# Quartus II Scripting Reference Manual

**For Command-Line Operation & Tool Command Language (Tcl) Scripting**

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

**MNL-Q2101904-9.0**

MNL-Q2101904-9.0

I.S. EN ISO 9001

# Contents

# About this Reference Manual

This manual provides comprehensive information about the Altera® Quartus® II software Command-Line operation and Tcl language scripting.

## Revision History

The following table shows the revision history for this manual.

| Date and Document Version | Changes Made | Summary of Changes |
|---|---|---|
| March 2009, v9.0 | ■ Updated for Quartus II version 9.0 | — |
| November 2008, v8.2 | ■ Updated for Quartus II version 8.1<br>■ Added Revision History to this reference manual<br>■ Removed Index section<br>■ Updated new document template | — |
| July 2008, v8.1 | ■ Updated for Quartus II version 8.0 | — |

## How to Contact Altera

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

| Contact *(Note 1)* | 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) | Email | nacomp@altera.com |
| (Software Licensing) | 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**$_{\mathbf{MAX}}$, **\qdesigns** directory, **d:** drive, **chiptrip.gdf** file. |

| Visual Cue | Meaning |
|---|---|
| *Italic Type with Initial Capital Letters* | Document titles are shown in italic type with initial capital letters. Example: *AN 75: High-Speed Board Design.* |
| *Italic type* | Internal timing parameters and variables are shown in italic type.<br>Examples: $t_{PIA}$, $n + 1$.<br>Variable names are enclosed in angle brackets (< >) and shown in italic type. Example: <*file name*>, <*project name*>**.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: `data1`, `tdi`, `input`. Active-low signals are denoted by suffix `n`, e.g., `resetn`.<br>Anything that must be typed exactly as it appears is shown in Courier type. For example: `c:\qdesigns\tutorial\chiptrip.gdf`. Also, sections of an actual file, such as a Report File, references to parts of files (e.g., the AHDL keyword `SUBDESIGN`), as well as logic function names (e.g., `TRI`) are shown in Courier. |
| 1., 2., 3., and<br>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. |
| ⚠ CAUTION | A caution calls attention to a condition or possible situation that can damage or destroy the product or the user’s work. |
| ⚠ WARNING | 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. |

# 1. Introduction to the Quartus II Scripting Reference Manual

## Introduction

The Quartus® II design software provides the FPGA industry's easiest-to-use and most powerful scripting environment available for command-line operation and tool command language (Tcl) scripting. This scripting environment is offered in addition to the Quartus II development software rich graphical user interface (GUI).

This overview covers the Quartus II design software support for command-line operation and Tcl scripting.

## Overview

Each stage of the Quartus II software design flow corresponds to a command-line executable file. Many of these executable files also support industry-standard Tcl scripting for custom functionality or processing beyond the GUI design flow. Quartus II design software offers the following scripting support benefits, also known as CAR:

- Custom Analysis
- Automation
- Reproducibility

*Custom analysis* allows you to build test procedures into the script and change design processing based on the test results. Scripts can *automate* design flows to perform on multiple computers simultaneously and easily archive and restore projects. *Reproducibility* ensures that scripts use the same project setup and assignments for every compile, even when you transfer a project from one engineer to another. In other words, you can use scripts as another level of design quality assurance.

The *Quartus II Scripting Reference Manual* is your reference guide to Quartus II software command-line executables and Tcl commands, including command details, usage, and examples.

All of the information included in the *Quartus II Scripting 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 ↵
```

### Quartus II Software Command-Line Operation Support

Use command-line operation for:

- Scripting simple design flows
- Compiling existing projects
- Making global project assignments

- Incorporating third-party EDA executable files
- Makefile operation

You can also group commands for Quartus II executable files into a script, batch file, or in a makefile to automate design flows. The Quartus II software command-line executables accept arguments to set project variables and access common settings.

## Quartus II Software Tcl Scripting Support

Use Tcl scripting for:

- Solving complex analysis
- Making individual assignments
- Generating custom reports
- Creating custom solutions

Tcl is an EDA industry-standard scripting language used by Synopsys, Mentor Graphics®, Synplicity, Altera, and others. The Tcl language supports control structures, variables, procedures, network socket access, and application programming interfaces (APIs). Altera's Tcl support is aligned with major EDA vendor solutions. It has an API format similar to the Synopsys design constraint (SDC) format used by the Synopsys PrimeTime and Design Compiler products.

## Command-Line Executables

Quartus II software provides command-line executables for each stage in the design flow. The software also provides additional executables for specific tasks.

Table 1 details the command-line executables and their respective descriptions.

**Table 1.** Quartus II Command-Line Executables and Descriptions (Part 1 of 3)

| Executable | Description |
|---|---|
| Analysis and Synthesis<br>quartus_map | Quartus II Analysis and Synthesis builds a single project database that integrates all the design files in a design entity or project hierarchy, performs logic synthesis to minimize the logic of the design, and performs technology mapping to implement the design logic using device resources such as logic elements. |
| Fitter<br>quartus_fit | The Quartus II Fitter performs place-and-route by fitting the logic of a design into a device. The Fitter selects appropriate interconnection paths, pin assignments, and logic cell assignments.<br><br>Quartus II Analysis and Synthesis must be run successfully before running the Fitter. |
| Signal Integrity<br>quartus_si | The Quartus II SSN Analyzer estimates the simultaneous switching noise contributions to voltage and timing noise.<br><br>Quartus II Analysis & Synthesis and the Fitter must be run successfully before running the SSN Analyzer. |

**Table 1.** Quartus II Command-Line Executables and Descriptions (Part 2 of 3)

| Executable | Description |
|---|---|
| Assembler<br>quartus_asm | The Quartus II Assembler generates a device programming image, in the form of one or more of the following from a successful fit (that is, place-and-route).<br>Programmer Object Files (**.pof**)<br>SRAM Object Files (**.sof**)<br>Hexadecimal (Intel-Format) Output Files (**.hexout**)<br>Tabular Text Files (**.ttf**)<br>Raw Binary Files (**.rbf**)<br><br>The **.pof** and **.sof** files are then processed by the Quartus II Programmer and downloaded to the device with the MasterBlaster™ or the ByteBlaster™ II download cable, or the Altera Programming Unit (APU). The Hexadecimal (Intel-Format) Output Files, Tabular Text Files, and Raw Binary Files can be used by other programming hardware manufacturers that provide support for Altera devices.<br><br>The Quartus II Fitter must be run successfully before running the Assembler. |
| Classic Timing Analyzer<br>quartus_tan | The Quartus II Classic Timing Analyzer computes delays for the given design and device, and annotates them on the netlist. Then, the Classic Timing Analyzer performs timing analysis, allowing you to analyze the performance of all logic in your design. The **quartus_tan** executable includes Tcl support.<br><br>Quartus II Analysis and Synthesis or the Fitter must be run successfully before running the Classic Timing Analyzer. |
| TimeQuest Timing Analyzer<br>quartus_sta | The Quartus II TimeQuest Timing Analyzer computes delays for the given design and device, and annotates them on the netlist. Then, the TimeQuest Timing Analyzer performs timing analysis, allowing you to analyze the performance of all logic in your design. The **quartus_sta** executable includes Tcl support and SDC support.<br><br>Quartus II Analysis and Synthesis or the Fitter must be run successfully before running the TimeQuest Timing Analyzer. |
| Design Assistant<br>quartus_drc | The Quartus II Design Assistant checks the reliability of a design based on a set of design rules. The Design Assistant is especially useful for checking the reliability of a design before converting the design for HardCopy® devices. The Design Assistant supports designs that target any Altera device supported by the Quartus II software, except MAX® 3000 and MAX 7000 devices.<br><br>Quartus II Analysis and Synthesis or the Fitter must be run successfully before running the Design Assistant. |
| Compiler Database Interface<br>quartus_cdb | The Quartus II Compiler Database Interface generates incremental netlists for use with LogicLock™ back-annotation, or back-annotates device and resource assignments to preserve the fit for future compilations. The **quartus_cdb** executable includes Tcl support.<br><br>Analysis and Synthesis must be run successfully before running the Compiler Database Interface. |

**Table 1.** Quartus II Command-Line Executables and Descriptions (Part 3 of 3)

| Executable | Description |
|---|---|
| EDA Netlist Writer<br>quartus_eda | The Quartus II EDA Netlist Writer generates netlist and other output files for use with other EDA tools.<br><br>Analysis and Synthesis, the Fitter, or Timing Analyzer must be run successfully before running the EDA Netlist Writer, depending on the arguments used. |
| Simulator<br>quartus_sim | The Quartus II Simulator tests and debugs the logical operation and internal timing of the design entities in a project. The Simulator can perform two types of simulation: functional simulation and timing simulation. The **quartus_sim** executable includes Tcl support.<br><br>Quartus II Analysis and Synthesis must be run successfully before running a functional simulation.<br><br>The Timing Analyzer must be run successfully before running a timing simulation. |
| Power Analyzer<br>quartus_pow | The Quartus II PowerPlay Power Analyzer estimates the thermal dynamic power and the thermal static power consumed by the design. For newer families such as Stratix® II and MAX II, the power drawn from each power supply is also estimated.<br><br>Quartus II Analysis and Synthesis or the Fitter must be run successfully before running the PowerPlay Power Analyzer. |
| Programmer<br>quartus_pgm | The Quartus II Programmer programs Altera devices. The Programmer uses one of the supported file formats:<br>Programmer Object Files (**.pof**)<br>SRAM Object Files (**.sof**)<br>Jam File (**.jam**)<br>Jam Byte-Code File (**.jbc**)<br><br>Make sure you specify a valid programming mode, programming cable, and operation for a specified device. |
| Convert Programming File<br>quartus_cpf | The Quartus II Convert Programming File module converts one programing file format to a different possible format.<br><br>Make sure you specify valid options and an input programming file to generate the new requested programming file format. |
| Quartus Shell<br>quartus_sh | The Quartus II Shell acts as a simple Quartus II Tcl interpreter. The Shell has a smaller memory footprint than the other command-line executables that support Tcl. The Shell may be started as an interactive Tcl interpreter (shell), used to run a Tcl script, or used as a quick Tcl command evaluator, evaluating the remaining command-line arguments as one or more Tcl commands. |
| TimeQuest Timing Analyzer GUI<br>quartus_staw | This executable opens the Quartus II TimeQuest Timing Analyzer GUI. This is helpful because you don't have to open the entire Quartus II GUI for certain operations. |
| Programmer GUI<br>quartus_pgmw | This executable opens up the programmer—a GUI to the quartus_pgm executable. This is helpful because users don't have to open the entire Quartus II GUI for certain operations |

## Tcl Commands

The Quartus II software Tcl commands are grouped into Tcl packages and loaded on demand. This reduces the run-time memory of executable files and makes memory available to application. Table 2 describes each Tcl package.

**Table 2.** Tcl Packages

| Package Name | Package Description |
|---|---|
| **advanced_timing** | Traverse the timing netlist and get information about timing nodes |
| **backannotate** | Back annotate assignments |
| **chip_editor** | Identify and modify resource usage and routing with the Chip Editor |
| **database_manager** | Manage version-compatible database files |
| **device** | Get device and family information from the device database |
| **flow** | Compile a project, run command-line executables and other common flows |
| **insystem_memory_edit** | Read and edit memory contents in Altera devices |
| **jtag** | Control the jtag chain |
| **logic_analyzer_interface** | Query and modify the logic analyzer interface output pin state |
| **logiclock** | Create and manage LogicLock regions |
| **misc** | Perform miscellaneous tasks |
| **project** | Create and manage projects and revisions, make any project assignments including timing assignments |
| **report** | Get information from report tables, create custom reports |
| **sdc** | Specifies constraints and exceptions to the TimeQuest Analyzer |
| **sdc_ext** | Altera-specific SDC commands |
| **simulator** | Configure and perform simulations |
| **sta** | Contains the set of Tcl functions for obtaining advanced information from the TimeQuest Timing Analyzer |
| **stp** | Run the SignalTap® II logic analyzer |
| **timing** | Annotate timing netlist with delay information, compute and report timing paths |
| **timing_assignment** | Contains the set of Tcl functions for making project-wide timing assignments, including clock assignments; all Tcl commands designed to process Classic Timing Analyzer assignments have been moved to this package |
| **timing_report** | List timing paths |

Table 3 lists the Quartus II Tcl packages available with Quartus II executables and indicates whether a package is loaded by default (●) or is available to be loaded as necessary (◓). A clear circle (○) means that the package is not available in that executable.

**Table 3.** Tcl Package Availability by Quartus II Executable *(Note 1)*, *(2)*, *(3)*

| **Packages** | **Quartus II Executable** | | | | | | |
|---|---|---|---|---|---|---|---|
| | **Quartus_sh** | **Quartus_tan** | **Quartus_cdb** | **Quartus_sim** | **Quartus_stp** | **Quartus_sta Quartus_staw** | **Tcl Console** |
| advanced_timing | ○ | ◓ | ○ | ○ | ○ | ○ | ○ |
| backannotate | ○ | ○ | ◓ | ○ | ○ | ○ | ◓ |
| chip_planner | ○ | ○ | ◓ | ○ | ○ | ○ | ○ |
| device | ● | ◓ | ● | ● | ○ | ● | ◓ |
| flow | ◓ | ◓ | ◓ | ◓ | ○ | ◓ | ◓ |
| insystem_memory_edit | ○ | ○ | ○ | ○ | ● | ○ | ○ |
| jtag | ○ | ○ | ○ | ○ | ● | ○ | ○ |
| logic_analyzer_interface | ○ | ○ | ○ | ○ | ● | ○ | ○ |
| logiclock | ○ | ◓ | ◓ | ○ | ○ | ○ | ◓ |
| misc | ● | ● | ● | ● | ● | ● | ● |
| old_api | ○ | ○ | ○ | ○ | ○ | ○ | ● |
| project | ● | ● | ● | ● | ● | ● | ● |
| report | ◓ | ◓ | ◓ | ● | ○ | ● | ◓ |
| sdc | ○ | ○ | ○ | ○ | ○ | ● | ○ |
| sdc_ext | ○ | ○ | ○ | ○ | ○ | ● | ○ |
| simulator | ○ | ○ | ○ | ● | ○ | ○ | ○ |
| sta | ○ | ○ | ○ | ○ | ○ | ● | ○ |
| stp | ○ | ○ | ○ | ○ | ● | ○ | ○ |
| timing | ○ | ● | ○ | ○ | ○ | ○ | ○ |
| timing_assignment | ● | ● | ● | ● | ● | ○ | ○ |
| timing_report | ○ | ◓ | ○ | ○ | ● | ○ | ● |

**Notes to Table 3:**

(1) A dark circle (●) indicates that the package is loaded automatically.

(2) A half-circle (◓) means that the package is available but not loaded automatically.

(3) A white circle (○) means that the package is not available for that executable.

## Related Documentation

Table 4 presents additional resources and documentation for the Quartus®II development software.

**Table 4.** Quartus II Software Resources & Documentation

| Resource Name | Resource Type | Access | Description | User Level |
|---|---|---|---|---|
| Quartus II Online Demonstrations | 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 `quartus_sh --qhelp` 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 & 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 |

# 2. Command-line Executables

## quartus_asm

The Quartus®II Assembler generates a device programming image, in the form of one or more Programmer Object Files (.pof), SRAM Object Files (.sof), Hexadecimal (Intel-Format) Output Files (.hexout), Tabular Text Files (.ttf), and Raw Binary Files(.rbf), from a successful fit (that is, place and route).

The .pof and .sof files can then be processed by the Quartus II Programmer and the MasterBlaster™or the ByteBlaster™II Download Cable, or the Altera®Programming Unit (APU). The .hexout, .ttf, and .rbf files can be used by other programming hardware manufacturers that provide programming support for Altera devices.

The Quartus II Fitter must be run successfully before running the Assembler.

### Usage

```
quartus_asm [-h | --help[=<option|topic>] | -v]
```

```
quartus_asm <project name> [<options>]
```

This command supports the following options:

| Option | Page |
|---|---|
| -c=<revision name> | 2–98 |
| -f=<argument file> | 2–94 |
| -h | 2–94 |
| -v | 2–94 |
| --64bit | 2–94 |
| --help[=<option\|topic>] | 2–95 |
| --lower_priority | 2–95 |
| --read_settings_files[=on\|off] | 2–100 |
| --rev=<revision name> | 2–98 |
| --set=<assignment=value> | 2–98 |
| --version | 2–95 |
| --write_settings_files[=on\|off] | 2–102 |

This command includes help on the following topics:

| Help Topic | Page |
|---|---|
| arguments | 2–95 |
| makefiles | 2–95 |
| return_codes | 2–96 |

## quartus_cdb

The Quartus II Compiler Database Interface manages version-compatible database files and generates incremental netlists for use with LogicLock™back-annotation, or back-annotates device and resource assignments to preserve the fit for future compilations. The quartus_cdb executable includes Tcl support. Analysis & Synthesis must be run successfully before running the Compiler Database Interface.

### Usage

```
quartus_cdb [-h | --help[=<option|topic>] | -v]
```

```
quartus_cdb <project name> [<options>]
```

```
quartus_cdb -t <script file> [<script args>]
```

```
quartus_cdb -s
```

```
quartus_cdb --tcl_eval <tcl command>
```

This command supports the following options:

This command includes help on the following topics:

## --back_annotate=<demotion type>

Option to back-annotate to the current Quartus II Settings File (.qsf) according to the specified demotion type. The back-annotation process retains the current resource and device assignments for future compilations. The demotion type is used to select the assignments that you want to back-annotate. The demotion type can be specified in one of the following forms:

| Demotion Type | Descriptions |
|---|---|
| device | For device assignments |
| pin_device | For pin device assignments |
| lc | For logic cell assignments |
| routing | For routing assignments |
| lab | For LAB assignments |
| megalab | For MegaLAB assignments |
| megalab_row | For MegaLAB row assignments |
| megalab_column | For MegaLAB column assignments |
| row | For row assignments |

Note: Not all demotion types are relevant for all device families. The demotion type "routing" applies only to the Cyclone™, Stratix®, and Stratix GX device families.

## --bottom_up_scripts_output_directory[=<value>]

Specifies the output directory for scripts and makefiles.  If none is set, the project directory is used by default.

## --bottom_up_scripts_virtual_input_pin_delay[=<value>]

Specifies a delay, in nanoseconds, used to constrain all paths from any of the newly created virtual input pins in the lower-levels. This is to help guide the lower-level placement and produce a better quality top-level result.

The value represents the maximum acceptable delay for an inter-partition signal to arrive at the project's virtual input pin from another module. The value helps guide lower-level placement.

## --bottom_up_scripts_virtual_output_pin_delay[=<value>]

Specifies a delay, in nanoseconds, used to constrain all paths to any newly created virtual output pins in the lower-levels. This helps guide the lower-level placement and produces a better quality top-level result.

The value represents the maximum acceptable delay for an inter-partition signal driven by the virtual output pin to arrive at its destination. The value helps guide lower-level placement.

## --create_companion[=<companion revision>]

Creates a HardCopy companion revision based on the current revision.

If <companion revision> is specified, it is used to create the HardCopy companion revision. Otherwise, the default companion revision name is used unless the COMPANION_REVISION_NAME assignment is found in the current revision's Quartus II Settings File (.qsf).

Reverse migration also supported when the originating revision is a HardCopy device (HardCopy II or newer). The companion created will be targeted to FPGA revision with a proper companion name if the <companion revision> is not specified and the COMPANION_REVISION_NAME assignment is not found.

The current revision should be fully compiled for this option to migrate all pin locations.

### Usage

```
quartus_cdb <project> [-c <current revision>] --create_companion=[<companion revision>]
```

Note that <companion revision> is the new current revision after executing this option.

### Examples

```
## The following example illustrates the option usage using
## Stratix II and HardCopy II migration scenario

## Compile the Stratix II revision
quartus_sh --flow compile myproject -c myfpga

## Create a HardCopy II revision named "myhcii" based on "myfpga"
quartus_cdb myproject -c myfpga --create_companion=myhcii

## Compile the HardCopy II revision
quartus_sh --flow compile myproject -c myhcii

## **********************************
## Or you can simply do the following
## **********************************

## Compile the Stratix II revision
quartus_sh --flow compile myproject

## The following command will create the
## HardCopy II revision named "myproject_hcii"
## unless the COMPANION_REVISION_NAME assignment
## is found in myproject.qsf.
quartus_cdb myproject --create_companion
```

```
## Compile the HardCopy II revision
quartus_sh --flow compile myproject -c myproject_hc
```

## --disable_auto_global_promotion_in_bottom_up_scripts[=on|off]

When this option is enabled, generated Tcl scripts contain commands that disable auto global signal promotion in the lower levels.

This option is enabled by default. Add the flag
--disable_auto_global_promotion_in_bottom_up_scripts=off to disable.

## --export_database=<directory>

Option to export the project database to version-compatible database files. These files are placed in the specified directory.

The following are the supported version-compatible database files to which the project database is exported:

Post-synthesis files
<directory>/<revision name>.map.atm
<directory>/<revision name>.map.hdbx

Post-fitting files
<directory>/<revision name>.cmp.atm
<directory>/<revision name>.cmp.hdbx
<directory>/<revision name>.cmp.rcf

Generic files
<directory>/<revision name>.cmp.xml
<directory>/<revision name>.db_info

## --generate_bottom_up_scripts[=on|off]

### Overview

This tool is designed for use with a top-level project containing incremental compilation design partitions. When run, it generates scripts and makefiles which allow an easy conversion from top-down design methodology (all partitions in one project) to a bottom-up design methodology (separate projects for each partition).

One Tcl script is generated for each partition. The Tcl script will contain all top-level assignments relevant to the given partition, and optionally contains commands to create the lower-level project for the partition if it does not exist. The scripts also contain optionally generated commands that can help guide the lower-level placement so that better results can be achieved when exporting to the top-level project. You can customize the content of the Tcl scripts by using any of the options described later.

In addition to generating Tcl scripts, you can also generate makefiles that can be used to create the lower-level projects with the auto-generated Tcl scripts and maintain them as source files change. The tool also builds a 'master_makefile' which builds all lower-level projects, exports the results to the top-level project and performs a top-level compilation. The makefiles are auto-generated and are designed for use with GNU make. The makefiles also support parallel compilation of the the lower-level projects by using the '-j' option of GNU make on systems with multiple processors.

### Optional Content

As mentioned above, you can customize the content of the Tcl files with any of the following command line directives. Each is explained in more detail in their own help sections.

```
--include_makefiles_with_bottom_up_scripts=<on|off>
Default is on.
```

```
--include_project_creation_in_bottom_up_scripts=<on|off>
Default is on.
```

```
--include_virtual_pins_in_bottom_up_scripts=<on|off>
Default is on.
```

```
--include_virtual_input_pin_timing_in_bottom_up_scripts=<on|off>
Default is on.
```

```
--include_virtual_output_pin_timing_in_bottom_up_scripts=<on|off>
Default is on.
```

```
--include_virtual_pin_locations_in_bottom_up_scripts=<on|off>
Default is on.
```

```
--include_logiclock_regions_in_bottom_up_scripts=<on|off>
Default is on.
```

```
--include_all_logiclock_regions_in_bottom_up_scripts=<on|off>
Default is on.
```

```
--include_global_signal_promotion_in_bottom_up_scripts=<on|off>
Default is off.
```

```
--include_pin_locations_in_bottom_up_scripts=<on|off>
Default is on.
```

```
--include_timing_assignments_in_bottom_up_scripts=<on|off>
Default is on.
```

```
--include_design_partitions_in_bottom_up_scripts=<on|off>
Default is on.
```

```
--remove_existing_regions_in_bottom_up_scripts=<on|off>
Default is on.
```

```
--disable_auto_global_promotion_in_bottom_up_scripts=<on|off>
Default is off.
```

```
--bottom_up_scripts_output_directory=<output_directory>
Default is current project directory.
```

```
--bottom_up_scripts_virtual_input_pin_delay=<delay_in_ns>
No default.  Must provide if including virtual input pin timing.
```

```
--bottom_up_scripts_virtual_output_pin_delay=<delay_in_ns>
No default.  Must provide if including virtual output pin timing.
```

## --generate_hc_files

Generates HardCopy Handoff files to an output directory.

The output directory is "hc_output" under the current project directory unless the output directory is specified by the HC_OUTPUT_DIR assignment in the Quartus II Settings File (.qsf).

## --generate_hc_pll_delay

Returns the PLL annotated delay.

Ggenerates the PLL annotated delay for both commercial and industrial speed grades of HardCopy devices. This option only returns the PLL annotated delay and does not update the collection.sdc file.

### Usage

```
quartus_cdb <project> -c <current Hardcopy revision> --generate_hc_pll_delay
```

### Examples

```
## Run TimeQuest for HardCopy revision
   quartus_sta myproject -c myproject_hcii

## Write the PLL Annotated Delay for either Commercial or Industry Device
   quartus_cdb myproject -c myproject_hcii --generate_hc_pll_delay
```

## --hc_archive[=<output file>]

Archives HardCopy Handoff Files into the specified output file name.

The Quartus II Archive File <output file>.qar is generated by this option. By default, if the output file name is not specified, the file name <current revision>.qar is  used.

The current revision and its companion revision should be fully compiled for this option to properly archive all necessary files. The companion revision is obtained from the COMPANION_REVISION_NAME assignment in the Quartus II Settings File (.qsf) for the current revision.

### Usage

```
quartus_cdb <project> [-c <current revision>] --hc_archive=[<output file>]
```

### Examples

```
   ## The following example illustrates the option usage using
   ## Stratix II and HardCopy II migration scenario

   ## Compile the Stratix II revision
   quartus_sh --flow compile myproject -c myfpga

   ## Create a HardCopy II revision named "myhcii" based on "myfpga"
   quartus_cdb myproject -c myfpga --create_companion=myhcii

   ## Compile the HardCopy II revision
   quartus_sh --flow compile myproject -c myhcii

   ## Verify that the design was migrated corerctly
    quartus_cdb myproject -c myhcii --compare=myfpga
    quartus_cdb myproject -c myfpga --compare=myhcii

   ## Generates Quartus II Archive File "myfpga.qar"
   quartus_cdb myproject -c myfpga --hc_archive

   ## Generate Quartus II Archive File "custom.qar"
   quartus_cdb myproject -c myfpga --hc_archive=custom
```

## --hc_min_archive

Option to archive the minimum set of files only.

Source files and other informational files are not included when archiving the HardCopy handoff files using this option.

## --hc_ready

Generate HardCopy Design Readiness Check report.

The report contains information on required settings for the design to be ready for HardCopy devices.

### Usage

```
quartus_cdb <project> [-c <current revision>] --hc_ready
```

### Examples

```
## The following example illustrates the option usage using
## Stratix II and HardCopy II migration scenario
## Run Analysis & Synthesis for Stratix II revision quartus_map myproject -c myfpga

## Run HardCopy Design Readiness Check for Stratix II revision quartus_cdb myproject -c
myfpga --hc_ready

## Create a HardCopy II revision named "myhcii" based on "myfpga"quartus_cdb myproject
-c myfpga --create_companion=myhcii

## Run Analysis & Synthesis for HardCopy II revision quartus_map myproject -c myhcii

## Run HardCopy Design Readiness Check for HardCopy II revision quartus_cdb myproject
-c myhcii --hc_ready
```

## --hc_review

Generates a HardCopy Handoff Report.

The current revision and its companion revision should be fully compiled for this option to properly review all necessary files. The companion revision is obtained from the COMPANION_REVISION_NAME assignment in the Quartus II Settings File (.qsf) for the current revision.

### Usage

```
quartus_cdb <project> [-c <current revision>] --hc_review

---------
Examples:
---------
   ## The following example illustrates the option usage using
   ## Stratix II and HardCopy II migration scenario

   ## Compile the Stratix II revision
    quartus_sh --flow compile myproject -c myfpga

   ## Create a HardCopy II revision named "myhcii" based on "myfpga"
    quartus_cdb myproject -c myfpga --create_companion=myhcii

   ## Compile the HardCopy II revision
    quartus_sh --flow compile myproject -c myhcii

   ## Verify that the design was migrated corerctly
    quartus_cdb myproject -c myhcii --compare=myfpga
    quartus_cdb myproject -c myfpga --compare=myhcii

   ## Review the HardCopy II design
    quartus_cdb myproject -c myfpga --hc_review
```

## --import_database=<directory>

Option to import the project database from version-compatible database files in the specified directory.

The following are the supported version-compatible database files from which the project database is imported:

Post-synthesis files
<directory>/<revision name>.map.atm
<directory>/<revision name>.map.hdbx

Post-fitting files
<directory>/<revision name>.cmp.atm
<directory>/<revision name>.cmp.hdbx
<directory>/<revision name>.cmp.rcf

Generic files
<directory>/<revision name>.cmp.xml
<directory>/<revision name>.db_info

## --include_all_logiclock_regions_in_bottom_up_scripts[=on|off]

When this option is enabled, every generated Tcl script contains all of the top-level LogicLock regions. Regions with logic not associated with the script's target partition act as placeholders and are empty. This command helps describes for the lower-level project the way it fits into the final, top-level design.

The option is ignored if --include_logiclock_regions_in_bottom_up_scripts=off is used.

This option is enabled by default. Add the flag --include_all_logiclock_regions_in_bottom_up_scripts=off to disable.

## --include_design_partitions_in_bottom_up_scripts[=on|off]

When this option is enabled, generated Tcl scripts contain all relevant design partition assignments from the top-level.

This option is enabled by default. Add the flag --include_design_partitions_in_bottom_up_scripts=off to disable.

## --include_global_signal_promotion_in_bottom_up_scripts[=on|off]

When this option is enabled, generated Tcl scripts contain commands that force any signals promoted to 'global' in the top-level to be promoted in the lower-level.

This option is enabled by default. Add the flag --include_global_signal_promotion_in_bottom_up_scripts=off to disable.

## --include_logiclock_regions_in_bottom_up_scripts[=on|off]

When this option is enabled, generated Tcl scripts contain the top-level LogicLock regions associated with this partition. This helps ensure the lower-level project places its logic where the top-level project expects it.

This option is enabled by default. Add the flag --include_logiclock_regions_in_bottom_up_scripts=off to disable.

## --include_makefiles_with_bottom_up_scripts[=on|off]

Option to generate makefiles for lower-level projects in addition to the Tcl scripts. One makefile is generated for each lower-level project, for the top-level project and for the overall design (known as the master_makefile). The master makefile simply invokes all other makefiles.

Makefiles are designed to work with GNU make and support the '-j' option which allows parallel compilation of the lower-level projects. The master makefile is all that needs to be called by the user to ensure the lower-level projects are up to date and that the top-level project has imported the latest versions of the lower-level projects. You can invoke the master makefile as follows (you must not turn off --include_project_creation_in_bottom_up_scripts for this to work without modification):

```
gnumake -f master_makefile.mak -j2
<The '-j2' means there are 2 processors to use.>
```

Makefiles are placed in the directory of the project they control if project creation is enabled and appropriate directories are automatically filled in. If you elect not to have the tool create projects for you, all makefiles are placed in the specified output directory and the user must fill in the directory variables at the top of each makefile so that the tool knows where the lower-level projects can be found. In both cases you must add the source file dependencies for each lower-level project's makefile if maintainance of the project after initial compilation is desired. By default no dependencies are created (other than one on the auto-generated Tcl script for that partition) and so after the first c ompilation, the rule is be up to date.

By default, makefile generation is enabled. Add the flag --include_makefiles_with_bottom_up_scripts=off to disable.

## --include_pin_locations_in_bottom_up_scripts[=on|off]

When this option is enabled, generated Tcl scripts contain commands that lock any lower-level pins connected directly to IOs in the top-level to the IO location they were placed at in the top-level. This helps keep a consistent pin placement amongst projects.

This option is enabled by default. Add the flag --include_pin_locations_in_bottom_up_scripts=off to disable.

## --include_project_creation_in_bottom_up_scripts[=on|off]

When this option is enabled, generated Tcl scripts contain commands to create the lower-level projects if the projects do not exist. The tool creates projects in subdirectories under the output directory, named according to name of the corresponding partition.

By default, project creation is enabled. Add the flag --include_project_creation_in_bottom_up_scripts=off to disable.

## --include_timing_assignments_in_bottom_up_scripts[=on|off]

When this option is enabled, generated Tcl scripts contain all relevant timing assignments from the top-level.

This option is enabled by default. Add the flag --include_timing_assignments_in_bottom_up_scripts=off to disable.

## --include_virtual_input_pin_timing_in_bottom_up_scripts[=on|off]

When this option is enabled, generated Tcl scripts contain INPUT_MAX_DELAY to constrain all paths to the newly created virtual input pins (see --include_virtual_pins_in_bottom_up_scripts). The value for this option is the inter-partition delay of paths driving the virtual inputs. For more information about the meaning of theseassignments, please see the Quartus II Help topics relating to INPUT_MAX_DELAY.

The option is ignored if --include_virtual_pins_in_bottom_up_scripts=off is used.

If you use this option, you must also specify the delay (in nanoseconds) to be used in the constraints with the --bottom_up_scripts_virtual_input_pin_delay option.

This option is enabled by default. Add the flag
--include_virtual_input_pin_timing_in_bottom_up_scripts=off to disable.

## --include_virtual_output_pin_timing_in_bottom_up_scripts[=on|off]

When this option is enabled, generated Tcl scripts contain OUTPUT_MAX_DELAY to constrain all paths to the newly created virtual output pins (see --include_virtual_pins_in_bottom_up_scripts). Specify the inter-partition delay seen by paths driven by the virtual outputs. For more information about the meaning of theseassignments, please see the Quartus II Help topics relating to OUTPUT_MAX_DELAY.

The option is ignored if --include_virtual_pins_in_bottom_up_scripts=off is used.

If you use this option, you must also specify the delay (in nanoseconds) to be used in the constraints with the --bottom_up_scripts_virtual_output_pin_delay=<delay> command.

This option is enabled by default. Add the flag --include_virtual_output_pin_timing_in_bottom_up_scripts=off to disable.

## --include_virtual_pin_locations_in_bottom_up_scripts[=on|off]

When this option is enabled, generated Tcl scripts contain location constraints on newly created virtual pins (see --include_virtual_pins_in_bottom_up_scripts). The pins are locked to their top-level source (for input pins) or sink (for output pins) location.

The option is ignored if --include_virtual_pins_in_bottom_up_scripts=off is used.

This option is enabled by default. Add the flag --include_virtual_pin_locations_in_bottom_up_scripts=off to disable.

## --include_virtual_pins_in_bottom_up_scripts[=on|off]

When enabled, this means that generated Tcl scripts contain commands to mark all lower-level pins that connect to other design entities in the top-level (i.e. not directly to IOs) as virtual pins. This helps prevent overuse of IOs and leads to a more accurate representation of the lower-level project.

This option is enabled by default. Add the flag --include_virtual_pins_in_bottom_up_scripts=off to disable.

## --incremental_compilation_export[=<.qxp file>]

Exports a design partition into a Quartus II Exported Partition (.qxp) file. The .qxp file contains the compilation results of the specified partition, and can be imported into one or more design partitions of another project.

The value <output file> is optional. If unspecified, the value specified by the INCREMENTAL_COMPILATION_EXPORT_FILE global assignment is used if present. Otherwise, a default file name is generated.

Other options you may use to control how the operation is performed include:

```
--incremental_compilation_export_partition_name[=<name>]
--incremental_compilation_export_routing[=<on|off>]
--incremental_compilation_export_netlist_type=<POST_SYNTH|POST_FIT>
```

Each of the above is explained in more detail in their own help sections.

## --incremental_compilation_export_netlist_type=<POST_SYNTH|POST_FIT>

This option must be used with the --incremental_compilation_export option. The values POST_FIT and POST_SYNTH are used to direct the software to export the post-fit and the post-synthesis netlist, respectively. If this option is omitted, the value specified by the INCREMENTAL_COMPILATION_EXPORT_NETLIST_TYPE global assignment is used if present. Otherwise, the top-level partition is exported by default.

## --incremental_compilation_export_partition_name[=<name>]

This option must be used with the --incremental_compilation_export option. Use this option to specify the name of the partition to be exported. If this option is omitted, or if an empty value is provided, the value specified by the INCREMENTAL_COMPILATION_EXPORT_PARTITION_NAME global assignment is used if present. Otherwise, the top-level partition is exported by default.

## --incremental_compilation_export_routing[=on|off]

This option must be used with the --incremental_compilation_export option. The value specifies whether routing is to be exported. The option only has an effect when a post-fit netlist is exported, because a post-synthesis netlist does not contain routing information. If this option is omitted, the value specified with the INCREMENTAL_COMPILATION_EXPORT_ROUTING global assignment is used if present. Otherwise, the default is to export routing unless the currently specified device family does not support it.

## --incremental_compilation_import[=on|off]

Imports one or more Quartus II Exported Partition (.qxp) files into the design partitions of the current project.

The option uses the following partition assignments to determine the location of the Quartus II Exported Partition files, and how importation should be performed, on a per-partition basis:

PARTITION_IMPORT_FILE
PARTITION_IMPORT_PROMOTE_ASSIGNMENTS
PARTITION_IMPORT_NEW_ASSIGNMENTS
PARTITION_IMPORT_EXISTING_ASSIGNMENTS
PARTITION_IMPORT_EXISTING_LOGICLOCK_REGIONS

## --merge[=on|off]

Merges all design partitions to prepare a netlist for the Fitter based on the current Partition Netlist Type assignments.

## --mif_dependency=<mif_check>

Run the Memory Initialization File dependency check script.

If your design has RAM without a Memory Initialization File, then the Memory Initialization File dependency script creates an extra revision called <rev_name>_mif_dependency with the ASM option (use checkerboard pattern) turned on to ensure that the design has no initialization dependency.

### Usage

```
quartus_cdb <project> -c <current FPGA revision> --mif_dependency=mif_check
```

```
quartus_cdb <project> -c <current FPGA revision> --mif_dependency=cleanup
```

### Examples

```
## Run Analysis & Synthesis for Stratix II revision
quartus_map myproject -c myfpga

## Run Fitter for Stratix II revision
quartus_fit myproject -c myfpga

## Run MIF Dependency Check for Stratix II revision
quartus_cdb myproject -c myfpga --mif_dependency=mif_check

## Run MIF Dependency Check cleanup for Stratix II revision
quartus_cdb myproject -c myfpga --mif_dependency=cleanup
```

## --netlist_type=<map|cmp|asm>

Loads the specified atom netlist type. This option is used in combination with the "-write_equation_file" option. Use "map" to specify the post Analysis & Synthesis netlist. Use "cmp" to specify the post Fitter netlist. Use "asm" to specify the post Assembler netlist. The post Assembler netlist is only supported for designs using the HardCopy device family.

## --override_partition_netlist_type=<value>

Overrides the netlist type setting for the specified Design Partition for this compililation. The option hmust be used with the --merge option.

<value> takes the form of "<partition name>=<netlist type>", including the double quotes. For example, to use netlist type POST_SYNTH for a partition named "alu", use the following argument:

```
--override_partition_netlist_type="alu=POST_SYNTH"
```

For non-imported partitions, the following netlist type values can be used:

```
   POST_SYNTH
   POST_FIT
   STRICT_POST_FIT
   EMPTY
```

For imported partitions, the following netlist type values can be used:

```
   IMPORT
   IMPORT_BASED_POST_FIT
   EMPTY
```

To override the netlist type for more than one partition, use the override_partition_netlist_type option as many times as needed.

## --post_map[=on|off]

Limits the --export_database and --import_database options to only export and import the output of Analysis and Synthesis (quartus_map) to and from the version-compatible database.

Note: This option must be used with either the --export_database or the --import_database option.

## --remove_existing_regions_in_bottom_up_scripts[=on|off]

When this option is enabled, generated Tcl scripts contain commands that remove any LogicLock regions that exist in the project the script is being called within.

This option is enabled by default. Add the flag --remove_existing_regions_in_bottom_up_scripts=off to disable.

## --update_mif

Option to update memory content from the Memory Initialization File (.mif) or Hexadecimal (Intel-Format) File (.hex) for all RAM or CAM atoms.

This option is useful for quickly changing memory contents without requiring a full compilation. After using this option, run the Assembler (quartus_asm) to generate new programming files for the device.

## --vqm[=<.vqm file>]

Option to generate a Verilog Quartus Mapping File (.vqm) netlist.

You must specify the .vqm file name unless the name can be taken from the LOGICLOCK_INCREMENTAL_COMPILE_FILE assignment in the Quartus II Settings File (.qsf).

This option overrides the settings specified in the .qsf.

## --write_eqn_file[=<.eqn file>]

Writes equation file to the specified filename. If filename is not specified, the filename used is: <revision name>.<map or fit>.eqn.

The netlist type must be specified using "--netlist_type" option. The valid netlist types are "map" and "cmp".

Use "map" to specify the post synthesis netlist and use "cmp" for the post fitter netlist.

### Examples

```
##Write the post synthesis equation file (using default filename.)
quartus_cdb <revision name> --write_eqn_file --netlist_type=map

##Write the post fitter equation file (using default filename.)
quartus_cdb <revision name> --write_eqn_file --netlist_type=cmp

##Write post synthesis equation file to the file "my_mapper_results.eqn".
quartus_cdb <revision name> --write_eqn_file=my_mapper_results.eqn
--netlist_type=map
```

## --write_rcf_for_vqm[=on|off]

Option to write the Routing Constraints File (.rcf) for the Verilog Quartus Mapping File (.vqm) netlist.

# quartus_cpf

The Quartus II Convert Programming Files converts one programming file format to a different possible format.

Make sure you specify valid options and an input programming file to generate the new requested programming file format. Refer to help topics for more information and examples.

## Usage

```
quartus_cpf [-h | --help[=<option|topic>] | -v]

quartus_cpf -c [options] <input_file> <output_file> --- to convert file

quartus_cpf -w <option_filename> --- to create option file for configuration device

quartus_cpf -e -k <keyfile>:<key_id>[:<key_id>] <input_sof_file> <output_ekp_file> ---
to generate an encryption key programming file
```

This command supports the following options:

| Option | Page |
|---|---|
| -a=<hexadecimal number> | 2–17 |
| -c | 2–17 |
| -d=<device name> | 2–17 |
| -f=<argument file> | 2–94 |
| -g=<voltage> | 2–17 |
| -h | 2–94 |
| -k=<filename and key id> | 2–17 |
| -m=<PS\|AP\|FPP\|PPA\|PPS> | 2–17 |
| -n=<pb\|v\|p> | 2–17 |
| -o=<filename> | 2–17 |
| -q=<frequency with units> | 2–18 |
| -s=<device name> | 2–18 |
| -u=<up\|down> | 2–18 |
| -v | 2–94 |
| -w | 2–18 |
| --64bit | 2–94 |
| --configuration_mode=<PS\|AP\|FPP\|PPA\|PPS> | 2–18 |
| --convert | 2–18 |
| --count_dir=<up\|down> | 2–18 |
| --device=<device name> | 2–18 |
| --frequency=<frequency with units> | 2–19 |
| --help[=<option\|topic>] | 2–95 |
| --key=<filename and key id> | 2–19 |
| --lower_priority | 2–95 |
| --operation=<pb\|v\|p> | 2–19 |
| --option=<filename> | 2–19 |
| --sfl_device=<device name> | 2–20 |
| --start_address=<hexadecimal number> | 2–20 |
| --version | 2–95 |
| --voltage=<voltage> | 2–20 |
| --write | 2–20 |

This command includes help on the following topics:

## -a=<hexadecimal number>

Refer to the help for --start_address=<hexadecimal number> on page 2–20

## -c

Refer to the help for --convert on page 2–18

## -d=<device name>

Refer to the help for --device=<device name> on page 2–18

## -g=<voltage>

Refer to the help for --voltage=<voltage> on page 2–20

## -k=<filename and key id>

Refer to the help for --key=<filename and key id> on page 2–19

## -m=<PS|AP|FPP|PPA|PPS>

Refer to the help for --configuration_mode=<PS | AP | FPP | PPA | PPS> on page 2–18

## -n=<pb|v|p>

Refer to the help for --operation=<pb | v | p> on page 2–19

## -o=<filename>

Refer to the help for --option=<filename> on page 2–19

## -q=<frequency with units>

Refer to the help for --frequency=<frequency with units> on page 2–19

## -s=<device name>

Refer to the help for --sfl_device=<device name> on page 2–20

## -u=<up|down>

Refer to the help for --count_dir=<up|down> on page 2–18

## -w

Refer to the help for --write on page 2–20

## --configuration_mode=<PS|AP|FPP|PPA|PPS>

Option to specify the configuration mode to be used. Use this option only if you want to generate a Raw Binary File (.rbf), Tabular Text File (.ttf) or a Hexadecimal (Intel-Format) Output File (.hexout).

| Value | Options |
|---|---|
| PS | Passive Serial |
| AP | Active Parallel |
| FPP | Fast Passive Parallel |
| PPA | Passive Parallel Asynchronous |
| PPS | Passive Parallel Synchronous |

## --convert

Option to convert input file(s) to output file format.

```
# To convert a .sof into a .pof, .jic, .rbf, .ttf or .hexout
quartus_cpf -c [options or option_file] <input_sof_file> <output file type (pof | jic |
rbf | ttf | hexout)>

# To convert multiple files into a .pof, .jic, .rbf, .ttf, or .hexout,
# use a Conversion Setup File (.cof) created with the
# Convert Programming Files dialog box in the UI
quartus_cpf -c <input_cof_file>

# To convert multiple filesinto a .jam, .jbc, .svf or .isc,
# use a Chain Description File (.cdf) created with the
# Programmer tool in the UI
quartus_cpf -c <input_cdf_file> <output file type (jam | jbc | svf | isc)>
```

## --count_dir=<up|down>

Option to specify the direction for the address for the Hexadecimal (Intel-Format) Output File (.hexout). Use this option only with the "-h" or "--hexout" option.

## --device=<device name>

Option to specify the name for the configuration device.

## --frequency=<frequency with units>

Option to specify the JTAG TCK clock frequency. Use this option only if you want to generate a Serial Vector Format (.svf) for the output file.

### Example

```
# To create an SVF for programming with a 10MHz JTAG TCK clock frequency
quartus_cpf -c -q 10.0MHz -n p <input_pof_file> <output_svf_file>
```

## --key=<filename and key id>

Option to specify a key (or keys) to be used for generating secured configuration bitstreams or the Encryption Key Programming File (.ekp).

Use the following format to specify key values:

```
<keyfile>:<keyid>[:<keyid2>]
```

where
keyfile is a valid Key File (.key)
keyid is a valid id for a key in the specified file
keyid2 is a valid id for a key in the specified file

### Examples

```
# To convert .sof to a secured configuration bitstream .rbf
# using a single key file
quartus_cpf --key <keyfile>:<keyid1>:<keyid2> <input_sof_file> <output_rbf_file>
```

```
# To generate an Encryption Key Programming File (.ekp)
# using two key files
quartus_cpf --key <keyfile1>:<keyid1> --key <keyfile2>:<keyid2> <input_sof_file>
<output_ekp_file>
```

## --operation=<pb|v|p>

Option to specify the programming options. Use this option only if you want to generate a Serial Vector Format File (.svf) for the output file.

| Value | Options |
|---|---|
| p | program |
| v | verify |
| pb | program and blank check |

### Example

```
# To create an SVF for programming
quartus_cpf -c -n p <input_pof_file> <output_svf_file>
```

## --option=<filename>

Option to create the configuration device file using the specified input option file.

You can create the option file using any text editor. Use the strings and the values listed below or use the "quartus_cpf -w" command to generate the strings in the text file automatically.

Use the following format to specify string values:

```
<string>=<value>
```

The following are valid strings and values for the option file:

| String | Description |
|---|---|
| compression=on\|off | Turns the compression on or off for enhanced configuration devices. |
| clock_divisor=1 to 16\|1.5\|2.5 | Specifies the value of the clock divisor for enhanced configuration devices. |
| clock_source=internal\|external | Specifies the clock source for enhanced configuration devices. |
| clock_frequency | Specifies the clock frequency for enhanced configuration devices. Internal values: 10\|33\|50\|66 MHz External values: 1 Hz to 133.0 MHz |
| jtag_usercode | Specifies the JTAG user code value, which must be stored as a hexadecimal number, for example, jtag_usercode=abcd1234. |
| disable_pullups=on\|off | Disables the nCS and OE internal pull-ups on the configuration device. |
| memory_map_file=on\|off | Turns the memory map file generation on or off. |
| auto_usercode=on\|off | Turns the auto usercode option on or off. |
| auto_jtag_usercode_inc=on\|off | Automatically increments JTAG user code in the second and subsequent configuration devices if the target device requires multiple configuration devices. |
| use_low_voltage=on\|off | Allows an EPC1 configuration device to operate in a 3.3-V environment. |
| bitstream_compression=on\|off | Turns the bitstream compression on or off for a Cyclone™device. |

## --sfl_device=<device name>

Option to specify the serial flash loader device name. Only the Cyclone™device family supports serial flash loader.

## --start_address=<hexadecimal number>

Option to specify the start address for the Hexadecimal (Intel-Format) Output File (.hexout). Use this option only with the "-h" or "--hexout" option.

## --voltage=<voltage>

Option to specify the VCC level. Use this option only if you want to generate a Serial Vector Format File (.svf) for the output file.

### Example

```
# To create an SVF for programming with a 3.3V supply
quartus_cpf -c -g 3.3 -n p <input_pof_file> <output_svf_file>
```

## --write

Option to write the option file for the configuration device(s).

## design_security

For families that support Design Security, use the 'key' option to specify the keys to be used for generating secured configuration bitstreams or the Encryption Key Programming File (.ekp).

Use the following format to specify key values:

```
<keyfile>:<keyid>[:<keyid2>]
```

where
keyfile is a valid Key File (.key)
keyid is a valid id for a key in the specified file
keyid2 is a valid id for a key in the specified file

#### Examples

```
# To convert .sof to a secured configuration bitstream .rbf
# using a single key file
quartus_cpf --key <keyfile>:<keyid1>:<keyid2> <input_sof_file> <output_rbf_file>

# To generate an Encryption Key Programming File (.ekp)
# using two key files
quartus_cpf --key <keyfile1>:<keyid1> --key <keyfile2>:<keyid2> <input_sof_file>
<output_ekp_file>
```

### hexout

To generate a Hexadecimal (Intel-Format) Output File (.hexout), specify the input file name and output file name. Make sure the file extension of the output file is .hexout. The input file can be either an SRAM Object File (.sof) or a Programmer Object File (.pof).

You can use optional arguments to specify the data start address and count direction. These arguments are not legal if you are trying to convert an enhanced configuration device .pof to a .hexout.

Optional arguments are as follows:

| | |
|---|---|
| -u \| --count_dir | Specifies the count direction for the data. |
| -a \| --start_address | Specifies the start address of the data. Make sure to enter the address as a hexadecimal number. |

#### Examples

```
# To convert .sof to .hexout
quartus_cpf -c <input_file> <output_hexout_file>

#start address = 0x200, data count direction = up
quartus_cpf -c -u up -a 0x200 <input_file> <output_hexout_file>

#start address = 0x0fffff, data count direction = down
quartus_cpf -c -u down -a 0x0fffff <input_file> <output_hexout_file>

# To use a Conversion Setup File (.cof) created with
# the Convert Programming Files dialog box in the UI
quartus_cpf -c <input_cof_file>
```

### isc

To generate an In System Configuration File (.isc), specify the input file name and output file name. The ISC File is generated for IEEE-1532 compliant or compatible devices. Make sure the file extension of the output file is .isc. The input file can be either a Programmer Object File (.pof) or a Chain Description File (.cdf).

#### Examples

```
quartus_cpf -c <input_pof_file> <output_isc_file>
quartus_cpf -c <input_cdf_file> <output_isc_file>
```

## jam

To generate a JEDEC STAPL Format File (.jam), specify the input file name and output file name. Make sure the file extension of the output file is .jam. The input file must be either an SRAM Object File (.sof), a Programmer Object File (.pof), or a Chain Description File (.cdf). Use the .cdf to generate the .jam for a multi-device chain.

### Examples

```
quartus_cpf -c <input_pof_file> <output_jam_file>
quartus_cpf -c <input_sof_file> <output_jam_file>
quartus_cpf -c <input_cdf_file> <output_jam_file>
```

## jbc

To generate a Jam STAPL Byte Code 2.0 File (.jbc), specify the input file name and output file name. Make sure the file extension of the output file is .jbc. The input file must be either an SRAM Object File (.sof), a Programmer Object File (.pof), or a Chain Description File (.cdf). Use a .cdf to generate the .jbc for a multi-device chain.

### Examples

```
quartus_cpf -c <input_pof_file> <output_jbc_file>
quartus_cpf -c <input_sof_file> <output_jbc_file>
quartus_cpf -c <input_cdf_file> <output_jbc_file>
```

## jic

To convert an SRAM Object File (.sof) to a JTAG Indirect Configuration Device Programming File (.jic), specify the input file name, configuration device name, serial flash loader device name, and output file name. Make sure the file extension of the output file is .jic. You can also generate a .jic using a Conversion Setup File (.cof) created with the Convert Programming Files dialog box in the UI. A .cof contains all the options for the configuration device along with the output .jic name.

The configuration device must be an Altera®serial configuration device and the serial flash loader device must be a Cyclone™device.

### Examples

```
# To convert .sof to .jic
quartus_cpf -c -d <config_device_name> -s <serial_flash_loader_device_name>
<input_sof_file> <output_jic_file>

# To use option file
quartus_cpf -c -o <option_file> -d <config_device_name>
-s <serial_flash_loader_device_name> <input_sof_file> <output_jic_file>

# To use .cof
quartus_cpf -c <input_cof_file>
```

## pof

To convert an SRAM Object File (.sof) to a Programmer Object File (.pof), specify the input file name, configuration device name and output file name. Make sure the file extension of the output file is .pof. You can also generate a .pof using a Conversion Setup File (.cof) created with the Convert Programming Files dialog box in the UI. A .cof contains all the options for the configuration device along with the output .pof name.

Alternatively, you can change configuration device options using an ASCII text option file. Refer to the help for the "-o" option for more information about the option file. If you do not specify an option file and a .cof, default values are used, or values are read from a .cof.

#### Examples

```
# To convert .sof to .pof
quartus_cpf -c -d <config_device_name> <input_sof_file> <output_pof_file>

# To use option file
quartus_cpf -c -o <option_file> -d <config_device_name> <input_sof_file>
<output_pof_file>

# To use .cof
quartus_cpf -c <input_cof_file>
```

### rbf

To generate a Raw Binary File (.rbf), specify the input file name and output file name. Make sure file extension of the output file is .rbf. The input file can be only an SRAM Object File (.sof).

#### Examples

```
# To convert .sof to .rbf
quartus_cpf -c <input_sof_file> <output_rbf_file>

# To use a Conversion Setup File (.cof) created with
# the Convert Programming Files dialog box in the UI
quartus_cpf -c <input_cof_file>
```

### rpd

To generate a Raw Programming Data File (.rpd), specify the input file name and output file name. Make sure the file extension of the output file is .rpd. The input file can be only a Programmer Object File (.pof).

#### Examples

```
# To convert .pof to .rpd
quartus_cpf -c <input_pof_file> <output_rpd_file>

# To use a Conversion Setup File (.cof) created with
# the Convert Programming Files dialog box in the UI
quartus_cpf -c <input_cof_file>
```

### svf

To generate a Serial Vector Format File (.svf), you must use three arguments: "-q" ("--frequency") to specify the JTAG TCK clock frequency, "-g" ("--voltage") to specify the VCC level, and "-n" ("--operation") to specify the programming operation.

Make sure to specify the units for frequency and voltage.

Use a Chain Description File (.cdf) to generate the .svf for a multi-device chain.

#### Examples

```
# To use 4.5 MHz TCK, 3.3V supply, and programming option
quartus_cpf -c -q 4.5MHz -g 3.3 -n p <input_pof_file> <output_svf_file>

# To use 10 MHz TCK, 3.3V supply, and verify option
quartus_cpf -c -q 10MHz -g 3.3 -n v <input_sof_file> <output_svf_file>

# To use 45 KHz TCK, 1.8V supply, and programming+blank_check option
quartus_cpf -c -q 45KHz -g 1.8 -n pb <input_cdf_file> <output_svf_file>
```

## ttf

To generate a Tabular Text File (.ttf), specify the input file name and output file name. Make sure the file extension of the output file is .ttf. The input file can be only an SRAM Object File (.sof).

### Examples

```
# To convert .sof to .ttf
quartus_cpf -c <input_sof_file> <output_ttf_file>
```

```
# To use a Conversion Setup File (.cof) created with
# the Convert Programming Files dialog box in the UI
quartus_cpf -c <input_cof_file>
```

## quartus_drc

The Quartus II Design Assistant checks the reliability of a design based on a set of design rules. The Design Assistant is especially useful for checking the reliability of a design before converting the design for HardCopy®devices.

The Design Assistant supports designs that target any Altera device supported by the Quartus II software.

Quartus II Analysis & Synthesis or the Fitter must be run successfully before running the Design Assistant.

### Usage

```
quartus_drc [-h | --help[=<option|topic>] | -v]

quartus_drc <project name> [<options>]

quartus_drc -t <script file> [<script args>]

quartus_drc -s

quartus_drc --tcl_eval <tcl command>
```

This command supports the following options:

| Option | Page |
|---|---|
| -c=<revision name> | 2–98 |
| -f=<argument file> | 2–94 |
| -h | 2–94 |
| -s | 2–104 |
| -t=<script file> | 2–104 |
| -v | 2–94 |
| --64bit | 2–94 |
| --hc[=on\|off] | 2–26 |
| --help[=<option\|topic>] | 2–95 |
| --lower_priority | 2–95 |
| --post_fit[=on\|off] | 2–26 |
| --post_syn[=on\|off] | 2–26 |
| --read_settings_files[=on\|off] | 2–100 |
| --rev=<revision name> | 2–98 |
| --rtl[=on\|off] | 2–26 |
| --script=<script file> | 2–104 |
| --set=<assignment=value> | 2–98 |
| --shell | 2–104 |
| --tcl_eval=<tcl command> | 2–105 |
| --version | 2–95 |
| --write_settings_files[=on\|off] | 2–102 |

This command includes help on the following topics:

| Help Topic | Page |
|---|---|
| TCL | 2–105 |
| arguments | 2–95 |
| makefiles | 2–95 |
| return_codes | 2–96 |

### --hc[=on|off]

Option to generate HardCopy®files.

This option overrides the settings specified in the Quartus II Settings File (.qsf).

### --post_fit[=on|off]

Option to perform post-fit design analysis.

### --post_syn[=on|off]

Option to perform post-synthesis design analysis.

### --rtl[=on|off]

Option to run Design Assistant in pre-synthesis (RTL) mode.

## quartus_eda

The Quartus II EDA Netlist Writer generates netlist and other output files for use with other EDA tools. Quartus II Analysis & Synthesis, the Fitter, or Timing Analyzer must be run successfully before running the EDA Netlist Writer, depending on the arguments used.

The options are grouped into two levels: top-level options and secondary options. A top-level option specifies a single task. You can specify only one top-level option.

The following top-level options are supported: --simulation, --timing_analysis, --formal_verification, --board_timing, --board_signal_integrity --board_symbol, --resynthesis, --gen_testbench, --hardcopy

For information on top-level options and corresponding secondary level options, use "--help=<option>" for the top-level option.

### Usage

```
quartus_eda [-h | --help[=<option|topic>] | -v]

quartus_eda <project name> [<options>]

quartus_eda -t <script file> [<script args>]

quartus_eda -s

quartus_eda --tcl_eval <tcl command>
```

This command supports the following options:

This command includes help on the following topics:

## --board_boundary_scan[=on|off]

A top-level option that indicates that all other options specified on the command line are meant for board level boundary scan-related file generation.

The exact type of output file(s) are specified by the secondary options --format=bsdl, and --output_directory The --output_directory option is optional.

## --board_signal_integrity[=on|off]

A top-level option that indicates that all other options specified on the command line are meant for board level signal integrity related file generation.

The exact type of output file(s) are specified by the secondary options --format=ibis and --format=hspice, and --output_directory The --output_directory option is optional.

## --board_symbol[=on|off]

A top-level option that indicates that all other options specified on the command line are meant for board level symbol related file generation.

The exact type of output file(s) are specified by the "--tool=viewdraw" and "--output_directory" options.

The --output_directory option is optional.

## --board_timing[=on|off]

A top-level option that indicates that all other options specified on the command line are meant for board level timing related file generation.

The exact type of output file(s) are specified by the "--format=stamp" and "--output_directory" options.

The --output_directory option is optional.

## --formal_verification[=on|off]

A top-level option that indicates that all other options specified on the command line are meant for formal verification-related file generation.

The exact type of output file(s) are specified by the "--tool" and the "--output_directory" option. The --output_directory option is optional.

## --format=<NONE>

Option to specify the format of a netlist or a test bench. This option is usually used with the "--tool" option.

The following format values are supported:

- verilog
- vhdl
- ibis
- hspice
- stamp
- psdf
- bsdl

This option overrides the settings specified in the Quartus II Settings File (.qsf).

## --gen_script=<NONE>

Option that tells the EDA Netlist Writer to generate a simulation command script for third-party EDA simulation tools.

This option can take three possible values:

```
rtl
gate_level
rtl_and_gate_level
```

The location of pre-compiled simulation library is specified with the option "--user_compiled_simlib_dir". This option is optional.

For more information on this option, use "--help=<option>".

This option overrides the settings specified in the Quartus II Settings File (.qsf).

## --gen_testbench

A top-level option that indicates that all other options specified on the command line are meant for HDL test bench-related file generation.

The exact type of output file(s) are specified by secondary options. These include:

```
    --tool
    --format
    --vector_source
    --testbench_file
    --check_outputs
```

All options are always optional, except "--tool" and "--format". The "--tool" and "--format" options may or may not be optional.

The "--vector_source", "--testbench_file", and "--check_outputs" options cannot be used unless the "--gen_testbench" option is used.

For more information on each option, use "--help=<option>".

## --glitch_filtering[=on|off]

Option to specify that output netlists and .sdo file be generated for glitch filtering. This option can only be used with the top-level option "--simulation".

This option overrides the settings specified in the Quartus II Settings File (.qsf).

## --output_directory=<NONE>

Option to specify the directory for generated output files.

This option overrides the settings specified in the Quartus II Settings File (.qsf).

## --resynthesis[=on|off]

A top-level option that indicates that all other options specified on the command line are meant for resynthesis- related file generation.

The exact type of output file(s) are specified by the "--tool" option.

## --simulation[=on|off]

A top-level option that indicates that all other options specified on the command line are meant for simulation-related file generation.

The exact type of output file(s) are specified by secondary options. These include:

```
    --tool
    --format
    --output_directory
    --glitch_filtering
    --no_top_vhdl_entity
    --disable_bidir_input_timing_checks
    --vhdl_architecture
    --vcd_tb_design_instance_name
    --vcd_type
    --functional
    --maintain_design_hierarchy
    --map_illegal_characters
    --short_hpath
    --timescale
    --flatten_buses
    --device_controls_as_ports
    --user_compiled_simlib_dir
```

All options are always optional, except "--tool" and "--format". The "--tool" and "--format" options may or may not be optional.

For more information on each option, use "--help=<option>".

## --timing_analysis[=on|off]

A top-level option that indicates that all other options specified on the command line are meant for timing analysis- related file generation.

The exact type of output file(s) are specified by secondary options. These include:

```
    --tool
    --format
    --output_directory
    --map_illegal_characters
    --short_hpath
    --flatten_buses
```

All options are always optional, except "--tool" and "--format". The "--tool" and "--format" options may or may not be optional.

For more information on each option, use "--help=<option>".

## --tool=<3rd-party eda tool>

Option to tell the EDA Netlist Writer to write out a netlist for the specified third-party EDA tool. You can choose the third-party EDA tool from one of the three categories of available tools: simulation, timing analysis, or board level design and analysis.

This option overrides the settings specified in the Quartus II Settings File (.qsf).

Both the tool name and format must be specified in order to generate a netlist. Available tools and their corresponding options are listed below:

| **Simulation Tool as shown in GUI** | **Command-line Options** |
|---|---|
| Tool: ModelSim Format: VHDL | --simulation --tool=modelsim --format=vhdl |
| Tool: ModelSim Format: Verilog | --simulation --tool=modelsim --format=verilog |
| Tool: ModelSim-Altera Format: VHDL | --simulation --tool=modelsim_oem --format=vhdl |
| Tool: ModelSim-Altera Format: Verilog | --simulation --tool=modelsim_oem --format=verilog |
| Tool: NC-VHDL Format: VHDL | --simulation --tool=ncsim --format=vhdl |
| Tool: NC-Verilog Format: Verilog | --simulation --tool=ncsim --format=verilog |
| Tool: VCS-MX | --simulation --tool=vcsmx |
| Tool: VCS | --simulation --tool=vcs |
| Tool: Verilog-XL | --simulation --tool=verilogxl |
| Tool: Active-HDL | --simulation --tool=activehdl |
| Tool: Riviera-PRO | --simulation --tool=rivierapro |

| **Timing Analysis Tool as shown in GUI** | **Command-line Options** |
|---|---|
| Tool: PrimeTime Format: Verilog | --timing_analysis --tool=primetime --format=verilog |

| **Board Level Symbol Tool as shown in GUI** | **Command-line Options** |
|---|---|
| Tool: Symbol Generation (ViewDraw) | --board_symbol --tool=viewdraw |

| **Board Level Timing Analysis Tool as shown in GUI** | **Command-line Options** |
|---|---|
| Tool: Stamp Generation Format: STAMP | --board_timing --format=stamp |

| **Board Level Signal Integrity Tool as shown in GUI** | **Command-line Options** |
|---|---|
| Tool: IBIS Generation Format: IBIS | --board_signal_integrity --format=ibis |
| Tool: HSPICE Generation Format: HSPICE | --board_signal_integrity --format=hspice |

| **Resynthesis Tool as shown in GUI** | **Command-line Options** |
|---|---|
| Tool: Blast FPGA | --resynthesis --tool=blast_fpga |
| Tool: Amplify | --resynthesis --tool=amplify |
| Tool: Precision Physical | --resynthesis --tool=precision |

| Formal Verification Tool as shown in GUI | Command-line Options |
|---|---|
| Tool: Conformal LEC | --formal_verification --tool=conformal |

## --user_compiled_simlib_dir=<NONE>

Option that specifies directory of a pre-compiled simulation library.

## --vcd_tb_design_instance_name=<NONE>

Option to specify the hierarchical path to the instance of the design in the testbench. This value is used in the VCD generation script

The value should be an absolute hierarchical path. For example "--vcd_tb_instance_name=/tb/u1".

## --vcd_type=<NONE>

Option to specify the type of VCD output file This option can take three possible values:

```
all
no_comb
none
```

This option overrides the settings specified in the Quartus II Settings File (.qsf).

## quartus_fit

The Quartus II Fitter performs place and route by fitting the logic of a design into a device. The Fitter selects appropriate interconnection paths, pin assignments, and logic cell assignments.

Quartus II Analysis & Synthesis must be run successfully before running the Fitter.

### Usage

```
quartus_fit [-h | --help[=<option|topic>] | -v]
```

```
quartus_fit <project name> [<options>]
```

This command supports the following options:

This command includes help on the following topics:

## --check_ios

Option to run until I/O placement is determined. This process includes placing all blocks in the periphery, such as PLLs, serializers, deserializers, and gigabit transceiver blocks (GXB).

The report file and the floorplan display I/O placement results. If all pins cannot be placed, the report file and floorplan display partial placement, results, and error messages to indicate why placement failed.

The "--check_ios" option should not be used when you use the "--io_smart_recompile" option. For example, after doing a complete place and route, if you change an I/O standard, it is advisable to use the "--io_smart_recompile" option, because the "--check_ios" option destroys the original place and route results.

## --check_netlist

Option to run only legality checking on the current netlist. Analysis & Synthesis (quartus_map) must be run successfully before you use this option. Currently changes made to placement or routing are not verified -- only functional changes (for example, I/O standards) are checked.

This option can be used to verify that the netlist is legal after you make changes using the Chip Editor.

## --early_timing_estimate[=<realistic|optimistic|pessimistic>]

Option to run an Early Timing Estimate. An Early Timing Estimate is an estimate of timing results for your design before performing full placement and routing. This feature runs the fitter up to 10 times faster than a full fit and generates a full timing report based on estimated delays for the design. The fit is not fully optimized or routed, and hence the timing report is only an estimate. Typically, the estimated delays are within 20% of what a full compilation can achieve.

The following table describes the types of timing estimates:

| Value | Description |
|---|---|
| realistic | Estimates delays that will likely be close to a full compilation's results. (default value) |
| optimistic | Estimates delays that are lower than those likely to be achieved by a full compilation. This makes the estimate of performance optimistic. |
| pessimistic | Estimates delays that are higher than those likely to be achieved by a full compilation. This makes the estimate of performance pessimistic. |

All Early Timing Estimate types have the same reduction of compilation time.

After successfully running "quartus_fit --early_timing_estimate," "quartus_tan --timing_analysis_only" must be run to generate a timing report.

## --effort=<standard|fast|auto>

Option to specify the level of effort you want the Fitter to use.

The following table describes level of effort values:

| **Value** | **Description** |
|---|---|
| standard | Directs the Fitter not to decrease effort. Preserves fmax but does not decrease compilation time. |
| fast | Directs the Fitter to decrease effort. Decreases compilation time by up to 50%, with a possible reduction in fmax. |
| auto | Directs the Fitter to reduce effort after meeting timing requirements. Decreases compilation time only when timing and fitting requirements can be met. |

## --fmax=<time unit>

Option to specify the fmax time value.

Fmax is the minimum acceptable clock frequency, that is, the maximum clock frequency that can be achieved without violating internal setup and hold time requirements.

Example usage:

```
quartus_fit one_wire --fmax=155.55mhz
```

The format is "<floating point time value><time unit>". In this example, "155.55" is the <floating point time value> and "mhz" is the <time unit>.

The following table displays possible time units:

| **Time Unit** | **Description** |
|---|---|
| s | second(s) |
| ms | millisecond(s) |
| us | microsecond(s) |
| ns | nanosecond(s) |
| ps | picosecond(s) |
| fs | femtosecond(s) |
| hz | hertz |
| khz | kilohertz |
| mhz | megahertz |
| ghz | gigahertz |

## --incremental_signaltap

Option to perform an incremental SignalTap®II compilation. Use this option when only SignalTap settings have changed since the last compilation.

This option cannot be used with most of the other quartus_fit options.

## --inner_num=<value>

Option to specify a value for the loop multiplier "inner_num" used during placement. Analysis & Synthesis (quartus_map) must be run successfully before you use this option. Use of a higher value increases compilation time, but may increase the quality of placement.

## --io_smart_recompile

Option to recompile the design for changed I/O assignments without repeating the entire Fitter flow. Analysis & Synthesis (quartus_map) must be run successfully before you use this option. You can recompile only with I/O assignment changes.

This option allows you to recompile the design quickly because only I/O changes and legality checks are run to determine if the new I/O assignments are compatible with the current post-fitting netlist.

## --one_fit_attempt[=on|off]

Option to perform only one fitting attempt, giving a no fit if that attempt fails. When this option is turned off, the Fitter may perform additional attempts.

## --optimize_io_register_for_timing[=on|off]

Option to optimize I/O register placement for timing. This option is used for timing-driven compilation.

## --pack_register=<off|normal|minimize_area|minimize_area_with_chains|auto>

Option to implement register packing for appropriate pairs of registers and logic functions.

The following table describes register packing values:

| Value | Description |
|---|---|
| off | The Fitter does not attempt to place a pair of logic functions in a single logic cell. |
| normal | The Fitter places both a combinational and a sequential operation in a logic cell when it is expected that the placement will not affect performance. |
| minimize area | The Fitter aggessively combines unrelated sequential and combinational functions into a single logic cell to reduce the logic element count, even at the expense of performance. |
| minimize area with | The Fitter aggressively combines sequential |
| chains | and combinational functions that are part of arithmetic or register cascade chains, or that can be converted to register cascade chains. |
| auto | The Fitter automatically chooses the best method to fit the design. |

The following table displays the device families that support the various values:

| Value | Device Family |
|---|---|
| off | All |
| normal | All |
| minimize_area | All |

| **Value** | **Device Family** |
|---|---|
| minimize_area_with_chains | Cyclone™, Cyclone II, Stratix®, Stratix II |
| auto | Cyclone II, Stratix II |

## --part=<device part>

Option to use the specified device.

This option overrides the settings specified in the Quartus II Settings File (.qsf) or the part used in Analysis & Synthesis (quartus_map). The specified part must be in the same device family used in Analysis & Synthesis.

## --seed=<value>

Option to use the specified seed value.

The Fitter uses the seed as the initial placement configuration when attempting to optimize the design's timing requirements, including fmax.

## --signalprobe

Option to perform an incremental SignalProbe™compilation. Use this option when only SignalProbe settings have changed since the last compilation.

This option cannot be used with most of the other quartus_fit options.

## --tco=<time unit>

Option to specify the tco time value.

Tco is the maximum acceptable clock to output delay to the output pin. The clock to output delay is the time required to obtain a valid output at an output pin that is fed by a register after a clock signal transition on an input pin that clocks the register. This time always represents an external pin-to-pin delay.

Example usage:

```
    quartus_fit one_wire --tco=10.55ns
```

The format is "<floating point time value><time unit>". In the example, "10.55" is the <floating point time value> and "ns" is the <time unit>.

The following table displays possible time units:

| **Time Unit** | **Description** |
|---|---|
| s | second(s) |
| ms | millisecond(s) |
| us | microsecond(s) |
| ns | nanosecond(s) |
| ps | picosecond(s) |
| fs | femtosecond(s) |
| hz | hertz |

| Time Unit | Description |
|---|---|
| khz | kilohertz |
| mhz | megahertz |
| ghz | gigahertz |

## --tdc[=on|off]

Option to use timing-driven compilation. This option optimizes place and route based on timing information.

## --tpd=<time unit>

Option to specify the tpd time value.

Tpd is the maximum acceptable input to non-registered output delay, that is, the time required for a signal from an input pin to propagate through combinatorial logic and appear at an output pin.

Example usage:

```
quartus_fit one_wire --tpd=20.55ns
```

The format is "<floating point time value><time unit>". In this example, "20.55" is the <floating point time value> and "ns" is the <time unit>.

The following table displays possible time units:

| Time Unit | Description |
|---|---|
| s | second(s) |
| ms | millisecond(s) |
| us | microsecond(s) |
| ns | nanosecond(s) |
| ps | picosecond(s) |
| fs | femtosecond(s) |
| hz | hertz |
| khz | kilohertz |
| mhz | megahertz |
| ghz | gigahertz |

## --tsu=<time unit>

Option to specify the tsu time value.

Tsu is the maximum acceptable clock setup time for the input (data) pin. The setup time is the length of time for which data that feeds a register via its data or enable input(s) must be present at an input pin before the clock signal that clocks the register is asserted at the clock pin.

Example usage:

```
quartus_fit one_wire --tsu=7.55ns
```

The format is "<floating point time value><time unit>". In this example, "7.55" is the <floating point time value> and "ns" is the <time unit>.

The following table displays possible time units:

| **Time Unit** | **Description** |
|---|---|
| s | second(s) |
| ms | millisecond(s) |
| us | microsecond(s) |
| ns | nanosecond(s) |
| ps | picosecond(s) |
| fs | femtosecond(s) |
| hz | hertz |
| khz | kilohertz |
| mhz | megahertz |
| ghz | gigahertz |

# quartus_jbcc

The Quartus II JAM Compiler converts Jam/STAPL files (JAM) to Jam/STAPL Byte Code files (JBC).

## Usage

```
quartus_jbcc [-h | --help[=<option|topic>] | -v]
```

```
quartus_jbcc [-c] [-d] <input.jam> <output.jbc>
```

This command supports the following options:

| Option | Page |
|---|---|
| -c | 2–41 |
| -d | 2–41 |
| -f=<argument file> | 2–94 |
| -h | 2–94 |
| -v | 2–94 |
| --64bit | 2–94 |
| --compress | 2–41 |
| --debug | 2–41 |
| --help[=<option\|topic>] | 2–95 |
| --lower_priority | 2–95 |
| --version | 2–95 |

This command includes help on the following topics:

| Help Topic | Page |
|---|---|
| arguments | 2–95 |
| makefiles | 2–95 |
| return_codes | 2–96 |

### -c

Refer to the help for --compress on page 2–41

### -d

Refer to the help for --debug on page 2–41

### --compress

Enable compression of large boolean arrays.

### --debug

Enable inclusion of debug information.

## quartus_jli

The Quartus II JBI Player executes Jam/STAPL files (JBC).

### Usage

```
quartus_jli [-h | --help[=<option|topic>] | -v]
quartus_jli -n --- list available hardware
quartus_jli -i <jam file> --- list available actions
quartus_jli [-c <cable>] <jam/jbc file> -a <action> [-d <proc>] [-e <proc>]
```

This command supports the following options:

| Option | Page |
|---|---|
| -a=<action to perform> | 2–42 |
| -c=<jtagserver cable number> | 2–43 |
| -d=<procedure to disable> | 2–43 |
| -e=<procedure to enable> | 2–43 |
| -f=<argument file> | 2–94 |
| -h | 2–94 |
| -i | 2–43 |
| -j | 2–43 |
| -l | 2–43 |
| -n | 2–43 |
| -v | 2–94 |
| --64bit | 2–94 |
| --action=<action to perform> | 2–43 |
| --cable=<jtagserver cable number> | 2–43 |
| --disable=<procedure to disable> | 2–43 |
| --dont_reset_jtag | 2–43 |
| --enable=<procedure to enable> | 2–43 |
| --enumerate_hardware | 2–44 |
| --help[=<option \| topic>] | 2–95 |
| --info | 2–44 |
| --loquacious | 2–44 |
| --lower_priority | 2–95 |
| --version | 2–95 |

This command includes help on the following topics:

| Help Topic | Page |
|---|---|
| arguments | 2–95 |
| makefiles | 2–95 |
| return_codes | 2–96 |

### -a=<action to perform>

Refer to the help for --action=<action to perform> on page 2–43

### -c=<jtagserver cable number>

Refer to the help for --cable=<jtagserver cable number> on page 2–43

### -d=<procedure to disable>

Refer to the help for --disable=<procedure to disable> on page 2–43

### -e=<procedure to enable>

Refer to the help for --enable=<procedure to enable> on page 2–43

### -i

Refer to the help for --info on page 2–44

### -j

Refer to the help for --dont_reset_jtag on page 2–43

### -l

Refer to the help for --loquacious on page 2–44

### -n

Refer to the help for --enumerate_hardware on page 2–44

### --action=<action to perform>

Option to specify the action to perform.

Use -i to display the actions supported by the Jam/STAPL ByteCode file.

### --cable=<jtagserver cable number>

Specify which cable connected to the local JTAGserver to use.

Use -n to list available hardware.

### --disable=<procedure to disable>

Option to specify the procedure to disable (-d <procedure to disable> or --disable==<procedure to disable>).

Use -i to display an action's procedures.

### --dont_reset_jtag

Disables resetting the JTAG TAP controller after execution is completed.

### --enable=<procedure to enable>

Option to specify the procedure to perform (-e <procedure to enable> or --enable==<procedure to enable>).

Use -i to display an procedures for a given action.

## --enumerate_hardware

Display available JTAG hardware.

## --info

Displays information on the Jam/STAPL ByteCode file

## --loquacious

Displays verbose information while executing the Jam/STAPL ByteCode file

## quartus_map

Quartus II Analysis & Synthesis builds a single project database that integrates all the design files in a design entity or project hierarchy. Analysis & Synthesis includes Quartus II Integrated Synthesis, which provides comprehensive Verilog HDL and VHDL language support, as well as support for Altera-specific languages such as AHDL.

### Usage

```
quartus_map [-h | --help[=<option|topic>] | -v]
```

```
quartus_map <project name> [<options>]
```

This command supports the following options:

| Option | Page |
|---|---|
| -c=<revision name> | 2–98 |
| -f=<argument file> | 2–94 |
| -h | 2–94 |
| -l=<path> | 2–46 |
| -v | 2–94 |
| --64bit | 2–94 |
| --analysis_and_elaboration | 2–47 |
| --analyze_file=<design file> | 2–47 |
| --convert_bdf_to_verilog=<.bdf file> | 2–47 |
| --convert_bdf_to_vhdl=<.bdf file> | 2–47 |
| --effort=<auto \| fast> | 2–47 |
| --enable_wysiwyg_resynthesis[=on \| off] | 2–47 |
| --family=<device family> | 2–47 |
| --generate_cmp_file=<design file> | 2–47 |
| --generate_functional_sim_netlist | 2–48 |
| --generate_inc_file=<design file> | 2–48 |
| --generate_inst_file=<design file> | 2–48 |
| --generate_symbol=<design file> | 2–48 |
| --help[=<option \| topic>] | 2–95 |
| --ignore_carry_buffers[=on \| off] | 2–48 |
| --ignore_cascade_buffers[=on \| off] | 2–48 |
| --incremental_compilation=<off \| full_incremental_compilation> | 2–48 |
| --lib_path=<path> | 2–48 |
| --lower_priority | 2–95 |
| --optimize=<area \| speed \| balanced> | 2–48 |
| --parallel[=on \| off] | 2–49 |
| --part=<device> | 2–49 |
| --partition=<NONE> | 2–49 |
| --restructure_multiplexers=<auto \| on \| off> | 2–49 |
| --rev=<revision name> | 2–98 |
| --set=<assignment=value> | 2–98 |
| --source=<source file> | 2–50 |
| --state_machine_encoding=<auto \| minimal_bits \| one_hot \| user_encoded> | 2–50 |
| --timing_driven_synthesis[=on \| off] | 2–50 |
| --update_wysiwyg_parameters | 2–50 |
| --verilog_macro=<NONE> | 2–50 |
| --version | 2–95 |

This command includes help on the following topics:

| Help Topic | Page |
|---|---|
| arguments | 2–95 |
| makefiles | 2–95 |
| return_codes | 2–96 |

## -l=<path>

Refer to the help for --lib_path=<path> on page 2–48

### --analysis_and_elaboration

Option to check all the design files in a design for syntax and semantic errors, and perform a netlist extraction.

### --analyze_file=<design file>

Option to check the specified design file for syntax and semantic errors.

### --convert_bdf_to_verilog=<.bdf file>

Option to create a Verilog Design File (.v) for the specified Block Design File (.bdf).

### --convert_bdf_to_vhdl=<.bdf file>

Option to create a VHDL Design File (.vhd) for the specified Block Design File (.bdf).

### --effort=<auto|fast>

Option to select synthesis effort level.

The following table displays available values:

| Value | Description |
|---|---|
| auto | Maximum synthesis effort. This is the default value. |
| fast | Synthesis process is streamlined to improve runtime at the cost of design performace and/or resource usage. Use this option when the Fitter early_timing_estimate mode is used, or when a fast-synthesis compilation is needed without the need to run the Fitter. When this option is used with the regular Fitter, Fitter performance may decrease as fast-synthesis netlists take longer to route. |

### --enable_wysiwyg_resynthesis[=on|off]

Option to unmap WYSIWYG primitives during synthesis and remap the gates back to WYSIWYG LCELL primitives.

This option is not applicable if Quartus II Integrated Synthesis is used.

### --family=<device family>

Option to target the specified device family. If the "--part" option is not used, the part is set to Auto.

The family name should not contain any spaces, for example, --family=APEXII. If you need to add space between words in the family name, make sure that you enclose the words in double quotation marks "", for example, --family="APEX II".

### --generate_cmp_file=<design file>

Option to create a default VHDL Component File (.cmp) that represents the entities in the specified Text Design File (.tdf), VHDL Design File (.vhd), Verilog Design File (.v), EDIF Input File (.edf), or Block Design File (.bdf), CusP file (.cpp) and MATLAB File (.mdl).

### --generate_functional_sim_netlist

Option to prepare the databases necessary for functional simulation.

### --generate_inc_file=<design file>

Option to create a default AHDL Include File (.inc) that represents the entities in the specified Text Design File (.tdf), VHDL Design File (.vhd), Verilog Design File (.v), EDIF Input File (.edf), or Block Design File (.bdf).

### --generate_inst_file=<design file>

Option to create a default Verilog Instantiation File (.inst) that represents the entities in the specified Text Design File (.tdf), VHDL Design File (.vhd), Verilog Design File (.v), EDIF Input File (.edf), Block Design File (.bdf), CusP file (.cpp) or MATLAB file (.mdl).

### --generate_symbol=<design file>

Option to create a Block Symbol File (.bsf) that represents the entities in the specified Text Design File (.tdf), VHDL Design File (.vhd), Verilog Design File (.v), EDIF Input File (.edf), or Block Design File (.bdf).

### --ignore_carry_buffers[=on|off]

Option to ignore CARRY_SUM buffers that are instantiated in the design. (This option also applies to MAX+PLUS II-style CARRY buffers.)

### --ignore_cascade_buffers[=on|off]

Option to ignore CASCADE buffers that are instantiated in the design.

### --incremental_compilation=<off|full_incremental_compilation>

Option to specify the incremental compilation mode.

The following table displays available values:

| Value | Description |
|---|---|
| off | Turn off incremental compilation. |
| full_incremental_compilation | Turn on full incremental compilation. |

### --lib_path=<path>

Option to use the specified library paths to find the design files of the project. For multiple library paths, use --lib_path=path1 --lib_path=path2 or --lib_path="path1;path2".

### --optimize=<area|speed|balanced>

Option to optimize the design to achieve maximum speed performance, minimum area usage, or high speed performance with miminal area cost during synthesis.

The following table displays available values:

| **Value** | **Description** |
|---|---|
| area | Makes the design as small as possible in order to minimize resource usage. |
| speed | Chooses a design implementation that has the fastest fmax. |
| balanced | Chooses a design implementation that has a high-speed performance with minimal logic usage. |

Note that the current version of the Quartus II software does not support the "balanced" setting for the following devices:

Mercury(TM), MAX(R) 7000B/7000AE/3000A/7000S/7000A,
FLEX(R) 6000, FLEX 10K(R), FLEX 10KE/10KA, and ACEX 1K.

## --parallel[=on|off]

Runs quartus_map in a mode that enables parallel synthesis of partitions using the number of processors specified by the Quartus II parallel compilation option.

## --part=<device>

Option to target the specified device. This option overrides the "--family" option or family assignment.

## --partition=<NONE>

Specifies a partition to synthesize manually.

This option overrides the netlist type and preservation level of the partition and disables any automatic resynthesis of other partitions, even if they require synthesis because of changes to your design.

You can specify a partition ID or name. For example, the root partition has ID 0 and name "Top".

To synthesize multiple partitions, use a separate option for each partition.

## --restructure_multiplexers=<auto|on|off>

Option to repack fragmented multiplexers more efficiently for area in designs targeting supported device familes

(Stratix(R), Stratix GX, Cyclone(TM), Stratix II, Cyclone II, MAX(R) II).

The Multiplexer Statistics table in the Analysis & Synthesis Optimization Results section of the Compilation Report provides an estimate of the potential area saving for each multiplexer bus in the design. In some cases, restructuring results in a 20% reduction in logic elements.

Note that the speed of the design may change as a result of the multiplexer restructuring.

The following table displays available values:

| **Value** | **Description** |
|---|---|
| auto | Restructures multiplexers if you optimize for area or for balanced optimization, and selectively restructures multiplexers ff you optimize for speed. For more information, refer to the --optimize option. |

| Value | Description |
|---|---|
| on | Restructures multiplexers. |
| off | Does not restructure multiplexers. |

Note that the current version of the Quartus II software. supports multiplexer restructuring only for the following devices:

## --source=<source file>

Option to use the specified source file. Add only one source file per tag. For multiple source files, use "--source=file1 --source=file2".

If you specify a relative path, the path must be relative to the project directory.

## --state_machine_encoding=<auto|minimal_bits|one_hot|user_encoded>

Option to set the state machine processing style used to compile a state machine.

The following table displays available values:

| Value | Description |
|---|---|
| auto | Allows Analysis & Synthesis to choose the best encoding for the state machine. |
| minimal_bits | Uses the minimal number of bits to encode the state machine. |
| one_hot | Encodes the state machine in the one-hot style. |
| user_encoded | Encodes the state machine in the manner specified by the user. |

## --timing_driven_synthesis[=on|off]

Option to make synthesis take timing constraints into account. The default value is off. When this option is turned on, synthesis runs timing analysis to obtain timing information about the netlist, and optimizes the netlist accordingly.

## --update_wysiwyg_parameters

Option to update, when possible, the parameters of a changed WYSIWYG PLL or CDR primitive in the Compiler netlists. This option assumes that the previous compilation was successful.

When you use this option, the quartus_map executable gives a message stating the next executable you need to run in order to complete a compilation, quartus_fit or quartus_asm. If it is not possible to update the parameters, the quartus_map executable runs normally and gives a message that you need to run quartus_fit.

When you use this option, all assignment changes you made since the last compilation are lost.

## --verilog_macro=<NONE>

Option to set a Verilog macro. Use the following format:

```
--verilog_macro="my_macro=2"
--verilog_macro="SUM(a,b)=(a+b)"
--verilog_macro="my_str_macro=\"string2\""
```

Those are equivalent to the following `define statements:

```
`define my_macro 2
`define SUM(a,b) (a+b)
`define my_str_macro "string2"
```

## quartus_pgm

The Quartus II Programmer programs Altera®devices. The Programmer uses one of the valid supported file formats: Programmer Object Files (.pof), SRAM Object Files (.sof), Jam File (.jam), or Jam Byte-Code File (.jbc).

Make sure you specify a valid programming mode, programming cable, and operation for a given device.

### Usage

```
quartus_pgm [-h | --help[=<option|topic>] | -v]

quartus_pgm -c <cable name> filname.cdf  --- If you want to use cdf file

quartus_pgm -c <cable name> -m <programming mode> -o <value> [-o <value>...] --- If you
want to use individual programming file(s)

quartus_pgm -l --- to display the list of available hardware

quartus_pgm -c <cable name> -a --- to display the list of devices connected to the cable
```

This command supports the following options:

| Option | Page |
|---|---|
| -a | 2–53 |
| -b | 2–53 |
| -c=<cable name> | 2–53 |
| -f=<argument file> | 2–94 |
| -h | 2–94 |
| -i | 2–53 |
| -l | 2–53 |
| -m=<programming mode> | 2–53 |
| -o=<programming operation> | 2–53 |
| -v | 2–94 |
| -z | 2–53 |
| --64bit | 2–94 |
| --auto | 2–53 |
| --bgp | 2–53 |
| --cable=<cable name> | 2–53 |
| --haltcc | 2–54 |
| --help[=<option \| topic>] | 2–95 |
| --initcfg | 2–54 |
| --list | 2–54 |
| --lower_priority | 2–95 |
| --mode=<programming mode> | 2–54 |
| --operation=<programming operation> | 2–54 |
| --version | 2–95 |

This command includes help on the following topics:

| Help Topic | Page |
|---|---|
| arguments | 2–95 |
| makefiles | 2–95 |
| return_codes | 2–96 |

### -a

Refer to the help for --auto on page 2–53

### -b

Refer to the help for --bgp on page 2–53

### -c=<cable name>

Refer to the help for --cable=<cable name> on page 2–53

### -i

Refer to the help for --initcfg on page 2–54

### -l

Refer to the help for --list on page 2–54

### -m=<programming mode>

Refer to the help for --mode=<programming mode> on page 2–54

### -o=<programming operation>

Refer to the help for --operation=<programming operation> on page 2–54

### -z

Refer to the help for --haltcc on page 2–54

### --auto

Option to detect and display all the devices in the device chain.

### --bgp

Allows a MAX II device to continue to run in-system while new programming data loads into the configuration flash memory (CFM). When you turn on this option, programming data loaded into the CFM does not immediately configure the device.

### --cable=<cable name>

Option to specify which programming hardware or cable to use.

The full syntax is as follows (depending on whether the hardware is on your local machine or a remote machine):

```
"<cable_name> [<port>]"
"<cable_name> on <host_name/IP_address> [<port>]"
```

You don't need to specify the hostname or port if they are unambiguous so just specifying the name of the cable will be sufficient if there is only one cable of that type available (on a local or remote machine).

The following syntax is supported for backward compatibility, as is the cable index (the number returned by jtagconfig to identify a cable):

```
<host_name/IP_address>::<cable_name>[<port>]@<baud_rate>
```

Examples:

```
ByteBlasterMV
"byteblaster [lpt1]"
"USB-Blaster on remote-machine [com1]"
APU
```

## --haltcc

Halts the on-chip auto-configuration controller of the device to allow programming via the JTAG interface.

## --initcfg

Specifies that configuration devices will configure attached devices automatically after the Programmer finishes programming the configuration devices.

## --list

Option to display all the available programming hardware cables.

## --mode=<programming mode>

Option to specify which programming mode to use.

Use one of the following programming mode values:

| Value | Mode |
|---|---|
| JTAG | JTAG programming |
| PS | Passive Serial programming |
| AS | Active Serial programming |
| SD | In-Socket programming |

## --operation=<programming operation>

Option to specify which programming operation(s) to perform on the device(s).

Use the following syntax for each device in a device chain:

```
-o <options>;<input_file>@<device_index>
```

Note: The device index starts with 1.

Exceptions to this syntax occur when you use the following options:

```
-o E;<output_file>;<device_name>@<device_index>
-o S;<device_name/input_file>@<device_index>
```

`<options> must be one of the following combinations:`

```
    P, BP, PV, BPV,
    PL, BPL, PVL, BPVL
    CP, CBP, CPV, CBPV,
    CPL, CBPL, CPVL, CBPVL
    IP, IBP, IPV, IBPV,

    V, CV, VL, CVL, IV

    B, CB, IB

    R, RB, CR, CRB, IR, IRB

    E, CE, IE

    L, CL

    S
where:
```

| Option | Description |
|---|---|
| P | Program |
| R | Erase |
| L | Lock/Security Bit |
| I | Initialize Bridge Device* |
| V | Verify |
| B | Blank-check |
| C | ISP Clamp |
| E | Examine** |
| S | Skip/Bypass** |

* Serial FLASH Loader option only
** Cannot be used in combination with other options

Note:

Specifying a <device_index> is optional, but if you specify a <device_index> for one device, you must specify a <device_index> for all devices. You cannot specify a <device_index> for devices in a Passive Serial chain. Each device in a multi-device chain must have a corresponding -o construction.

## Examples

| Behavior | Option Syntax |
|---|---|
| JTAG Program | -o pvb;file.pof -o pvbi;file.jic |
| JTAG Examine | -o e;file.pof;device_name -o ei;file.jic;device_name |
| Skip Device (JTAG Bypass | -o s;device_name |
| Passive Serial Program | -o file.sof |
| Active Serial Program | -o pl;file.pof |
| Passive Serial Chain | -o file1.sof -o file2.sof -o file3.sof |
| JTAG Chain | -o p;file1.pof -o s;file2.pof -o v;file1.pof@1 -o p;file2.pof@2 |
| CDF | quartus_pgm -c byteblastermv[lpt1] file.cdf |

## quartus_pow

The Quartus(R) II PowerPlay Power Analyzer estimates the thermal dynamic power and the thermal static power consumed by the design. For newer families such as Stratix II and MAX II, the current drawn from each power supply is also estimated.

Quartus II Analysis & Synthesis and the Fitter must be run successfully before running the PowerPlay Power Analyzer.

### Usage

```
quartus_pow [-h | --help[=<option|topic>] | -v]
```

```
quartus_pow <project name> [<options>]
```

This command supports the following options:

| Option | Page |
|---|---|
| -c=<revision name> | 2–98 |
| -f=<argument file> | 2–94 |
| -h | 2–94 |
| -v | 2–94 |
| --64bit | 2–94 |
| --default_input_io_toggle_rate=<toggle rate value> | 2–57 |
| --default_toggle_rate=<toggle rate value> | 2–57 |
| --estimate_power[=on \| off] | 2–57 |
| --help[=<option \| topic>] | 2–95 |
| --input_saf[=<SAF Filename>] | 2–57 |
| --input_vcd[=<VCD Filename>] | 2–58 |
| --lower_priority | 2–95 |
| --no_input_file | 2–58 |
| --output_epe=<EPE Filename> | 2–58 |
| --output_saf=<SAF Filename> | 2–58 |
| --read_settings_files[=on \| off] | 2–100 |
| --rev=<revision name> | 2–98 |
| --set=<assignment=value> | 2–98 |
| --use_vectorless_estimation[=on \| off] | 2–58 |
| --vcd_filter_glitches[=on \| off] | 2–59 |
| --version | 2–95 |
| --voltage=<value_in_mV> | 2–59 |
| --write_settings_files[=on \| off] | 2–102 |

This command includes help on the following topics:

| Help Topic | Page |
|---|---|
| arguments | 2–95 |
| makefiles | 2–95 |
| return_codes | 2–96 |

## --default_input_io_toggle_rate=<toggle rate value>

Option to specify a default toggle rate to be used on input I/O pin signals during power analysis. This value is used if an input I/O pin's toggle rate is not specified by some other mean such as an input file or user assignment. To specify a default toggle rate for all other signals in the design use the --default_toggle_rate command line option.

This value can be specified as a percenatge or an absolute value. If specified as an absolute value, the unit is transitions/s.

#### Examples

```
--default_input_io_toggle_rate=12.5%
--default_input_io_toggle_rate=150000transitions/s
--default_input_io_toggle_rate="150000 transitions/s"
```

This command line option can only appear once. If this command line option is not used, then the value stored in the Quartus II Settings File (.qsf) is used to determine the default input I/O pin toggle rate.

Note: The default static probability value used by the PowerPlay Power Analyzer is 0.5.

## --default_toggle_rate=<toggle rate value>

Option to specify a default toggle rate to be used for all output signals except input I/O pin signals during power analysis. This value is used if a signal's toggle rate is not specified by some other mean such as an input file or user assignment and vectorless estimation should not be used.

This value is specified as a percenatge or as an absolute value. If specified as an absolute value, the unit is transitions/s.

#### Examples

```
--default_toggle_rate=12.5%
--default_toggle_rate=150000transitions/s
--default_toggle_rate="150000 transitions/s"
```

This command line option can only appear once. If this command line option is not used, then the value stored in the Quartus II Settings File (.qsf) is used to determine the default toggle rate.

Note:

The default static probability value used by the PowerPlay Power Analyzer is 0.5.

## --estimate_power[=on|off]

Option to specify whether a power estimate should be produced.

Specifying a value of "off" reduces processing time. For example, specify the value "off" for this option if the only desired action is to process a Value Change Dump (VCD) file to produce a Signal Activity File (SAF).

By default, a power estimate is produced.

## --input_saf[=<SAF Filename>]

Option to use the specified Signal Activity File (.saf) as input. The SAF contains toggle rates and static probabilities for output signals in the design. If no filename is specified then the filename stored in the Quartus II Settings File (.qsf) is used or if no filename exists in thethe QSF, then <revision name>.saf is used. This command line option can only appear once.

The input_saf option should not be used if either the no_input_file or input_vcd option is specified. If neither the no_input_file, input_saf or input_vcd option is specified, then the settings in the QSF are used to determine the behavior of the Power Analyzer.

## --input_vcd[=<VCD Filename>]

Option to use the specified VCD File (.vcd) as input. If no filenames are specified then the filenames stored in the Quartus II Settings File (.qsf) are used or if no filenames exist in the QSF, then <revision name>.vcd will be used. This command line option can appear multiple times in the case that multiple VCD files are required.

The input_vcd option should not be used if either the no_input_file or input_saf option is specified. If none of the no_input_file, input_saf or input_vcd option is specified, then the settings in the QSF are used to determine the behavior of the Power Analyzer.

## --no_input_file

Option to instruct the Power Analyzer not to use an input file to initialize the toggle rates and static probabilities for output signals in the design.

The no_input_file option should not be used if either the input_saf or input_vcd option is specified. If neither the no_input_file, input_saf or input_vcd option is specified, then the settings in the Quartus II Settings File (.qsf) are used to determine the behavior of the Power Analyzer.

## --output_epe=<EPE Filename>

Option to write an Early Power Estimation file, summarizing the resources used by the design. The file can be used to import design information into the PowerPlay Early Power Estimator spreadsheet available from the Altera website.

The design must be compiled before the Early Power Estimator file can be written.

This command line option may only appear once.

## --output_saf=<SAF Filename>

Option to write out the toggle rates and static probabilities used by the Power Analyzer during the power analysis to the specified Signal Activity File (.saf). This command line option can only appear once.

## --use_vectorless_estimation[=on|off]

Option to specify whether or not vectorless estimation should be used to calculate unspecified toggle rates and static probabilities for the output signals in the design. If set to "on" then vectorless estimation is used by the PowerPlay Power Analyzer and the --default_toggle_rate command line option or the value stored in the Quartus II Settings File (.qsf) will be ignored. If set to "off" then the PowerPlay Power Analyzer uses the value specified by the command line option --default_toggle_rate or the value stored in the Quartus II Settings File (.qsf) as the default toggle rate.

This command line option can only appear once. If this command line option is not used, then the value stored in the Quartus II Settings File (.qsf) is used to determine whether or not vectorless estimation is used. This command line option only applies to the Stratix II and MAX II families and is ignored for all other families. For all other families the behaviour is equivalent to this command line option being set to "off".

Note: Regardless of the setting of this option, all unspecified toggle rates for input I/O pin signals use the default toggle rate specified by either the command line option --default_input_io_toggle_rate or the value stored in the Quartus II Settings File (.qsf).

The default static probability value used by the PowerPlay Power Analyzer is 0.5.

## --vcd_filter_glitches[=on|off]

Option to use glitch filtering when reading VCD Files (.vcd) as input. This command line option can only appear once. If this command line option is not used, the value stored in the Quartus II Settings File (.qsf) is used to determine whether or not glitch filtering is used when reading VCD Files. This option has no effect if the "input_vcd" option is not used.

## --voltage=<value_in_mV>

Option to specify the device voltage (mV) when running the PowerPlay Power Analyzer.

## quartus_sh

The Quartus II Shell is a simple Quartus II Tcl interpreter. The Shell has a smaller memory footprint than the other command-line executables that support Tcl: quartus_tan, quartus_cdb, and quartus_sim. The Shell can be started with a Tcl script to evaluate, used as an interactive Tcl interpreter (shell), or used as a quick Tcl command evaluator, evaluating the remaining command-line arguments as one or more Tcl commands.

### Usage

```
quartus_sh [-h | --help[=<option|topic>] | -v]

quartus_sh -g | --gui [<project_name>]

quartus_sh <other options>

quartus_sh -t <script file> [<script args>]

quartus_sh -s

quartus_sh --tcl_eval <tcl command>
```

This command supports the following options:

| Option | Page |
|---|---|
| -f=<argument file> | 2–94 |
| -h | 2–94 |
| -s | 2–104 |
| -t=<script file> | 2–104 |
| -v | 2–94 |
| --64bit | 2–94 |
| --archive | 2–61 |
| --determine_smart_action | 2–62 |
| --dse | 2–62 |
| --dtw | 2–67 |
| --flow | 2–67 |
| --help[=<option \| topic>] | 2–95 |
| --lower_priority | 2–95 |
| --prepare | 2–68 |
| --qboard | 2–68 |
| --qhelp | 2–68 |
| --qinstall | 2–68 |
| --qslave | 2–69 |
| --relcon | 2–70 |
| --restore | 2–70 |
| --script=<script file> | 2–104 |
| --set | 2–71 |
| --shell | 2–104 |
| --simlib_comp | 2–71 |
| --tcl_eval=<tcl command> | 2–105 |
| --version | 2–95 |

This command includes help on the following topics:

## --archive

Option to generate a Quartus II Archive File (.qar) for your project that contains specific sets of files.

### Usage

```
quartus_sh --archive [<options>] <project name>
```

### Available options

| | |
|---|---|
| -use_file_set <value> | Specify the archive file set ID to use. By default, the "basic" file set ID is used. Use the -list_file_sets option to view the list of possible file sets. |
| -list_file_sets | List available archive file sets. |
| -list_files | List files to be archived. If not specified, a Quartus II Archive file is generated. |
| -ascii <file name> | When combined with -list_files, this option generates the specified <file name> containing a newline-delimited list of files to be archived. |
| -no_discover | Option not to run Analysis & Elaboration. By default, Analysis & Elaboration is run unless the compiler database already exists for the revision. |
| -force | Forces the archiver to run Analysis & Elaboration and overwrite the compiler database for the revision. |
| -include_export | Include version-compatible database files. |
| -export | Export version-compatible database and include it in the archive file. |
| -include_output | Include full compilation database and output files. |
| -output <value> | Specify the output file name. By default, <revision name>.qar is used. If the file already exists, it is overwritten. |
| -input <value> | Specify the input file name containing a new-line delimited list of files to archive. This option can only be combined with the -output option. |
| -readme | Display the readme file. |
| -self_test | Run a short test on the Quartus II Archive (.qar) file after it is created. The test ensures that the .qar file contains a valid, complete and compilable design. |
| -fix_qsf | Modify the <revision>.qsf file to include all the necessary files in order to properly archive, restore and compile the design. A <revision>.archive.qip file is generated and specified in the <revision>.qsf file. |
| -all_revisions | Create an archive (named <revision>.qar) for each revision in the project. |
| -revision <value> | Specify the revision name. By default, the current project revision is archived. |
| <project name> | Specify the project name. |

### How to use the -fix_qsf option

The -fix_qsf option does the following:

1. Performs Analysis & Elaboration if necessary

2. Adds all files discovered or required by the compiler into your <revision>.qsf file quartus_sh --archive -fix_qsf top

You do not need to use the -fix_qsf option again unless you modify the design and add more design files. The .qsf file is now complete with all the required design files. Unless you add new files to the project, you can ask the archive project to always skip Analysis & Elaboration by passing the -no_discover option: quartus_sh --archive -no_discover top

### Examples

```
# Generate top.qar
quartus_sh --archive top

# Export the version-compatible database files
# and include them in the top.qar archive
quartus_sh --archive -export -output top.qar top

# Generate my_files.qar containing the files listed in my_files.txt
quartus_sh --archive -input my_files.txt -output my_files.qar

# Generate top.qar and run a short test to make sure
# top.qar contains a valid, complete and compilable design.
quartus_sh --archive -self_test top

# Generate top.txt containing a list of files to
# archive for the 'top' design.
quartus_sh --archive -ascii top.txt -list_files top
```

## --determine_smart_action

Option to open a project and determine the smart action jump.

### Usage

```
quartus_sh --determine_smart_action <project> [-c <revision>]
```

The smart action is defined as the earliest module in the Compiler flow that needs to be run based on current assignment files.

This option writes out a .chg file depending on what has changed in the source files. For a given quartus_<exe_name>, the associated .chg file name has the format <exe_name>.chg. For example, if quartus_map needs to be rerun, a file named map.chg is created.

If a timing requirement is changed, one of the following files is created or updated:

- fit.chg if timing-driven compilation is turned ON, which means that the Fitter subsequent modules need to be rerun
- tan.chg if timing-driven compilation is turned OFF and only the Timing Analyzer needs to be rerun

This option can be used to write a makefile that jumps to the correct module based on changed assignments. For example, the following makefile rules can be used:

```
qsh.log: project.qpf project.qsf ($SOURCE_FILES)
quartus_sh --determine_smart_action project > qsh.log

project.fit.rpt: fit.chg project.map.rpt
quartus_fit project

project.map.rpt: map.chg
quartus_map project
```

## --dse

THE ALTERA DESIGN SPACE EXPLORER (DSE)

The Design Space Explorer (DSE) is a tool for exploring the complex flow parameters in the Quartus II software. DSE takes the guess work out of selecting parameter values and helps you determine the optimal Quartus II software settings for a design.

## Version

```
9.0
```

## Synopsis

## Usage

```
quartus_sh --dse [options]
```

```
Options:
-archive
-concurrent-compiles [0..6]
-custom-file <filename>
-decision-column <"column name">
-exploration-space <"space">
-ignore-failed-base
-llr-restructuring
-lower-priority
-lsf-queue <queue name>
-nogui
-optimization-goal <"goal">
-project <project name>
-report-all-resource-usage
-revision <revision name>
-run-power
-search-method <"method">
-seeds <seed list>
-skip-base
-slaves <"slave list">
-stop-after-time <dd:hh:mm>
-stop-after-zero-failing-paths
-use-lsf
```

Note: To use DSE in command-line mode, specify the "-nogui" option. If you do not specify this option, the DSE graphical user interface (GUI) starts, regardless of the other command-line options used.

```
quartus_sh --dse
```

This command launches the DSE GUI.

```
quartus_sh --dse -nogui -project main
```

This command starts a default command-line exploration. The default seeds are used along with the default exploration space, optimization goal, and search method.

```
quartus_sh --dse -nogui -project main -seeds 2,4,8-10
-exploration-space "Extra Effort Search"
```

This command starts a command-line exploration of an "Extra Effort Search" space using the seeds 2, 4, 8 through 10, the default optimization goal, and the default search method.

## Options

```
-archive
```

Instructs DSE to archive all points during exploration. Without this option turned on, DSE archives only the best compilation. Archives are stored below the design directory in the sub-folder dse/result. In addition to the archive files, a set of *-dse-result.xml files hold the results for each compilation DSE performs on the design. These XML result files are for the internal use of DSE only.

`-concurrent-compiles [0..6]`

Changes the number of current compilations performed by DSE on your local system. By default DSE performs one compile at a time on your local system; increasing the number of concurrent local compilations can reduce the time it takes to explore a design space but requires additional computing resources and Quartus II licenses. Setting this option to zero prevents DSE from using your local system when running in distributed mode. When running DSE in standalone mode, setting this option to zero has the same effect as setting this option to one.

`-custom-file <filename>`

Loads the exploration space from a file instead of using a predefined exploration space. See the chapter entitled "Design Space Explorer" in the Quartus II Handbook for more information on custom exploration spaces. This option must be used with the following option:

`-exploration-space "Custom Space"`

If you do not use this option, the custom space file is ignored.

`-decision-column <"column name">`

Instructs DSE to use an the <column name> column from the DSE result table when it looks for values to make better/worse decisions. The default column is "Worst-case Slack".

`-exploration-space <"space">`

Changes the exploration space used by DSE. The default exploration space is "Seed Sweep". To see a list of available exploration spaces, enter an invalid exploration space name (like "foo") or check the list of exploration spaces for your project on the "Advanced" tab in the DSE graphical user interface.

`-ignore-failed-base`

Instructs DSE to continue exploring the space even if the base compilation fails. This is useful if the design does not fit into a device, and you want to use DSE to explore area-reducing options in the Quartus II software.

`-llr-restructuring`

Instructs DSE to try softening and even removing LogicLock regions from the design before exploring the space in an effort to maximize the effectiveness of Quartus II synthesis and fitting options.

`-lower-priority`

Lowers the priority of any thread spawned by DSE to compile a point in your design. This can reduce the impact DSE has on CPU resources while it is exploring a design space.

`-lsf-queue <queue name>`

Instructs DSE to use a non-default LSF queue at your site when distributing the search of the exploration space. For detailed information on using LSF to distribute the search of an exploration space, please see the chapter entitled "Design Space Explorer" in the Quartus II Handbook.

`-nogui`

Instructs DSE to operate in command-line mode instead of graphical user interface mode.

`-optimization-goal <"goal">`

Changes the optimization goal used by DSE. The default optimization goal is "Optimize for Speed". To see a list of available optimization goals, enter an invalid optimization goal name (like "foo") or check the list of optimization goals for your project on the "Advanced" tab in the DSE graphical user interface.

`-project <project name>`

The name of the Quartus II project to use while exploring a space.

`-report-all-resource-usage`

Instructs DSE to extract all the resouce usage information from your project and report it in the DSE report tables. If this option is not used DSE reports very few resource usage statistics in its tables.

`-revision <revision name>`

The name of the project revision to use while exploring a space. If left unspecified, DSE will use the default revision in the project.

`-run-power`

Runs the Quartus II PowerPlay Power Analyzer during exploration to produce power dissipation estimates for the project on every point in the design space. Ensuring that accurate signal activity and operating conditions have been specified for your project is essential to obtaining accurate power estimates for a design. For more information on specifying signal activity and operating conditions please see the chapter entitled "PowerPlay Power Analyzer" in the Quartus II Handbook.

`-search-method <method>`

Change the search method used by DSE. The default search method is "Accelerated Seach of Exploration Space". The available search methods are:

- "Exhaustive Search of Exploration Space"
  This method performs an exhaustive search of all combinations of all Quartus II settings in a design space to find the best combination for your project.
- "Accelerated Search of Exploration Space"
  This method performs intelligent pruning of an exploration space to arrive at the optimal combination of Quartus II settings for your project using fewer compiles.

`-seeds <seed list>`

A list of seeds to sweep as part of the exploration space. DSE accepts a comma separated list and hyphenated ranges of integer seed values. For example:

```
-seeds 1,2,8-10
```

would sweep seeds 1 and 2, and seeds 8, 9, and 10.

`-skip-base`

Instructs DSE to test your base project before trying to analyze or compile the specified revision. If the revision has already been compiled successfully, DSE will skip its own compilation of the base project. If DSE cannot determine if the base compilation can be skipped, it will issue a warning and proceed to compile the revision for you.

`-slaves <slave list>`

A list of computers on the local area network to distribute DSE compiles to and search the exploration space. Provide a comma-separate list of host names and/or IP addresses of computers that are running Quartus II qSlave instances. For more information on distributed DSE compiles, please see the "Design Space Explorer" chapter in the Quartus II Handbook.

`-stop-after-time <dd:hh:mm>`

Instructs DSE to stop exploring the space after a specified time has elapsed. The time value is specified in format "dd:hh:mm". Where "dd" is the number of days, "hh" is the number of hours and "mm" is the number of minutes to allow before the search is halted.

`-stop-after-zero-failing-paths`

Instructs DSE to stop exploring the space after it encounters any point, including the base point, that has zero failing paths. DSE uses the failing path count reported in the 'All Failing Paths' report column to make this decision.

```
-use-lsf
```

Instructs DSE to use the LSF resources available at your site when performing a distributed search of an exploration space. Specifying that DSE should use LSF resources automatically enters DSE into distributed search mode. For more information on distributed DSE compiles, please see the chapter entitled "Design Space Explorer" in the Quartus II Handbook.

### Applying Dse Optimizations

After you run Design Space Explorer, it writes its recommended optimization settings in a table to both the screen and to the <projectname>.dse.rpt output file. The recommended optimization settings are of the form:

```
+-----------------------------------------+------------------+
| Setting                                 | Value            |
+-----------------------------------------+------------------+
| ASSIGNMENT_NAME                         | ASSIGNMENT_VALUE |
+-----------------------------------------+------------------+
```

To implement the recommended optimizations when working in command-line mode, enter each optimization at the command prompt in a Tcl window in the form:

```
set_global_assignment -name ASSIGNMENT_NAME ASSIGNMENT_VALUE
```

Where:

```
<set_global_assignment> is the name of a Tcl command
```

```
<-name ASSIGNMENT_NAME> is the name of an assignment setting
```

```
<ASSIGNMENT_VALUE> is a valid value for the specified
assignment setting
```

### Examples

```
set_global_assignment -name STRATIX_OPTIMIZATION_TECHNIQUE SPEED
set_global_assignment -name ADV_NETLIST_OPT_SYNTH_GATE_RETIME ON
set_global_assignment -name ADV_NETLIST_OPT_SYNTH_WYSIWYG_REMAP ON
set_global_assignment -name AUTO_PACKED_REGISTERS_STRATIX OFF
set_global_assignment -name PHYSICAL_SYNTHESIS_COMBO_LOGIC ON
set_global_assignment -name PHYSICAL_SYNTHESIS_REGISTER_DUPLICATION OFF
set_global_assignment -name PHYSICAL_SYNTHESIS_REGISTER_RETIMING OFF
set_global_assignment -name PHYSICAL_SYNTHESIS_EFFORT NORMAL
set_global_assignment -name INNER_NUM 5
```

Note:

PHYSICAL_SYNTHESIS_EFFORT and INNER_NUM can only be applied through the Tcl window.

PHYSICAL_SYNTHESIS_EFFORT makes the physical synthesis algorithms try harder.

INNER_NUM controls the Fitter effort level.

### See Also

For more information please see the Design Space Explorer book in the Quartus II Help. Press the <F1> key to access this help from the DSE graphical user interface.

Additional information is also available in the "Design Space Explorer" chapter in the Quartus II Handbook, which is available at the Literature section of the Altera website (http://www.altera.com).

### Licensing

This script is copyrighted by Altera Corporation and provided subject to the rights granted by the Altera Legal Notice found in the file: quartus/common/tcl/apps/dse/dse.tcl.

## --dtw

Option to call a predefined Tcl/Tk script with a simple Graphical User Interface (GUI) wizard that can be used to define timing requirements for a DDR/DDR2-SDRAM memory interface.

### Usage

```
quartus_sh --dtw [<options>]
```

Use "quartus_sh --dtw -h" for help on the available options.

### Example

```
# Run the wizard. The wizard will query the user for the
# project and all necessary parameters, then apply the necessary
# timing requirements for the memory interface to the project.
quartus_sh --dtw
```

### Licensing

This script is copyrighted by Altera Corporation and provided subject to the rights granted by the Altera Legal Notice found in the file quartus/common/tcl/apps/gui/dtw/dtw.tcl

## --flow

Option to open a project and execute the specified flow.

### Usage:

Where <flow_name> is one of the following:

```
    compile
    compile_and_simulate
    signalprobe
    hardcopy_full_compile
    migrate_to_hardcopy
    functional_simulation_netlist_generation
    export_database
    import_database
    early_timing_estimate
    early_timing_estimate_with_synthesis
```

### Examples

```
    # Basic compilation
    quartus_sh --flow compile top
    quartus_sh --flow compile top -c rev1

    # Compile FPGA project, migrate to HardCopy, and
    # compile HardCopy project
    quartus_sh --flow hardcopy_full_compile top

    # You can do the same manually (assuming migration
    # creates top_hardcopy_optimization)
    quartus_sh --flow compile top
    quartus_sh --flow migrate_to_hardcopy
    cd top_hardcopy_optimization
    quartus_sh --flow compile top

    # Get an early timing estimate by running fast synthesis,
    # followed by early timing estimate and timing analysis
    quartus_sh --flow early_timing_estimate_with_synthesis top
```

```
# If synthesis has been run before you can run
# early timing estimate and timing analysis alone
quartus_sh --flow early_timing_estimate top
```

## --prepare

Option to create or open a project and make some assignments in order to prepare the project for compilation.

This option is intended to set up a project before compilation with the "--flow" option.

### Usage

```
quartus_sh --prepare [<options>] <project_name>
```

Use "quartus_sh --prepare -?" for help on the available options.

### Examples

```
# Set project and compile for Stratix
quartus_sh --prepare -f Stratix top
quartus_sh --flow compile top

# Set project and compile for Stratix using a revision
quartus_sh --prepare -r rev1 -f Stratix top
quartus_sh --flow compile top -c rev1

# Set project to compile a specified top-level entity
quartus_sh --prepare -t MyTopEntity top
quartus_sh --flow compile top
```

## --qboard

QBoard is a Tk based Graphical User Interface (GUI) that allows the user create project templates based on DevKits.

### Licensing

This script is copyrighted by Altera Corporation and provided subject to the rights granted by the Altera Legal Notice found in the file quartus/common/tcl/apps/qboard/qboard_script.tcl

## --qhelp

Option to call a predefined Tk script with a simple Graphical User Interface (GUI) that can be used to browse command-line executable and Tcl API help.

### Licensing

This script is copyrighted by Altera Corporation and provided subject to the rights granted by the Altera Legal Notice found in the file quartus/common/tcl/apps/qflow/qhelp.tcl

## --qinstall

Option to install Quartus II Device Archive (.qda) file(s). Quartus II Device Archive files can be downloaded from www.altera.com.

### Usage

```
quartus_sh --qinstall -qda <value>
```

### Available options

```
-qda <value>:
#Specify the Quartus II Device Archive file to install.
You may specify one or more -qda options.
```

### Examples

```
# Install stratixii.qda and hardcopyii.qda
quartus_sh --qinstall -qda stratixii.qda -qda hardcopyii.qda
```

## --qslave

### The Quartus Distributed Slave Utility

A utility to start the Distributed Master/Slave Toolkit's slave daemon on the slave host. The slave daemon must be started on each slave host in order to listen for job requests from the master host.

### VERSION

```
1.0
```

### Synopsis

```
   ------
   Usage:
   ------
   quartus_sh --qslave [<port_number> <jobs_limit_number> <working_directory>]

   Options [optional]:
      port=<port_number> defaults to 1977
      jobslimit=<jobs_limit_number> defaults to 1
      workdir=<working_directory> defaults to current directory
```

### Examples

```
quartus_sh --qslave
```

This command starts the Distributed Master/Slave Toolkit's slave daemon in command-line mode.

```
quartus_sh --qslave port=1977
```

This command starts the Distributed Master/Slave Toolkit's slave daemon to listen at port 1977.

```
quartus_sh --qslave jobslimit=1
```

This command starts the Distributed Master/Slave Toolkit's slave daemon to listen by setting the jobs limit to 1. This means the maximum number of jobs this slave host can accept is one. If this slave host receives more than one job, the second job is rejected.

```
quartus_sh --qslave workdir="d:/slave"
```

This command starts the Distributed Master/Slave Toolkit's slave daemon and set the working directory to "d:/slave". The working directory stores the temporary directories that are used by the slave while running the jobs. These temporary directories are deleted when the jobs are released successfully. If the jobs fail to be released for some reasons, you may need to delete these temporary directories manually to save disk space.

### Licensing

This script is copyrighted by Altera Corporation and provided subject to the rights granted by the Altera Legal Notice found in the file quartus/common/tcl/apps/qslave/qslave.tcl.

## --relcon

Option to call a predefined Tcl script that can be used to place logic registers relative to user-defined pin locations. One application of this script is to optimize timing margins for DDR/DDR2-SDRAM memory interfaces on Stratix II.

### Usage

```
quartus_sh --relcon [<options>]
```

Use "quartus_sh --relcon -?" for help on the available options.

### Example

```
# Place the read postamble registers in the LABs adjacent to the
# DQS pins to optimize read postamble setup margin.
quartus_sh --relcon -project top -pin_name "mem_dqs[*]" -reg_name
"*|postamble_en_pos_2x[*]" -row_offset 1 -apply

# Place the read resync registers in the LABs adjacent to
# the DQ pins to optimize read resync setup and hold margins.
quartus_sh --relcon -project top -pin_name "mem_dq[*]" -reg_name "*|rdata_p_ams[*]"
-row_offset 1 -apply
quartus_sh --relcon -project top -pin_name "mem_dq[*]" -reg_name "*|rdata_n_ams[*]"
-row_offset 1 -apply
```

### Licensing

This script is copyrighted by Altera Corporation and provided subject to the rights granted by the Altera Legal Notice found in the file quartus/common/tcl/apps/relcon/relative_constraint.tcl

## --restore

Option to restore the specified Quartus II Archive File (.qar).

### Usage

```
quartus_sh --restore [<options>] <.qar file name>
```

### Available options

| | |
|---|---|
| -content | List the contents of the specified Quartus II Archive file. |
| -ascii <file name> | When combined with the -content option, this option generates the specified <file name> containing a newline-delimited list of files contained in the specified Quartus II Archive file. |
| -output <value> | Specify the output directory. By default, the Quartus II Archive File is restored to the current directory. |
| -ui | Option to open the project from the Quartus II software after restoring the specified Quartus II Archive file. |
| <.qar file name> | Specify the Quartus II Archive File name. |

### Examples

```
# Only display the content of top.qar; don't restore yet.
quartus_sh --restore -content top.qar

# Create top.txt listing the content of top.qar; don't restore yet.
quartus_sh --restore -content -ascii top.txt top.qar
```

```
# Restore top.qar
quartus_sh --restore top.qar

# Restore top.qar into the top_restored/ directory
quartus_sh --restore -output top_restored top.qar
```

## --set

Option to call a predefined Tcl command to set or remove a given QSF assignment.

This command opens an existing revision, make or remove an assignment, and close the revision.

### Usage

```
quartus_sh --set [options] <name>[=<value>] <project_name>
```

### Available options

| | |
|---|---|
| -rev <revision_name> | : Revision name |
| -remove | : Remove Assignment |

### Examples

```
    # Make SMART_RECOMPILE=ON assignment
    quartus_sh --set SMART_RECOMPILE=ON top
    # Same as above but on revision rev1
    quartus_sh --set -rev rev1 SMART_RECOMPILE=ON top
    # Remove CUT_CLEAR_AND_PRESET assignment
    quartus_sh --set -remove CUT_CLEAR_AND_PRESET top
```

## --simlib_comp

Launches the Altera Simulation Library Compiler to compile Verilog and VHDL simulation libraries for all supported third-party simulators. Make sure the appropriate simulation tools are already installed and paths to the tools are either specified in the Quartus II software in the EDA Tool Options page of the Options dialog box, or are in the search path.

### Version

```
   8.1
```

### Synopsis

### Usage

```
   quartus_sh --simlib_comp [options]
```

### Options

```
-family
-tool <simulation tool name>
-language <language>
-directory<directory>
-log <filename>
-suppress_messages
-gui
```

### Options

```
   -family
```

Required option.
Specifies the device family for which you are compiling libraries.
This will result in the compilation of all libraries required for RTL and gate-level simulations.
Note: The family name should be specified in all lowercase, with no spaces.

```
-tool <simulation tool name>
```

Required option.
Specify one of the following tool names:

- modelsim
- vcs
- vcsmx
- ncsim
- activehdl
- rivierapro

Note: No libraries are generated for VCS. Instead a VCS options file, simlib_comp.vcs, is generated that specifies the library source files.

Note: Global libraries are created for for Active HDL, but not for Riviera-PRO

```
-language <language>
```

Required option.
This must be either verilog or vhdl.

```
-directory <directory>
```

Not a required option.
The directory in which to create the compiled library directories.
If not specified the default is the current directory ( ./ )

The libraries are compiled into a single directory (verilog_libs or vhdl_libs) containing subdirectories for each of the compiled libraries. The subdirectory names for Verilog libraries are always suffixed with _ver, whereas the VHDL library directories have no suffix.

For example, the Verilog version of the altera_mf library would be:
    <directory>/verilog_libs/altera_mf_ver
and the VHDL version would be:
    <directory>/vhdl_libs/altera_mf

```
-log <filename>
```

Not a required option.
Specifies the file to store all messages issued during the compilation that were not suppressed using the -suppress_messages option. If this option is not specified then no log file is used.

```
-suppress_messages
```

Not a required option.
Specifies whether or not to suppress simulator specific information and warning messages issued during compilation. This option does not apply to tool specific error messages.
Messages that are suppressed do not appear in a log file, if one was specified.
If this option is not specified then no messages are suppressed.

