Introduction

The Quartus® II TimeQuest Timing Analyzer is a powerful ASIC-style timing analysis tool that validates the timing performance of all logic in your design using an industry-standard constraint, analysis, and reporting methodology. Use the Quartus II TimeQuest Timing Analyzer's GUI or command-line interface to constrain, analyze, and report results for all timing paths in your design.

Before running the Quartus II TimeQuest Timing Analyzer, you must specify initial timing constraints that describe the clock characteristics, timing exceptions, and signal transition arrival and required times. You can specify timing constraints in the Synopsys Design Constraints (**.sdc**) file format using the GUI or command-line interface. The Quartus II Fitter optimizes the placement of logic to meet your constraints.

During timing analysis, the Quartus II TimeQuest Timing Analyzer analyzes the timing paths in the design, calculates the propagation delay along each path, checks for timing constraint violations, and reports timing results as slack in the Report pane and in the Console pane. If the Quartus II TimeQuest Timing Analyzer reports any timing violations, you can customize the reporting to view precise timing information about specific paths, and then constrain those paths to correct the violations. When your design is free of timing violations, you can be confident that the logic will operate as intended in the target device.

The Quartus II TimeQuest Timing Analyzer is a complete static timing analysis tool that you can use as a sign-off tool for Altera® FPGAs and HardCopy® ASICs.

This chapter contains the following sections:

■ "The TimeQuest Timing Analyzer GUI" on page 7–78

■ "Conclusion" on page 7–88

For more information about the TimeQuest Timing Analyzer and the SOPC Builder, refer to *Volume 4: SOPC Builder* in the *Quartus II Handbook*.

Getting Started with the Quartus II TimeQuest Timing Analyzer

The Quartus II TimeQuest Timing Analyzer caters to the needs of the most basic to the most advanced designs for FPGAs.

This section provides a brief overview of the Quartus II TimeQuest Timing Analyzer, including the necessary steps to properly constrain a design, perform a full place-and-route, and perform reporting on the design.

Setting Up the Quartus II TimeQuest Timing Analyzer

The Quartus II software version 7.2 and later supports two native timing analysis tools: Quartus II TimeQuest Timing Analyzer and Quartus II Classic Timing Analyzer. When you specify the Quartus II TimeQuest Timing Analyzer as the default timing analysis tool, the Quartus II TimeQuest Timing Analyzer guides the Fitter and analyzes timing results after compilation.

To specify the Quartus II TimeQuest Timing Analyzer as the default timing analyzer, on the Assignments menu, click **Settings**. In the **Settings** dialog box, in the **Category** list, select **Timing Analysis Settings** and turn on **Use TimeQuest Timing Analyzer during compilation**.

To add the TimeQuest icon to the Quartus II toolbar, on the Tools menu, click **Customize**. In the Customize dialog box, click the **Toolbars** tab, turn on **Processing**, and click **Close**.

Compilation Flow with the Quartus II TimeQuest Timing Analyzer Guidelines

When you enable the Quartus II TimeQuest Timing Analyzer as the default timing analyzer, everything from constraint validation to timing verification is performed by the Quartus II TimeQuest Timing Analyzer. Figure 7–1 shows the recommended design flow steps to maximize and leverage the benefits the Quartus II TimeQuest Timing Analyzer. Details about each step are provided after the figure.

Figure 7–1. Design Flow with the Quartus II TimeQuest Timing Analyzer
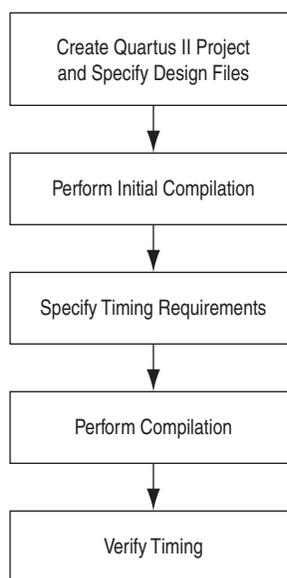
- **Create Quartus II Project and Specify Design Files**—Creates a project before you can compile design files. In this step you specify the target FPGA, any EDA tools used in the design cycle, and all design files.

    You can also modify existing design files for design optimization and add additional design files. For example, you can add HDL files or schematics to the project.

- **Perform Initial Compilation**—Creates an initial design database before you specify timing constraints for your design. Perform Analysis and Synthesis to create a post-map database, or perform a full compilation to create a post-fit database.

    Creating a post-map database for the initial compilation is faster than creating a post-fit database. A post-map database is sufficient for the initial database.

    Creating a post-fit database is recommended only if you previously created and specified an **.sdc** file for the project. A post-map database is sufficient for the initial compilation.

- **Specify Timing Requirements**—Timing requirements guide the Fitter as it places and routes your design.

    You must enter all timing constraints and exceptions in an **.sdc** file. This file must be included as part of the project. To add this file to your project, on the Project menu, click **Add/Remove Files in Project** and add the **.sdc** file in the **Files** dialog box.

- **Perform Compilation**—Synthesizes, places, and routes your design into the target FPGA.

    When compilation is complete, the TimeQuest Timing Analyzer generates summary clock setup and clock hold, recovery, and removal reports for all defined clocks in the design.

■ **Verify Timing**—Verifies timing in your design with the Quartus II TimeQuest Timing Analyzer. Refer to "The Quartus II TimeQuest Timing Analyzer Flow Guidelines" on page 7–21.

Running the Quartus II TimeQuest Timing Analyzer

You can run the Quartus II TimeQuest Timing Analyzer in one of the following modes:

■ Directly from the Quartus II software

■ Stand-alone mode

■ Command-line mode

This section describes each of the modes, and the behavior of the Quartus II TimeQuest Timing Analyzer.

Directly from the Quartus II Software

To run the Quartus II TimeQuest Timing Analyzer from the Quartus II software, on the Tools menu, click **TimeQuest Timing Analyzer**. The Quartus II TimeQuest Timing Analyzer is available after you have created a database for the current project. The database can be either a post-map or post-fit database; perform Analysis and Synthesis to create a post-map database, or a full compilation to create a post-fit database.

☞ After a database is created, you can create a timing netlist based on that database. If you create a post-map database, you cannot create a post-fit timing netlist in the Quartus II TimeQuest Timing Analyzer.

When you launch the TimeQuest Timing Analyzer directly from the Quartus II software, the current project opens by default.

Stand-Alone Mode

To run the Quartus II TimeQuest Timing Analyzer in stand-alone mode, type the following command at the command prompt:

```
quartus_staw ↵
```

In stand-alone mode, you can perform static analysis on any project that contains either a post-map or post-fit database. To open a project, double-click **Open Project** in the Tasks pane.

Command-Line Mode

Use command-line mode for easy integration with scripted design flows. Using the command-line mode avoids interaction with the user interface provided by the Quartus II TimeQuest Timing Analyzer, but allows the automation of each step of the static timing analysis flow. Table 7–1 provides a summary of the options available in the command-line mode.

