



SDC and TimeQuest API Reference Manual



101 Innovation Drive
San Jose, CA 95134
www.altera.com

MNL-SDCTMQ-5.0

Copyright © 2009 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

About this Reference Manual

Revision History	1-vii
How to Contact Altera	1-vii
Typographic Conventions	1-viii

Chapter 1. Introduction to the SDC and TimeQuest API Reference Manual

Introduction	1-1
What's Inside the SDC and TimeQuest API Reference Manual?	1-1
TimeQuest Timing Analyzer Scripting Support	1-1
Related Documentation	1-2

Chapter 2. SDC and TimeQuest API Package and Commands

sdc	2-1
all_clocks	2-3
all_inputs	2-4
all_outputs	2-5
all_registers	2-6
create_clock	2-7
create_generated_clock	2-8
derive_clocks	2-10
get_cells	2-11
get_clocks	2-13
get_nets	2-14
get_pins	2-15
get_ports	2-17
remove_clock_groups	2-18
remove_clock_latency	2-19
remove_clock_uncertainty	2-20
remove_disable_timing	2-21
remove_input_delay	2-22
remove_output_delay	2-23
reset_design	2-24
set_clock_groups	2-25
set_clock_latency	2-26
set_clock_uncertainty	2-28
set_disable_timing	2-29
set_false_path	2-30
set_input_delay	2-32
set_input_transition	2-34
set_max_delay	2-35
set_min_delay	2-37
set_multicycle_path	2-39
set_output_delay	2-41
sdc_ext	2-43
derive_clock_uncertainty	2-44
derive_pll_clocks	2-45
get_assignment_groups	2-46
get_fanins	2-47

get_fanouts	2-48
get_keepers	2-49
get_nodes	2-50
get_partitions	2-51
get_registers	2-52
remove_annotated_delay	2-53
remove_clock	2-54
reset_timing_derate	2-55
set_active_clocks	2-56
set_annotated_delay	2-57
set_max_skew	2-58
set_net_delay	2-60
set_scc_mode	2-61
set_time_format	2-62
set_timing_derate	2-63
sta	2-64
add_to_collection	2-67
check_timing	2-68
create_report_histogram	2-70
create_slack_histogram	2-72
create_timing_netlist	2-74
create_timing_summary	2-76
delete_timing_netlist	2-77
enable_ccpp_removal	2-78
enable_sdc_extension_collections	2-79
get_available_operating_conditions	2-80
get_cell_info	2-81
get_clock_domain_info	2-82
get_clock_fmax_info	2-83
get_clock_info	2-84
get_datasheet	2-86
get_default_sdc_file_names	2-88
get_edge_info	2-89
get_edge_slacks	2-90
get_min_pulse_width	2-91
get_net_info	2-92
get_node_info	2-93
get_object_info	2-94
get_operating_conditions	2-95
get_operating_conditions_info	2-96
get_partition_info	2-97
get_path	2-98
get_path_info	2-100
get_pin_info	2-103
get_point_info	2-104
get_port_info	2-107
get_register_info	2-108
get_timing_paths	2-109
locate	2-112
query_collection	2-114
read_sdc	2-115
remove_from_collection	2-116
report_advanced_io_timing	2-117
report_bottleneck	2-118

report_clock_fmax_summary	2-120
report_clock_transfers	2-121
report_clocks	2-122
report_datasheet	2-123
report_ddr	2-124
report_exceptions	2-125
report_max_skew	2-129
report_metastability	2-132
report_min_pulse_width	2-135
report_net_delay	2-137
report_net_timing	2-138
report_partitions	2-139
report_path	2-140
report_rskm	2-142
report_sdc	2-143
report_skew	2-144
report_tccs	2-147
report_timing	2-148
report_ucp	2-152
set_operating_conditions	2-153
timing_netlist_exist	2-154
update_timing_netlist	2-155
use_timequest_style_escaping	2-156
write_sdc	2-157

This manual provides comprehensive information about SDC and TimeQuest API operations.

Revision History

The following table shows the revision history for this manual.

Date and Document Version	Changes Made	Summary of Changes
December 2009, v5.0	<ul style="list-style-type: none"> Updated for Quartus II version 9.1 	—
March 2009, v4.0	<ul style="list-style-type: none"> Updated for Quartus II version 9.0 	—
November 2008, v3.1	<ul style="list-style-type: none"> Updated for Quartus II version 8.1 Added Revision History to this reference manual Updated new document template 	—
July 2008, v3.0	<ul style="list-style-type: none"> Updated for Quartus II version 8.0 	Updated the sdc and TimeQuest API operations based on the latest Quartus II software release.
December 2007, v2.0	<ul style="list-style-type: none"> Updated for Quartus II version 7.2 	Updated the sdc and TimeQuest API operations based on the latest Quartus II software release.
March 2007, v1.1	<ul style="list-style-type: none"> Updates to sdc, sdc_ext, and sta API operations. Added Revision History to this reference manual. 	Updated the sdc and TimeQuest API operations based on the latest Quartus II software release.
May 2006, v1.0	<ul style="list-style-type: none"> Initial release. 	—

How to Contact Altera

For the most up-to-date information about Altera® products, see the following table.






Contact <i>(Note 1)</i>	Contact Method	Address
Technical support	Website	www.altera.com/support
Technical training	Website	www.altera.com/training
	Email	custrain@altera.com
Product Literature	Website	www.altera.com/literature
Altera literature services	Email	literature@altera.com
Non-technical support (General) (Software Licensing)	Email	nacomp@altera.com
	Email	authorization@altera.com

Note:

(1) You can also contact your local Altera sales office or sales representative.

Typographic Conventions

The following table shows the typographic conventions that this document uses.

Visual Cue	Meaning
Bold Type with Initial Capital Letters	Command names, dialog box titles, checkbox options, and dialog box options are shown in bold, initial capital letters. Example: Save As dialog box.
bold type	External timing parameters, directory names, project names, disk drive names, file names, file name extensions, and software utility names are shown in bold type. Examples: f_{MAX} , \qdesigns directory, d: drive, chiptrip.gdf file.
<i>Italic Type with Initial Capital Letters</i>	Document titles are shown in italic type with initial capital letters. Example: <i>AN 75: High-Speed Board Design</i> .
<i>Italic type</i>	Internal timing parameters and variables are shown in italic type. Examples: <i>t_{PIA}</i> , <i>n + 1</i> . Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: < <i>file name</i> >, < <i>project name</i> >. .pof file.
Initial Capital Letters	Keyboard keys and menu names are shown with initial capital letters. Examples: Delete key, the Options menu.
“Subheading Title”	References to sections within a document and titles of on-line help topics are shown in quotation marks. Example: “Typographic Conventions.”
Courier type	Signal and port names are shown in lowercase Courier type. Examples: <code>data1</code> , <code>tdi</code> , <code>input</code> . Active-low signals are denoted by suffix <code>n</code> , e.g., <code>resetn</code> . Anything that must be typed exactly as it appears is shown in Courier type. For example: <code>c:\qdesigns\tutorial\chiptrip.gdf</code> . Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword <code>SUBDESIGN</code>), as well as logic function names (e.g., <code>TRI</code>) are shown in Courier.
1., 2., 3., and a., b., c., etc.	Numbered steps are used in a list of items when the sequence of the items is important, such as the steps listed in a procedure.
■ ■	Bullets are used in a list of items when the sequence of the items is not important.
✓	The checkmark indicates a procedure that consists of one step only.
	The hand points to information that requires special attention.
	A caution calls attention to a condition or possible situation that can damage or destroy the product or the user's work.
	A warning calls attention to a condition or possible situation that can cause injury to the user.
	The angled arrow indicates you should press the Enter key.
	The feet direct you to more information on a particular topic.

Introduction

The TimeQuest timing analyzer is a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in a design using industry standard constraint, analysis, and reporting methodology.

The TimeQuest timing analyzer includes support for Synopsis Design Constraints (SDC). Additionally, the TimeQuest timing analyzer includes an extensive tool command language (Tcl) scripting API. This reference manual includes the supported SDC constraints and commands, and the TimeQuest Tcl API commands.

This overview covers the TimeQuest timing analyzer support for scripted operation. At the end of this overview is a table of additional documentation and resources.

What's Inside the SDC and TimeQuest API Reference Manual?

The *Quartus II SDC and TimeQuest API Reference Manual* is your reference guide to TimeQuest timing analyzer constraints and commands, including command details, usage, and examples.

All the information included in the *Quartus II SDC and TimeQuest API Reference Manual*, as well as the most up-to-date list of commands, can also be found in the Quartus II software Tcl API and command-line executable online help reference, Qhelp. To access this information within Quartus II design software, type the following command at the command prompt:

```
quartus_sh --qhelp←
```

TimeQuest Timing Analyzer Scripting Support

The **sdc** and **sta** packages are supported in the **quartus_sta** command-line executable.

The **sdc** package contains the Synopsis Design Constraints (SDC) functions used to specify constraints and exceptions to the TimeQuest timing analyzer.

The **sta** package contains the set of Tcl functions for obtaining advanced information from the TimeQuest timing analyzer.

The **quartus_sta** command-line executable validates the timing performance of all logic in a design using industry standard constraint, analysis, and reporting methodology. You can use the TimeQuest analyzer's graphical user interface (GUI) or command-line interface to constrain, run, and view results for all timing paths in your design.



For information about other command-line executables and Tcl packages, refer to the *Quartus II Scripting Reference Manual*.

Related Documentation

Table 1 presents additional resources and documentation for the Quartus II development software.

Table 1. Quartus II Software Resources and Documentation

Resource Name	Resource Type	Access	Description	User Level
Quartus II Online Demonstration	Videos	www.altera.com/quartusdemos	Demonstration of the Quartus II software command-line operation and scripting features.	Beginning to Advanced
Introduction to Quartus II Manual	PDF	www.altera.com/literature/manual/intro_to_quartus2.pdf	An overview of the capabilities of Quartus II software in programmable logic design.	Beginning to Intermediate
Scripting and Constraint Entry section of the Quartus II Handbook	PDF	www.altera.com/literature/lit-qts.jsp	Detailed instruction for command-line operation and Tcl scripting in Quartus II software.	Beginning to Advanced
Qhelp	Quartus II software online help	Run <code>quartus_sh --qhelp</code> from the command line	Detailed listing of all command-line executables and Tcl commands including usage examples.	Beginning to Advanced
Enhance Your FPGA Design Flow With Command-Line and Tcl Scripting	Net Seminar (one hour)	www.altera.com/education/net_seminars/past/ns-tcl.html	Overview of Quartus II command-line operation and scripting support.	Beginning to Intermediate
Design Examples	html	www.altera.com/support/examples/quartus/quartus.html	Instructions for implementing various functions using Quartus II design software.	Beginning to Advanced
Online Training	PowerPoint (PPT) and audio	www.altera.com/training	Detailed instruction examples.	Beginning to Advanced

sdc

Synopsys Design Constraint (SDC) format is used to specify the design intent, including the timing and area constraints of the design. The TimeQuest Timing Analyzer only implements the set of SDC commands required to specify the timing constraints of the design. For area constraints, the QSF file should be used.

This package implements the SDC Spec Version 1.5 (June 2005).

Any command in this package can be specified in a TimeQuest SDC file.

This package is loaded by default in the following executable:

- quartus_sta

This package is available for loading in the following executable:

- quartus_map

This package includes the following commands:

Command	Page
all_clocks	2-3
all_inputs	2-4
all_outputs	2-5
all_registers	2-6
create_clock	2-7
create_generated_clock	2-8
derive_clocks	2-10
get_cells	2-11
get_clocks	2-13
get_nets	2-14
get_pins	2-15
get_ports	2-17
remove_clock_groups	2-18
remove_clock_latency	2-19
remove_clock_uncertainty	2-20
remove_disable_timing	2-21
remove_input_delay	2-22
remove_output_delay	2-23
reset_design	2-24
set_clock_groups	2-25
set_clock_latency	2-26
set_clock_uncertainty	2-28
set_disable_timing	2-29
set_false_path	2-30
set_input_delay	2-32
set_input_transition	2-34
set_max_delay	2-35
set_min_delay	2-37
set_multicycle_path	2-39
set_output_delay	2-41

all_clocks

Usage

all_clocks

Options

None

Description

Returns a collection of all clocks in the design.

Example

```
project_open chiptrip
create_timing_netlist
foreach_in_collection clk [all_clocks] {
    puts [get_clock_info -name $clk]
}
delete_timing_netlist
project_close
```

all_inputs

Usage

```
all_inputs
```

Options

None

Description

Returns a collection of all input ports in the design.

Example

```
project_open chiptrip
create_timing_netlist
foreach_in_collection in [all_inputs] {
    puts [get_port_info -name $in]
}
set_input_delay -clock clock1 2.0 [all_inputs]
delete_timing_netlist
project_close
```

all_outputs

Usage

all_outputs

Options

None

Description

Returns a collection of all output ports in the design.

Example

```
project_open chiptrip
create_timing_netlist
foreach_in_collection out [all_outputs] {
    puts [get_port_info -name $out]
}
set_output_delay -clock clock1 2.0 [all_outputs]
delete_timing_netlist
project_close
```

all_registers

Usage

```
all_registers
```

Options

None

Description

Returns a collection of all registers in the design.

Example

```
project_open chiptrip
create_timing_netlist
foreach_in_collection reg [all_registers] {
    puts [get_register_info -name $reg]
}
report_timing -from [all_registers] -to [all_registers]
delete_timing_netlist
project_close
```


create_clock

Usage

```
create_clock [-add] [-name <clock_name>] -period <value> [-waveform <edge_list>]
<targets>
```

Options

-add: Adds clock to a node with an existing clock

-name <clock_name>: Clock name of the created clock

-period <value>: Speed of the clock in terms of clock period

-waveform <edge_list>: List of edge values

<targets>: List or collection of targets

Description

Defines a clock. If the -name option is not used, the clock name is the same as the first target in the list or collection. The clock name is used to refer to the clock in other commands.

The -period option specifies the clock period. It is also possible to use this option to specify a frequency to define the clock period. This can be done by using -period option followed by either <frequency>MHz or "<frequency> MHz". However, this is a TimeQuest-only extension and makes the SDC syntax non-standard.

The -waveform option specifies the rising and falling edges (duty cycle) of the clock, and is specified as a list of two time values: the first rising edge and the next falling edge. The rising edge must be within the range [0, period]. The falling edge must be within one clock period of the rising edge. The waveform defaults to (0, period/2).

If a clock with the same name is already assigned to a given target, the create_clock command will overwrite the existing clock. If a clock with a different name exists on the given target, the create_clock command will be ignored unless the -add option is used. The -add option can be used to assign multiple clocks to a pin or port.

If the target of the clock is internal (i.e. not an input port), the source latency is zero by default.

If a clock is on a path after another clock, then it blocks or overwrites the previous clock from that point forward.

The value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See the help for use_timequest_style_escaping for details.

Example

```
# Create a simple 10ns with clock with a 60% duty cycle
create_clock -period 10 -waveform {0 6} -name clk [get_ports clk]

# Create a clock with a falling edge at 2ns, rising edge at 8ns,
# falling at 12ns, etc.
create_clock -period 10 -waveform {8 12} -name clk [get_ports clk]

# Assign two clocks to an input port that are switched externally
create_clock -period 10 -name clk100Mhz [get_ports clk]
create_clock -period 6.667 -name clk150Mhz -add [get_ports clk]

# Two ways to use MHz to define clock period (TimeQuest only)
create_clock -period 250MHz -name clk250MHz [get_ports clk]
create_clock -period "250 MHz" -name clk250MHz [get_ports clk]
```

create_generated_clock

Usage

```
create_generated_clock [-add] [-divide_by <factor>] [-duty_cycle <percent>]
[-edge_shift <shift_list>] [-edges <edge_list>] [-invert] [-master_clock <clock>]
[-multiply_by <factor>] [-name <clock_name>] [-offset <time>] [-phase <degrees>]
-source <clock_source> <targets>
```

Options

-add: Add clock to existing clock node

-divide_by <factor>: Division factor

-duty_cycle <percent>: Specifies the duty cycle as a percentage of the clock period--accepts floating point values

-edge_shift <shift_list>: List of edge shifts

-edges <edge_list>: List of edge values

-invert: Invert the clock waveform

-master_clock <clock>: Specifies clock of the source node

-multiply_by <factor>: Multiplication factor

-name <clock_name>: Name of generated clock

-offset <time>: Specifies the offset as an absolute time shift

-phase <degrees>: Specifies the phase shift in degrees

-source <clock_source>: Source node for the generated clock

<targets>: List or collection of targets

Description

Defines an internally generated clock. If -name is not specified, the clock name is the same as the first target in the list or collection. The clock name is used to refer to the clock in other commands.

If a clock with the same name is already assigned to a given target, the create_generated_clock command will overwrite the existing clock. If a clock with a different name exists on the given target, the create_generated_clock command will be ignored unless the -add option is used. The -add option can be used to assign multiple clocks to a pin or port, and is recommended be used with -master_clock option.

The source of the generated clock, specified by -source, is a port, pin, register, or net in the design. All waveform modifications are relative to this point. If more than one clock feeds the source node, the -master_clock option must be used to specify which clock to modify.

The source latency of the generated clock is based on the clock network of the generated clock, and not the clock network of the node specified using -source. This latency is added to any source latency of the master clock.

The -divide_by, -multiply_by, -invert, -duty_cycle, -edges, and -edge_shift options modify the waveform relative to the waveform at the source node.

Clock division and multiplication, using -divide_by and -multiply_by, is performed relative to the first rising edge. Clock division is based on edges in the master clock waveform, and scaled if the division is an odd number. Use the -duty_cycle option to specify the new duty cycle for clock multiplication. Use the -phase option to specify any phase shift relative to the new clock period. Use the -offset option to specify an arbitrary offset or time shift. Use the -invert option to invert the waveform.

Clock generation can also be specified with the `-edges` and `-edge_shift` options. The `-edges` option accepts a list of three numbers specifying the master clock edges to use for the first rising edge, the next falling edge, and next rising edge. Edges of the master clock are labeled according to the first rising edge (1), next falling edge (2), next rising edge (3), etc. For example, a basic clock divider can be specified equivalently with `-divide_by 2` or `-edges {1 3 5}`. The `-edge_shift` option accepts a list of three time values, the amount to shift each of the three edges.

The value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See the help for `use_timequest_style_escaping` for details.

Example

```
# Create a clock and a divide-by-2 generated clock
create_clock -period 10 [get_ports clk]
create_generated_clock -divide_by 2 -source [get_ports clk] -name clkdiv \
    [get_registers clkdiv]

# An equivalent generated clock
create_generated_clock -edges {1 3 5} -source [get_ports clk] -name \
    clkdiv [get_registers clkdiv]

# Specify a clock multiplier with a 60% duty cycle
create_generated_clock -multiply_by 2 -duty_cycle 60 \
    [get_pins clkmult|combout]

# Specify an inverted divide-by-2 clock relative to the output of the
# source clock
create_generated_clock -divide_by 2 -invert -source [get_ports clk] -name \
    nclkdiv [get_registers clkdiv]

# Specify a divide-by-2 clock with a 90-degree phase shift
create_generated_clock -divide_by 2 -phase 90 -source [get_ports clk] \
    -name clkdiv [get_registers clkdiv]

# Assign two clocks to an input port that are switched externally,
# along with an internal clock divider.
create_clock -period 10 -name clk100Mhz [get_ports clk]
create_clock -period 6.667 -name clk150Mhz -add [get_ports clk]
create_generated_clock -divide_by 2 -name clk50Mhz -source [get_ports \
    clk] -master_clock clk100Mhz -add [get_registers clkdiv]
create_generated_clock -divide_by 2 -name clk75Mhz -source [get_ports \
    clk] -master_clock clk150Mhz -add [get_registers clkdiv]
```

derive_clocks

Usage

```
derive_clocks -period <period_value> [-waveform <edge_list>]
```

Options

-period <period_value>: Speed of the default clock in terms of clock period

-waveform <edge_list>: List of edge values

Description

Creates a clock on sources of clock pins in the design that do not already have at least one clock sourcing the clock pin. This command is equivalent to calling `create_clock` on each clock source in the design that does not already have a clock assigned to it.

See the help for `create_clock` for more information.

Altera does not recommend using this command during final sign-off analysis of a design. `derive_clocks` should only be used early in the design phase when the clocks are not completely known. When possible, `create_clock` and `create_generated_clock` should be used instead.

Example

```
# Automatically create a 10ns, 60% duty cycle clock on all
# unconstrained clock sources
derive_clocks -period 10 -waveform {0 6}
```

get_cells

Usage

```
get_cells [-compatibility_mode] [-hierarchical] [-no_duplicates] [-nocase] [-nowarn]  
<filter>
```

Options

```
-compatibility_mode: Use simple Tcl matching (Classic Timing Analyzer style)  
-hierarchical: Specifies use of a hierarchical searching method  
-no_duplicates: Do not match duplicated cell names  
-nocase: Specifies case insensitive node name matching  
-nowarn: Do not issue warnings messages about unmatched patterns  
<filter>: Valid destinations (string patterns are matched using Tcl string matching)
```

Description

Returns a collection of cells in the design. All cell names in the collection match the specified pattern. Wildcards can be used to select multiple cells at once.

There are three Tcl string matching schemes available with this command: the default method, the -hierarchical option, and the -compatibility_mode option.

When you use the default matching scheme, use pipe characters to separate one hierarchy level from the next. They are treated as special characters and are taken into account when string matching with wildcards is performed. When this matching scheme is enabled, the specified pattern is matched against absolute cell names: the names that include the entire hierarchical path. A full cell name can contain multiple pipe characters in it to reflect the hierarchy. All hierarchy levels in the pattern are matched level by level. Any included wildcards refer to only one hierarchical level. For example, "*" and "*|*" produce different collections since they refer to the highest hierarchical level and second highest hierarchical level respectively.

When using the -hierarchical matching scheme, pipe characters are treated as special characters and are taken into account when string matching with wildcards is performed. This matching scheme forces the search to proceed recursively down the hierarchy. The specified pattern is matched against the relative cell names: the immediate names that do not include any of the hierarchy information. Note that a short cell name cannot contain pipe characters in it. Any included wildcards are expanded to match the relative pin names.

The -compatibility_mode matching scheme mimics the string matching behavior of the Classic Timing Analyzer. The simple Tcl string matching on full, absolute cell names is used. Pipe characters are not treated as special characters when used with wildcards.

The default matching scheme returns cells whose names match the specified filter and also cells automatically generated by the Quartus II software from these cells). Use -no_duplicates option to not include duplicated cells.

The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or TimeQuest-extension substitution rules. See help for the use_timequest_style_escaping command for details.

Example

```
# Find a cell called "reg" using case insensitive search  
get_cells -nocase reg  
# Create a collection of all cells whose names start with "reg"  
get_cells reg*  
# Create a collection of all cells on the highest hierararchical level
```

```
set mycollection [get_cells *]
# Create a collection of all cells in the design
# Output cell names.
foreach_in_collection cell $mycollection {
    puts [get_cell_info -name $cell]
}
set fullcollection [get_cells -hierarchical *]
# Output cell IDs and names.
foreach_in_collection cell $fullcollection {
    puts -nonewline $cell
    puts -nonewline ": "
    puts [get_cell_info -name $cell]
}
```

get_clocks

Usage

```
get_clocks [-nocase] [-nowarn] <filter>
```

Options

-nocase: Specifies the matching of node names to be case-insensitive

-nowarn: Do not issue warnings messages about unmatched patterns

<filter>: Valid destinations (string patterns are matched using Tcl string matching)

Description

Returns a collection of clocks in the design. When used as an argument to another command, such as the -from or -to options of set_multicycle_path, each node in the clock represents all nodes driven by the clocks in the collection.

```
# The following multicycle constraint applies to all paths ending at registers
```

```
# driven by clk
```

```
set_multicycle_path -to [get_clocks clk] 2
```

The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or TimeQuest-extension substitution rules. See the help for use_timequest_style_escaping for details.

Example

```
project_open chiptrip
create_timing_netlist
read_sdc
update_timing_netlist
set clocks [get_clocks c* -nocase]
foreach_in_collection clk $clocks {
    set name [get_clock_info -name $clk]
    set period [get_clock_info -period $clk]
    puts "$name: $period"
}
delete_timing_netlist
project_close
```

get_nets

Usage

```
get_nets [-no_duplicates] [-nocase] [-nowarn] <filter>
```

Options

-no_duplicates: Do not match duplicated net names
-nocase: Specifies case-insensitive node name matching
-nowarn: Do not issue warnings messages about unmatched patterns
<filter>: Valid destinations (string patterns are matched using Tcl string matching)

Description

Returns a collection of nets in the design. All net names in the collection match the specified pattern. Wildcards can be used to select multiple nets at once.

The default matching scheme returns nets whose names match the specified filter and nets that are automatically generated by the Quartus II software from these nets. Use the -no_duplicates option to exclude duplicated nets.

The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or TimeQuest-extension substitution rules. See help for the use_timequest_style_escaping command for details.

