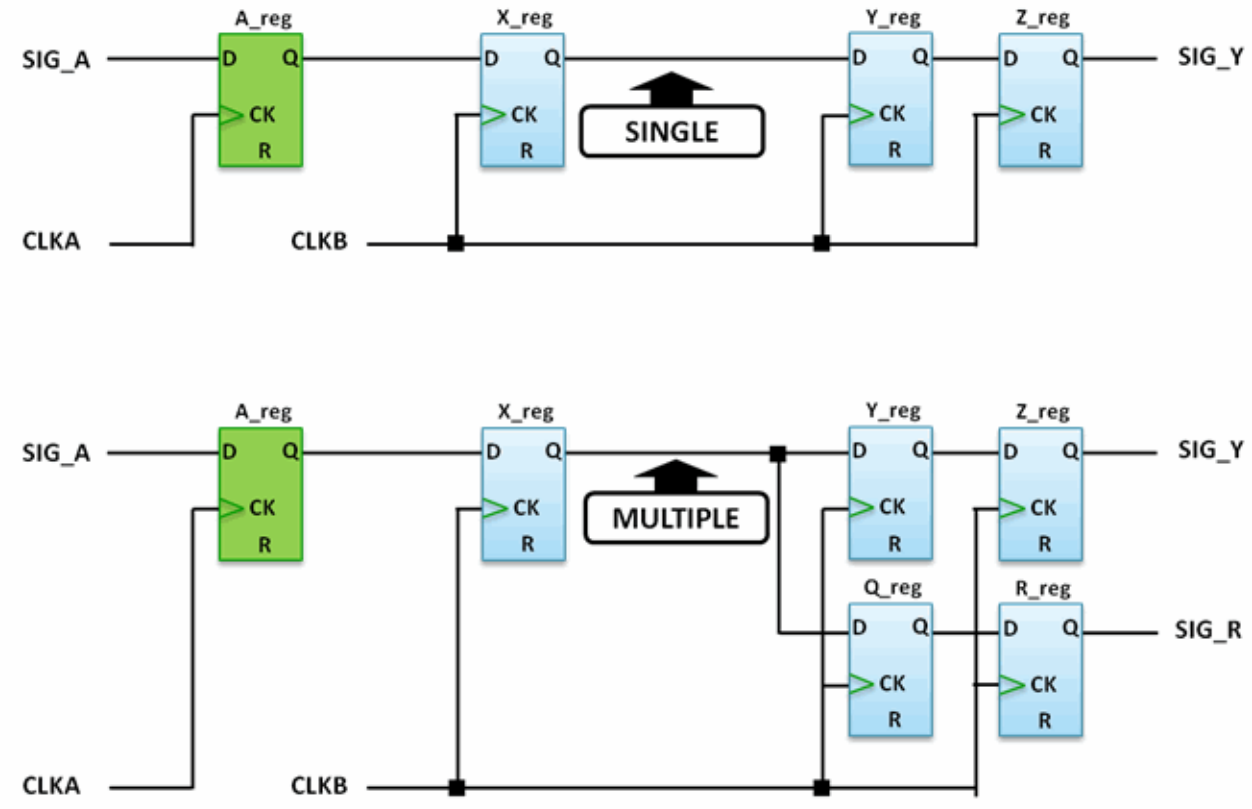
```
module test(data, clka, clkb, out);
input data;
input  clka, clkb;
output [1:0] out;
reg din, din1;
reg [1:0] out;
always @(posedge clka)
  din <= data;
always @(posedge clkb)
begin
  din1 <= din;
  out <= {din1,din1};
end
endmodule
```

To resolve the mismatch flagged by the atomic check (granted multiple chains are expected in the synchronization scheme; otherwise, this is a true bug and the design that needs to be fixed):

```
set_attribute [ find -ruleinst cdc_def_rs/cdc*] sync_chain_fanout "dff multiple"
```


© 1999-2023 All Rights Reserved.

Figure A-14 cdc_ctrl_multi_chain_check



© 1999-2023
All Rights Reserved.

cdc_ctrl_min_sync_chain_check

Checks that number of synchronizer DFFs in the sync chain meets with the minimum number specified by the `dff_sync_scheme` attribute of the `cdc_datactrl_sync_rule` check.

Synchronization Scheme

DFF

Example

In the following example, the `cdc_ctrl_min_sync_chain_check` fails because the DFF synchronizer in the design has only two DFFs in the design (line 10-11) when the `dff_sync_scheme` attribute specifies that the minimum is "3".

```
VERIFY> report rule check cdc_def_rs/cdc* -verbose -status fail

|  dff_sync_scheme               : min 3
...
...
|         structural : cdc_ctrl_min_sync_chain_check       : 2             : Fail
```

```
module test(data, clka, clkb, out);
input data;
input  clka, clkb;
output out;
reg out, din, din1;
always @(posedge clka)
  din <= data;
always @(posedge clkb)
begin
  din1 <= din;
  out <= din1;
end
endmodule

%cat dofile
add rule set -file ccd_default_cdc_ruleset.tcl
tclmode
set_attribute [ find -ruleinst cdc_def_rs/cdc*] dff_sync_scheme "min 3"
vpxmode
```

To resolve the mismatch flagged by the atomic check (granted the design expects a minimum depth of 2 instead of 3):

```
set_attribute [ find -ruleinst cdc_def_rs/cdc*] dff_sync_scheme "min 2"
```


© 1999-2023 All Rights Reserved.

cdc_ctrl_max_sync_chain_check

Checks that number of synchronizer DFFs in the sync chain does not exceed the maximum number specified by the `dff_sync_scheme` attribute of the `cdc_datactrl_sync_rule` check.

Synchronization Scheme

DFF

Example

In the following example, the `cdc_ctrl_max_sync_chain_check` fails because the DFF synchronizer in the design has three DFFs (lines 10-12), which exceeds the maximum number of DFFs (set to 2) allowed in the sync chain.

```
VERIFY> report rule check cdc_def_rs/cdc* -verbose -status fail

|  dff_sync_scheme               : min 2 max 2
...
...
|         structural : cdc_ctrl_max_sync_chain_check       : 3             : Fail
```

```
module test(data, clka, clkb, out);
input data;
input  clka, clkb;
output out;
reg out, din, din1, din2;
always @(posedge clka)
  din <= data;
always @(posedge clkb)
begin
  din1 <= din;
  din2 <= din1;
  out <= din2;
end
endmodule

%cat dofile
add rule set -file ccd_default_cdc_ruleset.tcl
tclmode
set_attribute [ find -ruleinst cdc_def_rs/cdc*] dff_sync_scheme "min 2 max 2"
vpxmode
```

To resolve the mismatch flagged by the atomic check (granted that the design expects a maximum depth of 3 instead of 2):

```
set_attribute [ find -ruleinst cdc_def_rs/cdc*] dff_sync_scheme "min 2 max 3"
```

If the synchronization maximum check does not need to be checks, specify the max to "`infinite`".