```
-gui
```

Not a required option.
This will launch the simlib_comp graphical user interface (GUI) regardless of the other command-line options used.

### See Also

For more information please see the EDA simulation tool specfic chapters of the verification volume in Quartus II Handbook which is available in the Literature section of the Altera website (http://www.altera.com).

### Licensing

This script is copyrighted by Altera Corporation and provided subject to the rights granted by the Altera Legal Notice found in the file: quartus/common/tcl/internal/simlib_comp.tcl.

## quartus_si

The Quartus II SSN Analyzer estimates the simultaneous switching noise contributions to voltage and timing noise.

Quartus II Analysis & Synthesis and the Fitter must be run successfully before running the SSN Analyzer.

### Usage

```
quartus_si [-h | --help[=<option|topic>] | -v]
quartus_si <project name> [<options>]
quartus_si -t <script file> [<script args>]
quartus_si -s
quartus_si --tcl_eval <tcl command>
```

This command supports the following options:

This command includes help on the following topics:

### -g[=on|off]

Refer to the help for --grouping[=on|off] on page 2–75

### --bank=<bank index>

Specify to perform SSN analysis only on the requested bank.

### --grouping[=on|off]

Specify for SSN analysis to group similar output-enable and synchronous pins together.

### --sso_inputs[=on|off]

Specify to include analysis of SSO on inputs when performing SSN analysis.

## quartus_sim

The Quartus II Simulator tests and debugs the logical operation and internal timing of the design entities in a project. The Simulator can perform two types of simulation: functional simulation and timing simulation. The quartus_sim executable includes Tcl support.

You must generate a functional simulation netlist successfully before running a functional simulation. You can generate a functional simulation netlist with the Generate Functional Simulation Netlist command (Processing menu) or with the quartus_map --generate_functional_sim_netlist <project> command at the command prompt.

The Timing Analyzer must be run successfully before running a timing simulation.

### Usage

```
quartus_sim [-h | --help[=<option|topic>] | -v]

quartus_sim <project name> [<options>]

quartus_sim -t <script file> [<script args>]

quartus_sim -s

quartus_sim --tcl_eval <tcl command>
```

This command supports the following options:

This command includes help on the following topics:

### -c=<revision name>

Refer to the help for --rev=<revision name> on page 2–79

### --cell_delay_model_type=<transport|inertial>

Specifies the type of delay model to be used for cell delays: transport or inertial.

### --check_outputs[=on|off]

Option to direct the Simulator to check expected outputs against actual outputs in the Simulation Report.

### --interconnect_delay_model_type=<transport|inertial>

Specifies the type of delay model to be used for interconnect delays: transport or inertial.

### --memory_limiter[=on|off]

Flushes signal transitions from memory to disk for memory optimization.

### --mode=<functional|timing|timing_using_fast_timing_model>

Option to specify the type of simulation to perform. The specified value can be either functional, timing or timing_using_fast_timing_model (Timing Simulation using Fast Timing Model).

### --overwrite_waveform[=on|off]

Option to overwrite simulation input file with simulation results.

### --perform_glitch_filtering=<auto|always|never>

Option to specify that the simulator perform glitch filtering when running timing simulation.

The following table displays available values:

| Value | Description |
|---|---|
| auto | Perform glitch filtering when running timing simulation only if Signal Activity File generation is turned on. |
| always | Always perform glitch filtering when running timing simulation. |
| never | Do not perform glitch filtering when running timing simulation. |

### --power_vcd_output=<target file>

Option to specify generation of (and a name for) the Value Change Dump file for the PowerPlay Power Analyzer.

The file is always generated in the project directory.

### --pvt_multicorner[=on|off]

Option to specify multicorner PVT timing models to be used for timing simulation.

## --pvt_temperature=<value_in_C>

Option to specify the device temperature (C) to use when running timing simulation with PVT timing model.

## --pvt_timing_model_type=<slow|fast>

Option to specify the type of PVT timing model type to use for timing simulation.

## --pvt_voltage=<value_in_mV>

Option to specify the device voltage (mV) to use when running timing simulation with PVT timing model.

## --rev=<revision name>

Option to specify which revision and its associated Quartus II Settings File (.qsf) to use.

This option directs the Quartus II®software to use the existing settings in the specified revision's .qsf, if any. However, all arguments passed from command line and all assignments obtained from the database override assignments from the .qsf.

If not specified, the revision name is assumed to be the same as that of the project. Altera®recommends using the same name for the project and the revision to avoid the use of this option.

## --saf_output=<target file>

Option to specify generation of the Signal Activity File as well as the name of the generated Signal Activity File.

The file will always be written to the project directory.

## --simulation_results_format=<VWF|CVWF|VCD>

Specifies the file format for the vector file that contains the results from simulation.

## --vector_comparison_rule_value_0=<0,1,X,L,H,W,Z,U and/or DC>

Specifies the waveform comparison rule for signal value 0: 0, 1, X, L, H, W, Z, U and/or DC. Example: "0,L,DC"

## --vector_comparison_rule_value_1=<0,1,X,L,H,W,Z,U and/or DC>

Specifies the waveform comparison rule for signal value 1 : 0, 1, X, L, H, W, Z, U and/or DC. Example: "1,H,DC"

## --vector_comparison_rule_value_dc=<0,1,X,L,H,W,Z,U and/or DC>

Specifies the waveform comparison rule for signal value DC: 0, 1, X, L, H, W, Z, U and/or DC. Example: "0,1,X,L,H,W,Z,U,DC"

## --vector_comparison_rule_value_h=<0,1,X,L,H,W,Z,U and/or DC>

Specifies the waveform comparison rule for signal value H : 0, 1, X, L, H, W, Z, U and/or DC. Example: "1,H,DC"

### --vector_comparison_rule_value_l=<0,1,X,L,H,W,Z,U and/or DC>

Specifies the waveform comparison rule for signal value L : 0, 1, X, L, H, W, Z, U and/or DC. Example: "0,L,DC"

### --vector_comparison_rule_value_u=<0,1,X,L,H,W,Z,U and/or DC>

Specifies the waveform comparison rule for signal value U: 0, 1, X, L, H, W, Z, U and/or DC. Example: "X,W,Z,U,DC"

### --vector_comparison_rule_value_w=<0,1,X,L,H,W,Z,U and/or DC>

Specifies the waveform comparison rule for signal value W : 0, 1, X, L, H, W, Z, U and/or DC. Example: "X,W,U,DC"

### --vector_comparison_rule_value_x=<0,1,X,L,H,W,Z,U and/or DC>

Specifies the waveform comparison rule for signal value X : 0, 1, X, L, H, W, Z, U and/or DC. Example: "X,W,U,DC"

### --vector_comparison_rule_value_z=<0,1,X,L,H,W,Z,U and/or DC>

Specifies the waveform comparison rule for signal value Z: 0, 1, X, L, H, W, Z, U and/or DC. Example: "Z,U,DC"

### --vector_source=<vector source file>

Option to specify the source of input vectors to be used for simulation. The specified value must be an existing file with a .cvwf, .vwf, .vec, .tbl, .scf or .vcd extension.

## quartus_sta

The TimeQuest Timing Analyzer computes delays for the given design and device and annotates them on the netlist for subsequent use. Then, the TimeQuest Analyzer performs timing analysis, allowing you to analyze the performance of all logic in your design. The quartus_sta executable includes Tcl support.

Quartus II Analysis & Synthesis or the Fitter must be run successfully before running the TimeQuest Timing Analyzer.

### Usage

```
quartus_sta [-h | --help[=<option|topic>] | -v]
quartus_sta <project name> [<options>]
quartus_sta -t <script file> [<script args>]
quartus_sta -s
quartus_sta --tcl_eval <tcl command>
```

This command supports the following options:

| Option | Page |
|---|---|
| -c=<revision name> | 2–98 |
| -f=<argument file> | 2–94 |
| -h | 2–94 |
| -s | 2–104 |
| -t=<script file> | 2–104 |
| -v | 2–94 |
| --64bit | 2–94 |
| --do_report_timing | 2–82 |
| --force_dat | 2–82 |
| --help[=<option \| topic>] | 2–95 |
| --lower_priority | 2–95 |
| --model=<fast \| slow> | 2–82 |
| --multicorner[=on \| off] | 2–82 |
| --post_map | 2–82 |
| --qsf2sdc | 2–82 |
| --report_script=<NONE> | 2–83 |
| --rev=<revision name> | 2–98 |
| --script=<script file> | 2–104 |
| --sdc=<NONE> | 2–83 |
| --set=<assignment=value> | 2–98 |
| --shell | 2–104 |
| --speed=<NONE> | 2–83 |
| --tcl_eval=<tcl command> | 2–105 |
| --temperature=<value_in_C> | 2–83 |
| --tq2hc | 2–83 |
| --tq2pt | 2–84 |
| --version | 2–95 |
| --voltage=<value_in_mV> | 2–84 |

This command includes help on the following topics:

## --do_report_timing

For every clock domain, this option reports the most critical path based on setup slack. This command is equivalent to:

```
     report_timing -npaths 1 -to_clock $clock
```

for every clock in the design (where $clock is the clock name).

## --force_dat

Force Delay Annotation. Using this option runs the Delay Annotator and new delays are annotated on the compiler netlist. The compiler netlist is the source from which a timing netlist is created. This option therefore ensures that new delays are used in the timing netlist. If this option is not set, the default flow attempts to re-use existing delays in the compiler netlist (if available).

## --model=<fast|slow>

Option to specify the timing model to use when running the TimeQuest Timing Analyzer.

Examples:

```
quartus_map top --family=Stratix
quartus_fit top --part=EP1S10F780C7
# Run Timing Analysis for every speed grade
quartus_sta top --model slow --temperature 0 -voltage 1200
```

## --multicorner[=on|off]

Creates slack summaries for all available operating conditions, enabling multi-corner timing analysis.

## --post_map

Analyzes output of Analysis and Synthesis (quartus_map), using a rough delay model to estimate the place and route results. Note that the error can be vary large.

## --qsf2sdc

Use this option to migrate assignments from the Classic Timing Analyzer. to SDC format. The migration includes existing QSF timing assignments, and the results found in the timing analysis report. The recommended migration flow is shown below:

```
    quartus_map rev
    quartus_fit rev
    quartus_tan rev
    quartus_sta rev --qsf2sdc
```

Warnings are given for any assignment that does not have a save conversion into an SDC command. The genrated SDC file is named <revision>.sdc, where revision is the current revision of your project.

### --report_script=<NONE>

Name of the custom Tcl script called at the end of the default script, but before the netlist is destoryed. The behavior of the default script is equivalent to the following if this option is specified:

```
     project_open <rev>
     create_timing_netlist <options>
     read_sdc if defined
     update_timing_netlist
     create summary panels

     # The custom script is loaded here
     source <script_name>

     delete_netlist
     project_close
```

An example of the script is shown next:

```
    set setup_domain_list [get_clock_domain_info -setup]

    # Report the Worst Case setup slack per clock
    foreach domain $setup_domain_list {
        report_timing -setup -to_clock [lindex $domain 0]
    }
```

### --sdc=<NONE>

Name of the SDC File to read. If this option is not specified, the TimeQuest Timing Analyzer reads the default <rev>.sdc file if it exists.

### --speed=<NONE>

Option to specify the device speed grade to use when running the TimeQuest Timing Analyzer.

Examples:

```
quartus_map top --family=Stratix
quartus_fit top --part=EP1S10F780C7
# Run Timing Analysis for every speed grade
quartus_sta top --speed 5
```

### --temperature=<value_in_C>

Option to specify the device temperature (C) to use when running the TimeQuest Timing Analyzer.

Examples:

```
quartus_map top --family=Stratix
quartus_fit top --part=EP1S10F780C7
# Run Timing Analysis for every speed grade
quartus_sta top --model slow --temperature 0 -voltage 1200
```

### --tq2hc

Generate temporary files to convert the TimeQuest Timing Analyzer SDC file(s) to a PrimeTime SDC file that can be used by the HardCopy Design Center (HCDC). The HardCopy SDC files are required to generate the required handoff files. BY default, when using this option quartus_sta converts the SDC files specified by the SDC_FILE QSF variable. Use "--sdc <file>" to manually specify the SDC to convert. This option is equivalent to the following Tcl commands:

```
project_open
create_timing_netlist
read_sdc
update_timing_netlist
write_sdc -hc
```

The Quartus II EDA Netlist Writer (quartus_eda) uses the generated files to generate the final SDC file that can be used by the HCDC. This option is not required if "HardCopy II" is selected as the device family. In that case, the default quartus_sta script automatically calls "write_sdc -hc" To manually generate the PrimeTime SDC file, the recommended flow is:

```
quartus_sta rev --tq2hc [--sdc <mysdcfile.sdc]
quartus_cdb rev --generate_hardcopy_files
```

Note that TimeQuest only SDC extensions like get_registers and get_keepers do not get converted.

## --tq2pt

Generate temporary files to convert the TimeQuest Timing Analyzer SDC file(s) to a PrimeTime SDC file. BY default, when uising this option quartus_sta converts the SDC files specified by the SDC_FILE QSF variable. Use "--sdc <file>" to manually specify the SDC to convert. This option is equivalent to executing the following Tcl commands:

```
project_open
create_timing_netlist
read_sdc
update_timing_netlist
write_sdc -primetime
```

The Quartus II EDA Netlist Writer (quartus_eda) uses the generated files to generate the final SDC file for use by PrimeTime. This option is not required if "PrimeTime" has already been selected as the EDA Timing Analyzer tool in the QSF file for the project revsion. In that case, the default quartus_sta script automatically calls "write_sdc -primetime" To manually generate the PrimeTime SDC file, the recommended flow is:

```
quartus_sta rev --tq2pt [--sdc <mysdcfile.sdc]
quartus_eda rev --timing_analysis --tool=primetime
```

Note that TimeQuest-only SDC extensions like get_registers and get_keepers do not get converted.

## --voltage=<value_in_mV>

Option to specify the device voltage (mV) to use when running the TimeQuest Timing Analyzer.

Examples:

```
quartus_map top --family=Stratix
quartus_fit top --part=EP1S10F780C7
# Run Timing Analysis for every speed grade
quartus_sta top --model slow --temperature 0 -voltage 1200
```

## quartus_stp

The Quartus II SignalTap®II Logic Analyzer captures signals from internal device nodes while the device is running at speed. The captured data is displayed as a waveform within the SignalTap II Logic Analyzer and can be saved as a SignalTap II File (.stp). A comprehensive trigger condition can be specified, and is also saved in a SignalTap II File (.stp). The quartus_stp executable creates a Quartus Setting File (.qsf) based on the SignalTap II File specified if enabled.  It also removes the settings if the SignalTap II Logic Analyzer is disabled.  It must be run successfully before running Quartus II Analysis & Synthesis.

This command also supports Quartus Setting File (.qsf) setup for Logic Analyzer Interface.  A Logic Analyzer Interface (.lai) file is used to determine how Quartus Setting File should be changed.

This command also loads Tcl packages for a number of tools that communicate with the device via the JTAG interface.  The ::quartus::stp package provides commands for SignalTap II acquisition; the ::quartus::jtag package provides commands for primitive JTAG shift and virtual JTAG shift; the ::quartus::insystem_memory_edit package provides commands that read and modify memory content when this feature is enabled; the ::quartus::logic_analyzer_interface package provides commands that control the drivers of output pins interfacing with the logic analyzer.

### Usage

```
quartus_stp [-h | --help[=<option|topic>] | -v]

quartus_stp <project name> [<options>]

quartus_stp -t <script file> [<script args>]

quartus_stp -s

quartus_stp --tcl_eval <tcl command>
```

This command supports the following options:

This command includes help on the following topics:

## -c=<revision name>

Refer to the help for --rev=<revision name> on page 2–87

## -d

Refer to the help for --disable on page 2–87

## -e

Refer to the help for --enable on page 2–87

## --create_signaltap_hdl_file

Option to back-annotate SignalTap®entities instantiated in the design to a SignalTap II File (.stp). This generated SignalTap II File is used to carry out SignalTap II acquisition from the device.

Example:

```
quartus_stp <project name> --create_signaltap_hdl_file --stp_file=<filename>
```

## --disable

Option to disable either the SignalTap®II Logic Analzyer or the Logic Analyzer Interface in the Quartus II Settings File (.qsf).

Example:

```
quartus_stp <project_name> --disable --signaltap
quartus_stp <project_name> --disable --logic_analyzer_interface
```

If neither --logic_analyzer_interface or --signaltap is used, both tools are disabled.

## --enable

Option to enable either the SignalTap®II Logic Analyzer or the Logic Analyzer Interface in the Quartus II Setting File (.qsf).

Example:

```
quartus_stp <project_name> --enable --signaltap --stp_file=<filename>
quartus_stp <project_name> --enable --logic_analyzer_interface --lai_file=<filename>
```

If neither --logic_analyzer_interface or --signaltap is used, both tools are enabled.

## --lai_file=<.lai file>

Option to specify which Logic Analyzer Interface file (.lai) to use for processing. If you do not specify a file name, the file name defined in the Quartus II Settings File (.qsf) is used by default. If there is no file name defined in the QSF, you must specify a valid file name.

## --logic_analyzer_interface

Option to set the enable option or disable option to affect the Logic Analyzer Interface settings only.

## --rev=<revision name>

Option to specify which revision and its associated Quartus II Settings File (.qsf) to use.

This option directs the Quartus II software to use the existing settings in the specified revision's QSF, if any. However, all arguments passed from command line and all assignments obtained from the database override assignments from the QSF.

If not specified, the revision name is assumed to be the same as that of the project. Altera®recommends using the same name for the project and the revision to avoid the use of this option.

## --signaltap

Option to set the enable option or disable option to affect SignalTap II settings only.

## --stp_file=<.stp>

Option to specify which SignalTap®II File (.stp) to use for processing. If you do not specify a file name, the file name defined in the Quartus II Settings File (.qsf) is used by default. If there is no file name defined in the QSF, you must specify a valid file name.

## quartus_tan

The Quartus II Timing Analyzer computes delays for the given design and device and annotates them on the netlist for subsequent use. Then, the Timing Analyzer performs timing analysis, allowing you to analyze the performance of all logic in your design. The quartus_tan executable includes Tcl support.

Quartus II Analysis & Synthesis or the Fitter must be run successfully before running the Timing Analyzer.

### Usage

```
quartus_tan [-h | --help[=<option|topic>] | -v]
quartus_tan <project name> [<options>]
quartus_tan -t <script file> [<script args>]
quartus_tan -s
quartus_tan --tcl_eval <tcl command>
```

This command supports the following options:

This command includes help on the following topics:

## --check_constraints[=<setup|hold|both>]

Option to check unconstrained keeper to keeper pairs (unconstrained paths) in the design. Results are reported in the Quartus Report Panel and ASCII format.

Optional argument can take value of:

- setupPerform setup analysis.
- holdPerform hold analysis.
- bothPerform both setup and hold analysis. Default.

### Examples

```
# Example 1. Check clock setup and hold time unconstrained paths on
# project chiptrip revision chiptrip.
quartus_tan chiptrip -c chiptrip --check_constraints

# Example 2. Check clock hold time unconstrained paths on project chiptrip.
# Force reporting results even the design is largely unconstrained.
quartus_tan chiptrip --check_constraints=hold
```

## --combined_model[=on|off]

Option to run Timing Analyzer using both minimum delays and maximum delays, producing reports for both.

## --create_timing_netlist

Create and save a timing netlist to disk.

### Example

```
execute_module -tool tan -args "--create_timing_netlist"
load_package advanced_timing
project_open chiptrip
read_timing_netlist
foreach_in_collection node [get_timing_nodes -type all] {
       set type [get_timing_node_info -info type $node]
       set name [get_timing_node_info -info name $node]
       puts "$node => $type <=> $name"
}
```

## --datasheet[=<output file name>]

Generates the actual delay timing datasheet for both Stratix and HardCopy revisions using the optionally specified output file name reporting the. By default, the command uses <revision>.datasheet.txt if you do not specify an output file name.

## --delay_annotation_only

Option to run only delay annotation and skip full timing analysis. This option can be used when you use a third-party timing analyzer or when you run timing simulation without timing analysis.

### Example

```
quartus_tan top --delay_annotation_only
quartus_sim top --mode=timing
quartus_eda top --timing_analysis --tool=primetime --format=verilog
```

## --do_min_analysis[=on|off]

Option to perform a minimum timing analysis.

The Timing Analyzer performs only the following analyses:

- clock hold
- tsu
- th
- minimum tpd
- minimum tco

## --dump_atom_generated_clocks

Generate file with the clocks, generated by Timing Analyzer from ATOMs, in the TCL format.

## --fast_model[=on|off]

Option to run Timing Analyzer (Fast Timing Model) using minimum delays. This option turns on the "--do_min_analysis" option, which forces the Timing Analyzer to perform only the following analyses:

- clock hold
- tsu
- th
- minimum tpd
- minimum tco

## --fmax=<time unit>

Option to specify global fmax requirement.

## --post_map

Option to run the Timing Analyzer (quartus_tan) based on the results from Analysis & Synthesis (quartus_map).

The estimated timing delays are primitive results, which are not as accurate as the results obtained by running the Timing Analyzer based on results from the Fitter (quartus_fit).

### Example

```
    quartus_map top
    quartus_tan top --post_map
```

## --speed=<device speed grade>

Option to specify the device speed grade for running Timing Analysis.

This option specifies the actual speed grade used by the delay annotator.

#### Examples

```
quartus_map top --family=Stratix
quartus_fit top --part=EP1S10F780C7
# Run Timing Analysis for every speed grade
quartus_tan top --speed 5 --tao=top_5.tao
quartus_tan top --speed 6 --tao=top_6.tao
quartus_tan top --speed 7 --tao=top_7.tao
```

### --tao[=<.tao file>]

Option to generate an ASCII Timing Analysis Output File (.tao). The .tao file contains results of the timing analysis similar to results in the report file (.tan.rpt) but uses a different format.

This option can be used as a way to redirect the results of a given analysis.

#### Examples

```
quartus_tan top --tao=max_results
quartus_tan top --fast_model --tao=min_results
```

### --tao_summary

Option to generate a Timing Analysis Summary File (.tao_summary). This file contains one line with the worst-case fmax number, and can be used by external scripts to parse the fmax result easily.

The .tao_summary file has one line with the following format:

```
<revision_name>, <clock_name>, <fmax number> MHz
```

### --tco=<time unit>

Option to specify global tco requirement.

### --th=<time unit>

Option to specify global th requirement.

### --timing_analysis_only

Option to skip delay annotation. This option can be used if you previously ran the Timing Analyzer (quartus_tan) with delay annotation, or if the Fitter (quartus_fit) ran delay annotation as the final step.

If delay annotation has not been run for the requested timing model(s), then quartus_tan ignores this option and performs delay annotation.

Some device families, such as Stratix™and Cyclone™, may use delay annotation as part of a set of post-fitting operations. When this occurs, the Fitter displays the following message:

```
    Info: Started post-fitting delay annotation
```

In this case, you can save time by skipping delay annotation.

If --fast_model has been used before, or if the fitter optimized to meet Fast Model Timing, then this option can be used together with --timing_analysis_only

#### Example

```
quartus_map top --family Stratix
quartus_fit top --set OPTIMIZE_FAST_CORNER_TIMING=ON
quartus_tan top --timing_analysis_only --tao=slow.tao
quartus_tan top --fast_model --timing_analysis_only --tao=fast.tao
```

### --tpd=<time unit>

Option to specify global tpd requirement.

### --tristate

Option to report tco tristate delays.

### --tsu=<time unit>

Option to specify global tsu requirement.

### --zero_ic_delays

Option to assume zero interconnect delays (0 ns) when performing timing analysis. This option can be used for a pre-fitting, best-case analysis. For this analysis, use the "--post_map" option together with this option.

#### Example

```
quartus_map top --family=Stratix
quartus_tan top --post_map --zero_ic_delays

quartus_map top --family=Stratix
quartus_fit top
quartus_tan top --zero_ic_delays
```

# Common Options

All command-line executables support the following options:

All command-line executables include help on the following topics:

## -f=<argument file>

Option to specify a file containing additional command-line arguments. The arguments are processed as if they were specified in place of this option. Therefore, arguments after this option may conflict or override options specified in the argument file.

## -h

Option to display help for an executable, including version, usage, description, option list, and help topics list.

## -v

Refer to the help for --version on page 2–95

## --64bit

Option to enable the 64-bit version of the executable.

This option is only applicable for the Linux platform. Executables that do not support this option give an error.

Using 64-bit versions of Quartus II executables allows you to access more than 4 GB of memory per process and facilitating designs that require large amounts of memory to compile.

However, compiling designs with 64-bit versions of Quartus II executables can require 50-100% more memory than the same design compiled with 32-bit versions. For example, if your design requires more than 3 GB of memory with the 32-bit version, you should have at least 6 GB of memory installed for a 64-bit compilation.

## --help[=<option|topic>]

Option to display help for the specified option or topic. If no option or topic is specified, the behavior is the same as for "-h".

## --lower_priority

Option to lower the priority of the current process. This option is useful if you use a single-processor computer, allowing you to use other applications more easily while the Quartus II software is running in the background.

## --version

Option to display the current version number.

## arguments

Quartus II command-line arguments support short options and long options in addition to file or non-option arguments. An argument is an option if it begins with a hyphen ("-") or two hyphens ("--"). Arguments can be specified in any order.

Short options begin with a single hyphen ("-"), and the option names are single characters. Short options that require an argument must specify that argument directly after the option name. Whitespace between the option and its argument is optional. If a short option does not take an argument, multiple short options may be specified with a single hyphen. For example, "-abc" and "-a -b -c" are equivalent. The last option in such a series may take an argument.

Long options begin with two hyphens ("--"), and the option name is one or more words joined by an underscore ("_"). Long options requiring arguments must specify the argument directly after the option name, separated by whitespace or an equal sign ("="). Long options with optional arguments cannot use whitespace between the option name and its argument.

Long option names can be abbreviated as long as the abbreviation does not conflict with another long option name. For example, the "--enable_wysiwyg_resynthesis" option for quartus_map can be abbreviated as "--enable_wys". It cannot be abbreviated as "-enable", since it conflicts with the "--enable_register_retiming" option.

Some options can be specified multiple times while others cannot, and not every combination of options is available. For restrictions, see the help for any option ("--help=<option>").

## makefiles

The Quartus II software supports makefile scripts that use the Quartus II executables, which allow you to integrate your scripts with a wide variety of scripting languages. The following is an excerpt from a standard makefile script.

```
###################################################################
# Project Configuration:
#
# Specify the name of the design (project), the Quartus II Settings
# File (.qsf), and the list of source files used.
###################################################################

PROJECT = chiptrip
SOURCE_FILES = auto_max.v chiptrip.v speed_ch.v tick_cnt.v time_cnt.v
ASSIGNMENT_FILES = chiptrip.qpf chiptrip.qsf
```

```
###################################################################
# Main Targets
#
# all: build everything
# clean: remove output files and database
###################################################################

all: smart.log $(PROJECT).asm.rpt $(PROJECT).tan.rpt

clean:
   rm -rf *.rpt *.chg smart.log *.htm *.eqn *.pin *.sof *.pof db

map: smart.log $(PROJECT).map.rpt
fit: smart.log $(PROJECT).fit.rpt
asm: smart.log $(PROJECT).asm.rpt
tan: smart.log $(PROJECT).tan.rpt
smart: smart.log

###################################################################
# Executable Configuration
###################################################################

MAP_ARGS = --family=Stratix
FIT_ARGS = --part=EP1S20F484C6
ASM_ARGS =
TAN_ARGS =

###################################################################
# Target implementations
###################################################################

STAMP = echo done >

$(PROJECT).map.rpt: map.chg $(SOURCE_FILES)
   quartus_map $(MAP_ARGS) $(PROJECT)
   $(STAMP) fit.chg

$(PROJECT).fit.rpt: fit.chg $(PROJECT).map.rpt
   quartus_fit $(FIT_ARGS) $(PROJECT)
   $(STAMP) asm.chg
   $(STAMP) tan.chg

$(PROJECT).asm.rpt: asm.chg $(PROJECT).fit.rpt
   quartus_asm $(ASM_ARGS) $(PROJECT)

$(PROJECT).tan.rpt: tan.chg $(PROJECT).fit.rpt
   quartus_tan $(TAN_ARGS) $(PROJECT)

smart.log: $(ASSIGNMENT_FILES)
   quartus_sh --determine_smart_action $(PROJECT) > smart.log

###################################################################
# Project initialization
###################################################################

$(ASSIGNMENT_FILES):
   quartus_sh --prepare $(PROJECT)

map.chg:
   $(STAMP) map.chg
fit.chg:
   $(STAMP) fit.chg
tan.chg:
   $(STAMP) tan.chg
asm.chg:
   $(STAMP) asm.chg
```

## return_codes

Quartus II command-line executables exit with the one of the following return codes.

| **Return Code** | **Description** |
|---|---|
| 0 | Execution was successful |
| 2 | Execution failed due to an internal error |
| 3 | Execution failed due to user error(s) |
| 4 | Execution was stopped by the user |

## Compiler Options

Command-line compiler executables support the following options:

### -c=<revision name>

Refer to the help for --rev=<revision name> on page 2–98

### --rev=<revision name>

Option to specify which revision and its associated Quartus II Settings File (.qsf) to use.

This option directs the Quartus II software to use the existing settings in the specified revision's .qsf, if any. However, all arguments passed from command line and all assignments obtained from the database override assignments from the .qsf.

If not specified, the revision name is assumed to be the same as that of the project. Altera®recommends using the same name for the project and the revision to avoid the use of this option.

### --set=<assignment=value>

Option to set a global assignment which does not belong to any entity. In general, global assignments do not take a source or target.

By default, assignments are written to the Quartus II Settings File (.qsf) unless you specify "--write_settings_files=off". An exception is the Compiler Database Interface (quartus_cdb), where assignments are written to the QSF only if you specify "--write_settings_files=on".

For example, to ignore LCELL buffers during Analysis & Synthesis (quartus_map) type:

```
quartus_map chiptrip --set=ignore_lcell_buffers=on
```

To prevent the Fitter from chosing the optimal delay chain to meet tsu and tco timing requirements for all I/O elements, type:

```
quartus_fit chiptrip --set=auto_delay_chains=off
```

Global assignments created or modified with this option are saved to the QSF.

For convenience, Quartus II command-line executables provide arguments for common assignments. For example, to set the device family to Stratix for Analysis & Synthesis (quartus_map), type:

```
quartus_map chiptrip --family=Stratix
```

Otherwise, you'd have to type:

```
quartus_map --set=family=Stratix chiptrip
```

## Parallel Processing Options

This command supports the following options:

### -p[=on|off]

Enables parallel compilation. The Quartus II software uses all processors detected on the system.

This option is equivalent to specifying the --parallel option without any arguments.

### --parallel[=<num_processors>]

Controls parallel compilation.

If specified without an option, the Quartus II software uses all processors detected on the system. Otherwise, the software attempts to use the specified number of processors. If the specified number is more than the number of processors that actually exist on the system, runtime may increase.

The default value is 1, which disables parallel compilation.

## Settings File Options

Command-line executables that read and write the Quartus II Settings File support the following options:

### --read_settings_files[=on|off]

#### Overview

Option to read the assignments from the Quartus II Settings File (.qsf) and override assignments obtained from the database. All options that pass from the command line still override any conflicting assignments found in the QSF.

By default, assignments are read from the QSF unless you specify "--read_settings_files=off".

#### Command-Line Option Details

Command-line options are provided for making many common global project settings and performing common tasks. You can use either of two methods to make assignments to an individual entity. If the project exists, open the project in the Quartus II GUI, change the assignment, and close the project. The changed assignment is updated in the QSF. Any command-line executables you run after this update use the updated assignment. See "Option Precedence" below for more information. You can also make assignments using the Quartus II Tcl scripting API. To completely script the creation of a Quartus II project, choose this method.

#### Option Precedence

If you are using the command-line executables, you need to be aware of the precedence of various project assignments and how to control the precedence. Assignments for a particular project exist in the QSF for the project. Assignments for a project can also be made by using command-line options, as described earlier. Project assignments are reflected in compiler database files that hold intermediate compilation results and reflect assignments made in the previous project compilation.

All command-line options override any conflicting assignments found in the QSF or the compiler database files. There are two command-line options to specify whether QSF or compiler database files take precedence for any assignments not specified as command-line options.

Note: Any assignment not specified as a command-line option or found in the QSF or compiler database files is set to its Quartus II software default value.

The file precedence command-line options are --read_settings_files and --write_settings_files. By default, the --read_settings_files and --write_settings_files options are turned on. Turning on the --read_settings_files option causes a command-line executable to read assignments from the QSF instead of from the compiler database files. Turning on the --write_settings_files option causes a command-line executable to update the QSF to reflect any specified options, as happens when closing a project in the Quartus II GUI.

Table 1 lists the precedence for reading assignments depending on the value of the --read_settings_files option.

Table 1. Precedence for Reading Assignments

| **Option Specified** | **Precedence for Reading Assignments** |
|---|---|
| --import_settings_files=on (Default) | 1. Command-line options<br>2. Quartus II Settings File (.qsf)<br>3. Compiler database (db directory, if it exists)<br>4. Quartus II software defaults |
| --import_settings_files=off | 1. Quartus II Settings File (.qsf)<br>2. Compiler database (db directory, if it exists)<br>3. Quartus II software defaults |

Table 2 lists the locations to which assignments are written, depending on the value of the --write_settings_files command-line option.

Table 2. Location for Writing Assignments

| **Option Specified** | **Location for Writing Assignments** |
|---|---|
| --export_settings_files=on (Default) | 1. Quartus II Settings File (.qsf) and<br>2. Compiler database |
| --export_settings_files=off | 1. Compiler database |

The following example assumes that a project named fir_filter exists, and that the Analysis & Synthesis step has been performed (using the quartus_map executable).

```
quartus_fit fir_filter --fmax=80MHz
quartus_tan fir_filter
quartus_tan fir_filter --fmax=100MHz --tao=timing_result-100.tao
--write_settings_files=off
```

The first command, quartus_fit fir_filter --fmax=80MHz, runs the Quartus II Fitter and specifies a global fMAX requirement of 80 MHz.

The second command, quartus_tan fir_filter, runs Quartus II Timing Analyzer for the results of the previous fit.

The third command reruns Quartus II Timing Analyzer with a global Fmax requirement of 100 MHz and saves the result in a file called timing_result-100.tao. By specifying the --write_settings_files=off option, the command-line executable does not update the QSF to reflect the changed Fmax requirement. The compiler database files reflect the changed Fmax requirement. If the --write_settings_files=off option is not specified, the command-line executable updates the .qsf to reflect the 100 MHz global fMAX requirement.

Use the --read_settings_files=off and --write_settings_files=off options (where appropriate) to optimize the way that the Quartus II software reads and updates the QSF. The following example shows how to avoid unnecessary QSF reading and writing.

```
quartus_map filtref --source=filtref --part=ep1s10f780c5
quartus_fit filtref --fmax=100MHz --read_settings_files=off
quartus_tan filtref --read_settings_files=off --write_settings_files=off
quartus_asm filtref --read_settings_files=off --write_settings_files=off
```

The quartus_tan and quartus_asm executables do not need to read or write settings files because they do not change any settings in the project.

## --write_settings_files[=on|off]

### Overview

Option to write out the settings obtained from command-line options to the Quartus II Settings File (.qsf).

By default, assignments are written to the QSF unless you specify "--write_settings_files=off".

### Command-Line Option Details

Command-line options are provided for making many common global project settings and performing common tasks. You can use either of two methods to make assignments to an individual entity. If the project exists, open the project in the Quartus II GUI, change the assignment, and close the project. The changed assignment is updated in the QSF. Any command-line executables that are run after this update use the updated assignment. See "Option Precedence" below for more information. You can also make assignments using the Quartus II Tcl scripting API. To completely script the creation of a Quartus II project, choose this method.

### Option Precedence

If you are using the command-line executables, you need to be aware of the precedence of various project assignments and how to control the precedence. You can find ssignments for a particular project exist in the QSF for the project. You can also make assignments for a project by using command-line options, as described earlier. Project assignments are reflected in compiler database files that hold intermediate compilation results and reflect assignments made in the previous project compilation.

All command-line options override any conflicting assignments found in the QSF or the compiler database files. There are two command-line options to specify whether QSF or compiler database files take precedence for any assignments not specified as command-line options.

Note: Any assignment not specified as a command-line option or found in the QSF or compiler database files is set to its Quartus II software default value.

The file precedence command-line options are --read_settings_files and --write_settings_files. By default, the --read_settings_files and --write_settings_files options are turned on. Turning on the --read_settings_files option causes a command-line executable to read assignments from the QSF instead of from the compiler database files. Turning on the --write_settings_files option causes a command-line executable to update the QSF to reflect any specified options, as happens when closing a project in the Quartus II GUI.

Table 1 lists the precedence for reading assignments depending on the value of the --read_settings_files option.

Table 1. Precedence for Reading Assignments

| **Option Specified** | **Precedence for Reading Assignments** |
|---|---|
| --import_settings_files=on (Default) | 1. Command-line options<br>2. Quartus II Settings File (QSF)<br>3. Compiler database (db directory, if it exists)<br>4. Quartus II software defaults |
| --import_settings_files=off | 1. Quartus II Settings File (QSF)<br>2. Compiler database (db directory, if it exists)<br>3. Quartus II software defaults |

Table 2 lists the locations to which assignments are written, depending on the value of the --write_settings_files command-line option.

Table 2. Location for Writing Assignments

| **Option Specified** | **Location for Writing Assignments** |
|---|---|
| --export_settings_files=on (Default) | 1. Quartus II Settings File (.qsf) and<br>2. Compiler database |
| --export_settings_files=off | 1. Compiler database |

The following example assumes that a project named fir_filter exists, and that the Analysis & Synthesis step has been performed (using the quartus_map executable).

```
quartus_fit fir_filter --fmax=80MHz
quartus_tan fir_filter
quartus_tan fir_filter --fmax=100MHz --tao=timing_result-100.tao
--write_settings_files=off
```

The first command, quartus_fit fir_filter --fmax=80MHz, runs the Quartus II Fitter and specifies a global fMAX requirement of 80 MHz.

The second command, quartus_tan fir_filter, runs Quartus II Timing Analyzer for the results of the previous fit.

The third command reruns Quartus II Timing Analyzer with a global Fmax requirement of 100 MHz and saves the result in a file called timing_result-100.tao. By specifying the --write_settings_files=off option, the command-line executable does not update the QSF to reflect the changed Fmax requirement. The compiler database files reflect the changed Fmax requirement. If the --write_settings_files=off option is not specified, the command-line executable updates the QSF to reflect the 100 MHz global fMAX requirement.

Use the --read_settings_files=off and --write_settings_files=off options (where appropriate) to optimize the way that the Quartus II software reads and updates the QSF. The following example shows how to avoid unnecessary QSF reading and writing.

```
quartus_map filtref --source=filtref --part=ep1s10f780c5
quartus_fit filtref --fmax=100MHz --read_settings_files=off
quartus_tan filtref --read_settings_files=off --write_settings_files=off
quartus_asm filtref --read_settings_files=off --write_settings_files=off
```

The quartus_tan and quartus_asm executables do not need to read or write settings files because they do not change any settings in the project.

## Tcl Options

Command-line executables that support Tcl support the following options:

Command-line executables that support Tcl include help on the following topics:

### -s

Refer to the help for --shell on page 2–104

### -t=<script file>

Refer to the help for --script=<script file> on page 2–104

### --script=<script file>

Option to load and execute the specified Tcl script.

Any remaining arguments are not interpreted as arguments to the executable, but passed to the Tcl script as an argument, set in the global variable quartus(args).

The following example shows how to access the arguments from a Tcl script:

```
args.tcl:
   proc show_arguments {} {

global quartus
       foreach argument $quartus(args) {
       puts "-> $argument"
       }
       }
       show_arguments

% quartus_sh -t args.tcl a b c
   % -> a
   % -> b
   % -> c
```

Note that this option does not allow other arguments to be specified before it.

### --shell

Option to start the executable in shell mode, an interactive Tcl interpreter.

Additional arguments are not allowed before or after this option.

## --tcl_eval=<tcl command>

Option to evaluate the remaining command-line arguments as a Tcl command, posting the results to stdout.

Any remaining arguments are not interpreted as an argument to the application, but treated as part of the Tcl command itself.

This option does not allow other arguments to be specified before it.

In Linux, use a backslash '\' plus a semi-colon ';' to separate Tcl commands instead of using only a semi-colon ';'. Example:

```
% quartus_sh --tcl_eval puts "Hello"\; puts "World"
   % Hello
   % World
```

In Windows platforms, using a semi-colon will suffice. Example:

```
% quartus_sh --tcl_eval puts "Hello"; puts "World"
   % Hello
   % World
```

## TCL

The Quartus II software offers three Tcl modes of operation: script, shell, and quick-eval.

Script Mode: The "-t <script file> | --script=<script file>" option evaluates a Tcl script.

See "--help=script" for more information.

Shell Mode: The "-s | --shell" option starts an interactive Tcl interpreter.

See "--help=shell" for more information.

Quick-Eval Mode: The "--tcl_eval" option evaluates the remaining command line as Tcl commands.

See "--help=tcl_eval" for more information.

Other command-line options are not available with these Tcl modes.

# 3. Tcl Packages & Commands

| Command Name | Package | Page |
|---|---|---|
| add_new_cell | chip_planner | 3–35 |
| add_new_io | chip_planner | 3–36 |
| add_row_to_table | report | 3–244 |
| add_usage | chip_planner | 3–37 |
| all_clocks | sdc | 3–268 |
| all_inputs | sdc | 3–269 |
| all_outputs | sdc | 3–270 |
| all_registers | sdc | 3–271 |
| apply_command | chip_planner | 3–38 |
| assignment_group | project | 3–182 |
| auto_partition_design | incremental_compilation | 3–107 |
| begin_logic_analyzer_interface_control | logic_analyzer_interface | 3–157 |
| begin_memory_edit | insystem_memory_edit | 3–126 |
| change_bank_to_output_pin | logic_analyzer_interface | 3–158 |
| check_netlist_and_save | chip_planner | 3–39 |
| check_node | chip_planner | 3–40 |
| check_timing | sta | 3–358 |
| checksum | misc | 3–163 |
| close_chip_planner | chip_planner | 3–41 |
| close_device | jtag | 3–141 |
| close_session | stp | 3–443 |
| compare_vector | simulator | 3–328 |
| compute_slack_on_edges | timing | 3–450 |
| connect_chain | chip_planner | 3–42 |
| convert_signal_probes | chip_planner | 3–43 |
| convert_vector | simulator | 3–332 |
| create_base_clock | timing_assignment | 3–460 |
| create_clock | sdc | 3–272 |
| create_generated_clock | sdc | 3–273 |
| create_migrated_script | chip_planner | 3–44 |
| create_p2p_delays | advanced_timing | 3–12 |
| create_partition | incremental_compilation | 3–109 |
| create_relative_clock | timing_assignment | 3–462 |
| create_report_panel | report | 3–245 |
| create_revision | project | 3–184 |

| Command Name | Package | Page |
|---|---|---|
| create_simulation_breakpoint | simulator | 3–333 |
| create_slack_histogram | sta | 3–360 |
| create_timing_netlist | sta | 3–362 |
| create_timing_netlist | timing | 3–451 |
| create_timing_summary | sta | 3–364 |
| delete_all_logiclock | incremental_compilation | 3–110 |
| delete_all_partitions | incremental_compilation | 3–111 |
| delete_all_simulation_breakpoint | simulator | 3–334 |
| delete_logiclock | incremental_compilation | 3–112 |
| delete_partition | incremental_compilation | 3–113 |
| delete_report_panel | report | 3–247 |
| delete_revision | project | 3–185 |
| delete_simulation_breakpoint | simulator | 3–335 |
| delete_sp | chip_planner | 3–45 |
| delete_timing_netlist | sta | 3–365 |
| delete_timing_netlist | timing | 3–453 |
| derive_clock_uncertainty | sdc_ext | 3–309 |
| derive_clocks | sdc | 3–275 |
| derive_pll_clocks | sdc_ext | 3–310 |
| design_has_ace_support | chip_planner | 3–46 |
| design_has_encrypted_ip | chip_planner | 3–47 |
| device_dr_shift | jtag | 3–142 |
| device_ir_shift | jtag | 3–144 |
| device_lock | jtag | 3–146 |
| device_run_test_idle | jtag | 3–147 |
| device_unlock | jtag | 3–148 |
| device_virtual_dr_shift | jtag | 3–149 |
| device_virtual_ir_shift | jtag | 3–151 |
| disable_all_simulation_breakpoint | simulator | 3–336 |
| disable_natural_bus_naming | misc | 3–164 |
| disable_simulation_breakpoint | simulator | 3–337 |
| disable_sp | chip_planner | 3–48 |
| discard_all_changes | chip_planner | 3–49 |
| discard_node_changes | chip_planner | 3–50 |
| enable_all_simulation_breakpoint | simulator | 3–338 |
| enable_ccpp_removal | sta | 3–366 |
| enable_natural_bus_naming | misc | 3–165 |
| enable_sdc_extension_collections | sta | 3–367 |
| enable_simulation_breakpoint | simulator | 3–339 |

| Command Name | Package | Page |
|---|---|---|
| enable_sp | chip_planner | 3–51 |
| end_insystem_source_probe | insystem_source_probe | 3–134 |
| end_logic_analyzer_interface_control | logic_analyzer_interface | 3–159 |
| end_memory_edit | insystem_memory_edit | 3–127 |
| escape_brackets | misc | 3–166 |
| execute_assignment_batch | project | 3–186 |
| execute_flow | flow | 3–99 |
| execute_hc | flow | 3–101 |
| execute_module | flow | 3–103 |
| export_assignments | project | 3–187 |
| export_database | database_manager | 3–85 |
| export_partition | incremental_compilation | 3–114 |
| export_stack_to | chip_planner | 3–52 |
| fast_write_to_simulation_memory | simulator | 3–340 |
| force_simulation_value | simulator | 3–341 |
| foreach_in_collection | misc | 3–168 |
| generate_bottom_up_scripts | database_manager | 3–86 |
| get_all_assignment_names | project | 3–188 |
| get_all_assignments | project | 3–189 |
| get_all_global_assignments | project | 3–192 |
| get_all_instance_assignments | project | 3–194 |
| get_all_parameters | project | 3–196 |
| get_all_quartus_defaults | project | 3–198 |
| get_all_user_option_names | project | 3–200 |
| get_assignment_groups | sdc_ext | 3–311 |
| get_assignment_info | project | 3–201 |
| get_assignment_name_info | project | 3–202 |
| get_available_operating_conditions | sta | 3–368 |
| get_back_annotation_assignments | backannotate | 3–30 |
| get_cell_info | sta | 3–369 |
| get_cells | sdc | 3–276 |
| get_clock_delay_path | advanced_timing | 3–13 |
| get_clock_domain_info | sta | 3–370 |
| get_clock_fmax_info | sta | 3–371 |
| get_clock_info | sta | 3–372 |
| get_clocks | sdc | 3–278 |
| get_clocks | timing_assignment | 3–464 |
| get_collection_size | misc | 3–171 |
| get_current_revision | project | 3–203 |

| Command Name | Package | Page |
|---|---|---|
| get_current_state_of_output_pin | logic_analyzer_interface | 3–160 |
| get_datasheet | sta | 3–374 |
| get_default_sdc_file_names | sta | 3–376 |
| get_delay_path | advanced_timing | 3–14 |
| get_delays_from_clocks | advanced_timing | 3–15 |
| get_delays_from_keepers | advanced_timing | 3–16 |
| get_device_names | jtag | 3–153 |
| get_edge_info | sta | 3–377 |
| get_edge_slacks | sta | 3–378 |
| get_editable_mem_instances | insystem_memory_edit | 3–128 |
| get_environment_info | misc | 3–172 |
| get_family_list | device | 3–92 |
| get_fanins | sdc_ext | 3–312 |
| get_fanouts | sdc_ext | 3–313 |
| get_fitter_resource_usage | report | 3–249 |
| get_global_assignment | project | 3–204 |
| get_hardware_names | jtag | 3–154 |
| get_illegal_delay_value | advanced_timing | 3–17 |
| get_info_parameters | chip_planner | 3–53 |
| get_instance_assignment | project | 3–205 |
| get_insystem_source_probe_instance_info | insystem_source_probe | 3–135 |
| get_iports | chip_planner | 3–54 |
| get_keepers | sdc_ext | 3–314 |
| get_location_assignment | project | 3–206 |
| get_logiclock | incremental_compilation | 3–115 |
| get_logiclock_contents | incremental_compilation | 3–116 |
| get_max_delay_value | advanced_timing | 3–18 |
| get_min_pulse_width | sta | 3–379 |
| get_name_info | project | 3–207 |
| get_names | project | 3–209 |
| get_net_info | sta | 3–380 |
| get_nets | sdc | 3–279 |
| get_node_by_name | chip_planner | 3–55 |
| get_node_info | chip_planner | 3–56 |
| get_node_info | sta | 3–381 |
| get_node_loc | chip_planner | 3–57 |
| get_nodes | chip_planner | 3–58 |
| get_nodes | sdc_ext | 3–315 |
| get_number_of_columns | report | 3–250 |

| Command Name | Package | Page |
|---|---|---|
| get_number_of_rows | report | 3–251 |
| get_object_info | sta | 3–382 |
| get_operating_conditions | sta | 3–383 |
| get_operating_conditions_info | sta | 3–384 |
| get_oports | chip_planner | 3–59 |
| get_parameter | project | 3–211 |
| get_part_info | device | 3–93 |
| get_part_list | device | 3–94 |
| get_partition | incremental_compilation | 3–117 |
| get_partition_file_list | incremental_compilation | 3–118 |
| get_partition_info | sta | 3–385 |
| get_partitions | sdc_ext | 3–316 |
| get_path | sta | 3–386 |
| get_path_info | sta | 3–388 |
| get_pin_info | sta | 3–391 |
| get_pins | sdc | 3–280 |
| get_point_info | sta | 3–392 |
| get_port_by_type | chip_planner | 3–60 |
| get_port_info | chip_planner | 3–61 |
| get_port_info | sta | 3–395 |
| get_ports | sdc | 3–282 |
| get_project_directory | project | 3–212 |
| get_project_revisions | project | 3–213 |
| get_register_info | sta | 3–396 |
| get_registers | sdc_ext | 3–317 |
| get_report_panel_column_index | report | 3–252 |
| get_report_panel_data | report | 3–253 |
| get_report_panel_id | report | 3–255 |
| get_report_panel_names | report | 3–256 |
| get_report_panel_row | report | 3–257 |
| get_report_panel_row_index | report | 3–258 |
| get_simulation_memory_info | simulator | 3–343 |
| get_simulation_time | simulator | 3–344 |
| get_simulation_value | simulator | 3–345 |
| get_sp_pin_list | chip_planner | 3–62 |
| get_stack | chip_planner | 3–63 |
| get_tile_power_setting | chip_planner | 3–64 |
| get_timing_analysis_summary_results | report | 3–259 |
| get_timing_edge_delay | advanced_timing | 3–19 |

| Command Name | Package | Page |
|---|---|---|
| get_timing_edge_info | advanced_timing | 3–20 |
| get_timing_edges | advanced_timing | 3–21 |
| get_timing_node_fanin | advanced_timing | 3–22 |
| get_timing_node_fanout | advanced_timing | 3–23 |
| get_timing_node_info | advanced_timing | 3–24 |
| get_timing_nodes | advanced_timing | 3–26 |
| get_timing_paths | sta | 3–397 |
| get_top_level_entity | project | 3–214 |
| get_user_option | project | 3–215 |
| group_simulation_signal | simulator | 3–346 |
| help | help | 3–105 |
| import_database | database_manager | 3–90 |
| import_partition | incremental_compilation | 3–119 |
| init_tk | misc | 3–173 |
| initialize_simulation | simulator | 3–347 |
| is_legal_delay_value | advanced_timing | 3–27 |
| is_project_open | project | 3–216 |
| list_path | timing_report | 3–478 |
| list_sps | chip_planner | 3–65 |
| load | misc | 3–174 |
| load_package | misc | 3–175 |
| load_report | report | 3–260 |
| locate | sta | 3–400 |
| logiclock_back_annotate | backannotate | 3–31 |
| make_ape_connection | chip_planner | 3–66 |
| make_input_port | chip_planner | 3–67 |
| make_output_port | chip_planner | 3–68 |
| make_sp | chip_planner | 3–69 |
| open_device | jtag | 3–155 |
| open_session | stp | 3–444 |
| p2p_timing_cut_exist | advanced_timing | 3–28 |
| partition_netlist_exists | incremental_compilation | 3–120 |
| partition_vector | simulator | 3–349 |
| post_message | misc | 3–176 |
| project_archive | project | 3–217 |
| project_close | project | 3–218 |
| project_exists | project | 3–219 |
| project_new | project | 3–220 |
| project_open | project | 3–221 |

| Command Name | Package | Page |
|---|---|---|
| project_restore | project | 3–222 |
| qexec | misc | 3–177 |
| qexit | misc | 3–178 |
| query_collection | sta | 3–402 |
| read_content_from_memory | insystem_memory_edit | 3–129 |
| read_from_simulation_memory | simulator | 3–350 |
| read_netlist | chip_planner | 3–70 |
| read_probe_data | insystem_source_probe | 3–136 |
| read_sdc | sta | 3–403 |
| read_source_data | insystem_source_probe | 3–137 |
| read_xml_report | report | 3–261 |
| release_simulation_value | simulator | 3–351 |
| remove_all_global_assignments | project | 3–223 |
| remove_all_instance_assignments | project | 3–225 |
| remove_all_parameters | project | 3–227 |
| remove_annotated_delay | sdc_ext | 3–318 |
| remove_ape_connection | chip_planner | 3–71 |
| remove_chain | chip_planner | 3–72 |
| remove_clock | sdc_ext | 3–319 |
| remove_clock_groups | sdc | 3–283 |
| remove_clock_latency | sdc | 3–284 |
| remove_clock_uncertainty | sdc | 3–285 |
| remove_disable_timing | sdc | 3–286 |
| remove_input_delay | sdc | 3–287 |
| remove_input_port | chip_planner | 3–73 |
| remove_old_cell | chip_planner | 3–74 |
| remove_output_delay | sdc | 3–288 |
| remove_output_port | chip_planner | 3–75 |
| remove_timing_tables | timing | 3–454 |
| remove_usage | chip_planner | 3–76 |
| report_advanced_io_timing | sta | 3–404 |
| report_bottleneck | sta | 3–405 |
| report_clock_fmax_summary | sta | 3–407 |
| report_clock_transfers | sta | 3–408 |
| report_clocks | sta | 3–409 |
| report_datasheet | sta | 3–410 |
| report_ddr | sta | 3–411 |
| report_device_info | device | 3–95 |
| report_exceptions | sta | 3–412 |

| **Command Name** | **Package** | **Page** |
|---|---|---|
| report_family_info | device | 3–96 |
| report_max_skew | sta | 3–416 |
| report_metastability | sta | 3–419 |
| report_min_pulse_width | sta | 3–422 |
| report_net_delay | sta | 3–424 |
| report_net_timing | sta | 3–425 |
| report_part_info | device | 3–97 |
| report_partitions | sta | 3–426 |
| report_path | sta | 3–427 |
| report_rskm | sta | 3–429 |
| report_sdc | sta | 3–430 |
| report_tccs | sta | 3–431 |
| report_timing | sta | 3–432 |
| report_timing | timing | 3–455 |
| report_ucp | sta | 3–436 |
| reset_design | sdc | 3–289 |
| resolve_file_path | project | 3–229 |
| revision_exists | project | 3–230 |
| routing_path | chip_planner | 3–77 |
| run | stp | 3–445 |
| run_multiple_end | stp | 3–446 |
| run_multiple_start | stp | 3–447 |
| run_simulation | simulator | 3–352 |
| save_content_from_memory_to_file | insystem_memory_edit | 3–130 |
| save_report_database | report | 3–262 |
| set_annotated_delay | sdc_ext | 3–320 |
| set_batch_mode | chip_planner | 3–78 |
| set_clock_groups | sdc | 3–290 |
| set_clock_latency | sdc | 3–291 |
| set_clock_latency | timing_assignment | 3–465 |
| set_clock_uncertainty | sdc | 3–293 |
| set_clock_uncertainty | timing_assignment | 3–467 |
| set_current_revision | project | 3–231 |
| set_disable_timing | sdc | 3–294 |
| set_false_path | sdc | 3–295 |
| set_global_assignment | project | 3–232 |
| set_input_delay | sdc | 3–297 |
| set_input_delay | timing_assignment | 3–469 |
| set_input_transition | sdc | 3–299 |

| Command Name | Package | Page |
|---|---|---|
| set_instance_assignment | project | 3–233 |
| set_io_assignment | project | 3–235 |
| set_location_assignment | project | 3–236 |
| set_logiclock | incremental_compilation | 3–121 |
| set_logiclock_contents | incremental_compilation | 3–122 |
| set_max_delay | sdc | 3–300 |
| set_max_skew | sdc_ext | 3–321 |
| set_min_delay | sdc | 3–302 |
| set_multicycle_assignment | timing_assignment | 3–471 |
| set_multicycle_path | sdc | 3–304 |
| set_net_delay | sdc_ext | 3–323 |
| set_node_info | chip_planner | 3–79 |
| set_operating_conditions | sta | 3–437 |
| set_output_delay | sdc | 3–306 |
| set_output_delay | timing_assignment | 3–473 |
| set_parameter | project | 3–237 |
| set_partition | incremental_compilation | 3–124 |
| set_port_info | chip_planner | 3–80 |
| set_power_file_assignment | project | 3–239 |
| set_scc_mode | sdc_ext | 3–324 |
| set_simulation_clock | simulator | 3–353 |
| set_tile_power_setting | chip_planner | 3–81 |
| set_time_format | sdc_ext | 3–325 |
| set_timing_cut_assignment | timing_assignment | 3–475 |
| set_user_option | project | 3–241 |
| start_insystem_source_probe | insystem_source_probe | 3–138 |
| stop | stp | 3–448 |
| stopwatch | misc | 3–179 |
| test_assignment_trait | project | 3–242 |
| timing_netlist_exist | sta | 3–438 |
| tristate_output_pin | logic_analyzer_interface | 3–161 |
| undo_command | chip_planner | 3–82 |
| unload_report | report | 3–263 |
| update_content_to_memory_from_file | insystem_memory_edit | 3–131 |
| update_node_loc | chip_planner | 3–83 |
| update_timing_netlist | sta | 3–439 |
| use_timequest_style_escaping | sta | 3–440 |
| write_content_to_memory | insystem_memory_edit | 3–132 |
| write_report_panel | report | 3–264 |

| Command Name | Package | Page |
|---|---|---|
| write_sdc | sta | 3–441 |
| write_source_data | insystem_source_probe | 3–139 |
| write_to_simulation_memory | simulator | 3–354 |
| write_xml_report | report | 3–266 |

# advanced_timing

This advanced package contains the set of Tcl functions for traversing the timing netlist and obtaining information about timing nodes.

The timing netlist is represented using a graph of nodes and edges (the netlist). Nodes can be of type "reg" (for registers and latches), "pin" (for top-level pins), "clk" (for nodes reported as clocks), "comb" (for all other combinational nodes), and "keeper" (for registers, latches, top-level pins, and clocks). Edges represent delays between nodes.

The "get_timing_nodes" command is the main command for access to the timing netlist. Most other commands are used to obtain information about a node or a path.

This package is available for loading in the following executable:

- quartus_tan

This package includes the following commands:

## create_p2p_delays

### Usage

```
create_p2p_delays [-clock_filter <names>] [-clock_hold] [-clock_setup] [-from <names>]
[-min_tco] [-min_tpd] [-tco] [-th] [-to <names>] [-tpd] [-tsu]
```

### Options

`-clock_filter <names>: Valid clocks for clock analyses`

`-clock_hold: Option to report clock hold paths`

`-clock_setup: Option to report clock setup paths`

`-from <names>: Valid sources`

`-min_tco: Option to report minimum tco paths`

`-min_tpd: Option to report minimum tpd paths`

`-tco: Option to report tco paths`

`-th: Option to report th paths`

`-to <names>: Valid destinations`

`-tpd: Option to report tpd paths`

`-tsu: Option to report tsu paths`

### Description

Creates a data structure with sources, delays, and clock paths for each keeper. This data is required by other commands such as "get_delay_path" or "get_clock_delay_path".

Use the options to filter specific paths between keepers.

A keeper is a node of the type pin, register, or clock.

### Example

```
# Print the longest paths from source clock
# to destination register pairs
load_package advanced_timing
project_open <design>
create_timing_netlist
create_p2p_delays
foreach_in_collection node [get_timing_nodes -type reg] {
    set reg_name [get_timing_node_info -info name $node]
    set delays_from_clock [get_delays_from_clocks $node]
    puts "register $reg_name has longest paths from clocks:"
    foreach delay $delays_from_clock {
        set src_node [lindex $delay 0]
        puts "-> clock is [get_timing_node_info -info name $src_node]"
        set path \
            [get_clock_delay_path -type longest -from $src_node -to $node]
        foreach el $path {
            puts "--> node is [get_timing_node_info -info name \
                [lindex $el 0]]"
            puts "--> delay is [lindex $el 1]"
        }
    }
}
project_close
```

## get_clock_delay_path

### Usage

```
get_clock_delay_path -from <node> -to <node> -type <type>
```

### Options

```
-from <node>: Source node
```

```
-to <node>: Destination node
```

```
-type <type>: Longest or shortest path
```

### Description

Returns a list of all nodes in the path between clock node and keeper.

A keeper is a node of the type pin, register, or clock.

### Example

```
# Print the longest paths from source clock
# to destination register pairs
load_package advanced_timing
project_open <design>
create_timing_netlist
create_p2p_delays
foreach_in_collection node [get_timing_nodes -type reg] {
    set reg_name [get_timing_node_info -info name $node]
    set delays_from_clock [get_delays_from_clocks $node]
    puts "register $reg_name has longest paths from clocks:"
    foreach delay $delays_from_clock {
        set src_node [lindex $delay 0]
        puts "-> clock is [get_timing_node_info -info name $src_node]"
        set path \
            [get_clock_delay_path -type longest -from $src_node -to $node]
        foreach el $path {
            puts "--> node is [get_timing_node_info -info name \
                [lindex $el 0]]"
            puts "--> delay is [lindex $el 1]"
        }
    }
}
project_close
```

## get_delay_path

### Usage

```
get_delay_path -from <node> -to <node> -type <type>
```

### Options

```
-from <node>: Source node
-to <node>: Destination node
-type <type>: Longest or shortest path
```

### Description

Returns a list of all nodes in the path between two keepers.

A keeper is a node of the type pin, register, or clock.

### Example

```
# Print the longest paths from source registers/pin
# to destination register pairs
load_package advanced_timing
project_open <design>
create_timing_netlist
create_p2p_delays
foreach_in_collection node [get_timing_nodes -type reg] {
    set reg_name [get_timing_node_info -info name $node]
    set delays_from_keeper [get_delays_from_keepers $node]
    puts "register $reg_name has longest paths from sources:"
    foreach delay $delays_from_keeper {
        set src_node [lindex $delay 0]
        puts "-> source is [get_timing_node_info -info name $src_node]"
        set path [get_delay_path -type longest -from $src_node -to $node]
        foreach el $path {
            puts "--> node is [get_timing_node_info -info name \
                [lindex $el 0]]"
            puts "--> delay is [lindex $el 1]"
        }
    }
}
project_close
```

## get_delays_from_clocks

### Usage

```
get_delays_from_clocks <node>
```

### Options

```
<node>: Node
```

### Description

Returns delays from clock pins to the specified register node in the following form:

```
    {{clock} {max_delay} {min_delay}}
```

### Example

```
# Print delays between clocks and registers
load_package advanced_timing
project_open <design>
create_timing_netlist
create_p2p_delays
foreach_in_collection node [get_timing_nodes -type reg] {
    set reg_name [get_timing_node_info -info name $node]
    set delays_from_clock [get_delays_from_clocks $node]
    puts "register $reg_name has delays from clocks:"
    foreach delay $delays_from_clock {
        set clock_name [get_timing_node_info -info name [lindex $delay \
            0]]
        set longest  [lindex $delay 1]
        set shortest  [lindex $delay 2]
        puts "-> clock is $clock_name"
        puts "-> longest delay is $longest"
        puts "-> shortest delay is $shortest"
    }
}
project_close
```

## get_delays_from_keepers

### Usage