Example

```
# Find a net called "reg" using case insensitive search
get_nets -nocase reg
# Create a collection of all nets whose names start with "reg"
get_nets reg*
# Create a collection of all nets in the design
set mycollection [get_nets *]
# Output net names.
foreach_in_collection net $mycollection {
    puts [get_net_info -name $net]
}
```


get_pins

Usage

```
get_pins [-compatibility_mode] [-hierarchical] [-no_duplicates] [-nocase] [-nowarn]  
<filter>
```

Options

```
-compatibility_mode: Use simple Tcl matching (Classic Timing Analyzer style)  
-hierarchical: Specifies use of a hierarchical searching method  
-no_duplicates: Do not match duplicated pin names  
-nocase: Specifies case-insensitive node name matching  
-nowarn: Do not issue warnings messages about unmatched patterns  
<filter>: Valid destinations (string patterns are matched using Tcl string matching)
```

Description

Returns a collection of pins in the design. All pin names in the collection match the specified pattern. Wildcards can be used to select multiple pins at once.

There are three Tcl string matching schemes available with this command: the default method, the -hierarchical option, and the -compatibility_mode option.

By default, pipe characters are used to separate one hierarchy level from the next. They are treated as special characters and are taken into account when string matching with wildcards is performed. When the default matching scheme is enabled, the specified pattern is matched against absolute pin names: the names that include the entire hierarchical path. All hierarchy levels in the pattern are matched level by level. Pin names of the form <absolute full cell name> | <pin suffix> are used for matching. Note that a full cell name can contain multiple pipe characters in it to reflect the hierarchy. Any included wildcards refer to only one hierarchical level. For example, "*" | "*" and "*" | * | "*" produce different collections since they refer to the highest hierarchical level and second highest hierarchical level respectively.

When using the -hierarchical matching scheme, pipe characters are treated as special characters and are taken into account when string matching with wildcards is performed. This matching scheme forces the search to proceed recursively through the hierarchy. The specified pattern is matched against the relative pin names: the immediate names that do not include any of the hierarchy information. Pin names of the form <relative short cell name> | <pin suffix> are used for matching. Note that a short cell name cannot contain pipe characters. Any included wildcards are expanded to match the relative pin names. For example, "*" and "*" | "*" match exactly the same pins since the former is expanded into the latter.

The -compatibility_mode matching scheme mimics the string matching behavior of the Classic timing analyzer for full, absolute pin names. Pipe characters are not treated as special characters when used with wildcards.

The default matching scheme returns not only pins whose names match the specified filter, but also pins duplicated from these pins (refers to pins are automatically generated by Quartus from the pins). Use -no_duplicates option to not include duplicated pins.

The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or TimeQuest-extension substitution rules. See help for the use_timequest_style_escaping command for details.

Example

```
# Get regout pin of "reg" cell  
get_pins -nocase reg|regout  
# Create a collection of all pins of "reg" cell
```

```
get_pins reg|*
# Create a collection of all pins on the highest hierararchical level
set mycollection [get_pins *]
# Output pin names.
foreach_in_collection pin $mycollection {
    puts [get_pin_info -name $pin]
}
# Create a collection of all pins in the design
set fullcollection [get_pins -hierarchical *]
# Output pin IDs and names.
foreach_in_collection pin $fullcollection {
    puts -nonewline $pin
    puts -nonewline ": "
    puts [get_pin_info -name $pin]
}
```

get_ports

Usage

```
get_ports [-nocase] [-nowarn] <filter>
```

Options

-nocase: Specifies case-insensitive node name matching

-nowarn: Do not issue warnings messages about unmatched patterns

<filter>: Valid destinations (string patterns are matched using Tcl string matching)

Description

Returns a collection of ports (design inputs and outputs) in the design.

The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or TimeQuest-extension substitution rules. See help for the use_timequest_style_escaping command for details.

Example

```
project_open chiptrip
create_timing_netlist

# Get all ports starting with "In".
set ports [get_ports In*]
foreach_in_collection port $ports {
    puts [get_port_info -name $port]
}

delete_timing_netlist
project_close
```

remove_clock_groups

Usage

```
remove_clock_groups -all
```

Options

-all: Specify remove all clock group settings

Description

Remove all clock group assignments. This command removes any clock groups that have been previously set. There is no way to remove specific groups.

Example

```
project_open top
create_timing_netlist
create_clock -period 10.000 -name clkA [get_ports sysclk[0]]
create_clock -period 10.000 -name clkB [get_ports sysclk[1]]

# Set clkA and clkB to be mutually exclusive clocks.
set_clock_groups -exclusive -group {clkA} -group {clkB}
set_clock_groups -exclusive -group {clkC} -group {clkD}

# Remove clock groups A, B, C, and D. Result is that there
# are no longer any mutually exclusive clocks.
remove_clock_groups -all
```

remove_clock_latency

Usage

```
remove_clock_latency -source <targets>
```

Options

-source: Specifies the source clock latency

<targets>: Valid destinations (string patterns are matched using Tcl string matching)

Description

Removes clock latency for a given clock or clock target.

There are two types of latency: network and source. Network latency is the clock network delay between the clock and register clock pins. Source latency is the clock network delay between the clock and its source (e.g., a system clock or a base clock of a generated clock).

The TimeQuest Timing Analyzer automatically computes network latencies for all register and generated clocks. Overriding clock network latencies is not supported by the TimeQuest analyzer. Therefore, the -source option must always be specified. Remove_clock_latency requires this option as well.

You can apply clock latency to a clock, which affects all targets of the clock, or to a specific clock target. Therefore, you can remove clock latency from a collection of clocks, or from a collection of target nodes. remove_clock_latency removes all latencies from a clock or node, so removing a node's clock latency with respect to a particular clock, or removing only latencies with particular conditions is not supported.

The value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See the help for use_timequest_style_escaping for details.

Example

```
create_clock -name SYSCLK -period 10.000 [get_ports inclk]
create_generated_clock -name OUTCLK -divide_by 1 -source [get_ports \
    inclk] [get_ports outclk]
create_generated_clock -name FDBKCLK -divide_by 1 -source \
    [get_ports outclk] [get_ports fdbkclk]

# Apply a simple 2.000 ns source latency to the system clock.
set_clock_latency -source 2.000 [get_clocks SYSCLK]

# Specify feedback clock latencies between output port outclk
# and the output port fdbkclk.
set_clock_latency -source -late -rise 0.800 [get_clocks FDBKCLK]
set_clock_latency -source -late -fall 0.750 [get_clocks FDBKCLK]
set_clock_latency -source -early -rise 0.500 [get_clocks FDBKCLK]
set_clock_latency -source -early -fall 0.460 [get_clocks FDBKCLK]

# Remove all clock latency from FDBKCLK
remove_clock_latency -source [get_clocks FDBKCLK]
```

remove_clock_uncertainty

Usage

```
remove_clock_uncertainty -from <from_clock> -to <to_clock>
```

Options

-from <from_clock>: Valid destinations (string patterns are matched using Tcl string matching)

-to <to_clock>: Valid destinations (string patterns are matched using Tcl string matching)

Description

Removes clock uncertainty from a collection of clocks to a collection of clocks. The source and destination clocks can be any arbitrary collection of clocks. This command removes all uncertainty between two clocks. If there does not exist uncertainty between two clocks specified in `remove_clock_uncertainty`, the command does nothing for those two clocks but continues to attempt to remove uncertainty between other clocks specified.

The values of the `-from` and `-to` options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See the help for `use_timequest_style_escaping` for details.

Example

```
set_clock_uncertainty -setup -rise_from {clk1 clk2} -fall_to {clk3 clk4} \  
    200ps  
set_clock_uncertainty -from {clk5 clk6} -to {clk7 clk8} 300ps  
remove_clock_uncertainty -from {clk3 clk5} -to {clk4 clk7}
```

remove_disable_timing

Usage

```
remove_disable_timing [-from <name>] [-to <name>] <cells>
```

Options

```
-from <name>: Valid source pin suffix  
-to <name>: Valid destination pin suffix  
<cells>: List of cells
```

Description

Adds a previously disabled edge (arc) back to a given cell(s). If no -from/-to value is specified, the missing value is substituted by a "*".

The values of the -from and -to are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See the help for use_timequest_style_escaping for details.

Example

```
remove_disable_timing -from datain -to combout A|B  
remove_disable_timing -from carryin *
```

remove_input_delay

Usage

```
remove_input_delay <targets>
```

Options

<targets>: Collection or list of input ports

Description

Removes input delay from a port. For each input port specified, removes all input delays for that port. This means that rise, fall, max, and min delays for each clock and reference pin on the input port are all removed.

The value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See help for the `use_timequest_style_escaping` command for details.

Example

```
# Simple input delay with the same value for min/max and rise/fall
set_input_delay -clock clk 1.5 [get_ports {in1 in2}]
set_input_delay -clock clk2 1.5 [get_ports {in1 in2}]
set_input_delay -clock clk 1.6 [get_ports {in3 in4}]

# Remove input delay on ports in1 and in4,
# for all flags and reference ports and flags
remove_input_delay [get_ports {in1 in4}]
```


remove_output_delay

Usage

```
remove_output_delay <targets>
```

Options

<targets>: Collection or list of output ports

Description

Removes output delay from a port. For each output port specified, removes all output delays for that port. Rise, fall, max, and min delays for each clock and reference pin on the output port are all removed.

The value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See help for the use_timequest_style_escaping command for details.

Example

```
# Simple output delay with the same value for min/max and rise/fall
set_output_delay -clock clk 1.5 [get_ports {out1 out2}]
set_output_delay -clock clk2 1.5 [get_ports {out1 out2}]
set_output_delay -clock clk 1.6 [get_ports {out3 out4}]

# Remove input delay on ports out1 and out4,
# for all flags and reference ports and flags
remove_output_delay [get_ports {out1 out4}]
```

reset_design

Usage

```
reset_design
```

Options

None

Description

Removes all assignments from the design. This includes clocks, generated clocks, derived clocks, input delays, output delays, clock latency, clock uncertainty, clock groups, false paths, multicycle paths, min delays, and max delays. After `reset_design` is called, the design should be in the same state as it would be if `create_timing_netlist` was just called.

Example

```
# Constrain design
create_clock -name clk -period 4.000 -waveform { 0.000 2.000 } \
  [get_ports clk]
set_input_delay -clock clk2 1.5 [get_ports in*]
set_output_delay -clock clk 1.6 [get_ports out*]
set_false_path -from [get_keepers in] -through [get_nets r1] -to \
  [get_keepers out]

# Reset the design to the state that it was in before any constraints
# were entered
reset_design
```

set_clock_groups

Usage

```
set_clock_groups [-asynchronous] [-exclusive] -group <names>
```

Options

-asynchronous: Specify mutually exclusive clocks (same as the -exclusive option). Exists for compatibility.

-exclusive: Specify mutually exclusive clocks

-group <names>: Valid destinations (string patterns are matched using Tcl string matching)

Description

Clock groups provide a quick and convenient way to specify which clocks are not related. Asynchronous clocks are those that are completely unrelated (e.g., have different ideal clock sources). Exclusive clocks are those that are not active at the same time (e.g., multiplexed clocks). TimeQuest treats both options, "-exclusive" and "-asynchronous", as if they were the same.

The result of `set_clock_groups` is that all clocks in any group are cut from all clocks in every other group. This command is equivalent to calling `set_false_path` from each clock in every group to each clock in every other group and vice versa, making `set_clock_groups` easier to specify for cutting clock domains. The use of a single -group option tells TimeQuest to cut this group of clocks from all other clocks in the design, including clocks that are created in the future.

Example

```
project_open top
create_timing_netlist
create_clock -period 10.000 -name clkA [get_ports sysclk[0]]
create_clock -period 10.000 -name clkB [get_ports sysclk[1]]

# Set clkA and clkB to be mutually exclusive clocks.
set_clock_groups -exclusive -group {clkA} -group {clkB}

# The previous line is equivalent to the following two commands.
set_false_path -from [get_clocks clkA] -to [get_clocks clkB]
set_false_path -from [get_clocks clkB] -to [get_clocks clkA]
```

set_clock_latency

Usage

```
set_clock_latency [-clock <clock_list>] [-early] [-fall] [-late] [-rise] -source
<delay> <targets>
```

Options

-clock <clock_list>: Valid clock destinations (string patterns are matched using Tcl string matching)

-early: Specifies the early clock latency

-fall: Specifies the falling transition clock latency

-late: Specifies the late clock latency

-rise: Specifies the rising transition clock latency

-source: Specifies the source clock latency

<delay>: Latency delay value

<targets>: Valid destinations (string patterns are matched using Tcl string matching)

Description

Specifies clock latency for a given clock or clock target.

There are two types of latency: network and source. Network latency is the clock network delay between the clock and register clock pins. Source latency is the clock network delay between the clock and its source (e.g., the system clock or base clock of a generated clock).

The TimeQuest Timing Analyzer automatically computes network latencies for all register and generated clocks. Overriding clock network latencies is not supported by the TimeQuest analyzer. Therefore, the -source option must always be specified.

You can apply clock latency to a clock, which affects all targets of the clock, or to a specific clock target. If you specify a specific clock target that is driven by more than one clock, use the -clock option to specify which clock to use. Latencies assigned to a clock target override any latencies assigned to a clock.

Different clock latencies can be specified for early (-early) and late (-late) latencies, as well as for rising edges (-rise) and falling edges (-fall). If only some combinations are specified, the other combinations are used by default. For example, if only a -rise -early latency and a -fall -early latency are specified, then the -rise -late latency is assumed to be the same as the -rise -early latency and the -fall -late latency is assumed to be the same as the -fall -early latency. If neither -rise nor -fall are used or neither -early nor -fall are used, then the latency applies to both conditions.

Source latency can also be assigned to generated clocks. This may be useful for specifying board level delays from a clock output port to a clock input port when the clock input port is acting as a feedback clock.

The value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See help for the use_timequest_style_escaping command for details.

Example

```
create_clock -name SYSCLK -period 10.000 [get_ports inclk]
create_generated_clock -name OUTCLK -divide_by 1 -source [get_ports \
    inclk] [get_ports outclk]
create_generated_clock -name FDBKCLK -divide_by 1 -source \
    [get_ports outclk] [get_ports fdbkclk]
```

```
# Apply a simple 2.000 ns source latency to the system clock.
set_clock_latency -source 2.000 [get_clocks SYSCLK]

# Specify feedback clock latencies between output port outclk
# and the output port fdbkclk.
set_clock_latency -source -late -rise 0.800 [get_clocks FDBKCLK]
set_clock_latency -source -late -fall 0.750 [get_clocks FDBKCLK]
set_clock_latency -source -early -rise 0.500 [get_clocks FDBKCLK]
set_clock_latency -source -early -fall 0.460 [get_clocks FDBKCLK]
```

set_clock_uncertainty

Usage

```
set_clock_uncertainty [-add] [-fall_from <fall_from_clock>] [-fall_to <fall_to_clock>]
[-from <from_clock>] [-hold] [-rise_from <rise_from_clock>] [-rise_to <rise_to_clock>]
[-setup] [-to <to_clock>] <uncertainty>
```

Options

-add: Specifies that this assignment is an addition to the clock uncertainty derived by derive_clock_uncertainty call

-fall_from <fall_from_clock>: Valid destinations (string patterns are matched using Tcl string matching)

-fall_to <fall_to_clock>: Valid destinations (string patterns are matched using Tcl string matching)

-from <from_clock>: Valid destinations (string patterns are matched using Tcl string matching)

-hold: Specifies the uncertainty value (applies to clock hold or removal checks)

-rise_from <rise_from_clock>: Valid destinations (string patterns are matched using Tcl string matching)

-rise_to <rise_to_clock>: Valid destinations (string patterns are matched using Tcl string matching)

-setup: Specifies the uncertainty value (applies to clock setup or recovery checks) (default)

-to <to_clock>: Valid destinations (string patterns are matched using Tcl string matching)

<uncertainty>: Uncertainty

Description

Specifies clock uncertainty or skew for clocks or clock-to-clock transfers. You can specify uncertainty separately for setup and hold, and can specify separate rising and falling clock transitions. The setup uncertainty is subtracted from the data required time for each applicable path, and the hold uncertainty is added to the data required time for each applicable path.

The values for the -from, -to, and similar options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See the help for use_timequest_style_escaping for details.

When -add option is used, clock uncertainty assignment is treated as an addition to the value calculated by derive_clock_uncertainty command for a particular clock transfer. Note that when -add option is not used and derive_clock_uncertainty is called, user specified clock uncertainty assignment will take priority.

When derive_clock_uncertainty command is not used, specifying -add option to set_clock_uncertainty command will not have any effect.

Example

```
set_clock_uncertainty -setup -rise_from clk1 -fall_to clk2 200ps
```

set_disable_timing

Usage

```
set_disable_timing [-from <name>] [-to <name>] <cells>
```

Options

-from <name>: Valid source pin suffix
-to <name>: Valid destination pin suffix
<cells>: List of cells

Description

Disables a timing edge (arc) from inside a given cell or cells. Disabling a timing edge prevents timing analysis through that edge. If either -from or -to (or both) are unspecified, the missing value or values are replaced by a "*" character.

The values of the -from and -to are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See the help for use_timequest_style_escaping for details.

Example

```
set_disable_timing -from datain -to combout A|B  
set_disable_timing -from carryin *
```

set_false_path

Usage

```
set_false_path [-fall_from <names>] [-fall_to <names>] [-from <names>] [-hold]
[-rise_from <names>] [-rise_to <names>] [-setup] [-through <names>] [-to <names>]
```

Options

-fall_from <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-fall_to <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-from <names>: Valid sources (string patterns are matched using Tcl string matching)

-hold: Specifies the false_path value (applies only to clock hold or removal checks)

-rise_from <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-rise_to <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-setup: Specifies the false_path value (applies only to clock setup or recovery checks)

-through <names>: Valid through nodes (string patterns are matched using Tcl string matching)

-to <names>: Valid destinations (string patterns are matched using Tcl string matching)

Description

Specifies a false-path exception, removing (or cutting) paths from timing analysis.

The -from and -to values are collections of clocks, registers, ports, pins, or cells in the design. If the -from or -to values are not specified, the collection is converted automatically into [get_keepers *]. It is worth noting that if the counterpart of the unspecified collection is a clock collection, it is more efficient to explicitly specify this collection as a clock collection only if the clock collection also generates the desired assignment.

Applying exceptions between clocks applies the exception from all register or ports driven by the -from clock to all registers or ports driven by the -to clock. Applying exceptions between a pair of clocks is more efficient than for specific node to node or node to clock paths.

If pin names or collections are used, the -from value must be a clock pin and the -to value must be any non-clock input pin to a register. Assignments from clock pins or to and from cells applies to all registers in the cell or driven by the clock pin.

The -through values are collections of pins or nets in the design. An exception applied through a node in the design applies only to paths through the specified node.

The -rise_from and -fall_from options can be used in place of the -from destination nodes. The rise or fall value of the option indicates that the "from" nodes are driven by the rising or falling edge of the clock that feeds this node, taking into consideration any logical inversions along the clock path. The -from option is the combination of both rising and falling "from" nodes. If the "from" collection is a clock collection, the assignment applies to those nodes that are driven by the respective rising or falling clock edge.

The -rise_to and -fall_to options behave similarly to the "from" options described previously. These assignments restrict the given assignment to only those nodes or clocks that correspond to the specified rise or fall value, taking into consideration any logical inversions that are along the clock path.

The `-setup` and `-hold` options allow the false path to only be applied to the corresponding setup/recovery or hold/removal checks. The default if neither value is specified is to apply the false path to both `-setup` and `-hold`.

The values of the `-from`, `-to`, `-through`, and other similar options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See help for the `use_timequest_style_escaping` command for details.

See help for the `set_clock_groups` command for information.

Example

```
# Set a false-path between two unrelated clocks
# See also set_clock_groups
set_false_path -from [get_clocks clkA] -to [get_clocks clkB]

# Set a false-path for a specific path
set_false_path -from [get_pins regA|clk] -to [get_pins regB|aclr]

# Set a false-path from a node to a falling clock
set_false_path -from [get_pins regA|clk] -fall_to [get_clocks clkB]
```

set_input_delay

Usage

```
set_input_delay [-add_delay] -clock <name> [-clock_fall] [-fall] [-max] [-min]
[-reference_pin <name>] [-rise] [-source_latency_included] <delay> <targets>
```

Options

-add_delay: Add to existing delays instead of overriding them

-clock <name>: Clock name

-clock_fall: Specifies that input delay is relative to the falling edge of the clock

-fall: Specifies the falling input delay at the port

-max: Applies value as maximum data arrival time

-min: Applies value as minimum data arrival time

-reference_pin <name>: Specifies a port in the design to which the input delay is relative

-rise: Specifies the rising input delay at the port

-source_latency_included: Specifies that input delay includes added source latency

<delay>: Time value

<targets>: List of input port type objects

Description

Specifies the data arrival times at the specified input ports relative the clock specified by the -clock option. The clock must refer to a clock name in the design.

Input delays can be specified relative to the rising edge (default) or falling edge (-clock_fall) of the clock.

If the input delay is specified relative to a simple generated clock (a generated clock with a single target), the clock arrival times to the generated clock are added to the data arrival time.

Input delays can be specified relative to a port (-reference_pin) in the clock network. Clock arrival times to the reference port are added to data arrival times. Non-port reference pins are not supported.

Input delays can already include clock source latency. By default the clock source latency of the related clock is added to the input delay value, but when the -source_latency_included option is specified, the clock source latency is not added because it was factored into the input delay value.

The maximum input delay (-max) is used for clock setup checks or recovery checks and the minimum input delay (-min) is used for clock hold checks or removal checks. If only -min or -max (or neither) is specified for a given port, the same value is used for both.

Separate rising (-rise) and falling (-fall) arrival times at the port can be specified. If only one of -rise and -fall are specified for a given port, the same value is used for both.

By default, set_input_delay removes any other input delays to the port except for those with the same -clock, -clock_fall, and -reference_pin combination. Multiple input delays relative to different clocks, clock edges, or reference pins can be specified using the -add_delay option.

The value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See help for the use_timequest_style_escaping command for details.

Example

```
# Simple input delay with the same value for min/max and rise/fall:
# 1) set on ports with names of the form myin*
set_input_delay -clock clk 1.5 [get_ports myin*]
# 2) set on all input ports
set_input_delay -clock clk 1.5 [all_inputs]

# Input delay with respect to the falling edge of clock
set_input_delay -clock clk -clock_fall 1.5 [get_ports myin*]

# Input delays for different min/max and rise/fall combinations
set_input_delay -clock clk -max -rise 1.4 [get_ports myin*]
set_input_delay -clock clk -max -fall 1.5 [get_ports myin*]
set_input_delay -clock clk -min -rise 0.7 [get_ports myin*]
set_input_delay -clock clk -min -fall 0.8 [get_ports myin*]

# Adding multiple input delays with respect to more than one clock
set_input_delay -clock clkA -min 1.2 [get_ports myin*]
set_input_delay -clock clkA -max 1.8 [get_ports myin*]
set_input_delay -clock clkA -clock_fall 1.6 [get_ports myin*] -add_delay
set_input_delay -clock clkB -min 2.1 [get_ports myin*] -add_delay
set_input_delay -clock clkB -max 2.5 [get_ports myin*] -add_delay

# Specifying an input delay relative to an external clock output port
set_input_delay -clock clk -reference_pin [get_ports clkout] 0.8 \
    [get_ports myin*]
```

set_input_transition

Usage

```
set_input_transition [-clock <name>] [-clock_fall] [-fall] [-max] [-min] [-rise]  
<transition> <ports>
```

Options

-clock <name>: Clock name

-clock_fall: Specifies that input delay is relative to the falling edge of the clock

-fall: Specifies the falling output delay at the port

-max: Applies value as maximum data required time

-min: Applies value as minimum data required time

-rise: Specifies the rising output delay at the port

<transition>: Time value

<ports>: Collection or list of input or bidir ports

Description

This constraint does not affect calculations performed by TimeQuest. It only affects PrimeTime analysis or HardCopy II devices. If you set this constraint in TimeQuest the constraint is written out to the SDC file when you call write_sdc.

set_max_delay

Usage

```
set_max_delay [-fall_from <names>] [-fall_to <names>] [-from <names>] [-rise_from  
<names>] [-rise_to <names>] [-through <names>] [-to <names>] <value>
```

Options

-fall_from <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-fall_to <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-from <names>: Valid sources (string patterns are matched using Tcl string matching)

-rise_from <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-rise_to <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-through <names>: Valid through nodes (string patterns are matched using Tcl string matching)

-to <names>: Valid destinations (string patterns are matched using Tcl string matching)

<value>: Time Value

Description

Specifies a maximum delay exception for a given path.

The maximum delay is similar to changing the setup relationship (latching clock edge - launching clock edge), except that it can be applied to input or output ports without input or output delays assigned to them. Maximum delays are always relative to any clock network delays (if the source or destination is a register) or any input or output delays (if the source or destination is a port). Therefore, input delays and clock latencies are added to the data arrival times. Clock latencies also added to data required times and output delays are subtracted from data required times.