© 1999-2023 All Rights Reserved.

cdc_ctrl_first_dlatch_check

Checks that the first sequential element type (DFF or latch) of DFFs in the sync chain matches the type specified by the `dff_sync_scheme` attribute of the `cdc_datactrl_sync_rule` check.

Synchronization Scheme

DFF

Example

In the following example, the `cdc_ctrl_first_dlatch_check` fails because the first sequential element of the DFF synchronizer in the design is a latch (lines 8-11) when the `dff_sync_scheme` attribute is set to the first as "`dff`".

```
VERIFY> report rule check cdc_def_rs/cdc* -verbose -status fail

|  dff_sync_scheme               : min 2 max infinite first dff
...
...

|          structural : cdc_ctrl_first_dlatch_check : DLATCH : Fail
```

```
module test(data, clka, clkb, out);
input data;
input  clka, clkb;
output out;
reg out, din, din1,din2;
always @(posedge clka)
  din <= data;
always @(clkb)
begin
  if (clkb) din1 <= din;
end
always @(posedge clkb)
begin
  din2 <= din1;
  out <= din2;
end
endmodule
```

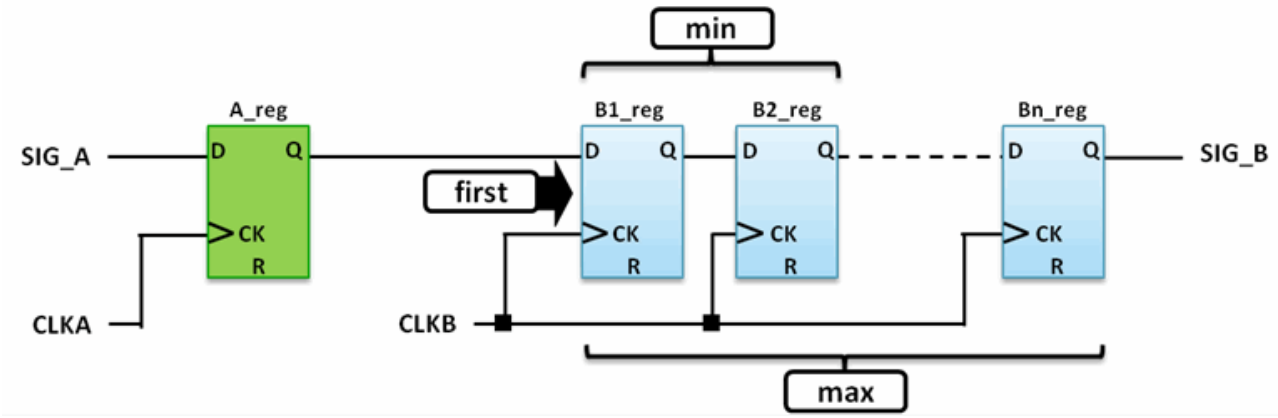
To resolve the mismatch flagged by the atomic check, use the following (if a latch is allowed):

```
set_attribute [ find -ruleinst cdc_def_rs/cdc*] dff_sync_scheme \
  "min 2 max infinite first latch"
```

If a latch is not allowed in the synchronization chain, then this is a true bug and the design that needs to be fixed.

© 1999-2023 All Rights Reserved.

Figure A-15 cdc_ctrl_min_sync_chain_check, cdc_ctrl_max_sync_chain_check, and cdc_ctrl_first_dlatch_check



© 1999-2023
All Rights Reserved.

cdc_ctrl_mixed_domain_check

Checks that all of the signals in the sync chain logic come from the destination clock domains.

Synchronization Scheme

DFF

Example

In the following example, the `cdc_ctrl_mixed_domain_check` fails because there are multiple clock domain sources (`clkb` for `din1`, and `clkc` for `enai`) driving the function "and" in the design (line 8). All of the signals in the sync chain logic needs to come from the destination clock domains (`clkb`).

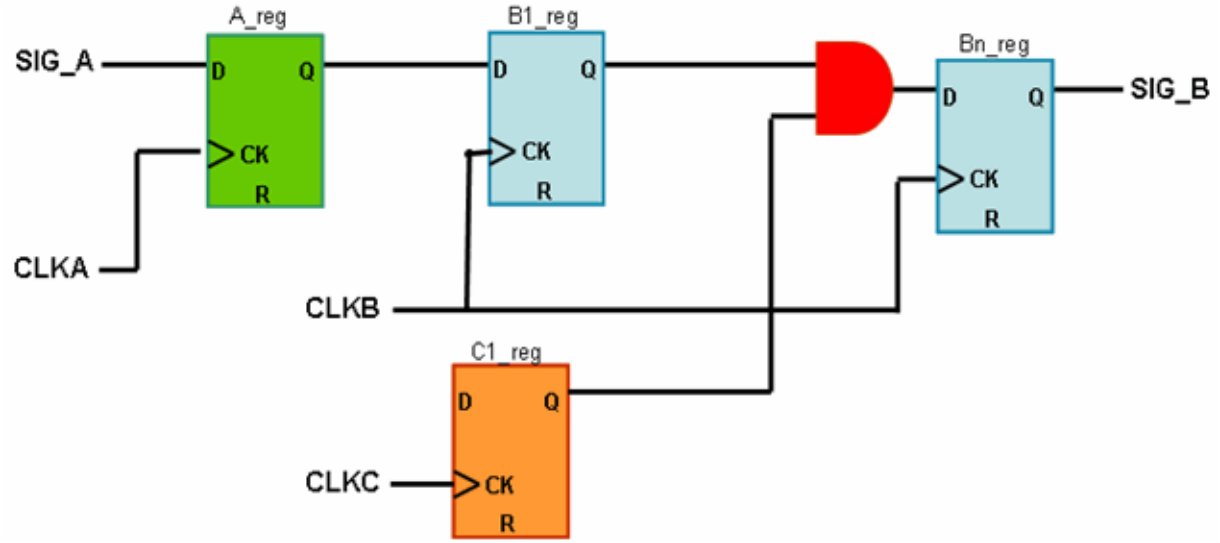
```
VERIFY> report rule check cdc_def_rs/cdc* -verbose -status fail
|          structural : cdc_ctrl_mixed_domain_check : MIXED : Fail

module test(data, ena, clka, clkb, clkc, out);
input data ,ena ;
input  clka, clkb, clkc;
output out;
reg out, din, din1,enai;
wire din2 ;
always @(posedge clka) din <= data;
assign din2 = din1 & enai;
always @(posedge clkb)
begin
  din1 <= din;
  out <= din2;
end
always @(posedge clkc) enai <= ena;
endmodule
```

To resolve the mismatch flagged by the atomic check, investigate whether mixed domains are expected. If so, check whether the clock grouping is correct using the `REPORT CLOCK GROUPS` command. Otherwise, this is a true bug that requires a change in the design.