```
get_delays_from_keepers <node>
```

### Options

```
<node>: Keeper node
```

### Description

Returns a list of delays to the specified keeper node from its source keeper nodes. If the specified keeper node is a register, then the command only returns delays to that register's synch ports. For example, data and clock-enable ports are included in the returned list, but clock, clear, and preset ports are excluded.

The returned list is in the following form:

```
    {{ keeper } {max delay} {min delay}}
```

A keeper is a node of the type pin, register, or clock.

If there is a "cut" assignment to the path, nothing is returned.

### Example

```
# Print delays between source and destination
# for registers/pin to register pairs
load_package advanced_timing
project_open <design>
create_timing_netlist
create_p2p_delays
foreach_in_collection node [get_timing_nodes -type reg] {
    set reg_name [get_timing_node_info -info name $node]
    set delays_from_keeper [get_delays_from_keepers $node]
    puts "register $reg_name has delays from keepers:"
    foreach delay $delays_from_keeper {
        set src_name [get_timing_node_info -info name [lindex $delay 0]]
        set longest  [lindex $delay 1]
        set shortest  [lindex $delay 2]
        puts "-> source is $src_name"
        puts "-> longest delay is $longest"
        puts "-> shortest delay is $shortest"
    }
}
project_close
```

## get_illegal_delay_value

### Usage

```
get_illegal_delay_value
```

### Options

None

### Description

Returns the value that designates an illegal delay.

Arguments: NONE

### Example

```
load_package advanced_timing
project_open <design>
create_timing_netlist
create_p2p_delays
foreach_in_collection node [get_timing_nodes -type all] {
    set node_name [get_timing_node_info -info name $node]
    set delays_from_keeper [get_delays_from_keepers $node]
    puts "node $node_name has delays from keepers:"
    foreach delay $delays_from_keeper {
        set src_name [get_timing_node_info -info name [lindex $delay 0]]
        set longest  [lindex $delay 1]
        set shortest  [lindex $delay 2]
        if {$longest == [get_illegal_delay_value] || $shortest == \
            [get_illegal_delay_value]} {
            puts "Node $node_name has illegal delay values"
        }
    }
}
```

## get_max_delay_value

### Usage

```
get_max_delay_value
```

### Options

None

### Description

Returns the value that designates a maximum delay.

Arguments: NONE

### Example

```
load_package advanced_timing
project_open <design>
create_timing_netlist
create_p2p_delays
foreach_in_collection node [get_timing_nodes -type all] {
set node_name [get_timing_node_info -info name $node]
   set delays_from_keeper [get_delays_from_keepers $node]
   puts "node $node_name has delays from keepers:"
   foreach delay $delays_from_keeper {
   set src_name [get_timing_node_info -info name [lindex $delay 0]]
   set longest  [lindex $delay 1]
   set shortest  [lindex $delay 2]
   if {$longest == [get_max_delay_value] || $shortest == \
       [get_max_delay_value]} {
   puts "Node $node_name has maximum delay values"
        }
    }
}
```

## get_timing_edge_delay

### Usage

```
get_timing_edge_delay [-cell] [-fall] [-ic] [-max] [-min] [-rise] [-total] <edge>
```

### Options

```
-cell: Get the Cell Delay component

-fall: Get Fall Delay

-ic: Get the Interconnect Delay component

-max: DEFAULT - Get Max Delay

-min: Get Min Delay

-rise: Get Rise Delay

-total: DEFAULT - Get the Total (IC + Cell) Delay

<edge>: Edge
```

### Description

Returns the requested delay for the specified timing edge.

Each edge stores two delay components, the interconnect and the cell delays. The interconnect delay represents the routing delay to the atom itself, while the cell delay represents the point to point delay in the atom. (Note that timing nodes always represent the output port of the atom). Use -ic, -cell, or -total to get the interconnect, cell, or sum of both delays

If neither -rise nor -fall is specified, then an undefined delay edge is assumed and the worst case or best case delay is returned based on the -max or -min options. If the family does not have Rise/Fall delays, then -max and -min will return the same value. (Note that this may change in the future when the Quartus®II software models On Chip Variation)

```
-min and -max are mutually exclusive
-rise and -fall are mutually exclusive
-ic, -cell and -total are mutually exclusive
```

### Example

```
# Count the number of clock edges in the design
load_package advanced_timing
project_open <design>
create_timing_netlist
foreach_in_collection edge [get_timing_edges] {
    set max_delay [get_timing_edge_delay -max $edge]
    set rise_max_delay [get_timing_edge_delay -max -rise $edge]
    set fall_max_delay [get_timing_edge_delay -max -fall $edge]
    puts "Worst Case Max = $max_delay, Rise Max = $rise_max_delay, Fall \
        Max = $fall_max_delay"
}

foreach_in_collection edge [get_timing_edges] {
    set max_delay [get_timing_edge_delay -max -total -rise $edge]
    set ic_max_delay [get_timing_edge_delay -max -ic -rise $edge]
    set cell_max_delay [get_timing_edge_delay -max -cell -rise $edge]
    puts "RISE: $max_delay = $ic_max_delay + $cell_max_delay"
}

project_close
```

## get_timing_edge_info

### Usage

```
get_timing_edge_info -info <info> <edge>
```

### Options

```
-info <info>: Type of information
<edge>: Edge
```

### Description

Returns the requested type of information for the specified timing edge.

Available information types include the following:

| Information Type | Description |
|---|---|
| type | Type of node (synch, asynch, clock). |
| src_node | Source node. |
| dst_node | Destination node. |
| atom_iport | Corresponding atom's iterm (if any). |
| ic_delay | IC delay of the edge. |
| cell_delay | Cell delay of the edge. |
| delay_string | Symbolic representation of the delay. |
| slack | Worst case slack of the edge (Available after calling "compute_slack_on_edges") |
| wysiwyg_port_type | Atom's iterm type. |

### Example

```
# Count the number of clock edges in the design
load_package advanced_timing
project_open <design>
create_timing_netlist
set count 0
foreach_in_collection edge [get_timing_edges] {
    set type [get_timing_edge_info -info type $edge]
    if { [string compare $type "clock"] == 0} {
        incr count
    }
}
puts "found $count clock edges"

# Print the worst case slack on every edge
# (This assumes timing constraints exist)
compute_slack_on_edges
foreach_in_collection edge [get_timing_edges] {
    set slack [get_timing_edge_info -info slack $edge]
    puts "Slack is $slack"
}

project_close
```

## get_timing_edges

### Usage

```
get_timing_edges
```

### Options

None

### Description

Returns a collection of edge ids.

### Example

```
# Count the number of synchronous edges in the design
load_package advanced_timing
project_open <design>
create_timing_netlist
set count 0
foreach_in_collection edge [get_timing_edges] {
    set type [get_timing_edge_info -info type $edge]
    if { [string compare $type "synch"] == 0} {
        incr count
    }
}

puts "found $count synchronous edges"
project_close
```

## get_timing_node_fanin

### Usage

```
get_timing_node_fanin -type <synch|clock|asynch> <node>
```

### Options

```
-type <synch|clock|asynch>: Type of feeding node (synch|clock|asynch)
```

```
<node>: Node
```

### Description

Returns a list of nodes feeding the specified node in the following form:

```
    {{node} {IC delay} {CELL delay}}
```

### Example

```
# Print synchronous source nodes for all registers
load_package advanced_timing
project_open <design>
create_timing_netlist
foreach_in_collection node [get_timing_nodes -type reg] {
    set reg_name [get_timing_node_info -info name $node]
    set fanins [get_timing_node_fanin -type synch $node]
    puts "register $reg_name has synch sources:"
    foreach fanin $fanins {
        set src_name [get_timing_node_info -info name [lindex $fanin 0]]
        puts "-> $src_name"
    }
}
project_close
```

## get_timing_node_fanout

### Usage

```
get_timing_node_fanout <node>
```

### Options

```
<node>: Node
```

### Description

Returns a list of nodes fed by the specified node in the following form:

```
    {{node} {IC delay} {CELL delay}}
```

### Example

```
# Print output nodes for all registers
load_package advanced_timing
project_open <design>
create_timing_netlist
foreach_in_collection node [get_timing_nodes -type reg] {
    set reg_name [get_timing_node_info -info name $node]
    set fanouts [get_timing_node_fanout $node]
    puts "register $reg_name has fanouts:"
    foreach fanout $fanouts {
        set dst_name [get_timing_node_info -info name [lindex $fanout 0]]
        puts "-> $dst_name"
    }
}
project_close
```

## get_timing_node_info

### Usage

```
get_timing_node_info -info <info> <node>
```

### Options

```
-info <info>: Type of information
<node>: Node
```

### Description

Returns the requested type of information for the specified timing node.

Available information types include the following:

| Information Type | Description |
|---|---|
| name | Signal name. |
| type | Type of node (reg, clk, pin, comb). |
| tsu | For a register, returns micro tsu. |
| th | For a register, returns micro th. |
| tco | For a register, returns micro tco. |
| location | Atom location in device. |
| is_loop | Detects whether the node is part of a strongly connected component. |
| is_clock_inverted | Detects whether a register is clocked by an inverted clock. |
| slack | Worst case slack of the node (Available after calling "compute_slack_on_edges") |
| delay | Delay of the node (if any). |
| is_tan_generated | Detects whether the Timing Analyzer (quartus_tan) synthesized the name of the node. |
| is_lvds_channel | Detects whether node is part of an LVDS circuit. |
| wysiwyg_port_type | Atom's oterm type. |
| atom_oport | Corresponding atom's oterm (if any). |
| synch_edges | Synch edges. |
| asynch_edges | Asynch edges. |
| clock_edges | Clock edges. |
| fanout_edges | Fanout edges. |
| clock_info | Clock information. |
| internal_location | Internal pin location (IOC_ instead of Pin_). Works only for Stratix®and newer device families. |
| clock_latency | For a clock get the early and late clock latency values |
| is_pll_out | Detects whether the node is a PLL clock output |
| is_clock_pin | Detects whether the node is a clock pin |
| phase_only | Phase shift independent of offset for a clock |

## Example

```
load_package advanced_timing
project_open <design>
create_timing_netlist

# For every timing node, print its type, name, and location
foreach_in_collection node [get_timing_nodes -type all] {
    set node_name [get_timing_node_info -info name $node]
    set node_type [get_timing_node_info -info type $node]
    set node_location [get_timing_node_info -info location $node]

    puts "$node_type : $node_location  : $node_name"
}

# Print all nodes that are members of a combinational loop
foreach_in_collection node [get_timing_nodes -type all] {
    if [get_timing_node_info -info is_loop $node] {
        set node_name [get_timing_node_info -info name $node]
        puts "Part of a combinational loop: $node_name"
    }
}

# Print the worst case slack on every register
# (This assumes timing constraints exist)
compute_slack_on_edges
foreach_in_collection node [get_timing_nodes -type all] {
    set node_name [get_timing_node_info -info name $node]
    set node_slack [get_timing_node_info -info slack $node]
    puts "Slack is $node_slack for $node_name"
}

project_close
```

## get_timing_nodes

### Usage

```
get_timing_nodes -type <all|pin|reg|clk|comb|keeper>
```

### Options

```
-type <all|pin|reg|clk|comb|keeper>: Category of node (all|pin|reg|clk|comb|keeper)
```

### Description

Returns a collection of node ids from the timing netlist. The collection can be filtered by node type using the "-type" option. Note that the node types in the timing netlist may not correspond to the node types in the Node Finder (or the "get_names" command in ::quartus::project). For example, nodes marked as memory in the Node Finder may be represented as registers or combinational nodes in the timing netlist.

The Timing Analyzer represents all designs using a graph of nodes and edges (the netlist). Nodes can be of type "reg" (for registers and latches), "pin" (for top-level pins), "clk" (for nodes reported as clocks), "comb" (for all other combinational nodes), and "keeper" (for registers, latches, top-level pins, and clocks). Edges represent delays between nodes.

Use the "get_timing_node_info" command to get specific information about every node,for example, node name, type, or location.

### Example

```
# Print names of the register and pin nodes
load_package advanced_timing
project_open <design>
create_timing_netlist
set count 1
puts "Nodes:"
foreach_in_collection node [get_timing_nodes -type keeper] {
    set node_type [get_timing_node_info -info type $node]
    if {![string equal $node_type  clk] } {
        set node_name [get_timing_node_info -info name $node]
        puts " $count : $node_type : $node_name"
        incr count
    }
}
project_close

# For every timing node, print its type
load_package advanced_timing
project_open <design>
create_timing_netlist
foreach_in_collection node [get_timing_nodes -type all] {
    puts "node name is [get_timing_node_info -info name $node]"
}
project_close
```

## is_legal_delay_value

### Usage

```
is_legal_delay_value <delay>
```

### Options

```
<delay>: Delay value
```

### Description

Returns true if the delay value is not an illegal delay and not a max delay.

Arguments: Delay value

### Example

```
load_package advanced_timing
project_open <design>
create_timing_netlist
create_p2p_delays
foreach_in_collection node [get_timing_nodes -type all] {
    set node_name [get_timing_node_info -info name $node]
    set delays_from_keeper [get_delays_from_keepers $node]
    puts "node $node_name has delays from keepers:"
    foreach delay $delays_from_keeper {
        set src_name [get_timing_node_info -info name [lindex $delay 0]]
        set longest  [lindex $delay 1]
        set shortest  [lindex $delay 2]
        if {![is_legal_delay_value $longest] || ![is_legal_delay_value \
            $shortest]} {
            puts "Node $node_name has illegal delay values"
        }
    }
}
```

## p2p_timing_cut_exist

### Usage

```
p2p_timing_cut_exist -from <from> -to <to>
```

### Options

```
-from <from>: source node id
```

```
-to <to>: destination node id
```

### Description

Returns whether a p2p path is cut or not

```
Arguments:-from <node_id>
          -to <node_id>
```

### Example

```
# iterate through all nodes and print all CUT assignments
# from registers to pins
load_package advanced_timing
project_open <design>
create_timing_netlist
load_package advanced_timing
puts "CUT Assignments:"
foreach_in_collection src_node [get_timing_nodes -type reg] {
    foreach_in_collection dst_node [get_timing_nodes -type pin] {
        if [p2p_timing_cut_exist -from $src_node -to $dst_node] {
            puts "from: [get_timing_node_info -info name $src_node] \
              to: [get_timing_node_info -info name $dst_node]"
        }
    }
}
```

# backannotate

This package contains the set of Tcl functions for back-annotating assignments for a project.

This package is available for loading in the following executables:

- quartus
- quartus_cdb
- quartus_tan

This package includes the following commands:

## get_back_annotation_assignments

### Usage

```
get_back_annotation_assignments
```

### Options

None

### Description

Returns an output collection of back-annotation assignments.

Each element of the collection is a list with the following format: { {<Source>} {<Destination>} {<Assignment name>} {<Assignment value>} {<Entity name>} }

### Example

```
## Print out all the back-annotation assignments
set asgn_col [get_back_annotation_assignments]
foreach_in_collection asgn $asgn_col {

    ## Each element in the collection has the following
    ## format:
    ## { {<Source>} {<Destination>} {<Assignment name>} {<Assignment
    # value>} {<Entity name>} }
    set from   [lindex $asgn 0]
    set to     [lindex $asgn 1]
    set name   [lindex $asgn 2]
    set value  [lindex $asgn 3]
    set entity [lindex $asgn 4]
    puts "$entity : $name ($from -> $to) = $value"
}
```

## logiclock_back_annotate

### Usage

```
logiclock_back_annotate [-exclude_from] [-exclude_to] [-from <source name>] [-lock]
[-no_contents] [-no_delay_chain] [-no_demote_lab] [-no_demote_mac] [-no_demote_pin]
[-no_demote_ram] [-no_dont_touch] [-path_exclude <path_exclude name>] [-region <region
name>] [-remove_assignments] [-resource_filter <resource_filter value>] [-routing] [-to
<destination name>]
```

### Options

`-exclude_from: Option to exclude the source node`

`-exclude_to: Option to exclude the destination node`

`-from <source name>: Name (or wildcard expression) of the source node to be back-annotated`

`-lock: Option to lock back-annotated regions`

`-no_contents: Option not to back-annotate contents`

`-no_delay_chain: Option not to back-annotate delay chain settings`

`-no_demote_lab: Option not to demote LAB or LE assignments`

`-no_demote_mac: Option not to demote DSP block assignments`

`-no_demote_pin: Option not to demote pin assignments`

`-no_demote_ram: Option not to demote RAM assignments`

`-no_dont_touch: Option not to set the don't_touch flag for each back-annotated node`

`-path_exclude <path_exclude name>: Option to exclude the specified node from the path filter`

`-region <region name>: Name (or wildcard expression) of region to be back-annotated`

`-remove_assignments: Option to remove matching assignments instead of creating them`

`-resource_filter <resource_filter value>: Option to use the resource filter`

`-routing: Option to back-annotate the LogicLock region's routing`

`-to <destination name>: Name (or wildcard expression) of the destination node to be back-annotated`

### Description

Back-annotates a LogicLock region and its contents.

When you use the "-routing" option, you must use the "-lock" and "-no_demote_lab" options, without the "-no_contents" option, or use the"-remove_assignments" option.

The "-remove_assignments" option removes all matching region contents. When you use the "-remove_assignments" option, the demotion options, "-no_contents" and "-lock", are not applicable.

The "-resource_filter" option allows you to back-annotate only specific resource types on the device. For example:

```
    logiclock_back_annotate -resource_filter "COMBINATORIAL"
```

This command back-annotates all combinatorial nodes in the design. The complete set of options is:

| | |
|---|---|
| COMBINATORIAL | combinatorial nodes |
| REGISTER | registered nodes |
| MEGA | M-RAMs |

| | |
|---|---|
| MEDIUM | M4K memory blocks |
| SMALL | M512 memory blocks |
| IO | I/O elements |
| MAC | DSP blocks |

Altera recommends that you use a Verilog Quartus®Mapping File (.vqm) as the source. When any of the advanced netlist optimizations are enabled, it is possible for the Fitter to create and rename nodes in the design during a place and route operation. Back annotation requires that on subsequent compilations the node names in the netlist match those in the constraint file. Write out a VQM netlist and create a new project using that netlist as its source. Copy all of the existing constraint files into the new project directory and remove all the design files except the new .vqm by using the Add/Remove Files in a Project command (Project menu) in the Quartus II GUI.

The Quartus II software will create a root region if you back-annotate nodes that are not members of a LogicLock region. The root region is device-size and locked. You can make assignments to the root region but you cannot delete it or modify its size or location.

## Example

```
# Open the project "example_project"
project_open example_project

# Compile the design
package require ::quartus::flow
execute_flow -compile

package require ::quartus::backannotate

# Back annotate all nodes and routing in the region "one_region"
logiclock_back_annotate -routing -lock -no_demote_lab -region one_region

# Back annotate the location of the nodes on all paths that
# start with a node that matches the "Data_in*" wildcard
# expression, and end with a node that matches the "Data_out*"
# wildcard expression
logiclock_back_annotate -from Data_in* -to Data_out*

# Back annotate the placement of all the registers in the design
logiclock_back_annotate -resource_filter "REGISTER"

# Close the project
project_close
```

## chip_planner

This package contains the set of Tcl functions for identifying and modifying resource usage and routing with the Chip Planner.

This package is available for loading in the following executable:

- quartus_cdb

This package includes the following commands:

## add_new_cell

### Usage

```
add_new_cell [-arith] -cell_name <cell name> [-comb] [-crc] [-ddio_in] [-ddio_oe]
[-ddio_out] [-extended] [-ff] [-hsadder] [-ibuf] [-location <location>] [-obuf]
[-partition_name <partition_name>]
```

### Options

```
-arith: Option to create a new LUT lcell_comb atom in arithmetic mode.  For Stratix II,
HardCopy II, and newer devices

-cell_name <cell name>: Name of new cell

-comb: Option to create a new lcell_comb atom.  For Stratix II, Cyclone II, HardCopy II,
and newer devices

-crc: Option to create a new CRC block

-ddio_in: Option to create a new DDIO Input block in devices with composite I/O

-ddio_oe: Option to create a new DDIO OE block in devices with composite I/O

-ddio_out: Option to create a new DDIO Output block in devices with composite I/O

-extended: Option to create a new extended LUT lcell_comb atom.  For Stratix II and newer
devices

-ff: Option to create a new lcell_ff atom.  For Stratix II, Cyclone II, HardCopy II, and
newer devices

-hsadder: Option to create a new lcell_hsadder atom.  For HardCopy II devices only

-ibuf: Option to create a new ibuf block

-location <location>: Location of new cell

-obuf: Option to create a new obuf block

-partition_name <partition_name>: Name of the partition where the cell is placed.  Only
for logic cells (default, comb, ff).
```

### Description

Creates a new atom of the specified type in the netlist. The atom is placed in at the top level of the design hierarchy by default, but it can also be placed into a specific partition.

If the atom type is not specified, it defaults to LCELL and is supported only on Stratix®, Stratix GX, and Cyclone™devices.

### Example

```
add_new_cell -cell_name test1 -location LC_X1_Y1_N1 -partition_name rx
add_new_cell -cell_name test1 -comb -location LC_X1_Y1_N0
```

## add_new_io

### Usage

```
add_new_io -cell_name <cell name> -direction <input|output|bidir> [-location <location>]
```

### Options

```
-cell_name <cell name>: Name of new pin
-direction <input|output|bidir>: Direction of new pin
-location <location>: Cell location
```

### Description

Adds a new I/O pin to the netlist.

### Example

```
add_new_io -cell_name test1 -location LC_X1_Y1_N1 -direction input
```

## add_usage

### Usage

```
add_usage [-gen_id <gen id>] [-node <node id>] [-port_id <port id>] -port_type <port
type>
```

### Options

`-gen_id <gen id>: Input port generic id`

`-node <node id>: Node id`

`-port_id <port id>: Input port id`

`-port_type <port type>: Type of port usage to add`

### Description

Adds a new usage to the specified input port.

Returns 1, if the usage is added. Returns 0, otherwise.

### Example

```
add_usage -node 0 -port_id 4 -port_type DATAC
add_usage -gen_id 21 -port_type DATAC
```

## apply_command

### Usage

```
apply_command <position>
```

### Options

```
<position>: Position
```

### Description

Execute the command at the specified position in the command stack.

### Example

```
apply_command 5
```

## check_netlist_and_save

### Usage

```
check_netlist_and_save
```

### Options

None

### Description

Checks that the netlist is legal and writes the netlist to disk.

If you make netlist connectivity changes, this command performs ECO fitting.

Returns 1, if the netlist is legal, and saves the netlist to disk. Returns 0, if the netlist is illegal, and restores the current netlist to the last legal netlist.

### Example

```
check_netlist_and_save
```

## check_node

### Usage

```
check_node [-gen_id <gen id>] [-node <node id>]
```

### Options

```
-gen_id <gen id>: Node generic ID
```

```
-node <node id>: Node ID
```

### Description

Checks whether the specified node is legal.

Returns 1, if the node is legal. Returns 0, otherwise.

Even if a node is legal, you still must run the check_netlist_and_save command to verify the node legality within the netlist.

### Example

```
check_node -node 3
```

## close_chip_planner

### Usage

```
close_chip_planner
```

### Options

None

### Description

Releases the chip planner netlist from use.

### Example

```
close_chip_planner
```

## connect_chain

### Usage

```
connect_chain [-gen_id <gen id>] [-node <node id>]
```

### Options

```
-gen_id <gen id>: Node generic id
-node <node id>: Node id
```

### Description

Connects a carry chain to this node. Supported for Stratix, Cyclone, and MAX II device families only.

### Example

```
connect_chain -node 3
```

## convert_signal_probes

### Usage

```
convert_signal_probes
```

### Options

None

### Description

Converts Existing QSF SignalProbe Assignments into ECOs.

### Example

```
convert_signal_probes
```

## create_migrated_script

### Usage

```
create_migrated_script [-file <file name>]
```

### Options

```
-file <file name>: Migrated Tcl Script File Name
```

### Description

Creates a Tcl script showing the ECO changes for HardCopy II based on the changes implemented in a design targeting Stratix II.

### Example

```
create_migrated_script
```

## delete_sp

### Usage

```
delete_sp -pin_name <pin name>
```

### Options

```
-pin_name <pin name>: SignalProbe Pin Name
```

### Description

Deletes a signal probe connected to the named pin.

### Example

```
delete_sp
```

## design_has_ace_support

### Usage

```
design_has_ace_support
```

### Options

None

### Description

Determines whether Chip Planner operations can be performed on the current design.

### Example

```
design_has_ace_support
```

## design_has_encrypted_ip

### Usage

```
design_has_encrypted_ip
```

### Options

None

### Description

Determines whether the current design contains encrypted IP.

Returns 1, if the design contains encrypted IP. You may be able to view or edit individual nodes of the design if they are not part of an encrypted IP. To check individual nodes, use the command "get_node_info -node <node id> -info encrypted". Returns 0, otherwise.

### Example

```
design_has_encrypted_ip
```

## disable_sp

### Usage

```
disable_sp -pin_name <pin name>
```

### Options

```
-pin_name <pin name>: SignalProbe Pin Name
```

### Description

Disables a signal probe connected to the named pin.

### Example

```
disable_sp
```

## discard_all_changes

### Usage

```
discard_all_changes
```

### Options

None

### Description

Discards all the changes made to the netlist since the last successful use of the "check_netlist_and_save" command.

### Example

```
discard_all_changes
```

## discard_node_changes

### Usage

```
discard_node_changes [-gen_id <gen id>] [-node <node id>]
```

### Options

```
-gen_id <gen id>: Node generic id
```

```
-node <node id>: Node id
```

### Description

Discards all the changes made to the specified node since the last successful use of the check_netlist_and_save command.

### Example

```
discard_node_changes -node 3
```

## enable_sp

### Usage

```
enable_sp -pin_name <pin name>
```

### Options

```
-pin_name <pin name>: SignalProbe Pin Name
```

### Description

Enables a signal probe connected to the named pin.

### Example

```
enable_sp
```

## export_stack_to

### Usage

```
export_stack_to [-applied] [-file <file name>] <position>
```

### Options

```
-applied: Whether to export only applied changes
-file <file name>: Name of file to which the stack is exported
<position>: Position
```

### Description

Exports the the command stack to the specified position as a Tcl script, CSV, or TXT file. If position is not specified, exports the entire stack. The type of file is set by the extension specified in the name argument. If a .csv or .txt extension is specified, a file of the specified type is produced. Otherwise, a Tcl script will be produced.

### Example

```
The following is a sample sequence of commands that
produces a Tcl script and a CSV file of the changes in the
Command Manager:

project_open <project_name>
read_netlist
export_stack_to -@ARG(name) new_tcl_file.tcl
export_stack_to -@ARG(name) new_tcl_file.csv
project_close
```

## get_info_parameters

### Usage

```
get_info_parameters [-file <file name>] [-for_chip]
```

### Options

```
-file <file name>: Name of output file
```

```
-for_chip: Option to display all of the chip info parameters
```

### Description

Returns a Tcl list of information parameters.

When you use the -file option, the list is redirected to the specified output file. If the output file already exists, it is overwritten without warning.

### Example

```
get_info_parameters
get_info_parameters -file the_list.txt
```

## get_iports

### Usage

```
get_iports [-as_gen_id] [-gen_id <gen id>] [-node <node id>] [-src_gen_id <gen id>]
```

### Options

`-as_gen_id: Option to return results as generic ID`

`-gen_id <gen id>: Node generic ID`

`-node <node id>: Node id`

`-src_gen_id <gen id>: Source port generic ID`

### Description

Returns a collection of input ports for the specified node.

You can use the collection with the "foreach_in_collection" command.

### Example

```
get_iports -node 3
get_iports -src_gen_id 5
```

## get_node_by_name

### Usage

```
get_node_by_name [-as_gen_id] -name <node name>
```

### Options

```
-as_gen_id: Option to return result as generic id
-name <node name>: Node name
```

### Description

Returns the node id of the specified node.

Returns -1 if the node cannot be found.

### Example

```
get_node_by_name -name 3
```

## get_node_info

### Usage

```
get_node_info [-gen_id <gen id>] -info <information type> [-node <node id>]
```

### Options

```
-gen_id <gen id>: Node generic id
```

```
-info <information type>: Type of information
```

```
-node <node id>: Node id
```

### Description

Returns the requested type of information for the specified node.

To get available information types, use the "get_info_parameters" command.

If the information type is legal for the specified node, the result is the requested information. Otherwise, the result is an empty string.

### Example

```
get_node_info -node 3 -info name
```

## get_node_loc

### Usage

```
get_node_loc
```

### Options

None

### Description

```
Dumps locations for all post-fit lcell nodes in a text file named
   node_loc.loc. To change any node locations using ECO fitter, update
   node_loc.loc file and call the update_node_loc command.  You must open
   a project and call read_netlist before using this command. The project
   must have passed fitter before using this command.
```

### Example

```
project_open my_proj
   read_netlist
   get_node_loc
   project_close
```

## get_nodes

### Usage

```
get_nodes -type <all|lcell|io|pll|dsp|ram>
```

### Options

```
-type <all|lcell|io|pll|dsp|ram>: Type of nodes to return
```

### Description

Returns a collection of nodes of the specified type.

You can use the collection with the foreach_in_collection Tcl command.

### Example

```
get_nodes -type all
```

## get_oports

### Usage

```
get_oports [-as_gen_id] [-gen_id <gen id>] [-node <node id>]
```

### Options

```
-as_gen_id: Option to return results as generic id
-gen_id <gen id>: Node generic id
-node <node id>: Node id
```

### Description

Returns a collection of output ports for the specified node.

You can use the collection with the foreach_in_collection command.

### Example

```
get_oports -node 3
```

## get_port_by_type

### Usage

```
get_port_by_type [-as_gen_id] [-gen_id <gen id>] [-literal_index <literal index>] [-node
<node id>] -port_type <port type> -type <iport|oport>
```

### Options

`-as_gen_id: Option to return result as generic ID`

`-gen_id <gen id>: Node generic id`

`-literal_index <literal index>: Literal index`

`-node <node id>: Node id`

`-port_type <port type>: Port type`

`-type <iport|oport>: Option to specify the port as an input or output port`

### Description

Returns the port index for the specified port type on the specified node.

Returns -1 if the port is not in use or is invalid for the specified node.

### Example

```
get_port_by_type -node 0 -port_type SLOAD -type iport
get_port_by_type -node 0 -port_type EXTCLK -literal_index 2 -type oport
```

## get_port_info

### Usage

```
get_port_info [-gen_id <gen id>] -info <information type> [-node <node id>] [-port_id
<port id>] [-type <iport|oport>]
```

### Options

```
-gen_id <gen id>: Port generic ID
```

```
-info <information type>: Type of information
```

```
-node <node id>: Node ID
```

```
-port_id <port id>: Port ID
```

```
-type <iport|oport>: Option to specify the port as an input or output port
```

### Description

Returns the requested type of information for the specified port.

To get available information types, use the get_info_parameters command.

If the information type is legal for the specified port, the result is the requested information. Otherwise, the result is an empty string.

### Example

```
get_port_info -node 3 -port_id 2 -type iport -info port_name
```

## get_sp_pin_list

### Usage

```
get_sp_pin_list
```

### Options

None

### Description

Returns a list of the pins availible for use as signal probe output pins.;

### Example

```
get_sp_pin_list
```

## get_stack

### Usage

```
get_stack [-line <line>] [-signalprobe] [-size] [-top_only]
```

### Options

```
-line <line>: Option to print a command in a comma-separated format
-signalprobe: Option to return a list of SignalProbe commands
-size: Option to return the number of top level commands
-top_only: Option to print the top level command only.  Use with the -line option
```

### Description

Returns specified information from the command stack.

### Example

```
get_stack -size
get_stack -line 4
```

## get_tile_power_setting

### Usage

```
get_tile_power_setting [-X <X location>] [-Y <Y location>] [-gen_id <gen id>]
```

### Options

```
-X <X location>: X location
```

```
-Y <Y location>: Y location
```

```
-gen_id <gen id>: Generic id
```

### Description

Returns the High-Speed/Low Power setting of the tile at the specified location.

### Example

```
get_tile_power_setting -gen_id 12345
get_tile_power_setting -X 12 -Y 5
```

## list_sps

### Usage

```
list_sps
```

### Options

None

### Description

Returns a list of all SignalProbe instances.

### Example

```
list_sps
```

## make_ape_connection

### Usage

```
make_ape_connection [-delay_chain_index <delay chain index>] [-dst_node <node id>]
[-gen_id <gen id>] [-gnd] [-literal_index <literal index>] -port_type <port type>
[-src_gen_id <gen id>] [-src_node <node id>] [-src_port <port id>] [-vcc]
```

### Options

```
-delay_chain_index <delay chain index>: Used for Connections from pins through the
PAD_TO_CORE delay chain. -2 = bypass, -1 = don't care, 0,1 = specific index
```

```
-dst_node <node id>: Destination node ID
```

```
-gen_id <gen id>: Destination node generic ID
```

```
-gnd: Option to connect to GND
```

```
-literal_index <literal index>: Literal index of new input port
```

```
-port_type <port type>: Type of new input port
```

```
-src_gen_id <gen id>: Source port generic ID
```

```
-src_node <node id>: Source node ID
```

```
-src_port <port id>: Source port ID
```

```
-vcc: Option to connect to VCC
```

### Description

Makes a netlist connection between the source and destination ports.

Returns 1, if a netlist connection is made between the two ports. Returns 0, otherwise.

### Example

```
make_ape_connection -src_node 0 -src_port 0 -dst_node 3 -port_type DATAC
```

## make_input_port

### Usage

```
make_input_port [-gen_id <gen id>] [-literal_index <literal index>] [-node <node id>]
-port_type <port type>
```

### Options

`-gen_id <gen id>: Generic ID of the node for the port`

`-literal_index <literal index>: Literal index of new input port`

`-node <node id>: ID of the node for the port`

`-port_type <port type>: Type of new input port`

### Description

Adds a new input port to the specified node.

To use this command, the input port must not exist.

Returns the port index, if a new input port is added to the node. Returns -1, otherwise.

### Example

```
make_input_port -node 0 -port_type DATAA
make_input_port -node 0 -port_type DATAA -literal_index 0
```

## make_output_port

### Usage

```
make_output_port [-gen_id <gen id>] [-literal_index <literal index>] [-node <node id>]
-port_type <port type>
```

### Options

```
-gen_id <gen id>: Generic ID of the node to which to add the port
```

```
-literal_index <literal index>: Literal index of new output port
```

```
-node <node id>: ID of the node for adding the port
```

```
-port_type <port type>: Type of new output port
```

### Description

Adds a new output port to the specified node. The output port must not already exist.

Returns the port index, if a new output port is added to the node. Returns -1, otherwise.

### Example

```
make_output_port -node 0 -port_type REGOUT
make_output_port -node 0 -port_type REGOUT -literal_index 0
```

## make_sp

### Usage

```
make_sp [-clk <clock signal name>] [-io_std <io standard>] -loc <location> -pin_name <pin
name> [-regs <num>] [-reset <reset signal name>] -src_name <source name>
```

### Options

```
-clk <clock signal name>: Clock signal for the register in the SignalProbe pipeline
-io_std <io standard>: I/O standard of the SignalProbe pin
-loc <location>: Pin location of the SignalProbe pin
-pin_name <pin name>: SignalProbe pin name
-regs <num>: Number of pipeline stages for the SignalProbe pin
-reset <reset signal name>: Reset signal for the register in the SignalProbe pipeline
-src_name <source name>: Name of the signal to probe
```

### Description

Creates a SignalProbe pin connected to the source signal using the specified number of registers. Columns are: Pin_name, pin_lcation, io_standard, oterm_name, number of registers, clock name, reset_name, is enabled, and has ECO changes.

### Example

```
make_sp
```

## read_netlist

### Usage

```
read_netlist [-force_post <pass|fail>]
```

### Options

```
-force_post <pass|fail>: Force cleanup procedures to run after a fit if -force was used
```

### Description

Reads the Chip Planner netlist from the last compilation.

You must open a project before using this command.

### Example

```
read_netlist
```

## remove_ape_connection

### Usage

```
remove_ape_connection [-dst_gen_id <gen id>] [-dst_node <node id>] [-dst_port <port id>]
```

### Options

```
-dst_gen_id <gen id>: Destination port generic id
```

```
-dst_node <node id>: Destination node id
```

```
-dst_port <port id>: Destination port id
```

### Description

Removes a netlist connection to the destination port.

This command also removes the destination port.

Returns 1, if the netlist connection is removed. Returns 0, otherwise.

### Example

```
remove_ape_connection -dst_node 3 -dst_port 5
```

## remove_chain

### Usage

```
remove_chain [-gen_id <gen id>] [-node <node id>]
```

### Options

```
-gen_id <gen id>: Node generic id
-node <node id>: Node id
```

### Description

Removes a carry chain connected to this node. Supported for Stratix, Cyclone, and MAX II device families only.

### Example

```
remove_chain -node 1
```

## remove_input_port

### Usage

```
remove_input_port [-gen_id <gen id>] [-literal_index <literal index>] [-node <node id>]
[-port_type <port type>]
```

### Options

```
-gen_id <gen id>: Input port generic ID
```

```
-literal_index <literal index>: Literal index of input port
```

```
-node <node id>: ID of the node to which to remove the port
```

```
-port_type <port type>: Type of input port
```

### Description

Deletes the specified input port.

To use this command, the input port must exist.

Returns 1 if successful, otherwise returns 0.

### Example

```
remove_input_port -node 0 -port_type DATAA
remove_input_port -node 0 -port_type DATAA -literal_index 0
```

## remove_old_cell

### Usage

```
remove_old_cell [-gen_id <gen id>] [-node <node id>]
```

### Options

```
-gen_id <gen id>: Node generic id
-node <node id>: Node id
```

### Description

Removes a cell from the netlist.

### Example

```
remove_old_cell -node 34
```

## remove_output_port

### Usage

```
remove_output_port [-gen_id <gen id>] [-literal_index <literal index>] [-node <node id>]
[-port_type <port type>]
```

### Options

`-gen_id <gen id>: Output port generic ID`

`-literal_index <literal index>: Literal index of output port`

`-node <node id>: ID of the node for removing the port`

`-port_type <port type>: Type of output port`

### Description

Deletes the specified output port.

To use this command, the output port must exist.

Returns 1 if successful. Returns 0, otherwise.

### Example

```
remove_output_port -node 0 -port_type DATAA
remove_output_port -node 0 -port_type DATAA -literal_index 0
```

## remove_usage

### Usage

```
remove_usage [-gen_id <gen id>] [-node <node id>] [-port_id <port id>] -port_type <port
type>
```

### Options

`-gen_id <gen id>: Input port generic id`

`-node <node id>: Node id`

`-port_id <port id>: Input port id`

`-port_type <port type>: Type of port usage to remove`

### Description

Removes a usage from the specified input port.

Returns 1, if the usage is removed. Returns 0, otherwise.

### Example

```
remove_usage -node 0 -port_id 4 -port_type DATAC
remove_usage -gen_id 21 -port_type DATAC
```

## routing_path

### Usage

```
routing_path [-dst_gen_id <gen id>] [-file <file name>] [-node <node id>] [-port_id <port
id>] [-table]
```

### Options

`-dst_gen_id <gen id>: Destination port generic ID`

`-file <file name>: File to which to append the routing path`

`-node <node id>: Destination node ID`

`-port_id <port id>: Destination port ID`

`-table: Option to use a tabular format`

### Description

Returns the routing path to the specified destination port of the specified destination node.

### Example

```
routing_path -node 3 -port_id 4
routing_path -node 3 -port_id 4 -table -file my_file
routing_path -dst_gen_id 37
```

## set_batch_mode

### Usage

```
set_batch_mode <on|off>
```

### Options

```
<on|off>: Option to turn batch mode on or off
```

### Description

Sets the batch mode to On or Off.

### Example

```
set_batch_mode on
```

## set_node_info

### Usage

```
set_node_info [-gen_id <gen id>] -info <information type> [-node <node id>]
```

### Options

```
-gen_id <gen id>: Node generic id
-info <information type>: Type of information and the new value
-node <node id>: Node id
```

### Description

Sets the requested type of information for the specified node.

To get available information types, use the "get_info_parameters" command.

Returns 1, if the information type and new value are legal for the specified node. Returns 0, otherwise, and the node remains unchanged.

### Example

```
set_node_info -node 3 -info zbt_dc OFF
```

## set_port_info

### Usage

```
set_port_info [-gen_id <gen id>] -info <information type> [-node <node id>] [-port_id
<port id>] [-type <iport|oport>]
```

### Options

`-gen_id <gen id>: Port generic ID`

`-info <information type>: Type of information and the new value`

`-node <node id>: Node ID`

`-port_id <port id>: Port ID`

`-type <iport|oport>: Option to specify the port as an input or output port`

### Description

Sets the requested information for the specified port.

To get available information types, use the get_info_parameters command.

Returns 1, if the information type and new value are legal for the specified port, or returns 0 and the port remains unchanged.

### Example

```
set_port_info -node 3 -info invert 1 -port_id 2 -type iport
```

## set_tile_power_setting

### Usage

```
set_tile_power_setting [-X <X location>] [-Y <Y location>] [-gen_id <gen id>] -setting
<power setting>
```

### Options

`-X <X location>: X location`

`-Y <Y location>: Y location`

`-gen_id <gen id>: Generic ID`

`-setting <power setting>: Power setting`

### Description

Sets the High-Speed/Low Power setting of the tile at the specified location.

### Example

```
set_tile_power_setting -gen_id 12345 -setting "High Speed"
set_tile_power_setting -X 12 -Y 5 -setting "Low Power"
```

## undo_command

### Usage

```
undo_command <position>
```

### Options

```
<position>: Position
```

### Description

Undo the command at the specified position in the command stack.

### Example

```
undo_command 5
```

## update_node_loc

### Usage

```
update_node_loc
```

### Options

None

### Description

This command uses the node_loc.loc file to update node locations using the ECO fitter. Any warnings are stored in the text_eco_warnings.loc file. You must open a project and call read_netlist before using this command. A node_loc.loc file must exist before running this command (see get_node_loc for instructions on creating a .loc file).

### Example

```
project_open my_proj
   read_netlist
   #ensure that the .loc file exists and is ready with desired node
   # locations
   update_node_loc
   project_close
```

## database_manager

This package contains the set of Tcl functions for managing the version-compatible database files.

This package is available for loading in the following executable:

- quartus_cdb

This package includes the following commands:

## export_database

### Usage

```
export_database <directory>
```

### Options

```
<directory>: Directory of version-compatible database files to which to write
```

### Description

Exports the project database to version-compatible database files that are placed in the specified directory.

### Example

```
## Open the project, run a compilation, and
## export the project database to
## version-compatible database files
load_package database_manager
load_package flow
set project chiptrip
project_new $project -overwrite
execute_flow -compile
export_database backup
project_close $project
```

## generate_bottom_up_scripts

### Usage

```
generate_bottom_up_scripts [-bottom_up_scripts_output_directory
<bottom_up_scripts_output_directory value>] [-disable_auto_global_promotion
<disable_auto_global_promotion value>] [-include_all_logiclock_regions
<include_all_logiclock_regions value>] [-include_design_partitions
<include_design_partitions value>] [-include_global_signal_promotion
<include_global_signal_promotion value>] [-include_logiclock_regions
<include_logiclock_regions value>] [-include_makefiles <include_makefiles value>]
[-include_pin_locations <include_pin_locations value>] [-include_project_creation
<include_project_creation value>] [-include_timing_assignments
<include_timing_assignments value>] [-include_virtual_input_pin_timing
<include_virtual_input_pin_timing value>] [-include_virtual_output_pin_timing
<include_virtual_output_pin_timing value>] [-include_virtual_pin_locations
<include_virtual_pin_locations value>] [-include_virtual_pins <include_virtual_pins
value>] [-remove_existing_regions <remove_existing_regions value>]
[-virtual_input_pin_delay <virtual_input_pin_delay value>] [-virtual_output_pin_delay
<virtual_output_pin_delay value>]
```

### Options

-bottom_up_scripts_output_directory <bottom_up_scripts_output_directory value>: Option to specify the destination directory for generated scripts

-disable_auto_global_promotion <disable_auto_global_promotion value>: Option to turn off automatic global signal promotion in the lower levels

-include_all_logiclock_regions <include_all_logiclock_regions value>: Option to include all LogicLock regions from top-level project in each script.

-include_design_partitions <include_design_partitions value>: Option to include design partitions from top-level in lower-level projects

-include_global_signal_promotion <include_global_signal_promotion value>: Option to promote any top-level global signal to global in the lower-levels

-include_logiclock_regions <include_logiclock_regions value>: Option to include LogicLock regions from top-level entity

-include_makefiles <include_makefiles value>: Option to also generate makefiles in addition to Tcl scripts

-include_pin_locations <include_pin_locations value>: Option to lock ports connected to top-level IOs to the top-level location

-include_project_creation <include_project_creation value>: Option to create the lower-level project and appropriate directory structure

-include_timing_assignments <include_timing_assignments value>: Option to include timing assignments from top-level in lower-level projects

-include_virtual_input_pin_timing <include_virtual_input_pin_timing value>: Option to include timing constraints representing maximum delay to all created virtual input pins from the driving modules

-include_virtual_output_pin_timing <include_virtual_output_pin_timing value>: Option to include timing constraints representing maximum delay from all created virtual output pins to the destination modules

-include_virtual_pin_locations <include_virtual_pin_locations value>: Option to lock created virtual pins to location assigned in top-level entity

-include_virtual_pins <include_virtual_pins value>: Option to include virtual pin creation on appropriate lower-level ports

-remove_existing_regions <remove_existing_regions value>: Option to include commands that remove any existing LogicLock regions in the lower-level

`-virtual_input_pin_delay <virtual_input_pin_delay value>: Option to specify the virtual input pin timing constraint value in nanoseconds`

`-virtual_output_pin_delay <virtual_output_pin_delay value>: Option to specify the virtual output pin timing constraint value in nanoseconds`

## Description

### Overview

This tool is designed for use with a top-level project containing incremental compilation design partitions. When run, it generates scripts and makefiles which allow an easy conversion from top-down design methodology (all partitions in one project) to a bottom-up design methodology (separate projects for each partition).

One Tcl script is generated for each partition. The Tcl script contains all top-level assignments relevant to the given partition, and optionally contains commands to create the lower-level project for the partition if it does not exist. Each script also contains optionally generated commands that can help guide the lower-level placement for better results when exporting to the top-level project. You can customize the content of the Tcl scripts with options.

In addition to generating Tcl scripts, you can also generate makefiles to create the lower-level projects with the generated Tcl scripts and maintain them as source files change. The tool will also build a 'master_makefile' which builds all lower-level projects, exports the results to the top-level project and performs a top-level compilation. The makefiles are auto-generated and are designed for use with GNU make. The makefiles also support parallel compilation of the the lower-level projects by using the '-j' option of GNU make on systems with multiple processors.

### Optional Content

As mentioned above, you can customize the content of the Tcl files with any of the following options.

`-include_makefiles <on|off>`

Default is on.

Option to generate makefiles for lower-level projects in addition to the Tcl scripts. One makefile is generated for each lower-level project, for the top-level project and for the overall design (known as the master_makefile). The master makefile simply invokes all other makefiles.

Makefiles are designed to work with GNU make and support the '-j' option which allows parallel compilation of the lower-level projects. The master makefile is all that needs to be called by the user to ensure the lower-level projects are up to date and that the top-level project has imported the latest versions of the lower-level projects. You can invoke the master makefile as follows (you must not turn off --include_project_creation_in_bottom_up_scripts for this to work without modification):

```
gnumake -f master_makefile.mak -j2
```

<The '-j2' means there are 2 processors to use.>

Makefiles are placed in the directory of the project they control if project creation is enabled and appropriate directories are automatically filled in. If you elect not to have the tool create projects for you, all makefiles are placed in the specified output directory and the user must fill in the directory variables at the top of each makefile so that the tool knows where the lower-level projects can be found.

`-include_project_creation <on|off>`

Default is on.

When you turn on this option, generated Tcl scripts contain commands to create the lower-level projects if they do not exist. The tool will create projects in subdirectories under the output directory, named according to the corresponding partition's name.

`-include_virtual_pins <on|off>`

Default is on.

When you turn on this option, generated Tcl scripts contain commands to mark all lower-level pins that connect to other design entities in the top-level (i.e. not directly to IOs) as 'virtual pins'.

`-include_virtual_pin_timing <on|off>`

Default is on.

When you turn on this option, generated Tcl scripts contain INPUT_MAX_DELAY or OUTPUT_MAX_DELAY assignments to constrain all paths to or from the newly created virtual pins (see -include_virtual_pins). This helps constrain the placement appropriately and produce a higher quality top-level solution.

The option is ignored if -include_virtual_pins off is used.

If you use this option, you must also specify the delay (in nanoseconds) to be used in the constraints with the -virtual_pin_delay=<val> command.

`-include_virtual_pin_locations <on|off>`

Default is on.

When you turn on this option, generated Tcl scripts contain location constraints on the newly created virtual pins (see -include_virtual_pins). The pins will be locked to their top-level source (for input pins) or sink (for output pins) location.

The option is ignored if '-include_virtual_pins off' is used.

`-include_logiclock_regions <on|off>`

Default is on.

When you turn on this option, generated Tcl scripts contain the top-level LogicLock regions associated with this partition. This helps ensure the lower-level project places its logic where the top-level project expects it.

`-include_all_logiclock_regions <on|off>`

Default is on.

When you turn on this option, generated Tcl scripts contain all of the top-level LogicLock regions. Regions with logic not associated with the script's target partition act as placeholders and will be both empty and marked as reserved. This command helps give the lower-level project a better idea of how it fits into the final, top-level design.

The option is ignored if '-include_logiclock_regions off' is used.

`-include_global_signal_promotion <on|off>`

Default is off.

When you turn on this option, generated Tcl scripts contain commands that force any signals that were promoted to 'global' in the top-level to be promoted in the lower-level.

`-include_pin_locations <on|off>`

Default is on.

When you turn on this option, generated Tcl scripts contain commands that lock any lower-level pins connected directly to IOs in the top-level to the IO bank they were placed at in the top-level. This helps keep a consistent pin placement amongst projects.

`-include_timing_assignments <on|off>`

Default is on.

When you turn on this option, generated Tcl scripts contain all relevant timing assignments from the top-level.

`-include_design_partitions <on|off>`

Default is on.

When you turn on this option, generated Tcl scripts contain all relevant design partition assignments from the top-level.

`-remove_existing_regions <on|off>`

Default is on.

When you turn on this option, generated Tcl scripts contain commands that remove any LogicLock regions that exist in the project the script is being called on.

`-disable_auto_global_promotion <on|off>`

Default is off.

When you turn on this option, generated Tcl scripts contain commands that disable auto global signal promotion in the lower levels.

`-bottom_up_scripts_output_directory <output_directory>`

Default is current project directory.

Specifies the output directory for the scripts and makefiles. If none is set, the working directory of the project is used as a default.

`-virtual_input_pin_delay <delay_in_ns>`

No default.

Specifies a delay, in nanoseconds, that is used to constrain all paths from any of the newly created virtual input pins in the lower-levels. This is to help guide the lower-level placement and produce a better quality top-level result.

The value represents the maximum acceptable delay for an inter-partition signal to arrive at the project's virtual input pin from another module. The value helps guide lower-level placement.

`-virtual_output_pin_delay <delay_in_ns>`

No default.

Specifies a delay, in nanoseconds, that is used to constrain all paths to any of the newly created virtual output pins in the lower-levels. This is to help guide the lower-level placement and produce a better quality top-level result.

The value represents the maximum acceptable delay for an inter-partition signal driven by the virtual output pin to arrive at its destination. The value helps guide lower-level placement.

## import_database

### Usage

```
import_database <directory>
```

### Options

```
<directory>: Directory of version-compatible database files from which to read
```

### Description

Imports the project database from version-compatible database files that reside in the specified directory.

### Example

```
## Open the project, import from version-compatible
## database files, and run the Timing Analyzer
load_package database_manager
load_package flow
set project chiptrip
project_open $project
import_database backup
execute_module -tool tan
project_close $project
```

## device

This package contains the set of Tcl functions for accessing information from the Quartus II device database.

This package is loaded by default in the following executables:

- quartus_cdb
- quartus_eda
- quartus_sh
- quartus_sim
- quartus_sta

This package is available for loading in the following executables:

- quartus
- quartus_si
- quartus_tan

This package includes the following commands:

## get_family_list

### Usage

```
get_family_list
```

### Options

None

### Description

Returns a list of available families.

### Example

```
get_family_list
```

## get_part_info

### Usage

```
get_part_info [-default_voltage] [-device] [-family] [-family_variant] [-package]
[-pin_count] [-speed_grade] [-temperature_grade] <part>
```

### Options

```
-default_voltage: Option to get the default core voltage (such as 0.9V or 1.1V)
-device: Option to get device name (such as EP1S25 or EP1S80)
-family: Option to get family name (such as Stratix or Cyclone)
-family_variant: Option to get family variant (such as Base, E or GX)
-package: Option to get package name (such as FBGA or BGA)
-pin_count: Option to get total number of pins in the package
-speed_grade: Option to get speed grade (such as 5, 6, or 7)
-temperature_grade: Option to get temperature grade of the package (such as COMMERCIAL
or INDUSTRIAL)
<part>: Part name
```

### Description

Returns part characteristics for the specified part.

If you use multiple options, the command returns a list in the following order:

```
    <family> <device> <package> <pin_count> <speed grade> <temperature_grade>
<family_variant>
```

### Example

```
tcl> get_part_info -family EP1S25F780C5
tcl> Stratix

tcl> get_part_info -family -device EP1S25F780C5
tcl> Stratix EP1S25

tcl> get_part_info -device -package -pin_count -speed_grade EP1S25F780C5
tcl> EP1S25 FBGA 780 5
```

## get_part_list

### Usage

```
get_part_list [-device <value>] [-family <value>] [-package <value>] [-pin_count
<value>] [-speed_grade <value>]
```

### Options

```
-device <value>: Option to match device name
-family <value>: Option to match family name
-package <value>: Option to match package name
-pin_count <value>: Option to match pin count
-speed_grade <value>: Option to match speed grade
```

### Description

Returns a list of available parts based on the options that are specified. Examples are as follows:

Return a list of all supported parts

```
get_part_list
```

Return a list of all supported parts for Cyclone

```
get_part_list -family Cyclone
```

Return a list of all supported parts with the FBGA package and 780 pins

```
get_part_list -package fbga -pin_count 780
```

### Example

```
get_part_list -family Stratix
```

## report_device_info

### Usage

```
report_device_info <device>
```

### Options

```
<device>: Device name
```

### Description

Returns a string value containing the report with information about the specified device, such as the following:

```
    Available parts
    Some additional information specific to the device
```

### Example

```
set report [report_device_info APEX20K1000E]
puts $report
```

## report_family_info

### Usage

```
report_family_info <family>
```

### Options

```
<family>: Family name
```

### Description

Returns a string value containing the report with information about the specified family, such as the following:

```
    Available devices
    Available packages
    Available speed grades
    Available pin counts
    Some additional information specific to the family
```

### Example

```
set report [report_family_info Stratix]
puts $report
```

## report_part_info

### Usage

```
report_part_info <part>
```

### Options

```
<part>: Part name
```

### Description

Returns a string value containing the report with information about the specified part, such as the following:

```
    Family name
    Device name
    Package name
    Pin count
    Speed grade
    Any additional information
```

### Example

```
set report [report_part_info EP20K1000EFC33-3]
puts $report
```

## flow

This package contains the set of Tcl functions for running flows or command-line executables.

This package is available for loading in the following executables:

- quartus
- quartus_cdb
- quartus_drc
- quartus_eda
- quartus_sh
- quartus_si
- quartus_sim
- quartus_sta
- quartus_tan

This package includes the following commands:

## execute_flow

### Usage

```
execute_flow [-analysis_and_elaboration] [-check_ios] [-check_netlist] [-compile]
[-compile_and_simulate] [-create_companion_revision] [-dont_export_assignments]
[-early_timing_estimate] [-eco] [-export_database] [-fast_model]
[-generate_functional_sim_netlist] [-import_database] [-incremental_compilation_export]
[-incremental_compilation_import] [-signalprobe] [-vqm_writer]
```

### Options

```
-analysis_and_elaboration: Option to run Analysis & Elaboration

-check_ios: Option to run I/O assignment analysis

-check_netlist: Option to run Check and Save Netlist

-compile: Option to run a full compilation

-compile_and_simulate: Option to run a full compilation followed by a simulation

-create_companion_revision: Option to run HardCopy Create Companion Revision

-dont_export_assignments: Option not to export assignments to file. By default, this
command exports assignments before running command-line executables.

-early_timing_estimate: Option to run an Early Timing Estimate

-eco: Option to run a Fitter ECO compilation

-export_database: Option to export a version-compatible database

-fast_model: Option to run Classic Timing Analyzer (Fast Timing Model)

-generate_functional_sim_netlist: Option to generate a functional simulation netlist

-import_database: Option to import a version-compatible database

-incremental_compilation_export: Option to export a design partition into a Quartus II
Exported Partition (QXP) file

-incremental_compilation_import: Option to import one or more Quartus II Exported
Partition (QXP) files into the design partitions of the current project

-signalprobe: Option to run a SignalProbe compilation

-vqm_writer: Option to run VQM Writer
```

### Description

Runs one or more of the command-line executables using one of the predefined flows, such as "-compile" or "-signalprobe". You can run only one flow at a time, so you must use only one option.

Some flows have limited device support or other limitations based on the features used. See documentation for the features in question for details.

The "-export_database" and "-import_database" options use the value of the VER_COMPATIBLE_DB_DIR assignment for the version-compatible database files directory, defaulting to "export_db".

The "-incremental_compilation_export" option uses the value of the INCREMENTAL_COMPILATION_EXPORT_FILE global assignment for the path of the Quartus II Exported Partition (QXP) file to be created. The value of the INCREMENTAL_COMPILATION_EXPORT_PARTITION_NAME global assignment should specify the name of the partition to be exported. The value of the INCREMENTAL_COMPILATION_EXPORT_NETLIST_TYPE global assignment (which can either have value POST_SYNTH or POST_FIT) determines whether post-synthesis or post-fitting results should be exported. Finally, the value of the INCREMENTAL_COMPILATION_EXPORT_ROUTING global assignment specifies whether routing should be exported when a post-fit netlist is generated.

The "-incremental_compilation_import" option uses the following partition assignments to determine the location of the QXP files, and how importation should be performed, on a per-partition basis:

PARTITION_IMPORT_FILE
PARTITION_IMPORT_PROMOTE_ASSIGNMENTS
PARTITION_IMPORT_NEW_ASSIGNMENTS
PARTITION_IMPORT_EXISTING_ASSIGNMENTS
PARTITION_IMPORT_EXISTING_LOGICLOCK_REGIONS

The "-vqm_writer" option uses the value of the LOGICLOCK_INCREMENTAL_COMPILE_FILE assignment for the VQM output directory, defaulting to "atom_netlists".

All assignments are exported first automatically, as if you called the "export_assignments" command first, unless the -dont_export_assignments option is specified.

You must use the Tcl command "catch" to determine whether the predefined flow ran successfully or not, as in the following example:

```
if {[catch {execute_flow -compile} result]} {
   puts "\nResult: $result\n"
   puts "ERROR: Compilation failed. See report files.\n"
} else {
   puts "\nINFO: Compilation was successful.\n"
}
```

## Example

```
# To run quartus_map, quartus_fit, quartus_tan, quartus_asm
# or other executables based on options. (Refer to "Using
# Compilation Flows," "Compiling Designs," and "Specifying
# Compiler Settings" in Quartus(R) II online Help for more
# information.)
execute_flow -compile

# To determine if compilation was successful or not
# and print out a personalized message.
if {[catch {execute_flow -compile} result]} {
   puts "\nResult: $result\n"
   puts "ERROR: Compilation failed. See report files.\n"
} else {
   puts "\nINFO: Compilation was successful.\n"
}

# To perform a full compilation followed by a simulation
execute_flow -compile_and_simulate
```

## execute_hc

### Usage

```
execute_hc [-archive <file name>] [-compare] [-create_companion <revision name>]
[-handoff_report] [-hc_ready] [-min_archive]
```

### Options

-archive <file name>: Option to archive HardCopy Handoff Files into the specified file name

-compare: Option to run HardCopy Companion Revision Comparison

-create_companion <revision name>: Option to create/overwrite the specified companion revision

-handoff_report: Option to run HardCopy Handoff Report

-hc_ready: Option to generate HardCopy Design Readiness Check report

-min_archive: Option to exclude design files when archiving HardCopy Handoff Files

### Description

Runs one of the predefined HardCopy flows.

All created or modified Quartus II Settings File (.qsf) assignments are automatically exported before running the predefined flow, as if you called the "export_assignments" command beforehand.

You must use the Tcl command "catch" to determine whether the predefined flow ran successfully or not, as in the following example:

```
if {[catch {execute_hc -compare} result]} {
   post_message "Result: $result"
   post_message -type error "HardCopy Companion Revision Comparison failed. See report
files."
} else {
   post_message "HardCopy Companion Revision Comparison was successful."
}
```

### Example

```
   # Load the ::quartus::flow Tcl package
load_package flow

   # Open the FPGA revision named "my_fpga"
project_open my_design -revision my_fpga

   # Compile the FPGA "my_fpga" revision
execute_flow -compile

   # Create a HardCopy companion revision named "my_hcii"
execute_hc -create_companion my_hcii

   # Set "my_hcii" as the current revision before compiling it
set_current_revision my_hcii

   # Compile the HardCopy "my_hcii" revision
execute_flow -compile

   # Compare the two HardCopy companion revisions
execute_hc -compare

   # Generate HardCopy Design Readiness Check report
execute_hc -hc_ready
```

```
   # Generate a HardCopy Handoff Report
execute_hc -handoff_report

   # Archive the HardCopy Handoff Files into
   # the file named "my_hcii_handoff.qar"
execute_hc -archive my_hcii_handoff.qar

   # Close the project
project_close
```

## execute_module

### Usage

```
execute_module [-args <arguments>] [-dont_export_assignments] [-tool
<asm|cdb|drc|eda|fit|map|pow|sta|sim|tan|si|cpf>]
```

### Options

```
-args <arguments>: Option to specify arguments for the executable
```

```
-dont_export_assignments: Option not to export assignments to file. By default, this
command exports assignments before running command-line executables.
```

```
-tool <asm|cdb|drc|eda|fit|map|pow|sta|sim|tan|si|cpf>: Option to run the specified
executable
```

### Description

Runs one of the command-line executables, such as quartus_map or quartus_fit. If the -args option is specified, the arguments are passed to the command-line executable.

All assignments are exported automatically first, as if the "export_assignments" command was called first, unless -dont_export_assignments option is specified.

You must use the Tcl command "catch" to determine whether the command-line executable ran successfully or not, as in the following example:

```
if {[catch {execute_module -tool map} result]} {
    puts "\nResult: $result\n"
    puts "ERROR: Analysis & Synthesis failed. See the report file.\n"
} else {
    puts "\nINFO: Analysis & Synthesis was successful.\n"
}
```

### Example

```
# Run quartus_map using device family Stratix and device part
# EP1S10B672C6.
execute_module -tool map -args "--family=Stratix --part=EP1S10B672C6"

# Compile using a set of executables
execute_module -tool map
execute_module -tool fit
execute_module -tool tan
execute_module -tool asm
execute_module -tool eda

# To determine if Analysis & Synthesis was successful or not
# and print out a personalized message.
if {[catch {execute_module -tool map} result]} {
    puts "\nResult: $result\n"
    puts "ERROR: Analysis & Synthesis failed. See the report file.\n"
} else {
    puts "\nINFO: Analysis & Synthesis was successful.\n"
}
```

# help

This package contains the set of Tcl functions for displaying help for the Quartus II API for Tcl.

This package includes the following commands:

## help

### Usage

```
help [-cmd <command name>] [-examples] [-pkg <package name>] [-quartus] [-tcl]
[-timequestinfo] [-version <version number>] <package or command name>
```

### Options

```
-cmd <command name>: Option to display long help for a Quartus II Tcl command

-examples: Option to display examples of Quartus II Tcl usage

-pkg <package name>: Option to display help for an available Quartus II Tcl package

-quartus: Option to display help on the predefined global "quartus" Tcl array

-tcl: Option to display overview on Quartus II Tcl usage

-timequestinfo: Option to display help on the predefined global "TimeQuestInfo" Tcl array

-version <version number>: Option to specify the package version for the specified
Quartus II Tcl package or command

<package or command name>: Name of an available Quartus II Tcl package or command. This
is functionally equivalent to the -pkg and -cmd options.
```

### Description

For convenience, you can omit the "::quartus::" package prefix when you query for help on a package. For example, you can type either of the following to view help for the ::quartus::project package:

```
     help ::quartus::project
     help project
```

### Example

```
1) Type "help <package name>"
   to view help for a Quartus II Tcl package.