The -from and -to values are collections of clocks, registers, ports, pins, or cells in the design. If the -from or -to values are not specified, the collection is converted automatically into [get_keepers *]. It is worth noting that if the counterpart to the unspecified collection is a clock collection, it is more efficient to explicitly specify this collection as a clock collection but only if the clock collection also generates the desired assignment.

Applying exceptions between clocks applies the exception from all register or ports driven by the -from clock to all registers or ports driven by the -to clock. Applying exceptions between a pair of clocks is more efficient than for specific node to node or node to clock paths.

If pin names or collections are used, the -from value must be a clock pin and the -to value must be any non-clock input pin to a register. Assignments from clock pins or to and from cells applies to all registers in the cell or driven by the clock pin.

The -through values are collections of pins or nets in the design. An exception applied through a node in the design applies only to paths through the specified node.

The -rise_from and -fall_from options can be used in place of the -from destination nodes. The rise or fall value of the option indicates that the "from" nodes are driven by the rising or falling edge of the clock that feeds this node taking into consideration any logical inversions along the clock path. The "-from" option is the combination of both rising and falling "from" nodes. If the "from" collection is a clock collection, the assignment applies to those nodes that are driven by the respective rising or falling clock edge.

The `-rise_to` and `-fall_to` options behave similarly to the "from1" options described previously. These assignments restrict the given assignment to only those nodes or clocks that correspond to the specified rise or fall value taking into consideration any logical inversions that are along the clock path.

The values of the `-from`, `-to`, `-through`, and other similar options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See help for the `use_timequest_style_escaping` command for details.

Example

```
# Apply a 10ns max delay between two unrelated clocks
set_max_delay -from [get_clocks clkA] -to [get_clocks clkB] 10.000

# Apply a 2ns max delay for an input port (TSU)
set_max_delay -from [get_ports in[*]] -to [get_registers *] 2.000

# Apply a 2ns max delay for an output port (TCO)
set_max_delay -from [get_registers *] -to [get_ports out[*]] 2.000

# Apply a 2ns max delay for an input port to an output port (TPD)
set_max_delay -from [get_ports in[*]] -to [get_ports out[*]] 2.000

# Apply a 2ns max delay for an input port only to nodes driven by
# the rising edge of clock CLK
set_max_delay -from [get_ports in[*]] -rise_to [get_clocks CLK] 2.000
```

set_min_delay

Usage

```
set_min_delay [-fall_from <names>] [-fall_to <names>] [-from <names>] [-rise_from  
<names>] [-rise_to <names>] [-through <names>] [-to <names>] <value>
```

Options

-fall_from <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-fall_to <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-from <names>: Valid sources (string patterns are matched using Tcl string matching)

-rise_from <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-rise_to <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-through <names>: Valid through nodes (string patterns are matched using Tcl string matching)

-to <names>: Valid destinations (string patterns are matched using Tcl string matching)

<value>: Time Value

Description

Specifies a minimum delay exception for a given path.

The minimum delay is similar to changing the hold relationship (launching clock edge - latching clock edge), except that it can be applied to input or output ports without input or output delays assigned to them. Minimum delays are always relative to any clock network delays (if the source or destination is register) or any input or output delays (if the source or destination is a port). Therefore, input delays and clock latencies are added to the data arrival times. Clock latencies also added to data required times and output delays are subtracted from data required times.

The -from and -to values are collections of clocks, registers, ports, pins, or cells in the design. If the -from or -to values are not specified, the collection is converted automatically into [get_keepers *]. It is worth noting that if the counterpart of the unspecified collection is a clock collection, it is more efficient to explicitly specify this collection as a clock collection, but only if the clock collection also generates the desired assignment.

Applying exceptions between clocks applies the exception from all register or ports driven by the -from clock to all registers or ports driven by the -to clock. Also, applying exceptions between a pair of clocks is more efficient than for specific node to node or node to clock paths.

If pin names or collections are used, the -from value must be a clock pin and the -to value must be any non-clock input pin to a register. Assignments from clock pins or to and from cells applies to all registers in the cell or driven by the clock pin.

The -through values are collections of pins or nets in the design. An exception applied through a node in the design applies only to paths through the specified node.

The `-rise_from` and `-fall_from` options can be used in place of the destination nodes specified using the `-from` option. The rise or fall value of the option indicates that the "from" nodes are driven by the rising or falling edge of the clock that feeds this node taking into consideration any logical inversions along the clock path. The `-from` option is the combination of both rising and falling "from" nodes. If the `-from` collection is a clock collection, the assignment applies to those nodes that are driven by the respective rising or falling clock edge.

The `-rise_to` and `-fall_to` options behave similarly to the "from" options described previously. These assignments restrict the given assignment to only those nodes or clocks that correspond to the specified rise or fall value taking into consideration any logical inversions that are along the clock path.

The values of the `-from`, `-to`, `-through`, and other similar options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See help for the `use_timequest_style_escaping` command for details.

Example

```
# Apply a 0ns min delay between two unrelated clocks
set_min_delay -from [get_clocks clkA] -to [get_clocks clkB] 0.000

# Apply a 0ns min delay for an input port (TH)
set_min_delay -from [get_ports in[*]] -to [get_registers *] -.000

# Apply a 0.5ns min delay for an output port (MIN_TCO)
set_min_delay -from [get_registers *] -to [get_ports out[*]] 0.500

# Apply a 0.5ns min delay for an input port to an output port (MIN_TPD)
set_min_delay -from [get_ports in[*]] -to [get_ports out[*]] 0.500

# Apply a 0.5ns min delay for an input port only to nodes driven by
# the falling edge of clock CLK
set_max_delay -from [get_ports in[*]] -fall_to [get_clocks CLK] 0.500
```


set_multicycle_path

Usage

```
set_multicycle_path [-end] [-fall_from <names>] [-fall_to <names>] [-from <names>]
[-hold] [-rise_from <names>] [-rise_to <names>] [-setup] [-start] [-through <names>]
[-to <names>] <value>
```

Options

-end: Specifies that the multicycle is relative to the destination clock waveform (default)

-fall_from <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-fall_to <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-from <names>: Valid sources (string patterns are matched using Tcl string matching)

-hold: Specifies that the multicycle value applies to clock hold or removal checks

-rise_from <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-rise_to <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-setup: Specifies that the multicycle value applies to clock setup or recovery checks (default)

-start: Specifies that the multicycle is relative to the source clock waveform

-through <names>: Valid through nodes (string patterns are matched using Tcl string matching)

-to <names>: Valid destinations (string patterns are matched using Tcl string matching)

<value>: Number of clock cycles

Description

Specifies a multicycle exception for a given set of paths.

Multicycles can be specified relative to the source clock (-start) or destination clock (-end). This is useful when the source clock and destination clock are operating at different frequencies. For example, if the source clock is twice as fast (half period) as the destination clock, a -start multicycle of 2 is usually required.

Hold multicycles (-hold) are computed relative to setup multicycles (-setup). The value of the hold multicycle represents the number clock edges away from the default hold multicycle. The default hold multicycle value is 0.

The -from and -to values are collections of clocks, registers, ports, pins, or cells in the design. If the -from or -to values are not specified, the collection is converted automatically into [get_keepers *]. It is worth noting that if the counterpart of the unspecified collection is a clock collection, it is more efficient to explicitly specify this collection as a clock collection but only if the clock collection also generates the desired assignment.

Applying exceptions between clocks applies the exception from all register or ports driven by the -from clock to all registers or ports driven by the -to clock. Also, applying exceptions between a pair of clocks is more efficient than for specific node to node or node to clock paths.

If pin names or collections are used, the `-from` value must be a clock pin and the `-to` value must be any non-clock input pin to a register. Assignments from clock pins or to and from cells applies to all registers in the cell or driven by the clock pin.

The `-through` values are collections of pins or nets in the design. An exception applied through a node in the design applies only to paths through the specified node.

The `-rise_from` and `-fall_from` options can be used in place of the `"-from"` destination nodes. The rise or fall value of the option indicates that the `"from"` nodes are driven by the rising or falling edge of the clock that feeds this node taking into consideration any logical inversions along the clock path. The `"-from"` option is the combination of both rising and falling `"from"` nodes. If the `"from"` collection is a clock collection, the assignment applies to those nodes that are driven by the respective rising or falling clock edge.

The `-rise_to` and `-fall_to` options behave similarly to the `"from"` options described previously. These assignments restrict the given assignment to only those nodes or clocks that correspond to the specified rise or fall value taking into consideration any logical inversions that are along the clock path.

The values of the `-from`, `-to`, `-through`, and similar options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See help for the `use_timequest_style_escaping` command for details.

Example

```
create_clock -period 10.000 -name CLK [get_ports clk]
create_generated_clock -divide_by 2 -source [get_ports clk] -name CLKDIV2 \
    [get_registers clkdiv]

# Apply a source multicycle of 2 with a hold multicycle of 1 for all
# paths from the CLK domain to the CLKDIV2 domain.
set_multicycle_path -start -setup -from [get_clocks CLK] -to \
    [get_clocks CLKDIV2] 2
set_multicycle_path -start -hold -from [get_clocks CLK] -to \
    [get_clocks CLKDIV2] 1

# Apply a multicycle constraint of 3 (with a default hold multicycle of
# 0) for a
# specific path in the design.
set_multicycle_path -end -setup -from [get_pins rega|clk] -to \
    [get_pins regb|*] 3

# Apply a multicycle constraint of 2 to a given cell, except for the
# reset pin.
set_multicycle_path -end -setup -to [get_cells regb] 2
set_multicycle_path -end -setup -to [get_pins regb|aclr] 1

#Apply a multicycle constraint of 3 rising from a clock and falling to a
# node
set_multicycle_path -end -setup -rise_from [get_clocks CLK] -fall_to \
    [get_pins regb|datab] 3
```

set_output_delay

Usage

```
set_output_delay [-add_delay] -clock <name> [-clock_fall] [-fall] [-max] [-min]  
[-reference_pin <name>] [-rise] [-source_latency_included] <delay> <targets>
```

Options

-add_delay: Add to existing delays instead of overriding them

-clock <name>: Clock name

-clock_fall: Specifies output delay relative to the falling edge of the clock

-fall: Specifies the falling output delay at the port

-max: Applies value as maximum data required time

-min: Applies value as minimum data required time

-reference_pin <name>: Specifies a port in the design to which the output delay is relative

-rise: Specifies the rising output delay at the port

-source_latency_included: Specifies input delay already includes added source latency

<delay>: Time value

<targets>: Collection or list of output ports

Description

Specifies the data required times at the specified output ports relative the clock specified by the -clock option. The clock must refer to a clock name in the design.

Output delays can be specified relative to the rising edge (default) or falling edge (-clock_fall) of the clock.

If the output delay is specified relative to a simple generated clock (a generated clock with a single target), the clock arrival times to the generated clock are added to the data required time.

Output delays can be specified relative to a port (-reference_pin) in the clock network. Clock arrival times to the reference port are added to the data required time. Non-port reference pins are not supported.

Output delays can include clock source latency. By default the clock source latency of the related clock is added to the output delay value, but when the -source_latency_included option is specified, the clock source latency is not added because it was factored into the output delay value.

The maximum output delay (-max) is used for clock setup checks or recovery checks and the minimum output delay (-min) is used for clock hold checks or removal checks. If only one of -min and -max (or neither) is specified for a given port, the same value is used for both.

Separate rising (-rise) and falling (-fall) required times at the port can be specified. If only one of -rise and -fall are specified for a given port, the same value is used for both.

By default, set_output_delay removes any other output delays to the port except for those with the same -clock, -clock_fall, and -reference_pin combination. Multiple output delays relative to different clocks, clock edges, or reference pins can be specified using the -add_delay option.

The value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See help for the use_timequest_style_escaping command for details.

Example

```
# Simple output delay with the same value for min/max and rise/fall:
# 1) set on ports with names of the form myout*
set_output_delay -clock clk 0.5 [get_ports myout*]
# 2) set on all output ports
set_output_delay -clock clk 0.5 [all_outputs]

# Output delay with respect to the falling edge of clock
set_output_delay -clock clk -clock_fall 0.5 [get_ports myout*]

# Output delays for different min/max and rise/fall combinations
set_output_delay -clock clk -max -rise 0.5 [get_ports myout*]
set_output_delay -clock clk -max -fall 0.4 [get_ports myout*]
set_output_delay -clock clk -min -rise 0.4 [get_ports myout*]
set_output_delay -clock clk -min -fall 0.3 [get_ports myout*]

# Adding multiple output delays with respect to more than one clock
set_output_delay -clock clkA -min 0.2 [get_ports myout*]
set_output_delay -clock clkA -max 0.8 [get_ports myout*]
set_output_delay -clock clkA -clock_fall 0.6 [get_ports myout*] \
-add_delay
set_output_delay -clock clkB -min 1.1 [get_ports myout*] -add_delay
set_output_delay -clock clkB -max 1.5 [get_ports myout*] -add_delay

# Specifying an output delay relative to an external clock output port
set_output_delay -clock clk -reference_pin [get_ports clkout] 0.8 \
[get_ports myout*]
```

sdc_ext

Timing Constraints not defined in the SDC Spec Version 1.5 are implemented in this package. Any command in this package can be specified in a TimeQuest SDC file.

This package is loaded by default in the following executable:

- quartus_sta

This package is available for loading in the following executable:

- quartus_map

This package includes the following commands:

Command	Page
derive_clock_uncertainty.....	2-44
derive_pll_clocks.....	2-45
get_assignment_groups	2-46
get_fanins	2-47
get_fanouts.....	2-48
get_keepers	2-49
get_nodes	2-50
get_partitions.....	2-51
get_registers.....	2-52
remove_annotated_delay.....	2-53
remove_clock.....	2-54
reset_timing_derate	2-55
set_active_clocks	2-56
set_annotated_delay	2-57
set_max_skew	2-58
set_net_delay	2-60
set_scc_mode	2-61
set_time_format.....	2-62
set_timing_derate.....	2-63

derive_clock_uncertainty

Usage

```
derive_clock_uncertainty [-add] [-overwrite]
```

Options

-add: Adds results user-defined clock uncertainty assignments

-overwrite: Overwrites user-defined clock uncertainty assignments

Description

Applies inter-clock, intra-clock and I/O interface uncertainties based on timing model characterization. This command calculates and applies setup and hold clock uncertainties for each clock-to-clock transfer found in the design. The calculation of the uncertainties is delayed until the next `update_timing_netlist` call.

To get I/O interface uncertainty in addition to inter-clock and intra-clock uncertainties, create a virtual clock to represent an off-chip clock for input or output delay specification and assign delays to input/output ports with `set_input_delay` and `set_output_delay` commands that specify the virtual clock.

If `set_input_delay` and `set_output_delay` commands specifying a non-virtual clock are called, `derive_clock_uncertainty` applies either inter-clock or intra-clock uncertainty for that clock transfer since those transfers represent a clock-to-clock domain rather than an I/O-to-register clock domain.

These uncertainties are applied in addition to any previous `set_clock_uncertainty` calls. However, if there is already a clock uncertainty assignment for a source clock and destination clock pair, the new one is ignored. Either use the `-overwrite` option to overwrite previous clock uncertainty assignments or manually remove them by using `remove_clock_uncertainty` command. Use the `-add` option to add the previous user-defined clock uncertainty values to the derived ones.

This command auto-generates a file named `PLLJ_PLLSPE_INFO.txt` (or `PLLJ_PLLSPE_INFO_M.txt` if a military temperature range is selected) that lists the names of the PLLs in the design as well as their jitter and SPE values. This text file can be used by `HCI_DTW_CU_Calculator`.

Example

```
# create a virtual clock
create_clock -name virtual -period 1

# apply input/output delays with the virtual clock to get
# I/O interface uncertainties
set_input_delay -clock virtual -add_delay 0 [all_inputs]
set_output_delay -clock virtual -add_delay 0 [all_outputs]

# call derive_clock_uncertainty. results will be calculated
# at the next update_timing_netlist call
derive_clock_uncertainty

update_timing_netlist
```

derive_pll_clocks

Usage

```
derive_pll_clocks [-create_base_clocks] [-use_tan_name]
```

Options

-create_base_clocks: Creates base clocks on input clock ports of the design that are feeding the PLL

-use_tan_name: Use net names as clock names

Description

Identifies PLLs or similar resources in the design and creates generated clocks for their output clock pins. Multiple generated clocks may be created for each output clock pin if the PLL is using clock switchover, one for the inclk[0] input clock pin and one for the inclk[1] input clock pin.

By default this command does not create base clocks on input clock ports that are driving the PLL. When you use the create_base_clocks option, derive_pll_clocks also creates the base clock on an input clock port deriving the PLL. This option does not overwrite an existing clock.

By default the clock name is the same as the output clock pin name. To use the net name (the same name the classic Timing Analyzer would use), use the -use_tan_name option.

Example

```
project_open top
create_timing_netlist

# Create the base clock for the input clock port driving the PLL
create_clock -period 10.0 [get_ports sysclk]

# Create the generated clocks for the PLL.
derive_pll_clocks

update_timing_netlist

# Other user actions
report_timing

delete_timing_netlist
project_close
```

get_assignment_groups

Usage

```
get_assignment_groups [-keepers] [-ports] [-registers] <name>
```

Options

-keepers: Returns a keeper collection from the assignment group matching the <name>

-ports: Returns a port collection from the assignment group matching the <name>

-registers: Returns a register collection from the assignment group matching the <name>

<name>: Assignment group name

Description

Returns a collection of <keepers> | <registers> | <ports> for the assignment group that matches <name>. This command can be used to retrieve the assignment group created and saved in the Quartus II Settings File.

The options -keepers, -registers and -ports are mutually exclusive. If no option is specified, the keeper collection is returned by default.

The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or TimeQuest-extension substitution rules. See help for the use_timequest_style_escaping command for details.

Example

```
get_assignment_groups my_assignments -registers
```


get_fanins

Usage

```
get_fanins [-asynch] [-clock] [-inverting_paths] [-no_logic] [-non_inverting_paths]
[-synch] [-through <names>] <filter>
```

Options

-asynch: Traverse through asynch edges

-clock: Traverse through clock edges

-inverting_paths: Only follow inverting combinational paths

-no_logic: Do not follow combinational paths

-non_inverting_paths: Only follow non-inverting combinational paths

-synch: Traverse through synch edges

-through <names>: Valid through nodes (string patterns are matched using Tcl string matching)

<filter>: Valid starting nodes (string patterns are matched using Tcl string matching or collection)

Description

Returns a collection of fanin nodes starting from the <filter> in the design. When you supply the -no_logic option, get_fanins ignores the paths that pass through combinational logic elements other than buffers and inverters.

When you use the -synch, -asynch or -clock options, get_fanins traverses the netlist through corresponding edges. More than one of these options can be specified. If you do not specify any of these three options, the command does not ignore any paths.

When the -non_inverting_paths option is used, no_logic does not follow any paths that includes odd number of inverters. Similarly, when the -inverting_paths option is used, no_logic does not follow any paths that includes even number of inverters. Both the -non_inverting_paths and -inverting_paths options require the -no_logic option and are mutually exclusive.

When the -through option is used, only the fanins that can be reached by going through those nodes are returned.

The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or TimeQuest-extension substitution rules. See the help for the use_timequest_style_escaping command for details.

Example

```
set fanins [get_fanins $item -synch -clock]
foreach_in_collection fanin_keeper $fanins {
    lappend fanin_keeper_list [get_node_info $fanin_keeper -name]
}

set fanins_no_logic [get_fanins $item -no_logic -asynch]
foreach_in_collection fanin_keeper $fanins_no_logic {
    lappend fanin_keeper_list_no_logic [get_node_info $fanin_keeper \
        -name]
}

#-through example
get_fanins inst18 -through inst11
```

get_fanouts

Usage

```
get_fanouts [-inverting_paths] [-no_logic] [-non_inverting_paths] [-through <names>]
<filter>
```

Options

-inverting_paths: Only follow inverting combinational paths

-no_logic: Do not follow combinational paths

-non_inverting_paths: Only follow non-inverting combinational paths

-through <names>: Valid through nodes (string patterns are matched using Tcl string matching)

<filter>: Valid starting nodes (string patterns are matched using Tcl string matching or collection)

Description

Returns a collection of fanout nodes starting from the <filter> in the design. When the -no_logic option is used, get_fanouts ignores the paths that pass through combinational logic elements other than buffers and inverters.

When the -non_inverting_paths option is used in conjunction with the -no_logic option, get_fanouts does not follow any paths that include an odd number of inverters. Similarly, when the -inverting_paths option is used in conjunction with the -no_logic option, get_fanouts does not follow any paths that include an even number of inverters. Both the -non_inverting_paths and -inverting_paths options require the -no_logic option and are mutually exclusive.

When the -through option is used, only the fanouts that can be reached by going through those nodes are returned.

The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or TimeQuest-extension substitution rules. See the help for use_timequest_style_escaping for details.

Example

```
set fanouts [get_fanouts $item]
foreach_in_collection fanout_keeper $fanouts {
    lappend fanout_keeper_list [get_node_info $fanout_keeper -name]
}

set fanouts_no_logic [get_fanouts $item -no_logic]
foreach_in_collection fanout_keeper $fanouts_no_logic {
    lappend fanout_keeper_list_no_logic \
        [get_node_info $fanout_keeper -name]
}

# Using through option to find the fanout registers whose enable input is
# connected to the signal while ignoring the inverting paths.
get_fanouts inst1 -no_logic -non_inverting_paths -through \
    [get_pins -hierarchical *|ena]
```

get_keepers

Usage

```
get_keepers [-no_duplicates] [-nocase] [-nowarn] <filter>
```

Options

-no_duplicates: Do not match duplicated keeper names

-nocase: Specifies the matching of node names to be case-insensitive

-nowarn: Do not issue warnings messages about unmatched patterns

<filter>: Valid destinations (string patterns are matched using Tcl string matching)

Description

Returns a collection of non-combinational or "keeper" nodes in the design.

The default matching scheme returns not only non-combinational nodes whose names match the specified filter, but also non-combinational nodes duplicated from these keepers (refers to cells are automatically generated by Quartus from these keepers). Use the -no_duplicates option to exclude duplicated keepers.

The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or TimeQuest-extension substitution rules. See help for the use_timequest_style_escaping command for details.

Example

```
project_open chiptrip
create_timing_netlist

set kprs [get_keepers *reg*]
foreach_in_collection kpr $kprs {
    puts [get_object_info -name $kpr]
}

delete_timing_netlist
project_close
```

get_nodes

Usage

```
get_nodes [-no_duplicates] [-nocase] [-nowarn] <filter>
```

Options

-no_duplicates: Do not match duplicated node names

-nocase: Specifies the matching of node names to be case-insensitive

-nowarn: Do not issue warnings messages about unmatched patterns

<filter>: Valid destinations (string patterns are matched using Tcl string matching)

Description

Returns a collection of nodes in the design.

The default matching scheme returns not only nodes whose names match the specified filter, but also nodes duplicated from these nodes (refers to cells are automatically generated by Quartus from these nodes). Use the -no_duplicates option to not include duplicated nodes.

The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or TimeQuest-extension substitution rules. See help for the use_timequest_style_escaping command for details.

Example

```
project_open chiptrip
create_timing_netlist

set nodes [get_nodes *name*]
foreach_in_collection node $nodes {
    puts [get_object_info -name $node]
}

delete_timing_netlist
project_close
```

get_partitions

Usage

```
get_partitions [-cell] [-hierarchical] [-nocase] <filter>
```

Options

-cell: Returns a cell collection inside the partitions matching the <filter>
-hierarchical: Specifies if hierarchical searching method should be used
-nocase: Specifies the matching of node names to be case-insensitive
<filter>: Valid partitions (string patterns are matched using Tcl string matching)

Description

Returns a collection of partitions matching the filter by default. All partition names in the collection match the specified pattern. Wildcards can be used to select multiple partitions at once.

The -cell option creates and returns the collection of cells found inside the partitions matching the <filter> instead of returning a partition collection.

There are three Tcl string matching schemes available with this command: default, -hierarchical, and -no_case.

When using the default matching scheme, pipe characters separate one hierarchy level from the next. They are treated as special characters and are taken into account when string matching with wildcards is performed. The default matching scheme does not force the search to proceed recursively down the hierarchy.

When using the hierarchical matching scheme, pipe characters are treated as special characters and are taken into account when string matching with wildcards is performed. This matching scheme forces the search to proceed recursively down the hierarchy.

The -nocase matching scheme uses case-insensitive matching behavior.

The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or TimeQuest-extension substitution rules. See the help for use_timequest_style_escaping for details.

Example

```
#Get the partitions matching the filter  
get_partitions *
```

```
#Get the collection of cells inside partitions matching the filter  
get_partitions * -cell
```

get_registers

Usage

```
get_registers [-no_duplicates] [-nocase] [-nowarn] <filter>
```

Options

-no_duplicates: Do not match duplicated register names
-nocase: Specifies the matching of node names to be case-insensitive
-nowarn: Do not issue warnings messages about unmatched patterns
<filter>: Valid destinations (string patterns are matched using Tcl string matching)

Description

Returns a collection of registers in the design.