Table 7–1. Summary of Command Line Options

Command Line Option	Description
`-h	--help`	Provides help information on `quartus_sta`.
`-t <script file>	--script=<script file>`	Sources the *<script file>*.
`-s	--shell`	Enters shell mode.
`--tcl_eval <tcl command>`	Evaluates the Tcl command *<tcl command>*.

`--do_report_timing`	For all clocks in the design, run the following commands:
	`report_timing -npaths 1 -to_clock $clock`
	`report_timing -setup -npaths 1 -to_clock $clock`
	`report_timing -hold -npaths 1 -to_clock $clock`
	`report_timing -recovery -npaths 1 -to_clock $clock`
	`report_timing -removal -npaths 1 -to_clock $clock`

`--force_dat`	Forces the Delay Annotator to annotate the new delays from the recently compiled design to the compiler database.
`--lower_priority`	Lowers the computing priority of the `quartus_sta` process.
`--post_map`	Uses the post-map database results.
`--qsf2sdc`	Converts assignments from the Quartus II Settings File (**.qsf**) format to the Synopsys Design Constraints File format.
`--sdc=<SDC file>`	Specifies the **.sdc** file to read.
`--fast_model`	Uses the fast corner delay models.
`--report_script=<script>`	Specifies a custom report script to be called.
`--speed=<value>`	Specifies the device speed grade to be used for timing analysis.
`--tq2hc`	Generate temporary files to convert the Quartus II TimeQuest Timing Analyzer **.sdc** file(s) to a PrimeTime **.sdc** file that can be used by the HardCopy Design Center (HCDC).
`--tq2pt`	Generates temporary files to convert the Quartus II TimeQuest Timing Analyzer **.sdc** file(s) to a PrimeTime **.sdc** file.
`-f <argument file>`	Specifies a file containing additional command-line arguments.
`-c <revision name>	--rev=<revision_name>`	Specifies which revision and its associated Quartus II Settings File (**.qsf**) to use.
`--multicorner`	Specifies that all slack summary reports be generated for both slow and fast corners.

To run the Quartus II TimeQuest Timing Analyzer in command-line mode, type the following command at the command prompt:

`quartus_sta <options>` ↵

Timing Analysis Overview

This section provides an overview of the Quartus II TimeQuest Timing Analyzer concepts. Understanding these concepts allows you to take advantage of the powerful timing analysis features available in the Quartus II TimeQuest Timing Analyzer.

The Quartus II TimeQuest Timing Analyzer follows the flow shown in Figure 7–2 when it analyzes your design. Table 7–2 lists the most commonly used commands for each step.

Figure 7–2. The Quartus II TimeQuest Timing Analyzer Flow

```
                        ┌─────────────────────────────┐
                        │        Open Project         │
                        │        project_open         │
                        └─────────────────────────────┘
                                       │
                                       ▼
                        ┌─────────────────────────────┐
                        │     Create Timing Netlist    │
                        │     create_timing_netlist    │
                        └─────────────────────────────┘
                                       │
                                       ▼
                        ┌─────────────────────────────┐
                        │       Constrain the Design   │
                        │  create_clock   create_generated_clock │
                        │  set_clock_uncertainty  derive_pll_clocks │
                        │  set_clock_latency   set_input_delay │
                        │                      set_output_delay │
                        │                              ...      │
                        └─────────────────────────────┘
                                       │
                                       ▼
                        ┌─────────────────────────────┐
                        │     Update Timing Netlist    │
                        │     update_timing_netlist    │
                        └─────────────────────────────┘
                                       │
                                       ▼
                        ┌─────────────────────────────┐
                        │    Verify Static Timing Analysis │
                        │              Results         │
                        │  report_clocks_transfers   report_sdc │
                        │  report_min_pulse_width   report_timing │
                        │  report_net_timing    report_clocks │
                        │                  report_min_pulse_width │
                        │                           report_ucp │
                        └─────────────────────────────┘
```

Table 7–2 describes Quartus II TimeQuest Timing Analyzer terminology.

**Table 7–2.** Quartus II TimeQuest Timing Analyzer Terms

Terminology	Definition
Nodes	Most basic timing netlist unit. Use to represent ports, pins, and registers.
Keepers	Ports or registers. *(1)*
Cells	Look-up table (LUT), registers, digital signal processing (DSP) blocks, TriMatrix memory, IOE, and so on. *(2)*
Pins	Inputs or outputs of cells.
Nets	Connections between pins.
Ports	Top-level module inputs or outputs; for example, device pins.
Clocks	Abstract objects outside of the design.

Notes to Table 7–2:

(1) Pins can indirectly refer to keepers. For example, when the value in the `-from` field of a constraint is a clock pin to a dedicated memory. In this case, the clock pin refers to a collection of registers.

(2) For Stratix® devices and other early device families, the LUT and registers are contained in logic elements (LE) and act as cells for these device families.

The Quartus II TimeQuest Timing Analyzer requires a timing netlist before it can perform a timing analysis on any design. For example, for the design shown in Figure 7–3, the Quartus II TimeQuest Timing Analyzer generates a netlist equivalent to the one shown in Figure 7–4.