© 1999-2023 All Rights Reserved.

Figure A-16 cdc_ctrl_mixed_domain_check



© 1999-2023
All Rights Reserved.

cdc_conv_in_vector_check

Checks whether convergence is from the same vector (invector convergence).

Synchronization Scheme

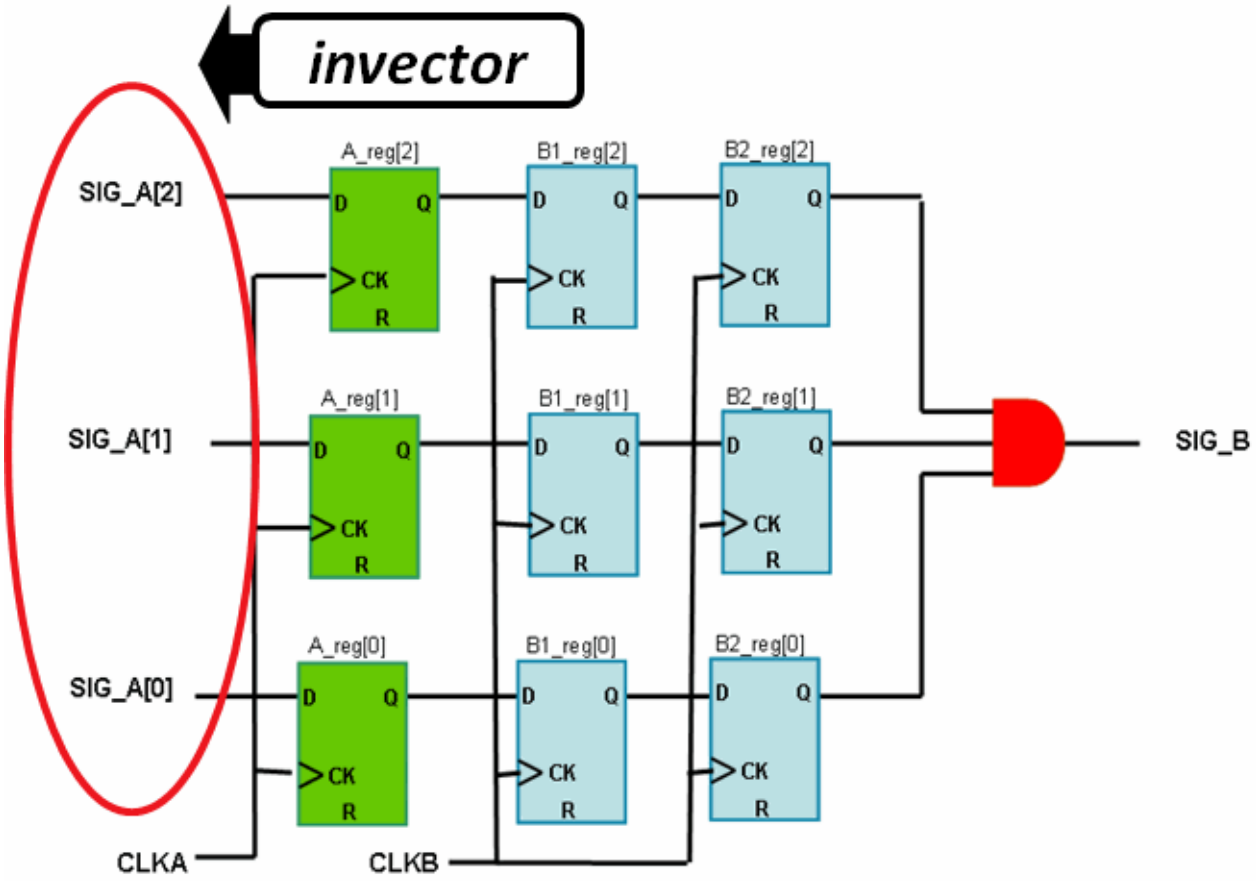
CONV

Example

For example, to specify outvector convergence:

```
set_attribute [find -ruleinst cdc_def_rs/conv*] \
   check_vector_conv outvector
```

Figure A-17 cdc_conv_in_vector_check



© 1999-2023
All Rights Reserved.

cdc_conv_out_vector_check

Checks whether convergence is from different vectors (outvector convergence).

Synchronization Scheme

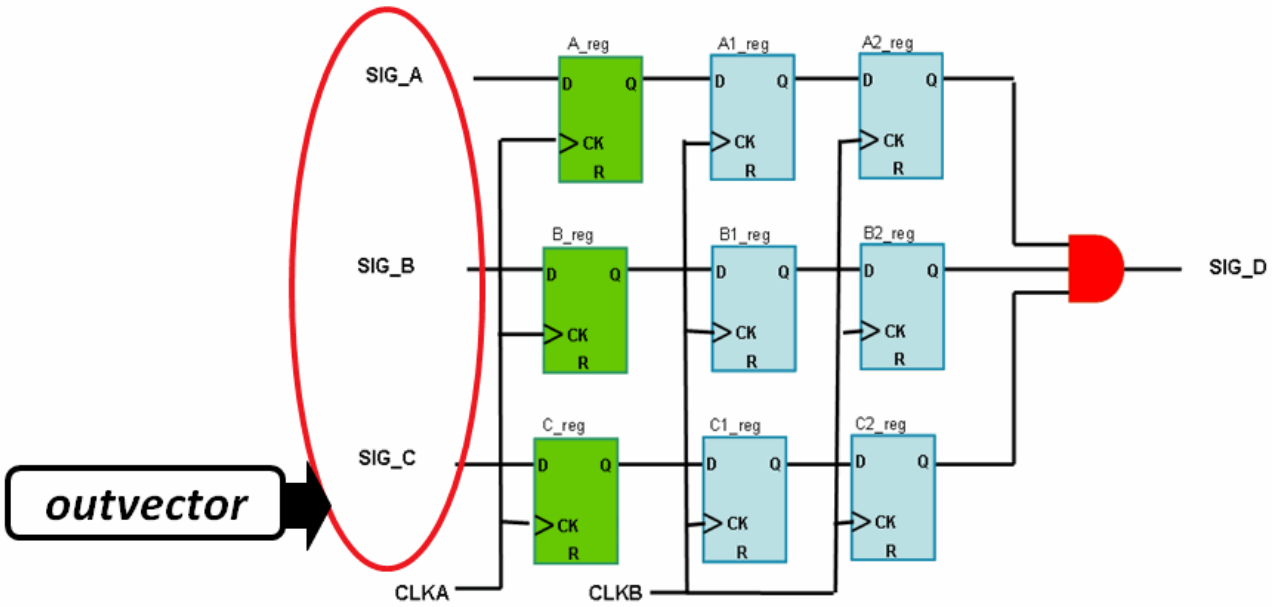
CONV

Example

For example, to specify outvector convergence:

```
set_attribute $rule_instance check_vector_conv outvector
```

Figure A-18 cdc_conv_out_vector_check



© 1999-2023
All Rights Reserved.

cdc_conv_same_domain_check

Checks for convergence from the same source clock domain.