The default matching scheme returns not only registers whose names match the specified filter, but also returns registers duplicated from these registers (cells automatically generated from these registers by the Quartus II software). Use the -no_duplicates option to exclude duplicated registers.

The filter for the collection is a Tcl list of wildcards, and must follow standard Tcl or TimeQuest-extension substitution rules. See help for the use_timequest_style_escaping command for details.

Example

```
project_open chiptrip
create_timing_netlist

set regs [get_registers *reg*]
foreach_in_collection reg $regs {
    puts [get_object_info -name $reg]
}

delete_timing_netlist
project_close
```

remove_annotated_delay

Usage

```
remove_annotated_delay -all
```

Options

-all: Specifies removal of all annotated delays

Description

Removes annotated delays from the design.

Example

```
# annotate delay
set_annotated_delay -net -from [get_pins clk] 0.1
update_timing_netlist

# remove all annotated delays
remove_annotated_delay -all
update_timing_netlist
```

remove_clock

Usage

```
remove_clock [-all] <clock_list>
```

Options

-all: Removes all clocks from the design

<clock_list>: Clock(s) to be removed

Description

Removes the specified clock(s) from the design.

Example

```
# Create a clock and then remove it.  
create_clock -period 10 -name CLK [get_ports clk]  
remove_clock CLK
```


reset_timing_derate

Usage

```
reset_timing_derate
```

Options

None

Description

Resets all derate factors set on the design.

Example

```
# set timing derate
set_timing_derate -late 0.2 [get_cells *]
update_timing_netlist

# reset all derate factors
reset_timing_derate
update_timing_netlist
```

set_active_clocks

Usage

```
set_active_clocks <clocks>
```

Options

<clocks>: List or collection of clocks

Description

Sets the list of active clocks for timing analysis. All other clocks not in the list or collection are considered inactive.

Timing analysis is only performed on active clocks. All clocks are active by default. Generated clocks that are generated from inactive clocks are considered inactive. Therefore, to make a generated clock active, specify both the parent and generated clock when calling `set_active_clocks`.

To reset all clocks to active, call `"set_active_clocks *"` or `"set_active_clocks [all_clocks]"`.

`set_active_clocks` does not affect all reports. For example, inactive clocks are still reported by `report_clocks`, `report_clock_transfers`, and similar commands.

Example

```
# Only analyze clk1
set_active_clocks [get_clocks clk1]

# Only analyze clk2
set_active_clocks [get_clocks clk2]

# Analyze all clocks
set_active_clocks [all_clocks]
```

set_annotated_delay

Usage

```
set_annotated_delay [-cell] [-ff] [-fr] [-from <names>] [-max] [-min] [-net]
[-operating_conditions <operating_conditions>] [-rf] [-rr] [-to <names>] <delay>
```

Options

-cell: Specifies that cell delay must be set

-ff: Specifies that FF delay must be set

-fr: Specifies that FR delay must be set

-from <names>: Valid source pins or ports (string patterns are matched using Tcl string matching)

-max: Specifies that only max delay should be set

-min: Specifies that only min delay should be set

-net: Specifies that net delay must be set

-operating_conditions <operating_conditions>: Operating conditions Tcl object

-rf: Specifies that RF delay must be set

-rr: Specifies that RR delay must be set

-to <names>: Valid destination pins or ports (string patterns are matched using Tcl string matching)

<delay>: The delay value in default time units

Description

Annotates the cell delay between two or more pins/nodes on a cell, or the interconnect delay between two or more pins on the same net, in the current design. Multiple transition edges (rr, fr, rf, ff) can be specified. If no transition is specified, then the given delay is assigned to all four values. If either -from or -to (or both) values are left unspecified, the missing value or values are substituted by an "*" character. Options -max and -min allow users to specify max or only min delay. If neither -max or -min is specified, both delays are set. Using this command to reduce delay pessimism might lead to optimistic results from timing analysis.

The values for -from and -to are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See help for the use_timequest_style_escaping command for details.

Delay annotation is deferred until the next time update_timing_netlist is called. To remove annotated delays, use remove_annotated_delay command.

Example

```
set_annotated_delay -cell 100 -from A|B|C|datain -to A|B|C|combout -rr \
  -ff
set_annotated_delay -net 100 -to A|carryin
update_timing_netlist

# To clear all net delays
set_annotated_delay -net 0
update_timing_netlist

# To remove all annotated delay assignments
remove_annotated_delay -all
update_timing_netlist
```

set_max_skew

Usage

```
set_max_skew [-exclude <Tcl list>] [-fall_from_clock <names>] [-fall_to_clock <names>]
[-from <names>] [-from_clock <names>] [-include <Tcl list>] [-rise_from_clock <names>]
[-rise_to_clock <names>] [-through <names>] [-to <names>] [-to_clock <names>] <skew>
```

Options

-exclude <Tcl list>: A Tcl list of parameters to exclude during skew analysis. This list can include 1 or more of the following: utsu, uth, utco, from_clock, to_clock, clock_uncertainty, input_delay, output_delay

-fall_from_clock <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-fall_to_clock <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-from <names>: Valid sources (string patterns are matched using Tcl string matching)

-from_clock <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-include <Tcl list>: Tcl list of parameters to include during skew analysis. This list can include 1 or more of the following: utsu, uth, utco, from_clock, to_clock, clock_uncertainty, input_delay, output_delay

-rise_from_clock <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-rise_to_clock <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-through <names>: Valid through nodes (string patterns are matched using Tcl string matching)

-to <names>: Valid destinations (string patterns are matched using Tcl string matching)

-to_clock <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

<skew>: Required skew

Description

Use the set_max_skew constraint to perform maximum allowable skew analysis between sets of registers or ports. In order to constrain skew across multiple paths, all such paths must be defined within a single set_max_skew constraint. set_max_skew timing constraint is not affected by set_max_delay, set_min_delay and set_multicycle_path but it does obey set_false_path and set_clock_groups.

Legal values for the -from and -to options are collections of clocks, registers, ports, pins, nets, cells or partitions in a design.

Applying maximum skew constraints between clocks applies the constraint from all register or ports driven by the clock specified with the -from option to all registers or ports driven by the clock specified with the -to option.

If pin names or collections are used, the -from value must be a clock pin and the -to value must be any non-clock input pin to a register. Assignments from clock pins or to and from cells apply to all registers contained in the cell or driven by the clock pin. Similarly, -to and -from partition specifications apply to all registers in the specified partition.

Use the `-include` and `-exclude` options to include or exclude one or more of the following: register micro parameters (`utsu`, `uth`, `utco`), clock arrival times (`from_clock`, `to_clock`), clock uncertainty (`clock_uncertainty`) and input and output delays (`input_delay`, `output_delay`). By default, max skew analysis includes data arrival times, clock arrival times, register micro parameters and clock uncertainty. When `-include` is used, those in the inclusion list will be added to the default analysis. Similarly, when `-exclude` is used, those in the exclusion list will be excluded from the default analysis. When both the `-include` and `-exclude` options specify the same parameter, that parameter will be excluded.

When this constraint is used, results of max skew analysis are displayed in the Report Max Skew (`report_max_skew`) report from the TimeQuest Timing Analyzer. Since skew is defined between two or more paths, no results are displayed if the `-from/-from_clock` and `-to/-to_clock` filters satisfy less than two paths.

The Fitter does not include `set_max_skew` constraints in design optimization. Use placement, routing, or other timing constraints to drive the fitter to meet any `set_max_skew` constraints.

Example

```
# Constrain the skew on an input port to all registers it feeds
set_max_skew -from [get_ports din] 0.200

# Constrain the skew on output bus dout[*]
set_max_skew -to [get_ports dout\[*\]] 0.200

# Create a max skew constraint that includes only data path arrival
set_max_skew -from [get_keepers inst1|*] -to [get_keepers inst2|*] 0.200 \
-exclude { from_clock to_clock clock_uncertainty }

# Create a max skew constraint that includes input and output delays
# as well as the default data arrivals, clock arrivals and clock
# uncertainty
set_max_skew -from [get_keepers inst1|*] -to [get_keepers inst2|*] 0.200 \
-include { input_delay output_delay }

# Report the results of max skew assignments
report_max_skew -panel_name "Report Max Skew" -npaths 10 -detail \
  path_only
```

set_net_delay

Usage

```
set_net_delay -from <names> [-max] [-min] [-to <names>] <delay>
```

Options

-from <names>: Valid source pins, ports, registers or nets (string patterns are matched using Tcl string matching)

-max: Specifies maximum delay

-min: Specifies minimum delay

-to <names>: Valid destination pins, ports, registers or nets (string patterns are matched using Tcl string matching)

<delay>: Required delay

Description

Use the `set_net_delay` command to query the net delays and perform minimum or maximum timing analysis across nets. The `-from` and `-to` options can be string patterns or pin, port, register, or net collections. When pin or net collection is used, the collection should include output pins or nets.

If the `-to` option is unused or if the `-to` filter is an "*" character, all the output pins and registers on timing netlist became valid destination points.

When you use the `-min` option, slack is calculated by looking at the minimum delay on the edge. If you use `-max` option, slack is calculated with the maximum edge delay.

Example

```
project_open my_project
create_timing_netlist
read_sdc
update_timing_netlist

# add min delay constraint
set_net_delay -min 0.160 -from [get_pins inst9|combout] -to \
    [get_pins *|dataf]

# add max delay constraint
set_net_delay -max 0.500 -from inst8|combout

# this is same as the previous call
set_net_delay -max 0.500 -from inst8|combout -to *

update_timing_netlist

report_net_delay -panel "Net Delay"
```

set_scc_mode

Usage

```
set_scc_mode [-size <size>] [-use_heuristic]
```

Options

-size <size>: Maximum SCC loop size

-use_heuristic: Always use heuristic for SCC processing

Description

Allows you to set maximum Strongly Connected Components (SCC) loop size or force TimeQuest Timing Analyzer to always estimate delays through SCCs.

When TimeQuest Timing Analyzer encounters a loop of size greater than the specified maximum SCC loop size, it uses a heuristic which only estimates delays through the loop.

If the loop is smaller than the maximum SCC loop size, a full processing of loops is performed unless the -use_heuristic option is used.

Example

```
# Make TimeQuest Timing Analyzer use normal processing for all loops
# the size of which is less than or equal to 100. For loops of size
# greater than 100, a runtime-saving heuristic will be used
set_scc_mode -size 100

# Force TimeQuest Timing Analyzer to use heuristic for all SCCs
# disregarding their size
set_scc_mode -use_heuristic
```

set_time_format

Usage

```
set_time_format [-decimal_places <decimal_places>] [-unit <unit>]
```

Options

-decimal_places <decimal_places>: Number of decimal places to use

-unit <unit>: Default time unit to use

Description

Sets time format, including time unit and decimal places.

Time units are assumed to be nanoseconds (ns) by default. The "-unit" option overrides the default time units. Legal time unit values are: ps, ns, us, ms.

Time units are displayed with three decimal places by default. The "-decimal_places" option overrides the default number of decimal places to show.

The smallest resolution of all times units is one picosecond (ps). Any additional specified precision will be truncated.

Example

```
# Create two clocks with a clock period of 8 nanoseconds.  
create_clock -period 8.000 clk1
```

```
set_time_format -unit ps -decimal_places 0  
create_clock -period 8000 clk2
```


set_timing_derate

Usage

```
set_timing_derate [-cell_delay] [-early] [-late] [-net_delay] [-operating_conditions  
<operating_conditions>] <derate_value> <cells>
```

Options

-cell_delay: Specifies that derating factors are only to apply to cell delays

-early: Specifies the minimum derating factor. This factor specifies how early the signal can arrive

-late: Specifies the maximum derating factor. This factor specifies how late the signal can arrive

-net_delay: Specifies that derating factors are only to apply to net delays

-operating_conditions <operating_conditions>: Operating conditions Tcl object

<derate_value>: Timing derate value

<cells>: List of cell type objects

Description

Sets the global derate factors for the current design. The maximum and minimum delays of all timing arcs in the design are multiplied by the factors specified with the -late and -early options respectively. Only positive derate factors are allowed. If neither the -cell_delay nor -net_delay option is used, the derating factors apply to both cell and net delays.

Specifying a derate value of less than 1.0 for the -late option or a derate value of greater than 1.0 for the -early option reduces delay pessimism, which might lead to optimistic results from timing analysis.

The effect of set_timing_derate command is deferred until the next time update_timing_netlist is called. To reset derate factors to original values, use the reset_timing_derate command.

Example

```
set_timing_derate -early 0.9 [get_cells *]  
set_timing_derate -late 1.1 [get_cells *]
```

sta

This package contains the set of Tcl functions for obtaining information from the TimeQuest Timing Analyzer.

This package is loaded by default in the following executable:

- quartus_sta

This package is available for loading in the following executable:

- quartus_map

This package includes the following commands:

Command	Page
add_to_collection	2-67
check_timing	2-68
create_report_histogram	2-70
create_slack_histogram	2-72
create_timing_netlist	2-74
create_timing_summary	2-76
delete_timing_netlist	2-77
enable_ccpp_removal	2-78
enable_sdc_extension_collections	2-79
get_available_operating_conditions	2-80
get_cell_info	2-81
get_clock_domain_info	2-82
get_clock_fmax_info	2-83
get_clock_info	2-84
get_datasheet	2-86
get_default_sdc_file_names	2-88
get_edge_info	2-89
get_edge_slacks	2-90
get_min_pulse_width	2-91
get_net_info	2-92
get_node_info	2-93
get_object_info	2-94
get_operating_conditions	2-95
get_operating_conditions_info	2-96
get_partition_info	2-97
get_path	2-98
get_path_info	2-100
get_pin_info	2-103
get_point_info	2-104
get_port_info	2-107
get_register_info	2-108
get_timing_paths	2-109
locate	2-112
query_collection	2-114
read_sdc	2-115
remove_from_collection	2-116
report_advanced_io_timing	2-117
report_bottleneck	2-118
report_clock_fmax_summary	2-120
report_clock_transfers	2-121
report_clocks	2-122
report_datasheet	2-123
report_ddr	2-124
report_exceptions	2-125
report_max_skew	2-129
report_metastability	2-132
report_min_pulse_width	2-135
report_net_delay	2-137
report_net_timing	2-138
report_partitions	2-139

report_path	2-140
report_rskm	2-142
report_sdc.....	2-143
report_skew	2-144
report_tccs	2-147
report_timing.....	2-148
report_ucp	2-152
set_operating_conditions.....	2-153
timing_netlist_exist.....	2-154
update_timing_netlist	2-155
use_timequest_style_escaping	2-156
write_sdc	2-157

add_to_collection

Usage

```
add_to_collection <collection_obj_1> <collection_obj_2>
```

Options

```
<collection_obj_1>: First object collection
```

```
<collection_obj_2>: Second object collection
```

Description

This command takes two collections and returns a new collection that is a union of the two. The second collection is allowed to be a string, whereas the first has to be previously-created collection, either by passing any of the "get_" functions directly, or by passing a variable that contains a collection (see code examples for this command). If a collection is used for the second argument, the types in the second collection must be the same as or a subset of the types in the first collection.

If the first collection consists of keepers, the second collection can only consist of keepers, registers or ports. If the first collection consists of partitions, the second collection can only consist of partitions or cells. If the first collection consists of nodes, the second collection can only consist of nodes, keepers, registers, ports, pins, nets or combinational nodes.

Example

```
set kprs1 [get_keepers b*]
set regs1 [get_registers a*]

set regs_union [add_to_collection $kprs1 $regs1]

#or:
set regs_union [add_to_collection [get_keepers b*] $regs1]

#or even:
set regs_union [add_to_collection $kprs1 a*]

# note that the last statement will actually add all keepers with name
# a* not only registers! (will add IOs with name a*, if any)

# Get the first 100 nodes in the collection.
query_collection $regs_union -limit 100
```

check_timing

Usage

```
check_timing [-append] [-file <name>] [-include <check_list>] [-panel_name <name>]
[-stdout]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-include <check_list>: Checks to perform

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

Description

Checks for problems in the design or problems with design constraints. The `check_timing` command performs a series of different checks based on user-specified variables and options. There is no default list of checks. Use the `-include` option to specify which checks to perform. You must precede `check_timing` with `update_timing_netlist`.

The `no_clock` check reports whether registers have at least one clock at their clock pin, and that ports determined to be clocks have a clock assigned to them, and also checks that PLLs have a clock assignment.

The `multiple_clock` check verifies that registers have at most one clock at their clock pin. (When multiple clocks reach a register clock pin, it is undefined which clock is used for analysis.)

The `generated_clock` check verifies that generated clocks are valid. Generated clocks must have a source that is triggered by a valid clock.

The `no_input_delay` check verifies that every input port that is not determined to be a clock has an input delay assignment.

The `no_output_delay` check verifies that every output port has an output delay constraint.

The `partial_input_delay` check verifies that input delays are complete, and ensures that input delays have a rise-min, fall-min, rise-max, and fall-max portion set.

The `partial_output_delay` check verifies that output delays are complete, and makes sure that output delays have a rise-min, fall-min, rise-max, and fall-max portion set.

The `io_min_max_delay_consistency` check verifies that min delay values specified by `set_input_delay` or `set_output_delay` assignments are less than max delay values.

The `reference_pin` check verifies that reference pins specified in `set_input_delay` and `set_output_delay` using the `-reference_pin` option are valid. A `reference_pin` is valid if the `-clock` option specified in the same `set_input_delay/set_output_delay` command matches the clock that is in the direct fanin of the `reference_pin`. Being in the direct fanin of the `reference_pin` means that there must be no keepers between the clock and the `reference_pin`.

The `latency_override` check reports whether the clock latency set on a port or pin overrides the more generic clock latency set on a clock. Clock latency can be set on a clock, where the latency applies to all keepers clocked by the clock, whereas clock latency can also be set on a port or pin, where the latency applies to registers in the fanout of the port or pin.

The loops check verifies that there are no strongly connected components in the netlist. These loops prevent a design from being properly analyzed. The loops check also reports if loops exist but were marked so that they would not be traversed.

The latches check reports latches in the design and warns that latches may not be analyzed properly. For best results, change your design to remove latches whenever possible.

The pos_neg_clock_domain check determines if any register is clocked by both the rising and falling edges of the same clock. If this scenario is necessary such as in a clock multiplexer, create two separate clocks that have similar settings and are assigned to the same node.

The pll_cross_check checks the clocks that are assigned to a PLL against the PLL settings defined in design files. Inconsistent settings or an unmatched number of clocks associated with the PLL are reported to the user.

The uncertainty check reports each clock-to-clock transfer that does not have a clock uncertainty assignment set between the two clocks. When a device family has derive_clock_uncertainty support, this report also checks if a user-defined set_clock_uncertainty assignment has a less than recommended clock uncertainty value.

The virtual_clock check reports all unreferenced virtual clocks. It also reports if design does not have any virtual clock assignment.

The partial_multicycle check ensures that each setup multicycle assignment has a corresponding hold multicycle assignment, and each hold multicycle assignment has a corresponding setup multicycle assignment.

The multicycle_consistency check reports all the multicycle cases where a setup multicycle does not equal one greater than the hold multicycle. Hold multicycle assignments are usually one cycle less than setup multicycle assignments.

The partial_min_max_delay check verifies that each minimum delay assignment has a corresponding maximum delay assignment, and vice versa.

The clock_assignments_on_output_ports check reports all the clock assignments that have been applied to output ports.

The input_delay_assigned_to_clock check verifies that no input delay value is set for a clock. Input delays set on clock ports are ignored because clock-as-data analysis takes precedence.

The generated_io_delay check reports all the IO delays that have no specifications for -reference_pin, -clock (generated clocks), or -source_latency_included.

Example

```
# Constrain design
create_clock -name clk -period 4.000 -waveform { 0.000 2.000 } \
  [get_ports clk]
set_input_delay -clock clk2 1.5 [get_ports in*]
set_output_delay -clock clk 1.6 [get_ports out*]
set_false_path -from [get_keepers in] -through [get_nets r1] -to \
  [get_keepers out]

# Check if there were any problems
check_timing -include {loops latches no_input_delay partial_input_delay}
```

create_report_histogram

Usage

```
create_report_histogram [-color_div <color_div>] [-color_list <color_list>] [-max_data
<max_data>] [-min_data <min_data>] [-num_bins <num_bins>] [-panel_name <panel_name>]
[-x_label <x_label>] [-x_unit <x_unit>] [-y_label <y_label>] <data>
```

Options

```
-color_div <color_div>: Color divisions for the created histogram
-color_list <color_list>: List of colors for painting the created histogram
-max_data <max_data>: Maximum data value of the created histogram
-min_data <min_data>: Minimum data value of the created histogram
-num_bins <num_bins>: Number of bins
-panel_name <panel_name>: Path and name of the histogram
-x_label <x_label>: Text label on x-axis
-x_unit <x_unit>: Unit to be displayed on x-axis
-y_label <y_label>: Text label on y-axis
<data>: List of data to be analyzed
```

Description

Create a user defined histogram.

Use <data> to specify the data entries to be displayed on the histogram. It can be a tcl list of either one of the following two formats or a mix of the two: {time_integer} or {time_integer number_count}, where time_integer is an integer possibly with a unit representing time (default unit is second), and number_count is a positive integer specifying number of entries (y value) of the corresponding time_integer.

Use -num_bins to specify the number of bins, or the number of bars to be displayed on the histogram.

Use -color_div and -color_list to specify the color of each bin. -color_div takes a tcl list of time_integers (see <data> above). Each entry in the list specifies the upper bound of each color division and therefore is forced to be a boundary of bins. -color_list takes a tcl list of colors. Each color in the list is used in the order specified and if less color is given than color divisions, the list will be re-used. For example, if specified "-color_div {-1 0 1} -color_list {red green}", then bins below -1 will be red, bins between -1 and 0 will be green, bins between 0 and 1 will be red again, and bins larger than 1 will be blue again. Possible choices of colors: black, blue, brown, green, grey, light_grey, orange, purple, red, white. Default -color_div is {0} and default -color_list is {red blue}.

Use -max_data to specify the upper bound, i.e. largest number to be included in the histogram.

Use -min_data to specify the lower bound, i.e. smallest number to be included in the histogram.

Use -panel_name to specify the path and panel name of the created histogram. e.g. "-panel_name {Folder 1 | | Histogram 1}" will create a histogram named "Histogram 1" and put it in a folder with the name "Folder 1".

Use -x_label to specify the text label on x_axis.

Use -y_label to specify the text label on y_axis.

Use -x_unit to specify a text unit to be attached to x_axis.

Example

```
# create a path-based slack histogram
project_open my_project
create_timing_netlist
read_sdc
update_timing_netlist

# get path-based slack data in the format of a tcl list
set data [list]
set paths [get_timing_paths -setup -npaths 1000]
foreach_in_collection path $paths {
    lappend data [get_path_info $path -slack]
}

# output data to histogram
create_report_histogram $data -panel_name {Path-based Slack Histogram} \
    -num_bins 20

delete_timing_netlist
project_close
```

create_slack_histogram

Usage

```
create_slack_histogram [-append] [-clock_name <name>] [-file <name>] [-hold]
[-max_slack <max_slack>] [-min_slack <min_slack>] [-num_bins <num_bins>] [-panel_name
<name>] [-recovery] [-removal] [-setup] [-stdout]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-clock_name <name>: Name of the Clock Domain

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-hold: Hold Analysis

-max_slack <max_slack>: Maximum slack value of the created histogram

-min_slack <min_slack>: Minimum slack value of the created histogram

-num_bins <num_bins>: Number of bins

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-recovery: Recovery Analysis

-removal: Removal Analysis

-setup: Setup Analysis

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

Description

Creates a slack histogram in the timing report for the specified clock domain "-clock_name," showing the number of timing edges within various ranges of slacks for a clock setup analysis. The histogram can be named using the "-panel_name" option.

Use the "-setup", "-hold", "-recovery", or "-removal" options to specify which kind of analysis should be performed. If none is specified, setup analysis is used by default.

Reports can be directed to the Tcl console ("-stdout", default), a file ("-file"), the TimeQuest graphical interface ("-panel_name"), or any combination of the three.

The range of reported slack values can be controlled by specifying the "-min_slack" and "-max_slack" options. The number of bins (histogram bars) can also be specified using the "-num_bins" option.

Example

```
project_open top
create_timing_netlist
read_sdc
update_timing_netlist

# Create a slack histogram for clk1, defaulting to
# the name "Slack Histogram (clk1)"
create_slack_histogram -clock_name clk1

# Create a slack histogram for clk2 named "MyHistogram"
create_slack_histogram -clock_name clk2 -panel_name MyHistogram
```

```
delete_timing_netlist  
project_close
```

create_timing_netlist

Usage

```
create_timing_netlist [-force_dat] [-grade <c|i|m|e|a>] [-model <fast|slow>]
[-no_latch] [-post_map] [-speed <speed>] [-temperature <value_in_C>] [-voltage
<value_in_mV>] [-zero_ic_delays] <operating_conditions>
```

Options

```
-force_dat: Option to force delay annotation
-grade <c|i|m|e|a>: Option to specify temperature grade
-model <fast|slow>: Option to specify timing model
-no_latch: Option to disable the analysis of latches as synchronous elements
-post_map: Option to perform timing analysis on the post-synthesis netlist
-speed <speed>: Speed grade
-temperature <value_in_C>: Operating temperature
-voltage <value_in_mV>: Operating voltage
-zero_ic_delays: Option to set all IC delays to zero
<operating_conditions>: Operating conditions Tcl object name string
```

Description

Creates the timing netlist by annotating the atom netlist with delay information using post-fitting results. Use the `-post_map` option to obtain post-synthesis results.