   Example: help ::quartus::project

2) Type "help <command name>"
   to view long help for a Quartus II Tcl command.

   Example: help project_open

3) Type "help -quartus"
   to view help on the predefined global "quartus" Tcl array.

   Example: help -quartus

4) Type "help -examples"
   to view examples of Quartus II Tcl usage.

   Example: help -examples
```

## incremental_compilation

This package contains the set of Tcl functions for manipulating QIC Partitions and LogicLock regions and settings related to the QIC and LogicLock features.

This package is available for loading in the following executables:

- quartus
- quartus_cdb
- quartus_sh
- quartus_sta
- quartus_tan

This package includes the following commands:

## auto_partition_design

### Usage

```
auto_partition_design [-logic_max <param>] [-logic_min <param>] [-max_partitions
<param>] [-percent_to_partition <param>] [-top_level <param>]
```

### Options

```
-logic_max <param>: Maximum logic (LEs/Comb ALUTs + Registers) in new partitions.
-logic_min <param>: Minimum logic (LEs/Comb ALUTs + Registers) in new partitions.
-max_partitions <param>: Maximum number of new partitions to create.
-percent_to_partition <param>: Target % of design to add to partitions
-top_level <param>: Top-level for partitioning. Top-level & decendents are partitioned
```

### Description

Invokes the Quartus II Auto-Partitioner to automatically create design partitions for incremental compilation.

The Quartus II Auto-Partitioner searches through an existing design hierarchy to extract modules that make good candidates for partitioning. The partitioner uses previous compilation results, either after Analysis & Synthesis and Partition Merge, or after the Fitter, to make partitioning decisions.

By default, the Quartus II Auto-Partitioner will consider making new partitions involving the designÔæís top-level module and any of its corresponding descendant modules which are not already contained in another design partition.

Use the -top_level <name> option to specify a particular design module to begin partitioning exploration from. Only those design modules which are decendents of Ôæñtop_level in the hierarchy are considered for partitioning.

Use the -max_partitions <number> option to specify the maximum number of new design partitions to create on this partitioning iteration.

Use the -percent_to_partition <PCT> option to set a target on how much design logic should be added to partitions as a percentage of overall design size.

Use the -logic_min <number> and -logic_max <number> options to specify ranges for the amount of logic that can be contained in any identified partitions. For the purpose of this option, logic is computed as: Logic = (LEs or Combinational ALUTS) + (Dedicated Logic Registers)

ADDITIONAL NOTES:

The Quartus II Auto-Partitioner does not remove or modify any existing design partitions defined prior to auto-partitioning.

Device resource utilization changes after making design partitions. Device utilization thresholds used to drive partitioning decisions are based on previous compilation results.

The Quartus II Auto-Partitioner is meant to serve as a guide for quickly creating an initial design partitioning scheme. The ability of this feature to provide high-quality partitioning solutions depends heavily on the suitability of the design HDL for use with incremental compilation.

For more information on best practices for creating partitions for use with Quartus II Incremental compilation, please refer to the Quartus II Handbook.

## Example

```
## Partition the design using default settings.
auto_partition_design

## Partition the design with limits on partition size
auto_partition_design -logic_min 3000 -logic_min 20000

## Partition approximately 65% of the design using up to 20 partitions
auto_partition_design -percent_to_partition 65 -max_partitions 20
```

## create_partition

### Usage

```
create_partition -contents <hierarchy name> -partition <partition name>
```

### Options

```
-contents <hierarchy name>: Partition contents (hierarchy assigned to Partition)
-partition <partition name>: Partition name
```

### Description

Creates a partition, unless the partition specified with the -partition option already exists. The Quartus II software creates a partition assignment to the specified hierarchy name.

### Example

```
package require ::quartus::incremental_compilation

project_open my_design

## Create a partition on child:inst1
create_partition -partition my_partition -contents child:inst1

project_close
```

## delete_all_logiclock

### Usage

```
delete_all_logiclock
```

### Options

None

### Description

Removes all LogicLock regions from the design.

### Example

```
package require ::quartus::incremental_compilation

project_open my_design

delete_all_logiclock

project_close
```

## delete_all_partitions

### Usage

```
delete_all_partitions
```

### Options

None

### Description

Removes all design partitions.

### Example

```
package require ::quartus::incremental_compilation

project_open my_design

delete_all_partitions

project_close
```

## delete_logiclock

### Usage

```
delete_logiclock -region <region name>
```

### Options

```
-region <region name>: Region name
```

### Description

Deletes the LogicLock™region specified by the "-region_name" option.

### Example

```
package require ::quartus::logiclock

## Remove the region named "one"
project_open my_design

delete_logiclock -region one

project_close
```

## delete_partition

### Usage

```
delete_partition -partition <partition name>
```

### Options

```
-partition <partition name>: Partition name
```

### Description

Deletes the partition specified with the -partition option.

### Example

```
package require ::quartus::incremental_compilation

project_open my_design

delete_partition -partition my_partition

project_close
```

## export_partition

### Usage

```
export_partition [-netlist_type <netlist type>] -partition <partition name> -qxp <QXP
filename> [-routing <export routing>]
```

### Options

```
-netlist_type <netlist type>: Netlist type
-partition <partition name>: Partition name
-qxp <QXP filename>: QXP fiilename
-routing <export routing>: Export routing
```

### Description

Exports the partition specified with the -partition option to a QXP file with the name specified by the -qxp option.

### Example

```
package require ::quartus::incremental_compilation

project_open my_design

export_partition -partition my_partition -qxp top.qxp

project_close
```

## get_logiclock

### Usage

```
get_logiclock [-auto_size] [-height] [-member_state] [-origin] [-parent] [-region
<region name>] [-reserved] [-soft] [-state] [-width]
```

### Options

```
-auto_size: Region's auto-size property

-height: Region height

-member_state: Region's member-state property

-origin: Region origin

-parent: Region's parent

-region <region name>: Region name

-reserved: Region's reserved property

-soft: Region's soft property

-state: Region's state name

-width: Region width
```

### Description

Returns a list of LogicLock™regions or a specific region property. The command returns a list of all the regions in the current project if you do not specify any options.

You must use the region property options when you use the "-region" option.

### Example

```
package require ::quartus::logiclock

project_open my_design

## Get a list of regions
set regions [get_logiclock]

## Print the size of each region
foreach region $regions {
   puts "size of $region is [get_logiclock -region $region -height] x \
        [get_logiclock -region $region -width]"
}

project_close
```

## get_logiclock_contents

### Usage

```
get_logiclock_contents [-node_locations] [-region <region name>] [-root_region]
```

### Options

```
-node_locations: Option to list node-locations instead of region assignments
-region <region name>: Region name
-root_region: Root region name
```

### Description

Returns assigned contents or back-annotated nodes for the specified region. The command returns a list of all the region assignments to the LogicLock™region specified by the "-region" option.

Each member is represented by one element in the list. Each member has the following format:

```
{{<from>} {<to>} {<region>} {<exclude_from>} {<exclude_to>} {<exclude_node>}}
```

When the member is a simple member, rather than a path member, only the second and third elements are valid. (The first element is always empty in this case so that common parsing code can be used.)

When you use the "-node_locations" option, the back-annotated contents are returned.

### Example

```
package require ::quartus::logiclock

project_open my_design

## Initialize interface before doing anything else
initialize_logiclock

## Get a list of region members
set members [get_logiclock_contents -region region_one]

## Parse elements of member list
foreach member $members {
   set from         [lindex $member 0]
   set to           [lindex $member 1]
   set region       [lindex $member 2]
   set exclude_from [lindex $member 3]
   set exclude_to   [lindex $member 4]
   ## Do something with member attributes
   ...
}

project_close
```

## get_partition

### Usage

```
get_partition [-base_filename] [-color] [-contents] [-full_filename] [-netlist_type]
[-partition <partition name>] [-preservation_level] [-qxp]
```

### Options

`-base_filename: Base filename`

`-color: Partition display color`

`-contents: Partition contents (hierarchy assigned to Partition)`

`-full_filename: Full filename`

`-netlist_type: Netlist type`

`-partition <partition name>: Partition name`

`-preservation_level: Preservation level`

`-qxp: QXP filename (if partition is set to Import)`

### Description

Returns a list of partitions or a specific partition property. The command returns a list of all the partitions in the current project if you do not specify any options.

You must use the partition property options when you use the -partition option.

### Example

```
package require ::quartus::incremental_compilation

project_open my_design

## Get a list of all partitions
set partitions [get_partition]

foreach p $partitions {
   puts "Partition $p is using Netlist Type \
        [get_partition -partition $p -netlist_type]"
}

project_close
```

## get_partition_file_list

### Usage

```
get_partition_file_list [-exclude_compiled_partitions] [-use_placeholders]
```

### Options

```
-exclude_compiled_partitions: Option to exclude compiled_partitions folder from the
filelist
```

```
-use_placeholders: Option to generate filelist using a placeholder
```

### Description

Returns the list of files that represent the incremental compilation database for the active revision.

### Example

```
package require ::quartus::incremental_compilation

set filelist [get_partition_file_list]
foreach f $filelist {
    puts $f
}
```

## import_partition

### Usage

```
import_partition [-create_assignments <create assignments>] [-include_pin_assignments
<include pin assignments>] -partition <partition name> [-promote_assignments <promote
assignments>] -qxp <QXP filename> [-update_assignments <update assignments>]
[-update_logiclock <update logiclock>]
```

### Options

```
-create_assignments <create assignments>: Create assignments

-include_pin_assignments <include pin assignments>: Import pin assignments

-partition <partition name>: Partition name

-promote_assignments <promote assignments>: Promote assignments

-qxp <QXP filename>: QXP fiiilename

-update_assignments <update assignments>: Update assignments

-update_logiclock <update logiclock>: Update LogicLock region assignments
```

### Description

Imports the Partition specified with the -partition option from a QXP file with the name specified by the -qxp option.

### Example

```
package require ::quartus::incremental_compilation

project_open my_design

import_partition -partition my_partition -qxp top.qxp

project_close
```

## partition_netlist_exists

### Usage

```
partition_netlist_exists -netlist_type <netlist type> -partition <partition name>
```

### Options

```
-netlist_type <netlist type>: Netlist type
-partition <partition name>: Partition name
```

### Description

Checks whether the netlist type exists for the specified partition.

### Example

```
package require ::quartus::incremental_compilation

project_open my_design
set partitions [get_partition]
foreach p $partitions {
    set netlist_type [get_partition -partition $p -netlist_type]
    if {![partition_netlist_exists -partition $p -netlist_type \
        $netlist_type]} {
    puts "Netlist type $netlist_type does not exist for Partition \
        $p!"
    } else {
    puts "Netlist type $netlist_type exists for Partition $p!"
    }
}
project_close
```

## set_logiclock

### Usage

```
set_logiclock [-auto_size <true|false>] [-flip] [-floating <true|false>] [-height
<height in labs>] [-members_floating <true|false>] [-origin <chip location>] [-parent
<parent region name>] -region <region name> [-reserved <true|false>] [-soft <true|false>]
[-width <width in labs>]
```

### Options

```
-auto_size <true|false>: Option to set auto-size property
-flip: Option to flip (mirror) region right-to-left on the device
-floating <true|false>: Option to set floating property
-height <height in labs>: Region height
-members_floating <true|false>: Option to set members floating property
-origin <chip location>: Region origin
-parent <parent region name>: Parent region
-region <region name>: Region name
-reserved <true|false>: Option to set reserved property
-soft <true|false>: Option to set soft property
-width <width in labs>: Region width
```

### Description

Creates or changes attributes of LogicLock™regions.

If the region specified by the "-region" option does not exist, the Quartus II software creates it with the specified properties.

Otherwise, the Quartus II software changes the properties of the region specified by the "-region" option to the values specified by the options. Any properties not included as options remain unchanged.

### Example

```
package require ::quartus::logiclock

project_open my_design

## Create a region with default attributes
set_logiclock -region region_one

## Change the size of the region
set_logiclock -region region_one -height 2 -width 3

project_close
```

## set_logiclock_contents

### Usage

```
set_logiclock_contents [-delete_member_of] [-exceptions <resource_string>]
[-exclude_from <true|false>] [-exclude_node <node name>] [-exclude_to <true|false>]
[-from <node name>] [-location <chip location>] [-priority <priority integer>] [-region
<region name>] [-root_region] [-to <node name>]
```

### Options

`-delete_member_of: Option to delete MEMBER_OF statement for assignment`

`-exceptions <resource_string>: If specified, the Fitter assigns all nodes under the target design entity or path to be members of the LogicLock region, except for nodes of types in the specified colon-delimited exceptions list.`

`-exclude_from <true|false>: Option to exclude source in assignment <true|false>`

`-exclude_node <node name>: Option to exclude all paths through the specified node`

`-exclude_to <true|false>: Option to exclude destination in assignment <true|false>`

`-from <node name>: Assignment source`

`-location <chip location>: Location to which node is back-annotated`

`-priority <priority integer>: Priority of wildcard or path-based LogicLock region member`

`-region <region name>: Region name`

`-root_region: Root region`

`-to <node name>: Assignment destination`

### Description

Sets or changes LogicLock™content options. The command assigns nodes, entities, or path-based assignments to a LogicLock region. It also allows you to back-annotate nodes to a specific location.

You can use the -exceptions option to define LogicLock region membership exceptions. If specified, the Fitter assigns all nodes under the target design entity or path to be members of the LogicLock region, except for nodes of types specified in the exceptions list. The exceptions list is a colon-delimited string of the following keywords:

| **Keyword** | **Resource Represented** |
|---|---|
| REGISTER | Registers in logic cells |
| COMBINATIONAL | Combinational elements in logic cells |
| SMALL_MEM | M512 memory blocks, or MLAB cells |
| MEDIUM_MEM | M4K or M9K memory blocks |
| LARGE_MEM | M-RAM or M144K memory blocks |
| DSP | DSP blocks |
| VIRTUAL_PIN | Virtual pins |

You can use the "-exceptions" option only with LogicLock member assignments.

You can use the "-from", "-exclude_node", "-exclude_from", and "-exclude_to" options only with path-based assignments.

The node names may be specified as wildcards.

You can use the "-priority" option only with path-based and wildcard assignments.

You can use the "-location" option only with node assignments.

## Example

```
package require ::quartus::logiclock

project_open my_design

## Initialize interface before doing anything else
initialize_logiclock

## Get all region members
set members [get_logiclock_contents -region region_one]

## Parse elements of member list
foreach member $members {
   set from         [lindex $member 0]
   set to           [lindex $member 1]
   ## Delete all members in this region
   set_logiclock_contents -region region_one -delete_member_of -from \
       from -to to
}
## Get a list of back-annotated contents
set_logiclock_contents -region region_one

project_close
```

## set_partition

### Usage

```
set_partition [-color <color>] [-netlist_type <netlist type>] -partition <partition
name> [-preservation_level <preservation level >] [-qxp <QXP filename>]
```

### Options

```
-color <color>: Partition display color

-netlist_type <netlist type>: Netlist type

-partition <partition name>: Partition name

-preservation_level <preservation level >: Preservation level

-qxp <QXP filename>: QXP filename (if partition is set to Import)
```

### Description

Changes partition attributes.

The Quartus II software changes the properties of the partition specified by the -partition option to the values specified by the options. Any properties not included as options remain unchanged.

### Example

```
package require ::quartus::incremental_compilation

project_open my_design

## Get a list of regions
set partitions [set_partition]

foreach p $partitions {
   set_partition -partition $p -netlist_type "Post-Fit"
}

project_close
```

## insystem_memory_edit

This package contains the set of Tcl functions for editing and reading memory content in an Altera device using the In-System Memory Content Editor.

This package is loaded by default in the following executable:

- quartus_stp

This package includes the following commands:

## begin_memory_edit

### Usage

```
begin_memory_edit -device_name <device name> -hardware_name <hardware name>
```

### Options

```
-device_name <device name>: Name of the device that holds the editable memory instances
-hardware_name <hardware name>: Name of the hardware that connects to the JTAG chain
```

### Description

Start the memory editing sequence. The editing sequence should be terminated with end_memory_edit. The sequence does not have to be terminated unless the device configuration is changed or a different device is edited.

The hardware and device name can be obtained with the get_hardware_names and get_device_names commands from the jtag package.

### Example

```
# Instance 0 is configured as {0 1024 8 RW ROM/RAM mem0}

# Initiate a editing sequence
begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name \
    "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"

# Write memory content using binary string
write_content_to_memory -instance_index 0 -start_address 575 -word_count \
    2 -content "0000001011011100"

# Read back memory content in binary string written
puts \
    [read_content_from_memory -instance_index 0 -start_address 575 \
    -word_count 2 ]

# Write memory content using hexadecimal string
write_content_to_memory -instance_index 0 -start_address 575 -word_count \
    2 -content "E2F1" -content_in_hex

# Read back memory content in hexadecimal string written
puts \
    [read_content_from_memory -instance_index 0 -start_address 575 \
    -word_count 2 -content_in_hex]

# End the editing sequence
end_memory_edit
```

## end_memory_edit

### Usage

```
end_memory_edit
```

### Options

None

### Description

Terminate the memory editing sequence. The sequence does not have to be terminated unless the device configuration is changed or a different device is edited.

### Example

```
# Instance 0 is configured as {0 1024 8 RW ROM/RAM mem0}

# Initiate a editing sequence
begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name \
    "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"

# Write memory content using binary string
write_content_to_memory -instance_index 0 -start_address 575 -word_count \
    2 -content "0000001011011100"

# Read back memory content in binary string written
puts \
    [read_content_from_memory -instance_index 0 -start_address 575 \
    -word_count 2 ]

# Write memory content using hexadecimal string
write_content_to_memory -instance_index 0 -start_address 575 -word_count \
    2 -content "E2F1" -content_in_hex

# Read back memory content in hexadecimal string written
puts \
    [read_content_from_memory -instance_index 0 -start_address 575 \
    -word_count 2 -content_in_hex]

# End the editing sequence
end_memory_edit
```

## get_editable_mem_instances

### Usage

```
get_editable_mem_instances -device_name <device name> -hardware_name <hardware name>
```

### Options

```
-device_name <device name>: Name of the device that holds the editable memory instances
-hardware_name <hardware name>: Name of the hardware that connects to the JTAG chain
```

### Description

Retrieve a list of editable memory, ROM, or lpm_constant instances.

A list is returned, each element of which shows the configuration of each instance.  This element is an another list that specifies the configuration in the following order: <instance index> <depth> <width> <read/write mode> <instance type> <instance name>. The <read/write mode> can be either "RW" or "W"; <instance type> can be either "ROM/RAM" or "CONSTANT". An example showing a list of two instances of different types is shown below:

```
{0 1024 8 RW ROM/RAM mem0} {1 1 32 RW CONSTANT con0}
```

The hardware and device name can be obtained with the get_hardware_names and get_device_names commands from the jtag package.

It is recommended that you call this command before the TCL command, begin_memory_edit. Within a memory edit sequence, this command can be applied only to the same device, on which the memory edit sequence has started.

### Example

```
# List information of all editable memories
puts "Information on all editable memories:"
puts "index,depth,width,mode,type,name"
foreach instance \
    [get_editable_mem_instances -hardware_name "USB-Blaster \[USB-0\]" \
    -device_name "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"] {
   puts "[lindex $instance 0],[lindex $instance 1],[lindex $instance \
       2],[lindex $instance 3],[lindex $instance 4],[lindex $instance 5]"
}
```

## read_content_from_memory

### Usage

```
read_content_from_memory [-content_in_hex] -instance_index <instance index>
-start_address <starting address> -word_count <word count>
```

### Options

```
-content_in_hex: The memory content string is represented in hexadecimal format
-instance_index <instance index>: Index of the editable memory instance to read
-start_address <starting address>: The lowest memory address to be read
-word_count <word count>: The number of contiguous memory words to be read
```

### Description

Retrieves the memory content represented in the bit stream from the specified editable memory instance starting from the specified address.

The memory content string is in the same format as the input content string in the TCL command write_content_to_memory.

### Example

```
# Instance 0 is configured as {0 1024 8 RW ROM/RAM mem0}

# Initiate a editing sequence
begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name \
    "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"

# Write memory content using binary string
write_content_to_memory -instance_index 0 -start_address 575 -word_count \
    2 -content "0000001011011100"

# Read back memory content in binary string written
puts \
    [read_content_from_memory -instance_index 0 -start_address 575 \
    -word_count 2 ]

# Write memory content using hexadecimal string
write_content_to_memory -instance_index 0 -start_address 575 -word_count \
    2 -content "E2F1" -content_in_hex

# Read back memory content in hexadecimal string written
puts \
    [read_content_from_memory -instance_index 0 -start_address 575 \
    -word_count 2 -content_in_hex]

# End the editing sequence
end_memory_edit
```

## save_content_from_memory_to_file

### Usage

```
save_content_from_memory_to_file -instance_index <instance index> -mem_file_path <path>
-mem_file_type <type>
```

### Options

```
-instance_index <instance index>: Index of the editable memory instance to read
```

```
-mem_file_path <path>: Path to the memory file in which to save the memory content
```

```
-mem_file_type <type>: Type of the memory file such as "mif" or "hex"
```

### Description

Retrieves the entire memory contents from the specified editable memory instance starting from address 0 and saves it into the specified memory file.

### Example

```
# Initiate a editing sequence
begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name \
    "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"

# Write memory content using the hex memory file
update_content_to_memory_from_file -instance_index 0 -mem_file_path \
    "image_8x1024.hex" -mem_file_type hex

# Read memory content and save back to a hex memory file
save_content_from_memory_to_file -instance_index 0 -mem_file_path \
    "exported_image_8x1024.hex" -mem_file_type hex

# Write memory content using the mif memory file
update_content_to_memory_from_file -instance_index 0 -mem_file_path \
    "exported_image_8x1024.mif" -mem_file_type mif

# Read memory content and save back to a mif memory file
save_content_from_memory_to_file -instance_index 0 -mem_file_path \
    "image_8x1024.mif" -mem_file_type mif

# End the editing sequence
end_memory_edit
```

## update_content_to_memory_from_file

### Usage

```
update_content_to_memory_from_file -instance_index <instance index> -mem_file_path
<path> -mem_file_type <file type>
```

### Options

`-instance_index <instance index>: Index of the editable memory instance to modify`

`-mem_file_path <path>: Path to the memory file to load the memory content`

`-mem_file_type <file type>: Type of the memory file such as "mif" or "hex"`

### Description

Writes the data stored in the memory file into the specified memory instance starting from address 0.

### Example

```
# Initiate a editing sequence
begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name \
    "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"

# Write memory content using the hex memory file
update_content_to_memory_from_file -instance_index 0 -mem_file_path \
    "image_8x1024.hex" -mem_file_type hex

# Read memory content and save back to a hex memory file
save_content_from_memory_to_file -instance_index 0 -mem_file_path \
    "exported_image_8x1024.hex" -mem_file_type hex

# Write memory content using the mif memory file
update_content_to_memory_from_file -instance_index 0 -mem_file_path \
    "exported_image_8x1024.mif" -mem_file_type mif

# Read memory content and save back to a mif memory file
save_content_from_memory_to_file -instance_index 0 -mem_file_path \
    "image_8x1024.mif" -mem_file_type mif

# End the editing sequence
end_memory_edit
```

## write_content_to_memory

### Usage

```
write_content_to_memory -content <content string> [-content_in_hex] -instance_index
<instance index> -start_address <starting address> -word_count <word count>
```

### Options

-content <content string>: A string that represents all word values concatenated together in order in either binary or hexadecimal format

-content_in_hex: The memory content string is represented in hexadecimal format

-instance_index <instance index>: Index of the editable memory instance to modify

-start_address <starting address>: The lowest memory address to be modified

-word_count <word count>: The number of contiguous memory words to be modified

### Description

Writes the data represented in the bit stream into the specified editable memory instance starting from the specified address.

The bit stream should be ordered by word from high address to low address, contiguously without gaps or delimiters. If the starting address is ADDR, and word count is N, the order is <word @ ADDR + N - 1> ... <word @ ADDR + 1><word @ ADDR> In each word, the MSB is on the left, LSB is on the right. The bit stream can be in either binary or hexadecimal. For example, if the word width is 8, and two words, 1 and 128, are written to address 0 and 1 respectively, the bitstream should be "1000000000000001" in binary or "8001" in hexadecimal. The TCL command is write_content_to_memory -instance_index 0 -start_address 0 -word_count 2 -content "1000000000000001" or write_content_to_memory -instance_index 0 -start_address 0 -word_count 2 -content "8001" -content_in_hex

### Example

```
# Instance 0 is configured as {0 1024 8 RW ROM/RAM mem0}

# Initiate a editing sequence
begin_memory_edit -hardware_name "USB-Blaster \[USB-0\]" -device_name \
    "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"

# Write memory content using binary string
write_content_to_memory -instance_index 0 -start_address 575 -word_count \
    2 -content "0000001011011100"

# Read back memory content in binary string written
puts \
    [read_content_from_memory -instance_index 0 -start_address 575 \
    -word_count 2 ]

# Write memory content using hexadecimal string
write_content_to_memory -instance_index 0 -start_address 575 -word_count \
    2 -content "E2F1" -content_in_hex

# Read back memory content in hexadecimal string written
puts \
    [read_content_from_memory -instance_index 0 -start_address 575 \
    -word_count 2 -content_in_hex]

# End the editing sequence
end_memory_edit
```

## insystem_source_probe

This package contains the set of Tcl functions for interacting with the In-System Sources and Probes tool in an Altera device.

This package is loaded by default in the following executable:

- quartus_stp

This package includes the following commands:

## end_insystem_source_probe

### Usage

```
end_insystem_source_probe
```

### Options

None

### Description

This command releases the JTAG chain. Use when finished performing In-System Sources and Probes transactions.

### Example

```
#List probe data of instance 0
start_insystem_source_probe -hardware_name "USB-Blaster \[USB-0\]" \
    -device_name "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"
puts "probe data of instance 0"
puts [read_probe_data -instance_index 0]
end_insystem_source_probe
```

## get_insystem_source_probe_instance_info

### Usage

```
get_insystem_source_probe_instance_info -device_name <device name> -hardware_name
<hardware name>
```

### Options

`-device_name <device name>: Name of the device programmed with the design that includes In-System Sources and Probes instances`

`-hardware_name <hardware name>: Name of the hardware that connects to the JTAG chain`

### Description

Returns a list of the available In-System Sources and Probes instances and their configuration.
{instance_index source_width probe_width instance_name}

Example:

```
{0 4 3 src1} {1 5 5 src2} {2 3 6 none}
```

### Example

```
# List information of all In-System Sources and Probes instances
puts "Information on all In-System Sources and Probes instances:"
puts "index,source_width,probe_width,name"
foreach instance \
    [get_insystem_source_probe_instance_info -hardware_name "USB-Blaster \
    \[USB-0\]" -device_name "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE \
    (0x020030DD)"] {
   puts "[lindex $instance 0],[lindex $instance 1],[lindex $instance \
       2],[lindex $instance 3]"
}
```

## read_probe_data

### Usage

```
read_probe_data -instance_index <instance_index> [-value_in_hex]
```

### Options

```
-instance_index <instance_index>: Index of the In-System Sources and Probes instance to
communicate with
```

```
-value_in_hex: Specifies that the value string is represented in hexadecimal format
```

### Description

Retrieves the current value of the probes.

A string is returned specifying the status of each probe, with the MSB on the left and LSB on the right. By default, the value is represented as a binary string.  Optionally, the option -value_in_hex makes the value a hex string.

### Example

```
#List probe data of instance 0
start_insystem_sourc_probe -hardware_name "USB-Blaster \[USB-0\]" \
    -device_name "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"
puts "probe data of instance 0"
puts [read_probe_data -instance_index 0]
end_insystem_source_probe
```

## read_source_data

### Usage

```
read_source_data -instance_index <instance_index> [-value_in_hex]
```

### Options

```
-instance_index <instance_index>: Index of the In-System Sources and Probes instance to
communicate with
```

```
-value_in_hex: Specifies that the value string is represented in hexadecimal format
```

### Description

Retrieves the current value of the sources.

A string is returned specifying the status of each source, with the MSB on the left and LSB on the right. By default, the value is represented as a binary string. Optionally, the option -value_in_hex will make the value a hex string.

### Example

```
#List source data of instance 0
start_insystem_source_probe -hardware_name "USB-Blaster \[USB-0\]" \
    -device_name "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"
puts "source data of instance 0"
puts [read_source_data -instance_index 0]
end_insystem_source_probe
```

## start_insystem_source_probe

### Usage

```
start_insystem_source_probe -device_name <device name> -hardware_name <hardware name>
```

### Options

`-device_name <device name>`: Name of the device that holds the In-System Sources and Probes instances

`-hardware_name <hardware name>`: Name of programming hardware connected to the JTAG chain

### Description

Use this command  before beginning any In-System Sources and Probes transactions

### Example

```
#List probe data of instance 0
start_insystem_source_probe -hardware_name "USB-Blaster \[USB-0\]" \
    -device_name "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"
puts "probe data of instance 0"
puts [read_probe_data -instance_index 0]
end_insystem_source_probe
```

## write_source_data

### Usage

```
write_source_data -instance_index <instance_index> -value <value> [-value_in_hex]
```

### Options

```
-instance_index <instance_index>: Index of the In-System Sources and Probes instance to
communicate with
```

```
-value <value>: Value for the source
```

```
-value_in_hex: Specify that the value string is represented in hexadecimal format
```

### Description

Sets values for the sources.

A value string is sent to the source values.  MSB is on the left and LSB is on the right. The value string is is truncated on the left (MSB) side if necessary.

By default, the values are represented as a binary string. Optionally, the option -value_in_hex makes the values hex strings.

### Example

```
#List probe data of instance 0
start_insystem_source_probe -hardware_name "USB-Blaster \[USB-0\]" \
    -device_name "@1: EP1S25/_HARDCOPY_FPGA_PROTOTYPE (0x020030DD)"
puts "write source data 10010"
write_source_data -instance_index 0 -value "10010"
puts "source data of instance 0"
puts [read_source_data -instance_index 0]
end_insystem_source_probe
```

# jtag

This package contains the set of Tcl functions for controlling the JTAG chain using Altera programming hardware.

This package is loaded by default in the following executable:

- quartus_stp

This package includes the following commands:

## close_device

### Usage

```
close_device
```

### Options

None

### Description

End the shared communication with the opened device.

### Example

```
# List all available programming hardwares, and select the USBBlaster.
# (Note: this example assumes only one USBBlaster connected.)
puts "Programming Hardwares:"
foreach hardware_name [get_hardware_names] {
    puts $hardware_name
    if { [string match "USB-Blaster*" $hardware_name] } {
    set usbblaster_name $hardware_name
    }
}
puts "\nSelect JTAG chain connected to $usbblaster_name.\n";

# List all devices on the chain, and select the first device on the
# chain.
puts "\nDevices on the JTAG chain:"
foreach device_name [get_device_names -hardware_name $usbblaster_name] {
    puts $device_name
    if { [string match "@1*" $device_name] } {
    set test_device $device_name
    }
}
puts "\nSelect device: $test_device.\n";

# Open device
open_device -hardware_name $usbblaster_name -device_name $test_device

# Retrieve device id code.
# IDCODE instruction value is 6; The ID code is 32 bits long.

# IR and DR shift should be locked together to ensure that other
# applications
# will not change the instruction register before the id code value is
# shifted
# out while the instruction register is still holding the IDCODE
# instruction.
device_lock -timeout 10000
device_ir_shift -ir_value 6 -no_captured_ir_value
puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]"
device_unlock

# Close device
close_device
```

## device_dr_shift

### Usage

```
device_dr_shift [-dr_value <data register value>] -length <data register length>
[-no_captured_dr_value] [-value_in_hex]
```

### Options

```
-dr_value <data register value>: Value of string oprand type in either default binary or
hexadecimal format to be written into the data register in the JTAG tap controller of
the open device
```

```
-length <data register length>: Length of the data register in the JTAG tap controller
in the open device
```

```
-no_captured_dr_value: Option not to return the data instruction register value. If this
is specified, this DR scan may be packed together with the subsequent IR or DR scan until
the device is unlocked or a captured value is requested
```

```
-value_in_hex: Option to specify that the value string is represented in hexadecimal
format
```

### Description

Writes the specified value into the data register of the JTAG tap controller of the open device. Returns the captured data register value. The captured value return can be disabled to improve the JTAG communication speed by packing multiple IR or DR scans together.

The value is specified using either a binary string or a hexadecimal string. The bit on the left most side is the first bit shifted in. For example, using binary string "010001", the first bit shifted into the dr register is 1; the last bit is 0. The same string can be represented in hexadecimal as "11".

The device must be locked before you can perform this operation.

### Example

```
# List all available programming hardware, and select the USB-Blaster.
# (Note: this example assumes only one USB-Blaster is connected.)
puts "Programming Hardware:"
foreach hardware_name [get_hardware_names] {
   puts $hardware_name
   if { [string match "USB-Blaster*" $hardware_name] } {
   set usbblaster_name $hardware_name
   }
}
puts "\nSelect JTAG chain connected to $usbblaster_name.\n";

# List all devices on the chain, and select the first device on the
# chain.
puts "\nDevices on the JTAG chain:"
foreach device_name [get_device_names -hardware_name $usbblaster_name] {
   puts $device_name
   if { [string match "@1*" $device_name] } {
   set test_device $device_name
   }
}
puts "\nSelect device: $test_device.\n";

# Open device
open_device -hardware_name $usbblaster_name -device_name $test_device

# Retrieve device id code.
# IDCODE instruction value is 6; The ID code is 32 bits long.
# IR and DR shift should be locked together to ensure that other
# applications
# will not change the instruction register before the id code value is
# shifted
# out while the instruction register is still holding the IDCODE
# instruction.
device_lock -timeout 10000
device_ir_shift -ir_value 6 -no_captured_ir_value
puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]"
device_unlock

# Close device
close_device
```

## device_ir_shift

### Usage

```
device_ir_shift -ir_value <instruction register value> [-no_captured_ir_value]
```

### Options

```
-ir_value <instruction register value>: Value of numeric oprand type to be written into
the instruction register in the JTAG tap controller of the open device
```

```
-no_captured_ir_value: Option to not return the captured instruction register value. If
this is specified, this IR scan may be packed together with the subsequent IR or DR scan
until the device is unlocked or a captured value is requested
```

### Description

Writes the specified value into the instruction register of the JTAG tap controller of the open device. Returns the captured instruction register value. The captured value return can be disabled to improve the JTAG communication speed by packing multiple IR or DR scans together.

The instruction register length is determined automatically by the Quartus II JTAG server.

The device must be locked first before this operation.

### Example

```
# List all available programming hardware, and select the USB-Blaster.
# (Note: this example assumes only one USB-Blaster is connected.)
puts "Programming Hardware:"
foreach hardware_name [get_hardware_names] {
    puts $hardware_name
    if { [string match "USB-Blaster*" $hardware_name] } {
    set usbblaster_name $hardware_name
    }
}
puts "\nSelect JTAG chain connected to $usbblaster_name.\n";

# List all devices on the chain, and select the first device on the
# chain.
puts "\nDevices on the JTAG chain:"
foreach device_name [get_device_names -hardware_name $usbblaster_name] {
    puts $device_name
    if { [string match "@1*" $device_name] } {
    set test_device $device_name
    }
}
puts "\nSelect device: $test_device.\n";

# Open device
open_device -hardware_name $usbblaster_name -device_name $test_device

# Retrieve device id code.
# IDCODE instruction value is 6; The ID code is 32 bits long.

# IR and DR shift should be locked together to ensure that other
# applications
# will not change the instruction register before the id code value is
# shifted
# out while the instruction register is still holding the IDCODE
# instruction.
device_lock -timeout 10000
device_ir_shift -ir_value 6 -no_captured_ir_value
```

```
puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]"
device_unlock

# Close device
close_device
```

## device_lock

### Usage

```
device_lock -timeout <timeout>
```

### Options

```
-timeout <timeout>: The amount of time in millisecond to wait for the access to the
device.
```

### Description

Obtain an exclusive JTAG communication to the device for the subsequent IR and DR shift operations. The device must be locked before any instruction and/or data register shift operation.

This should be used as little time as possible as it denies the access of other applications to this chain. The command, unlock, should be called as soon as possible to allow other applications to access the device.

### Example

```
# List all available programming hardwares, and select the USBBlaster.
# (Note: this example assumes only one USBBlaster connected.)
puts "Programming Hardwares:"
foreach hardware_name [get_hardware_names] {
   puts $hardware_name
   if { [string match "USB-Blaster*" $hardware_name] } {
   set usbblaster_name $hardware_name
   }
}
puts "\nSelect JTAG chain connected to $usbblaster_name.\n";

# List all devices on the chain, and select the first device on the
# chain.
puts "\nDevices on the JTAG chain:"
foreach device_name [get_device_names -hardware_name $usbblaster_name] {
   puts $device_name
   if { [string match "@1*" $device_name] } {
   set test_device $device_name
   }
}
puts "\nSelect device: $test_device.\n";

# Open device
open_device -hardware_name $usbblaster_name -device_name $test_device

# Retrieve device id code.
# IDCODE instruction value is 6; The ID code is 32 bits long.

# IR and DR shift should be locked together to ensure that other
# applications
# will not change the instruction register before the id code value is
# shifted
# out while the instruction register is still holding the IDCODE
# instruction.
device_lock -timeout 10000
device_ir_shift -ir_value 6 -no_captured_ir_value
puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]"
device_unlock

# Close device
close_device
```

## device_run_test_idle

### Usage

```
device_run_test_idle [-num_clocks <state cycle count>]
```

### Options

```
-num_clocks <state cycle count>: The number of times the Run_Test_Idle state is cycled
through.  If not specified, this value is 1
```

### Description

Drive the JTAG controller into the Run_Test_Idle state for a number cycles specified with the -num_clocks option.

The device must be locked before you can perform this operation.

### Example

```
# List all available programming hardware, and select the USB-Blaster.
# (Note: this example assumes only one USB-Blaster is connected.)
puts "Programming Hardware:"
foreach hardware_name [get_hardware_names] {
    puts $hardware_name
    if { [string match "USB-Blaster*" $hardware_name] } {
    set usbblaster_name $hardware_name
    }
}
puts "\nSelect JTAG chain connected to $usbblaster_name.\n";

# List all devices on the chain, and select the first device on the
# chain.
puts "\nDevices on the JTAG chain:"
foreach device_name [get_device_names -hardware_name $usbblaster_name] {
    puts $device_name
    if { [string match "@1*" $device_name] } {
    set test_device $device_name
    }
}
puts "\nSelect device: $test_device.\n";

# Open device
open_device -hardware_name $usbblaster_name -device_name $test_device

# Retrieve device id code.
# IDCODE instruction value is 6; The ID code is 32 bits long.

# IR and DR shift should be locked together to ensure that other
# applications
# will not change the instruction register before the id code value is
# shifted
# out while the instruction register is still holding the IDCODE
# instruction.
device_lock -timeout 10000
device_ir_shift -ir_value 6 -no_captured_ir_value
puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]"

# Goto the Run_Test_Idle state and stay there for 8 cycles.
device_run_test_idle -num_clocks 8
device_unlock

# Close device
close_device
```

## device_unlock

### Usage

```
device_unlock
```

### Options

None

### Description

Release the exclusive JTAG communication lock on the device.

### Example

```
# List all available programming hardwares, and select the USBBlaster.
# (Note: this example assumes only one USBBlaster connected.)
puts "Programming Hardwares:"
foreach hardware_name [get_hardware_names] {
   puts $hardware_name
   if { [string match "USB-Blaster*" $hardware_name] } {
   set usbblaster_name $hardware_name
   }
}
puts "\nSelect JTAG chain connected to $usbblaster_name.\n";

# List all devices on the chain, and select the first device on the
# chain.
puts "\nDevices on the JTAG chain:"
foreach device_name [get_device_names -hardware_name $usbblaster_name] {
   puts $device_name
   if { [string match "@1*" $device_name] } {
   set test_device $device_name
   }
}
puts "\nSelect device: $test_device.\n";

# Open device
open_device -hardware_name $usbblaster_name -device_name $test_device

# Retrieve device id code.
# IDCODE instruction value is 6; The ID code is 32 bits long.

# IR and DR shift should be locked together to ensure that other
# applications
# will not change the instruction register before the id code value is
# shifted
# out while the instruction register is still holding the IDCODE
# instruction.
device_lock -timeout 10000
device_ir_shift -ir_value 6 -no_captured_ir_value
puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]"
device_unlock

# Close device
close_device
```

## device_virtual_dr_shift

### Usage

```
device_virtual_dr_shift [-dr_value <data register value>] -instance_index <instance
index> -length <data register length> [-no_captured_dr_value]
[-show_equivalent_device_ir_dr_shift] [-value_in_hex]
```

### Options

`-dr_value <data register value>`: Value of string oprand type in either default binary or hexadecimal format to be written into the data register in this instance

`-instance_index <instance index>`: The index of the virtual JTAG instance

`-length <data register length>`: Length of the data register in this instance

`-no_captured_dr_value`: Option to not return the data instruction register value

`-show_equivalent_device_ir_dr_shift`: Option to show equivalent device ir dr shifts performed by this command

`-value_in_hex`: Option to specify that the value string is represented in hexadecimal format

### Description

Writes the specified value into the data register of the JTAG tap controller of the open device. Returns the captured data register value. The captured value return can be disabled to improve the JTAG communication speed by packing multiple IR or DR scans together.

The value is specified using either a binary string or a hexadecimal string. The bit on the left most side is the first bit shifted in. For example, using the binary string "010001", the first bit shifted into the dr register is 1; the last bit is 0. The same string can be represented in hexadecimal as "11".

The device must be locked first, and the target instance must be activated using the device_virtual_ir_shift command before this operation. Moreover, the device should be locked before the virtual IR shift operation to prevent another application from activating another instance.

### Example

```
# List all available programming hardware, and select the USB-Blaster.
# (Note: this example assumes only one USB-Blaster is connected.)
puts "Programming Hardware:"
foreach hardware_name [get_hardware_names] {
   puts $hardware_name
   if { [string match "USB-Blaster*" $hardware_name] } {
   set usbblaster_name $hardware_name
   }
}
puts "\nSelect JTAG chain connected to $usbblaster_name.\n";

# List all devices on the chain, and select the first device on the
# chain.
puts "\nDevices on the JTAG chain:"
foreach device_name [get_device_names -hardware_name $usbblaster_name] {
   puts $device_name
   if { [string match "@1*" $device_name] } {
   set test_device $device_name
   }
}
puts "\nSelect device: $test_device.\n";

# Open device
open_device -hardware_name $usbblaster_name -device_name $test_device
```

```
# The follow virtual JTAG IR and DR shift sequence engage with
# the example virtual JTAG instance.
#
# Two instructions: SAMPLE (1) FEED (2)
# SAMPLE instruction samples a 8-bit bus; the captured value shows the
# number of sample performed.
# FEED instruction supplies a 8-bit value to the logic connected to this
# instance.
# Both data registers corresponding to the IR are 8 bit wide.

# Send SAMPLE instruction to IR, read captured IR for the sampling
# number.
# Capture the DR register for the current sampled value.
device_lock -timeout 10000
puts "Current LED Value (sample #[device_virtual_ir_shift -instance_index \
    0 -ir_value 1]): \
    [device_virtual_dr_shift -instance_index 0  -length 8 -value_in_hex]"
device_unlock

# Send FEED instruction to IR, read a two-digit hex string from the
# console,
# then send the new value to the DR register.
puts "\nType in 2 digits in hexadecimal to update the LED:"
gets stdin update_value

device_lock -timeout 10000
device_virtual_ir_shift -instance_index 0 -ir_value 2 \
    -no_captured_ir_value
device_virtual_dr_shift -instance_index 0  -length 8 -dr_value \
    $update_value -value_in_hex -no_captured_dr_value
device_unlock

# Close device
close_device
```

## device_virtual_ir_shift

### Usage

```
device_virtual_ir_shift -instance_index <instance index> -ir_value <instruction register
value> [-no_captured_ir_value] [-show_equivalent_device_ir_dr_shift]
```

### Options

```
-instance_index <instance index>: The index of the virtual JTAG instance
```

```
-ir_value <instruction register value>: Value of numeric oprand type to be written into
the instruction register in this instance
```

```
-no_captured_ir_value: Option to not return the captured instruction register value.  If
this is specified, this IR scan may be packed together with the subsequent IR or DR scan
until the device is unlocked or a captured value is requested
```

```
-show_equivalent_device_ir_dr_shift: Option to show equivalent device ir and dr shifts
performed by this command
```

### Description

Writes the specified value into the instruction register of the specified virtual JTAG instance in the open device. Returns the captured instruction register value. You can disable the captured value return to improve the JTAG communication speed by packing multiple IR or DR scans together.

The command also activates the target instance such that the consequent virtual DR shift operations are applied to this instance before the device is unlocked.  Before any virtual DR shift operation, this command must be executed first to activate the instance.

The device must be locked first before this operation.

### Example

```
# List all available programming hardwares, and select the USBBlaster.
# (Note: this example assumes only one USBBlaster connected.)
puts "Programming Hardwares:"
foreach hardware_name [get_hardware_names] {
   puts $hardware_name
   if { [string match "USB-Blaster*" $hardware_name] } {
   set usbblaster_name $hardware_name
   }
}
puts "\nSelect JTAG chain connected to $usbblaster_name.\n";

# List all devices on the chain, and select the first device on the
# chain.
puts "\nDevices on the JTAG chain:"
foreach device_name [get_device_names -hardware_name $usbblaster_name] {
   puts $device_name
   if { [string match "@1*" $device_name] } {
   set test_device $device_name
   }
}
puts "\nSelect device: $test_device.\n";

# Open device
open_device -hardware_name $usbblaster_name -device_name $test_device

# The follow virtual JTAG IR and DR shift sequence engage with
# the example virtual JTAG instance.
#
# Two instructions: SAMPLE (1) FEED (2)
# SAMPLE instruction samples a 8-bit bus; the captured value shows the
# number of sample performed.
# FEED instruction supplies a 8-bit value to the logic connected to this
# instance.
# Both data registers corresponding to the IR are 8 bit wide.

# Send SAMPLE instruction to IR, read captured IR for the sampling
# number.
# Capture the DR register for the current sampled value.
device_lock -timeout 10000
puts "Current LED Value (sample #[device_virtual_ir_shift -instance_index \
    0 -ir_value 1]): \
    [device_virtual_dr_shift -instance_index 0  -length 8 -value_in_hex]"
device_unlock

# Send FEED instruction to IR, read a two-digit hex string from the
# console,
# then send the new value to the DR register.
puts "\nType in 2 digits in hexadecimal to update the LED:"
gets stdin update_value
device_lock -timeout 10000
device_virtual_ir_shift -instance_index 0 -ir_value 2 \
    -no_captured_ir_value
device_virtual_dr_shift -instance_index 0  -length 8 -dr_value \
    $update_value -value_in_hex -no_captured_dr_value
device_unlock

# Close device
close_device
```

## get_device_names

### Usage

```
get_device_names -hardware_name <hardware name>
```

### Options

```
-hardware_name <hardware name>: The name of the programming hardware that connects to
the JTAG chain.  The name can be obtained from command: get_hardware_names.
```

### Description

Retrieves a list of names of the devices on the JTAG chain to which the specified programming hardware is connected.

The name of the device is in the following format: <number on circuit board>: <JTAG ID code>: <device name>. For example, in the device name @1: 0x082000DD: EP20K200C, @1 indicates that it is the first device on the circuit board, 0x082000DD is the JTAG ID code for the device, and EP20K200C is the device name.

### Example

```
# List all available programming hardwares, and select the USBBlaster.
# (Note: this example assumes only one USBBlaster connected.)
puts "Programming Hardwares:"
foreach hardware_name [get_hardware_names] {
    puts $hardware_name
    if { [string match "USB-Blaster*" $hardware_name] } {
    set usbblaster_name $hardware_name
    }
}
puts "\nSelect JTAG chain connected to $usbblaster_name.\n";

# List all devices on the chain, and select the first device on the
# chain.
puts "\nDevices on the JTAG chain:"
foreach device_name [get_device_names -hardware_name $usbblaster_name] {
    puts $device_name
    if { [string match "@1*" $device_name] } {
    set test_device $device_name
    }
}
puts "\nSelect device: $test_device.\n";

# Open device
open_device -hardware_name $usbblaster_name -device_name $test_device

# Retrieve device id code.
# IDCODE instruction value is 6; The ID code is 32 bits long.

# IR and DR shift should be locked together to ensure that other
# applications
# will not change the instruction register before the id code value is
# shifted
# out while the instruction register is still holding the IDCODE
# instruction.
device_lock -timeout 10000
device_ir_shift -ir_value 6 -no_captured_ir_value
puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]"
device_unlock

# Close device
close_device
```

## get_hardware_names

### Usage

```
get_hardware_names
```

### Options

None

### Description

Retrieves a list of the names of the programming hardware attached to and configured for the JTAG server.

The hardware name is in the following format: <hardware type> [<port>]. For example, in the hardware name USB-Blaster [USB-0], USB-Blaster is the name of the programming hardware, and USB-0 is the name of the port to which the hardware is connected.

### Example

```
# List all available programming hardware, and select the USB-Blaster.
# (Note: this example assumes only one USB-Blaster is connected.)
puts "Programming Hardware:"
foreach hardware_name [get_hardware_names] {
    puts $hardware_name
    if { [string match "USB-Blaster*" $hardware_name] } {
    set usbblaster_name $hardware_name
    }
}
puts "\nSelect JTAG chain connected to $usbblaster_name.\n";

# List all devices on the chain, and select the first device on the
# chain.
puts "\nDevices on the JTAG chain:"
foreach device_name [get_device_names -hardware_name $usbblaster_name] {
    puts $device_name
    if { [string match "@1*" $device_name] } {
    set test_device $device_name
    }
}
puts "\nSelect device: $test_device.\n";

# Open device
open_device -hardware_name $usbblaster_name -device_name $test_device

# Retrieve device id code.
# IDCODE instruction value is 6; The ID code is 32 bits long.

# IR and DR shift should be locked together to ensure that other
# applications
# will not change the instruction register before the id code value is
# shifted
# out while the instruction register is still holding the IDCODE
# instruction.
device_lock -timeout 10000
device_ir_shift -ir_value 6 -no_captured_ir_value
puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]"
device_unlock

# Close device
close_device
```

## open_device

### Usage

```
open_device -device_name <device name> -hardware_name <hardware name>
```

### Options

```
-device_name <device name>: The name of the device on the JTAG chain.  The name can be
obtained from command: get_device_names
```

```
-hardware_name <hardware name>: The name of the programming hardware that connects to
the JTAG chain.  The name can be obtained from command: get_hardware_names
```

### Description

Initiate a shared JTAG communication with the device. The command, close_device, is called to end the communication with the device.

Only one device can be opened per process. Multiple devices can be opened in multiple processes.

### Example

```
# List all available programming hardwares, and select the USBBlaster.
# (Note: this example assumes only one USBBlaster connected.)
puts "Programming Hardwares:"
foreach hardware_name [get_hardware_names] {
   puts $hardware_name
   if { [string match "USB-Blaster*" $hardware_name] } {
   set usbblaster_name $hardware_name
   }
}
puts "\nSelect JTAG chain connected to $usbblaster_name.\n";

# List all devices on the chain, and select the first device on the
# chain.
puts "\nDevices on the JTAG chain:"
foreach device_name [get_device_names -hardware_name $usbblaster_name] {
   puts $device_name
   if { [string match "@1*" $device_name] } {
   set test_device $device_name
   }
}
puts "\nSelect device: $test_device.\n";

# Open device
open_device -hardware_name $usbblaster_name -device_name $test_device

# Retrieve device id code.
# IDCODE instruction value is 6; The ID code is 32 bits long.

# IR and DR shift should be locked together to ensure that other
# applications
# will not change the instruction register before the id code value is
# shifted
# out while the instruction register is still holding the IDCODE
# instruction.
device_lock -timeout 10000
device_ir_shift -ir_value 6 -no_captured_ir_value
puts "IDCODE: 0x[device_dr_shift -length 32 -value_in_hex]"
device_unlock

# Close device
close_device
```

## logic_analyzer_interface

This package contains the set of Tcl functions for querying and modifying the Logic Analyzer Interface output pin state in the Altera device.

This package is loaded by default in the following executable:

- quartus_stp

This package includes the following commands:

## begin_logic_analyzer_interface_control

### Usage

```
begin_logic_analyzer_interface_control -device_name <device name> -file_path <file path>
-hardware_name <hardware name>
```

### Options

```
-device_name <device name>: Name of the device to be controlled
```

```
-file_path <file path>: Path of the Logic Analyzer Interface (.lai) file
```

```
-hardware_name <hardware name>: Name of the hardware that connects to the JTAG chain
```

### Description

Starts the Logic Analyzer Interface control sequence to query the Logic Analyzer Interface output pin state and change output pins state. The control sequence should be terminated with end_logic_analyzer_interface_control.

The hardware and device name can be obtained by using get_hardware_names and get_device_names respectively from the jtag package.

### Example

```
# Start a new control sequence.
begin_logic_analyzer_interface_control -hardware_name "USB-Blaster \
    \[USB-0\]" -device_name "@1: EP1C20 (0x020840DD)" -file_path \
    "lai_demo.lai"

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# Change input bank source to the output pins
change_bank_to_output_pin -instance_name "auto_lai_0" -bank_name "Bank 1"

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# Change input bank source to the output pins
change_bank_to_output_pin -instance_name "auto_lai_0" -bank_index 0

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# Tristate the output pins
tristate_output_pin -instance_name "auto_lai_0"

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# End the control sequence.
end_logic_analyzer_interface_control
```

## change_bank_to_output_pin

### Usage

```
change_bank_to_output_pin [-bank_index <bank index>] [-bank_name <bank name>]
-instance_name <instance name>
```

### Options

`-bank_index <bank index>: Index of the bank on the mux to be used as the source of the output pins`

`-bank_name <bank name>: Name of the bank to be used as the source of the output pins`

`-instance_name <instance name>: Name of the Logic Analyzer Interface instance to change`

### Description

Change the Logic Analyzer Interface output pin's source on the specified instance to use the specified bank.

### Example

```
# Start a new control sequence.
begin_logic_analyzer_interface_control -hardware_name "USB-Blaster \
    \[USB-0\]" -device_name "@1: EP1C20 (0x020840DD)" -file_path \
    "lai_demo.lai"

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# Change input bank source to the output pins
change_bank_to_output_pin -instance_name "auto_lai_0" -bank_name "Bank 1"

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# Change input bank source to the output pins
change_bank_to_output_pin -instance_name "auto_lai_0" -bank_index 0

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# Tristate the output pins
tristate_output_pin -instance_name "auto_lai_0"

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# End the control sequence.
end_logic_analyzer_interface_control
```

## end_logic_analyzer_interface_control

### Usage

```
end_logic_analyzer_interface_control
```

### Options

None

### Description

Terminate the Logic Analyzer Interface control sequence.

### Example

```
# Start a new control sequence.
begin_logic_analyzer_interface_control -hardware_name "USB-Blaster \
    \[USB-0\]" -device_name "@1: EP1C20 (0x020840DD)" -file_path \
    "lai_demo.lai"

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# Change input bank source to the output pins
change_bank_to_output_pin -instance_name "auto_lai_0" -bank_name "Bank 1"

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# Change input bank source to the output pins
change_bank_to_output_pin -instance_name "auto_lai_0" -bank_index 0

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# Tristate the output pins
tristate_output_pin -instance_name "auto_lai_0"

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# End the control sequence.
end_logic_analyzer_interface_control
```

## get_current_state_of_output_pin

### Usage

```
get_current_state_of_output_pin -instance_name <instance name>
```

### Options

```
-instance_name <instance name>: Name of the Logic Analyzer Interface instance to change
```

### Description

Query the device to get the current state of the output pins of the specified instance.

The result is either the bank name or "tristated".

### Example

```
# Start a new control sequence.
begin_logic_analyzer_interface_control -hardware_name "USB-Blaster \
    \[USB-0\]" -device_name "@1: EP1C20 (0x020840DD)" -file_path \
    "lai_demo.lai"

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# Change input bank source to the output pins
change_bank_to_output_pin -instance_name "auto_lai_0" -bank_name "Bank 1"

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# Change input bank source to the output pins
change_bank_to_output_pin -instance_name "auto_lai_0" -bank_index 0

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# Tristate the output pins
tristate_output_pin -instance_name "auto_lai_0"

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# End the control sequence.
end_logic_analyzer_interface_control
```

## tristate_output_pin

### Usage

```
tristate_output_pin -instance_name <instance name>
```

### Options

```
-instance_name <instance name>: Name of the Logic Analyzer Interface instance to change
```

### Description

Tristate the output pins of the specified Logic Analyzer Interface instance.

### Example

```
# Start a new control sequence.
begin_logic_analyzer_interface_control -hardware_name "USB-Blaster \
    \[USB-0\]" -device_name "@1: EP1C20 (0x020840DD)" -file_path \
    "lai_demo.lai"

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# Change input bank source to the output pins
change_bank_to_output_pin -instance_name "auto_lai_0" -bank_name "Bank 1"

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# Change input bank source to the output pins
change_bank_to_output_pin -instance_name "auto_lai_0" -bank_index 0

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# Tristate the output pins
tristate_output_pin -instance_name "auto_lai_0"

# Query the output pin state.
puts "Current output pin state of instance auto_lai_0:"
puts [get_current_state_of_output_pin -instance_name "auto_lai_0"]

# End the control sequence.
end_logic_analyzer_interface_control
```

## misc

This package contains a set of Tcl functions for performing miscellaneous tasks.

This package is loaded by default in the following executables:

- quartus
- quartus_cdb
- quartus_drc
- quartus_eda
- quartus_sh
- quartus_si
- quartus_sim
- quartus_sta
- quartus_stp
- quartus_tan

This package includes the following commands:

## checksum

### Usage

```
checksum [-algorithm <crc32|adler32>] <input_file>
```

### Options

```
-algorithm <crc32|adler32>: Option to specify the checksum algorithm. Uses the CRC-32
algorithm by default.
```

```
<input_file>: Option to specify the input file
```

### Description

Returns the checksum value in hexadecimal format based on the specified algorithm which defaults to crc32.

### Example

```
set file "one_wire.sof"
   # Use CRC-32
puts "$file -> [checksum $file]"
puts "$file -> [checksum $file -algorithm crc32]"
   # Use ADLER-32
puts "$file -> [checksum $file -algorithm adler32]"
```

## disable_natural_bus_naming

### Usage

```
disable_natural_bus_naming
```

### Options

None

### Description

Disables natural bus naming.

You may choose to disable natural bus naming to string match patterns such as "in\[024\]". In this example, you are string matching the names "in0", "in2", and "in4".

Note that if you disable natural bus naming, then square brackets must be escaped twice (\\\[ or \\\]) so that the strings are interpreted as bus names during a string match, such as:

```
set bus_name "address\[0\]" string match [escape_brackets $bus_name]
$bus_name
```

The "escape_brackets" command escapes "address\[0\]" into "address\\\[0\\\]".

To enable natural bus naming again, type "enable_natural_bus_naming".

For more information about natural bus naming, type "enable_natural_bus_naming -h".

### Example

```
# Disables natural bus naming
disable_natural_bus_naming
```

## enable_natural_bus_naming

### Usage

```
enable_natural_bus_naming
```

### Options

None

### Description

Enables natural bus naming so that square brackets for bus names do not have to be escaped to prevent Tcl from interpreting them as sub-commands.

Bus names have the following format:

```
<bus name>[<bus index>] or <bus name>[*]
```

The <bus name> portion is a string of alphanumeric characters. The <bus index> portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the <bus index> is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names.

Many Quartus II Tcl commands allow bus names in their arguments, such as:

```
set_location_assignment -to address[10] Pin_M20
```

If natural bus naming is enabled, you can just use address[10] instead of having to excape the square brackets into address\[10\].

There are also Quartus II Tcl commands that take Tcl string match patterns in their arguments, such as:

```
get_all_instance_assignments -name location -to address[10]
```

Since Tcl string matching take string patterns containing special characters from the set "*?\[]" as values, address[10] would be interpreted incorrectly. By enabling natural bus naming, these Tcl commands will automatically detect address[10] as a bus name so that you don't have to doubly escape the brackets into address\\\[10\\\].

To disable natural bus naming, type "disable_natural_bus_naming".

For more information on the effects of disabling natural bus naming, type "disable_natural_bus_naming -h".

### Example

```
# Enables natural bus naming
enable_natural_bus_naming
```

## escape_brackets

### Usage

```
escape_brackets <str>
```

### Options

```
<str>: String to escape
```

### Description

Escapes square brackets in bus name patterns for use in string matching. Also escapes the escape character; for example, the string "\" is escaped into "\\".

Note that natural bus naming is supported by default. This means that bus names, not bus name patterns, are automatically detected and do not need to be escaped by using this command. Bus names have the following format:

```
<bus name>[<bus index>] or <bus name>[*]
```

The <bus name> portion is a string of alphanumeric characters. The <bus index> portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the <bus index> is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names.

All other uses of square brackets must be escaped if you do not intend to use the brackets as part of a string pattern in a string match. For example, the bus name pattern "a\[0-2\]" must be escaped using this "escape_brackets" command since the "0-2" does not satisfy the <bus index> requirement to be a bus name.

Many Quartus II Tcl commands allow string matching in option arguments. A common error is using bus name patterns in these arguments, such as:

```
address\[0-2\]
```

Square brackets for bus name patterns must already be preceded by an escape character (\[ or \]) to prevent Tcl from interpreting them as sub-commands. String matching, however, also uses square brackets to match any character between the brackets. The previous example, when used as a string match pattern, searches for the string patterns address0, address1, and address2. It does not search for address[0], address[1], and address[2].

Therefore, for arguments that support string matching, square brackets must be escaped twice (\\\[ or \\\]) so that the strings are interpreted as bus name patterns. For example, to search for address[0], address[0], and address[2], type the following string match pattern:

```
address\\\[\[0-2\]\\\]
or, equivalently,
"address[escape_brackets \[]\[0-2\][escape_brackets \]]"
```

Quartus II Tcl commands do not convert bus name patterns automatically, since they cannot determine if the string is intended as a bus name pattern or a regular string match pattern. Therefore, "escape_brackets" is provided for your convenience.

You may choose to disable natural bus naming in order to string match patterns such as "in\[024\]". In this example, you are string matching the names "in0", "in2", and "in4".

To disable natural bus naming, type "disable_natural_bus_naming".

Note that if you disable natural bus naming, then square brackets must be escaped twice (\\\[ or \\\]) so that the strings are interpreted as bus names during a string match, such as:

```
set bus_name "address\[0\]"
string match [escape_brackets $bus_name] $bus_name
```

The "escape_brackets" command escapes "address\[0\]" into "address\\\[0\\\]".

To enable natural bus naming again, type "enable_natural_bus_naming".

For more information about natural bus naming, type "enable_natural_bus_naming -h".

## Example

```
# Get all location assignments for bus address[]
set address_names address[*]
set address_locations [get_all_instance_assignments -to $address_names] \
    -name LOCATION

# Get location assignment for bus address[0]
set address_names address[0]
set address_locations [get_all_instance_assignments -to $address_names] \
    -name LOCATION

# Get location assignments for bus address[0],
# address[1], and address[2]
set address_names address[0-2]
set address_locations [get_all_instance_assignments -to \
    [escape_brackets $address_names]] -name LOCATION
```

## foreach_in_collection

### Usage

```
foreach_in_collection <variable_name> <collection> <body>
```

### Options

```
<variable_name>: Variable name
<collection>: Collection
<body>: Body
```

### Description

Accesses each element of a collection.

Some Tcl commands return a collection. The following table shows examples of commands that return a collection:

| Tcl package | Tcl commands (returning a collection) |
|---|---|
| ::quartus::project | get_all_quartus_defaults<br>get_all_global_assignments<br>get_all_instance_assignments<br>get_all_parameters<br>get_names<br>assignment_group (only for the "-get_members" and "-get_exceptions" options) |
| ::quartus::chip_editor | get_nodes<br>get_iports<br>get_oports |
| ::quartus::advanced_timing | get_timing_nodes<br>get_timing_edges |

The command is used in the following format:

```
foreach_in_collection <variable name> <collection> {
      # This is the body of "foreach_in_collection"
      ...
}
```

Unlike a Tcl list, a collection is a container specific to the Quartus II software, whose elements can be accessed by using the "foreach_in_collection" command.

### Example

```
## Get a collection of global assignments
set collection_of_global_assignments [get_all_global_assignments -name *]
## Display the collection string representation
puts $collection_of_global_assignments
## Iterate through the collection and display
## the information for each global assignment
foreach_in_collection global $collection_of_global_assignments {

   set sect_id [lindex $global 0]
   set name [lindex $global 1]
   set value [lindex $global 2]

   ## Now, display the content of the global assignment
   puts "Section ID ($sect_id)"
   puts "Assignment Name ($name)"
   puts "Assignment Value ($value)"
}

## Get a collection of instance assignments
set collection_of_instance_assignments \
    [get_all_instance_assignments -name *]
## Display the collection string representation
puts $collection_of_instance_assignments
## Iterate through the collection and display
## the information for each instance assignment
foreach_in_collection instance $collection_of_instance_assignments {

   set sect_id [lindex $instance 0]
   set src [lindex $instance 1]
   set dest [lindex $instance 2]
   set name [lindex $instance 3]
   set value [lindex $instance 4]

   ## Now, display the content of the instance assignment
   puts "Section ID ($sect_id)"
   puts "Source ($src)"
   puts "Destination ($dest)"
   puts "Assignment Name ($name)"
   puts "Assignment Value ($value)"
}

## Get a collection of parameters
set collection_of_parameters [get_all_parameters -name *]
## Display the collection string representation
puts $collection_of_parameters
## Iterate through the collection and display
## the information for each parameter
foreach_in_collection parameter $collection_of_parameters {

   set dest [lindex $parameter 0]
   set name [lindex $parameter 1]
   set value [lindex $parameter 2]

   ## Now, display the content of the parameter
   puts "Destination ($dest)"
   puts "Parameter Name ($name)"
   puts "Parameter Value ($value)"
}

## Get a collection of all node name ids from a successful
## compilation
set collection_of_name_ids [get_names -filter *]
## Display the collection string representation
puts $collection_of_name_ids
## Iterate through the collection and display
## the information for each name id
foreach_in_collection name_id $collection_of_name_ids {

   set parent_name_id [get_name_info -info parent_name_id $name_id]
   set base_name [get_name_info -info base_name $name_id]
   set entity_name [get_name_info -info entity_name $name_id]
   set instance_name [get_name_info -info instance_name $name_id]
   set full_path [get_name_info -info full_path $name_id]
   set short_full_path [get_name_info -info short_full_path $name_id]
   set node_type [get_name_info -info node_type $name_id]
   set creator [get_name_info -info creator $name_id]
   set signaltapii [get_name_info -info signaltapii $name_id]
```

```
    set file_location [get_name_info -info file_location $name_id]

    ## Now, display information about the name
    puts "Parent Name Id ($parent_name_id)"
    puts "Base Name ($base_name)"
    puts "Entity Name ($entity_name)"
    puts "Instance Name ($instance_name)"
    puts "Full Path ($full_path)"
    puts "Short Full Path ($short_full_path)"
    puts "Node Type ($node_type)"
    puts "Creator ($creator)"
    puts "Signaltapii ($signaltapii)"
    puts "File location ($file_location)"
}

# Display the members of a particular assignment group named "tg1"
foreach_in_collection member [assignment_group "tg1" -get_members] {

    # Print the name of the member
    puts $member
}

# Display the exception to a particular assignment group named "tg1"
foreach_in_collection exception [assignment_group "tg1" -get_exceptions] \
     {

    # Print the name of the exception
    puts $exception
}
```

## get_collection_size

### Usage

```
get_collection_size <collection>
```

### Options

```
<collection>: Collection
```

### Description

Returns the size of a collection.

Unlike a Tcl list, a collection is a container specific to the Quartus II software, whose elements can be accessed by using the "foreach_in_collection" command.

### Example

```
## Displays the number of global assignments
project_open chiptrip
set num_global_asgns [get_collection_size \
    [get_all_global_assignments -name {*}]]
puts $num_global_asgns
project_close
```

## get_environment_info

### Usage

```
get_environment_info [-num_logical_processors] [-num_physical_processors]
[-operating_system]
```

### Options

```
-num_logical_processors: Option to return the number of available logical processors
(cores including hyper-threading)
```

```
-num_physical_processors: Option to return the number of available physical processors
(sockets)
```

```
-operating_system: Option to return the operating system name
```

### Description

Returns information about the system environment depending on the options specified.

### Example

```
# Get the number of physical processors available on my computer.
get_environment_info -num_physical_processors
```

## init_tk

### Usage

```
init_tk
```

### Options

None

### Description

Initializes a Tk window. If you are using Tk functionality in Tcl, you must run this command first before running any Tcl scripts.

### Example

```
# Initialize the Tk library
init_tk

# Create a top level and add a title
toplevel .top
wm title .top "Hello World"

# Add widgets
button .top.hello -text Hello -command {puts stdout "Hello, World!"}
pack .top.hello -padx 20 -pady 10

# Without "tkwait", the script finishes at this point and the
# window is destroyed. The "tkwait" command prevents the
# script from finishing until the you close the window.
tkwait window .top
```

## load

### Usage

```
load <load_args>
```

### Options

```
<load_args>: Arguments to load
```

### Description

Loads machine code and initializes new commands.

This command works exactly like Tcl's native "load" command.

### Example

Refer to documentation for Tcl's native "load" command at the Tcl/Tk web site at www.tcl.tk.

## load_package

### Usage

```
load_package [-version <version number>] <package name>
```

### Options

```
-version <version number>: Option to specify the Quartus II Tcl package version to load
<package name>: Name of Quartus II Tcl package to load
```

### Description

Loads the specified Quartus II Tcl package with the specified version number. If you do not specify the "-version" option, the latest version is loaded by default.

The Quartus II Tcl package names have the "::quartus::" prefix, such as "::quartus::project". For convenience, you can omit the "::quartus::" prefix when you use the <package name> argument.

This command is similar to the "package require" command.  The advanatage of using "load_package" is that you can alternate freely between different versions of the same package.

For example, if you loaded version 2.0, and now want to load version 1.0, you can type:

```
"load_package -version 1.0 <package name>".
```

### Example

```
# Load version 1.0 of the ::quartus::project package
load_package project -version 1.0

# Load version 2.0 of the ::quartus::project package
load_package project -version 2.0
```

## post_message

### Usage

```
post_message [-file <file name>] [-line <file line number>] [-type
<info|extra_info|warning|critical_warning|error>] <string>
```

### Options

`-file <file name>: File name associated with the message`

`-line <file line number>: Line number used when locating the specified <file name> through the Quartus II software`

`-type <info|extra_info|warning|critical_warning|error>: Type of message to display`

`<string>: Message to be displayed`

### Description

Displays the message of the specified type.

The message type can be "info", "extra_info", "warning", "critical_warning", or "error". If you do not use the "-type" option, the default message type is "info".

### Example

```
# Display an error message
post_message -type error "Can't open file test.tcl"

# Display an info message
post_message "Generated output file: test.out"
# OR
post_message -type info "Generated output file: test.out"

# Display a warning message
post_message -type warning "Defaulting fmax to 155.55mhz"

# Display a extra info message
post_message -type extra_info "Input file test.in had 100 lines"

# Display a critical warning message
post_message -type critical_warning "Invalid fmax was specified - \
    defaulting to 155.55mhz"
```

## qexec

### Usage

```
qexec <command>
```

### Options

```
<command>: Command
```

### Description

Runs the specified command.

Usage for this command is as follows:

```
    qexec "<command>"
```

The Quartus II Tcl command "qexec" is the equivalent of the Tcl command "exec".

### Example

```
qexec ls
```

## qexit

### Usage

```
qexit [-error] [-success]
```

### Options

```
-error: Option to exit with an equivalent Quartus II error code
-success: Option to exit with an equivalent Quartus II success code
```

### Description

Exits the Quartus II software.

The Quartus II Tcl command "qexit" is equivalent to the Tcl command "exit".

When used with a particular option, this command exits the Quartus II software with the corresponding Quartus II exit code. For example, typing "qexit -success" is equivalent to typing "exit 0". When the "-success" option is specified, the corresponding Quartus II exit code is "0".

If no option is specified, the default exit code is the same as for the "-success" option.

### Example

1) To exit the Quartus II software with an equivalent Quartus II success code, type:

```
    qexit -success
```

2) To exit the Quartus II software with an equivalent Quartus II error code, type:

```
    qexit -error
```

## stopwatch

### Usage

```
stopwatch [-lap_time] [-number_format] [-reset] [-start]
```

### Options

`-lap_time: Option to get the lap time`

`-number_format: Option to show the lap time in seconds without appending the "s", i.e. seconds, character`

`-reset: Option to reset the stopwatch`

`-start: Option to start the stopwatch`

### Description

Provides a stopwatch interface.

### Example

```
# Begin the stopwatch
stopwatch -start
exec sleep 1
   # Get the lap time
puts [stopwatch -lap_time]
exec sleep 1
   # Get the lap time
puts [stopwatch -lap_time]
   # Reset the stopwatch
stopwatch -reset
```

## project

This package contains the set of Tcl functions for making project-wide assignments.

In versions before 4.0 of this package, the full path of the source file assignment was returned when you accessed the assignment through the "get_global_assignment" or "get_all_global_assignments" command.

In version 4.0 of this package, the actual value of the source file assignment stored in the Quartus II Settings File (.qsf) is returned. To get the resolved full path of the file, use the "resolve_file_path" command. For more information about resolving file names and view an example, type "resolve_file_path -long_help".

In version 5.0 of this package, two new Tcl commands "get_all_assignments" and "get_assignment_info" have been introduced to replace the following commands:

```
get_all_quartus_defaults
    get_all_global_assignments
    get_all_instance_assignments
    get_all_parameters
```

These two new commands simplify the interface to retrieve information about Quartus II Settings File (.qsf) and Quartus II Default Settings File (.qdf) assignments.

In addition, the new "assignment_group" command replaces the deprecated "timegroup" command.

In version 6.0, all Tcl commands designed to process Timing Analyzer assignments have been moved to the ::quartus::timing_assignment package.

This package is loaded by default in the following executables:

- quartus
- quartus_cdb
- quartus_drc
- quartus_eda
- quartus_sh
- quartus_si
- quartus_sim
- quartus_sta
- quartus_stp
- quartus_tan

This package includes the following commands:

## assignment_group

### Usage

```
assignment_group [-add_exception <name>] [-add_member <name>] [-comment <comment>]
[-disable] [-get_exceptions] [-get_members] [-overwrite] [-remove] [-remove_exception
<name>] [-remove_member <name>] [-tag <data>] <group_name>
```

### Options

```
-add_exception <name>: Tcl list of exception names to add
-add_member <name>: Tcl list of member names to add
-comment <comment>: Comment
-disable: Option to disable assignment
-get_exceptions: Option to get collection of assignment group exceptions
-get_members: Option to get collection of assignment group members
-overwrite: Option to overwrite existing assignment group with the same group name
-remove: Option to remove assignment group
-remove_exception <name>: Tcl list of exception names to remove
-remove_member <name>: Tcl list of member names to remove
-tag <data>: Option to tag data to this assignment
<group_name>: Assignment group name
```

### Description

Adds, removes, gets members of, or gets exceptions to an assignment group.

The "assignment_group" command replaces the deprecated "timegroup" command.

An assignment group is a custom group of registers and pins. You can use the "-add_member" option to specify register or pin names you want to include in the assignment group. You can use the "-add_exception" option to specify names you want to exclude from the assignment group.

You can specify the names using wildcards, that is, using "?" or "*". For example, to add all registers and pins that start with a "b" except those that start with "b|c|" to a particular assignment group named "group_b", type:

```
assignment_group "group_b" -add_member "b*" -add_exception "b|c|*"
```

To remove members or exceptions from a assignment group, use the "-remove_member" or "-remove_exception" options respectively.

The "-get_members" option returns a collection of members in the assignment group. The "-get_exceptions" option returns a collection of exceptions to the assignment group. To access each element of the collection, use the Tcl command "foreach_in_collection". To see example usage, type "assignment_group -long_help" or "foreach_in_collection -long_help".

Specifying registers and pins in terms of an assignment group allows you to set timing constraints easily. For example, to make a multicycle assignment from nodes "a1" and "a2" to nodes "b1", "b2", and "b3", type the following:

```
assignment_group "group_a" -add_member [list "a1" "a2"]
assignment_group "group_b" -add_member [list "b1" "b2" "b3"]

set_multicycle_assignment -from "group_a" -to "group_b" 2
```

This command sets a multicycle assignment from every member of "group_a" to every member of "group_b". Quartus II timing analysis is optimized to use assignment groups in handling timing constraints.

To disable assignment group assignments for the entire group, use the "-disable" option, for example:

```
assignment_group "group_a" -disable
```

To disable a particular assignment group assignment, use the "-disable" option with the "-add_member" or "-add_exception" options, for example:

```
assignment_group "group_a" -add_member "m1" -disable
assignment_group "group_a" -add_exception "e1" -disable
```

Assignments created or modified by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments
- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

## Example

```
# Make timing cut assignment from nodes starting
# with "r" except those starting with "r|s|"
# and except those starting with "r|t|"
# to nodes "t1", "t2", and "t3"
assignment_group "tg1" -add_member "r*" -add_exception "r|s|*"
assignment_group "tg1" -add_exception "r|t|*"

assignment_group "tg2" -add_member [list "t1" "t2" "t3"]

set_timing_cut_assignment -from "group_a" -to "group_b" 2

# Remove the "t1" from a particular assignment group named "tg2"
assignment_group "tg2" -remove_member "t1"

# Display the members of a particular assignment group named "tg1"
foreach_in_collection member [assignment_group "tg1" -get_members] {

    # Print the name of the member
    puts $member
}

# Display the exceptions to a particular assignment group named "tg1"
foreach_in_collection exception [assignment_group "tg1" -get_exceptions] \
     {

    # Print the name of the exception
    puts $exception
}
```

## create_revision

### Usage

```
create_revision [-based_on <revision_name>] [-copy_results] [-set_current]
<revision_name>
```

### Options

```
-based_on <revision_name>: Revision name on which new revision bases its settings
```

```
-copy_results: Option to copy results from "based_on" revision
```

```
-set_current: Option to set new revision as current revision
```

```
<revision_name>: Revision name
```

### Description

Creates the specified revision. If the revision is not included in the current project, a new revision is created in the project with default settings.

If you specify the "-set_current" option, this command sets the newly created revision as the current revision.

If you specify the "-based_on" option, the command creates a new revision in the project based on the settings of the based-on revision specified by the option.

### Example

```
## Create a new revision called "tmp"
create_revision tmp

## Create a new revision called "tmp"
## and set it as the current revision
create_revision tmp -set_current
## This method is the same as
create_revision tmp
set_current_revision tmp

## Create a new revision called "speed_ch"
## with settings based on "chiptrip"
## and set it as the current revision
create_revision speed_ch -based_on chiptrip -set_current
```

## delete_revision

### Usage

```
delete_revision <revision_name>
```

### Options

```
<revision_name>: Revision name
```

### Description

Deletes the specified revision from the current project. The corresponding <revision name>.qsf file is deleted as well.

### Example

```
## Delete the revision called "tmp"
delete_revision tmp
```

## execute_assignment_batch

### Usage

```
execute_assignment_batch <tcl commands>
```

### Options

```
<tcl commands>: Tcl list of Tcl commands
```

### Description

Iterates through the specified Tcl list of Tcl commands and executes each command sequentially in batch mode.

In batch mode, Tcl commands that set Quartus II Settings File (.qsf) assignments are optimized to prevent them from repeatedly write-locking and write-unlocking the QSF during consecutive calls, thereby slowing down the execution.

Currently, only the following commands are supported:

```
     assignment_group
     remove_all_global_assignments
     remove_all_instance_assignments
     remove_all_parameters
     set_global_assignment
     set_instance_assignment
     set_io_assignment
     set_location_assignment
     set_parameter
     set_power_file_assignment
```

### Example

```
project_open one_wire
set tcl_cmds [list [list set_global_assignment -name FAMILY StratixII] \
    [list set_global_assignment -name DEVICE AUTO] \
    [list set_global_assignment -name TOP_LEVEL_ENTITY one_wire] \
    [list set_global_assignment -name SAVE_DISK_SPACE OFF] \
    [list set_location_assignment PIN_1 -to in1] \
    [list set_instance_assignment -name MULTICYCLE 4 -from in1 -to out1] \
    [list set_parameter -name STYLE FAST]]
execute_assignment_batch $tcl_cmds
project_close
```

## export_assignments

### Usage

```
export_assignments [-reorganize]
```

### Options

```
-reorganize: Option to reorganize the Quartus II Settings File (.qsf)
```

### Description

Exports assignments for the current revision to the Quartus II Settings File (.qsf).

Assignments created or modified during an open project are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments
- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

### Example

```
## The most common use of export_assignments is to
## call it before doing a system call
## to call a compiler command-line executable
project_open $project_name
set_global_assignment -name FAMILY Stratix

## Before calling quartus_map,
## write out the FAMILY assignment
export_assignments

## Now, call quartus_map
qexec "[file join $::quartus(binpath) quartus_map] $project_name"
```

## get_all_assignment_names

### Usage

```
get_all_assignment_names [-family <family>] [-module
<all|map|fit|tan|asm|eda|drc|generic>] [-type <all|global|instance>]
```

### Options

`-family <family>: Option to filter based on the specified device family. Defaults to all families.`

`-module <all|map|fit|tan|asm|eda|drc|generic>: Option to filter based on the specified flow module. Defaults to all.`

`-type <all|global|instance>: Option to filter based on the specified assignment type. Defaults to all.`

### Description

Returns a filtered output list of all available, matching assignment names.

The module option takes one of the following values:

| Module | Description |
|---|---|
| map | Analysis & Synthesis assignment names |
| fit | Fitter assignment names |
| tan | Classic Timing Analyzer assignment names |
| asm | Assembler assignment names |
| eda | EDA Netlist Writer assignment names |
| drc | Design Assistant assignment names |
| generic | Other assignment names not included in any of the above flow modules |
| all | All assignment names |

### Example

```
## Print out all available global assignments
foreach i [get_all_assignment_names -type global] {
    puts $i
}

## Print out all available global assignments
## for the Stratix family
foreach i [get_all_assignment_names -type global -family Stratix] {
    puts $i
}

## Print out all available global assignments
## for the Stratix family required
## by the Analysis & Synthesis module
foreach i \
    [get_all_assignment_names -type global -family Stratix -module map] {
    puts $i
}
```

## get_all_assignments

### Usage

```
get_all_assignments [-entity <entity_name>] [-from <source>] -name <name> [-section_id
<section id>] [-tag <data>] [-to <destination>] -type
<global|instance|parameter|default>
```

### Options

```
-entity <entity_name>: Entity name
-from <source>: Source name (string pattern is matched using Tcl string matching)
-name <name>: Assignment name (string pattern is matched using Tcl string matching)
-section_id <section id>: Section id
-tag <data>: Option to tag data to this assignment
-to <destination>: Destination name (string pattern is matched using Tcl string matching)
-type <global|instance|parameter|default>: Option to specify the type of assignments to
return
```

### Description

Returns a collection of all matching global, instance, parameter, or default assignment ids. To iterate through each assignment id in this collection, use the Tcl command "foreach_in_collection".

To view details for the assignment that is associated with the assignment id, use the Tcl command "get_assignment_info".

The "get_all_assignments" command is easier to use than the deprecated commands listed in Table 1.

Table 1. The -type Option

| Value for -type Option | Deprecated Tcl command |
|---|---|
| default | get_all_quartus_defaults |
| global | get_all_global_assignments |
| instance | get_all_instance_assignments |
| parameter | get_all_parameters |

The "-name" option is not case sensitive. The "-to" and "-from" options are case sensitive.

These options can take string patterns containing special characters from the set "*?\[]" as values. The values are matched using Tcl string matching. Note that bus names are automatically detected and do not need to be escaped. Bus names have the following format:

```
<bus name>[<bus index>] or <bus name>[*]
```

The <bus name> portion is a string of alphanumeric characters. The <bus index> portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the <bus index> is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names and can be used as follows:

```
# To match index 0 of bus "a", type:
get_all_assignments -type instance -name LOCATION -to a[0]

# To match all indices of bus "a", type:
get_all_assignments -type instance -name LOCATION -to a[*]
```

All other uses of square brackets must be escaped if you do not intend to use them as string patterns. For example, to match indices 0, 1, and 2 of the bus "a", type:

```
get_all_assignments -type instance LOCATION -to "a[escape_brackets
\[]\[0-2\][escape_brackets \]]"
```

For more information about escaping square brackets, type "escape_brackets -h".

This Tcl command reads in the global, instance, and parameter assignments found in the Quartus II Settings File (.qsf) and reads in the default assignments found inside the Quartus II Default Settings File (.qdf).

If you tagged data by making assignments with the -tag option, then the information can be searched using the -tag option.

Certain sections in the .qsf can appear more than once. For example, because there may be more than one clock used in a project, there may be more than one clock section each containing its own set of clock assignments. To uniquely identify sections of this type, use the -section_id option.

For entity-specific assignments, use the "-entity" option to retrieve assignments from a specific entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.

## Example

```
    ## View all the timing requirements using wildcards
    ## to match TSU_REQUIREMENT, TCO_REQUIREMENT,
    ## and others.
foreach_in_collection asgn_id \
    [get_all_assignments -type instance -name *_REQUIREMENT] {

    set from   [get_assignment_info $asgn_id -from]
    set to     [get_assignment_info $asgn_id -to]
    set name   [get_assignment_info $asgn_id -name]
    set value  [get_assignment_info $asgn_id -value]
    set entity [get_assignment_info $asgn_id -entity]
    set sid    [get_assignment_info $asgn_id -section_id]
    set tag    [get_assignment_info $asgn_id -tag]

    puts "$entity: $name ($from -> $to) = $value"
}

    ## View all global assignments
foreach_in_collection asgn_id [get_all_assignments -type global -name *] \
    {

    set name   [get_assignment_info $asgn_id -name]
    set value  [get_assignment_info $asgn_id -value]
    set entity [get_assignment_info $asgn_id -entity]
    set sid    [get_assignment_info $asgn_id -section_id]
    set tag    [get_assignment_info $asgn_id -tag]

    puts "$entity: $name = $value"
}

    ## View all project-wide default parameter values
foreach_in_collection asgn_id \
    [get_all_assignments -type parameter -name *] {

    set name  [get_assignment_info $asgn_id -name]
    set value [get_assignment_info $asgn_id -value]
    set tag    [get_assignment_info $asgn_id -tag]

    puts "$name = $value"
}

    ## View all entity-specific parameter values
```

```
foreach_in_collection asgn_id \
    [get_all_assignments -type parameter -name * -to *] {

    set dest  [get_assignment_info $asgn_id -to]
    set name  [get_assignment_info $asgn_id -name]
    set value [get_assignment_info $asgn_id -value]
    set tag    [get_assignment_info $asgn_id -tag]

    puts "$name (-> $dest) = $value"
}

    ## View all default assignments
foreach_in_collection asgn_id \
    [get_all_assignments -type default -name * -to *] {

    set name    [get_assignment_info $asgn_id -name]
    set value   [get_assignment_info $asgn_id -value]

    puts "$name = $value"
}
```

## get_all_global_assignments

### Usage

```
get_all_global_assignments [-entity <entity_name>] -name <name> [-section_id <section
id>] [-tag <data>]
```

### Options

```
-entity <entity_name>: Entity to which assignment belongs
-name <name>: Assignment name (string pattern is matched using Tcl string matching)
-section_id <section id>: Section id
-tag <data>: Option to tag data to this assignment
```

### Description

Returns a filtered output collection of all matching global assignment values. To access each element of the output collection, use the Tcl command "foreach_in_collection". To see example usage, type "foreach_in_collection -long_help".

In version 5.0 of the ::quartus::project package, two new Tcl commands "get_all_assignments" and "get_assignment_info" have been introduced to replace the "get_all_global_assignments" command. These two new commands simplify the interface to retrieve information about Quartus II Settings File (.qsf) assignments. The "get_all_global_assignments" command is still supported for backward compatibility.

The "-name" option is not case sensitive. This option can take string patterns containing special characters from the set "*?\[]" as the value. The value is matched using Tcl string matching.

This Tcl command reads the global assignments found in the Quartus II Settings File (.qsf). This Tcl command filters the assignment data in the .qsf and outputs the data based on the values given by the "-name" option.

Each element of the collection is a list with the following format: { {<Section Id>} {<Assignment name>} {<Assignment value>} {<Entity name>} {<Tag data>} }

Certain sections in the .qsf can appear more than once. For example, because there may be more than one clock used in a project, there may be more than one CLOCK section each containing its own set of clock assignments. To uniquely identify sections of this type, a <Section Id> is used. <Section Id> can be one of three types. It can be the same as the revision name, or it can be some unique name. The following is a list of sections requiring a <Section Id> and the associated <Section Id> description:

| Section Id | Description |
|---|---|
| CHIP | Same as revision name |
| LOGICLOCK_REGION | A unique name |
| EDA_TOOL_SETTINGS | A unique name |
| CLIQUE | A unique name |
| BREAKPOINT | A unique name |
| CLOCK | A unique name |
| AUTO_INSERT_SLD_NODE_ENTITY | A unique name |

If you tagged data by making assignments with the -tag option, then the information will be displayed in the <Tag data> field.

For entity-specific assignments, use the "-entity" option to retrieve the assignment(s) from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.

## Example

```
## Print out all the registered source files
## using the foreach_in_collection method
set file_asgn_col [get_all_global_assignments -name SOURCE_FILE]
foreach_in_collection file_asgn $file_asgn_col {

    ## Each element in the collection has the following
    ## format: {} {SOURCE_FILE} {<file_name>} {} {}
    puts [lindex $file_asgn 2]
}

## Print out all global assignments
set asgn_col [get_all_global_assignments -name *]

foreach_in_collection asgn $asgn_col {

    ## Each element in the collection has the following
    ## format: { {} {<Assignment name>} {<Assignment value>} {<Entity
    # name>} {<Tag data>} }
    set name   [lindex $asgn 1]
    set value  [lindex $asgn 2]
    set entity [lindex $asgn 3]
    set tag    [lindex $asgn 4]

    puts "$entity: $name = $value"
}
```

## get_all_instance_assignments

### Usage

```
get_all_instance_assignments [-entity <entity_name>] [-from <source>] -name <name>
[-section_id <section id>] [-tag <data>] [-to <destination>]
```

### Options

`-entity <entity_name>: Entity to which assignment belongs`

`-from <source>: Source of assignment (string pattern is matched using Tcl string matching)`

`-name <name>: Assignment name (string pattern is matched using Tcl string matching)`

`-section_id <section id>: Section id`

`-tag <data>: Option to tag data to this assignment`

`-to <destination>: Destination of assignment (string pattern is matched using Tcl string matching)`

### Description

Returns a filtered output collection of all matching instance assignment values. To access each element of this output collection, use the Tcl command "foreach_in_collection". To see example usage, type "foreach_in_collection -long_help".

In version 5.0 of the ::quartus::project package, two new Tcl commands "get_all_assignments" and "get_assignment_info" have been introduced to replace the "get_all_instance_assignments" command. These two new commands simplify the interface to retrieve information about Quartus II Settings File (.qsf) assignments. The "get_all_instance_assignments" command is still supported for backward compatibility.

The "-name" option is not case sensitive. The "-to" and "-from" options are case sensitive.

These options can take string patterns containing special characters from the set "*?\[]" as values. The values are matched using Tcl string matching. Note that bus names are automatically detected and do not need to be escaped. Bus names have the following format:

```
<bus name>[<bus index>] or <bus name>[*]
```

The <bus name> portion is a string of alphanumeric characters. The <bus index> portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the <bus index> is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names and can be used as follows:

```
# To match index 0 of bus "a", type:
get_all_instance_assignments -name LOCATION -to a[0]

# To match all indices of bus "a", type:
get_all_instance_assignments -name LOCATION -to a[*]
```

All other uses of square brackets must be escaped if you do not intend to use them as string patterns. For example, to match indices 0, 1, and 2 of the bus "a", type:

```
get_all_instance_assignments -name LOCATION -to "a[escape_brackets
\[]\[0-2\][escape_brackets \]]"
```

For more information about escaping square brackets, type "escape_brackets -h".

This Tcl command reads in the instance assignments found in the Quartus II Settings File (.qsf). The command filters the assignments data found in the .qsf and outputs the data based on the values specified by the "-name", "-from", and "-to" options.

Each element of the collection is a list with the following format: { {<Section Id>} {<Source>} {<Destination>} {<Assignment name>} {<Assignment value>} {<Entity name>} {<Tag data>} }

Certain sections in the .qsf can appear more than once. For example, because there may be more than one clock used in a project, there may be more than one CLOCK section each containing its own set of clock assignments. To uniquely identify sections of this type, a <Section Id> is used. <Section Id> can be one of three types. It can be the same as the revision name, or it can be some unique name. The following is a list of sections requiring a <Section Id> and the associated <Section Id> description:

| Section Id | Description |
|---|---|
| CHIP | Same as revision name |
| LOGICLOCK_REGION | A unique name |
| EDA_TOOL_SETTINGS | A unique name |
| CLIQUE | A unique name |
| BREAKPOINT | A unique name |
| CLOCK | A unique name |
| AUTO_INSERT_SLD_NODE_ENTITY | A unique name |

If you tagged data by making assignments with the -tag option, then the information will be displayed in the <Tag data> field.

For entity-specific assignments, use the "-entity" option to retrieve the assignment(s) from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.

## Example

```
## Print out all the timing requirements
## using the foreach_in_collection method.
## Use wildcards to catch TSU_REQUIREMENT, TCO_REQUIREMENT,
## and others.
set asgn_col [get_all_instance_assignments -name *_REQUIREMENT]

foreach_in_collection asgn $asgn_col {

    ## Each element in the collection has the following
    ## format: { {} {<Source>} {<Destination>} {<Assignment name>}
    # {<Assignment value>} {<Entity name>} {<Tag data>} }
    set from   [lindex $asgn 1]
    set to     [lindex $asgn 2]
    set name   [lindex $asgn 3]
    set value  [lindex $asgn 4]
    set entity [lindex $asgn 5]
    set tag    [lindex $asgn 6]

    puts "$entity: $name ($from -> $to) = $value"
}

## Get all the location assignments with
## the destination bus name "timeo".
set bus_name "timeo"
set location_asgns \
    [get_all_instance_assignments -name LOCATION -to $bus_name[*]]
```

## get_all_parameters

### Usage

```
get_all_parameters [-entity <entity_name>] -name <name> [-tag <data>] [-to
<destination>]
```

### Options

```
-entity <entity_name>: Entity to which parameter belongs
```

```
-name <name>: Parameter name (string pattern is matched using Tcl string matching)
```

```
-tag <data>: Option to tag data to this assignment
```

```
-to <destination>: Destination of the parameter (string pattern is matched using Tcl
string matching)
```

### Description

Returns a filtered output collection of all matching parameter values. To access each element of this output collection, use the Tcl command "foreach_in_collection". To see example usage, type "foreach_in_collection -long_help".

In version 5.0 of ::quartus::project package, two new Tcl commands "get_all_assignments" and "get_assignment_info" have been introduced to replace the "get_all_parameters" command. These two new commands simplify the interface to retrieve information about Quartus II Settings File (.qsf) assignments. The "get_all_parameters" command is still supported for backward compatibility.

The "-name" option is not case sensitive. The "-to" option is case sensitive.

If the "-to" argument is specified, the function returns the parameter values for the current entity. The values are retrieved from the PARAMETERS section of the entity. Otherwise, the function returns the project-wide default parameter values obtained from the DEFAULT_PARAMETERS section.

This Tcl command filters the parameter data found in the Quartus II Settings File (.qsf) and outputs the data based on the values specified by the "-name" and "-to" options. These options can take string patterns containing special characters from the set "*?\[]" as values. The values are matched using Tcl string matching. Note that bus names are automatically detected and do not need to be escaped. Bus names have the following format:

```
<bus name>[<bus index>] or <bus name>[*]
```

The <bus name> portion is a string of alphanumeric characters. The <bus index> portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the <bus index> is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names and can be used as follows:

```
# To match index 0 of bus "a", type:
get_all_parameters -name * -to a[0]
```

```
# To match all indices of bus "a", type:
get_all_parameters -name * -to a[*]
```

All other uses of square brackets must be escaped if you do not intend to use them as string patterns. For example, to match indices 0, 1, and 2 of the bus "a", type:

```
get_all_parameters -name * -to "a[escape_brackets \[]\[0-2\][escape_brackets \]]"
```

For more information about escaping square brackets, type "escape_brackets -h".

Each element of the collection is a list with the following format: { {<Destination>} {<Parameter name>} {<Parameter value>} {<Entity name>} {<Tag data>} }

If you tagged data by making assignments with the -tag option, then the information will be displayed in the <Tag data> field.

Use the "-entity" option to retrieve the parameter values from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.

## Example

```
## Display all project-wide default parameter values
set parameter_col [get_all_parameters -name *]

foreach_in_collection parameter $parameter_col {

    ## Each element in the collection has the following
    ## format: { {} {<Parameter name>} {<Parameter value>} {} {} }
   set name [lindex $parameter 1]
   set value [lindex $parameter 2]

   ## Now, display the content of the parameter
   puts "Parameter Name ($name)"
   puts "Parameter Value ($value)"
}

## Display all entity-specific parameter values
foreach_in_collection parameter [get_all_parameters -name * -to *] {

    ## Each element in the collection has the following
    ## format: { {Destination} {<Parameter name>} {<Parameter value>} {}
    # {} }
   set dest [lindex $parameter 0]
   set name [lindex $parameter 1]
   set value [lindex $parameter 2]

   ## Now, display the content of the parameter
   puts "Destination ($dest)"
   puts "Parameter Name ($name)"
   puts "Parameter Value ($value)"
}
```

## get_all_quartus_defaults

### Usage

```
get_all_quartus_defaults [-name <name>] [-section_id <section id>]
```

### Options

```
-name <name>: Assignment name (string pattern is matched using Tcl string matching)
-section_id <section id>: Section id
```

### Description

Returns a filtered output collection of all matching default assignment values. To access each element of the output collection, use the Tcl command "foreach_in_collection". To see example usage, type "foreach_in_collection -long_help".

In version 5.0 of ::quartus::project package, two new Tcl commands "get_all_assignments" and "get_assignment_info" have been introduced to replace the "get_all_quartus_defaults" command. These two new commands simplify the interface to retrieve information about Quartus II Settings File (.qsf) assignments. The "get_all_quartus_defaults" command is still supported for backward compatibility.

The "-name" option is not case sensitive. This option can take string patterns containing special characters from the set "*?\[]" as the value. The value is matched using Tcl string matching.

This Tcl command reads in the default assignments found inside the Quartus II Default Settings File (.qdf). It filters the assignments data found inside the .qdf and outputs the data based on the values specified by the "-name" option.

Each element of the collection is a list with the following format: { {<Section Id>} {<Assignment name>} {<Assignment value>} }

Certain sections in the .qsf can appear more than once. For example, because there may be more than one clock used in a project, there may be more than one CLOCK section each containing its own set of clock assignments. To uniquely identify sections of this type, a <Section Id> is used. <Section Id> can be one of three types. It can be the same as the revision name, or it can be some unique name. The following is a list of sections requiring a <Section Id> and the associated <Section Id> description:

| Section Id | Description |
|---|---|
| CHIP | Same as revision name |
| LOGICLOCK_REGION | A unique name |
| EDA_TOOL_SETTINGS | A unique name |
| CLIQUE | A unique name |
| BREAKPOINT | A unique name |
| CLOCK | A unique name |
| AUTO_INSERT_SLD_NODE_ENTITY | A unique name |

### Example

```
## Print out all the default assignments using
## the foreach_in_collection method

set default_asgns_col [get_all_quartus_defaults]
foreach_in_collection default $default_asgns_col {
    set sect_id [lindex $default 0]
```

```
   set name [lindex $default 1]
   set value [lindex $default 2]

   ## Now, display the content of the assignment
   puts "Section ID ($sect_id)"
   puts "Assignment Name ($name)"
   puts "Assignment Value ($value)"
}

## Using wildcards
set default_asgns_col [get_all_quartus_defaults -name *]
foreach_in_collection default $default_asgns_col {
   set sect_id [lindex $default 0]
   set name [lindex $default 1]
   set value [lindex $default 2]

   ## Now, display the content of the assignment
   puts "Section ID ($sect_id)"
   puts "Assignment Name ($name)"
   puts "Assignment Value ($value)"
}
```

## get_all_user_option_names

### Usage

```
get_all_user_option_names [-name <name>]
```

### Options

```
-name <name>: User option name (string pattern is matched using Tcl string matching)
```

### Description

Returns a filtered output list of all available, matching user option names.

If the "-name" option is not specified, all available user option names are returned. Otherwise, only the matching user option names are returned.

The "-name" option is not case sensitive. This option can take string patterns containing special characters from the set "*?\[]" as the value. The value is matched using Tcl string matching.

### Example

```
## Print out all available user option names
foreach i [get_all_user_option_names] {
    puts $i
}

## Display all user option names that contain
## the word "talkback" and also display the
## value for each of the user option names
foreach i [get_all_user_option_names -name *talkback*] {
   set name $i
   set value [get_user_option -name $i]
    puts "$name = $value"
}
```

## get_assignment_info

### Usage

```
get_assignment_info [-entity] [-from] [-get_tcl_command] [-name] [-section_id] [-tag]
[-to] [-value] <asgn_id>
```

### Options

`-entity: Option to get the assignment entity`

`-from: Option to get the assignment source`

`-get_tcl_command: Option to get the tcl command that sets the assignment`

`-name: Option to get the assignment name`

`-section_id: Option to get the assignment section id`

`-tag: Option to get the assignment tag`

`-to: Option to get the assignment destination`

`-value: Option to get the assignment value`

`<asgn_id>: Assignment id`

### Description

Returns information for the assignment id based on the specified option.

The assignment id is obtained from the "get_all_assignments" Tcl command.

### Example

```
    ## View all the instance assignments
foreach_in_collection asgn_id [get_all_assignments -type instance -name \
    *] {

    set from   [get_assignment_info $asgn_id -from]
    set to     [get_assignment_info $asgn_id -to]
    set name   [get_assignment_info $asgn_id -name]
    set value  [get_assignment_info $asgn_id -value]
    set entity [get_assignment_info $asgn_id -entity]
    set tag    [get_assignment_info $asgn_id -tag]

    puts "$entity: $name ($from -> $to) = $value"
}
```

## get_assignment_name_info

### Usage

```
get_assignment_name_info <name>
```

### Options

```
<name>: Assignment name
```

### Description

Returns information for the specified assignment name.

### Example

```
    ## View information for all assignment names
foreach name [get_all_assignment_names] {
    puts [get_assignment_name_info $name]
}
```

## get_current_revision

### Usage

```
get_current_revision <project_name>
```

### Options

```
<project_name>: Project name
```

### Description

Returns the name of the current revision for the specified project. If the project name is not specified, the current project name is used.

### Example

```
# Get the current revision name for
# the currently open project "chiptrip"
project_open chiptrip
set revision_name [get_current_revision]
project_close

# Get the current revision name for
# a project that is not currently open
set revision_name [get_current_revision chiptrip]
```

## get_global_assignment

### Usage

```
get_global_assignment [-entity <entity_name>] [-front] -name <name> [-section_id
<section id>] [-tag <data>]
```

### Options

`-entity <entity_name>: Entity to which assignment belongs`

`-front: Option to return the first assignment if there is more than one assignment found`

`-name <name>: Assignment name`

`-section_id <section id>: Section id`

`-tag <data>: Option to tag data to this assignment`

### Description

Returns the value of the global assignment.

The "-name" option is not case sensitive.

For entity-specific assignments, use the "-entity" option to retrieve the assignment from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.

### Example

```
## Get the value of the FAMILY assignment
get_global_assignment -name FAMILY
```

## get_instance_assignment

### Usage

```
get_instance_assignment [-entity <entity_name>] [-from <source>] [-front] -name <name>
[-section_id <section id>] [-tag <data>] [-to <destination>]
```

### Options

`-entity <entity_name>: Entity to which assignment belongs`

`-from <source>: Source of assignment`

`-front: Option to return the first assignment if there is more than one assignment found`

`-name <name>: Assignment name`

`-section_id <section id>: Section id`

`-tag <data>: Option to tag data to this assignment`

`-to <destination>: Destination of assignment`

### Description

Returns the value of the instance assignment.

The "-name" option is not case sensitive. The "-entity", "-to", and "-from" options are case sensitive.

For entity-specific assignments, use the "-entity" option to retrieve the assignment from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.

### Example

```
## Get the TSU_REQUIREMENT from mypin to any register
set value \
    [get_instance_assignment -from "mypin" -to * -name TSU_REQUIREMENT]
puts "TSU_REQUIREMENT(mypin->*) = $value"
```

## get_location_assignment

### Usage

```
get_location_assignment [-tag <data>] -to <destination>
```

### Options

```
-tag <data>: Option to tag data to this assignment
-to <destination>: Destination of assignment
```

### Description

Returns the value of a location assignment.

The "-chip" option is not case sensitive. The "-to" option is case sensitive.

### Example

```
get_location_assignment -to dst
```

## get_name_info

### Usage

```
get_name_info [-get_synonyms] [-info
<parent_name_id|base_name|entity_name|instance_name|full_path|short_full_path|node_typ
e|creator|signaltapii|file_location>] [-observable_type
<all|pre_synthesis|post_synthesis|post_fitter|post_asm|stp_pre_synthesis>]
[-use_cached_database] <name_id>
```

### Options

```
-get_synonyms: Option to get a Tcl list of synonym name ids.
```

```
-info
<parent_name_id|base_name|entity_name|instance_name|full_path|short_full_path|node_typ
e|creator|signaltapii|file_location>: Option to specify the type of information to
display.
```

```
-observable_type
<all|pre_synthesis|post_synthesis|post_fitter|post_asm|stp_pre_synthesis>: Option to
specify the observable type of the name ID
```

```
-use_cached_database: Option not to check the compilation database for updates. Use this
option only if you do not compile the project after getting the name IDs from "get_names"
command. If you compile the project, you must re-run "get_names" command again to refresh
the cached database information before using this option.
```

```
<name_id>: Option to specify the node name ID
```

### Description

Displays the specified type of information for the specified node name id. Type "get_names -long_help" to view how to get a collection of node name IDs.

If the "-observable_type" option is not specified, the default value is "all". The specified observable type must have the same observable type as specified in the "get_names" Tcl command which returned the currently specified node name id.

The value for "-observable_type" option can be one of the following:

| Observable Type | Description |
|---|---|
| all | Use post-Fitter information. If it is not available, post-synthesis information is used. Otherwise, pre-synthesis information is used if it exists. |
| pre_synthesis | Use pre-synthesis information. |
| post_synthesis | Use post-synthesis information. |
| post_fitter | Use post-Fitter information. |
| post_asm | Use post-Assembler information. The post-Assembler information is only supported for designs using the HardCopy II device family. |
| stp_pre_synthesis | Use SignalTap II pre-synthesis information. |

The info type for the "-info" option can be one of the following:

| Info Type | Description |
|---|---|
| parent_name_id | The name id for the node's parent. |
| base_name | The node name, which consists of an entity name and/or an instance name separated by a colon if necessary. |
| entity_name | The entity name. |
| instance_name | The instance name. |
| full_path | The full hierarchy path name, which consists of entity name(s) and/or the instance name(s). This path name excludes the current focus entity. If there is nothing shown, the name id is the current focus entity's name id. |
| short_full_path | The short full hierarchy path name, which consists of the instance name(s). This path name excludes the current focus entity. If nothing is shown, the name id is the current focus entity's name id. |
| node_type | The node type, which can be one of the types supported by "get_names", namely, "input", "output", "bidirectional", "register", "combinational", "hierarchy", "memory", or "bus". If "pin" type was specified for "get_names" command, the node type shown here is expanded to be "input", "output", or "bidirectional". Node type value of "qsf" indicates name originates from qsf settings file. |
| creator | The creator of the node, which is either "user_entered" or "compiler_generated". |
| signaltapii | If this node can be connected to a SignalTap®II embedded logic analyzer, 1 is shown. Otherwise, 0 is shown. |
| file_location | The source file location. For example, the source file location for the entity chiptrip is "chiptrip.v". To get the full path to the source file, use the command "resolve_file_path" which exists only in version 4.0 or later of ::quartus::project package. |

## Example

```
# Get the name id of the current focus entity
set current_focus_entity_id [get_top_level_entity]

# The full path name of the current focus entity
# is empty because the full path excludes the
# current focus entity
set msg "Full path of the current focus entity => ("
append msg [get_name_info -info full_path $current_focus_entity_id]
append msg ")"
puts $msg
puts ""

# Get the node type of the current focus entity
# The node type should be a hierarchy type
set msg "Node type of the current focus entity => ("
append msg [get_name_info -info node_type $current_focus_entity_id]
append msg ")"
puts $msg
```

## get_names

### Usage

```
get_names -filter <wildcard> [-node_type
<all|comb|reg|pin|input|output|bidir|hierarchy|mem|bus|qsf>] [-observable_type
<all|pre_synthesis|post_synthesis|post_fitter|post_asm|stp_pre_synthesis>]
```

### Options

```
-filter <wildcard>: Option to specify the node's full path name and/or wildcard
character(s)
```

```
-node_type <all|comb|reg|pin|input|output|bidir|hierarchy|mem|bus|qsf>: Option to
filter based on the specified node type.
```

```
-observable_type
<all|pre_synthesis|post_synthesis|post_fitter|post_asm|stp_pre_synthesis>: Option to
filter based on the specified observable type
```

### Description

Returns a filtered output collection of all matching node name IDs found in a compiled Quartus II project.

To access each element of the output collection, use the Tcl command "foreach_in_collection". To see example usage, type "get_names -long_help" or "foreach_in_collection -long_help".

If the "-node_type" option is not specified, the default value is "all". Similarly, if the "-observable_type" option is not specified, the default value is "all".

The node type "pin" includes "input", "output", and "bidir". The node type "qsf" include names from qsf settings file. The node type "all" includes all node types.

The value for "-observable_type" option can be one of the following:

| Observable Type | Description |
|---|---|
| all | Use post-Fitter information. If it is not available, post-Synthesis information is used. Otherwise, pre-synthesis information is used if it exists. |
| pre_synthesis | Use pre-synthesis information. |
| post_synthesis | Use post-synthesis information. |
| post_fitter | Use post-Fitter information. |
| post_asm | Use post-Assembler information. The post-Assembler information is only supported for designs using the HardCopy II device family. |
| stp_pre_synthesis | Use SignalTap II pre-synthesis information. |

### Example

```
# Search for a single post-Fitter pin with the name accel and
# make assignments
set accel_name_id \
    [get_names -filter accel -node_type pin -observable_type post_fitter]
foreach_in_collection name_id $accel_name_id {

   # Get the full path name of the node
   set target [get_name_info -info full_path $name_id]

   # Set multicycle assignment
   set_multicycle_assignment -to $target 2
```

```
    # Set location assignment
    set_location_assignment -to $target Pin_E22
}
# Search for nodes of any post-Fitter node type with name length <= 5
# The default node type is "all"
set name_ids [get_names -filter ????? -observable_type post_fitter]
foreach_in_collection name_id $name_ids {

    # Print the name id
    puts $name_id

    # Print the node type
    puts [get_name_info -info node_type $name_id]

    # Print the full path (which excludes the current
    # focus entity from the path)
    puts [get_name_info -info full_path $name_id]
}
# Search for nodes of any post-Fitter node type that end in "eed".
# The default node type is "all"
set name_ids [get_names -filter *eed -observable_type post_fitter]
foreach_in_collection name_id $name_ids {

    # Print the name id
    puts $name_id

    # Print the node type
    puts [get_name_info -info node_type $name_id]

    # Print the full path (which excludes the current
    # focus entity from the path)
    puts [get_name_info -info full_path $name_id]
}
```

## get_parameter

### Usage

```
get_parameter [-entity <entity_name>] -name <name> [-tag <data>] [-to <destination>]
```

### Options

```
-entity <entity_name>: Entity to which parameter belongs
```

```
-name <name>: Parameter name
```

```
-tag <data>: Option to tag data to this assignment
```

```
-to <destination>: Destination of parameter
```

### Description

Returns the value of the parameter.

The "-name" option is not case sensitive. The "-to" option is case sensitive.

If the "-to" argument is specified, the function returns the parameter value for the current entity. The value is retrieved from the PARAMETERS section of the entity. Otherwise, the function returns the project-wide default parameter value obtained from the DEFAULT_PARAMETERS section.

Use the "-entity" option to retrieve the parameter from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.

### Example

```
## Get project-wide, default parameter value
get_parameter -name WIDTH

## Get entity-specific parameter value
get_parameter -name inst1 -to SIZE
```

## get_project_directory

### Usage

```
get_project_directory
```

### Options

None

### Description

Returns the project directory for currently open project.

### Example

```
project_open one_wire
   # Print the current project directory
puts [get_project_directory]
project_close
```

## get_project_revisions

### Usage

```
get_project_revisions <project_name>
```

### Options

```
<project_name>: Project name
```

### Description

Returns a list of revisions included in the specified project. If the project name is not specified, the current project name is used by default.

The first element in the list of revisions is the current revision and is the same as the return value for the "get_current_revision" command.

### Example

```
# Set the device family assignment to Stratix
# for all revisions
project_open chiptrip
set original_revision [get_current_revision]

foreach revision [get_project_revisions] {
   puts "$revision"
   set_current_revision $revision
   set_global_assignment -name FAMILY Stratix
   export_assignments
}

set_current_revision $original_revision
project_close

# Open the project with the first available revision
# and set the device family assignment to Stratix
set revision [lindex [get_project_revisions chiptrip] 0]
open_project -revision $revision chiptrip
set_global_assignment -name FAMILY Stratix
project_close
```

## get_top_level_entity

### Usage

```
get_top_level_entity
```

### Options

None

### Description

Returns the name id for the current focus entity.

### Example

```
# Get the name id of the current focus entity
set current_focus_entity_id [get_top_level_entity]

# Print out the entity name of the current focus entity
set msg "Entity name of the current focus entity => ("
append msg [get_name_info -info entity_name $current_focus_entity_id]
append msg ")"
puts ""
puts $msg
```

## get_user_option

### Usage

```
get_user_option -name <name>
```

### Options

```
-name <name>: User option name
```

### Description

Returns the user option value for the name specified by the "-name" option.

To get a list of all available user option names, use the "get_all_user_option_names" command.

### Example

```
## Get the value for the user option
## "TALKBACK_ENABLED"
set value [get_user_option -name TALKBACK_ENABLED]
puts "TALKBACK_ENABLED = $value"
```

## is_project_open

### Usage

```
is_project_open
```

### Options

None

### Description

Checks whether a project is currently open. Returns 1, if a project is currently open; returns 0, otherwise.

### Example

```
## Close the project if open
if [is_project_open] {
   project_close
}
```

## project_archive

### Usage

```
project_archive [-all_revisions] [-include_libraries] [-include_outputs] [-overwrite]
[-use_file_set <file_set>] [-version_compatible_database] <archive_name>
```

### Options

```
-all_revisions: Option to archive all revisions
-include_libraries: Option to include related system libraries
-include_outputs: Option to include output files in archive
-overwrite: Option to overwrite any currently existing archive file
-use_file_set <file_set>: Option to create the archive using the specified file set
-version_compatible_database: Option to include version-compatible database if supported
<archive_name>: Archive file name
```

### Description

Archives an open project and its related files into a Quartus II Archive File (.qar).

The description of operations is as follows:

| Option | Description |
|---|---|
| use_file_set | Creates the archive using the specified file set. By default, the 'basic' file set is used. For more information about file sets, type: quartus_sh --archive -list_file_sets |
| all_revisions | Archives all revisions. |
| overwrite | Overwrites existing archive file. |
| include_outputs | Includes output files in archive. |
| include_libraries | Includes related Megafunction and IP library files. |
| version_compatible_database | Includes version-compatible database if supported. |

### Example

```
## Default mode: Archive current revisions without output files or
# libraries
project_archive chiptrip.qar

## Archive all revisions without output files or libraries
project_archive chiptrip.qar -all_revisions

## Archive current revision with version-compatible database if supported
project_archive chiptrip.qar -version_compatible_database

## Same as first one, but overwrite any existing archive file
project_archive chiptrip.qar -overwrite

## Include outut files and libraries
project_archive chiptrip.qar -include_outputs -include_libraries
```

## project_close

### Usage

```
project_close [-dont_export_assignments]
```

### Options

```
-dont_export_assignments: Do not export assignments to file
```

### Description

Closes an open project.

The assignments created or modified during an open project are committed to the Quartus II Settings File (.qsf) during a "project_close", unless you use the "-dont_export_assignments" option.

### Example

```
## Close the project if open
if [is_project_open] {
   project_close
}
## Close the project if open
## and do not export the assignments
if [is_project_open] {
   project_close -dont_export_assignments
}
```

## project_exists

### Usage

```
project_exists <project_name>
```

### Options

```
<project_name>: Project name
```

### Description

Checks whether a project exists. Returns 1, if a project exists; returns 0, otherwise.

### Example

```
## Create project if one does not exist.
## Open existing project otherwise.
if [project_exists chiptrip] {
   project_open chiptrip
} else {
   project_new chiptrip
}
```

## project_new

### Usage

```
project_new [-family <family>] [-overwrite] [-part <part>] [-revision <revision_name>]
<project_name>
```

### Options

```
-family <family>: Family name
```

```
-overwrite: Option to overwrite existing project and revision
```

```
-part <part>: Part name
```

```
-revision <revision_name>: Revision name
```

```
<project_name>: Project name
```

### Description

Creates and opens a new project with the specified project name.

If the "-revision" option is not specified, the project name is used to create the revision.

Assignments created or modified by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments
- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

### Example

```
## Create project "chiptrip" and revision "chiptrip"
project_new chiptrip

## Create project "chiptrip" and revision "auto_max"
project_new -revision auto_max chiptrip

## Create project "chiptrip" and revision "chiptrip"
## Overwrite any Quartus II Settings File (.qsf) if it exists
project_new chiptrip -overwrite

## Create project "chiptrip" and revision "chiptrip"
## Set the FAMILY assignment to Stratix
project_new chiptrip -family Stratix
```

## project_open

### Usage

```
project_open [-current_revision] [-force] [-revision <revision_name>] <project_name>
```

### Options

```
-current_revision: Option to open the current revision automatically
```

```
-force: Option to open the project and overwrite the compilation database if the database
version is incompatible.
```

```
-revision <revision_name>: Revision name
```

```
<project_name>: Project name
```

### Description

Opens an existing project. To create a new project, use the project_new command.

If the -revision option is not specified, the project name is specified as the revision name.

The project_open command gives an error when the compilation database version is not compatible with the current version of Quartus II software.  You may specify the "-force" option to avoid the error and overwrite the database.

### Example

```
## Open project "chiptrip" and revision "chiptrip"
project_open chiptrip

## Open project "chiptrip" and revision "auto_max"
project_open -revision auto_max chiptrip

## Get the current revision before opening
## the project with the current revision
set project_name chiptrip
set current_revision [get_current_revision $project_name]
project_open -revision $current_revision $project_name
puts [get_global_assignment -name FAMILY]
project_close
```

## project_restore

### Usage

```
project_restore [-destination <directory>] [-overwrite] [-update_included_file_info]
<archive_file>
```

### Options

```
-destination <directory>: Directory where restored files are placed
```

```
-overwrite: Option to overwrite files in destination directory
```

```
-update_included_file_info: Option to update included file information
```

```
<archive_file>: Archive file name
```

### Description

Restores a Quartus II Archive File (.qar) that contains the project and its related files.

By default, the archive is restored into the current directory. Use the "-destination" option to restore the files into a new directory.

By default, the command fails if the archive already contains files in the destination directory. Use the "-overwrite" option to overwrite any existing files in the destination directory.

### Example

```
## Restore archive and expand files into current directory
project_restore chiptrip.qar
## or
project_restore chiptrip.qar -destination

## Restore archive. Expand files into current directory,
## but overwrite any existing files in "."
project_restore chiptrip.qar -destination . -overwrite

## Restore project into a "restored" subdirectory
project_restore chiptrip.qar -destination "restored" -overwrite
```

## remove_all_global_assignments

### Usage

```
remove_all_global_assignments [-entity <entity_name>] -name <name> [-section_id <section
id>] [-tag <data>]
```

### Options

```
-entity <entity_name>: Entity to which assignment belongs
-name <name>: Assignment name (string pattern is matched using Tcl string matching)
-section_id <section id>: Section id
-tag <data>: Option to tag data to this assignment
```

### Description

Removes all matching global assignments.

The "-name" option is not case sensitive. This option can take string patterns containing special characters from the set "*?\[]" as the value. The value is matched using Tcl string matching.

This Tcl command reads the global assignments found in the Quartus II Settings File (.qsf). This Tcl command filters the assignments data found in the .qsf and removes the data based on the values specified by the "-name" option.

Certain sections in the .qsf can appear more than once. For example, because there may be more than one clock used in a project, there may be more than one CLOCK section each containing its own set of clock assignments. To uniquely identify sections of this type, a <Section Id> is used. <Section Id> can be one of three types. It can be the same as the revision name, or it can be some unique name. The following is a list of sections requiring a <Section Id> and the associated <Section Id> description:

| Section Id | Description |
|---|---|
| CHIP | Same as revision name |
| LOGICLOCK_REGION | A unique name |
| EDA_TOOL_SETTINGS | A unique name |
| CLIQUE | A unique name |
| BREAKPOINT | A unique name |
| CLOCK | A unique name |
| AUTO_INSERT_SLD_NODE_ENTITY | A unique name |

For entity-specific assignments, use the "-entity" option to remove the assignment(s) from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.

Assignments removed by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments
- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

## Example

```
## Remove all the registered source files

remove_all_global_assignments -name SOURCE_FILE

# Using wildcards
remove_all_global_assignments -name SOURCE*
```

## remove_all_instance_assignments

### Usage

```
remove_all_instance_assignments [-entity <entity_name>] [-from <source>] -name <name>
[-section_id <section id>] [-tag <data>] [-to <destination>]
```

### Options

```
-entity <entity_name>: Entity to which assignment belongs
```

```
-from <source>: Source of the assignment (string pattern is matched using Tcl string
matching)
```

```
-name <name>: Assignment name (string pattern is matched using Tcl string matching)
```

```
-section_id <section id>: Section id
```

```
-tag <data>: Option to tag data to this assignment
```

```
-to <destination>: Destination of the assignment (string pattern is matched using Tcl
string matching)
```

### Description

Removes all matching instance assignment values.

The "-name" option is not case sensitive. The "-to" and "-from" options are case sensitive.

These options can take string patterns containing special characters from the set "*?\[]" as values. The values are matched using Tcl string matching. Note that bus names are automatically detected and do not need to be escaped. Bus names have the following format:

```
<bus name>[<bus index>] or <bus name>[*]
```

The <bus name> portion is a string of alphanumeric characters. The <bus index> portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the <bus index> is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names and can be used as follows:

```
# To match index 0 of bus "a", type:
remove_all_instance_assignments -name LOCATION -to a[0]
```

```
# To match all indices of bus "a", type:
remove_all_instance_assignments -name LOCATION -to a[*]
```

All other uses of square brackets must be escaped if you do not intend to use them as string patterns. For example, to match indices 0, 1, and 2 of the bus "a", type:

```
remove_all_instance_assignments -name LOCATION -to "a[escape_brackets
\[]\[0-2\][escape_brackets \]]"
```

For more information about escaping square brackets, type "escape_brackets -h".

This Tcl command reads the instance assignments found in the Quartus II Settings File (.qsf) and removes this data based on the values specified by the "-name", "-from", and "-to" options.

Certain sections in the .qsf can appear more than once. For example, because there may be more than one clock used in a project, there may be more than one CLOCK section each containing its own set of clock assignments. To uniquely identify sections of this type, a <Section Id> is used. <Section Id> can be one of three types. It can be the same as the revision name, or it can be some unique name. The following is a list of sections requiring a <Section Id> and the associated <Section Id> description:

| Section Id | Description |
|---|---|
| CHIP | Same as revision name |
| LOGICLOCK_REGION | A unique name |
| EDA_TOOL_SETTINGS | A unique name |
| CLIQUE | A unique name |
| BREAKPOINT | A unique name |
| CLOCK | A unique name |
| AUTO_INSERT_SLD_NODE_ENTITY | A unique name |

For entity-specific assignments, use the "-entity" option to remove the assignment(s) from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.

Assignments removed by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments
- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

### Example

```
## Remove all the timing requirements
## Use wildcards to catch TSU_REQUIREMENT, TCO_REQUIREMENT,
## and others
remove_all_instance_assignments -name *_REQUIREMENT

## Remove all the location assignments with
## the destination bus name "timeo".
set bus_name "timeo"
remove_all_instance_assignments -name LOCATION -to $bus_name[*]
```

## remove_all_parameters

### Usage

```
remove_all_parameters [-entity <entity_name>] -name <name> [-tag <data>] [-to
<destination>]
```

### Options

```
-entity <entity_name>: Entity to which parameter belongs
```

```
-name <name>: Parameter name (string pattern is matched using Tcl string matching)
```

```
-tag <data>: Option to tag data to this assignment
```

```
-to <destination>: Destination of the parameter (string pattern is matched using Tcl
string matching)
```

### Description

Removes all matching parameters.

The "-name" option is not case sensitive. The "-to" option is case sensitive.

If the "-to" argument is specified, the function removes the parameters from the current entity. The parameters are removed from the PARAMETERS section of the entity. Otherwise, the function removes the project-wide default parameters obtained from the DEFAULT_PARAMETERS section.

This Tcl command filters the parameter data found in the Quartus II Settings File (.qsf) and removes the data based on the values specified by the "-name" and "-to" options. These options can take string patterns containing special characters from the set "*?\[]" as values. The values are matched using Tcl string matching. Note that bus names are automatically detected and do not need to be escaped. Bus names have the following format:

```
<bus name>[<bus index>] or <bus name>[*]
```

The <bus name> portion is a string of alphanumeric characters. The <bus index> portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the <bus index> is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names and can be used as follows:

```
# To match index 0 of bus "a", type:
remove_all_parameters -name * -to a[0]

# To match all indices of bus "a", type:
remove_all_parameters -name * -to a[*]
```

All other uses of square brackets must be escaped if you do not intend to use them as string patterns. For example, to match indices 0, 1, and 2 of the bus "a", type:

```
remove_all_parameters -name * -to "a[escape_brackets \[]\[0-2\][escape_brackets \]]"
```

For more information about escaping square brackets, type "escape_brackets -h".

Use the "-entity" option to remove the parameters from the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.

The parameters removed by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments
- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

### Example

```
## The following 3 examples remove project-wide,
## default parameter values
remove_all_parameters -name WIDTH
remove_all_parameters -name *ID*
remove_all_parameters -name *

## The following 3 examples remove entity-specific
## parameter values
remove_all_parameters -name inst1 -to SIZE
remove_all_parameters -name inst1 -to *IZ*
remove_all_parameters -name inst1 -to *
```

## resolve_file_path

### Usage

```
resolve_file_path <file_name>
```

### Options

```
<file_name>: Option to specify the file name
```

### Description

Returns the resolved full path of the specified file name. If the file does not exist, the original file name is returned.

The Quartus II software resolves relative paths by searching for the file in the following directories in the following order:

1) Project directory, which is the directory where the Quartus II Settings File (.qsf) is found.
2) Project database directory, which is the "db" directory found under the project directory.
3) Project library directories, which are the directories containing the user-specified libraries that are used only by the current project.
4) User library directories, which are the directories containing the user-specified libraries that are used by all Quartus II projects.
5) Quartus II library directory, which is the directory containing Quartus II libraries.

### Example

```
project_new chiptrip -overwrite

# Set one Verilog source file assignment
set_global_assignment -name VERILOG_FILE chiptrip.v

# Display the resolved full path of the Verilog
# source file assignment
set filename [get_global_assignment -name VERILOG_FILE]
set resolved_fullpath [resolve_file_path $filename]

puts "Full Path: $resolved_fullpath"

# Set more Verilog source file assignments
set_global_assignment -name VERILOG_FILE auto_max.v
set_global_assignment -name VERILOG_FILE speed_ch.v
set_global_assignment -name VERILOG_FILE tick_cnt.v
set_global_assignment -name VERILOG_FILE time_cnt.v

# Display the resolved full path of all the Verilog
# source file assignments
set file_asgns [get_all_global_assignments -name VERILOG_FILE]
foreach_in_collection file_asgn $file_asgns {

    ## Each element in the collection has the following
    ## format: {} {VERILOG_FILE} {<file_name>}

    set filename [lindex $file_asgn 2]
    set resolved_fullpath [resolve_file_path $filename]

    puts "Full Path: $resolved_fullpath"
}

project_close
```

## revision_exists

### Usage

```
revision_exists [-project <project_name>] <revision_name>
```

### Options

```
-project <project_name>: Project name
<revision_name>: Revision name
```

### Description

Checks whether the revision exists for the specified project or currently open project.

Returns 1, if the revision exists; returns 0, otherwise.

### Example

```
## Check if the specified revision exists
## in the specified project
if [revision_exists -ARG(project) chiptrip speed_ch] {
    puts "Revision exists"
} else {
    puts "Revision does not exist"
}

## Create revision for the currently open
## project if it does not exist
## Set the current revision otherwise
project_open chiptrip
if [revision_exists speed_ch] {
    set_current_revision speed_ch
} else {
    create_revision speed_ch
}
project_close
```

## set_current_revision

### Usage

```
set_current_revision [-force] <revision_name>
```

### Options

`-force: Option to open the revision and overwrite the compilation database if the database version is incompatible.`

`<revision_name>: Revision name`

### Description

Sets the specified revision name as the current revision.

In 8.1 or later versions of Quartus II software, set_current_revision gives an error when the compilation database version is not compatible with the current version of Quartus II software. You may specify the "-force" option to avoid the error and overwrite the database.

### Example

```
## Sets "auto_max" as the current revision
set_current_revision auto_max
```

## set_global_assignment

### Usage

```
set_global_assignment [-comment <comment>] [-disable] [-entity <entity_name>] -name
<name> [-remove] [-section_id <section id>] [-tag <data>] <value>
```

### Options

```
-comment <comment>: Comment
```

```
-disable: Option to disable assignment
```

```
-entity <entity_name>: Entity to which to add assignment
```

```
-name <name>: Assignment name
```

```
-remove: Option to remove assignment
```

```
-section_id <section id>: Section id
```

```
-tag <data>: Option to tag data to this assignment
```

```
<value>: Assignment value
```

### Description

Sets or removes a global assignment.

Assignments created or modified by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands (from the ::quartus::project Tcl package):

- export_assignments
- project_close (unless -dont_export_assignments is specified as an

You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands execute_flow and execute_module (from the ::quartus::flow Tcl package) call "export_assignments" before they run command-line executables.

For entity-specific assignments, use the -entity option to force the assignment to specified entity. If the -entity option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.

If the Quartus II Settings File contains a USER_LIBRARIES assignment and you call set_global_assignment to set a SEARCH_PATH or USER_LIBRARIES assignment, the existing USER_LIBRARIES assignment expands into one or more SEARCH_PATH assignments.

Note that values that begin with a dash ("-") should be enclosed in a backslash followed by a quote. In the following example, -02 is enclosed by \" at the beginning and the end.