Synchronization Scheme

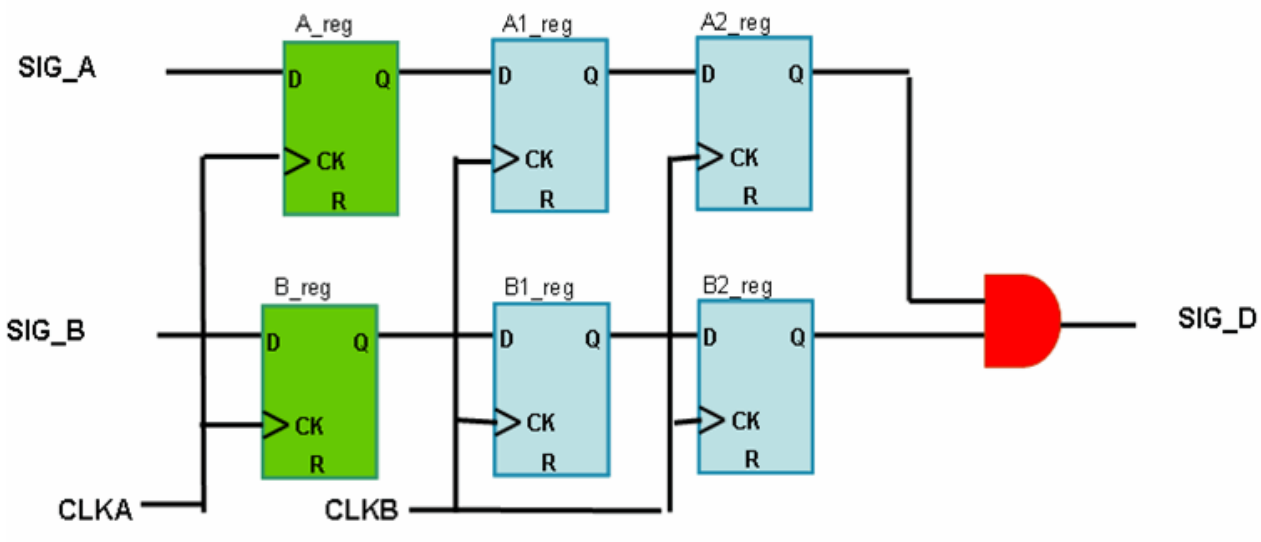
CONV

Example

For example, to specify that convergence comes from the same source clock domain:

```
set_attribute [find -ruleinst cdc_def_rs/conv*] \
   check_conv_type same_clock_groups_conv
```

Figure A-19 cdc_conv_same_domain_check



© 1999-2023
All Rights Reserved.

cdc_conv_diff_domain_check

Checks for convergence from any source clock domain.

Synchronization Scheme

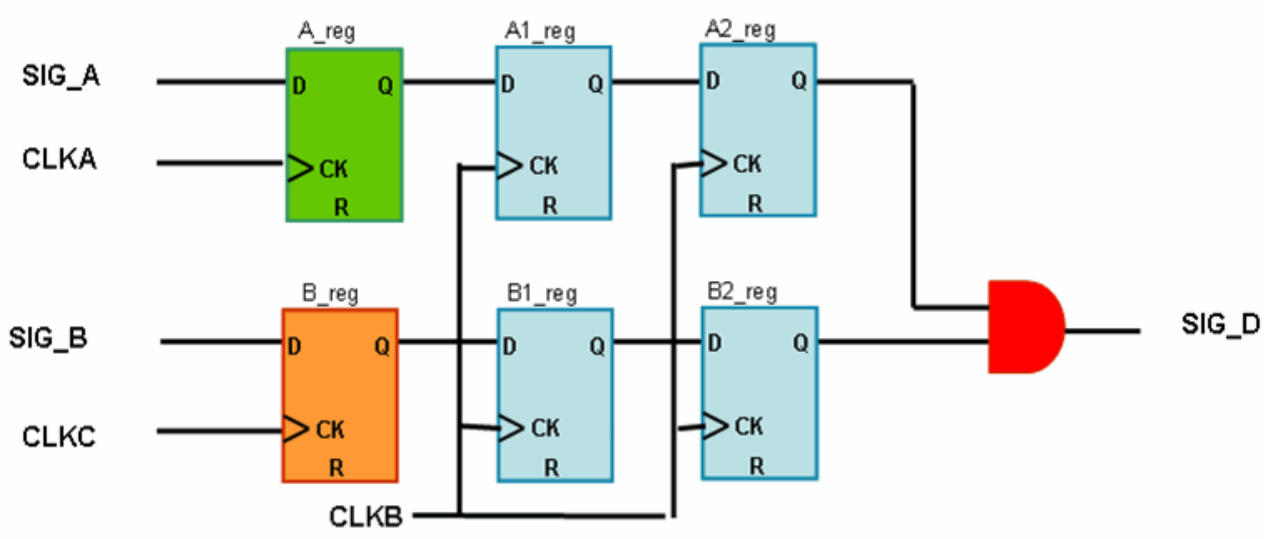
CONV

Example

For example, to specify that convergence can come from a difference source clock domain:

```
set_attribute [find -ruleinst cdc_def_rs/conv*] \
   check_conv_type diff_clock_groups_conv
```

Figure A-20 cdc_conv_diff_domain_check



© 1999-2023
All Rights Reserved.

cdc_conv_source_filtered

Indicates that the specified convergence path has been filtered. Filtered paths are specified using the `filter_paths` attribute of the `cdc_conv_check_rule` rule check.

Synchronization Scheme

CONV

Example

To specify the convergence paths to filter:

```
set_attribute $rule_instance filter_paths [list \
    [from [find -instance A_reg] to \
    [find -instance B1_reg]] ]
```


© 1999-2023 All Rights Reserved.

cdc_setreset_deassertion_check

In the case where set/reset synchronization chain asynchronously asserts the set/reset of the destination sequential elements, checks whether the set/reset can be synchronously de-asserted.

This check does not run by default (it is included in the `excluded_atomic_check` attribute). To enable this atomic check, remove it from the `excluded_atomic_check` attribute for the specific rule instance.