The `create_timing_netlist` command skips delay annotation by default. Use `-force_dat` to rerun delay annotation. This is required if any delay annotation setting is changed in the Quartus II project revision (e.g. `OUTPUT_PIN_LOAD`).

Use `"-model fast"` to run the analysis using the fast corner delay models first. The `-temperature`, `-voltage`, and `-speed`, options are also available. See help for `set_operating_conditions` for details on these options.

You can use `model`, `temperature` and `voltage` options to specify operating conditions while creating timing netlist (`temperature` and `voltage` options are not supported by all families). You can also set operating conditions by passing an operating conditions object name as a positional argument to `create_timing_netlist` command. After timing netlist has been created, you can use `set_operating_conditions` command to change timing models without deleting and re-creating the timing netlist.

Use the `-grade` option to analyze the design at a different temperature grade. This option is provided to support what-if analysis and is not recommended for final sign-off analysis.

Use the `-no_latch` option to analyze latches as combinational loops instead of synchronous elements.

Use the `-zero_ic_delays` option to set all IC delays in the netlist to zero.

Example

```
project_open my_top

# Create timing netlist before calling
# any report functions
create_timing_netlist

# Read SDC and update timing
read_sdc
update_timing_netlist
```

```
# Ready to call report functions
report_timing -npaths 1 -clock_setup

# The following command is optional
delete_timing_netlist

project_close

project_open my_top

# Report worst case period for -9 speed grade
create_timing_netlist -speed 9

# Read SDC and update timing
read_sdc
update_timing_netlist

report_timing -clock_setup -clock_filter clk
delete_timing_netlist

# Report hold violation for fastest corner
# Use set_operating_conditions instead
create_timing_netlist -model fast

# Read SDC and update timing
read_sdc
update_timing_netlist

report_timing -clock_hold -clock_filter clk
delete_timing_netlist

# If Delay Annotation has been run for the fast corner
# Force Delay Annotation
create_timing_netlist -model fast -force_dat

# Read SDC and update timing
read_sdc
update_timing_netlist

report_timing -clock_hold -clock_filter clk
delete_timing_netlist

# Report worst case period for post-technology mapping netlist
create_timing_netlist -post_map

# Read SDC and update timing
read_sdc
update_timing_netlist

report_timing -clock_setup -clock_filter clk
delete_timing_netlist

project_close
```

create_timing_summary

Usage

```
create_timing_summary [-append] [-file <name>] [-hold] [-panel_name <name>] [-recovery]
[-removal] [-setup] [-stdout]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-hold: Hold Analysis

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-recovery: Recovery Analysis

-removal: Removal Analysis

-setup: Setup Analysis (Default)

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

Description

Reports the worst-case Clock Setup and Clock Hold slacks and endpoint TNS (total negative slack) per clock domain. Total negative slack is the sum of all slacks less than zero for either destination registers or ports in the clock domain.

This command shows the worst-case slack for each clock domain. You right click in these reports to run more detailed reports like Histograms and Report Timing.

By default, this command creates a Setup Summary. This command can also generate a Hold Summary (-hold), Recovery Summary (-recovery), Removal Summary (-removal), or Minimum Pulse Width Summary (-mpw).

The report can be directed to the Tcl console (-stdout, default), a file (-file), the TimeQuest graphical interface (-panel_name), or any combination of the three.

Example

```
project_open my_project

# Always create the netlist first and process constraints
create_timing_netlist
read_sdc my_project.sdc
update_timing_netlist

# Create Clock Domain Summary
create_timing_summary -panel_name "Setup Summary"
create_timing_summary -hold -panel_name "Hold Summary"

# The following command is optional
delete_timing_netlist

project_close
```

delete_timing_netlist

Usage

```
delete_timing_netlist
```

Options

None

Description

Use this command to delete a timing netlist previously created using `create_timing_netlist`. This should be done at the end of a script or before calling `create_timing_netlist` again using different options or after recompiling the design.

Use the `set_operating_conditions` command instead of `delete_timing_netlist` and `create_timing_netlist` to change timing models. This avoids the cost of deleting and re-creating the timing netlist, and also preserves current timing assignments.

enable_ccpp_removal

Usage

```
enable_ccpp_removal [-depth <depth>] [-off] [-on]
```

Options

-depth <depth>: maximum clock tree depth for ccpp removal

-off: Disable this setting.

-on: Enable this setting.

Description

Enables (or disables) common clock path pessimism (CCPP) removal during slack computation. CCPP removal can improve timing results by removing min/max delay differences from common portions of clock paths. Enabling CCPP removal increases the time required to perform timing analysis.

When specified, the optional depth parameter limits the clock tree depth used for CCPP removal. This is generally not applicable to FPGA compilations where the clock tree is fixed, but for large HardCopy designs with potentially deep synthesized clock trees this can reduce outlier run time.

When not specified, or when specified with a value of 0, the complete clock tree is used for CCPP removal (i.e. full clock-tree depth).

Example

```
project_open top
create_timing_netlist
read_sdc

# Report timing without CCPP removal
report_timing

# Enable CCPP removal and re-report timing.
enable_ccpp_removal
report_timing

delete_timing_netlist
project_close
```


enable_sdc_extension_collections

Usage

```
enable_sdc_extension_collections [-off] [-on]
```

Options

-off: Disable this setting.

-on: Enable this setting.

Description

Enable the support of SDC extension collections, such as keeper, register and node collections. When `enable_sdc_extension_collections` is not used, using these collections causes an error. Default to -on option.

Example

```
project_open top
enable_sdc_extension_collections -on
create_timing_netlist
read_sdc

report_timing -to_clock clk1

delete_timing_netlist
project_close
```

get_available_operating_conditions

Usage

```
get_available_operating_conditions [-all]
```

Options

-all: Returns all available operating conditions

Description

Returns a Tcl collection of available operating conditions for the current device. The Tcl collection contains the most extreme operating conditions within a user-specified junction temperature range. Use the -all option to return all available operating conditions.

Example

```
#do report timing for different operating conditions
foreach_in_collection op [get_available_operating_conditions] {
    set_operating_conditions $op
    update_timing_netlist
    report_timing
}

#see detailed information about operating conditions
foreach_in_collection op [get_available_operating_conditions] {
    puts "Delay Model: [get_operating_conditions_info $op -model]"
}
```

get_cell_info

Usage

```
get_cell_info [-buried_nodes] [-buried_regs] [-in_pin_names] [-in_pins] [-location]
[-name] [-out_pin_names] [-out_pins] [-pin_names] [-pins] [-type] [-wysiwyg_type]
<cell_object>
```

Options

-buried_nodes: Return a collection of buried node IDs
-buried_regs: Return a collection of buried register IDs
-in_pin_names: Return a list of input pin names
-in_pins: Return a collection of input pin IDs
-location: Return the atom location in device
-name: Return the cell name
-out_pin_names: Return a list of output pin names
-out_pins: Return a collection of output pin IDs
-pin_names: Return a list of input and output pin names
-pins: Return a collection of input and output pin IDs
-type: Return the cell type
-wysiwyg_type: Return the WYSIWYG type of the cell
<cell_object>: Cell object

Description

Gets information about the specified cell (referenced by cell ID). You can obtain cell using the get_cells Tcl command.

The "-type" option returns "cell".

Options "-name", "-type", "-pin_name", "-in_pin_names", "-out_pin_names", "-pins", "-clock_pins", "-in_pins", "-out_pins", "-buried_nodes", "-buried_regs", "-location", and "-wysiwyg_type" are mutually exclusive.

Example

```
project_open chiptrip
create_timing_netlist
set cells [get_cells]
foreach_in_collection cell $cells {
    puts "[get_cell_info $cell -name]: [get_cell_info $cell -type]"
}
delete_timing_netlist
project_close
```

get_clock_domain_info

Usage

```
get_clock_domain_info [-hold] [-mpw] [-recovery] [-removal] [-setup]
```

Options

```
-hold: Hold Analysis
-mpw: Minimum Pulse Width Analysis
-recovery: Recovery Analysis
-removal: Removal Analysis
-setup: Setup Analysis (Default)
```

Description

Similar to `create_timing_summary`, the `get_clock_domain_info` command returns a Tcl list of information about each clock domain. Each entry in the list is a list of four elements: the clock name, worst-case slack, endpoint TNS, and edge TNS. TNS is total negative slack, and it is the sum of all slacks less than zero for either destination registers or ports in the clock domain (endpoint TNS) or for all edges affecting the clock domain (edge TNS).

By default, this command creates a Setup Summary. This command can also generate a Hold Summary (-hold), Recovery Summary (-recovery), Removal Summary (-removal), or Minimum Pulse Width Summary (-mpw).

Example

```
project_open my_project

# Always create the netlist first
create_timing_netlist
read_sdc my_project.sdc
update_timing_netlist

# Get domain summary object
set domain_list [get_clock_domain_info -setup]
foreach domain $domain_list {
    set name [lindex $domain 0]
    set slack [lindex $domain 1]
    set keeper_tns [lindex $domain 2]
    set edge_tns [lindex $domain 3]

    puts "Clock $name : Slack = $slack , TNS = ( $keeper_tns , $edge_tns \
        )"
}

# The following command is optional
delete_timing_netlist

project_close
```

get_clock_fmax_info

Usage

```
get_clock_fmax_info
```

Options

None

Description

Reports potential Fmax for every clock in the design, regardless of the user-specified clock periods. Fmax is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored. For paths between a clock and its inversion, Fmax is computed as if the rising and falling edges of the clock are scaled along with fmax, such that the duty cycle (in terms of a percentage) is maintained.

Restricted Fmax considers hold timing in addition to setup timing, as well as minimum pulse and minimum period restrictions. Similar to unrestricted Fmax, the restricted Fmax is computed as if the rising and falling edges of the clock are scaled along with Fmax, such that the duty cycle (in terms of a percentage) is maintained. Refer to hold timing reports (e.g., report_timing with the -hold option) or minimum pulse width reports (report_min_pulse_width) for details about specific paths, registers, or ports.

This command is similar to report_clock_fmax_summary, except that it returns the results as a Tcl list for use in Tcl scripts. Each entry in the list represents one clock domain. Each entry is a Tcl list of the clock name, fmax (MHz), and restricted Fmax (MHz).

Example

```
project_open my_project

# Always create the netlist first
create_timing_netlist
read_sdc my_project.sdc
update_timing_netlist

# Get domain summary object
set domain_list [get_clock_fmax_info]
foreach domain $domain_list {
    set name [lindex $domain 0]
    set fmax [lindex $domain 1]
    set restricted_fmax [lindex $domain 2]

    puts "Clock $name : Fmax = $fmax (Restricted Fmax = \
        $restricted_fmax)"
}

# The following command is optional
delete_timing_netlist

project_close
```

get_clock_info

Usage

```
get_clock_info [-divide_by] [-duty_cycle] [-edge_shifts] [-edges] [-fall]
[-is_inverted] [-latency] [-master_clock] [-master_clock_pin] [-max] [-min]
[-multiply_by] [-name] [-nreg_neg] [-nreg_pos] [-offset] [-period] [-phase] [-rise]
[-targets] [-type] [-waveform] <clk_object>
```

Options

-divide_by: Return the frequency divider (to the base clock)

-duty_cycle: Return the duty cycle

-edge_shifts: Return a list of edge shifts that the specified edges are to undergo to yield the final generated clock waveform

-edges: Return a list of integer representing edges from the source clock that are to form edges of the generated clock

-fall: Return clock fall latency

-is_inverted: Return a boolean value to indicate if the generated clock is inverted

-latency: Return clock latency

-master_clock: Return the master clock name

-master_clock_pin: Return the master clock source pin

-max: Return max clock latency

-min: Return min clock latency

-multiply_by: Return the frequency multiplier (to the base clock)

-name: Return the clock name

-nreg_neg: Return number of registers negatively clocked by clock

-nreg_pos: Return number of registers positively clocked by clock

-offset: Return the clock offset

-period: Return the clock period

-phase: Return the clock phase

-rise: Return clock rise latency

-targets: Return the clock targets collection

-type: Return the clock type

-waveform: Return the waveform (rise time and fall time)

<clk_object>: Clock object

Description

Returns information about the specified clock (referenced by clock ID). Clock IDs can be obtained by Tcl commands such as `get_clocks`.

The "-type" option returns "clk".

Options "-name", "-type", "-period", "-duty_cycle", "-waveform", "-edges", "-edge_shifts", "-multiply_by", "-divide_by", "-is_inverted", "-latency", "-master_clock", and "-targets" are mutually exclusive. The "-latency" option requires a specified "-max" or "-min" option as well as a "-rise" or "-fall" option.

Example

```
project_open chiptrip
create_timing_netlist
set clocks [get_clocks]
foreach_in_collection clk $clocks {
    puts "[get_clock_info $clk -name]: [get_clock_info $clk -period]"
}
delete_timing_netlist
project_close
```

get_datasheet

Usage

get_datasheet

Options

None

Description

This function returns a tcl collection which contains the datasheet report. Its format is as follows:

```

{
  { tsu
    { <tsu rise time>
      <tsu fall time>
      <input port>
      <clock port>
      <clock edge>
      <clock reference>
    }
  }
  { th
    { <th rise time>
      <th fall time>
      <input port>
      <clock port>
      <clock edge>
      <clock reference>
    }
  }
  { tco
    { <tco rise time>
      <tco fall time>
      <output port>
      <clock port>
      <clock edge>
      <clock reference>
    }
  }
  { mintco
    { <mintco rise time>
      <mintco fall time>
      <output port>
      <clock port>
      <clock edge>
      <clock reference>
    }
  }
  { tpd
    { <tpd rise-rise time>
      <tpd rise-fall time>
      <tpd fall-rise time>
      <tpd fall-fall time>
      <input port>
      <output port>
    }
  }
  { mintpd
    { <mintpd rise-rise time>

```



```
    <mintpd rise-fall time>
    <mintpd fall-rise time>
    <mintpd fall-fall time>
    <input port>
    <output port>
  }
}
```

There are no options for this command, and the data returned is the same as from the `report_datasheet` command.

Example

```
project_open proj1
create_timing_netlist
read_sdc
update_timing_netlist

# get the datasheet collection
set datasheet [get_datasheet]

# loop through contents of datasheet collection
foreach i $datasheet {
  foreach j $i {
    foreach k $j {
      #
      # extract individual items or
      # manipulate as necessary
      #
    }
  }
}
```

get_default_sdc_file_names

Usage

```
get_default_sdc_file_names
```

Options

None

Description

Returns the default SDC file name(s) used by the Quartus II Compiler when doing timing-driven optimizations.

Returns the value for the QSF variable SDC_FILE. If multiple assignments are found, return them as a list. If not specified, return <revision_name>.sdc.

Example

```
project_new test
create_timing_netlist
foreach file [get_default_sdc_file_names] {
    read_sdc $file
}
update_timing_netlist

report_timing

delete_timing_netlist
project_close
```

get_edge_info

Usage

```
get_edge_info [-delay] [-delay_type] [-dst] [-ff] [-fr] [-max] [-min] [-name] [-rf]
[-rr] [-src] [-type] [-unateness] <edge_object>
```

Options

-delay: Return the delay.
-delay_type: Return the type of the delay (ic/cell).
-dst: Return the destination node ID.
-ff: Return the fall-to-fall delay
-fr: Return the fall-to-rise delay
-max: Max delay
-min: Min delay
-name: Return the edge name
-rf: Return the rise-to-fall delay
-rr: Return the rise-to-rise delay
-src: Return the source node ID
-type: Return the edge type.
-unateness: Return the unateness.
<edge_object>: Edge object

Description

Returns information about the specified edge (referenced by edge ID). Edge ID's can be obtained by Tcl commands such as `get_node_info <node_id> -synch_edges`.

The "-type" option Returns "edge".

The "-delay" option returns the delay associated to the edge. Use -max/min and -rr/rf/fr/ff options to specify the type of returned delay. One of the -max/min options must be specified. One of the -rr/rf/fr/ff options must be specified.

The -unateness option returns the unateness associated to the edge.

Example

```
project_open chiptrip
create_timing_netlist
set nodes [get_nodes Reg*]
foreach_in_collection node $nodes {
    set edges [get_node_info $node -fanout_edges]
    foreach edge $edges {
        set rr_delay [get_edge_info $edge -delay -rr]
        set rf_delay [get_edge_info $edge -delay -rf]
        set fr_delay [get_edge_info $edge -delay -fr]
        set ff_delay [get_edge_info $edge -delay -ff]
        puts "Total cell delay of edge $edge: $rr_delay $rf_delay \
            $fr_delay $ff_delay"
    }
}
delete_timing_netlist
project_close
```

get_edge_slacks

Usage

```
get_edge_slacks [-hold] [-recovery] [-removal] [-setup]
```

Options

```
-hold: Hold analysis  
-recovery: Recovery analysis  
-removal: Removal analysis  
-setup: Setup analysis
```

Description

Returns a collection of edge slack pairs for the specified analysis type. A setup analysis is performed by default if no option is specified. Results are sorted by slack, then by the name of the source node for the edge, and last by the node name of the destination of the edge.

Example

```
project_open top  
create_timing_netlist  
read_sdc  
update_timing_netlist  
  
foreach_in_collection edge_slack [get_edge_slacks -setup] {  
    # Each item in the collection is an {edge slack} pair  
    set edge [lindex $edge_slack 0]  
    set slack [lindex $edge_slack 1]  
  
    set src_node [get_edge_info -src $edge]  
    set dst_node [get_edge_info -dst $edge]  
  
    post_message -type info "Found edge [get_node_info -name $src_node] \  
        -> [get_node_info -name $dst_node] with slack $slack"  
}  
  
delete_timing_netlist  
project_close
```

get_min_pulse_width

Usage

```
get_min_pulse_width [-nworst <number>] <targets>
```

Options

-nworst <number>: Specifies the number of pulse width checks to report (default=1)

<targets>: Registers or ports

Description

This command returns a Tcl list which contains the minimum pulse width report. Its format is as follows:

```
{  
  { <slack>,  
    <actual width>,  
    <required width>,  
    <pulse>,  
    <clock>,  
    <clock edge>,  
    <target>  
  }  
}
```

Refer to help for the report_min_pulse_width command for help on the -nworst and -targets options.

get_net_info

Usage

```
get_net_info [-name] [-pin] [-type] <net_object>
```

Options

```
-name: Return the net name  
-pin: Return the pin ID of this net  
-type: Return the net type.  
<net_object>: Net object
```

Description

Returns information about the specified net (referenced by net ID). Net ID's can be obtained by Tcl commands such as `get_nets`.

The "-type" option returns "net".

The options "-name", "-type", and "-pin" are mutually exclusive.

Example

```
project_open chiptrip  
create_timing_netlist  
  
set nets [get_nets]  
foreach_in_collection net $nets {  
    puts [get_net_info $net -name]  
}  
  
delete_timing_netlist  
project_close
```

get_node_info

Usage

```
get_node_info [-asynch_edges] [-cell] [-clock_edges] [-fanout_edges] [-location]
[-name] [-synch_edges] [-type] <node_object>
```

Options

-asynch_edges: Return a list of asynchronous edge IDs
-cell: Return the host cell
-clock_edges: Return a list of clock edge IDs
-fanout_edges: Return a list of fanout edge IDs
-location: Return the atom location in device
-name: Return the node name
-synch_edges: Return a list of synchronous edge IDs
-type: Return the node type
<node_object>: Node object

Description

Gets information about the specified node (referenced by node ID). Use Tcl commands such as `get_nodes` to obtain node IDs. The `-type` option returns "reg", "port", "pin", "net", or "comb". The `-name`, `-type`, `-clock_edges`, `-synch_edges`, `-asynch_edges`, `-fanout_edges`, `-cell` and `-location` options are mutually exclusive.

Example

```
project_open chiptrip
create_timing_netlist
set registers [get_registers]
foreach_in_collection reg $registers {
    puts "[get_node_info $reg -name]: [get_node_info $reg -type]"
}
delete_timing_netlist
project_close
```

get_object_info

Usage

```
get_object_info [-name] [-type] <object>
```

Options

-name: Return the object name

-type: Return the object type

<object>: Object

Description

Gets information about the specified object (referenced by object ID). Object IDs can be obtained by Tcl commands such as `get_clocks`, `get_ports`, `get_cells`, and others. The `-type` option returns "clk", "reg", "port", "cell", "pin", "comb", "net", or "edge". The `-name` and `-type` options are mutually exclusive.

Example

```
project_open chiptrip
create_timing_netlist
set ports [get_ports]
foreach_in_collection port $ports {
    puts [get_object_info $port -name]
}
delete_timing_netlist
project_close
```


get_operating_conditions

Usage

`get_operating_conditions`

Options

None

Description

Returns the current `operating_conditions` Tcl Obj.

Example

```
puts "Delay Model : [get_operating_conditions_info \  
    [get_operating_conditions] -model]"
```

get_operating_conditions_info

Usage

```
get_operating_conditions_info [-display_name] [-grade] [-model] [-name] [-speed]
[-temperature] [-voltage] <operating_condition>
```

Options

-display_name: Returns the operating conditions display name

-grade: Returns the temperature grade of the current device

-model: Returns the operating corner

-name: Returns the operating conditions Tcl_Obj name

-speed: Returns the speed grade of the current device

-temperature: Returns the operating temperature

-voltage: Returns the operating voltage

<operating_condition>: Operating condition object

Description

Returns information about the operating_conditions Tcl object.

Example

```
#see detailed information about operating conditions
foreach_in_collection op [get_available_operating_conditions] {
    puts "Delay Model: [get_operating_conditions_info $op -model]"
}
```

get_partition_info

Usage

```
get_partition_info [-child] [-name] [-parent] [-type] <partition_object>
```

Options

-child: Return child partition name(s)
-name: Return the partition name
-parent: Return parent partition name
-type: Return the partition type
<partition_object>: Partition object

Description

Gets information about the specified partition (referenced by partition ID). Partition ID's can be obtained by Tcl commands such as `get_partitions`.

The `-name`, `-type`, `-parent`, and `-child` options are mutually exclusive.

Example

```
project_open chiptrip
create_timing_netlist
set partitions [get_partitions *]
foreach_in_collection partition $partitions {
    puts "[get_partition_info $partition -name]"
}
delete_timing_netlist
project_close
```

get_path

Usage

```
get_path [-from <names>] [-min_path] [-npaths <number>] [-nworst <number>]
[-pairs_only] [-show_routing] [-through <names>] [-to <names>]
```

Options

- from <names>: Valid sources (string patterns are matched using Tcl string matching)
- min_path: Find the minimum delay path(s)
- npaths <number>: Specifies the number of paths to report. The default value is 1 or the same value as nworst, if nworst is specified
- nworst <number>: Specifies the maximum number of paths to report for each endpoint. If unspecified, there is no limit. If nworst is specified, but npaths is not, npaths defaults to the same value as nworst
- pairs_only: When set, paths with the same start and end points are considered equivalent. Only the longest delay path for each unique combination is displayed.
- show_routing: Option to display detailed routing in the path
- through <names>: Valid through nodes (string patterns are matched using Tcl string matching)
- to <names>: Valid destinations (string patterns are matched using Tcl string matching)

Description

Returns a collection of path objects for the longest delay paths between arbitrary points in the netlist.

This command behaves the same as the report_path command. However, instead of reporting the paths, it returns a Tcl collection of path objects. You can retrieve path object data using the get_path_info and get_point_info commands.

Note that get_path_info does not provide any clock-related information, required points, or meaningful slack values, for paths represented by the path objects returned by this function.

For help on the options shared with report_path, see help for the report_path command.

Example

```
# Define a few helper procedures to print out points
# on a path, and the path itself

proc print_point { point } {
    set total    [ get_point_info $point -total    ]
    set incr     [ get_point_info $point -incr     ]
    set node_id  [ get_point_info $point -node     ]
    set type     [ get_point_info $point -type     ]
    set rf       [ get_point_info $point -rise_fall]
    set node_name ""

    if { $node_id ne "" } {
        set node_name [ get_node_info $node_id -name ]
    }

    puts \
        [format "%10s %8s %2s %-6s %s" $total $incr $rf $type $node_name ]
}

proc print_path { path } {
    puts "Delay      : [ get_path_info $path -arrival_time]"
    puts ""
}
```