Figure 7–3. Sample Design
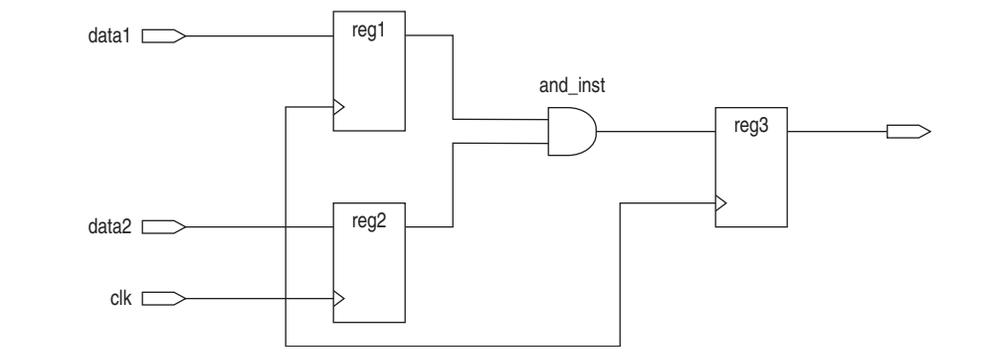
**Figure 7–4.** The Quartus II TimeQuest Timing Analyzer Timing Netlist
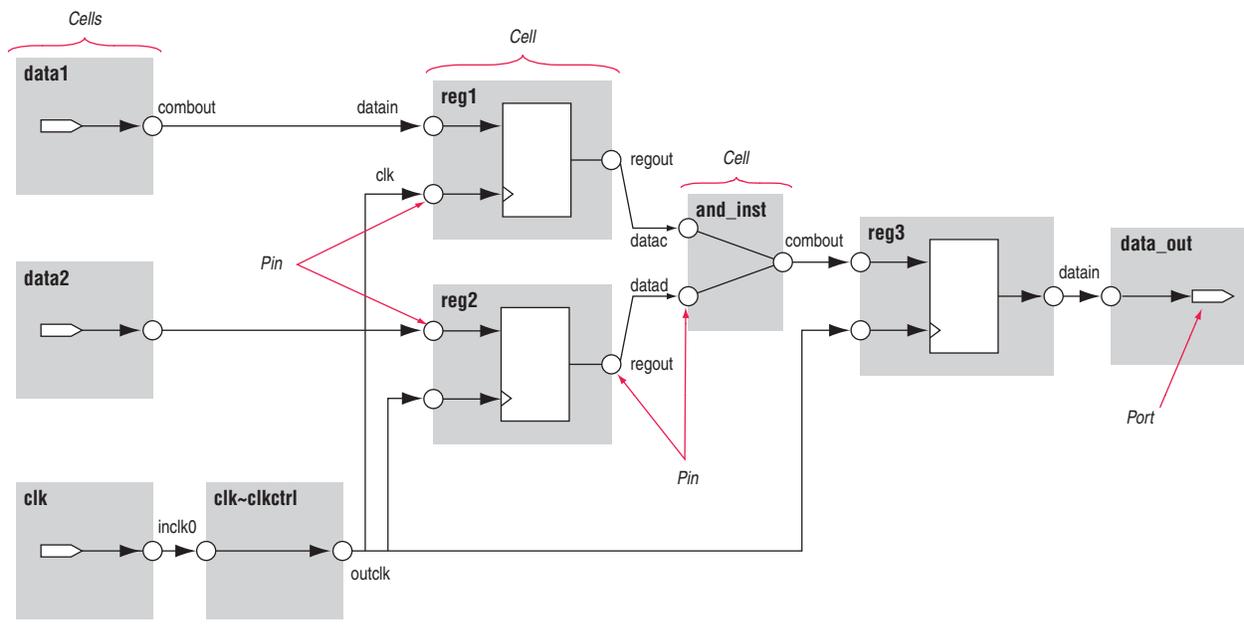
Figure 7–4 shows various cells, pins, nets, and ports. The following sample cell names are included:

■ `reg1`

■ `reg2`

■ `and_inst`

The following sample pins names are included:

■ `data1|combout`

■ `reg1|regout`

■ `and_inst|combout`

The following net names are included:

- `data1~combout`
- `reg1`
- `and_inst`

The following port names are included:

- `data1,clk`
- `data_out`

Paths connect two design nodes, such as the output of a register to the input of another register. Timing paths play a significant role in timing analysis. Understanding the types of timing paths is important to timing closure and optimization. The following list shows some of the commonly analyzed paths that are described in this section:

- **Edge paths**—the connections from ports-to-pins, from pins-to-pins, and from pins-to-ports.
- **Clock paths**—the edges from device ports or internally generated clock pins to the clock pin of a register.
- **Data paths**—the edges from a port or the data output pin of a sequential element to a port or the data input pin of another sequential element.
- **Asynchronous paths**—the edges from a port or sequential element to the asynchronous set or clear pin of a sequential element.

Figure 7–5 shows some of these commonly analyzed path types.

Figure 7–5. Path Types
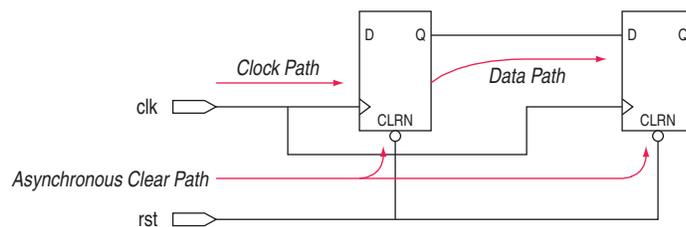
After the Quartus II TimeQuest Timing Analyzer identifies the path type, it can report data and clock arrival times for valid register-to-register paths. The Quartus II TimeQuest Timing Analyzer calculates data arrival time by adding the delay from the clock source to the clock pin of the source register, the micro clock-to-out (μt_{CO}) of the source register, and the delay from the source register's Q pin to the destination register's D pin, where the μt_{CO} is the intrinsic clock-to-out for the internal registers in the FPGA.

The Quartus II TimeQuest Timing Analyzer calculates clock arrival time by adding the delay from the clock source to the destination register's clock pin. Figure 7–6 shows a data arrival path and a clock arrival path. The Quartus II TimeQuest Timing Analyzer calculates data required time by accounting for the clock arrival time and micro setup time (μt_{SU}) of the destination register, where the μt_{SU} is the intrinsic setup for the internal registers in the FPGA.

**Figure 7–6.** Data Arrival and Clock Arrival
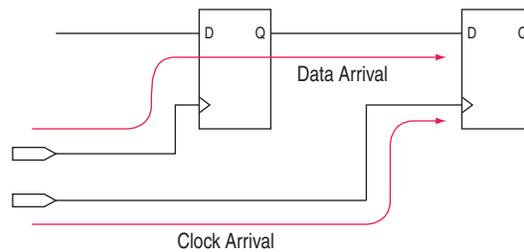
In addition to identifying various paths in a design, the Quartus II TimeQuest Timing Analyzer analyzes clock characteristics to compute the worst-case requirement between any two registers in a single register-to-register path. You should constrain all clocks in your design before performing this analysis.

The launch edge is an active clock edge that sends data out of a sequential element, acting as a source for the data transfer. A latch edge is the active clock edge that captures data at the data port of a sequential element, acting as a destination for the data transfer.

Figure 7–7 shows a single-cycle system that uses consecutive clock edges to transfer and capture data, a register-to-register path, and the corresponding launch and latch edges timing diagram. In this example, the launch edge sends the data out of register reg1 at 0 ns, and register reg2 latch edge captures the data at 5 ns.

Figure 7–7. Launch Edge and Latch Edge
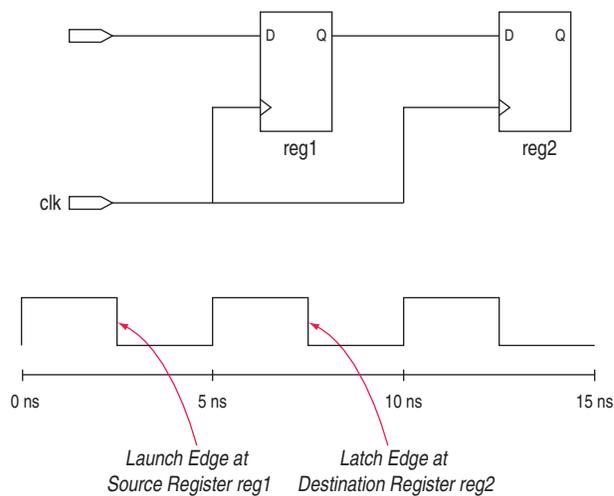
The Quartus II TimeQuest Timing Analyzer validates clock setup and hold requirements relative to the launch and latch edges.