```
set_global_assignment -name ARM_CPP_COMMAND_LINE \"-O2\"
```

### Example

```
## Specify Stratix as the family to use when compiling
set_global_assignment -name FAMILY Stratix

## If the family name has empty spaces, use quotes
set_global_assignment -name FAMILY "Stratix GX"

## or remove any empty space
set_global_assignment -name FAMILY StratixGX
```

## set_instance_assignment

### Usage

```
set_instance_assignment [-comment <comment>] [-disable] [-entity <entity_name>] [-fall]
[-from <source>] -name <name> [-remove] [-rise] [-section_id <section id>] [-tag <data>]
[-to <destination>] <value>
```

### Options

```
-comment <comment>: Comment

-disable: Option to disable assignment

-entity <entity_name>: Entity to which to add assignment

-fall: Option applies to falling edge

-from <source>: Source of assignment

-name <name>: Assignment name

-remove: Option to remove assignment

-rise: Option applies to rising edge

-section_id <section id>: Section id

-tag <data>: Option to tag data to this assignment

-to <destination>: Destination of assignment

<value>: Assignment value
```

### Description

Sets or removes an instance assignment.

Assignments created or modified by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments
- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

For entity-specific assignments, use the "-entity" option to force the assignment to specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.

### Example

```
## Specify a TSU_REQUIREMENT of 2ns from mypin to any register
set_instance_assignment -from "mypin" -to * -name TSU_REQUIREMENT 2ns

## Remove the TSU_REQUIREMENT from mypin to all registers
set_instance_assignment -from "mypin" -to * -name TSU_REQUIREMENT -remove

## Specify the entity to which the assignment is added,
## use the -entity option
## This is needed if the top-level entity name is other than
## that of the project name
## The following command generates a top_level entity
set_instance_assignment -from "mypin" -to * -entity top_level -name \
    TSU_REQUIREMENT 2ns
```

## set_io_assignment

### Usage

```
set_io_assignment [-comment <comment>] [-disable] [-io_standard <io standard>] -name
<name> [-remove] [-tag <data>] <value>
```

### Options

```
-comment <comment>: Comment

-disable: Option to disable assignment

-io_standard <io standard>: Option to specify the io standard

-name <name>: Assignment name

-remove: Option to remove assignment

-tag <data>: Option to tag data to this assignment

<value>: Assignment value
```

### Description

Sets or removes an io assignment.

Assignments created or modified by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments
- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

### Example

```
## Specify LVTTL as the IO Standard for OUTPUT_PIN_LOAD assignment
set_io_assignment 30 -name OUTPUT_PIN_LOAD -io_standard LVTTL
```

## set_location_assignment

### Usage

```
set_location_assignment [-comment <comment>] [-disable] [-remove] [-tag <data>] -to
<destination> <value>
```

### Options

```
-comment <comment>: Comment
```

```
-disable: Option to disable assignment
```

```
-remove: Option to remove assignment
```

```
-tag <data>: Option to tag data to this assignment
```

```
-to <destination>: Destination of assignment
```

```
<value>: Assignment value
```

### Description

Sets or removes a location assignment.

Assignments created or modified by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments
- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

### Example

```
set_location_assignment -to dst LOC
```

## set_parameter

### Usage

```
set_parameter [-comment <comment>] [-disable] [-entity <entity_name>] -name <name>
[-remove] [-tag <data>] [-to <destination>] <value>
```

### Options

```
-comment <comment>: Comment

-disable: Option to disable parameter

-entity <entity_name>: Entity to which to add parameter

-name <name>: Parameter name

-remove: Option to remove parameter

-tag <data>: Option to tag data to this assignment

-to <destination>: Destination of parameter

<value>: Parameter value
```

### Description

Sets or removes the specified parameter name.

The "-name" option is not case sensitive. The "-to" option is case sensitive.

The parameters created or modified by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments
- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

Use the "-entity" option to force the parameter to the specified entity. If the "-entity" option is not specified, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.

A parameter is an attribute of a megafunction, macrofunction, or certain primitives that determines the logic created or used to implement the function. The parameter information can be used to determine the actual primitives and other subdesigns needed to implement the logic of the function.

The following general guidelines apply to parameters:

- All logic options can be assigned as parameters for individual instances of megafunctions or macrofunctions. For a given logic OPTION the precedence for parameters are:

  1) Instance specific logic option settings
  2) Instance specific parameter settings
  3) Project-wide default parameter settings

- You cannot assign a value to the predefined Altera®parameter DEVICE_FAMILY, which represents the device family assigned for the project. However, you can use the parameter value in comparisons.
- The predefined Altera LPM_PIPELINE and LATENCY parameters can be assigned to an instance of a megafunction or macrofunction. However, the parameter applies only to that instance, and is not inherited by the subdesigns of that instance.

- All logic options can be assigned as parameters for individual megafunctions or macrofunctions. However, logic options cannot be assigned global, project-wide default parameter values.

### Example

```
## Set project-wide, default WIDTH parameter value
set_parameter -name WIDTH 8

## Set entity-specific SIZE parameter value
## to "my_ram" entity
set_parameter -entity my_ram -name SIZE 16

## Specify the same parameter to my_ram
## but inside "top_level" entity
set_parameter -entity top_level -to my_ram -name SIZE 16
```

## set_power_file_assignment

### Usage

```
set_power_file_assignment [-remove] [-saf_file <saf_file>] [-section_id <section_id>]
[-to <to>] [-vcd_end_time <vcd_end_time>] [-vcd_file <vcd_file>] [-vcd_start_time
<vcd_start_time>]
```

### Options

```
-remove: Option to remove assignment

-saf_file <saf_file>: SAF file name

-section_id <section_id>: Section id

-to <to>: Entity to which to apply power input file

-vcd_end_time <vcd_end_time>: End time for VCD file parsing

-vcd_file <vcd_file>: VCD file name

-vcd_start_time <vcd_start_time>: Start time for VCD file parsing
```

### Description

Sets or removes a power input file assignment. Power input file assignments are specified using multiple global assignments, and a single instance assignment as illustrated in the following example:

```
set_global_assignment -name POWER_INPUT_FILE_NAME "test.vcd" -section_id test.vcd
set_global_assignment -name POWER_INPUT_FILE_TYPE VCD -section_id test.vcd
set_global_assignment -name POWER_VCD_FILE_START_TIME "10 ns" -section_id test.vcd
set_global_assignment -name POWER_VCD_FILE_END_TIME "1000 ns" -section_id test.vcd
set_instance_assignment -name POWER_READ_INPUT_FILE test.vcd -to test_design
```

The power input file assignment serves as a wrapper for all of the above assignments. If the "-remove" setting is not set, the set_power_file_assignment will also make the following assignment to enable the use of input files:

```
set_global_assignment -name POWER_USE_INPUT_FILES ON
```

If you do not specify a "-section_id", a new section identifier is created for the input file assignment. If a "-section_id" is specified and it does not already exist, it is used as the new section identifier. If a "-section_id" is specified and it does exist, the existing input file assignments are removed and a new input file assignment is created using the given parameters and section identifier.

If an entity name given by "-to" is not specified, the input file assignment applys to the top level design entity.

If the "-remove" setting is used, the input file assignment given by the "-section_id", "-vcd_file", or "-saf_file" is removed from the project.

Assignments created or modified by using this Tcl command are saved to the Quartus II Settings File (.qsf).

### Example

```
## Specify an input SAF file applied to the top level entity
## A default section will be created
set_power_file_assignment -saf_file test.saf

## Specify an input VCD file applied to design_top|counter1
## Use the given section_id to create a new section
set_power_file_assignment -vcd_file test.vcd -to design_top|counter1 \
    -section_id test.vcd

## Update the previous input VCD file assignment to specify a
## start and end time
```

```
set_power_file_assignment -vcd_file test.vcd -to design_top|counter1 \
    -vcd_start_time 10ns -vcd_end_time 100ns -section_id test.vcd

## Remove the input SAF file assignment using the file name
set_power_file_assignment -saf_file test.saf -remove

## Remove the input VCD file assignment using the section identifier
set_power_file_assignment -section_id test.vcd -remove
```

## set_user_option

### Usage

```
set_user_option -name <name> <value>
```

### Options

```
-name <name>: User option name
```

```
<value>: User option value
```

### Description

Sets the user option value for the name specified by the "-name" option. The user option is written to the quartus2.ini file.

To get a list of all available user option names, use the "get_all_user_option_names" command.

### Example

```
## Set TALKBACK_ENABLED to "on"
set_user_option -name TALKBACK_ENABLED on
```

## test_assignment_trait

### Usage

```
test_assignment_trait -name <name> -trait <trait_name>
```

### Options

```
-name <name>: Assignment name
-trait <trait_name>: Trait name
```

### Description

Checks whether the assignment name has the specified trait. Returns 1, if the assignment name has the trait; returns 0, otherwise.

### Example

```
## Test if the assignment name is case-sensitive
if {[test_assignment_trait -name VHDL_FILE -trait CASE_SENSITIVE]} {
   puts "VHDL_FILE assignment is case-sensitive."
} else {
   puts "VHDL_FILE assignment is not case-sensitive."
}
```

## report

This package contains a set of Tcl functions for accessing and updating information in the report database.

This package is loaded by default in the following executable:

- quartus_sim

This package is available for loading in the following executables:

- quartus
- quartus_cdb
- quartus_drc
- quartus_eda
- quartus_sh
- quartus_si
- quartus_sta
- quartus_tan

This package includes the following commands:

## add_row_to_table

### Usage

```
add_row_to_table [-id <table_id>] [-name <table_name>] <row>
```

### Options

```
-id <table_id>: Id of table to update
-name <table_name>: Name of table to update
<row>: Tcl list of strings to add to table
```

### Description

Adds the list of strings to the table as a row.

Using the panel id provides faster data access than using the panel name.

Panel ids that you have cached may become outdated or invalid if the report gets unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands.

Panel names support wildcards.

The table of content portion of the UI Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the UI Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of content shows the path "Analysis & Synthesis||Summary". However, the corresponding full path used by this Tcl command is "Analysis & Synthesis||Analysis & Synthesis Summary".

### Example

```
load_package report
project_open chiptrip
load_report

# Set panel name and id
set panel {Fitter||Fitter Settings}
set id [get_report_panel_id $panel]

if {$id != -1} {
    # If panel exists, add a row to it
    add_row_to_table -id $id {{New Field} Yes No}
    # Save the changes to the report database
    save_report_database
} else {
    # Otherwise print an error message
    puts "Error: Table $panel does not exist."
}

unload_report
project_close
```

## create_report_panel

### Usage

```
create_report_panel [-folder] [-table] <panel_name>
```

### Options

```
-folder: Option to create folder
```

```
-table: Option to create table
```

```
<panel_name>: Name of the panel to create
```

### Description

Creates a new report panel with the specified name.

If -table option is specified, a table is created. If -folder option is specified, a folder is created.

The name must be the full path to the new report panel, such as "Fitter||My Table". If the specified panel is created successfully, the corresponding panel id is returned.

The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis & Synthesis||Summary". However, the corresponding full path used by this Tcl command is "Analysis & Synthesis||Analysis & Synthesis Summary".

### Example

```
load_package report
project_open chiptrip
load_report

# Set folder name and id
set folder    "My Folder"
set folder_id [get_report_panel_id $folder]

# Check if specified folder exists. If not, create it.
if {$folder_id == -1} {
    set folder_id [create_report_panel -folder $folder]
}

# Set table name and id
set table    "$folder||My Table"
set table_id [get_report_panel_id $table]

# Check if specified table exists. If so, delete it.
if {$table_id != -1} {
    delete_report_panel -id $table_id
}

# Create the specified table and get its id
set table_id [create_report_panel -table $table]

# Add Timing Analyzer Summary to the table
add_row_to_table -id $table_id {{Name} {Value}}
add_row_to_table -id $table_id {{Number of Registers} {100}}
```

```
# Save the changes to the report database
save_report_database

unload_report
project_close
```

## delete_report_panel

### Usage

```
delete_report_panel [-id <panel_id>] [-name <panel_name>]
```

### Options

```
-id <panel_id>: Id of panel to delete
-name <panel_name>: Name of panel to delete
```

### Description

Deletes the report panel with the specified id or name. The panel can either be a report table or report folder.

Using the panel id provides faster data access than using the panel name.

Panel ids that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands.

Panel names support wildcards.

The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis & Synthesis||Summary". However, the corresponding full path used by this Tcl command is "Analysis & Synthesis||Analysis & Synthesis Summary".

### Example

```
load_package report
project_open chiptrip
load_report

# Set table name and id
set table    "Fitter||My Table"
set table_id [get_report_panel_id $table]

# Delete the table if the it already exists
if {$table_id != -1} {
    delete_report_panel -id $table_id
}

# Re-create the table
create_report_panel -table $table
add_row_to_table -name $table {{Name} {Value}}
add_row_to_table -name $table {{Number of Registers} {100}}

# This time, use table name instead of table id to delete it.
delete_report_panel -name $table

# Now, delete a folder
set folder    "My Folder"
set folder_id [get_report_panel_id $folder]

# Delete it if the specified folder already exists
if {$folder_id != -1} {
    delete_report_panel -id $folder_id
}
```

```
# Save the changes to the report database
save_report_database

unload_report
project_close
```

## get_fitter_resource_usage

### Usage

```
get_fitter_resource_usage [-alm] [-alut] [-available] [-io_pin] [-lab] [-le] [-mem_bit]
[-percentage] [-reg] [-used] [-utilization]
```

### Options

```
-alm: Get total adaptive logic modules
```

```
-alut: Get total Combinational ALUTs
```

```
-available: Get available resource summary
```

```
-io_pin: Get total I/O pins
```

```
-lab: Get total logic array blocks
```

```
-le: Get total logic elements
```

```
-mem_bit: Get total memory bits
```

```
-percentage: Get used resource summary in percentage
```

```
-reg: Get total registers
```

```
-used: Get used resource summary
```

```
-utilization: Get total logic utilization
```

### Description

Gets the Fitter resource usage results.

You must use one of the following options: "-alut", "-reg", "-le", "-alm", "-lab", "-io_pin", "-mem_bit" or "-resource".

If the above option is not "-resource", you may also optionally use one of the following options: "-used", "-available" or "-percentage".

Option "-resource" takes resource name as parameter, which supports wildcards.

### Example

```
load_package report
project_open chiptrip
load_report

# Shortcut of get_fitter_resource_usage command
set cmd     get_fitter_resource_usage

# Get total registers, logic elements, and DSP block 9-bit elements.
set registers [$cmd -reg]
set les       [$cmd -le -used]
set io_pin    [$cmd -io_pin -available]
set mem_bit   [$cmd -mem_bit -percentage]
set dsp       [$cmd -@ARG(resource) "DSP blck 9*"]
puts "Registers usage: $registers"
puts "Total used logic elements: $les"
puts "Total available I/O pins: $io_pin"
puts "Total used memory bits in percentage: ${mem_bit}%"
puts "DSP block 9-bit elements: $dsp"

unload_report
project_close
```

## get_number_of_columns

### Usage

```
get_number_of_columns [-id <table_id>] [-name <table_name>]
```

### Options

```
-id <table_id>: Id of panel from which to get data
-name <table_name>: Name of panel from which to get data
```

### Description

Returns the number of columns for the specified panel.

Using the panel id provides faster data access than using the panel name.

Panel ids that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands.

Panel names support wildcards.

The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis & Synthesis||Summary". However, the corresponding full path used by this Tcl command is "Analysis & Synthesis||Analysis & Synthesis Summary".

### Example

```
## Loop through a panel row and print out all its element information

load_package report
project_open chiptrip
load_report

# Set panel name and id
set panel   {*Input Pins}
set id      [get_report_panel_id $panel]

# Get the number of columns
set col_cnt [get_number_of_columns -id $id]

# Set row name and get row index
set rname   {[Cc]lock}
set rindex  [get_report_panel_row_index -id $id $rname]
set rname   [get_report_panel_data -row $rindex -col 0 -id $id]

puts "\[Input Pins - $rname\]"
for {set i 1} {$i < $col_cnt} {incr i} {
    set    result "[get_report_panel_data -row 0 -col $i -id $id]: "
    append result [get_report_panel_data -row $rindex -col $i -id $id]
    puts   $result
}

unload_report
project_close
```

## get_number_of_rows

### Usage

```
get_number_of_rows [-id <table_id>] [-name <table_name>]
```

### Options

```
-id <table_id>: Id of panel from which to get data
-name <table_name>: Name of panel from which to get data
```

### Description

Returns the number of rows for the specified panel.

Using the panel id provides faster data access than using the panel name.

Panel ids that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands.

Panel names support wildcards.

The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis & Synthesis||Summary". However, the corresponding full path used by this Tcl command is "Analysis & Synthesis||Analysis & Synthesis Summary".

### Example

```
## Loop through a panel and print out all its row information

load_package report
project_open chiptrip
load_report

# Set panel name and id
set panel   {*Fitter Settings}
set id      [get_report_panel_id $panel]

# Get the number of rows
set row_cnt [get_number_of_rows -id $id]

for {set i 0} {$i < $row_cnt} {incr i} {
    puts [get_report_panel_row -row $i -id $id]
}

unload_report
project_close
```

## get_report_panel_column_index

### Usage

```
get_report_panel_column_index [-id <table_id>] [-name <table_name>] <col_name>
```

### Options

```
-id <table_id>: Id of panel from which to get column index
```

```
-name <table_name>: Name of panel from which to get column index
```

```
<col_name>: Column name
```

### Description

Gets the column index of the specified panel column name.

Column name refers to the specified name in the header row. The column index is a non-negative integer for an existing column. The column index is -1 if the column name is not found in the panel.

Using the column index and panel id provides faster data access than using column name and panel name.

Column indices and panel ids that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands.

Column and panel names support wildcards.

The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of content shows the path "Analysis & Synthesis||Summary". However, the corresponding full path used by this Tcl command is "Analysis & Synthesis||Analysis & Synthesis Summary".

### Example

```
load_package report
project_open chiptrip
load_report

# Set panel name and id
set panel  {*Input Pins}
set id     [get_report_panel_id $panel]

# Set column name and index
set cname  {Pin #}
set cindex [get_report_panel_column_index -id $id $cname]

# Get data out of the specified panel
set clock  [get_report_panel_data -id $id -row_name clock -col $cindex]
set enable [get_report_panel_data -id $id -row_name enable -col $cindex]

# Output results
puts "$cname of clock: $clock"
puts "$cname of enable: $enable"

unload_report
project_close
```

## get_report_panel_data

### Usage

```
get_report_panel_data [-col <column>] [-col_name <column_name>] [-id <table_id>] [-name
<table_name>] [-row <row>] [-row_name <row_name>]
```

### Options

```
-col <column>: column (or X) coordinate

-col_name <column_name>: column (or X) name

-id <table_id>: id of panel from which to get data

-name <table_name>: Name of panel from which to get data

-row <row>: row (or Y) coordinate

-row_name <row_name>: row (or Y) name
```

### Description

Returns non-empty data for the specified row and column of the specified panel. If the data is empty or if the row, column, or panel do not exist, an error will be generated. To properly handle the error, make sure to catch the result as in the following example:

```
if [catch {set data [get_report_panel_data ...]} result] {
  puts "No data found"
} else {
  puts "Got $data"
}
```

Using the panel id and row and column indices provides faster data access than using panel, row, and column names.

Panel ids and row and column indices that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands.

Panel, row, and panel names support wildcards.

The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis & Synthesis||Summary". However, the corresponding full path used by this Tcl command is "Analysis & Synthesis||Analysis & Synthesis Summary".

### Example

```
load_package report
project_open chiptrip
load_report

# Set panel name and id
set panel {Flow Settings}
set id    [get_report_panel_id $panel]
# Set row name
set rname {Revision Name}

# Get row 2 - column 1 data
get_report_panel_data -row 2 -col 1 -id $id
# Get row {Revision Name} - column 1 data
get_report_panel_data -row_name $rname -col 1 -id $id
```

```
# Get row {Revision Name} - column {Setting} data
get_report_panel_data -row_name $rname -col_name Setting -id $id
# If unsure the case of a row or column name, use glob-style pattern
get_report_panel_data -row_name {[Rr]evision*} -col_name {[Ss]etting} -id \
    $id

unload_report
project_close
```

## get_report_panel_id

### Usage

```
get_report_panel_id <name>
```

### Options

```
<name>: Name of panel for which to get id
```

### Description

Gets the id of a panel with the specified name.

The panel id is a non-negative integer for an existing panel. If the specified panel cannot be found, the id is -1.

Using the panel id provides faster data access than using the panel name.

You can use the "get_report_panel_id" command to check for the existence of a panel.(Refer to the example under the "get_report_panel_id" command.)

Panel ids that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands.

Panel names support wildcards.

The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis & Synthesis||Summary". However, the corresponding full path used by this Tcl command is "Analysis & Synthesis||Analysis & Synthesis Summary".

### Example

```
load_package report
project_open chiptrip
load_report

# Set panel name
set panel {Fitter||Fitter Settings}
# Get the panel id
set id [get_report_panel_id $panel]

if {$id != -1} {
    # If panel exists, add a row to it
    add_row_to_table -id $id {{New Field} Yes No}
    # Save the changes to the report database
    save_report_database
} else {
    # Otherwise print an error message
    puts "Error: Table $panel does not exist."
}

unload_report
project_close
```

## get_report_panel_names

### Usage

```
get_report_panel_names
```

### Options

None

### Description

Returns a list of panel names for the current report.

The table of contents portion of the Compilation Report window shows short panel names for better readability. However, each panel name returned by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis & Synthesis||Summary". However, the corresponding full path returned by this Tcl command is "Analysis & Synthesis||Analysis & Synthesis Summary".

### Example

```
## Load report database and write Timing Analyzer Summary
## panel to file in HTML format

load_package report
project_open chiptrip
load_report

# Set panel name
set fmax_panel "Timing Analyzer Summary"

# Iterate through all the accessable panels
foreach panel [get_report_panel_names] {
    # If find the panel '*Timing Analyzer Summary',
    # write its content to file fmax.htm
    if {[string match "*$fmax_panel" $panel] == 1} {
        write_report_panel -file fmax.htm -html $panel
        break
    }
}

unload_report
project_close
```

## get_report_panel_row

### Usage

```
get_report_panel_row [-id <table_id>] [-name <table_name>] [-row <row>] [-row_name
<row_name>]
```

### Options

```
-id <table_id>: Id of panel from which to get data
-name <table_name>: Name of panel from which to get data
-row <row>: Row (or Y) coordinate
-row_name <row_name>: Row (or Y) name
```

### Description

Reports data of the specified panel row.

Using the panel id and row index provides faster data access than using panel name and row name, respectively.

Panel ids and row indices that you have cached may become outdated or invalid if the report gets unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands.

Panel and row names support wildcards.

The table of content portion of the UI Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the UI Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of content shows the path "Analysis & Synthesis||Summary". However, the corresponding full path used by this Tcl command is "Analysis & Synthesis||Analysis & Synthesis Summary".

### Example

```
load_package report
project_open chiptrip
load_report

# Set panel name and id
set panel {Flow Settings}
set id    [get_report_panel_id $panel]
# Set row name
set rname {Revision Name}

# Get row {Revision Name} data (use panel id)
get_report_panel_row -row_name $rname -id $id
# Get row 2 data (use panel id)
get_report_panel_row -row 2 -id $id
# Get row 3 data (use panel name)
get_report_panel_row -row 3 -name $panel

## Get the last row data
set row_cnt     [get_number_of_rows -id $id]
set last_rindex [expr $row_cnt - 1]
get_report_panel_row -row $last_rindex -id $id

unload_report
project_close
```

## get_report_panel_row_index

### Usage

```
get_report_panel_row_index [-id <table_id>] [-name <table_name>] <row_name>
```

### Options

```
-id <table_id>: Id of panel from which to get row index
-name <table_name>: Name of panel from which to get row index
<row_name>: Row name
```

### Description

Gets the row index of the specified panel row name.

Row name refers to the first element content of the specified row. The row index is a non-negative integer for an existing row. Row index is -1 if the row name is not found in the panel.

Using the row index and panel id provides faster data access than using row name and panel name.

Row indices and panel ids that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands.

Row and panel names support wildcards.

The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis & Synthesis||Summary". However, the corresponding full path used by this Tcl command is "Analysis & Synthesis||Analysis & Synthesis Summary".

### Example

```
load_package report
project_open chiptrip
load_report

# Set panel name and id
set panel   {*Input Pins}
set id      [get_report_panel_id $panel]

# Set row name and index
set rname   {[Cc]lock}
set rindex  [get_report_panel_row_index -id $id $rname]

# Get data out of the specified panel
set pc_str  [get_report_panel_data -id $id -row 0 -col 1]
set pin_cnt [get_report_panel_data -id $id -row $rindex -col 1]
set iob_str [get_report_panel_data -id $id -row 0 -col 2]
set io_bank [get_report_panel_data -id $id -row $rindex -col 2]

# Output results
puts "$pc_str: $pin_cnt"
puts "$iob_str: $io_bank"

unload_report
project_close
```

## get_timing_analysis_summary_results

### Usage

```
get_timing_analysis_summary_results [-actual] [-clock_hold <clock_hold>] [-clock_setup
<clock_setup>] [-min_tco] [-min_tpd] [-required] [-slack] [-tco] [-th] [-tpd] [-tsu]
```

### Options

```
-actual: Actual time
-clock_hold <clock_hold>: Clock hold name
-clock_setup <clock_setup>: Clock setup name
-min_tco: Minimum clock to output delay
-min_tpd: Minimum propagation delay
-required: Required time
-slack: Slack
-tco: Clock to output delay
-th: Hold time
-tpd: Propagation delay
-tsu: Setup time
```

### Description

Gets the Timing Analysis Summary results. This command uses the "get_report_panel_data" command to access the Timing Analysis Summary panel. It provides easy access to the panel data without the need to know the names of the corresponding row and column.

You must use one of the following options: "-tsu", "-tco", "-tpd", "-th", "-min_tco", "-min_tpd", "-clock_setup", or "-clock_hold".

You also must use one of the following options: "-slack", "-required" or "-actual".

Clock names support wildcards.

### Example

```
load_package report
project_open chiptrip
load_report

# Shortcut of get_timing_analysis_summary_results command
set cmd     get_timing_analysis_summary_results

# Get actual tsu, clock slack, and required tco
set act_tsu [$cmd -tsu -actual]
set clock   [$cmd -clock_setup {[Cc]lock} -actual]
set req_tco [$cmd -tco -required]
puts "Actual tsu: $act_tsu"
puts "Actual clock setup: $clock"
puts "Required tco: $req_tco"

unload_report
project_close
```

## load_report

### Usage

```
load_report [-simulator]
```

### Options

```
-simulator: Option to load the Simulation Report. If this option isn't specified, the
Compilation Report is loaded instead.
```

### Description

By default, loads the Compilation Report for the current revision or the specified revision name. If the -simulator option is specified, loads the Simulation Report instead.

After the report is loaded or reloaded, the cached panel ids, and row and column indices may become outdated or invalid. Altera recommends that you update them before using them.

### Example

```
# Load report package
load_package report
# Open chiptrip project
project_open chiptrip
# Load the current revision report
load_report

# Set panel name and id
set panel {Fitter||Fitter Summary}
set id    [get_report_panel_id $panel]

# Get total logic elements
set rname  {Total [Ll]ogic [Ee]lements}
set rindex [get_report_panel_row_index -id $id $rname]
set rname  [get_report_panel_data -id $id -row $rindex -col 0]
set data   [get_report_panel_data -id $id -row $rindex -col 1]
puts "$rname: $data"

# Get total pins
set rname  {Total [Pp]ins}
set rindex [get_report_panel_row_index -id $id $rname]
set rname  [get_report_panel_data -id $id -row $rindex -col 0]
set data   [get_report_panel_data -id $id -row $rindex -col 1]
puts "$rname: $data"

# Unload the report
unload_report
# Close the project
project_close
```

## read_xml_report

### Usage

```
read_xml_report <filename>
```

### Options

```
<filename>: Name of XML Report Database File from which to read
```

### Description

Creates the current Compilation Report from the specified XML Report Database File (.xml).

### Example

```
# Set project name
set project_name "chiptrip"

load_package report
project_open $project_name

# Read XML Report Database File (.xml)
read_xml_report $project_name.xml

load_report

# Get all the panel names
puts {All Report Panel Names:}
puts [get_report_panel_names]

unload_report
project_close
```

## save_report_database

### Usage

```
save_report_database
```

### Options

None

### Description

Saves the report database, including any new report panel.

### Example

```
load_package report
project_open chiptrip
load_report

# Set panel name and id
set panel {Fitter||Fitter Settings}
set id    [get_report_panel_id $panel]

# If panel exists, add a row to it. Otherwise, print an error message.
if {$id != -1} {
    add_row_to_table -id $id {{New Field} Yes No}
    # Save the changes to the report database
    save_report_database
} else {
    puts "Error: Table $panel does not exist."
}

unload_report
project_close
```

## unload_report

### Usage

```
unload_report
```

### Options

None

### Description

Unloads the report for the current revision or the specified revision name.

After the report is loaded or reloaded, the cached panel ids, and row and column indices, may become outdated or invalid. Altera recommends that you update them before using them.

### Example

```
# Load report package
load_package report
# Open chiptrip project
project_open chiptrip
# Load the current revision report
load_report

# Set panel name and id
set panel {Fitter||Fitter Summary}
set id    [get_report_panel_id $panel]

# Get total logic elements
set rname  {Total [Ll]ogic [Ee]lements}
set rindex [get_report_panel_row_index -id $id $rname]
set rname  [get_report_panel_data -id $id -row $rindex -col 0]
set data   [get_report_panel_data -id $id -row $rindex -col 1]
puts "$rname: $data"

# Get total pins
set rname  {Total [Pp]ins}
set rindex [get_report_panel_row_index -id $id $rname]
set rname  [get_report_panel_data -id $id -row $rindex -col 0]
set data   [get_report_panel_data -id $id -row $rindex -col 1]
puts "$rname: $data"

# Unload the report
unload_report
# Close the project
project_close
```

## write_report_panel

### Usage

```
write_report_panel -file <output file name> [-html] [-id <table_id>] [-name <table_name>]
[-xml] <name>
```

### Options

```
-file <output file name>: Name of output file to be generated
-html: Option to generate output file in HTML format
-id <table_id>: id of panel from which to get data
-name <table_name>: Name of panel from which to get data
-xml: Option to generate output file in XML format
<name>: Name of panel from which to get data
```

### Description

Writes data from the specified report panel to the specified output file. If the "-html" option is specified, the output file is generated in HTML format. If the "-xml" option is specified, the output file is generated in XML format. Otherwise, the output file is generated in ASCII format.

Using the panel id provides faster data access than using the panel name.

Panel ids that you have cached may become outdated or invalid if the report is unloaded or reloaded. This error occurs after compilation or with calls to the "project_close", "unload_report", and "load_report" commands.

Panel names support wildcards.

The table of contents portion of the Compilation Report window shows short panel names for better readability. However, the panel name used by this command is the full panel name as shown in the right-hand side frame of the Compilation Report window or the .rpt file of the corresponding command-line executable. For example, the table of contents shows the path "Analysis & Synthesis||Summary". However, the corresponding full path used by this Tcl command is "Analysis & Synthesis||Analysis & Synthesis Summary".

### Example

```
## Load report database and write Timing Analyzer Summary
## panel to file in HTML format

load_package report
project_open chiptrip
load_report

# Set panel name and id
set panel "*Timing Analyzer Summary"
set id    [get_report_panel_id $panel]

# If the specified panel exists, write it to
# fmax.htm and fmax.xml.
# Otherwise, print out an error message
if {$id != -1} {
    write_report_panel -file fmax.htm -html -id $id
    write_report_panel -file fmax.xml -xml -id $id
} else {
    puts "Error: report panel could not be found."
}

unload_report
project_close
```

## write_xml_report

### Usage

```
write_xml_report <filename>
```

### Options

```
<filename>: Name of XML Report Database File to which to write
```

### Description

Writes the current Compilation Report or Simulation Report to the specified XML Report Database File (.xml).

### Example

```
# Load the ::quartus::flow and ::quartus::report packages
load_package flow
load_package report

# Create new project
set project_name "chiptrip"
project_new $project_name -overwrite

# Run quartus_map
execute_module -tool map

## Create XML Report Database File (.xml)
load_report
file delete -force $project_name.xml
puts [write_xml_report $project_name.xml]
unload_report

# Close project
project_close
```

## 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 includes the following commands:

## 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 regisers in the design.

### Example

```
project_open chiptrip
create_timing_netlist
foreach_in_collection reg [all_registers] {
    puts [get_register_info -name $reg]
}
report_timig -from [all_regisers] -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_clockcommand 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 multipler 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 hierarachical 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 uisng 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 hierarachical 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 calculted 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.

### Example

## 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 includes the following commands:

## 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 HCII_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_timimg_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_timimg_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
```

## 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.

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>] [-from <names>] [-include <Tcl list>] [-to <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
```

```
-from <names>: Valid sources (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
```

```
-to <names>: Valid destinations (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 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
```

## simulator

This package contains the set of Tcl functions for performing simulation.

In line with the default usage of the stand-alone Simulator executable quartus_sim.exe, the old Simulator TCL APIs are no longer supported.

| **Available Tcl APIs** | **Old Tcl APIs (obsolete)** |
|---|---|
| initialize_simulation | sim initialize, testbench_mode |
| run_simulation | sim run |
| get_simulation_value | sim get_value |
| force_simulation_value | sim force_value |
| release_simulation_value | sim release_value |
| get_simulation_time | sim get_time |
| get_simulation_memory_info | sim get_memory_depth, get_memory_width |
| read_from_simulation_memory | sim read_from_memory |
| write_to_simulation_memory | sim write_to_memory |
| fast_write_to_simulation_memory | sim fast_write_to_memory |
| create_simulation_breakpoint | sim create_breakpoint |
| enable_simulation_breakpoint | sim enable_breakpoint |
| disable_simulation_breakpoint | sim disable_breakpoint |
| delete_simulation_breakpoint | sim delete_breakpoint |
| group_simulation_signal | None (New API) |
| set_simulation_clock | None (New API) |
| convert_vector | None (New API) |
| compare_vector | None (New API) |
| partition_vector | None (New API) |

This package is loaded by default in the following executable:

- quartus_sim

This package includes the following commands:

## compare_vector

### Usage

```
compare_vector [-all_signals <on | off>] [-begin <time with unit>] [-compare_clocks
<clock name, period, offset, duty cycle, ON | OFF>] [-compare_rule_value_0
<0,1,X,L,H,W,Z,U and/or DC>] [-compare_rule_value_1 <0,1,X,L,H,W,Z,U and/or DC>]
[-compare_rule_value_dc <0,1,X,L,H,W,Z,U and/or DC>] [-compare_rule_value_h
<0,1,X,L,H,W,Z,U and/or DC>] [-compare_rule_value_l <0,1,X,L,H,W,Z,U and/or DC>]
[-compare_rule_value_u <0,1,X,L,H,W,Z,U and/or DC>] [-compare_rule_value_w
<0,1,X,L,H,W,Z,U and/or DC>] [-compare_rule_value_x <0,1,X,L,H,W,Z,U and/or DC>]
[-compare_rule_value_z <0,1,X,L,H,W,Z,U and/or DC>] [-default_tolerance <time with
unit>] [-end <time with unit>] [-expected <vector file name>] [-file <vector file name>]
[-max_mismatches <positive number>] [-signals <signal name, ON | OFF, tolerance |
<<default>>, ON | OFF>] [-trigger_mode <INPUT_EDGE | ALL_EDGE | SELECTED_EDGE>]
```

### Options

```
-all_signals <on | off>: Option to compare or not compare all signals

-begin <time with unit>: Start time of waveform comparison

-compare_clocks <clock name, period, offset, duty cycle, ON | OFF>: List of comparison
clock signals to be used to trigger waveform comparison, their clock period, offset, and
duty cycle settings and waveform comparison trigger settings

-compare_rule_value_0 <0,1,X,L,H,W,Z,U and/or DC>: Option to specify the waveform
comparison rule setting for signal value 0. Example: "0,L,DC"

-compare_rule_value_1 <0,1,X,L,H,W,Z,U and/or DC>: Option to specify the waveform
comparison rule setting for signal value 1. Example: "1,H,DC"

-compare_rule_value_dc <0,1,X,L,H,W,Z,U and/or DC>: Option to specify the waveform
comparison rule setting for signal value DC. Example: "0,1,X,L,H,W,Z,U,DC"

-compare_rule_value_h <0,1,X,L,H,W,Z,U and/or DC>: Option to specify the waveform
comparison rule setting for signal value H. Example: "1,H,DC"

-compare_rule_value_l <0,1,X,L,H,W,Z,U and/or DC>: Option to specify the waveform
comparison rule setting for signal value L. Example: "0,L,DC"

-compare_rule_value_u <0,1,X,L,H,W,Z,U and/or DC>: Option to specify the waveform
comparison rule setting for signal value U. Example: "X,W,Z,U,DC"

-compare_rule_value_w <0,1,X,L,H,W,Z,U and/or DC>: Option to specify the waveform
comparison rule setting for signal value W. Example: "X,W,U,DC"

-compare_rule_value_x <0,1,X,L,H,W,Z,U and/or DC>: Option to specify the waveform
comparison rule setting for signal value X. Example: "X,W,U,DC"

-compare_rule_value_z <0,1,X,L,H,W,Z,U and/or DC>: Option to specify the waveform
comparison rule setting for signal value Z. Example: "Z,U,DC"

-default_tolerance <time with unit>: Default comparison timing tolerance for all signals

-end <time with unit>: End time of waveform comparison

-expected <vector file name>: File name of the expected vector file

-file <vector file name>: File name of the current vector file

-max_mismatches <positive number>: Maximum mismatches reported in a waveform comparison

-signals <signal name, ON | OFF, tolerance | <<default>>, ON | OFF>: List of signals to
be compared/not compared, their tolerance settings, and waveform comparison trigger
settings

-trigger_mode <INPUT_EDGE | ALL_EDGE | SELECTED_EDGE>: Waveform comparison trigger
mode--trigger comparison based on transition edges of all input signals only, all
signals, or specified signals only
```

## Description

Compare two simulation waveform vectors or two waveform files.

Note:

1. The "-file" and "-expected" options are optional for waveform comparisons performed in Quartus II simulation. For comparisons performed outside of the simulation flow, these 2 options are required.
2. Waveform comparisons are performed from 0ns to file end time in the expected waveform if neither the "-begin" or "-end" option is specified.
3. To determine comparison scope, either the "-signals" or the "-all_signals" option must be specified. If both options are specified, the "-signals" option has higher precedence than the "-all_signals" option.

Examples 1 - 15 show different ways of manipulating the compare_vector command. Refer to example 16 to see how this command is used in the simulation flow.

## Example

```
Example 1
---------
# Compare db/chiptrip.sim.vwf against expected results in chiptrip.vwf
# from
# 10ns to 50ns. Any mismatches found beyond the time frame are ignored.
# All signals are compared.
compare_vector -file db/chiptrip.sim.vwf -expected chiptrip.vwf -begin \
    10ns -end 50ns -all_signals on

Example 2
---------
# Compare db/chiptrip.sim.vwf against expected results in chiptrip.vwf.
# Only signal "dir" and "ticket" are compared. Mismatches of signal "dir"
# which come within 0.02ns earlier or later than the expected results are
# ignored.
compare_vector -file db/chiptrip.sim.vwf -expected chiptrip.vwf -signals \
    {dir, ON, 0.02ns} {ticket, ON}

Example 3
---------
# Compare db/chiptrip.sim.vwf against expected results in chiptrip.vwf.
# Signal settings for both parent and child exist, but parent settings
# are honored. Only "dir" and its children are compared. Mismatches
# of signal "dir" and its children signals which come within 0.01ns
# earlier/later than the expected results are ignored.
compare_vector -file db/chiptrip.sim.vwf -expected chiptrip.vwf -signals \
    {dir[1], ON, 0.02ns} {dir, ON, 0.01ns}

Example 4
---------
# Compare db/chiptrip.sim.vwf against expected results in chiptrip.vwf.
# All signals will be compared. Comparison stops when 50 mismatches are
# identified and reported.
compare_vector -file db/chiptrip.sim.vwf -expected chiptrip.vwf \
    -max_mismatches 50 -all_signals on

Example 5
---------
# Compare db/chiptrip.sim.vwf against expected results in chiptrip.vwf.
# All signals are compared. Any mismatches which come <=0.02ns earlier/
# later than the expected results are ignored.
compare_vector -file db/chiptrip.sim.vwf -expected chiptrip.vwf \
    -default_tolerance 0.02ns -all_signals on

Example 6
```

```
---------
# Compare db/chiptrip.sim.vwf against expected results in chiptrip.vwf.
# Only "dir" and "ticket" are compared. Default compare tolerance of
# 0.02ns applies to "dir". Signal "ticket" has its own tolerance settings
# as 0.01ns.
compare_vector -file db/chiptrip.sim.vwf -expected chiptrip.vwf \
    -default_tolerance 0.02ns -signals {dir, ON} {ticket, ON; 0.01ns}

Example 7
---------
# Compare db/chiptrip.sim.vwf against expected results in chiptrip.vwf.
# All but signal "dir" are compared. Default compare tolerance of
# 0.02ns applies to all compared signal except "ticket". Signal "ticket"
# has its own tolerance settings set as 0.01ns.
compare_vector -file db/chiptrip.sim.vwf -expected chiptrip.vwf \
    -default_tolerance 0.02ns -all_signals on -signals {dir; OFF} {ticket; \
    ON; 0.01ns}

Example 8
---------
# Compare db/chiptrip.sim.vwf against expected results in chiptrip.vwf.
# All but signal "dir" are compared. Default compare tolerance of
# 0.02ns applies to all compared signal except "ticket". Signal "ticket"
# has its own tolerance settings set as 0.01ns. Perform waveform
# comparison
# on every input transition edges.
compare_vector -file db/chiptrip.sim.vwf -expected chiptrip.vwf \
    -default_tolerance 0.02ns -trigger_mode INPUT_EDGE -all_signals on \
    -signals {dir; OFF} {ticket; ON; 0.01ns}

Example 9
---------
# Compare db/chiptrip.sim.vwf against expected results in chiptrip.vwf.
# All but signal "dir" are compared. Default compare tolerance of
# 0.02ns applies to all compared signal except "ticket". Signal "ticket"
# has its own tolerance settings set as 0.01ns. Perform waveform
# comparison
# based on transition edges from selected signal "ticket".
compare_vector -file db/chiptrip.sim.vwf -expected chiptrip.vwf \
    -default_tolerance 0.02ns -trigger_mode SELECTED_EDGE -all_signals on \
    -signals {dir; OFF} {ticket; ON; 0.01ns; ON}

Example 10
----------
# Compare db/chiptrip.sim.vwf against expected results in chiptrip.vwf.
# All but signal "dir" are compared. Default compare tolerance of
# 0.02ns applies to all compared signal except "ticket". Signal "ticket"
# has its own tolerance settings set as 0.01ns. Perform waveform
# comparison
# based on transition edges from the specified waveform comparison clock
# signal "vc1".
compare_vector -file db/chiptrip.sim.vwf -expected chiptrip.vwf \
    -default_tolerance 0.02ns -trigger_mode SELECTED_EDGE -all_signals on \
    -signals {dir; OFF} {ticket; ON; 0.01ns; OFF} -compare_clocks {vc1; 10ns; \
    2ns; 50; ON}

Example 11
----------
# Compare db/chiptrip.sim.vwf against expected results in chiptrip.vwf.
# Only signal "ticket" is compared with tolerance settings as 0.01ns.
compare_vector -file db/chiptrip.sim.vwf -expected chiptrip.vwf \
    -all_signals off -signals {ticket; ON; 0.01ns}

Example 12
```

```
----------
# Compare db/chiptrip.sim.vwf against expected results in chiptrip.vwf.
# Signal "ticket" are compared with tolerance settings as 0.01ns.
# Signal "dir" will be compared using the default tolerance value
# that is only available during execution of the comparison process.
compare_vector -file db/chiptrip.sim.vwf -expected chiptrip.vwf \
    -all_signals off -signals {ticket; ON; 0.01ns} {dir; ON; <<default>>}

Example 13
----------
# Compare db/chiptrip.sim.vwf against expected results in chiptrip.vwf.
# Signal "ticket" is compared with tolerance settings as 0.01ns.
# Signal "dir" is compared using the default tolerance value that is
# only available during execution of the comparison process.
compare_vector -file db/chiptrip.sim.vwf -expected chiptrip.vwf \
    -all_signals off -signals {ticket; ON; 0.01ns} {dir; ON; ; OFF}

Example 14
----------
# Compare db/chiptrip.sim.vwf against expected results in chiptrip.vwf.
# Default compare tolerance of 0.02ns applies to signal "dir".
# Signal "ticket" is compared with tolerance settings as 0.01ns.
compare_vector -file db/chiptrip.sim.vwf -expected chiptrip.vwf \
    -default_tolerance 0.02ns -all_signals off -signals {ticket; ON; 0.01ns} \
    {dir; ON; ; OFF}

Example 15
----------
# Compare db/chiptrip.sim.vwf against expected results in chiptrip.vwf.
# Change comparison rule for signal value 1 - match 1 in expected results
# with values 1 and DC in compared vector file.
# Signal "ticket" is compared with tolerance settings as 0.01ns.
compare_vector -file db/chiptrip.sim.vwf -expected chiptrip.vwf \
    -all_signals off -signals {ticket; ON; 0.01ns} -compare_rule_value_1 {1, \
    DC}


# Specify waveform compare settings in simulation flow
Example 16
----------
project open chiptrip

# Initialize simulator and turn on check_outputs option
initialize_simulation -check_outputs on

# Compare simulation results against source vector, from 20ns to 80ns.
# Only signal "clock" and "ticket" are involved in comparison
compare_vector -begin 20ns -end 80ns -signals {clock; ON} {ticket; ON}

# Run simulation and waveform comparison will be performed upon
# simulation completion
run_simulation
```

## convert_vector

### Usage

```
convert_vector -file <file name> -format <VWF | TBL | CVWF | VCD>
```

### Options

```
-file <file name>: File name of the vector file to be converted
```

```
-format <VWF | TBL | CVWF | VCD>: Format of the vector file to be converted to
```

### Description

Convert vector file to another format (VWF, TBL, CVWF or VCD). The original vector file remains unchanged. New vector file in <format> is generated and located in the same directory as the original vector file.

### Example

```
Example 1
---------
# Full vector file path is not given, assume it is located at
# current working directory
convert_vector -file tctin4s.vwf -format tbl
# Upon conversion, tctin4s.vwf remains unchanged, tctin4s.tbl
# will be generated at current working directory

Example 2
---------
# Full vector file path is given
convert_vector -file d:/quartus/designs/chiptrip/chiptrip.vwf -format tbl
# chiptrip.tbl will be generated at d:/quartus/designs/chiptrip

Example 3
---------
# Convert VWF into compressed version of CVWF
convert_vector -file d:/quartus/designs/chiptrip/chiptrip.vwf -format \
    cvwf
# chiptrip.cvwf will be generated at d:/quartus/designs/chiptrip

Example 4
---------
# Convert CVWF into text version of VWF
convert_vector -file d:/quartus/designs/chiptrip/chiptrip.cvwf -format \
    vwf
# chiptrip.vwf will be generated at d:/quartus/designs/chiptrip

Example 5
---------
# Convert CVWF into VCD
convert_vector -file d:/quartus/designs/chiptrip/chiptrip.cvwf -format \
    vcd
# chiptrip.vcd will be generated at d:/quartus/designs/chiptrip
```

## create_simulation_breakpoint

### Usage

```
create_simulation_breakpoint -action <Give Warning | Give Info | Give Error> -breakpoint
<breakpoint_name> -equation <equation> [-user_message <message_text>]
```

### Options

`-action <Give Warning | Give Info | Give Error>: Action to be done at the breakpoint`

`-breakpoint <breakpoint_name>: Name of the breakpoint to be created`

`-equation <equation>: Equation used to trigger the breakpoint`

`-user_message <message_text>: Optional user-customized message to be displayed when breakpoint is triggered`

### Description

Creates a breakpoint with the specified equation and action.

### Example

```
project_open one_wire

initialize_simulation

# Create breakpoint bp1 which gives warning when it reaches 10ns
create_simulation_breakpoint -breakpoint bp1 equation "TIME_EQ(10ns)" \
    -action "GIVE_WARNING"

# Create breakpoint bp2 which gives info when in_br has 0 in value
create_simulation_breakpoint -breakpoint bp2 -equation \
    "EQUAL_TO(HPATH(|one_wire|in_br),BUS(0)))" -action "GIVE_INFO"

# Create breakpoint bp3 which gives error when value of in_br and out_br
# is 1
create_simulation_breakpoint -breakpoint bp3 -equation \
    "EQUAL_TO(HPATH(|one_wire|out_br),BUS(1)),EQUAL_TO(HPATH(|one_wire|in_br), \
    BUS(1))" -action "GIVE_ERROR"

# Before running simulation, disable bp2
# Therefore it shouldn't break at time=0ns when in_br=0
disable_simulation_breakpoint -breakpoint bp2

# When running simulation for the first 50ns,
# expect to see bp1 breaks at time=10ns
run_simulation -time 50ns

delete_simulation_breakpoint -breakpoint bp1

enable_simulation_breakpoint -breakpoint bp2

# Since bp2 is enabled, expect to see bp2 breaks at time=100ns
# when in_br becomes 0 again
run_simulation -time 60ns

# Run simulation until the end of completion
run_simulation

project_close
```

## delete_all_simulation_breakpoint

### Usage

```
delete_all_simulation_breakpoint
```

### Options

None

### Description

Deletes all breakpoints.

### Example

```
project_open one_wire

initialize_simulation

# Create breakpoint bp1 which will give warning message when simulation
# time reaches 10ns
create_simulation_breakpoint -breakpoint bp1 -equation "TIME_EQ(10ns)" \
    -action "GIVE_WARNING"

# Create breakpoint bp2 which will give info message when time reaches
# 100ns
create_simulation_breakpoint -breakpoint bp2 -equation "TIME_EQ(100ns)" \
    -action "GIVE_INFO"

# Create breakpoint bp3 which will give error message when time reaches
# 200ns
create_simulation_breakpoint -breakpoint bp3 -equation "TIME_EQ(200ns)" \
    -action "GIVE_ERROR"

# All breakpoints will be deleted
delete_all_simulation_breakpoint

# Run simulation to the end. There should not be any breaks since all
# breakpoints have been deleted
run_simulation

project_close
```

## delete_simulation_breakpoint

### Usage

```
delete_simulation_breakpoint -breakpoint <breakpoint_name>
```

### Options

```
-breakpoint <breakpoint_name>: Option to delete a specified breakpoint
```

### Description

Deletes a breakpoint with the specified name.

### Example

```
project_open one_wire

initialize_simulation

# Create breakpoint bp1 which will give warning message when simulation
# time reaches 10ns
create_simulation_breakpoint -breakpoint bp1 -equation "TIME_EQ(10ns)" \
    -action "GIVE_WARNING"

# Create breakpoint bp2 which will give info message when in_br has 0 in
# value
create_simulation_breakpoint -breakpoint bp2 -equation \
    "EQUAL_TO(HPATH(|one_wire|in_br),BUS(0)))" -action "GIVE_INFO"

# Create breakpoint bp3 which will give error message when both in_br and
# out_br have 1 in value
create_simulation_breakpoint -breakpoint bp3 -equation \
    "EQUAL_TO(HPATH(|one_wire|out_br),BUS(1)),EQUAL_TO(HPATH(|one_wire|in_br), \
    BUS(1))" -action "GIVE_ERROR"

# Before running simulation, disable bp2
# Therefore it shouldn't break at time=0ns when in_br=0
disable_simulation_breakpoint -breakpoint bp2

# When running simulation for the first 50ns,
# expect to see bp1 breaks at time=10ns
run_simulation -time 50ns

delete_simulation_breakpoint -breakpoint bp1

enable_simulation_breakpoint -breakpoint bp2

# Since bp2 is enabled, expect to see bp2 breaks at time=100ns
# when in_br becomes 0 again
run_simulation -time 60ns

# Run simulation until the end of completion
run_simulation

project_close
```

## disable_all_simulation_breakpoint

### Usage

```
disable_all_simulation_breakpoint
```

### Options

None

### Description

Disables all breakpoints.

### Example

```
project_open one_wire

initialize_simulation

# Create breakpoint bp1 which will give warning message when simulation
# time reaches 10ns
create_simulation_breakpoint -breakpoint bp1 -equation "TIME_EQ(10ns)" \
    -action "GIVE_WARNING"

# Create breakpoint bp2 which will give info message when time reaches
# 100ns
create_simulation_breakpoint -breakpoint bp2 -equation "TIME_EQ(100ns)" \
    -action "GIVE_INFO"

# Create breakpoint bp3 which will give error message when time reaches
# 200ns
create_simulation_breakpoint -breakpoint bp3 -equation "TIME_EQ(200ns)" \
    -action "GIVE_ERROR"

# All breakpoints will be disabled
disable_all_simulation_breakpoint

# Run simulation. It shouldn't break at time=10ns since bp1 has been
# disabled
run_simulation -time 50ns

enable_all_simulation_breakpoint

# Run simulation to the end. Expect bp2 and bp3 to break at 100ns and
# 200ns since all breakpoints have been enabled
run_simulation

project_close
```

## disable_simulation_breakpoint

### Usage

```
disable_simulation_breakpoint -breakpoint <breakpoint_name>
```

### Options

```
-breakpoint <breakpoint_name>: Option to disable a specified breakpoint
```

### Description

Disables a breakpoint with the specified name.

### Example

```
project_open one_wire

initialize_simulation

# Create breakpoint bp1 which will give warning message when simulation
# time reaches 10ns
create_simulation_breakpoint -breakpoint bp1 -equation "TIME_EQ(10ns)" \
    -action "GIVE_WARNING"

# Create breakpoint bp2 which will give info message when in_br has 0 in
# value
create_simulation_breakpoint -breakpoint bp2 -equation \
    "EQUAL_TO(HPATH(|one_wire|in_br),BUS(0)))" -action "GIVE_INFO"

# Create breakpoint bp3 which will give error message when both in_br and
# out_br have 1 in value
create_simulation_breakpoint -breakpoint bp3 -equation \
    "EQUAL_TO(HPATH(|one_wire|out_br),BUS(1)),EQUAL_TO(HPATH(|one_wire|in_br), \
    BUS(1))" -action "GIVE_ERROR"

# Before running simulation, disable bp2
# Therefore it shouldn't break at time=0ns when in_br=0
disable_simulation_breakpoint -breakpoint bp2

# When running simulation for the first 50ns,
# expect to see bp1 breaks at time=10ns
run_simulation -time 50ns

delete_simulation_breakpoint -breakpoint bp1

enable_simulation_breakpoint -breakpoint bp2

# Since bp2 is enabled, expect to see bp2 breaks at time=100ns
# when in_br becomes 0 again
run_simulation -time 60ns

# Run simulation until the end of completion
run_simulation

project_close
```

## enable_all_simulation_breakpoint

### Usage

```
enable_all_simulation_breakpoint
```

### Options

None

### Description

Enables all breakpoints.

### Example

```
project_open one_wire

initialize_simulation

# Create breakpoint bp1 which will give warning message when simulation
# time reaches 10ns
create_simulation_breakpoint -breakpoint bp1 -equation "TIME_EQ(10ns)" \
    -action "GIVE_WARNING"

# Create breakpoint bp2 which will give info message when time reaches
# 100ns
create_simulation_breakpoint -breakpoint bp2 -equation "TIME_EQ(100ns)" \
    -action "GIVE_INFO"

# Create breakpoint bp3 which will give error message when time reaches
# 200ns
create_simulation_breakpoint -breakpoint bp3 -equation "TIME_EQ(200ns)" \
    -action "GIVE_ERROR"

# All breakpoints will be disabled
disable_all_simulation_breakpoint

# Run simulation. It shouldn't break at time=10ns since bp1 has been
# disabled
run_simulation -time 50ns

enable_all_simulation_breakpoint

# Run simulation to the end. Expect bp2 and bp3 to break at 100ns and
# 200ns since all breakpoints have been enabled
run_simulation

project_close
```

## enable_simulation_breakpoint

### Usage

```
enable_simulation_breakpoint -breakpoint <breakpoint_name>
```

### Options

```
-breakpoint <breakpoint_name>: Option to enable a specified breakpoint
```

### Description

Enables a breakpoint with the specified name.

### Example

```
project_open one_wire

initialize_simulation

# Create breakpoint bp1 which will give warning message when simulation
# time reaches 10ns
create_simulation_breakpoint -breakpoint bp1 -equation "TIME_EQ(10ns)" \
    -action "GIVE_WARNING"

# Create breakpoint bp2 which will give info message when in_br has 0 in
# value
create_simulation_breakpoint -breakpoint bp2 -equation \
    "EQUAL_TO(HPATH(|one_wire|in_br),BUS(0)))" -action "GIVE_INFO"

# Create breakpoint bp3 which will give error message when both in_br and
# out_br have 1 in value
create_simulation_breakpoint -breakpoint bp3 -equation \
    "EQUAL_TO(HPATH(|one_wire|out_br),BUS(1)),EQUAL_TO(HPATH(|one_wire|in_br), \
    BUS(1))" -action "GIVE_ERROR"

# Before running simulation, disable bp2
# Therefore it shouldn't break at time=0ns when in_br=0
disable_simulation_breakpoint -breakpoint bp2

# When running simulation for the first 50ns,
# expect to see bp1 breaks at time=10ns
run_simulation -time 50ns

delete_simulation_breakpoint -breakpoint bp1

enable_simulation_breakpoint -breakpoint bp2

# Since bp2 is enabled, expect to see bp2 breaks at time=100ns
# when in_br becomes 0 again
run_simulation -time 60ns

# Run simulation until the end of completion
run_simulation

project_close
```

## fast_write_to_simulation_memory

### Usage

```
fast_write_to_simulation_memory -address <address> -data <data> -node <hpath>
```

### Options

```
-address <address>: Address of the group of memory words to which you want to write
-data <data>: Data records you want to write to the group of memory words specified by
"-address <address>"
-node <hpath>: Hierarchical path name of the logic memory
```

### Description

Writes the specified data records to the memory address of the specified group of memory words.