```
puts \
[format "%10s %8s %-2s %-6s %s" "Total" "Incr" "RF" "Type" "Name"]
puts \
"=====

foreach_in_collection pt [ get_path_info $path -arrival_points ] {
print_point $pt
}
}

project_open my_project

# Always create the netlist first
create_timing_netlist
read_sdc my_project.sdc
update_timing_netlist

# And now simply iterate over the 10 longest delay paths,
# printing each as we go.
foreach_in_collection path [get_path -nworst 10] {
    print_path $path
    puts ""
}

delete_timing_netlist
project_close
```

get_path_info

Usage

```
get_path_info [-arrival_points] [-arrival_time] [-clock_relationship] [-clock_skew]
[-data_delay] [-from] [-from_clock] [-from_clock_is_inverted] [-hold_end_multicycle]
[-hold_start_multicycle] [-latch_time] [-launch_time] [-num_logic_levels]
[-required_points] [-required_time] [-setup_end_multicycle] [-setup_start_multicycle]
[-slack] [-to] [-to_clock] [-to_clock_is_inverted] [-type] <path_ref>
```

Options

- arrival_points: Return a collection of point objects for the arrival path
- arrival_time: Return the data arrival time for the path
- clock_relationship: Return the clock relationship for the path
- clock_skew: Return the clock skew for the path
- data_delay: Return the data delay for the path
- from: Return the source node ID
- from_clock: Return the source clock ID
- from_clock_is_inverted: Return 1 if the source clock is inverted, 0 otherwise
- hold_end_multicycle: Return the hold end multicycle for the path
- hold_start_multicycle: Return the hold start multicycle for the path
- latch_time: Return the latch time for the path
- launch_time: Return the launch time for the path
- num_logic_levels: Return the number of logic levels on the path between the to node and from node
- required_points: Return a collection of point objects for the required path
- required_time: Return the data required time for the path
- setup_end_multicycle: Return the setup end multicycle for the path
- setup_start_multicycle: Return the setup start multicycle for the path
- slack: Return the slack for the path
- to: Return the destination node ID
- to_clock: Return the destination clock ID
- to_clock_is_inverted: Return 1 if the destination clock is inverted, 0 otherwise
- type: Return the type of this path. Possible return values are: "setup", "hold", "recovery", "removal", "max_path", "min_path"
- <path_ref>: Path object

Description

Get information about the referenced timing path object.

References to path objects can be generated using the `get_timing_paths` and `get_path` functions.

The `-type` option returns one of the following types: "setup", "hold", "recovery", "removal", "max_path", "min_path".

sta

The `-from` and `-to` options return the ID of the nodes at the start and end, respectively, of the arrival path. If there is no node, an empty string is returned. The "to" node remains the same, regardless of the level of clock detail provided (that is, it is always the first node clocked by the "from" clock in the data arrival path). The node ID may be used with the `get_node_info` function to obtain additional information about the node.

The `-from_clock` and `-to_clock` options return the ID of the launching and latching clocks, respectively. If there is no clock, an empty string is returned. Additional information on the clocks can be obtained using the `get_clock_info` function.

The `-arrival_points` and `-required_points` options return a collection of point objects for the arrival and required paths, respectively. By iterating over the collection, and using the `get_point_info` function, the specific details of each portion of the path can be obtained.

If a path was created with additional clock detail, the elements of the clock path will be included in each collection of points.

The values of the `-from`, `-to`, etc. are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See the help for `use_timequest_style_escaping` for details.

Path objects generated by `get_path` do not have clock information, required points, or meaningful slack values.

Example

```
# Define a few helper procedures to print out points
# on a path, and the path itself
proc get_clock_string { path clk } {
    set clk_str ""
    set clk_id [ get_path_info $path -${clk}_clock ]

    if { $clk_id ne "" } {
        set clk_str [ get_clock_info $clk_id -name ]
    }

    if { [ get_path_info $path -${clk}_clock_is_inverted ] } {
        append clk_str " (INVERTED)"
    }
}

return $clk_str
}

proc print_point { point } {
    set total [ get_point_info $point -total ]
    set incr [ get_point_info $point -incr ]
    set node_id [ get_point_info $point -node ]
    set type [ get_point_info $point -type ]
    set rf [ get_point_info $point -rise_fall ]
    set node_name ""

    if { $node_id ne "" } {
        set node_name [ get_node_info $node_id -name ]
    }

    puts \
        [format "%10s %8s %2s %-6s %s" $total $incr $rf $type $node_name ]
}

proc print_path { path } {
    puts "Slack      : [ get_path_info $path -slack]"
    puts "To Clock   : [ get_clock_string $path to ]"
```

```
puts "From Clock : [ get_clock_string $path from]"
puts ""
puts \
  [format "%10s %8s %-2s %-6s %s" "Total" "Incr" "RF" "Type" "Name"]
puts \
  "=====

foreach_in_collection pt [ get_path_info $path -arrival_points ] {
  print_point $pt
}

}

project_open my_project

# Always create the netlist first
create_timing_netlist
read_sdc my_project.sdc
update_timing_netlist

# And now simply iterate over the 10 worst setup paths, printing each
# path
foreach_in_collection path [ get_timing_paths -npaths 10 -setup ] {
  print_path $path
  puts ""
}

delete_timing_netlist
project_close
```


get_pin_info

Usage

```
get_pin_info [-is_clock_pin] [-is_in_pin] [-is_out_pin] [-name] [-net] [-parent_cell]
[-suffix] [-type] <pin_object>
```

Options

-is_clock_pin: Return true if it is a clock pin, or false otherwise
-is_in_pin: Return true if it is an input pin, or false otherwise
-is_out_pin: Return true if it is an output pin, or false otherwise
-name: Return the pin name
-net: Return the net ID if this is an output pin
-parent_cell: Return the parent cell ID
-suffix: Return the suffix of the pin
-type: Return the pin type
<pin_object>: Pin object

Description

Gets information about the specified pin (referenced by pin ID). Pin ID's can be obtained by Tcl commands such as get_pins.

The -type option returns "pin".

Options -name, -type, -parent_cell, -net, -suffix, -is_clk_pin, -is_in_pin and -is_out_pin are mutually exclusive.

Example

```
project_open chiptrip
create_timing_netlist
set pins [get_pins]
foreach_in_collection pin $pins {
    set pin_name [get_pin_info $pin -name]
    set parent_cell [get_pin_info $pin -parent_cell]
    puts "Pin $pin_name belongs to cell [get_cell_info -name \
        $parent_cell]"
}
delete_timing_netlist
project_close
```

get_point_info

Usage

```
get_point_info [-edge] [-incremental_delay] [-location] [-node] [-number_of_fanout]
[-rise_fall] [-total_delay] [-type] <point_ref>
```

Options

-edge: Return the edge ID for the edge associated with this point. If the point has no edge, this returns an empty string

-incremental_delay: Return the incremental delay through this point

-location: Return a string indicating the location of the point's node, if there is one, else an empty string

-node: Return the node ID for the node associated with this point. If the point has no node, this returns an empty string

-number_of_fanout: Return the number of fanout that this point has in the netlist

-rise_fall: Return a string indicating the rise_fall type of this point. Return values are r, f, rr, rf, fr, ff, or an empty string for undefined

-total_delay: Return the total delay of the path at this point. This includes the incremental delay for the point itself

-type: Return a string indicating the type of the point

<point_ref>: Point object

Description

Returns information about the referenced timing point object. References to path objects can be generated using the get_path_info function.

A point object is the equivalent of a row in a path in the output from report_timing.

The -node option returns a node ID for the corresponding node in the path. For points that do not have a corresponding node (such as points for the lumped clock network delay, launch time, latch time, individual routing elements, etc.), the node ID is an empty string. A non-empty node ID can be used in conjunction with the get_node_info function to obtain additional information about the node.

The -edge option returns an edge ID for the corresponding edge in the path. Only points of type "ic", "cell", and "comp" may have edges. For other point types, an empty string will be returned. A non-empty edge ID can be used in conjunction with the get_edge_info function to obtain additional information about the edge.

The -total_delay option returns the total delay along the path, up to and including the current point. The -incremental_delay option returns the delay incurred by going through this point in the path. Both delays are formatted in terms of the current time units, excluding the unit string.

The -number_of_fanout option returns the number of fanouts that the corresponding node has in the timing netlist. If there is no node for this point, the return value is 0.

The -location option returns a string indicating the location of the corresponding node in the part. If there is no corresponding node, this returns an empty string.

The -rise_fall option returns the transition type of this point.

Possible values for -rise_fall are:

Value	Description
(empty)	Unknown transition
r	Rising output
f	Falling output
rr	Rising input, rising output
rf	Rising input, falling output
fr	Falling input, rising output
ff	Falling input, falling output

The -type option returns a string indicating the type of delay that this point represents in the path.

Possible return values for -type are:

Value	Description
cell	Cell delay
clknet	Lumped clock network delay
clksrc	Clock source. Used to ensure that the end-point of a clock segment is marked in the path when source latency is specified, or when the actual path cannot be found.
comp	PLL clock network compensation delay
ic	Interconnect delay
iext	External input delay
latch	Clock latch time
launch	Clock launch time
loop	Lumped combinational loop delay
oext	External output delay
re	Routing element (only for paths generated with the -show_routing option)
srclat	Source latency for a clock segment. This will appear if latency was specified between two clocks, or if a path could not be found between them.
unc	Clock uncertainty
utco	Register micro-Tco time
utsu	Register micro-Tsu time
uth	Register micro-Th time

Example

```
# Define a few helper procedures to print out points
# on a path, and the path itself
proc get_clock_string { path clk } {
    set clk_str ""
    set clk_id [ get_path_info $path -${clk}_clock ]

    if { $clk_id ne "" } {
        set clk_str [ get_clock_info $clk_id -name ]
    }

    if { [ get_path_info $path -${clk}_clock_is_inverted ] } {
        append clk_str " (INVERTED)"
    }
}
```

```

    }
}

return $clk_str
}

proc print_point { point } {
    set total      [ get_point_info $point -total      ]
    set incr       [ get_point_info $point -incr       ]
    set node_id    [ get_point_info $point -node       ]
    set type       [ get_point_info $point -type       ]
    set rf         [ get_point_info $point -rise_fall ]
    set node_name  ""

    if { $node_id ne "" } {
        set node_name [ get_node_info $node_id -name ]
    }

    puts \
        [format "%10s %8s %2s %-6s %s" $total $incr $rf $type $node_name ]
}

proc print_path { path } {
    puts "Slack      : [ get_path_info $path -slack]"
    puts "To Clock   : [ get_clock_string $path to ]"
    puts "From Clock  : [ get_clock_string $path from]"
    puts ""
    puts \
        [format "%10s %8s %-2s %-6s %s" "Total" "Incr" "RF" "Type" "Name"]
    puts \
        "=====

    foreach_in_collection pt [ get_path_info $path -arrival_points ] {
        print_point $pt
    }
}

project_open my_project

# Always create the netlist first
create_timing_netlist
read_sdc my_project.sdc
update_timing_netlist

# And now simply iterate over the 10 worst setup paths, printing each
# path
foreach_in_collection path [ get_timing_paths -npaths 10 -setup ] {
    print_path $path
    puts ""
}

delete_timing_netlist
project_close

```

get_port_info

Usage

```
get_port_info [-edge_rate] [-is_inout_port] [-is_input_port] [-is_output_port] [-name]
[-type] <port_object>
```

Options

-edge_rate: Return the edge_rate value
-is_inout_port: Return true if it is an inout port, or false otherwise
-is_input_port: Return true if it is an input port, or false otherwise
-is_output_port: Return true if it is an output port, or false otherwise
-name: Return the port name
-type: Return the port type
<port_object>: Port object

Description

Returns information about the specified port (referenced by port ID). Port ID's can be obtained by Tcl commands such as get_ports. The -type option returns "port". The -name, -type, -edge_rate, -is_input_port, -is_output_port and is_inout_port options are mutually exclusive.

Example

```
project_open chiptrip
create_timing_netlist
set ports [get_ports]
foreach_in_collection port $ports {
    set port_type ""
    if [get_port_info $port -is_inout_port] {
        set port_type "bidir"
    } elseif [get_port_info $port -is_input_port] {
        set port_type "in"
    } else {
        set port_type "out"
    }
    puts "[get_port_info $port -name]: $port_type"
}
delete_timing_netlist
project_close
```

get_register_info

Usage

```
get_register_info [-asynch_edges] [-clock_edges] [-fanout_edges] [-is_latch] [-name]
[-synch_edges] [-tch] [-tcl] [-tco] [-th] [-tmin] [-tsu] [-type] <reg_object>
```

Options

```
-asynch_edges: Return a list of asynchronous edge IDs
-clock_edges: Return a list of clock edge IDs
-fanout_edges: Return a list of fanout edge IDs
-is_latch: Return "1" if this is a latch node, or "0" otherwise
-name: Return the object name
-synch_edges: Return a list of synchronous edge IDs
-tch: Return the Tch value
-tcl: Return the Tcl value
-tco: Return the Tco value
-th: Return the Th value
-tmin: Return the Tmin value
-tsu: Return the Tsu value
-type: Return the object type
<reg_object>: Register object
```

Description

Gets information about the specified register (referenced by register ID). Register IDs can be obtained by Tcl commands such as `get_registers`.

The `-type` option returns "reg". The `-name`, `-type`, `-tco`, `-tsu`, `-th`, `-tch`, `-tcl`, `-tmin`, `-clock_edges`, `-synch_edges`, `-asynch_edges`, `-fanout_edges` and `-is_latch` options are mutually exclusive.

Example

```
project_open chiptrip
create_timing_netlist
set registers [get_registers]
foreach_in_collection reg $registers {
    set name [get_register_info $reg -name]
    set tco [get_register_info $reg -tco]
    puts "Tco of $name is $tco"
}
delete_timing_netlist
project_close
```

get_timing_paths

Usage

```
get_timing_paths [-detail <SUMMARY|PATH_ONLY|PATH_AND_CLOCK|FULL_PATH>]
[-fall_from_clock <names>] [-fall_to_clock <names>] [-false_path] [-from <names>]
[-from_clock <names>] [-hold] [-less_than_slack <slack limit>] [-npaths <number>]
[-nworst <number>] [-pairs_only] [-recovery] [-removal] [-rise_from_clock <names>]
[-rise_to_clock <names>] [-setup] [-show_routing] [-through <names>] [-to <names>]
[-to_clock <names>]
```

Options

-detail <SUMMARY|PATH_ONLY|PATH_AND_CLOCK|FULL_PATH>: Option to determine how much detail should be shown in the path report

-fall_from_clock <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-fall_to_clock <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-false_path: Report only paths that are cut by a false path assignment

-from <names>: Valid sources (string patterns are matched using Tcl string matching)

-from_clock <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-hold: Option to report clock hold paths

-less_than_slack <slack limit>: Limit the paths reported to those with slack values less than the specified limit.

-npaths <number>: Specifies the number of paths to report (default=1, or the same value as nworst, if nworst is specified)

-nworst <number>: Specifies the maximum number of paths to report for each endpoint. If unspecified, there is no limit. If nworst is specified, but npaths is not, npaths defaults to the same value as nworst

-pairs_only: When set, paths with the same start and end points are considered equivalent. Only the worst case path for each unique combination is displayed.

-recovery: Option to report recovery paths

-removal: Option to report removal paths

-rise_from_clock <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-rise_to_clock <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-setup: Option to report clock setup paths

-show_routing: Option to display detailed routing in the path

-through <names>: Valid through nodes (string patterns are matched using Tcl string matching)

-to <names>: Valid destinations (string patterns are matched using Tcl string matching)

-to_clock <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

Description

Get a collection of path objects for the worst-case paths.

This command behaves the same as the `report_timing` command. However, instead of reporting the paths, it returns a Tcl collection of path objects. You can retrieve path object data using the `get_path_info` and `get_point_info` commands.

For help on the options shared with `report_timing`, see the `report_timing` help page.

Example

```
# Define a few helper procedures to print out points
# on a path, and the path itself
proc get_clock_string { path clk } {
    set clk_str ""
    set clk_id [ get_path_info $path -${clk}_clock ]

    if { $clk_id ne "" } {
        set clk_str [ get_clock_info $clk_id -name ]
    }

    if { [ get_path_info $path -${clk}_clock_is_inverted ] } {
        append clk_str " (INVERTED)"
    }
}

return $clk_str
}

proc print_point { point } {
    set total [ get_point_info $point -total ]
    set incr [ get_point_info $point -incr ]
    set node_id [ get_point_info $point -node ]
    set type [ get_point_info $point -type ]
    set rf [ get_point_info $point -rise_fall ]
    set node_name ""

    if { $node_id ne "" } {
        set node_name [ get_node_info $node_id -name ]
    }

    puts \
        [format "%10s %8s %2s %-6s %s" $total $incr $rf $type $node_name ]
}

proc print_path { path } {
    puts "Slack : [ get_path_info $path -slack]"
    puts "To Clock : [ get_clock_string $path to ]"
    puts "From Clock : [ get_clock_string $path from]"
    puts ""
    puts \
        [format "%10s %8s %-2s %-6s %s" "Total" "Incr" "RF" "Type" "Name"]
    puts \
        "======"

    foreach_in_collection pt [ get_path_info $path -arrival_points ] {
        print_point $pt
    }
}

project_open my_project

# Always create the netlist first
create_timing_netlist
read_sdc my_project.sdc
update_timing_netlist

# And now simply iterate over the 10 worst setup paths, printing each
```



```
# path
foreach_in_collection path [ get_timing_paths -npaths 10 -setup ] {
    print_path $path
    puts ""
}

delete_timing_netlist
project_close
```

locate

Usage

```
locate [-chip] [-color
<black|blue|brown|green|grey|light_grey|orange|purple|red|white>] [-cps] [-label
<label>] [-rpe] [-tmv] <items>
```

Options

-chip: Locate the object in the Chip Planner

-color <black|blue|brown|green|grey|light_grey|orange|purple|red|white>: Specify the color to be used to identify the objects you are locating

-cps: Locate the object in the Critical Path Settings dialog of the Chip Planner

-label <label>: Specify a label used to identify the objects you are locating

-rpe: Locate in the Resource Property Editor

-tmv: Locate the object in the Technology Map Viewer

<items>: Items to locate. Any collection or object (such as paths, points, nodes, nets, keepers, registers, etc) may be located by passing a reference to the corresponding collection or object.

Description

Locate an object from TimeQuest in another Quartus II tool.

With this command, one or more objects, or collections of objects, can be located in a supported Quartus tool from TimeQuest.

The destination can be specified with one of the following options:

Option	Destination Tool
-chip	Chip Planner
-rpe	Resource Property Editor
-tmv	Technology Map Viewer
-cps	Critical Path Settings Dialog of the Chip Planner

The -label option can be used to specify a label for the located objects. The -color command can be used to specify a color to be used to identify the located objects in the destination tool.

Example

```
proc prepare_design { } {
    set sleep_for 2000

    create_timing_netlist -rise_fall

    post_message -type info "Give the GUI some time to catch up to the \
        new netlist. Sleep for $sleep_for ms"
    after $sleep_for

    read_sdc
    update_timing_netlist
}

prepare_design
```

```
# Locate all of the nodes in the longest ten paths
# into the Resource Property Editor
locate [get_path -npaths 10] -rpe

# Locate ten paths into the chip planner, labelling
# each one individually.
set path_col [get_timing_paths -npaths 10]
set path_id 0

foreach_in_collection path $path_col {
    incr path_id

    locate -label "Path #$path_id" $path -chip
}

# locate all keepers that begin with the letter t
# to the Tech Map Viewer
locate [get_keepers t*] -tmv

# locate all nodes that begin with the letter a
#
# The TimeQuest GUI will prompt the user for the
# tool to which the nodes should be located.
#
# Pause first to allow the previous locations to
# appear, as the dialog that pops up, to ask
# the user for a location, will block the rest
# of the GUI until cleared.

after 5000

post_message -type info "Interactive locate"
locate a*
```

query_collection

Usage

```
query_collection [-all] [-limit <limit_value>] [-list_format] [-report_format]  
<collection>
```

Options

```
-all: Return all the collection objects.  
-limit <limit_value>: Set number of collection objects to return.  
-list_format: Return collection objects in a list format.  
-report_format: Return collection objects in a format of one element per line.  
<collection>: Object collection
```

Description

Query collection objects.

Collections can be obtained by Tcl commands such as `get_clocks`, `get_ports`, `get_cells`. If neither the `-limit` nor the `-all` option is specified, then first 20 objects (if the collection has more than 20 objects) or all objects (if the collection has less than or equal to 20 objects) are returned.

Example

```
project_open chiptrip  
create_timing_netlist  
  
set nodes [get_nodes Reg*]  
# Get the first 100 nodes in the collection.  
query_collection $nodes -limit 100  
  
delete_timing_netlist  
project_close
```

read_sdc

Usage

```
read_sdc [-hdl] <file_name>
```

Options

-hdl: Read SDC commands embedded in HDL

<file_name>: Name of the SDC file

Description

Reads an SDC file with all current constraints and exceptions.

If an SDC file is specified, read_sdc only reads that SDC file. If the -hdl option is specified, read_sdc only reads SDC commands that were embedded in HDL.

If no arguments are specified, read_sdc reads the default SDC files along with any SDC commands that were embedded in HDL. If one or more SDC_FILE assignments exists in the QSF, read_sdc reads all of them in order. Otherwise, read_sdc reads the file <revision>.sdc if it exists.

Example

```
project_new test
create_timing_netlist

# Read SDC commands from test_constraints.sdc
read_sdc test_constraints.sdc

# Read SDC commands embedded in HDL
read_sdc -hdl

update_timing_netlist

report_timing

delete_timing_netlist
project_close
```

remove_from_collection

Usage

```
remove_from_collection <base_collection> <items_to_remove>
```

Options

<base_collection>: Collection to remove items from

<items_to_remove>: Items to be removed from base_collection

Description

This command takes two collections and returns a new collection that is the difference of the two, effectively the second collection subtracted from the first collection. The second collection can be a string, but the first has to be previously-created collection: either by passing any of the "get_" functions directly, or by passing a variable that contains a collection (see code examples for this command). If a collection is used for the second argument, the types in the second collection must be the same as or a subset of the types in the first collection.

If the first collection consists of keepers, the second collection can only consist of keepers, registers or ports. If the first collection consists of partitions, the second collection can only consist of partitions or cells. If the first collection consists of nodes, the second collection can only consist of nodes, keepers, registers, ports, pins, nets or combinational nodes.

Example

```
set a_keeprs [get_keepers a*]
set a1_regs [get_registers a1*]

set keeprs_without_a1 [remove_from_collection $a_keeprs $a1_regs]

#or:
set keeprs_without_a1 [remove_from_collection $a_keeprs \
    [get_registers a1*]]

#or even:
set keeprs_without_a1 [remove_from_collection $a_keeprs a1*]

# - note that the last statement will actually remove all keepers with
# name a1*
#   not only registers! (will remove IOs with name a1*, if any)

# Get the first 100 nodes in the collection.
query_collection $keeprs_without_a1 -limit 100
```

report_advanced_io_timing

Usage

```
report_advanced_io_timing [-append] [-file <name>] [-panel_name <name>] [-stdout]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

Description

This command creates a report containing all of the relevant signal integrity measurements computed during I/O buffer simulation.

You must perform delay annotation with Advanced I/O Timing enabled before using this command. This option can be enabled from TimeQuest Timing Analyzer Page of the Settings dialog box.

Example

```
project_open my_project

# Always create the netlist first
create_timing_netlist
read_sdc my_project.sdc
update_timing_netlist

# Create "Advanced I/O Timing" report panel
report_advanced_io_timing -panel_name "Advanced I/O Timing"

# The following command is optional
delete_timing_netlist

project_close
```

report_bottleneck

Usage

```
report_bottleneck [-cmetric <cmetric_name>] [-details] [-metric
<default|tns|num_paths|num_fpaths|num_fanins|num_fanouts>] [-nworst <number>]
[-panel_name <panel_name>] [-stdout] <paths>
```

Options

-cmetric <cmetric_name>: Custom metric function to evaluate individual nodes

-details: Show the detailed information (number of failing edges, number of fanins, etc)

-metric <default|tns|num_paths|num_fpaths|num_fanins|num_fanouts>: Indicate the metric to use to rate individual nodes

-nworst <number>: Specifies the maximum number of nodes to report. If unspecified, there is no limit

-panel_name <panel_name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Output the result onto stdout

<paths>: Paths to be analyzed

Description

Reports bottleneck nodes in a design based on user-specified criteria for rating each node.

The following considerations are pre-defined

- num_fpaths: (default) returns the number of paths that fail timing through the node.
- num_fanins: returns the number of fanin edges from the node.
- num_fanouts: returns the number of fanout edges from the node.
- num_paths: returns the number of paths through the node.
- tns: returns the total negative slack of all the paths through the node.

The paths to be analyzed can be specified by passing the result of any `get_timing_paths` call as the last argument to `report_bottleneck`. If no paths are specified, `report_bottleneck` analyzes the worst 1000 setup paths in the design.

You can also create your own custom criteria for evaluating nodes based on the combination of the number of fanouts, fanins, failing paths, and total paths.