Clock Analysis

A comprehensive static timing analysis includes analysis of register-to-register, I/O, and asynchronous reset paths. The Quartus II TimeQuest Timing Analyzer uses data required times, data arrival times, and clock arrival times to verify circuit performance and detect possible timing violations. The Quartus II TimeQuest Timing Analyzer determines the timing relationships that must be met for the design to correctly function and checks arrival times against required times to verify timing.

Clock Setup Check

To perform a clock setup check, the Quartus II TimeQuest Timing Analyzer determines a setup relationship by analyzing each launch and latch edge for each register-to-register path. For each latch edge at the destination register, the Quartus II TimeQuest Timing Analyzer uses the closest previous clock edge at the source register as the launch edge. In Figure 7–8, two setup relationships are defined and are labeled setup A and setup B. For the latch edge at 10 ns, the closest clock that acts as a launch edge is at 3 ns and is labeled setup A. For the latch edge at 20 ns, the closest clock that acts as a launch edge is 19 ns and is labeled setup B.

Figure 7–8. Setup Check
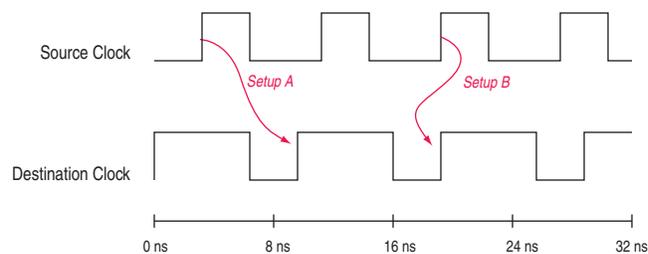
The Quartus II TimeQuest Timing Analyzer reports the result of clock setup checks as slack values. Slack is the margin by which a timing requirement is met or not met. Positive slack indicates the margin by which a requirement is met; negative slack indicates the margin by which a requirement is not met. The Quartus II TimeQuest Timing Analyzer determines clock setup slack, as shown in Equation 7–1, for internal register-to-register paths.

Equation 7–1.

Clock Setup Slack $=$ Data Required Time $-$ Data Arrival Time

Data Arrival Time $=$ Launch Edge $+$ Clock Network Delay to Source Register $+$

    $\mu\, t_{CO} +$ Register-to-Register Delay

Data Required $=$ Clock Arrival Time $- \mu\, t_{SU} -$ Setup Uncertainty

Clock Arrival Time $=$ Latch Edge $+$ Clock Network Delay to Destination Register

If the data path is from an input port to a internal register, the Quartus II TimeQuest Timing Analyzer uses the equations shown in Equation 7–2 to calculate the setup slack time.

Equation 7–2.

Clock Setup Slack Time $=$ Data Required Time $-$ Data Arrival Time

Data Arrival Time $=$ Launch Edge $+$ Clock Network Delay $+$

   Input Maximum Delay of Pin $+$ Pin-to-Register Delay

Data Required Time $=$ Latch Edge $+$ Clock Network Delay to Destination Register $-\mu t_{SU}$

If the data path is an internal register to an output port, the Quartus II TimeQuest Timing Analyzer uses the equations shown in Equation 7–3 to calculate the setup slack time.

Equation 7–3.

Clock Setup Slack Time $=$ Data Required Time $-$ Data Arrival Time

Data Arrival Time $=$ Launch Edge $+$ Clock Network Delay to Source Register $+$

   $\mu t_{CO} +$ Register-to-Pin Delay

Data Required Time $=$ Latch Edge $+$ Clock Network Delay $-$ Output Maximum Delay of Pin

Clock Hold Check

To perform a clock hold check, the Quartus II TimeQuest Timing Analyzer determines a hold relationship for each possible setup relationship that exists for all source and destination register pairs. The Quartus II TimeQuest Timing Analyzer checks all adjacent clock edges from all setup relationships to determine the hold relationships. The Quartus II TimeQuest Timing Analyzer performs two hold checks for each setup relationship. The first hold check determines that the data launched by the current launch edge is not captured by the previous latch edge. The second hold check determines that the data launched by the next launch edge is not captured by the current latch edge. Figure 7–9 shows two setup relationships labeled setup A and setup B. The first hold check is labeled hold check A1 and hold check B1 for setup A and setup B, respectively. The second hold check is labeled hold check A2 and hold check B2 for setup A and setup B, respectively.

Figure 7–9. Hold Checks
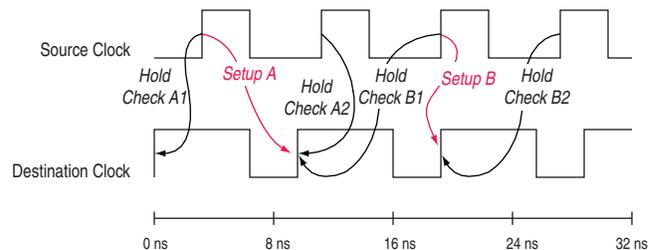
From the possible hold relationships, the Quartus II TimeQuest Timing Analyzer selects the hold relationship that is the most restrictive. The hold relationship with the largest difference between the latch and launch edges (that is, latch – launch and not the absolute value of latch and launch) is selected because this determines the minimum allowable delay for the register-to-register path. For Figure 7–9, the hold relationship selected is hold check A2.

The Quartus II TimeQuest Timing Analyzer determines clock hold slack as shown in Equation 7–4.

Equation 7–4.

Clock Hold Slack $=$ Data Arrival Time $-$ Data Required Time

Data Arrival Time $=$ Launch Edge $+$ Clock Network Delay to Source Register $+ \mu t_{CO} +$

Register-to-Register Delay

Data Required Time $=$ Clock Arrival Time $+ \mu t_{H} +$ Hold Uncertainty

Clock Arrival Time $=$ Latch Edge $+$ Clock Network Delay to Destination Register

If the data path is from an input port to an internal register, the Quartus II TimeQuest Timing Analyzer uses the equations shown in Equation 7–5 to calculate the hold slack time.

Equation 7–5.

Clock Hold Slack Time $=$ Data Arrival Time $-$ Data Required Time

Data Arrival Time $=$ Launch Edge $+$ Clock Network Delay $+$

Input Minimum Delay of Pin $+$ Pin-to-Register Delay

Data Required Time $=$ Latch Edge $+$ Clock Network Delay to Destination Register $+ \mu t_{H}$

If the data path is an internal register to an output port, the Quartus II TimeQuest Timing Analyzer uses the equations shown in Equation 7–6 to calculate the setup hold time.

Equation 7–6.

Clock Hold Slack Time $=$ Data Arrival Time $-$ Data Required Time

Data Arrival Time $=$ Latch Edge $+$ Clock Network Delay to Source Register $+ \mu t_{CO} +$

Register-to-Pin Delay

Data Required Time $=$ Latch Edge $+$ Clock Network Delay $-$ Output Minimum Delay of Pin

Recovery and Removal

Recovery time is the minimum length of time the de-assertion of an asynchronous control signal; for example, `clear` and `preset`, must be stable before the next active clock edge. The recovery slack time calculation is similar to the clock setup slack time calculation, but it applies to asynchronous control signals. If the asynchronous control signal is registered, the Quartus II TimeQuest Timing Analyzer uses Equation 7–7 to calculate the recovery slack time.