Synchronization Scheme

Set Reset

© 1999-2023 All Rights Reserved.

cdc_setreset_min_dff_check

Checks whether the sync chain has enough flops (as specified in the `dff_sync_scheme` attribute of the `cdc_setreset_sync_rule` check).

Synchronization Scheme

Set Reset

Example

To specify the minimum number of required flops in the sync chain:

```
set_attribute $rule_instance dff_sync_scheme [min #]
```

Where # is 2 or more.

© 1999-2023 All Rights Reserved.

cdc_setreset_mixed_domain_check

This atomic check checks if all the signals to the cascaded chain (for more information on cascade chains, refer to `sr_always_consider_cascaded_chain` attribute) are coming from the clock domain of the target flip flop.

Synchronization Scheme

Set Reset

© 1999-2023 All Rights Reserved.

cdc_setreset_pi_association_check

This check is performed on the PI that drives the set/reset pin; it checks whether the PI is associated with the correct clock.

Synchronization Scheme

Set Reset

© 1999-2023 All Rights Reserved.

cdc_setreset_sync_chain_mix_sr_check

Checks that there is only one source driving the set/reset port of all the synchronizers. If there are different sources driving the set/reset port of synchronizers, this check fails.

Synchronization Scheme

Set Reset

© 1999-2023 All Rights Reserved.

cdc_setreset_sync_chain_nosr_check

Checks whether both set and resets are disabled for the first (from the set/reset driver side) sync chain register.

Synchronization Scheme

Set Reset

© 1999-2023 All Rights Reserved.

cdc_setreset_logic_type_check

Checks whether the logic type in the sync chain (from driver to the first key point) is as specified in the `allowed_logic` attribute of the `cdc_setreset_sync_rule` check.

Synchronization Scheme

Set Reset

Example

To specify the logic type in the sync chain:

```
set_attribute $rule_instance [allowed_logic <wire| inv |logic>]
```


© 1999-2023 All Rights Reserved.

cdc_setreset_source_driver_check

Checks if the source driver of the synchronizer exists in the user provided source register list.

Synchronization Scheme

Set Reset

© 1999-2023
All Rights Reserved.

cdc_setreset_target_clock_sync_check

Checks whether the clock of the set/reset synch chain registers is synchronous with the target registers. A target register is a register whose set/reset port is driven by the set/reset sync chain.

Synchronization Scheme

Set Reset

© 1999-2023 All Rights Reserved.

cdc_user_sync_module_check

Checks for user sync modules for sync-chain.

Synchronization Scheme

DFF, MUX, and Set Reset

© 1999-2023 All Rights Reserved.

cdc_inactive_path_check

Checks the data input of the source or destination flip-flop is a tied constant. If the data input is tied, this check passes.

When a signal is constant (inactive), it does not have any potential metastability issues even if it is clocked by asynchronous clocks.

Synchronization Scheme

DFF

Example

In the following example, `cdc_inactive_path_check` fails because the data input to clock `clka` is not a tied constant:

```
module test(clka, clkb, out);
input  clka, clkb;
output out;
reg out, din, din1;
always @(posedge clka) din <= 1'b1;
always @(posedge clkb)
begin
  din1 <= din;
  out <= din1;
end
endmodule
```


© 1999-2023 All Rights Reserved.

cdc_fifo_crossing_check

Checks if the CDC path is a FIFO crossing.

Synchronization Scheme

DFF, MUX, and Set Reset

© 1999-2023
All Rights Reserved.

cdc_clock_group_check

Checks if the CDC path is from the same clock group.

Synchronization Scheme

DFF, MUX, and Set Reset

© 1999-2023 All Rights Reserved.

cdc_path_not_processed

Checks if crossing has not been validated.

Synchronization Scheme

DFF, MUX, and Set Reset

© 1999-2023 All Rights Reserved.

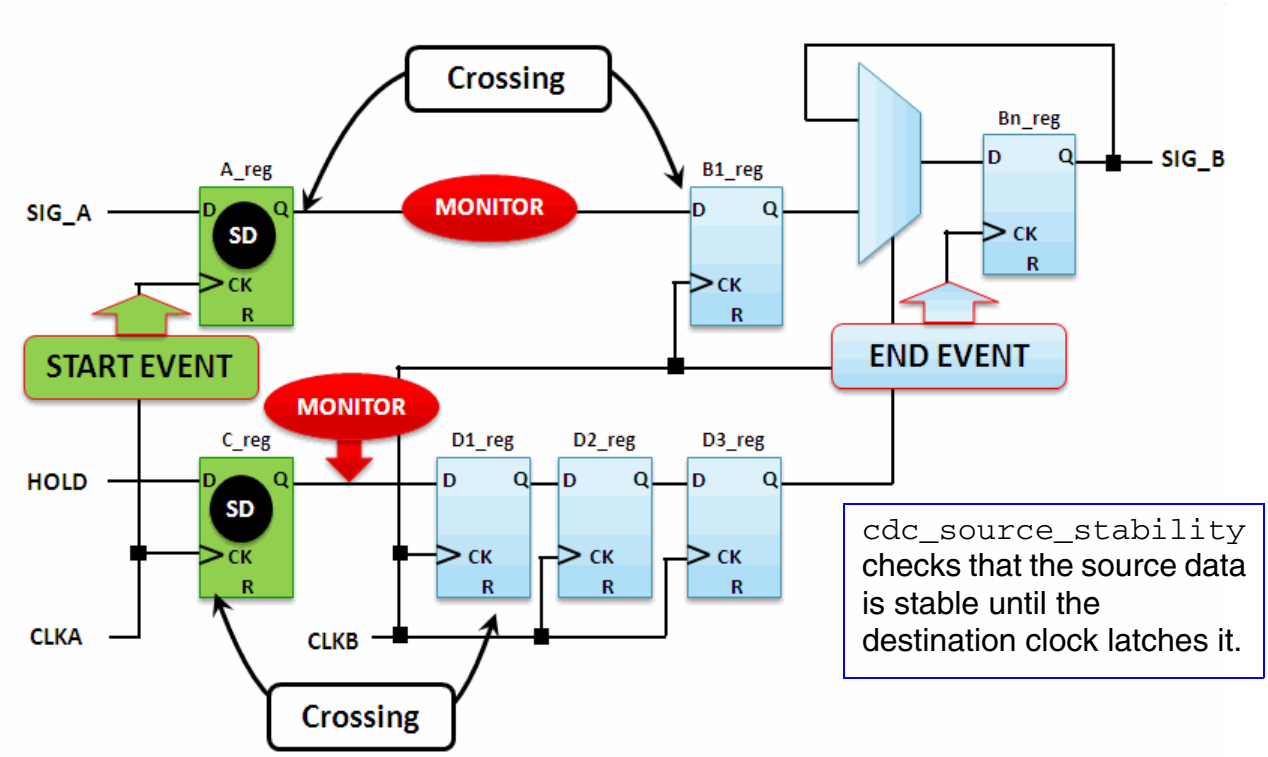
cdc_source_stability

Functionally verifies that the data that leaves the source register (that is, the output of the source register) is held stable for the destination register to latch it.