To use custom criteria, do the following:

1. Create a Tcl procedure that takes one argument, "arg", for example.
2. Use "upvar \$arg metric" in the procedure.
3. Calculate the rating based on `$metric(tns)`, `$metric(num_fanouts)`, `$metric(num_fanins)`, and `$metric(num_fpaths)`.
4. Return the rating with "return \$rating".
5. Pass the name of your custom criteria procedure to `report_bottleneck` using the `-cmetric` option.

Reports can be directed to the Tcl console (`-stdout`), the TimeQuest graphical interface (`-panel`), or a combination of the two.

Example

```
project_open my_project
create_timing_netlist
read_sdc
update_timing_netlist

# use the worst 500 hold paths
set paths [ get_timing_paths -npaths 500 -hold ]
report_bottleneck -metric default -panel "Timing Analysis Bottleneck \
  Report - Default Metric" $paths
report_bottleneck -metric tns -panel "Timing Analysis Bottleneck Report - \
  TNS" $paths
report_bottleneck -metric num_paths -panel "Timing Analysis Bottleneck \
  Report - Number of Paths" $paths
report_bottleneck -metric num_fpaths -panel "Timing Analysis Bottleneck \
  Report - Number of Failing Paths" $paths
report_bottleneck -metric num_fanouts -panel "Timing Analysis Bottleneck \
  Report - Number of Fanouts" $paths

# create custom metric and use the worst 2000 setup paths
proc report_bottleneck_custom_metric {arg} {
  # Description: use the number of fanins as the custom metric.
  upvar $arg metric
  set rating $metric(num_fanins)
  return $rating
}

set paths [ get_timing_paths -npaths 2000 -setup ]
report_bottleneck -cmetric report_bottleneck_custom_metric -panel "Timing \
  Analysis Bottleneck Report - Custom" $paths
```

report_clock_fmax_summary

Usage

```
report_clock_fmax_summary [-append] [-file <name>] [-panel_name <name>] [-stdout]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

Description

Reports potential fmax for every clock in the design, regardless of the user-specified clock periods. Fmax is only computed for paths where the source and destination registers or ports are driven by the same clock. Paths of different clocks, including generated clocks, are ignored. For paths between a clock and its inversion, fmax is computed as if the rising and falling edges of the clock are scaled along with fmax, such that the duty cycle (in terms of a percentage) is maintained.

Restricted fmax considers hold timing in addition to setup timing, as well as minimum pulse and minimum period restrictions. Similar to unrestricted fmax, the restricted fmax is computed as if the rising and falling edges of the clock are scaled along with fmax, such that the duty cycle (in terms of a percentage) is maintained. The "Note" column reports which analyses restricted fmax. Refer to hold timing reports (e.g., report_timing with the -hold option) or minimum pulse width reports generated by the report_min_pulse_width command for details of specific paths, registers, or ports.

Example

```
project_open my_project

# Always create the netlist first
create_timing_netlist
read_sdc my_project.sdc
update_timing_netlist

# Output results in the form of messages
report_clock_fmax_summary
# Create "Fmax" report panel
report_clock_fmax_summary -panel_name Fmax
# Report both with report panel and messages
report_clock_fmax_summary -panel_name Fmax -stdout

# The following command is optional
delete_timing_netlist

project_close
```

report_clock_transfers

Usage

```
report_clock_transfers [-append] [-file <name>] [-hold] [-panel_name <name>]  
[-recovery] [-removal] [-setup] [-stdout]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-hold: Creates a clock transfer summary for hold analysis

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-recovery: Creates a clock transfer summary for recovery analysis

-removal: Creates a clock transfer summary for removal analysis

-setup: Creates a clock transfer summary for setup analysis

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

Description

Generates a timing report table showing all clock transfers (i.e., data paths between one clock domain and another clock domain). The from and to clocks are shown as well as the number of paths for each transfer: RR, RF, FR, FF. An RF transfer, for example, occurs when the source register of path is clocked by the rising edge of its clock and the destination register is clocked by the falling edge of its clock.

The report also indicates what clock transfers are cut ("false paths") by set_clock_groups or clock-to-clock set_false_path commands. For transfers that are not cut, the number of paths reported does not take into account paths cut by path-specific set_false_path commands. Actual path counts may be lower than reported.

The report can be directed to the Tcl console ("-stdout", default), a file ("-file"), the TimeQuest graphical user interface ("-panel_name"), or any combination of the three.

The -setup, -hold, -recovery, and -removal options determine the analysis type of the report, particularly the reporting of false_paths that apply to only one analysis type. If you do not specify any of these options, a report is generated for each analysis.

Example

```
project_open top  
create_timing_netlist -skip_dat  
report_clock_transfers -panel_name  
delete_timing_netlist  
project_close
```

report_clocks

Usage

```
report_clocks [-append] [-desc] [-file <name>] [-panel_name <name>] [-stdout]
[-summary] [-waveform]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-desc: Sort the clocks by name in descending order (ascending order is default)

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

-summary: Create a single table with a summary of each clock

-waveform: Display the clocks graphically as waveforms

Description

Report can be directed to the Tcl console ("-stdout", default), a file ("-file"), the TimeQuest graphical interface ("-panel_name"), or any combination of the three.

For the Tcl console, the clock details are reported in two sections. The first sections show all clocks, their period, and their waveform. This includes generated clocks after an `update_timing_netlist`. The second section shows details for all generated clocks. For the timing report, both sections are combined into a single timing report.

Example

```
project_open top
create_timing_netlist
read_sdc
update_timing_netlist

report_clocks

delete_timing_netlist
project_close
```

report_datasheet

Usage

```
report_datasheet [-append] [-expand_bus] [-file <name>] [-panel_name <name>] [-stdout]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-expand_bus: If set, bus is reported as individual ports

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

Description

This function creates a datasheet report which summarizes the timing characteristics of the design as a whole. It reports setup (tsu), hold (th), clock-to-output (tco), minimum clock-to-output (mintco), propagation delay (tpd), and minimum propagation delay (mintpd) times. These delays are reported for each clock or port for which they are relevant. If there is a case where there are multiple paths for a clock (for example if there are multiplexed clocks), then the maximum delay is reported for the tsu, th, tco and tpd, and the minimum delay is reported for mintco and mintpd.

The datasheet can be outputted to the Tcl console ("-stdout", default), a file ("-file"), or a report panel ("-panel_name"). Additionally if the "-file" option is used then the "-append" option can be used to specify that new data should be written to the end of the specified file.

Example

```
project_open proj1
create_timing_netlist
read_sdc
update_timing_netlist

# Report the datasheet to a report panel
report_datasheet -panel_name Datasheet

# Report the datasheet to a file
report_datasheet -file file1.txt
```

report_dds

Usage

```
report_dds [-append] [-file <name>] [-panel_name <name>] [-stdout]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

Description

This command generates custom timing reports for DDR instantiations using the ALTMEMPHY megafunction.

Example

```
project_new test
create_timing_netlist
read_sdc
update_timing_netlist

report_dds -panel "Report DDR"

delete_timing_netlist
project_close
```

report_exceptions

Usage

```
report_exceptions [-append] [-detail  
<SUMMARY|PATH_SUMMARY|PATH_ONLY|PATH_AND_CLOCK|FULL_PATH>] [-fall_from_clock <names>]  
[-fall_to_clock <names>] [-file <name>] [-from <names>] [-from_clock <names>] [-hold]  
[-less_than_slack <slack limit>] [-npaths <number>] [-nworst <number>] [-pairs_only]  
[-panel_name <name>] [-recovery] [-removal] [-rise_from_clock <names>] [-rise_to_clock  
<names>] [-setup] [-stdout] [-through <names>] [-to <names>] [-to_clock <names>]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-detail <SUMMARY|PATH_SUMMARY|PATH_ONLY|PATH_AND_CLOCK|FULL_PATH>: Option to determine how much detail should be shown in the path report

-fall_from_clock <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-fall_to_clock <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-from <names>: Valid sources (string patterns are matched using Tcl string matching)

-from_clock <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-hold: Option to report clock hold paths

-less_than_slack <slack limit>: Limit the paths reported to those with slack values less than the specified limit.

-npaths <number>: Specifies the number of paths to report (default=1, or the same value as nworst, if nworst is specified)

-nworst <number>: Specifies the maximum number of paths to report for each endpoint. If unspecified, there is no limit. If nworst is specified, but npaths is not, npaths defaults to the same value as nworst

-pairs_only: When set, paths with the same start and end points are considered equivalent. Only the worst case path for each unique combination is displayed.

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-recovery: Option to report recovery paths

-removal: Option to report removal paths

-rise_from_clock <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-rise_to_clock <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-setup: Option to report clock setup paths

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

-through <names>: Valid through nodes (string patterns are matched using Tcl string matching)

-to <names>: Valid destinations (string patterns are matched using Tcl string matching)

-to_clock <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

Description

Reports the status and timing analysis results per timing exception.

For each timing exception, it first reports the status: Complete, Partially Overridden, Fully Overridden, or Invalid. The status is relative to the paths and analyses covered by the `-from`, `-to`, and other options. Therefore, a timing exception that reports a status of "Complete" when using the `-from` and `-to` options may not actually be complete with respect to the full design.

Complete: The exception has not been overridden and is valid (i.e., there are paths affected by this exception).

Partially Overridden: The exception includes some paths that have been overridden by one or more higher-precedence exceptions.

Fully Overridden: All paths affected by this exception have been overridden by one or more higher-precedence exceptions.

Invalid: There are no paths applicable covered by this exception. This occurs when a timing exception has valid `-from`, `-to`, or `-through` collections and there are no actual paths from the `-from` nodes to the `-to` nodes.

Use the `-setup` (default), `-hold`, `-recovery`, or `-removal` options to specify which kind of analysis should be performed.

The report can be directed to the Tcl console using `-stdout` (default), a file using `-file`, the TimeQuest graphical user interface using `-panel_name`, or any combination of those three options.

You can limit the analysis performed by this command to specific start and end points, using the `-from` and `-to` options. The analysis can be further limited to clocks using the `-from_clock` and `-to_clock` options, or to specific edges of the clock using the `-rise_from_clock`, `-fall_from_clock`, `-rise_to_clock`, and `-fall_to_clock` options. Additionally, the `-through` option can be used to restrict analysis to paths which go through specified pins or nets.

To determine which timing exceptions override another timing exception when the status is Partially Overridden or Fully Overridden, use the same `-from` and `-to` options that were used with the timing exception.

Use the `-npaths` option to limit the number of paths to report per timing exception. If you do not specify this option, only the single worst-case path per timing exception is provided. Use the `-less_than_slack` option to limit output to all paths with slack less than the specified value, up to the number specified by `-npaths`.

Use the `-nworst` option to limit the number of paths reported for each unique endpoint. If you do not specify this option, the number of paths reported for each destination node is bounded only by the `-npaths` option. If this option is used, but `-npaths` is not specified, then `-npaths` will default to the same value specified for `-nworst`.

Use the `-detail` option to specify the desired level of report detail. Specifying "summary" generates a single table listing only the highlights of each timing exception (status and worst-case slack). "path_summary" generates a table per timing exception listing only the highlights of each path. "path_only" reports the path from the source to the destination without any detail about the clock path. Instead, the clock network delay is shown as a single number. This is the default behavior. "path_and_clock" extends the arrival and required paths back to the launch and latch clocks. "full_path" will continue tracing back through generated clocks to the underlying base clock.

Use the `-pairs_only` option to filter the output further, restricting the results to only unique combinations of start and end points. This filtering is performed after the number of paths has been generated in accordance to the `-npaths` option. As a result, there may be fewer paths displayed than specified by `-npaths`, if a particular set of start and end points appeared multiple times.

False path exceptions (set_false_path) report paths as if the false path was not applied, similar to report_timing's -false_path option.

The "RF" column in the report output uses a two-letter symbol to indicate the rise/fall transition that occurs at that point in the path.

Possible "RF" values are:

Value	Description
(empty)	Unknown transition
R	Rising output
F	Falling output
RR	Rising input, rising output
RF	Rising input, falling output
FR	Falling input, rising output
FF	Falling input, falling output

The "Type" column in the report uses a symbol to indicate what type of delay occurs at that point in the path.

Possible "Type" values are:

Value	Description
CELL	Cell delay
COMP	PLL clock network compensation delay
IC	Interconnect delay
iExt	External input delay
LOOP	Lumped combinational loop delay
oExt	External output delay
RE	Routing element (only for paths generated with the -show_routing option)
uTco	Register micro-Tco time
uTsu	Register micro-Tsu time
uTh	Register micro-Th time

The values of the "-from", "-to", and "-through" options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See the help for use_timequest_style_escaping for details.

Example

```
# Reports all timing exceptions for a setup analysis.
report_exceptions

# Reports all timing exceptions for a hold analysis.
report_exceptions -hold

# Reports all timing exceptions affecting input paths for
```

```
# recovery analysis, reporting the 10 worst paths per exception.  
report_exceptions -from [all_inputs] -to [all_registers] \  
  -recovery -npaths 10 -detail full_path
```

report_max_skew

Usage

```
report_max_skew [-append] [-detail <SUMMARY|PATH_ONLY|PATH_AND_CLOCK|FULL_PATH>] [-file  
<name>] [-less_than_slack <slack limit>] [-npaths <number>] [-panel_name <name>]  
[-show_routing] [-stdout]
```

Options

`-append`: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

`-detail <SUMMARY|PATH_ONLY|PATH_AND_CLOCK|FULL_PATH>`: Option to determine how much detail should be shown in the path report

`-file <name>`: Sends the results to an ASCII or HTML file. Depending on the extension

`-less_than_slack <slack limit>`: Limit the paths reported to those with slack values less than the specified limit.

`-npaths <number>`: Specifies the number of paths to report for each latest and earliest arrival skew result per `set_max_skew` assignment (default=1)

`-panel_name <name>`: Sends the results to the panel and specifies the name of the new panel

`-show_routing`: Option to display detailed routing in the path report

`-stdout`: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

Description

Reports max skew analysis results for all `set_max_skew` commands in a single report. For each valid `set_max_skew` constraint, this command computes skew with respect to the latest and the earliest arrival of each path.

Skew for the Latest Arrival is computed by comparing the latest arrival of each path with the earliest arrival of the path that has the smallest value for early arrival of all other paths included in the constraint. Similarly, "Skew for the Earliest Arrival" is computed by comparing the earliest arrival of each path with the latest arrival of the path that has the largest value for late arrival of all other paths included in the constraint. No path is compared with itself.

Use the `-stdout` option to direct the report to the Tcl console (default), the `-file` option to write the report to a file or the `-panel_name` option to direct the report to the TimeQuest graphical user interface. You can use these options in any combination.

Use the `-npaths` option to limit the number of path result pairs reported for each `set_max_skew` constraint. If you do not specify this option, `report_max_skew` only reports the result pair for the single worst-case path. Use the `-less_than_slack` option to limit output to all paths with slack less than the specified value, up to the number specified with `-npaths`.

Use the `-detail` option to specify the desired level of report detail. The `-detail summary` option generates a single table listing only the highlights of each path (and is the same as `-summary` option, which this replaces. `-detail path_only` (default) reports the path from the source to the destination without any detail about the clock path. Instead, the clock network delay is shown as a single number. `-detail path_and_clock` extends the arrival and required paths back to the launch and latch clocks. `-detail full_path` continues tracing back through generated clocks to the underlying base clock.

The `-show_routing` option displays detailed routing information in the path. Lines marked "IC" without the option are shown, but only as a placeholder. The routing elements for that line are broken out individually and listed before the line.

The return value of this command is a two-element list. The first number is the number of paths found in the analysis. The second is the worst-case slack, in terms of the current default time unit.

The "RF" column in the report output uses a two-letter symbol to indicate the rise/fall transition that occurs at that point in the path.

Possible "RF" values are:

Value	Description
(empty)	Unknown transition
R	Rising output
F	Falling output
RR	Rising input, rising output
RF	Rising input, falling output
FR	Falling input, rising output
FF	Falling input, falling output

The "Type" column in the report uses a symbol to indicate what type of delay occurs at that point in the path.

Possible "Type" values are:

Value	Description
CELL	Cell delay
COMP	PLL clock network compensation delay
IC	Interconnect delay
iExt	External input delay
LOOP	Lumped combinational loop delay
oExt	External output delay
RE	Routing element (only for paths generated with the -show_routing option)
uTco	Register micro-Tco time
uTsu	Register micro-Tsu time
uTh	Register micro-Th time

Example

```
project_open my_project
create_timing_netlist
read_sdc
update_timing_netlist

# create max skew constraints
set_max_skew -from [get_ports data_ports[*]] -to [get_keepers *] 0.200
set_max_skew -from [get_keepers *] -to [get_ports output_ports[*]] 0.100

# show worst 10 paths for each earliest and latest arrival results
# per max_skew assignment assuming that their slack is less than 0.100
report_max_skew -panel_name "Report Max Skew" -npaths 10 -less_than_slack \
```

```
0.100 -detail full_path  
delete_timing_netlist  
project_close
```

report_metastability

Usage

```
report_metastability [-append] [-file <name>] [-panel_name <name>] [-stdout]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

Description

Report can be directed to the Tcl console ("-stdout", default), a file ("-file"), the TimeQuest graphical interface ("-panel_name"), or any combination of the three.

The report_metastability function can be used to estimate the robustness of asynchronous transfers in your design.

Background

Synchronization register chains should be used when transferring data between unrelated clock domains to greatly reduce the probability of the captured data signal becoming metastable. A synchronization register chain is a sequence of registers with the same clock, that is driven by a pin, or logic from an unrelated clock domain. The output of all but the last register in the chain must connect only to the next register, either directly or indirectly through logic.

When a register is metastable, its output hovers at a voltage between high and low for a length of time beyond the normal Tco for the register. The design can fail if subsequent registers that use this metastable signal latch different values. Therefore, it is important to properly synchronize data signals to prevent such occurrences.

Output

The report_metastability function generates a list of synchronization register chains found in the design, and can provide estimates of the Mean Time Between Failures (MTBF) of each chain. The design MTBF is an estimate of the overall robustness of the design, computed from the MTBF results from all synchronization chains with calculated MTBFs. The design MTBF metric is reported only when the design meets timing. Therefore, it is important to fully timing constrain your design.

The typical MTBF result assumes typical silicon characteristics for the selected device speed grade, with nominal operating conditions.

The worst case MTBF result uses the worst case silicon characteristics for the selected device speed grade, with worst case operating conditions.

Settings

To get a list of possible synchronization chains, set "Synchronizer Identification" to AUTO in the TimeQuest Timing Analyzer Page in the Settings dialog box. This will set the "Synchronizer Identification" QSF assignment in your QSF file. TimeQuest will use timing constraints to automatically detect synchronization chains in the design. Metastability analysis checks for signal transfers between circuitry in unrelated or asynchronous clock domains, so clock domains must be related correctly with the timing constraints.

Set the maximum number of registers to consider as part of one synchronization chain, via the "Synchronization Register Chain Length" setting under Analysis and Synthesis Page in the Settings dialog box. The default length is 2. All the registers in a chain (up to this length) will be protected from optimizations that can decrease MTBF.

Note that if you change the "Synchronizer Identification" setting, you should rerun the Fitter, as this setting can impact some optimization algorithms.

Report Panels

The MTBF Summary report provides the estimated mean time between failure for the design. This is an estimate for the overall robustness of the design in terms of metastability, and it is computed from all available synchronization chain MTBFs present in the design.

The MTBF metric of automatically identified synchronization chains is not computed. To compute the MTBF of a synchronization chain, set "Synchronizer Identification" to "Forced If Asynchronous" or "Forced" for all registers of the synchronization chain. By explicitly specifying that this synchronization chain is valid, this chain will then be optimized during the Fitter, and its MTBF will be computed. Its MTBF will then be included in the computation of the design MTBF.

The Synchronizer Summary table lists all the synchronization chains found in your design. It is possible that the analysis performed might erroneously interpret certain structures, such as shift registers, as synchronization chains. If some synchronization chains are misidentified and you wish to remove them from the report, you can turn off analysis of these paths by making node-based assignments via the Assignment Editor, set "Synchronizer Identification" to "Off" for the first register in these synchronization chains. Conversely, if there are synchronization chains in your design that were not detected, you can set "Synchronizer Identification" assignment to "Forced If Asynchronous" for all registers in this chain through the Assignment Editor, and this chain will be reported if it meets the criteria for being a synchronization chain. This can often occur if there is logic present between the registers of the synchronization chain. In the automatic mode of synchronizer identification, these structures are not considered to be synchronizers. If you want to force a register to be identified as the head of a synchronizer, set the "Synchronizer Identification" assignment to "Forced" to the register, and it will always be identified as the first of a synchronization chain. This setting should not be applied to the entire design, since this will identify every register in the design as a synchronizer.

The MTBF estimates assume the data being synchronized is switching at a toggle rate of 12.5% of the source clock frequency. That is, the estimates assume that the arriving data signal switches once every 8 source clock cycles. If multiple clocks apply, the highest frequency is used. If no source clocks can be determined, then the data rate is taken as 12.5% of the synchronization clock frequency.

If you know the approximate rate at which the data changes, and would like to obtain a more accurate MTBF, use the "Synchronizer Toggle Rate" assignment in the Assignment Editor. Set the data toggle rate, in number of transitions per second, on the first register of a synchronization chain. TimeQuest will then take the specified rate into account when computing the MTBF of that particular chain. You can also apply this assignment to an entity or the entire design. Since a "Synchronizer Toggle Rate" assignment of 0 indicates that the data signal never toggles, the affected synchronization chain will not be reported since it does not affect the reliability of the design.

Please refer to the Metastability Optimization section in the Timing Optimization Advisor for further information.

Example

```
project_open top
create_timing_netlist
read_sdc
update_timing_netlist

report_metastability

delete_timing_netlist
project_close
```


report_min_pulse_width

Usage

```
report_min_pulse_width [-append] [-detail <SUMMARY|FULL_PATH>] [-file <name>] [-nworst <number>] [-panel_name <name>] [-stdout] <targets>
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-detail <SUMMARY|FULL_PATH>: Option to determine how much detail should be shown in the report

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-nworst <number>: Specifies the number of pulse width checks to report (default=1)

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

<targets>: Registers or ports

Description

Reports the results of minimum pulse width and minimum period checks.

A minimum pulse width check verifies that a clock high ("High") or low ("Low") pulse sustains long enough to qualify as a recognizable change in the clock signal at a register clock pin. A failed minimum pulse width check indicates that the register may not recognize the clock transition. Each register in the design is reported twice per clock for minimum pulse width checks: once for the high pulse and once for the low pulse.

A minimum period check verifies that the clock period ("Period") is large enough for the device to operate. Minimum period checks apply to I/O edge rate limits for clock ports and minimum period restrictions for RAM and DSP registers. Output clock ports (e.g., source synchronous clocks) require generated clocks in order to check I/O edge rate limits for those those ports.

The results of the minimum pulse width checks can be output to the Tcl console ("-stdout," the default), a report panel ("-panel"), a file ("-file"), or a combination of the three.

Results are sorted from worst-case slack to best-case slack. To limit the number of checks reported, use the "-nworst" option.

Results can be shown in summary ("-detail summary") or in detail ("-detail full_path," the default), showing the path details for the clock arrival times and how they affect the actual pulse width.

The value of the targets is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See the help for use_timequest_style_escaping for details.

Example

```
project_open top
create_timing_netlist
read_sdc
update_timing_netlist

# Report the worst 100 minimum pulse width checks
report_min_pulse_width -nworst 100
```

```
# Report minimum pulse width checks for the register test_reg[*]
report_min_pulse_width test_reg[*]

# Output the previous results to a report panel and a file.
report_min_pulse_width -panel_name "Min Pulse (test_reg)" test_reg[*]

# Output the previous results to a file.
report_min_pulse_width -file min_pulse_test_reg.txt test_reg[*]
```

report_net_delay

Usage

```
report_net_delay [-append] [-file <name>] [-nworst <number>] [-panel_name <name>]
[-stdout]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-nworst <number>: Specifies the maximum number of paths to report for each analysis. If unspecified, there is no limit.

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

Description

Reports net delay analysis results based on set_net_delay commands. Each set_net_delay command is treated as a separate analysis and report_net_delay reports the results of all set_net_delay commands in a single report.

The report contains each set_net_delay command with the worst case slack result followed by the results of each edge matching the criteria set by that set_net_delay command. These results are ordered based on the slack value.

Use -nworst option to limit the number of lines reported for a set_net_delay command.

Example

```
project_open my_project
create_timing_netlist
read_sdc
update_timing_netlist

set_net_delay -min 0.160 -from [get_pins inst9|combout] -to \
[get_pins *|dataf]
set_net_delay -max 0.500 -from inst8|combout

report_net_delay -panel "Net Delay"
```

report_net_timing

Usage

```
report_net_timing [-append] [-file <name>] [-nworst_delay <number>] [-nworst_fanout
<number>] [-panel_name <name>] [-stdout] <name>
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-nworst_delay <number>: Report worst net delays

-nworst_fanout <number>: Report worst fanout nets

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

<name>: Signal or collection name

Description

Reports delay and fanout information about a net in the design. A net corresponds to a cell output pin.