Equation 7–7.

Recovery Slack Time $=$ Data Required Time $-$ Data Arrival Time

Data Arrival Time $=$ Launch Edge $+$ Clock Network Delay to Source Register $+$

$\mu t_{CO} +$ Register-to-Register Delay

Data Required Time $=$ Latch Edge $+$ Clock Network Delay to Destination Register $- \mu t_{SU}$

If the asynchronous control is not registered, the Quartus II TimeQuest Timing Analyzer uses the equations shown in Equation 7–8 to calculate the recovery slack time.

Equation 7–8.

Recovery Slack Time $=$ Data Required Time $-$ Data Arrival Time

Data Arrival Time $=$ Launch Edge $+$ Clock Network Delay $+$ Maximum Input Delay $+$

   Port-to-Register Delay

Data Required Time $=$ Latch Edge $+$ Clock Network Delay to Destination Register Delay $- \mu t_{SU}$

☞ If the asynchronous reset signal is from a port (device I/O), you must make an Input Maximum Delay assignment to the asynchronous reset port for the Quartus II TimeQuest Timing Analyzer to perform recovery analysis on that path.

Removal time is the minimum length of time the de-assertion of an asynchronous control signal must be stable after the active clock edge. The Quartus II TimeQuest Timing Analyzer removal time slack calculation is similar to the clock hold slack calculation, but it applies asynchronous control signals. If the asynchronous control is registered, the Quartus II TimeQuest Timing Analyzer uses the equations shown in Equation 7–9 to calculate the removal slack time.

Equation 7–9.

Removal Slack Time $=$ Data Arrival Time $-$ Data Required Time

Data Arrival Time $=$ Launch Edge $+$ Clock Network Delay to Source Register $+$

   μt_{CO} of Source Register $+$ Register-to-Register Delay

Data Required Time $=$ Latch Edge $+$ Clock Network Delay to Destination Register $+ \mu t_{H}$

If the asynchronous control is not registered, the Quartus II TimeQuest Timing Analyzer uses the equations shown in Equation 7–10 to calculate the removal slack time.

Equation 7–10.

Removal Slack Time $=$ Data Arrival Time $-$ Data Required Time

Data Arrival Time $=$ Launch Edge $+$ Clock Network Delay $+$ Input Minimum Delay of Pin $+$

   Minimum Pin-to-Register Delay

Data Required Time $=$ Latch Edge $+$ Clock Network Delay to Destination Register $+ \mu t_{H}$

☞ If the asynchronous reset signal is from a device pin, you must specify the Input Minimum Delay constraint to the asynchronous reset pin for the Quartus II TimeQuest Timing Analyzer to perform a removal analysis on this path.

Multicycle Paths

Multicycle paths are data paths that require more than one clock cycle to latch data at the destination register. For example, a register may be required to capture data on every second or third rising clock edge. Figure 7–10 shows an example of a multicycle path between a multiplier's input registers and output register where the destination latches data on every other clock edge.

Figure 7–10. Example Diagram of a Multicycle Path
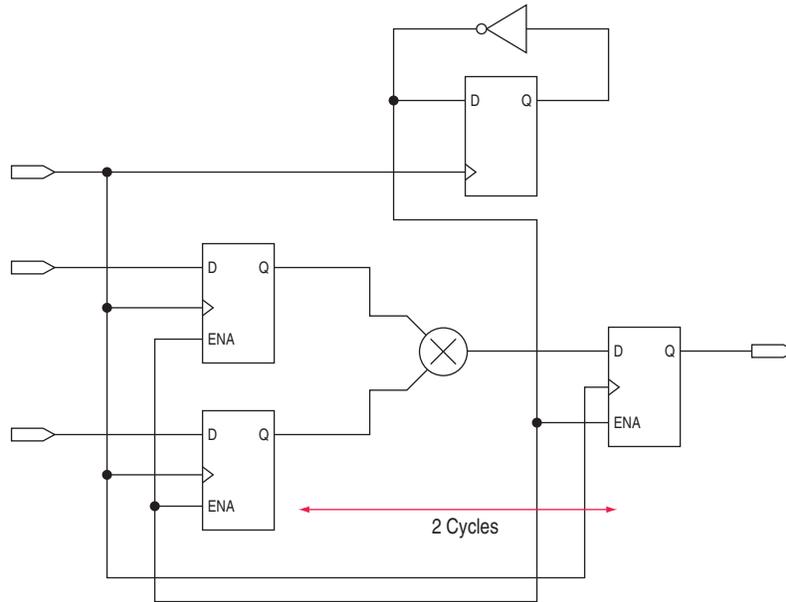
Figure 7–11 shows a register-to-register path where the source clock, src_clk, has a period of 10 ns and the destination clock, dst_clk, has a period of 5 ns.

Figure 7–11. Register-to-Register Path
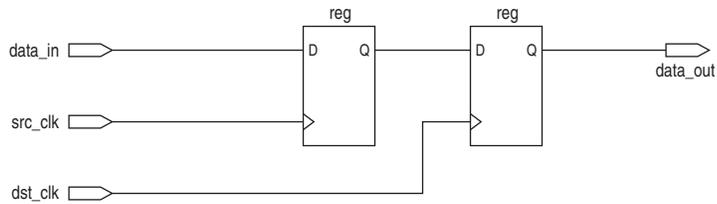
Figure 7–12 shows the respective timing diagrams for the source and destination clocks and the default setup and hold relationships. The default setup relationship is 5 ns; the default hold relationship is 0 ns.

Figure 7–12. Default Setup and Hold Timing Diagram
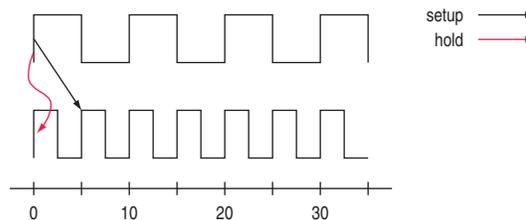
The default setup and hold relationships can be modified with the set_multicycle_path command to accommodate the system requirements.

Table 7–3 shows the commands used to modify either the launch or latch edge times that the Quartus II TimeQuest Timing Analyzer Timing Analyzer uses to determine a setup relationship or hold relationship.

Table 7–3. Commands to Modify Edge Times

Command	Description of Modification
`set_multicycle_path -setup -end`	Latch edge time of the setup relationship
`set_multicycle_path -setup -start`	Launch edge time of the setup relationship
`set_multicycle_path -hold -end`	Latch edge time of the hold relationship
`set_multicycle_path -hold -start`	Launch edge time of the hold relationship

Figure 7–13 shows the timing diagram after a multicycle setup of two has been applied. The command moves the latch edge time to 10 ns from the default 5 ns.