Synchronization Scheme

DFF and MUX

Figure A-21 cdc_source_stability



© 1999-2023
All Rights Reserved.

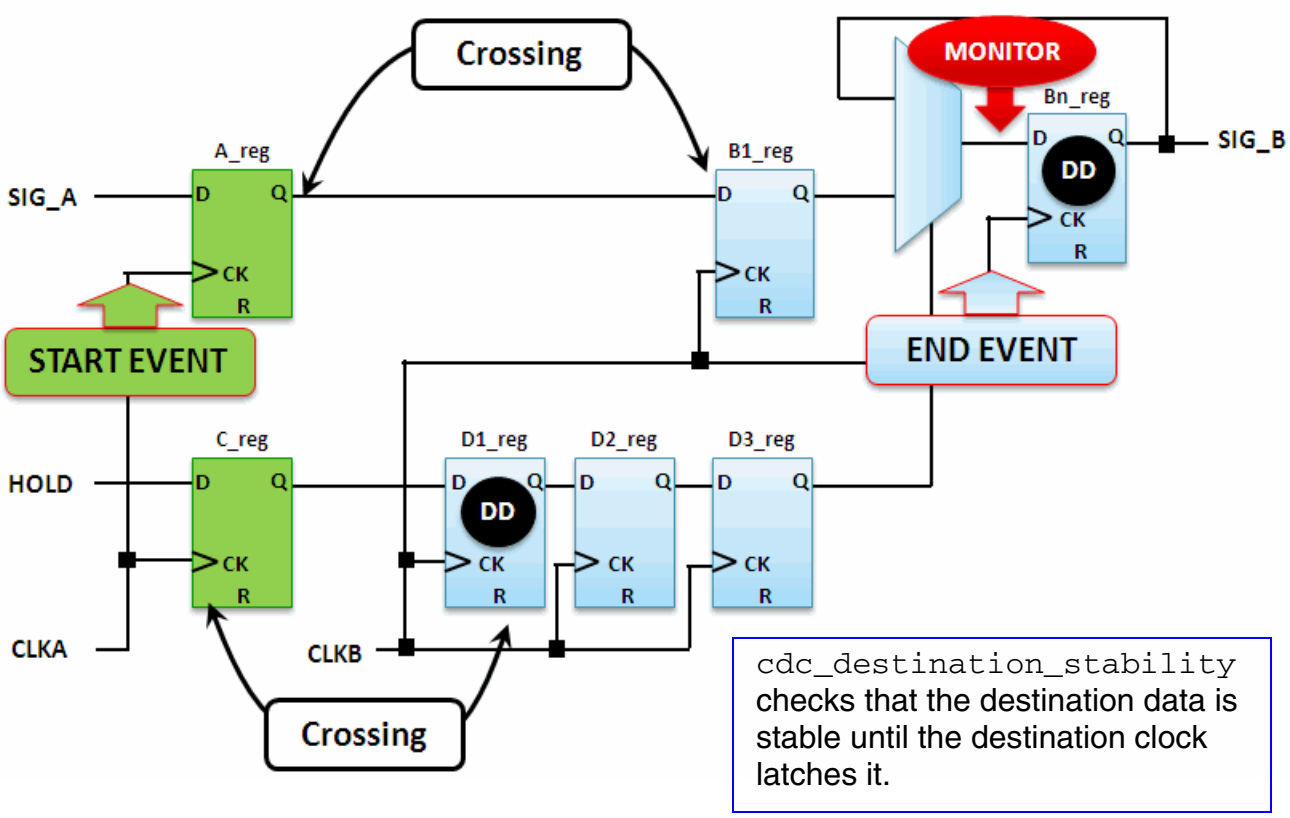
cdc_destination_stability

Functionally verifies that the data entering the destination register is held stable for the destination data to latch it.