Report can be directed to the Tcl console ("-stdout", default), a file ("-file"), the TimeQuest graphical interface ("-panel_name"), or any combination of the three.

The value of the name is either a collection or a Tcl list of wildcards used to create a collection of the appropriate type. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See the help for use_timequest_style_escaping for details.

Example

```
project_open <design>
create_timing_netlist

# Show delay and fanout information for all nets
# that match "abc*"
report_net_timing [get_nets abc*]

# Report delay and fanout information for the 10
# nets showing higher delays
report_net_timing -nworst_delay 10

# Report delay and fanout information for the 10
# nets showing higher fanout
report_net_timing -nworst_fanout 10

project_close
```

report_partitions

Usage

```
report_partitions [-nworst <number>] [-panel_name <name>] [-stdout]
```

Options

-nworst <number>: Specifies the maximum number of paths to report between partitions. If unspecified, the limit defaults to 1000

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Output the result onto stdout

Description

Reports timing information related to design partitions.

The report_partitions command analyzes the worst 1000 setup paths in the design by default, but you can optionally set the nworst option to increase or decrease this number.

This function reports the total number of failing paths for each partition and the worst-case slack for any path involving the partition in a Partition Timing Overview table.

The function also creates a Partition Timing Details table that lists the number of failing paths and worst-case slack from each partition to the others, which provides a more detailed breakdown of where the critical paths in the design are with respect to design partitions.

Reports can be directed to the Tcl console (-stdout, by default) or the TimeQuest graphical interface (-panel_name <name>) or both.

Example

```
project_open my_project
create_timing_netlist
read_sdc
update_timing_netlist

# Report a maximum of 500 failing paths between partitions to the
# TimeQuest graphical interface and to the Tcl console.
report_partitions -panel_name "Partition Timing Report" -nworst 500 \
  -stdout
```

report_path

Usage

```
report_path [-append] [-file <name>] [-from <names>] [-min_path] [-npaths <number>]
[-nworst <number>] [-pairs_only] [-panel_name <name>] [-show_routing] [-stdout]
[-summary] [-through <names>] [-to <names>]
```

Options

- append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten
- file <name>: Sends the results to an ASCII or HTML file. Depending on the extension
- from <names>: Valid sources (string patterns are matched using Tcl string matching)
- min_path: Find the minimum delay path(s)
- npaths <number>: Specifies the number of paths to report (default=1, or the same value as nworst, if nworst is specified)
- nworst <number>: Specifies the maximum number of paths to report for each endpoint. If unspecified, there is no limit. If nworst is specified, but npaths is not, npaths defaults to the same as nworst
- pairs_only: When set, paths with the same start and end points will be considered to be equivalent. Only the longest delay path for each unique combination will be displayed.
- panel_name <name>: Sends the results to the panel and specifies the name of the new panel
- show_routing: Option to display detailed routing in the path
- stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.
- summary: Create a single table with a summary of each path found
- through <names>: Valid through nodes (string patterns are matched using Tcl string matching)
- to <names>: Valid destinations (string patterns are matched using Tcl string matching)

Description

Reports the longest delay paths and the corresponding delay value.

The report can be directed to the Tcl console ("-stdout", default), a file ("-file"), the TimeQuest graphical user interface ("-panel_name"), or any combination of the three.

You can limit the analysis performed by this command to specific start and end points, using the "-from" and "-to" options. Any node or cell in the design is considered a valid endpoint. Additionally, the "-through" option can be used to restrict analysis to paths which go through specified pins or nets. Paths that are reported can not start before or go beyond a keeper node (register or port); this restriction considers register pins as combinational nodes in the design.

Use "-npaths" to limit the number of paths to report. If this option is not specified, only the single longest delay path is provided.

Use "-nworst" to limit the number of paths reported for each unique endpoint. If you do not specify this option, the number of paths reported for each destination node is bounded only by the "-npaths" option. If this option is used, but "-npaths" is not specified, then "-npaths" will default to the same value specified for "-nworst".

Use the "-pairs_only" option to filter the output further, restricting the results to only unique combinations of start and end points. This filtering is performed after the number of paths has been generated in accordance with the "-npaths" option. As a result, there may be fewer paths displayed than specified by "-npaths", if a particular set of start and end points appeared multiple times.

Use the "-summary" option to generate a single table listing only the highlights of each path.

The "-min_path" option will find the minimum delay path(s) rather than the maximum delay paths which is the default behavior.

The "-show_routing" option will display detailed routing information in the path. Lines that were marked as "IC" without the option will still be shown, but only as a placeholder. The routing elements for that line will be broken out individually and listed before the line.

The return value of this command is a two-element list. The first number is the number of paths found in the analysis. The second is the longest delay, in terms of the current default time unit.

The values of the "-from", "-to", "-through" options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See the help for use_timequest_style_escaping for details.

Example

```
project_open my_project

# Always create the netlist first
create_timing_netlist
read_sdc my_project.sdc
update_timing_netlist

# Report path delay between nodes "foo" and "bar",
# reporting the longest delay if a path is found.

set my_list [report_path -from foo -to bar]
set num_paths [lindex $my_list 0]
set longest_delay [lindex $my_list 1]
if { $num_paths > 0 } {
    puts "Longest delay -from foo -to bar is $longest_delay"
}

# The following command is optional
delete_timing_netlist

project_close
```

report_rskm

Usage

```
report_rskm [-append] [-file <name>] [-panel_name <name>] [-stdout]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

Description

Reports RSKM for dedicated LVDS circuitry.

In designs that use dedicated LVDS circuitry, receiver input skew margin (RSKM) is the time margin available before the LVDS receiver megafunction fails to operate. RSKM is defined as the total time margin that remains after subtracting the sampling window (SW) size and the receiver channel-to-channel skew (RCCS) from the time unit interval (TUI), as expressed in the following formula:

$$RSKM = (TUI - SW - RCCS) / 2$$

The time unit interval is the LVDS clock period (1/fmax). The sampling window is the period of time that the input data must be stable to ensure that the data is successfully sampled by the LVDS receiver megafunction. The sampling window size varies by device speed grade. RCCS is the difference between the fastest and slowest data output transitions, including the tco variation and clock skew. To obtain an accurate analysis of an LVDS circuit, you should assign an appropriate input delay to the LVDS receiver megafunction. RCCS is equal to the difference between maximum input delay and minimum input delay. If no input delay is set, RCCS defaults to zero.

Example

```
project_open top
create_timing_netlist
read_sdc
update_timing_netlist

# Ensure a tccs of 1ns
set_input_delay -max -clock lvds_clk 2ns [get_ports lvds_input]
set_input_delay -min -clock lvds_clk 1ns [get_ports lvds_input]

# Show lvds information
report_rskm
```


report_sdc

Usage

```
report_sdc [-append] [-file <name>] [-ignored] [-panel_name <name>] [-stdout]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-ignored: Reports full history of assignments to locate ignored ones

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

Description

Reports all SDC constraints used in the design. Use the -ignored option to report SDC constraints that were ignored and the reason they were ignored.

Example

```
project_new test
create_timing_netlist
create_clock -period 10 -name clk10 clk
set_multicycle_path -from [get_cells a] -to [get_cells b]
update_timing_netlist

report_sdc -panel_name sdc_report_panel

report_timing

delete_timing_netlist
project_close
```

report_skew

Usage

```
report_skew [-append] [-detail <SUMMARY|PATH_ONLY|PATH_AND_CLOCK|FULL_PATH>] [-exclude
<Tcl list>] [-fall_from_clock <names>] [-fall_to_clock <names>] [-file <name>] [-from
<names>] [-from_clock <names>] [-greater_than_skew <slack limit>] [-include <Tcl list>]
[-npaths <number>] [-panel_name <name>] [-rise_from_clock <names>] [-rise_to_clock
<names>] [-show_routing] [-stdout] [-through <names>] [-to <names>] [-to_clock <names>]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-detail <SUMMARY|PATH_ONLY|PATH_AND_CLOCK|FULL_PATH>: Option to determine how much detail should be shown in the path report

-exclude <Tcl list>: A Tcl list of parameters to exclude during skew analysis. This list can include 1 or more of the following: utsu, uth, utco, from_clock, to_clock, clock_uncertainty, input_delay, output_delay

-fall_from_clock <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-fall_to_clock <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-from <names>: Valid sources (string patterns are matched using Tcl string matching)

-from_clock <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-greater_than_skew <slack limit>: Limit the paths reported to those with skew values greater than the specified limit.

-include <Tcl list>: Tcl list of parameters to include during skew analysis. This list can include 1 or more of the following: utsu, uth, utco, from_clock, to_clock, clock_uncertainty, input_delay, output_delay

-npaths <number>: Specifies the number of paths to report for each latest and earliest arrival skew result (default=1)

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-rise_from_clock <names>: Valid source clocks (string patterns are matched using Tcl string matching)

-rise_to_clock <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

-show_routing: Option to display detailed routing in the path report

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

-through <names>: Valid through nodes (string patterns are matched using Tcl string matching)

-to <names>: Valid destinations (string patterns are matched using Tcl string matching)

-to_clock <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

Description

This report performs skew analysis on paths selected by the user. As opposed to report generated by `report_max_skew` command, this command does not depend on the existence of `set_max_skew` assignments.

For each valid `set_max_skew` constraint, this report computes skew with respect to the latest and the earliest arrival of each path.

Skew for the Latest Arrival is computed by comparing the latest arrival of each path with the earliest arrival of the path that has the smallest value for early arrival of all other paths included in the constraint. Similarly, Skew for the Earliest Arrival is computed by comparing the earliest arrival of each path with the latest arrival of the path that has the largest value for late arrival of all other paths included in the constraint. No path is compared with itself.

Use the `-stdout` option to direct the report to the Tcl console (default), the `-file` option to write the report to a file or the `-panel_name` option to direct the report to the TimeQuest graphical user interface. You can use these options in any combination.

Use the `-include` and `-exclude` options to include or exclude one or more of the following: register micro parameters (`utsu`, `uth`, `utco`), clock arrival times (`from_clock`, `to_clock`), clock uncertainty (`clock_uncertainty`) and input and output delays (`input_delay`, `output_delay`). By default, max skew analysis includes data arrival times, clock arrival times, register micro parameters and clock uncertainty. When `-include` is used, those in the inclusion list will be added to the default analysis. Similarly, when `-exclude` is used, those in the exclusion list will be excluded from the default analysis. When both the `-include` and `-exclude` options specify the same parameter, that parameter will be excluded.

Use the `-npaths` option to limit the number of path result pairs reported for each `set_max_skew` constraint. If you do not specify this option, `report_skew` only reports the result pair for the single worst-case path. Use the `-less_than_slack` option to limit output to all paths with skew greater than the specified value, up to the number specified with `-npaths`.

Use the `-detail` option to specify the desired level of report detail. `"-detail summary"` generates a single table listing only the highlights of each path (and is the same as `-summary` option, which this replaces. `"-detail path_only"` (default) reports the path from the source to the destination without any detail about the clock path. Instead, the clock network delay is shown as a single number. `"-detail path_and_clock"` extends the arrival and required paths back to the launch and latch clocks. `"-detail full_path"` continues tracing back through generated clocks to the underlying base clock.

The `-show_routing` option displays detailed routing information in the path. Lines marked "IC" without the option are shown, but only as a placeholder. The routing elements for that line are broken out individually and listed before the line.

The return value of this command is a two-element list. The first number is the number of paths found in the analysis. The second is the worst-case skew, in terms of the current default time unit.

The "RF" column in the report output uses a two-letter symbol to indicate the rise/fall transition that occurs at that point in the path.

Possible "RF" values are:

Value	Description
(empty)	Unknown transition
R	Rising output
F	Falling output

Value	Description
RR	Rising input, rising output
RF	Rising input, falling output
FR	Falling input, rising output
FF	Falling input, falling output

The "Type" column in the report uses a symbol to indicate what type of delay occurs at that point in the path.

Possible "Type" values are:

Value	Description
CELL	Cell delay
COMP	PLL clock network compensation delay
IC	Interconnect delay
iExt	External input delay
LOOP	Lumped combinational loop delay
oExt	External output delay
RE	Routing element (only for paths generated with the -show_routing option)
uTco	Register micro-Tco time
uTsu	Register micro-Tsu time
uTh	Register micro-Th time

Example

```
project_open my_project
create_timing_netlist
read_sdc
update_timing_netlist

# show worst 10 paths for each earliest and latest arrival results
report_skew -from [get_ports input[*]] -to [get_registers *] -panel_name \
  "Report Skew" -npaths 10 -greater_than_skew 0.100 -detail full_path

delete_timing_netlist
project_close
```

report_tccs

Usage

```
report_tccs [-append] [-file <name>] [-panel_name <name>] [-stdout]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

Description

Reports TCCS for dedicated LVDS transmitters.

In designs that implement the LVDS I/O standard, transmitter channel-to-channel skew (TCCS) is the timing difference between the fastest and slowest output transitions, including tco variations and clock skew.

Example

```
project_open top
create_timing_netlist
read_sdc
update_timing_netlist

# Show lvds information
report_tccs
```

report_timing

Usage

```
report_timing [-append] [-detail <SUMMARY|PATH_ONLY|PATH_AND_CLOCK|FULL_PATH>]
[-fall_from_clock <names>] [-fall_to_clock <names>] [-false_path] [-file <name>] [-from
<names>] [-from_clock <names>] [-hold] [-less_than_slack <slack limit>] [-npaths
<number>] [-nworst <number>] [-pairs_only] [-panel_name <name>] [-recovery] [-removal]
[-rise_from_clock <names>] [-rise_to_clock <names>] [-setup] [-show_routing] [-stdout]
[-through <names>] [-to <names>] [-to_clock <names>]
```

Options

- append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten
- detail <SUMMARY|PATH_ONLY|PATH_AND_CLOCK|FULL_PATH>: Option to determine how much detail should be shown in the path report
- fall_from_clock <names>: Valid source clocks (string patterns are matched using Tcl string matching)
- fall_to_clock <names>: Valid destination clocks (string patterns are matched using Tcl string matching)
- false_path: Report only paths that are cut by a false path assignment
- file <name>: Sends the results to an ASCII or HTML file. Depending on the extension
- from <names>: Valid sources (string patterns are matched using Tcl string matching)
- from_clock <names>: Valid source clocks (string patterns are matched using Tcl string matching)
- hold: Option to report clock hold paths
- less_than_slack <slack limit>: Limit the paths reported to those with slack values less than the specified limit.
- npaths <number>: Specifies the number of paths to report (default=1, or the same value as nworst, if nworst is specified)
- nworst <number>: Specifies the maximum number of paths to report for each endpoint. If unspecified, there is no limit. If nworst is specified, but npaths is not, npaths defaults to the same value as nworst
- pairs_only: When set, paths with the same start and end points are considered equivalent. Only the worst case path for each unique combination is displayed.
- panel_name <name>: Sends the results to the panel and specifies the name of the new panel
- recovery: Option to report recovery paths
- removal: Option to report removal paths
- rise_from_clock <names>: Valid source clocks (string patterns are matched using Tcl string matching)
- rise_to_clock <names>: Valid destination clocks (string patterns are matched using Tcl string matching)
- setup: Option to report clock setup paths
- show_routing: Option to display detailed routing in the path
- stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.
- through <names>: Valid through nodes (string patterns are matched using Tcl string matching)

sta

-to <names>: Valid destinations (string patterns are matched using Tcl string matching)
-to_clock <names>: Valid destination clocks (string patterns are matched using Tcl string matching)

Description

Reports the worst-case paths and associated slack.

Use the "-setup", "-hold", "-recovery", or "-removal" options to specify which kind of analysis should be performed.

The report can be directed to the Tcl console ("-stdout", default), a file ("-file"), the TimeQuest graphical user interface ("-panel_name"), or any combination of the three.

You can limit the analysis performed by this command to specific start and end points, using the "-from" and "-to" options. The analysis can be further limited to clocks using the "-from_clock" and "-to_clock" options, or to specific edges of the clock using the "-rise_from_clock", "-fall_from_clock", "-rise_to_clock", and "-fall_to_clock" options. Additionally, the "-through" option can be used to restrict analysis to paths which go through specified pins or nets.

Use "-npaths" to limit the number of paths to report. If you do not specify this option, only the single worst-case path is provided. Use the "-less_than_slack" option to limit output to all paths with slack less than the specified value, up to the number specified by "-npaths".

Use "-nworst" to limit the number of paths reported for each unique endpoint. If you do not specify this option, the number of paths reported for each destination node is bounded only by the "-npaths" option. If this option is used, but "-npaths" is not specified, then "-npaths" will default to the same value specified for "-nworst".

Use the "-detail" option to specify the desired level of report detail. "summary" generates a single table listing only the highlights of each path (and is the same as "-summary" option, which this replaces). "path_only" reports the path from the source to the destination without any detail about the clock path. Instead, the clock network delay is shown as a single number. This is the default behavior. "path_and_clock" extends the arrival and required paths back to the launch and latch clocks. "full_path" will continue tracing back through generated clocks to the underlying base clock.

Use the "-pairs_only" option to filter the output further, restricting the results to only unique combinations of start and end points. This filtering is performed after the number of paths has been generated in accordance to the "-npaths" option. As a result, there may be fewer paths displayed than specified by "-npaths", if a particular set of start and end points appeared multiple times.

The "-show_routing" option displays detailed routing information in the path. Lines that were marked as "IC" without the option are still shown, but only as a placeholder. The routing elements for that line are broken out individually and listed before the line.

The "-false_path" option reports only those paths that are normally hidden by false_path assignments or clock to clock cuts. Like the default report, this option only reports constrained paths.

The return value of this command is a two-element list. The first number is the number of paths found in the analysis. The second is the worst-case slack, in terms of the current default time unit.

The "RF" column in the report output uses a two-letter symbol to indicate the rise/fall transition that occurs at that point in the path.

Possible "RF" values are:

Value	Description
(empty)	Unknown transition
R	Rising output
F	Falling output
RR	Rising input, rising output
RF	Rising input, falling output
FR	Falling input, rising output
FF	Falling input, falling output

The "Type" column in the report uses a symbol to indicate what type of delay occurs at that point in the path.

Possible "Type" values are:

Value	Description
CELL	Cell delay
COMP	PLL clock network compensation delay
IC	Interconnect delay
iExt	External input delay
LOOP	Lumped combinational loop delay
oExt	External output delay
RE	Routing element (only for paths generated with the -show_routing option)
uTco	Register micro-Tco time
uTsu	Register micro-Tsu time
uTh	Register micro-Th time

The values of the "-from", "-to", and "-through" options are either collections or a Tcl list of wildcards used to create collections of appropriate types. The values used must follow standard Tcl or TimeQuest-extension substitution rules. See the help for use_timequest_style_escaping for details.

Example

```
project_open my_project

# Always create the netlist first
create_timing_netlist
read_sdc my_project.sdc
update_timing_netlist

# Run a setup analysis between nodes "foo" and "bar",
# reporting the worst-case slack if a path is found.

set my_list [report_timing -from foo -to bar]
set num_paths [lindex $my_list 0]
set wc_slack [lindex $my_list 1]
if { $num_paths > 0 } {
    puts "Worst case slack -from foo -to bar is $wc_slack"
}
```



```
# The following command is optional  
delete_timing_netlist  
  
project_close
```

report_ucp

Usage

```
report_ucp [-append] [-file <name>] [-panel_name <name>] [-stdout] [-summary]
```

Options

-append: If output is sent to a file, this option appends the result to that file. Otherwise, the file will be overwritten

-file <name>: Sends the results to an ASCII or HTML file. Depending on the extension

-panel_name <name>: Sends the results to the panel and specifies the name of the new panel

-stdout: Send output to stdout, via messages. You only need to use this option if you have selected another output format, such as a file, and would also like to receive messages.

-summary: Generate only the summary panel.

Description

Reports unconstrained paths.

Example

```
project_open chiptrip
create_timing_netlist
read_sdc
update_timing_netlist
report_ucp
delete_timing_netlist
project_close
```

set_operating_conditions

Usage

```
set_operating_conditions [-force_dat] [-grade <c|i|m|e|a>] [-model <fast|slow>] [-speed <speed>] [-temperature <value_in_C>] [-voltage <value_in_mV>] <operating_conditions>
```

Options

-force_dat: Option to force delay annotation

-grade <c|i|m|e|a>: Option to specify temperature grade

-model <fast|slow>: Option to specify timing model

-speed <speed>: Speed grade

-temperature <value_in_C>: Operating temperature

-voltage <value_in_mV>: Operating voltage

<operating_conditions>: Operating conditions Tcl object

Description

Use this command to specify operating conditions different from the initial conditions used to create the timing netlist. When a timing model is not specified, the slow model is used.

Voltage and temperature options must be used together. These two options are not available for all devices. The `get_available_operating_conditions` command returns the list of available operating conditions for your device.

Use the `-speed` option to analyze the design at a different speed grade of the selected device.

Use the `-grade` option to analyze the design at a different temperature grade. This option is provided to support what-if analysis and is not recommended for final sign-off analysis.

By default, delay annotation is skipped if previously performed. Use `-force_dat` to rerun delay annotation.

Example

```
#do report timing for different operating conditions
foreach_in_collection op [get_available_operating_conditions] {
    set_operating_conditions $op
    update_timing_netlist
    report_timing
}

#manually set operating conditions
set_operating_conditions -model fast -temperature 85 -voltage 1200
update_timing_netlist

#change device speed grade and set operating conditions
set_operating_conditions -speed 3 -model slow -temperature 0 -voltage \
    1100
update_timing_netlist
```

timing_netlist_exist

Usage

```
timing_netlist_exist
```

Options

None

Description

Checks if the timing netlist exists.

Returns 1, if the timing netlist exists. Returns 0, otherwise.

Example

```
if {[timing_netlist_exist]} {  
    create_timing_netlist  
}
```

update_timing_netlist

Usage

```
update_timing_netlist [-full]
```

Options

-full: Forces creation of an updated timing netlist to ensure correctness

Description

Updates and applies SDC commands to the timing netlist. The `update_timing_netlist` command expands and validates generated clocks, warns about sources in the design that require clock settings, identifies and removes combinational loops, and warns about undefined input/output delays.

Most Tcl commands (e.g., `report_timing`) automatically update the timing netlist when necessary. You can use the `update_timing_netlist` command explicitly to control when updating occurs, or to force a full update using the `-full` option.

Example

```
project_open top
create_timing_netlist
read_sdc
update_timing_netlist

report_timing -to_clock clk1
report_timing -to_clock clk2

delete_timing_netlist
project_close
```

use_timequest_style_escaping

Usage

```
use_timequest_style_escaping [-off] [-on]
```

Options

-off: Disable this setting.

-on: Enable this setting.

Description

Use TimeQuest-style escaping. (TimeQuest-style escaping is enabled by default.)

The values used to create a collection, whether explicitly using a collection command or implicitly as a value specified as a "-from", "-to", or similar option to various SDC and report commands, are a Tcl list of wildcards. This includes a single name with an exact match. The value must follow standard Tcl substitution rules for Tcl lists and "string match" as described below, unless using TimeQuest-style escaping (default).

For special characters such as '\$', the character must be escaped using a single '\' character to prevent Tcl from interpreting the word after '\$' as a Tcl variable, such as: Clk\Signal.

A '\' character itself must be escaped with another '\' as in the '\$' case, must be escaped again for the Tcl list, and must be escaped yet again for Tcl "string match." The final result is eight '\' characters, such as: Clk\\\\\\\\\\\\Signal.

Using Tcl "list" eliminates one level of escaping, since it will escape any '\' characters automatically for the Tcl list, such as:

```
[list Clk\\\\Signal]
```

Using '{' and '}' characters also eliminates the need for one or two levels of escaping, since '{' and '}' prevent string substitution in the contents, such as:

```
[List {Clk\\Signal}]
{{Clk\\Signal}}
{{Clk\\Signal}}
```

The use_timequest_style_escaping option, which is on by default, allows the user to specify a name containing '\' characters with only two '\' characters in all cases, such as: Clk\\Signal. The extra '\' characters required for Tcl list string substitution and "string match" are added automatically by TimeQuest.

To disable TimeQuest style string escaping, call "use_timequest_style_escaping -off" before adding any timing constraints or exceptions.

Example

```
project_open top
use_timequest_style_escaping -on
create_timing_netlist
set res [get_cells my_test|special_\\reg]
query_collection $res -all

delete_timing_netlist
project_close
```

write_sdc

Usage

```
write_sdc [-expand] [-history] <file_name>
```

Options

-expand: Generate SDC file by expanding the macros

-history: Reports full history of assignments

<file_name>: Name of output file

Description

Generates an SDC file with all current constraints and exceptions. When you use the -expand option, derive_clocks, derive_pll_clocks, derive_lvds_clocks and derive_clock_uncertainty macros are expanded to corresponding sdc assignments before they are written to a file. If you do not use the -expand option, these macros are preserved.

Example

```
project_new test
create_timing_netlist
create_clock -period 10 -name clk10 clk
set_multicycle_path -from [get_cells a] -to [get_cells b]
update_timing_netlist

report_timing

write_sdc my_sdc_file.sdc

delete_timing_netlist
project_close
```