**Figure 7–13.** Modified Setup Diagram
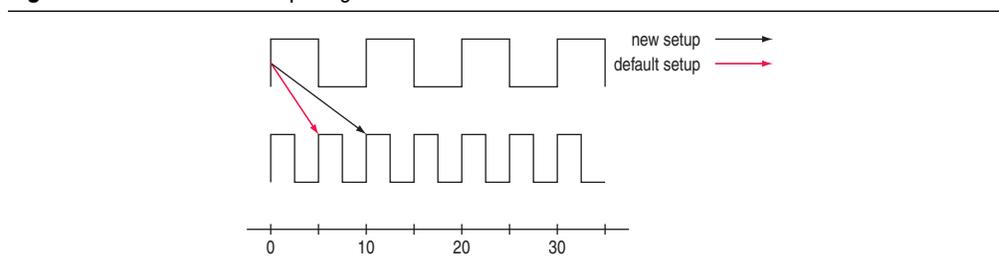
Metastability

All registers have defined setup and hold time requirements that allow the registers to capture data at the inputs. The captured data is then sent to the output of the register, which has a high- or low-voltage signal, depending on the signal captured at the input. However, if the data violates the setup and/or hold times, the output of the register may go into a metastable state. In this state, the output of a register hovers at a value between the high and low states. When this value propagates through the circuit, registers may latch the wrong value, which may cause the system to fail.

Metastability problems commonly occur when a data signal is transferred between two sets of circuitry that are in unrelated clock domains. To minimize the failures due to metastability, circuit designers generally use a sequence of back-to-back registers (synchronization register chains) in the destination clock domain to resynchronize the data signals to the new clock domain.

Synchronization register chains can be defined as a sequence of registers, that meet the following requirements:

■ The registers are all clocked by the same or phase-related clocks

■ The first register is driven from an unrelated clock domain, or asynchronously, and the fanout of the register is 1

Figure 7–14 shows a sample synchronization register chain.

Figure 7–14. Synchronization Chain
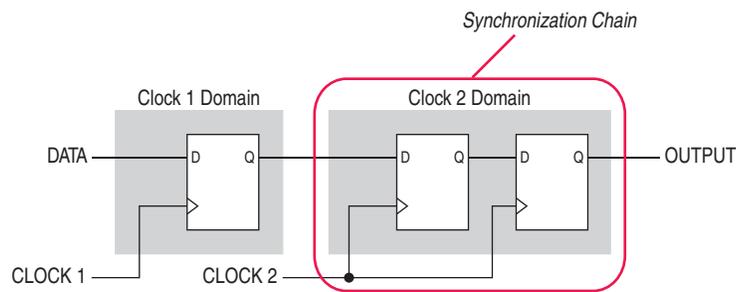
The length of the synchronization register chain is defined by the number of registers in this synchronizing clock domain that meets the above requirements.

☞ Logic can exist between these synchronization registers, as long as each net within the synchronizing clock domain has a valid setup slack.

An example of what is considered to be a synchronization register chain is shown in Figure 7–15.

Figure 7–15. Synchronization Register Chain
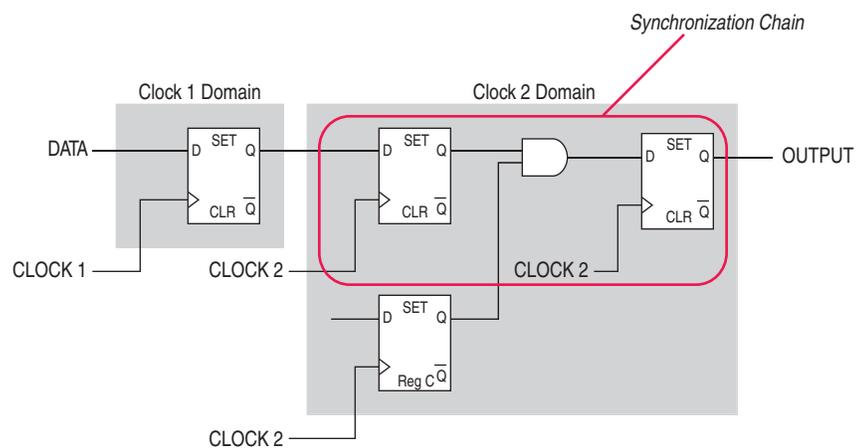
When metastability occurs the clock-to-out time can be extended beyond the register's nominal t_{CO} time. The additional time beyond the t_{CO} time for the output signal to settle to a known state is known as the settling time (t_{MET}). The t_{MET} of a synchronization register chain is the sum of all output slacks of the registers in the chain. Figure 7–16 shows the t_{MET} for a metastable signal. If the metastable event takes longer to resolve than the settling time of a synchronization chain, this metastable event may be propagated forward, and cause the design to fail.

Figure 7–16. Metastability Timing Parameters
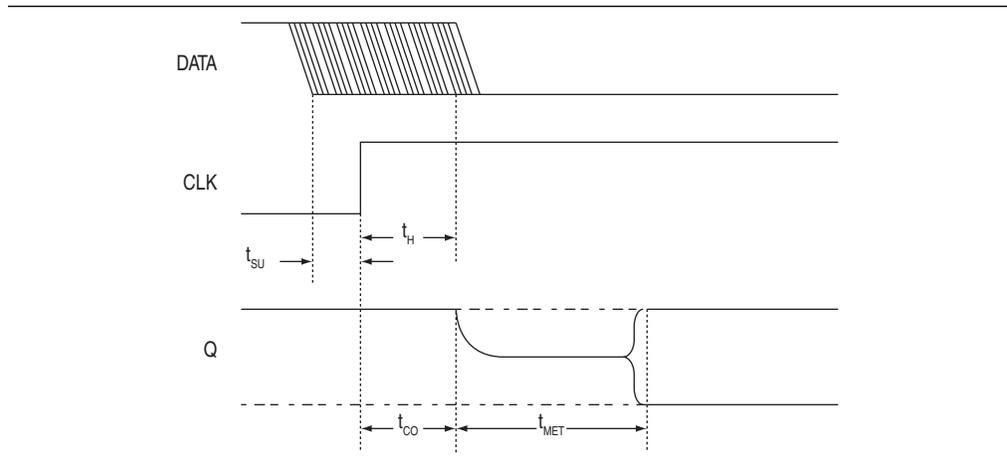
Based on the t_{MET} calculated, the Mean Time Between Failure (MTBF) of the synchronization chain can be calculated as shown in Equation 7–11.

Equation 7–11.

$$MTBF = \frac{e^{(C_2 \times t_{MET})}}{C_1 \times f_{CLOCK} \times f_{DATA}}$$

The MTBF is an estimate of the average time between instances of failure. The constants **C1** and **C2** are determined by the process technology used to manufacture the device and are specified by the TimeQuest Timing Analyzer for the appropriate device.