Synchronization Scheme

DFF and MUX

Figure A-22 cdc_destination_stability



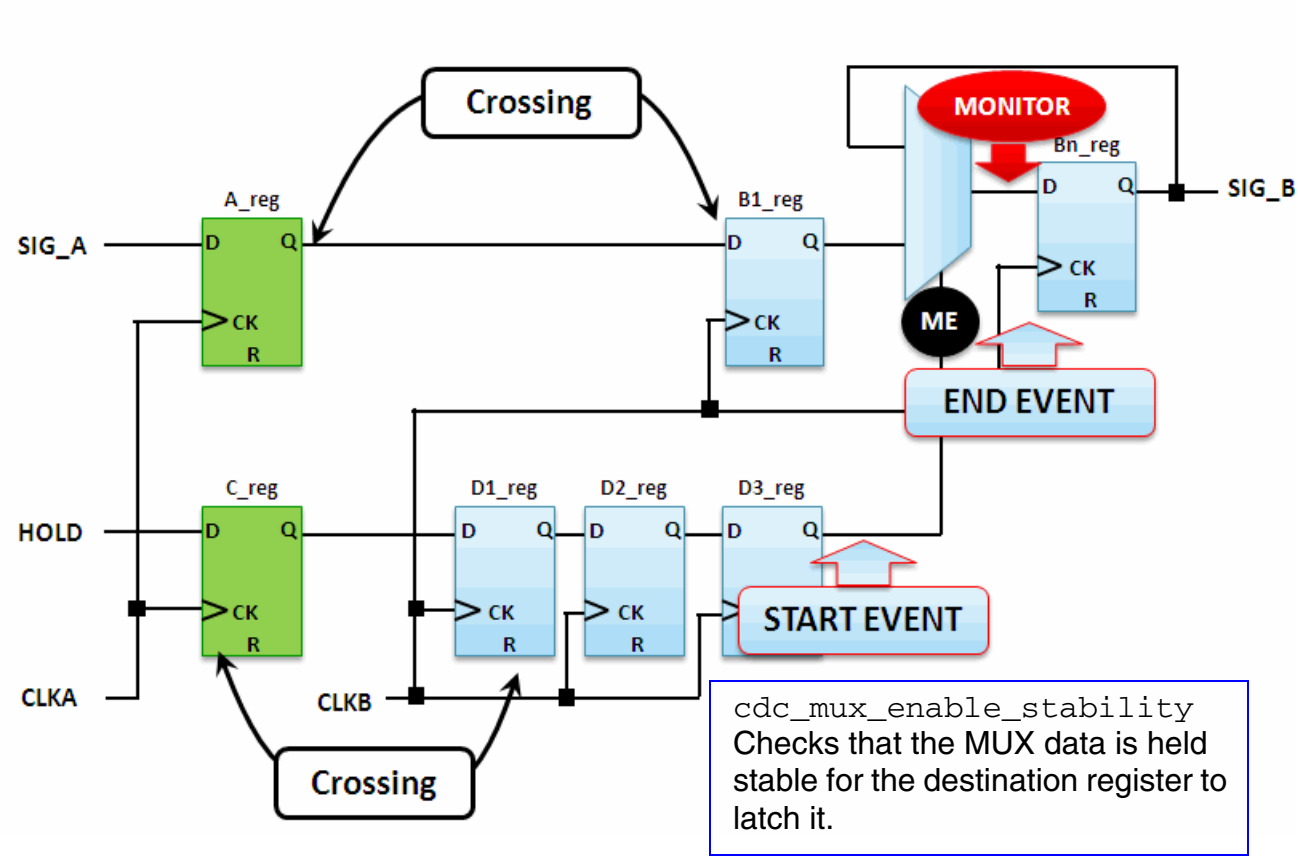
cdc_mux_enable_stability

Functionally verifies that the data at the output of the multiplexer synchronizer and the select line of the synchronizer are held stable (after changing at the select input) for the destination register to latch it.

Synchronization Scheme

MUX

Figure A-23 cdc_mux_enable_stability



© 1999-2023
All Rights Reserved.

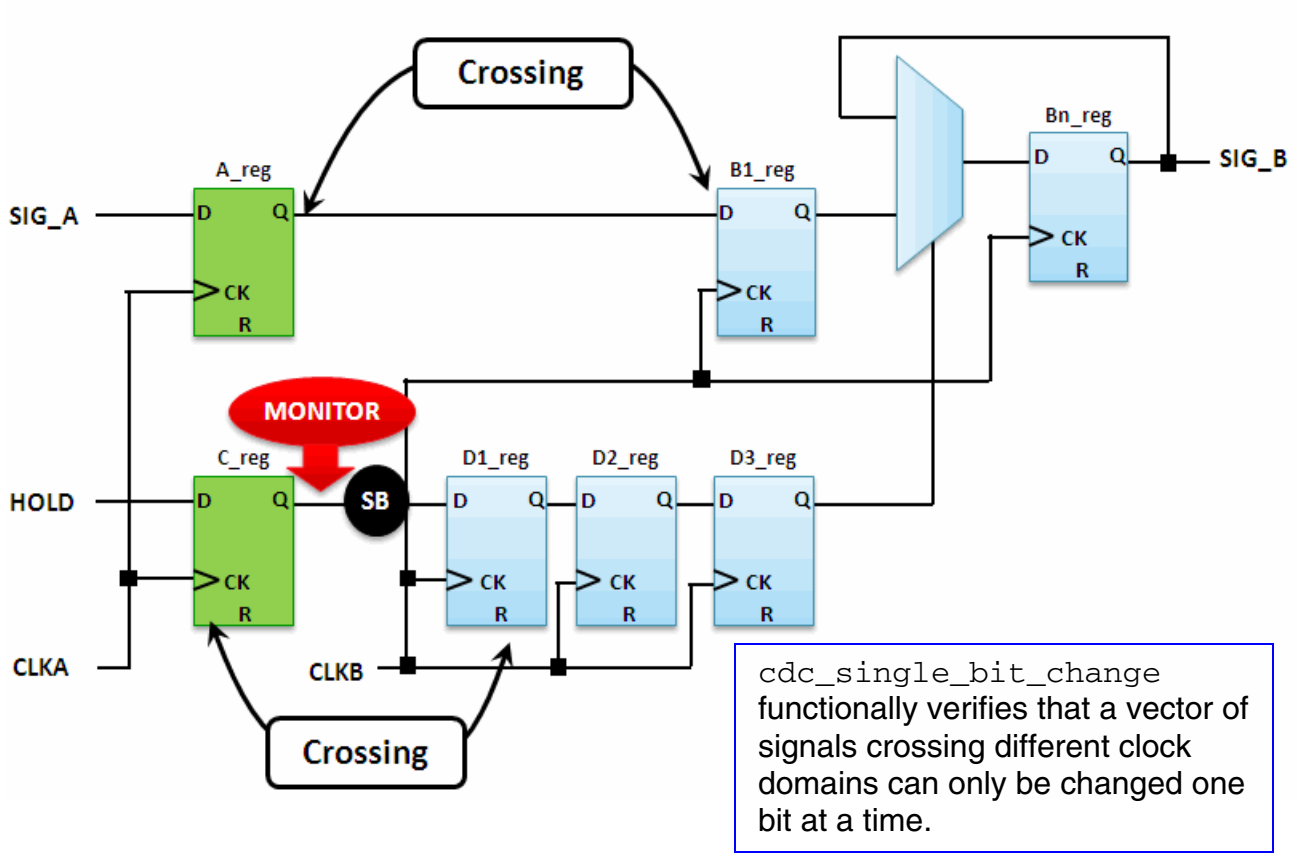
cdc_single_bit_change

Functionally verifies that a vector of signals crossing different clock domains can only be changed one bit at a time.

Synchronization Scheme

DFF and CONV

Figure A-24 cdc_single_bit_change



© 1999-2023
All Rights Reserved.

FIFO Atomic Checks

Alphabetical List of FIFO Atomic Checks

The following is the list of all the FIFO-related atomic checks, listed alphabetically. For a list of atomic checks listed by FIFO attribute, refer to "FIFO Atomic Checks by Attribute" on page 530.

Each atomic check has an ID #; this number is used in Figure A-24 on page 531 to help illustrate how each part of the FIFO synchronization scheme is covered by an atomic check.

#	Name	Description
1	`fifo_chk_atomic_gray_comb_loop`	Checks that there is a combinational loop for a gray code register
	`fifo_chk_atomic_gray_func`	Checks the functionality of FIFO read/ write code registers
2	`fifo_chk_atomic_gray_size`	Checks that the gray code size is equal to or greater than the minimum read graycode size
3	`fifo_chk_atomic_async_mem`	Checks if the memory and output registers are asynchronous
4	`fifo_chk_atomic_mem_out_exclusive`	Checks if the memory and output registers are exclusive
5	`fifo_chk_atomic_mem_out_size`	Checks if the memory size is a multiple of the output size.
6	`fifo_chk_atomic_mem_size`	Checks that the memory size is equal to or larger than the minimum memory size
7	`fifo_chk_atomic_mem_supported_cell_type`	Checks if all the element types are supported cell types
8	`fifo_chk_atomic_mem_two_dimension`	Checks if the memory is a two-dimensional register array

© 1999-2023 All Rights Reserved.