### Example

```
project_open fast_write

initialize_simulation -ignore_vector_file on -end_time 500ns

# Read the content of inst1|altrom:srom at the address = 14
read_from_simulation_memory -node inst1|altrom:srom -address 14
# Reading memory word inst1|altrom:srom at address 14 returns value X

# Write data = 1 to memory word inst1|altrom:srom at address 14
write_to_simulation_memory -node inst1|altrom:srom -address 14 -data 1

# Read the content of inst1|altrom:srom at the address = 14
read_from_simulation_memory -node inst1|altrom:srom -address 14
# Reading memory word inst1|altrom:srom at address 14 returns value 1

# Writes the data = 0101010101010101 for memory word inst1|altrom:srom
# start with the memory address 0
fast_write_to_simulation_memory -node inst1|altrom:srom -address 0 -data \
    0101010101010101

# Read the content of inst1|altrom:srom to check against the
# fast_write_to_simulation_memory result
read_from_simulation_memory -node inst1|altrom:srom -address 0
read_from_simulation_memory -node inst1|altrom:srom -address 1
read_from_simulation_memory -node inst1|altrom:srom -address 2
read_from_simulation_memory -node inst1|altrom:srom -address 3
read_from_simulation_memory -node inst1|altrom:srom -address 4
read_from_simulation_memory -node inst1|altrom:srom -address 5
read_from_simulation_memory -node inst1|altrom:srom -address 6
read_from_simulation_memory -node inst1|altrom:srom -address 7
read_from_simulation_memory -node inst1|altrom:srom -address 8
read_from_simulation_memory -node inst1|altrom:srom -address 9
read_from_simulation_memory -node inst1|altrom:srom -address 10
read_from_simulation_memory -node inst1|altrom:srom -address 11
read_from_simulation_memory -node inst1|altrom:srom -address 12
read_from_simulation_memory -node inst1|altrom:srom -address 13
read_from_simulation_memory -node inst1|altrom:srom -address 14
read_from_simulation_memory -node inst1|altrom:srom -address 15
# Content of inst1|altrom:srom: 0101010101010101

run_simulation

project_close
```

## force_simulation_value

### Usage

```
force_simulation_value -node <hpath> <value>
<ASCII|Binary|Fractional|Hexadecimal|Octal|Signed Decimal|Unsigned Decimal>
```

### Options

```
-node <hpath>: Hierarchical path name of the signal

<value>: Value to which you want to force the signal

<ASCII|Binary|Fractional|Hexadecimal|Octal|Signed Decimal|Unsigned Decimal>: Radix of
the value to which you want to force the signal. Default value is binary. Value can be
specified using only first 3 (except 4 for fractional) characters. E.g. sig, asc, frac
```

### Description

Forces the specified signal or group of signals to the specified value in the specified radix.

### Example

```
Example 1
---------
project_open gates

initialize_simulation -ignore_vector_file on -end_time 600ns

run_simulation -time 100ns

# Set input values for node "ina" and "inb"
force_simulation_value -node ina 0
force_simulation_value -node inb 1

run_simulation -time 100ns

# Check the value of "$ina NAND $inb"
if {[force_simulation_value -node nand_out] != [expr !($ina && $inb)]} {
   puts "$ina NAND $inb = [force_simulation_value -node nand_out]"
}

run_simulation

project_close


Example 2
---------
project_open lelut

initialize_simulation -ignore_vector_file on -end_time 100ns

# Group "inDataa inDatab inDatac inDatad" as "input_bus"
group_simulation_signal -name input_bus {inDataa inDatab inDatac inDatad}

# Set value "1011" for the group of signals
force_simulation_value -node input_bus 1011

# Set value 0 for node "clk1"
force_simulation_value -node clk1 0

run_simulation -time 20ns

# Set value 1 for node "clk1"
```

```
force_simulation_value -node clk1 1

run_simulation

project_close

Example 3
---------
project_open block1

initialize_simulation

force_simulation_value -node d 130 unsigned

project_close

Example 4
---------
project_open block1

initialize_simulation

force_simulation_value -node d 130 "Unsigned Decimal"

project_close
```

## get_simulation_memory_info

### Usage

```
get_simulation_memory_info -node <hpath> <depth|width>
```

### Options

```
-node <hpath>: Hierarchical path name of the logic memory
```

```
<depth|width>: Depth or width of the specified logic memory
```

### Description

Returns the depth (number of words or addresses) or width (number of bits per memory word) of the specified logic memory.

If neither "depth" nor "width" is specified as the positional argument, both depth and width information are returned.

### Example

```
Example 1
---------
# Get both memory depth and width information for
# node "|fast_write|lpm_rom:inst1|altrom:srom|content"
get_simulation_memory_info -node \
    |fast_write|lpm_rom:inst1|altrom:srom|content


Example 2
---------
# Get memory depth information for
# node "|fast_write|lpm_rom:inst1|altrom:srom|content"
get_simulation_memory_info -node \
    |fast_write|lpm_rom:inst1|altrom:srom|content depth


Example 3
---------
# Get memory width information for
# node "|fast_write|lpm_rom:inst1|altrom:srom|content"
get_simulation_memory_info -node \
    |fast_write|lpm_rom:inst1|altrom:srom|content width
```

## get_simulation_time

### Usage

```
get_simulation_time
```

### Options

None

### Description

Returns the amount of time, in nanoseconds, that the current simulation has been running.

### Example

```
project_open lelut

initialize_simulation

# Simulate the design until the it is complete
run_simulation -time 10ns

puts "Current time is [get_simulation_time] nanoseconds"

run_simulation

project_close
```

## get_simulation_value

### Usage

```
get_simulation_value -node <hpath>
```

### Options

```
-node <hpath>: Hierarchical path name of the signal
```

### Description

Returns the value of the specified signal.

### Example

```
project_open gates

initialize_simulation -ignore_vector_file on -end_time 600ns

run_simulation -time 100ns

# Set input values for node "ina" and "inb"
force_simulation_value -node ina 0
force_simulation_value -node inb 1

run_simulation -time 100ns

# Check the value of "$ina NAND $inb"
if {[get_simulation_value -node nand_out] != [expr !($ina && $inb)]} {
   puts "$ina NAND $inb = [get_simulation_value -node nand_out]"
}

run_simulation

project_close
```

## group_simulation_signal

### Usage

```
group_simulation_signal -name <group_name> <group_member>
```

### Options

```
-name <group_name>: Name of the group of signals
<group_member>: List of member instance names that you want to group
```

### Description

Groups the specified signals.

### Example

```
project_open lelut

initialize_simulation -ignore_vector_file on -end_time 100ns

# Group "inDataa inDatab inDatac inDatad" as "input_bus"
group_simulation_signal -name input_bus {inDataa inDatab inDatac inDatad}

# Set value "1011" for the group of signals
force_simulation_value -node input_bus 1011

# Set value 0 for node "clk1"
force_simulation_value -node clk1 0

run_simulation -time 20ns

# Set value 1 for node "clk1"
force_simulation_value -node clk1 1

run_simulation

project_close
```

## initialize_simulation

### Usage

```
initialize_simulation [-cell_delay_model_type <transport | inertial>] [-check_outputs
<On | Off>] [-end_time <end_time>] [-ignore_vector_file <On | Off>]
[-interconnect_delay_model_type <transport | inertial>] [-memory_limiter <On | Off>]
[-perform_glitch_filtering <auto | always | never>] [-power_vcd_output <target_file>]
[-pvt_multicorner <On | Off>] [-pvt_temperature <pvt_temperature>]
[-pvt_timing_model_type <slow | fast>] [-pvt_voltage <pvt_voltage>]
[-read_settings_files <On | Off>] [-saf_output <target_file>] [-sim_mode <functional |
timing | timing_using_fast_timing_model >] [-simulation_results_format <VWF | CVWF |
VCD>] [-vector_source <vector_source_file>] [-write_settings_files <On | Off>]
```

### Options

```
-cell_delay_model_type <transport | inertial>: Option to specify the type of cell delay
model to be used - transport or inertial

-check_outputs <On | Off>: Option to direct the Simulator to check expected outputs
against actual outputs in the Simulation Report

-end_time <end_time>: Option to set simulation end time when source vector file is ignored

-ignore_vector_file <On | Off>: Option to ignore source vector file

-interconnect_delay_model_type <transport | inertial>: Option to specify the type of
interconnect delay model to be used - transport or inertial

-memory_limiter <On | Off>: Option to direct the Simulator to flush signal transitions
from memory to disk for memory optimization

-perform_glitch_filtering <auto | always | never>: Option to specify that the simulator
perform glitch filtering when running timing simulation.

-power_vcd_output <target_file>: Option to specify generation of the VCD File for
PowerPlay Power Analyzer as well as the name of the generated VCD File

-pvt_multicorner <On | Off>: Option to use multicorner PVT timing model

-pvt_temperature <pvt_temperature>: Option to set the PVT temperature to be used (value
in C)

-pvt_timing_model_type <slow | fast>: Option to specify the PVT timing model to be used

-pvt_voltage <pvt_voltage>: Option to set the PVT voltage to be used (value in mV)

-read_settings_files <On | Off>: Option to read the assignments from the Quartus II
Settings File (.qsf) and override assignments obtained from the database

-saf_output <target_file>: Option to specify generation of the Signal Activity File as
well as the name of the generated Signal Activity File

-sim_mode <functional | timing | timing_using_fast_timing_model >: Option to specify the
type of simulation to perform

-simulation_results_format <VWF | CVWF | VCD>: Option to specify the type of file format
for the output vector file

-vector_source <vector_source_file>: Option to specify the source of input vectors to be
used for simulation. The specified value must be an existing file with a .vwf, .vec, or
.tbl extension

-write_settings_files <On | Off>: Option to write out the settings obtained from
command-line options to the Quartus II Settings File (.qsf).
```

### Description

Initializes the simulation for the current design. During initialization, the Simulator builds the simulation netlist and sets the simulation time to zero.

The option "-ignore_vector_file" is set to Off by default, when the source vector file exists for simulation. The Quartus II software ignores the source vector file during simulation if the option "-ignore_vector_file" is set to On. The "-end_time" option is used only when the "-ignore_vector_file" option is set to On. Use "-pvt_timing_model_type", "-pvt_voltage","-pvt_temperature" to specify an exact PVT timing model. The option "-pvt_multicorner" is used to indicate simulation to be run with multi-corner PVT timing models.

## Example

```
Example 1
---------
project_open lelut

# Source vector file exists. Initialize Simulator by getting input
# vectors
# from the vector file during simulation
initialize_simulation

# Run simulation with end time from source vector file
run_simulation

project_close


Example 2
---------
project_open lelut

# Ignore source vector file if exists. Honor specified simulation end
# time
initialize_simulation -ignore_vector_file on -end_time 350ns

# Run 350ns simulation
run_simulation

project_close

Example 3
---------
project_open chiptrip

# Run timing simulation with exact PVT timing model
initialize_simulation -pvt_timing_model_type fast -pvt_voltage 1200 \
    -pvt_temperature 0

run_simulation

project_close


Example 4
---------
project_open chiptrip

# Run timing simulation with multi-corner PVT timing models
initialize_simulation -pvt_multicorner on

run_simulation

project_close
```

## partition_vector

### Usage

```
partition_vector -duration <duration> -file <file name>
```

### Options

```
-duration <duration>: Fixed duration for the vector file to be partitioned
```

```
-file <file name>: File name of the vector file to be partitioned
```

### Description

Partitions a vector file into the specified fixed duration. The original vector file remains unchanged. New vector files are generated of the specified fixed duration and located in the same directory as the original vector file.

### Example

```
Example 1
---------
# Partition a 1us vector file by fixed duration 400ns
partition_vector -file tctin4s.vwf -duration 400ns
# Upon completion, tctin4s.vwf remains unchanged, tctin4s.0_to_400ns.vwf,
# tctin4s.400_to_800ns.vwf and tctin4s.400_to_800ns.vwf will be generated
# at the same working directory
```

## read_from_simulation_memory

### Usage

```
read_from_simulation_memory -address <address> -node <hpath>
```

### Options

```
-address <address>: Address of the memory word (or data record) from which you want to
read
-node <hpath>: Hierarchical path name of the logic memory
```

### Description

Reads the content of the specified memory address or data record of the specified logical memory.

### Example

```
project_open fast_write

initialize_simulation -ignore_vector_file on -end_time 500ns

# Read the content of inst1|altrom:srom at the address = 14
read_from_simulation_memory -node inst1|altrom:srom -address 14
# Reading memory word inst1|altrom:srom at address 14 returns value X

# Write data = 1 to memory word inst1|altrom:srom at address 14
write_to_simulation_memory -node inst1|altrom:srom -address 14 -data 1

# Read the content of inst1|altrom:srom at the address = 14
read_from_simulation_memory -node inst1|altrom:srom -address 14
# Reading memory word inst1|altrom:srom at address 14 returns value 1

# Writes the data = 0101010101010101 for memory word inst1|altrom:srom
# start with the memory address 0
fast_write_to_simulation_memory -node inst1|altrom:srom -address 0 -data \
    0101010101010101

# Read the content of inst1|altrom:srom to check against the
# fast_write_to_simulation_memory result
read_from_simulation_memory -node inst1|altrom:srom -address 0
read_from_simulation_memory -node inst1|altrom:srom -address 1
read_from_simulation_memory -node inst1|altrom:srom -address 2
read_from_simulation_memory -node inst1|altrom:srom -address 3
read_from_simulation_memory -node inst1|altrom:srom -address 4
read_from_simulation_memory -node inst1|altrom:srom -address 5
read_from_simulation_memory -node inst1|altrom:srom -address 6
read_from_simulation_memory -node inst1|altrom:srom -address 7
read_from_simulation_memory -node inst1|altrom:srom -address 8
read_from_simulation_memory -node inst1|altrom:srom -address 9
read_from_simulation_memory -node inst1|altrom:srom -address 10
read_from_simulation_memory -node inst1|altrom:srom -address 11
read_from_simulation_memory -node inst1|altrom:srom -address 12
read_from_simulation_memory -node inst1|altrom:srom -address 13
read_from_simulation_memory -node inst1|altrom:srom -address 14
read_from_simulation_memory -node inst1|altrom:srom -address 15
# Content of inst1|altrom:srom: 0101010101010101

run_simulation

project_close
```

## release_simulation_value

### Usage

```
release_simulation_value -node <hpath>
```

### Options

```
-node <hpath>: Hierarchical path name of the signal or group of signals
```

### Description

Releases the forced value of the specified signal or group of signals. After the signal's value is released, the Simulator can overwrite the current value when simulating at a future time.

### Example

```
project_open gates

initialize_simulation

run_simulation -time 10ns

puts "Value of out_a = [get_simulation_value -node out_a]"
# Value of out_a = X

force_simulation_value -node out_a 0

puts "Value of out_a = [get_simulation_value -node out_a]"
# Value of out_a = 0

run_simulation -time 10ns

puts "Value of out_a = [get_simulation_value -node out_a]"
# out_a has been forced to 0, it won't be updated during the last 10ns
# simulation.
# Value of out_a = 0

# Release the value 0 hold by out_a
release_simulation_value -node out_a

run_simulation

puts "Value of out_a = [get_simulation_value -node out_a]"
# Previous forced value has been release, out_a is updated with the
# latest value
# Value of out_a = 1

project_close
```

## run_simulation

### Usage

```
run_simulation [-time <time>]
```

### Options

```
-time <time>: Length of time the simulation runs in nanoseconds
```

### Description

Instructs the Simulator to simulate for a specified time. If you do not specify the length of time the simulation runs, the Simulator simulates until the simulation is complete.

### Example

```
Example 1
---------
project_open lelut

initialize_simulation

# Simulate the design until the it is complete
run_simulation

project_close


Example 2
---------
project_open lelut

initialize_simulation

# Simulate design for 10ns
run_simulation -time 10ns

# Change value of node "clk1" to 0
force_simulation_value -node clk1 0

# Simulate design for another 20ns
run_simulation -time 20ns

# Change value of node "clk0" to 1
force_simulation_value -node clk0 1

# Simulate until it is complete
run_simulation

project_close
```

## set_simulation_clock

### Usage

```
set_simulation_clock -clock <clock_name> [-duty_cycle <duty_cycle>] [-offset <offset>]
-period <period in ns>
```

### Options

```
-clock <clock_name>: Name of the clock signal
-duty_cycle <duty_cycle>: Duty cycle of the clock signal
-offset <offset>: Offset of the clock signal
-period <period in ns>: Period of the clock signal in nanoseconds
```

### Description

Creates a continuous clock signal in the Simulator.

The default value for the duty cycle of the clock signal is 50 and for the offset is zero nanoseconds.

### Example

```
project_open lelut

initialize_simulation -ignore_vector_file on -end_time 100ns

# Group "inDataa inDatab inDatac inDatad" under the name "input_bus"
group_simulation_signal -name input_bus {inDataa inDatab inDatac inDatad}

# Create continuous clock signal named "clk0" with duty_cycle = 50
# and offset = 25ns
set_simulation_clock -clock clk0 -period 10ns -duty_cycle 50 -offset 25

# Set value "1011" for the group of signals
force_simulation_value -node input_bus 1011

# Set value 0 for node "clk1"
force_simulation_value -node clk1 0

run_simulation -time 20ns

# Set value 1 for node "clk1"
force_simulation_value -node clk1 1

run_simulation

project_close
```

## write_to_simulation_memory

### Usage

```
write_to_simulation_memory -address <address> -data <data> -node <hpath>
```

### Options

```
-address <address>: Address of the memory word to which you want to write
-data <data>: Data record you want to write to the memory word specified by "-address
<address>"
-node <hpath>: Hierarchical path name of the logic memory
```

### Description

Writes the specified data record to the memory address of the specified memory word.

### Example

```
project_open fast_write

initialize_simulation -ignore_vector_file on -end_time 500ns

# Read the content of inst1|altrom:srom at the address = 14
read_from_simulation_memory -node inst1|altrom:srom -address 14
# Reading memory word inst1|altrom:srom at address 14 returns value X

# Write data = 1 to memory word inst1|altrom:srom at address 14
write_to_simulation_memory -node inst1|altrom:srom -address 14 -data 1

# Read the content of inst1|altrom:srom at the address = 14
read_from_simulation_memory -node inst1|altrom:srom -address 14
# Reading memory word inst1|altrom:srom at address 14 returns value 1

# Writes the data = 0101010101010101 for memory word inst1|altrom:srom
# start with the memory address 0
fast_write_to_simulation_memory -node inst1|altrom:srom -address 0 -data \
    0101010101010101

# Read the content of inst1|altrom:srom to check against the
# fast_write_to_simulation_memory result
read_from_simulation_memory -node inst1|altrom:srom -address 0
read_from_simulation_memory -node inst1|altrom:srom -address 1
read_from_simulation_memory -node inst1|altrom:srom -address 2
read_from_simulation_memory -node inst1|altrom:srom -address 3
read_from_simulation_memory -node inst1|altrom:srom -address 4
read_from_simulation_memory -node inst1|altrom:srom -address 5
read_from_simulation_memory -node inst1|altrom:srom -address 6
read_from_simulation_memory -node inst1|altrom:srom -address 7
read_from_simulation_memory -node inst1|altrom:srom -address 8
read_from_simulation_memory -node inst1|altrom:srom -address 9
read_from_simulation_memory -node inst1|altrom:srom -address 10
read_from_simulation_memory -node inst1|altrom:srom -address 11
read_from_simulation_memory -node inst1|altrom:srom -address 12
read_from_simulation_memory -node inst1|altrom:srom -address 13
read_from_simulation_memory -node inst1|altrom:srom -address 14
read_from_simulation_memory -node inst1|altrom:srom -address 15
# Content of inst1|altrom:srom: 0101010101010101

run_simulation

project_close
```

## 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 includes the following commands:

## 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 preceed 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 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 muticycle 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 vica 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_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] [-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
-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 -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
```

```
-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 generates the most important report as it 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"), or Removal Summary ("-removal").

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 [-off] [-on]
```

### Options

```
-off: Disable this setting.
-on: Enable this setting.
```

### Description

Enables 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.

### 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] [-recovery] [-removal] [-setup]
```

### Options

```
-hold: Hold Analysis
-recovery: Recovery Analysis
-removal: Removal Analysis
-setup: Setup Analysis (Default)
```

### Description

Similar to create_timing_summary, get_clock_domain_info 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 stands for 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).

This command can generate a Setup Summary ("-setup"), Hold Summary ("-hold"), Recovery Summary ("-recovery"), or Removal Summary ("-removal").

### 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]
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_info, 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>
    }
  }
  { th,
    { <th rise time>,
      <th fall time>,
      <input port>,
      <clock>
    }
  }
  { tco,
    { <tco rise time>,
      <tco fall time>,
      <output port>,
      <clock>
    }
  }
  { mintco,
    { <mintco rise time>,
      <mintco fall time>,
      <output port>,
      <clock>
    }
  }
  { tpd,
    { <tpd rise time>,
      <tpd fall time>,
      <input port>,
      <output port>
    }
  }
  { mintpd,
    { <mintpd rise time>,
      <mintpd 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.

### Example

## 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] [-model] [-name] [-speed] [-temperature]
[-voltage] <operating_condition>
```

### Options

`-display_name: Returns the operating conditions display name`

`-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 ] {
   print_path $path
   puts ""
}

delete_timing_netlist
project_close
```

## get_path_info

### Usage

```
get_path_info [-arrival_points] [-arrival_time] [-ccpp] [-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

-ccpp: Return the common clock path pessimism 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".

The -from and -to options return the ID of the nodes at the start and end, respecitvely, 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 informaion 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, respecitvely. 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, theelements 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 corrsponding 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 formated 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 -risefall

   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
```

## 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. For example, create a Tcl procedure that takes one argument, "arg".
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 outputed 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_ddr

### Usage

```
report_ddr [-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_ddr -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 Overriden, Fully Overridden, or Invalid. The status is relative to the paths and analyses coverred by the "-from", "-to", and other options. Therefore, a timing exception that reports a status of "Complete" when using "-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 the three.

You can limit the analysis performed by this command to specific start and end points, using the "-from" and "-to" options. The anlaysis 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 overrode 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 "-npaths" 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 "-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 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 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 -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. "-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 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 relibility of the design.

Please refer to the Metastabiliity 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 mininum 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," the default) or in detail ("-detail full_path"), 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 decrase 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_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)

```
-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 anlaysis 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 be 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
```

## stp

This package contains the set of Tcl functions for acquiring SignalTap II data from the Altera device.

This package is loaded by default in the following executable:

- quartus_stp

This package includes the following commands:

## close_session

### Usage

```
close_session
```

### Options

None

### Description

Saves the current session to the existing SignalTap®II File (.stp).

### Example

```
#opens signaltap session
open_session -name stp1.stp

#capture data to log named log1, timeout after 5 seconds if no trigger
# occurs
run -instance auto_signaltap_0 -signal_set signal_set_1 -trigger \
    trigger_1 -data_log log_1 -timeout 5

#close signaltap session
close_session
```

## open_session

### Usage

```
open_session -name <.stp file name>
```

### Options

```
-name <.stp file name>: SignalTap II File (.stp) name
```

### Description

Opens a session from the specified SignalTap®II File (.stp).

### Example

```
#opens signaltap session
open_session -name stp1.stp

#capture data to log named log1, timeout after 5 seconds if no trigger
# occurs
run -instance auto_signaltap_0 -signal_set signal_set_1 -trigger \
    trigger_1 -data_log log_1 -timeout 5

#close signaltap session
close_session
```

## run

### Usage

```
run [-check] [-data_log <data log>] [-device_name <device name>] [-hardware_name
<hardware name>] [-instance <instance>] [-signal_set <signal set>] [-timeout <timeout>]
[-trigger <trigger>]
```

### Options

```
-check: Option to check with last compilation result

-data_log <data log>: Name of data log to be recorded

-device_name <device name>: Device to use instead of the one specified in the stp file.
Tcl command, get_device_names, can be used to obtain the valid hardware names

-hardware_name <hardware name>: JTAG programming hardware to use instead of the one
specified in the stp file.  Tcl command, get_hardware_names, can be used to obtain the
valid hardware name

-instance <instance>: Name of instance that defines data acquisition

-signal_set <signal set>: Name of signal set that defines data acquisition

-timeout <timeout>: Timeout period for data acquisition in seconds

-trigger <trigger>: Name of trigger that defines data acquisition
```

### Description

Starts data acquisition with the specified conditions in the session and saves data into the specified data log within the timeout period.

The "-check" option indicates whether to compare the data acquisition conditions with the last compilation result.

### Example

```
#opens signaltap session
open_session -name stp1.stp

#capture data to log named log1, timeout after 5 seconds if no trigger
# occurs
run -instance auto_signaltap_0 -signal_set signal_set_1 -trigger \
    trigger_1 -data_log log_1 -timeout 5

#close signaltap session
close_session
```

## run_multiple_end

### Usage

```
run_multiple_end
```

### Options

None

### Description

Defines the end of a set of "run" commands. This command is used when multiple instances of data acquisition are started simultaneously. Add "run_multiple_start" before the set of "run" commands that specify data acquisition. Add this command after the set of commands.

If "run_multiple_end" is not included, the "run" commands do not execute.

### Example

```
#opens signaltap session
open_session -name stp1.stp

#start acquisition of instance auto_signaltap_0 and auto_signaltap_1 at
# the same time
#calling run_multiple_end will start all instances run after
# run_multiple_start call
run_multiple_start
run -instance auto_signaltap_0 -signal_set signal_set_1 -trigger \
    trigger_1 -data_log log_1 -timeout 5
run -instance auto_signaltap_1 -signal_set signal_set_1 -trigger \
    trigger_1 -data_log log_1 -timeout 5
run_multiple_end

#close signaltap session
close_session
```

## run_multiple_start

### Usage

```
run_multiple_start
```

### Options

None

### Description

Defines the start of a set of "run" commands. This command is used when multiple instances of data acquisition are started simultaneously. Add this command before the set of "run" commands that specify data acquisition. Add "run_multiple_end" after the set of commands.

If "run_multiple_end" is not included, the "run" commands do not execute.

### Example

```
#opens signaltap session
open_session -name stp1.stp

#start acquisition of instance auto_signaltap_0 and auto_signaltap_1 at
# the same time
#calling run_multiple_end will start all instances run after
# run_multiple_start call
run_multiple_start
run -instance auto_signaltap_0 -signal_set signal_set_1 -trigger \
    trigger_1 -data_log log_1 -timeout 5
run -instance auto_signaltap_1 -signal_set signal_set_1 -trigger \
    trigger_1 -data_log log_1 -timeout 5
run_multiple_end

#close signaltap session
close_session
```

## stop

### Usage

```
stop
```

### Options

None

### Description

Stops all data acquisition.

### Example

```
stop
```

## timing

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

This package is loaded by default in the following executable:

- quartus_tan

This package includes the following commands:

## compute_slack_on_edges

### Usage

```
compute_slack_on_edges [-min]
```

### Options

```
-min: Option to compute minimum slack
```

### Description

Computes slack on all edges between non-combinational nodes.

This command computes slack on every edge in all constrained data paths.The algorithm used to compute slack is the same as that used by the Fitter.

### Example

```
load_package advanced_timing
set max_slack "2147483.647"

project_open chiptrip
create_timing_netlist -skip_dat

# Perform timing analysis on the design
# and compute slack on every edge that is part
# of a constrained path
compute_slack_on_edges

# Iterate through all edges to display any edge with negative slack
set fail_count 0
set no_constraint_count 0
set ok_count 0
foreach_in_collection edge [get_timing_edges] {
    set slack [lindex [get_timing_edge_info -info slack $edge] 0]

    if { $slack < $max_slack } {
        if { $slack < 0 } {
           puts "$edge : Slack = $slack"
           incr fail_count
        } else {
           incr ok_count
        }
    } else {
        incr no_constraint_count
    }
}

puts "Edges meeting timing:        $ok_count"
puts "Edges failing timing:        $fail_count"
puts "Edges without constraints:   $no_constraint_count"

delete_timing_netlist
project_close
```

## create_timing_netlist

### Usage

```
create_timing_netlist [-fast_model] [-post_map] [-set_fmax] [-skip_dat] [-speed <speed>]
```

### Options

```
-fast_model: Option to use fast timing model
-post_map: Option to perform timing analysis on post-synthesis netlist
-set_fmax: Option to set global fmax for slack ratio
-skip_dat: Option to skip delay annotation
-speed <speed>: Speed grade
```

### 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.

Use "-skip_dat" to create the timing netlist without rerunning delay annotation. This option can be used if you previously ran the Timing Analyzer (quartus_tan) with delay annotation, or if the Fitter (quartus_fit) ran delay annotation as the final step.

Some device families, such as Stratix™and Cyclone™, may use delay annotation as part of a set of post-fitting operations. When this occurs, the Fitter displays the following message:

```
Info: Started post-fitting delay annotation
```

In this case, you can save time by skipping delay annotation.

If "--fast_model" has been used before, or if the fitter optimized to meet Fast Model Timing, then this option can be used together with "--timing_analysis_only."

Run "create_timing_netlist" before using the "report_timing" command if you have not run the Timing Analyzer (quartus_tan).

### Example

```
project_open my_top

# Create timing netlist before calling
# any report functions
create_timing_netlist

# Ready to call report functions
report_timing -npaths 1 -clock_setup
report_timing -npaths 1 -tsu
report_timing -npaths 1 -tco

# 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
report_timing -clock_setup -clock_filter clk
delete_timing_netlist

# Report hold violation for fastest corner
```

```
create_timing_netlist -fast_model
report_timing -clock_hold -clock_filter clk
delete_timing_netlist

# If Delay Annotation has been run for the fast corner
# Skip running it again
create_timing_netlist -fast_model -skip_dat
report_timing -clock_hold -clock_filter clk
delete_timing_netlist

# Report worst case period for post-technology mapping netlist
create_timing_netlist -post_map
report_timing -clock_setup -clock_filter clk
delete_timing_netlist

project_close
```

## delete_timing_netlist

### Usage

```
delete_timing_netlist
```

### Options

None

### Description

Deletes the timing netlist.

Use this command to delete a timing netlist previously created or to read from the database.

### Example

```
project_open my_top

create_timing_netlist -speed 5
report_timing -npath 1 -clock_setup
delete_timing_netlist

# Unless you delete the netlist, the following
# command gives an error
create_timing_netlist -speed 6
report_timing -npath 1 -clock_setup
delete_timing_netlist

project_close
```

## remove_timing_tables

### Usage

```
remove_timing_tables
```

### Options

None

### Description

Removes all custom timing tables from the database Requires that the ::quartus::report package be loaded

### Example

```
project_open chiptrip

# Required by creating a report panel
load_package report
load_report

# Create timing netlist for reporting.
create_timing_netlist -skip_dat

# Remove all the previous timing tables.
remove_timing_tables

# Generate Top_10_Clock_Setup timing table.
report_timing -clock_setup -table "Top_10_Clock_Setup" -npaths 10

# Save changes to database. Otherwise changes will be discarded
# when the project gets closed or the report gets unloaded.
write_report_database

unload_report
project_close
```

## report_timing

### Usage

```
report_timing [-all_failures] [-append] [-clock_filter <names>] [-clock_hold]
[-clock_hold_core] [-clock_hold_io] [-clock_setup] [-clock_setup_core]
[-clock_setup_io] [-dqs_read_capture] [-file <name>] [-from <names>] [-longest_paths]
[-min_tco] [-min_tpd] [-npaths <number>] [-recovery] [-removal] [-shortest_paths]
[-src_clock_filter <names>] [-stdout] [-table <name>] [-tco] [-th] [-to <names>] [-tpd]
[-tsu]
```

### Options

```
-all_failures: Option to report only paths with negative slack

-append: Option to append results to output file

-clock_filter <names>: Legal clocks for clock analyses (string patterns are matched using
Tcl string matching)

-clock_hold: Option to report clock hold paths

-clock_hold_core: Option to report clock hold core paths

-clock_hold_io: Option to report clock hold I/O paths

-clock_setup: Option to report clock setup paths

-clock_setup_core: Option to report clock setup core paths

-clock_setup_io: Option to report clock setup I/O paths

-dqs_read_capture: Option to report DQS strobe-to-core paths

-file <name>: File to which to write the report

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

-longest_paths: Option to report worst-case longest paths between specified nodes

-min_tco: Option to report minimum tco paths

-min_tpd: Option to report minimum tpd paths

-npaths <number>: Number of paths to report (default is 1)

-recovery: Option to report recovery paths

-removal: Option to report removal paths

-shortest_paths: Option to report worst-case shortest paths between specified nodes

-src_clock_filter <names>: Legal source clocks for clock analyses (string patterns are
matched using Tcl string matching)

-stdout: Option to send report to standard output

-table <name>: Custom timing table to be created

-tco: Option to report tco paths

-th: Option to report th paths

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

-tpd: Option to report tpd paths

-tsu: Option to report tsu paths
```

### Description

Reports timing paths using Altera®or ASIC (industry-standard) format.

You can use one or more of the filter options to specify the number of paths displayed. You can redirect the output to a file by using the "-file" option. Use the"-append" option together with "-file" to append results to an existing file.

The "-from", "-to", "-src_clock_filter", and "-clock_filter" options are case sensitive. These options can take list of names. For example, from Reg0 or inst3 to Reg1 or Out[3] clocked by clk0 or clk1 can be expressed as:

```
report_timing -from Reg0 inst3 -to Reg1 Out[3] -clock_filter clk0 clk1
```

The "-from", "-to", "-src_clock_filter", and "-clock_filter" options can take stringpatterns containing special characters from the set "*?\[]" as values. The values are matched using Tcl string matching. Bus names are detected automatically and do not need to be escaped. Bus names have the following format:

```
<bus name>[<bus index>] or <bus name>[*]
```

The <bus name> portion is a string of alphanumeric characters. The <bus index> portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the <bus index> is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names and can be used as follows:

```
# To match index 0 of bus "a", type:
report_timing -to a[0]

# To match all indices of bus "a", type:
report_timing -to a[*]
```

All other uses of square brackets must be escaped if you do not intend to use them as string patterns. For example, to match indices 0, 1, and 2 of the bus "a", type:

```
report_timing -to "a[escape_brackets \[]\[0-2\][escape_brackets \]]"
```

For more information about escaping square brackets, type "escape_brackets -h".

This command returns the number of timing paths reported. It returns "0" if no paths are found for the specified filters.

This command is similar to the "list_path" command in the ::quartus::timing_report package (available in the quartus_tan executable). The difference is that the "report_timing" command does not require a completed timing analysis. The "report_timing" command computes the timing paths and reports them.

The "report_timing" command operates in a manner similar to commands in other tools, such as the Synopsys®PrimeTime software. Running "report_timing" may take a little longer than running the "list_path" command, but is more memory efficient.

## Example

```
project_open my_project

# Always create the netlist first
create_timing_netlist

# List paths that represent input setup paths
report_timing -tsu

# List clock setup paths for clock clk
report_timing -clock_setup -clock_filter clk

# List clock setup paths for clock clk from source clock src_clock
report_timing -clock_setup -clock_filter clk -src_clock_filter src_clock

# List clock setup paths for clock clk
# from registers abc* to registers xyz*
report_timing -clock_setup -clock_filter clk -from abc* -to xyz*
```

```
# List the top 5 pin-to-pin combinational paths
report_timing -tpd -npaths 5

# List the top 5 pin-to-pin combinational paths and
# write output to an out.tao file
report_timing -tpd -npaths 5 -file out.tao

# Compute min tpd and append results to existing out.tao
report_timing -min_tpd -npaths 5 -file out.tao -append

# Show longest path (register to register data path) between a* and b*
report_timing -longest_paths -npaths 1

# Specify bus
set path_count [report_timing -from in[0]]
if { $path_count == 0 } {
    puts "No paths found from in[0]"
}

############################################################
# Use this script to create a custom report that contains
# both the Fast Timing Model results and the Slow Model results
# in one combined report
# Use report_timing with the -table
# Usage:
#    quartus_map top
#    quartus_fit top
#    quartus_tan -t <script>.tcl top
#    (Then open Quartus II GUI's report)
############################################################

load_package report
load_package advanced_timing

# Assume one argument: Name of project (where project == revision)
project_open [lindex $quartus(args) 0]

load_report
remove_timing_tables

# First, Build Netlist usign the Fast Timing Model
create_timing_netlist -fast_model

# Create panels using Fast Model
# Use -table option to create Quartus report panels
foreach_in_collection clk [get_timing_nodes -type clk] {
    set name [get_timing_node_info -info name $clk]
    report_timing -clock_hold -clock_filter $name -npaths 200 -table "Min \
        Clock Hold: $name"
}
report_timing -min_tco -npaths 200 -table "Min Tco"
report_timing -tsu -npaths 200 -table "Min Tsu"
report_timing -th -npaths 200 -table "Min Th"

# Save changes to the report file before deleting netlist
save_report_database
delete_timing_netlist

# Now do it again, but this time with the default slow timing model
create_timing_netlist

foreach_in_collection clk [get_timing_nodes -type clk] {
    set name [get_timing_node_info -info name $clk]
    report_timing -clock_setup -clock_filter $name -npaths 200 -table \
        "Max Clock Setup: $name"
    report_timing -clock_hold -clock_filter $name -npaths 200 -table "Max \
        Clock Hold: $name"
}
report_timing -tsu -npaths 200 -table "Max Tsu"
report_timing -th -npaths 200 -table "Max Th"
report_timing -tco -npaths 200 -table "Max Tco"

# Save changes again, delete netlist, unload report and close project
save_report_database
delete_timing_netlist
unload_report
project_close
```

# timing_assignment

This package contains the set of Tcl functions for making project-wide timing assignments, including clock assignments.

In version 6.0 of ::quartus::project package, all Tcl commands designed to process Timing Analyzer assignments have been moved to this package.

This package is loaded by default in the following executables:

- quartus
- quartus_cdb
- quartus_sh
- quartus_sim
- quartus_stp
- quartus_tan

This package includes the following commands:

## create_base_clock

### Usage

```
create_base_clock [-comment <comment>] [-disable] [-duty_cycle <integer>] [-entity
<entity>] -fmax <fmax> [-no_target] [-tag <data>] [-target <name>] [-virtual]
<clock_name>
```

### Options

```
-comment <comment>: Comment
-disable: Option to disable assignment
-duty_cycle <integer>: Duty cycle
-entity <entity>: Entity to which to add clock assignment
-fmax <fmax>: Clock frequency
-no_target: Option to not assign clock to node
-tag <data>: Option to tag data to this assignment
-target <name>: Clock node name
-virtual: Option to specify the clock as a virtual clock
<clock_name>: Clock name
```

### Description

Creates the base clock. The base clock is an absolute clock.

The "-fmax" option can take the format:

```
<floating point time value><time unit>
```

For example, if the fmax is 10.55ns, "10.55" is the <floating point time value> and "ns" is the <time unit>.

The following table displays possible time units:

| Time Unit | Description |
|---|---|
| s | second(s) |
| ms | millisecond(s) |
| us | microsecond(s) |
| ns | nanosecond(s) |
| ps | picosecond(s) |
| fs | femtosecond(s) |
| Hz | hertz |
| KHz | kilohertz |
| MHz | megahertz |
| GHz | gigahertz |

If you specify the "-virtual" option, the base clock is not assigned to any node in the timing netlist. You cannot specify the "-virtual" option and the "-target" option at the same time.

For entity-specific assignments, use the "-entity" option to force the assignment to specified entity. If you do not specify the "-entity" option, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.

Assignments created or modified by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments
- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

## Example

```
# Specify a clock named "clk50" with
# a 50ns period
# The command specifies a CLOCK section
# in the active project with the 50ns
# specification, and adds a
# "clk50 : CLOCK_SETTING=clk50" assignment
# to the current entity
create_base_clock -fmax 50ns clk50

# Specify the same clk50 to a pin with
# a different name (myclkpin)
create_base_clock -fmax 50ns -target myclkpin clk50

# Specify the entity name to which the clock
# is added, using the -entity option
# This is needed if the top-level entity name
# is other than that of the project
# The following command generates a "top_level" entity.
create_base_clock -fmax 50ns -entity top_level clk50
```

## create_relative_clock

### Usage

```
create_relative_clock -base_clock <Base clock> [-comment <comment>] [-disable] [-divide
<integer>] [-duty_cycle <integer>] [-entity <entity>] [-invert] [-multiply <integer>]
[-no_target] [-offset <offset>] [-phase_shift <integer>] [-tag <data>] [-target <name>]
[-virtual] <clock_name>
```

### Options

```
-base_clock <Base clock>: Base clock name
-comment <comment>: Comment
-disable: Option to disable assignment
-divide <integer>: Base clock division factor
-duty_cycle <integer>: Duty cycle
-entity <entity>: Entity to which to add clock assignment
-invert: Option to invert base clock
-multiply <integer>: Base clock multiplication factor
-no_target: Option to not assign clock to node
-offset <offset>: Offset from base clock
-phase_shift <integer>: Phase shift from base clock
-tag <data>: Option to tag data to this assignment
-target <name>: Clock node name
-virtual: Option to specify the clock as a virtual clock
<clock_name>: Clock name
```

### Description

Creates a relative clock that derived from the absolute clock.

The "-offset" option can take the format:

```
<floating point time value><time unit>
```

For example, if the offset is 10.55ns, "10.55" is the <floating point time value> and "ns" is the <time unit>.

The following table displays possible time units:

| Time Unit | Description |
|---|---|
| s | second(s) |
| ms | millisecond(s) |
| us | microsecond(s) |
| ns | nanosecond(s) |
| ps | picosecond(s) |
| fs | femtosecond(s) |

The "-phase_shift" option takes an integer that represents degrees of phase shift from the base clock period. For example, if a base clock has a period of 10ns and clk2 is a relative clock derived from the base clock. A phase shift value of 45 applies a 45 degree phase shift to clk2, producing an offset of 1.25ns from the base clock. For a given relative clock, you may specify a phase shift, an offset, or both. If both are specified, they are additive.

If you specify the "-virtual" option, the relative clock is not assigned to any node in the timing netlist. You cannot specify the "-virtual" option and the "-target" option at the same time.

For entity-specific assignments, use the "-entity" option to force the assignment to specified entity. If you do not specify the "-entity" option, the value for the FOCUS_ENTITY_NAME assignment is used. If the FOCUS_ENTITY_NAME value is not found, the revision name is used.

Assignments created or modified by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments
- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

## Example

```
## Specify a base clock of 10ns
create_base_clock -fmax 10ns clk10

## Specify a relative clock with 2/3 the period
create_relative_clock -base_clock clk10 -multiply 2 -divide 3 clk2_3

## Specify a relative clock with a phase shift of 45 degrees
create_relative_clock -base_clock clk10 -phase_shift 45 clk_45
## or, equivalently, with an offset of 1.25ns
create_relative_clock -base_clock clk10 -offset 1.25ns clk_45

## Specify the entity name to which the clock
## is added, using the -entity option
## This is needed if the top-level entity name is
## other than that of the project
## The following command generates a "top_level" entity
create_relative_clock -base_clock clk10 -entity top_level -multiply 2 \
    -divide 3 clk2_3
```

## get_clocks

### Usage

```
get_clocks [-tag <data>]
```

### Options

```
-tag <data>: Option to tag data to this assignment
```

### Description

Returns a list of lists consisting of node name and clock setting name. The output has the following format:

```
{{<node name #1> <clock setting name #1>}
 {<node name #2> <clock setting name #2>}
 ...
 {<node name #N> <clock setting name #N>}}
```

If <node name> is empty, <clock setting name> was not assigned to any node. If <clock setting name> is empty, <node name> was assigned to an undefined <clock setting name>.

You can create <clock setting name> using the "create_base_clock" or "create_relative_clock" commands. You can create node names using the following command:

```
set_instance_assignment -name CLOCK_SETTINGS -to <clock setting name> <node name>
```

### Example

```
# Search for all pairs of node names and clock setting names
# and print the information

set clock_lists [get_clocks]

foreach clock_asgn $clock_lists {
     set node_name [lindex $clock_asgn 0]
     set clock_setting_name [lindex $clock_asgn 1]

     if { $node_name == "" } {
          puts "No node uses the clock \"$clock_setting_name\""
     } elseif {$clock_setting_name == ""} {
          puts "The node \"$node_name\" uses an undefined clock setting";
     } else {
          puts "The node \"$node_name\" uses the clock \
              \"$clock_setting_name\""
     }
}
```

## set_clock_latency

### Usage

```
set_clock_latency [-early] [-late] [-tag <data>] -to <to> <value>
```

### Options

```
-early: Early clock latency
-late: Late clock latency
-tag <data>: Option to tag data to this assignment
-to <to>: Destination clock name
<value>: Input delay value after rise of reference clock
```

### Description

Specifies the required early or late clock latency on the clock or input assignment group as specified by the "-to" option. The Quartus II®timing analysis propagates this latency and checks it against the actual latency. For more information about assignment groups, type "assignment_group -h".

The "assignment_group" command replaces the deprecated "timegroup" command in ::quartus::project, version 5.0.

If both "-early" and "-late" options are not used, then the assignment is set for both early and late latency.

The <value> is the latency of the reference clock. The format of <value> is "<numerical value><time unit>", for example, "7.55ns".

The following table shows the available time units:

| Time Unit | Description |
|---|---|
| s | second(s) |
| ms | millisecond(s) |
| us | microsecond(s) |
| ns | nanosecond(s) |
| ps | picosecond(s) |
| fs | femtosecond(s) |
| Hz | Hertz |
| KHz | KiloHertz |
| MHz | MegaHertz |
| GHz | GigaHertz |

Assignments created or modified by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments
- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

## Example

```
## Set early clock latency to Clk0
set_clock_latency 2ns -to Clk0 -early

## Set early and late clock latency to Clk*
set_clock_latency 2ns -to Clk*

## Use assignment_group to set clock latency.
assignment_group "clock_group" -add_member "Clk*" -add_exception "Clk0"
set_clock_latency 2ns -to "clock_group" -early

## Commit the assignments to .qsf file.
export_assignments
```

## set_clock_uncertainty

### Usage

```
set_clock_uncertainty [-comment <comment>] [-disable] [-from <src_clock_name>] [-hold]
[-remove] [-setup] [-tag <data>] -to <dst_clock_name> <value>
```

### Options

```
-comment <comment>: Comment
```

```
-disable: Option to disable assignment
```

```
-from <src_clock_name>: Source clock name
```

```
-hold: Option to specify hold time uncertainty
```

```
-remove: Option to remove assignment
```

```
-setup: Option to specify setup time uncertainty
```

```
-tag <data>: Option to tag data to this assignment
```

```
-to <dst_clock_name>: Destination clock name
```

```
<value>: Amount of expected clock jitter
```

### Description

Specifies simple or interlock clock uncertainty (or clock jitter) used during setup and/or hold time analysis. The optional source clock <src_clock_name> is specified for interlock uncertainty and is not specified for simple uncertainty.

You may specify the -setup option, the -hold option, or neither. If neither is specified, then the command applies to both setup and hold.

The <value> is the expected amount of clock jitter. The format of <value> is "<numerical value><time unit>", for example, "0.5ns".

The following table displays available time units:

| **Time Unit** | **Description** |
|---|---|
| s | second(s) |
| ms | millisecond(s) |
| us | microsecond(s) |
| ns | nanosecond(s) |
| ps | picosecond(s) |
| fs | femtosecond(s) |
| Hz | hertz |
| KHz | kilohertz |
| MHz | megahertz |
| GHz | gigahertz |

Assignments created or modified by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments
- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

### Example

```
## Specify a simple setup uncertainty of .3ns
## and a simple hold uncertainty of .1ns
set_clock_uncertainty 0.3ns -to clk -setup
set_clock_uncertainty 0.1ns -to clk -hold

## Both of the following commands specify a
## simple setup and hold uncertainty of .3ns
set_clock_uncertainty 0.3ns -to clk
## or
set_clock_uncertainty 0.3ns -to clk -setup -hold

## Specify an interlock setup uncertainty of .3ns
## and an interlock hold uncertainty of .1ns
set_clock_uncertainty 0.3ns -to clk -from src_clk -setup
set_clock_uncertainty 0.1ns -to clk -from src_clk -hold
```

## set_input_delay

### Usage

```
set_input_delay [-clk_ref <clock>] [-clock_fall] [-comment <comment>] [-disable] [-max]
[-min] [-remove] [-tag <data>] -to <input_pin> <value>
```

### Options

```
-clk_ref <clock>: Reference clock name
-clock_fall: Option to specify that delay is relative to falling edge of reference clock
-comment <comment>: Comment
-disable: Option to disable assignment
-max: Option to set maximum delay
-min: Option to set minimum delay
-remove: Option to remove input delay requirement
-tag <data>: Option to tag data to this assignment
-to <input_pin>: Input pin name or input assignment group
<value>: Input delay value after rise of reference clock
```

### Description

Specifies the required minimum or maximum delay on the input pin or input assignment group as specified by the "-to" option. The Quartus II®timing analysis propagates this delay and checks it against the actual delay. For more information about assignment groups, type "assignment_group -h".

The "assignment_group" command replaces the deprecated "timegroup" command in ::quartus::project, version 5.0.

If both "-min" and "-max" options are not used, then the assignment is set for both minimum and maximum delay.

The "-clock_fall" option specifies that the input delay is relative to the falling edge of the reference clock. The rising edge of the reference clock is the default.

The <value> is the input delay after the rise or fall of the reference clock. If you do not specify the "-clk_ref" option, all clocks are assumed to be reference clocks by default.

The format of <value> is "<numerical value><time unit>", for example, "7.55ns".

The following table shows the available time units:

| Time Unit | Description |
|---|---|
| s | second(s) |
| ms | millisecond(s) |
| us | microsecond(s) |
| ns | nanosecond(s) |
| ps | picosecond(s) |
| fs | femtosecond(s) |
| Hz | Hertz |
| KHz | KiloHertz |

| Time Unit | Description |
|---|---|
| MHz | MegaHertz |
| GHz | GigaHertz |

## Example

```
## Specify the required minimum and maximum input
## delays on the input pin named "ipin" relative to
## the rising edge of the reference clock named "clk1"
set_input_delay 2ns -to "ipin" -clk_ref "clk1"
## Or, equivalently,
set_input_delay 2ns -to "ipin" -clk_ref "clk1" -min -max

## Specify the required minimum input delay on
## the input pin named "ipin" relative to the
## falling edge of the reference clock named "clk1"
set_input_delay 2ns -to "ipin" -clk_ref "clk1" -min -clock_fall

## Specify the required maximum input delay on
## input pins with names that start with "i"
## except those that start with "ibus"
assignment_group "input_pins" -add_member "i*" -add_exception "ibus*"
set_input_delay 2ns -to "input_pins" -max
```

## set_multicycle_assignment

### Usage

```
set_multicycle_assignment [-comment <comment>] [-disable] [-end] [-from <from_list>]
[-hold] [-remove] [-setup] [-start] [-tag <data>] [-to <to_list>] <path_multiplier>
```

### Options

`-comment <comment>`: Comment

`-disable`: Option to disable multicycle assignment

`-end`: Option to indicate that destination clock cycles should be considered for path multiplier

`-from <from_list>`: List of clock names, node names, and/or assignment group names that represent start or source points of multicycle path, for example, {node1 node2 ...}

`-hold`: Option to indicate that path multiplier is meant for hold

`-remove`: Option to remove multicycle assignment

`-setup`: Option to indicate that path multiplier is meant for setup

`-start`: Option to indicate that source clock cycles must be considered for path multiplier

`-tag <data>`: Option to tag data to this assignment

`-to <to_list>`: List of clock names, node names, and/or assignment group names that represent end or destination points of multicycle path, for example, {node1 node2 ...}

`<path_multiplier>`: Multicycle path multiplier

### Description

Specifies that the given timing paths have multicycle setup or hold delays with the number of cycles specified by the "-path_multiplier" option.

If neither the "-setup" nor "-hold" options are used, the "-setup" option is the default option. If neither the "-start" nor "-end" options are used, the "-end" option is the default option.

You must use either the "-from <from_list>" or "-to <to_list>" option.

Note that Quartus II timing analysis is optimized to use assignment groups for timing constraints instead of a list of nodes. Of the following two methods to make multicycle assignments, method (1) is the optimal method.

```
(1)  assignment_group "src_group" -add_member "s1"
     assignment_group "src_group" -add_member "s2"
     assignment_group "src_group" -add_member "s3"
     assignment_group "dst_group" -add_member "d1"
     assignment_group "dst_group" -add_member "d2"
     set_multicycle_assignment -from "src_group" -to "dst_group"

(2)  set_multicycle_assignment -from {s1 s2 s3} -to {d1 d2}
```

For more information about assignment groups, type "assignment_group -h".

The "assignment_group" command replaces the deprecated "timegroup" command in ::quartus::project, version 5.0.

The meaning of multicycle hold differs between the Quartus II software timing analysis and the Synopsys PrimeTime software timing analysis. Refer to the online Help of each software for more information.

Assignments created or modified by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments

- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

## Example

```
## Multicycle "setup" from reg1 and reg2 to any destination points
assignment_group "src_group" -add_member reg1
assignment_group "src_group" -add_member reg2
set_multicycle_assignment 2 -setup -from "src_group"

## or
assignment_group "src_group" -add_member reg1
assignment_group "src_group" -add_member reg2
assignment_group "dst_group" -add_member *
set_multicycle_assignment 2 -setup -from "src_group" -to "dst_group"

## Source multicycle "setup" to reg1 and reg2 from any source points
assignment_group "dst_group" -add_member reg1
assignment_group "dst_group" -add_member reg2
set_multicycle_assignment 2 -setup -start -to "dst_group"

## or
assignment_group "dst_group" -add_member reg1
assignment_group "dst_group" -add_member reg2
assignment_group "src_group" -add_member *
set_multicycle_assignment 2 -setup -start -from "src_group" -to \
    "dst_group"

## Source multicycle "hold" from src1 to dst1 and dst2 and
## from src2 to dst1 and dst2
assignment_group "src_group" -add_member src1
assignment_group "src_group" -add_member src2
assignment_group "dst_group" -add_member dst1
assignment_group "dst_group" -add_member dst2
set_multicycle_assignment 2 -hold -from "src_group" -to "dst_group"

## Source multicycle "hold" from registers clocked by clk1
## to registers clocked by clk2
## Timegroups are useful for making assignments to
## more than one node. Timegroups are not necessary
## for making an assignment from only one clock node
## to another clock
set_multicycle_assignment 2 -hold -start -from clk1 -to clk2
```

## set_output_delay

### Usage

```
set_output_delay [-clk_ref <clock>] [-clock_fall] [-comment <comment>] [-disable] [-max]
[-min] [-remove] [-tag <data>] -to <output_pin> <value>
```

### Options

```
-clk_ref <clock>: Reference clock name
-clock_fall: Option to specify that delay is relative to falling edge of reference clock
-comment <comment>: Comment
-disable: Option to disable assignment
-max: Option to set maximum delay
-min: Option to set minimum delay
-remove: Option to remove output delay requirement
-tag <data>: Option to tag data to this assignment
-to <output_pin>: Output pin name or output assignment group
<value>: Delay value after rise of reference clock
```

### Description

Specifies the required minimum or maximum delay on the output pin or output assignment group as specified by the "-to" option. The Quartus II®timing analysis propagates this delay and checks it against the actual delay. For more information about assignment groups, type "assignment_group -h".

The "assignment_group" command replaces the deprecated "timegroup" command in ::quartus::project, version 5.0.

If both "-min" and "-max" options are not used, then the assignment is set for both minimum and maximum delay.

The "-clock_fall" option specifies that the output delay is relative to the falling edge of the reference clock. The rising edge of the reference clock is the default.

The <value> is the output delay after the rise or fall of the reference clock. If you do not specify the "-clk_ref" option, all clocks are assumed to be reference clocks by default.

The format of <value> is "<numerical value><time unit>", for example, "7.55ns".

The following table displays available time units:

| Time Unit | Description |
|---|---|
| s | second(s) |
| ms | millisecond(s) |
| us | microsecond(s) |
| ns | nanosecond(s) |
| ps | picosecond(s) |
| fs | femtosecond(s) |
| Hz | Hertz |
| KHz | KiloHertz |

| Time Unit | Description |
|---|---|
| MHz | MegaHertz |
| GHz | GigaHertz |

## Example

```
## Specify the required minimum and maximum output
## delays on the output pin named "opin" relative to
## the rising edge of the reference clock named "clk1"
set_output_delay 2ns -to "opin" -clk_ref "clk1"
## Or, equivalently,
set_output_delay 2ns -to "opin" -clk_ref "clk1" -min -max

## Specify the required minimum output delay on
## the output pin named "opin" relative to the
## falling edge of the reference clock named "clk1"
set_output_delay 2ns -to "opin" -clk_ref "clk1" -min -clock_fall

## Specify the required maximum output delay on
## output pins with names that start with "o"
## except those that start with "obus"
assignment_group "output_pins" -add_member "o*" -add_exception "obus*"
set_output_delay 2ns -to "output_pins" -max
```

## set_timing_cut_assignment

### Usage

```
set_timing_cut_assignment [-comment <comment>] [-disable] [-from <from_pin_list>]
[-remove] [-tag <data>] [-to <to_pin_list>]
```

### Options

`-comment <comment>: Comment`

`-disable: Option to disable assignment`

`-from <from_pin_list>: List of start or source node names and/or assignment group names for timing path, for example, {node1 node2 ...}`

`-remove: Option to remove timing cut assignment`

`-tag <data>: Option to tag data to this assignment`

`-to <to_pin_list>: List of end node names and/or assignment group names for timing path, for example, {node1 node2 ...}`

### Description

Specifies that the timing paths that start from the designated <from_pin_list> and end in the designated <to_pin_list> are false paths.

Nodes for the <from_pin_list> can be input pins, internal nodes, clock pins, or assignment groups. Nodes for the <to_pin_list> can be output pins, internal nodes, clock pins, or assignment groups.

You must use either the "-from <from_pin_list>" or the "-to <to_pin_list>" option.

Note that Quartus II timing analysis is optimized to use assignment groups for timing constraints instead of a list of nodes. Of the following two methods to make timing cut assignments, method (1) is the optimal method.

```
(1) assignment_group "src_group" -add_member "s1"
    assignment_group "src_group" -add_member "s2"
    assignment_group "src_group" -add_member "s3"
    assignment_group "dst_group" -add_member "d1"
    assignment_group "dst_group" -add_member "d2"
    set_timing_cut_assignment -from "src_group" -to "dst_group"
```

```
(2) set_timing_cut_assignment -from {s1 s2 s3} -to {d1 d2}
```

For more information about assignment groups, type "assignment_group -h".

The "assignment_group" command replaces the deprecated "timegroup" command in ::quartus::project, version 5.0.

Assignments created or modified by using this Tcl command are not saved to the Quartus II Settings File (.qsf) unless you explicitly call one of the following two Tcl commands:

- export_assignments
- project_close (unless "-dont_export_assignments" is specified)

These two Tcl commands reside in the ::quartus::project Tcl package. You must save assignment changes before you run Quartus II command-line executables. Note, however, that the Tcl commands "execute_flow" and "execute_module" (part of the ::quartus::flow Tcl package) automatically call "export_assignments" before they run command-line executables.

## Example

```
## Set timing cut from any source points to dst1 and dst2
assignment_group "dst_group" -add_member dst1
assignment_group "dst_group" -add_member dst2
set_timing_cut_assignment -to "dst_group"

## or
assignment_group "src_group" -add_member *
assignment_group "dst_group" -add_member dst1
assignment_group "dst_group" -add_member dst2
set_timing_cut_assignment -from "src_group" -to "dst_group"

## Set timing cut from src1 and src2 to any end points
assignment_group "src_group" -add_member src1
assignment_group "src_group" -add_member src2
set_timing_cut_assignment -from "src_group"

## or
assignment_group "src_group" -add_member src1
assignment_group "src_group" -add_member src2
assignment_group "dst_group" -add_member *
set_timing_cut_assignment -from "src_group" -to "dst_group"
```

# timing_report

This package contains the set of Tcl functions for traversing the timing netlist and obtaining information about timing paths.

This package is loaded by default in the following executable:

- quartus

This package is available for loading in the following executable:

- quartus_tan

This package includes the following commands:

## list_path

### Usage

```
list_path [-append] [-clock_filter <names>] [-clock_hold] [-clock_hold_io]
[-clock_setup] [-clock_setup_io] [-dqs_read_capture] [-file <name>] [-from <names>]
[-min_tco] [-min_tpd] [-npaths <number>] [-src_clock_filter <names>] [-stdout] [-tco]
[-th] [-to <names>] [-tpd] [-tsu]
```

### Options

```
-append: Option to append results to output file
-clock_filter <names>: Legal clocks for clock analyses (string patterns are matched using
Tcl string matching)
-clock_hold: Option to report clock hold paths
-clock_hold_io: Option to report clock hold I/O paths
-clock_setup: Option to report clock setup paths
-clock_setup_io: Option to report clock setup I/O paths
-dqs_read_capture: Option to report maximum DQS read capture paths
-file <name>: File to which to write the report
-from <names>: Legal sources (string patterns are matched using Tcl string matching)
-min_tco: Option to report minimum tco paths
-min_tpd: Option to report minimum tpd paths
-npaths <number>: Number of paths to report (default is 1)
-src_clock_filter <names>: Legal source clocks for clock analyses (string patterns are
matched using Tcl string matching)
-stdout: Option to send report to standard output
-tco: Option to report tco paths
-th: Option to report th paths
-to <names>: Legal destinations (string patterns are matched using Tcl string matching)
-tpd: Option to report tpd paths
-tsu: Option to report tsu paths
```

### Description

Reports the timing paths using the list path format. You can use one or several of the filter options to specify the number of paths displayed. You can redirect the output to a file using the "-file" option. Use the "-append" option to append the results to an existing file.

The "-clock_filter", "-to", and "-from" options are case sensitive. These options can take string patterns containing special characters from the set "*?\[]" as values. The values are matched using Tcl string matching. Note that bus names are automatically detected and do not need to be escaped. Bus names have the following format:

```
<bus name>[<bus index>] or <bus name>[*]
```

The <bus name> portion is a string of alphanumeric characters. The <bus index> portion is an integer greater than or equal to zero or it can be the character "*" used for string matching. Notice that the <bus index> is enclosed by the square brackets "[" and "]". For example, "a[0]" and "a[*]" are supported bus names and can be used as follows:

```
# To match index 0 of bus "a", type:
list_path -to a[0]
```

```
# To match all indices of bus "a", type:
list_path -to a[*]
```

All other uses of square brackets must be escaped if you do not intend to use them as string patterns. For example, to match indices 0, 1, and 2 of the bus "a", type:

```
list_path -to "a[escape_brackets \[]\[0-2\][escape_brackets \]]"
```

For more information about escaping square brackets, type "escape_brackets -h".

You must run the Timing Analyzer (quartus_tan) successfully before using this command, so that the Timing Analyzer report sections generated for the project appear in the Compilation Report. Note that if the requested path(s) are not found in the Timing Analysis report panel, the command does not report anything, and the return value simply shows "0". Otherwise, the command returns the number of reported paths.

This command is similar to the "report_timing" command in the ::quartus::timing package (accessible from the quartus_tan executable). The "report_timing" command does not require a completed Timing Analysis report, and therefore can report paths even if they do not appear in the report.

## Example

```
# The following examples work from within the Quartus II
# Tcl Console after a successful Timing Analysis:

# List paths that represent input setup paths
if {[list_path -tsu] == 0} {
    puts "No Tsu paths exist in the design"
}

# List clock setup paths for clock clk
list_path -clock_setup -clock_filter clk

# List clock setup paths from clock clk to clock clk
list_path -clock_setup -src_clock_filter clk -clock_filter clk

# List clock setup paths for clock clk
# from registers abc* to registers xyz*
list_path -clock_setup -clock_filter clk -from abc* -to xyz*

# List the top 5 pin-to-pin combinational paths
list_path -tpd -npaths 5

# List the top 5 pin-to-pin combinational paths and
# write output to an out.tao file
set path_count [list_path -tpd -npaths 5 -file out.tao]
puts "$path_count == 5"

# Append min tpd results to the existing out.tao file
list_path -tpd -npaths 5 -file out.tao -append

# Specify bus
list_path -from in[0]
```