The TimeQuest Timing Analyzer analyzes the robustness of the design for metastability by computing the MTBF for each synchronization register chain in the design. The MTBF of the entire design is then estimated based on the synchronization chains it contains.

In addition to reporting synchronization register chains found in the design, the Quartus II software also protects these registers from optimizations that might negatively impact MTBF, such as register duplication and logic retiming. The Quartus II software can also optimize the MTBF of your design if the MTBF is too low.

Refer to "report_metastability" on page 7–55 for information about how to enable metastability analysis and report metastability in the TimeQuest Timing Analyzer.

For more information about enabling metastability analysis and reporting metastability in TimeQuest, refer to the *Area and Timing Optimization* chapter of volume 2 in the Quartus II Handbook.

Common Clock Path Pessimism

Common clock path pessimism (CCPP) removal accounts for the minimum and maximum delay variation associated with common clock paths during a static timing analysis. CCPP removal accounts for this variation by adding the difference between the maximum and minimum delay value of the common clock path to the appropriate slack equation.

The minimum and maximum delay variation might arise when two different delay values are used for the same clock path. For example, in a simple setup analysis, the maximum clock path delay to the source register is used to determine the data arrival time. The minimum clock path delay to the destination register is used to determine the data required time. However, if the clock path to the source register and to the destination register share a common clock path, the analysis uses both the maximum delay and the minimum delay to model the common clock path. This results in an overly pessimistic analysis since two different delay values, the maximum and minimum delays, cannot be used to model the same clock path.

Figure 7–17 shows a typical register-to-register path with the maximum and minimum delay values shown.

**Figure 7–17.** Common Clock Path
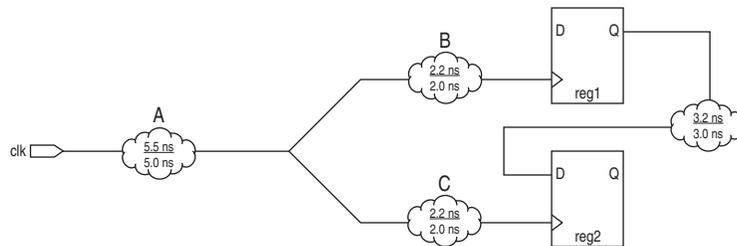
Segment A is the common clock path between reg1 and reg2. The minimum delay is 5.0 ns; the maximum delay is 5.5 ns. The difference between the maximum and minimum delay value equals the CCPP removal value; in this case, CCPP equals 0.5 ns. The CCPP removal value is then added to the appropriate slack equation. Therefore, if the setup slack for the register-to-register in Figure 7–17 equals 0.7 ns *without* CCPP removal, the slack would be 1.2 ns *with* CCPP removal.

CCPP is also used when determining the minimum pulse width of a register. A clock signal must meet a register's minimum pulse width requirement to be recognized by the register. A minimum high time defines the minimum pulse width for a positive-edge triggered register. A minimum low time defines the minimum pulse width for a negative-edge triggered register.

Clock pulses that violate the minimum pulse width of a register prevent data from being latched at the data pin of the register. To calculate the slack of the minimum pulse width, the required minimum pulse width time is subtracted from the actual minimum pulse width time. The actual minimum pulse width time is determined by the clock requirement specified for the clock that feeds the clock port of the register. The required minimum pulse width time is determined by the maximum rise, minimum rise, maximum fall, and minimum fall times. Figure 7–18 shows a diagram of the required minimum pulse width time for both the high pulse and low pulse.

Figure 7–18. Required Minimum Pulse Width
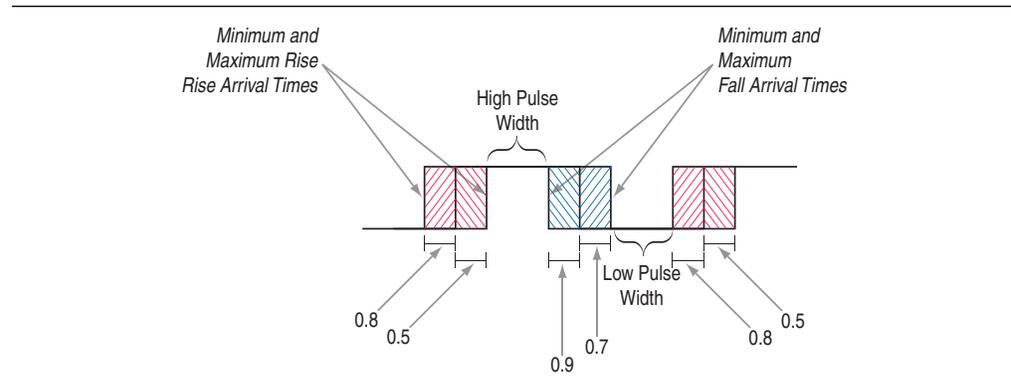
With CCPP, the minimum pulse width slack can be increased by the smallest value of either the maximum rise time minus the minimum rise time, or the maximum fall time minus the minimum fall time. For Figure 7–18, the slack value can be increased by 0.2 ns, which is the smallest value between 0.3 ns (0.8 ns – 0.5 ns) and 0.2 ns (0.9 ns – 0.7 ns).

Refer to "report_min_pulse_width" on page 7–59 for more information about reporting CCPP in the TimeQuest Timing Analyzer.

You must use the **Enable common clock path pessimism removal** option to account for CCPP in the Fitter and timing analysis. This option defaults to ON for Stratix III, Cyclone III, and newer device families.

To access this option, perform the following steps:

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.

2. In the **Category** list, next to **Timing Analysis Settings**, click the "+" icon to expand the menu. Click **TimeQuest settings**.

3. Turn **Enable common clock path pessimism removal** on.

4. Click **OK**.

Clock-As-Data

The majority of FPGA designs contain simple connections between any two nodes known as either a data path or a clock path. A data path is a connection between the output of a synchronous element to the input of another synchronous element. A clock is a connection to the clock pin of a synchronous element. However, as FPGA designs become more complex, such as using source-synchronous interfaces, this simplified view is no longer sufficient.

The connection between port `clk_in` and port `clk_out` can be treated either as a clock path or a data path. The clock path is from the port `clk_in` to the register `reg_data` clock pin. The data path is from port `clk_in` to the port `clk_out`. In the design shown in Figure 7–19, the path from port `clk_in` to port `clk_out` is both a clock and a data path.

Figure 7–19. Simplified Source Synchronous Output
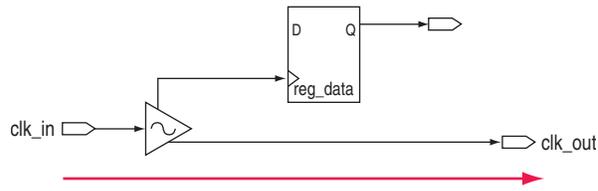
With clock-as-data analysis, the TimeQuest Timing Analyzer provides a more accurate analysis of the path based on the user constraints. For the clock path analysis, any phase shift associated with the PLL is taken into consideration. For the data path, any phase shift associated with the PLL is taken into consideration instead of being ignored.