9	fifo_chk_atomic_out_size	Checks that the output size is equal to or greater than the minimum output size.
10	fifo_chk_atomic_readptr_size	Checks that the read point size is equal to or larger than the minimum read pointer size
11	fifo_chk_atomic_readptr_sync	Checks if the read pointer is synchronous to the output
12	fifo_chk_atomic_single_readptr	Checks if there is only one read pointer candidate.
13	fifo_chk_atomic_single_rgray	Checks if there is only one read graycode register candidate
14	fifo_chk_atomic_single_sync	Checks if there is only one sync candidate
15	fifo_chk_atomic_single_wgray	Checks if there is only one write graycode register candidate
16	fifo_chk_atomic_single_writeptr	Checks if there is only one write pointer candidate
17	fifo_chk_atomic_sync_size	Checks if the sync size is equal to the gray code candidate size
18	fifo_chk_atomic_wdata_size	Checks if the minimum wdata size has been exceeded
19	fifo_chk_atomic_writeptr_size	Checks if the write point size is equal to or greater than the minimum write pointer size
20	fifo_chk_atomic_writeptr_sync	Checks if the write pointer is synchronous to the memory

© 1999-2023 All Rights Reserved.

FIFO Atomic Checks by Attribute

The following lists the FIFO attributes and the checks that prove them.

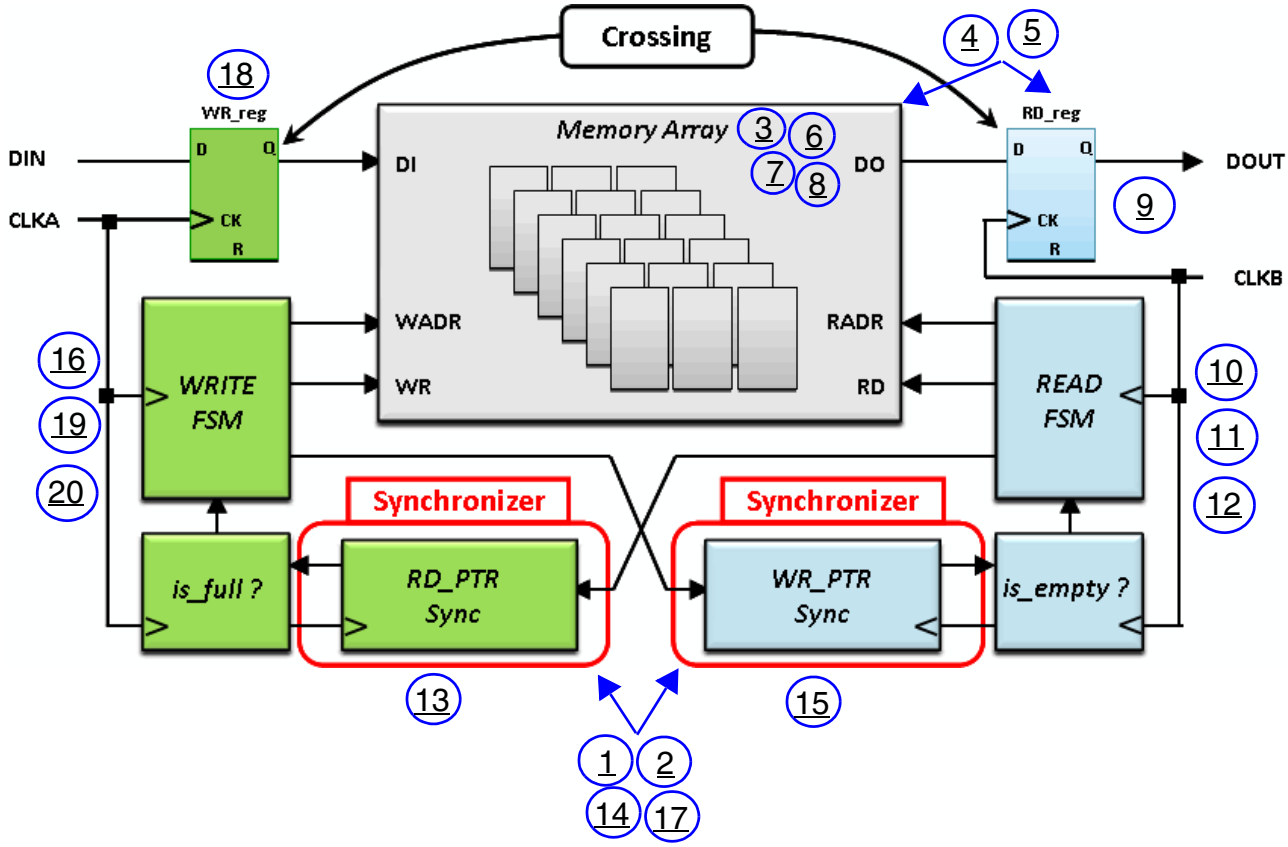
Attribute Name	Applicable Atomic Checks
atomic_check_fifo	fifo_chk_atomic_single_readptr
	fifo_chk_atomic_single_writeptr
	fifo_chk_atomic_single_rgray
	fifo_chk_atomic_single_wgray
	fifo_chk_atomic_mem_two_dimension
	Note that the checks done to prove atomic_chk_fifo are global, in that they apply to the entire FIFO instance. The checks listed below apply to particular components within the FIFO instance.
atomic_check_memory	fifo_chk_atomic_mem_size
	fifo_chk_atomic_mem_supported_cell_type
	fifo_chk_atomic_out_size
	fifo_chk_atomic_mem_out_size
	fifo_chk_atomic_mem_out_exclusive
	fifo_chk_atomic_async_mem
atomic_check_wdata	fifo_chk_atomic_wdata_size
atomic_check_raddr	fifo_chk_atomic_readptr_size
	fifo_chk_atomic_readptr_sync
atomic_check_waddr	fifo_chk_atomic_writeptr_size
	fifo_chk_atomic_writeptr_sync
atomic_check_rgray	fifo_chk_atomic_gray_comb_loop
	fifo_chk_atomic_single_sync
	fifo_ck_atomic_gray_size
	fifo_chk_atomic_sync_size

© 1999-2023 All Rights Reserved.

atomic_chk_wgray	fifo_chk_atomic_gray_comb_loop
	fifo_chk_atomic_single_sync
	fifo_chk_atomic_gray_size
	fifo_chk_atomic_sync_size

FIFO Synchronization Scheme

The following figure illustrates the relative portion of the FIFO that each atomic check, described in "FIFO Atomic Checks" on page 528, covers.



© 1999-2023
All Rights Reserved.