The clock-as-data analysis also applies to internally generated clock dividers similar to Figure 7–20.

Figure 7–20. Clock Divider   *(Note 1)*
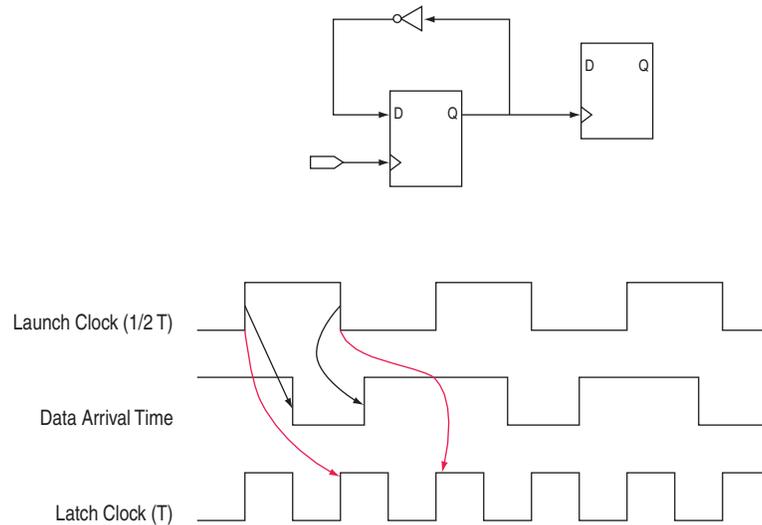
Note to Figure 7–20:

(1)   In this figure, the inverter feedback path is analyzed during timing analysis. The output of the divider register is used to determine the launch time and the clock port of the register is used to determine the latch time.

A source-synchronous interface contains a clock signal that travels in parallel with data signals. The clock and data pair originates or terminates at the same device.

The Quartus II TimeQuest Timing Analyzer Flow Guidelines

Use the steps shown in Figure 7–21 to verify timing in the TimeQuest Timing Analyzer.

Figure 7–21. Timing Verification in the TimeQuest Timing Analyzer

```
┌─────────────────────────┐
│  Create a Timing Netlist │
└─────────────────────────┘
            │
            ▼
┌─────────────────────────┐
│     Read .sdc File       │
└─────────────────────────┘
            │
            ▼
┌─────────────────────────┐
│   Update Timing Netlist  │
└─────────────────────────┘
            │
            ▼
┌─────────────────────────┐
│  Generate Timing Reports │
└─────────────────────────┘
```
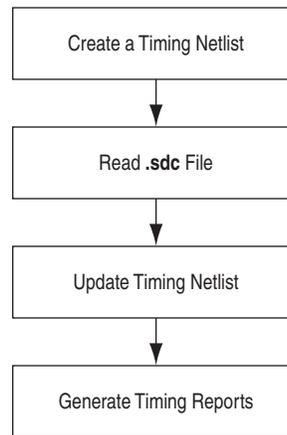
The following sections describe each of the steps shown in Figure 7–21.

Create a Timing Netlist

After you perform a full compilation, you must create a timing netlist based on the fully annotated database from the post-fit results.

To create the timing netlist, double-click Create Timing Netlist in the Tasks pane, or type the following command in the Console pane:

```
create_timing_netlist ↵
```

Read the Synopsys Design Constraints File

After you create a timing netlist, you must read an **.sdc** file. This step reads all constraints and exceptions defined in the **.sdc** file.

You can read the **.sdc** file from either the Task pane or the Console pane.

To read the **.sdc** file from the Tasks pane, double-click the **Read SDC File** command.

☞ The **Read SDC File** task reads the *<current revision>*.**sdc** file.

To read the **.sdc** file from the Console, type the following command in the Console:

```
read_sdc ↵
```

For more information about reading **.sdc** files in the TimeQuest Timing Analyzer, refer to "Synopsys Design Constraints File Precedence" on page 7–25.

Update Timing Netlist

You must update the timing netlist after you read an **.sdc** file. The TimeQuest Timing Analyzer applies all constraints to the netlist for verification and removes any invalid or false paths in the design from verification.

To update the timing netlist, double-click Update Timing Netlist in the Tasks pane, or type the following command in the Console pane:

```
update_timing_netlist ↵
```

Generate Timing Reports

You can generate timing reports for all critical paths in your design. The Tasks pane contains the commonly used reporting commands. Individual or custom reports can be generated for your design.

For more information about reporting, refer to the section "Timing Reports" on page 7–52.

For a full list of available report application program interfaces (APIs), refer to the *SDC & TimeQuest API Reference Manual*.

As you verify timing, you may encounter failures along critical paths. If this occurs, you can refine the existing constraints or create new ones to change the effects of existing constraints. If you modify, remove, or add constraints, you should perform a full compilation. This allows the Fitter to re-optimize the design based on the new constraints and brings you back to the Perform Compilation step in the process. This iterative process allows you to resolve your timing violations in the design.

For a sample Tcl script to automate the timing analysis flow, refer to the TimeQuest Quick Start Tutorial.

Collections

The Quartus II TimeQuest Timing Analyzer supports collection APIs that provide easy access to ports, pins, cells, or nodes in the design. Use collection APIs with any valid constraints or Tcl commands specified in the Quartus II TimeQuest Timing Analyzer.

Table 7–4 describes the collection commands supported by the Quartus II TimeQuest Timing Analyzer.

Table 7–4. Collection Commands (Part 1 of 2)

Command	Description
all_clocks	Returns a collection of all clocks in the design.
all_inputs	Returns a collection of all input ports in the design.
all_outputs	Returns a collection of all output ports in the design.
all_registers	Returns a collection of all registers in the design.
get_cells	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 the same time.

Table 7–4. Collection Commands (Part 2 of 2)

Command	Description
`get_clocks`	Returns a collection of clocks in the design. When used as an argument to another command, such as the `-from` or `-to` of `set_multicycle_path`, each node in the clock represents all nodes clocked by the clocks in the collection. The default uses the specific node (even if it is a clock) as the target of a command.
`get_nets`	Returns a collection of nets in the design. All net names in the collection match the specified pattern. You can use wildcards to select multiple nets at the same time.
`get_pins`	Returns a collection of pins in the design. All pin names in the collection match the specified pattern. You can use wildcards to select multiple pins at the same time.
`get_ports`	Returns a collection of ports (design inputs and outputs) in the design.

Table 7–5 describes the SDC extension collection commands supported by the Quartus II TimeQuest Timing Analyzer.

Table 7–5. SDC Extension Collection Commands

Command	Description
`get_fanouts <filter>`	Returns a collection of fan-out nodes starting from <*filter*>.
`get_keepers <filter>`	Returns a collection of keeper nodes (non-combinational nodes) in the design.
`get_nodes <filter>`	Returns a collection of nodes in the design. The `get_nodes` collection cannot be used when specifying constraints or exceptions.
`get_partitions <filter>`	Returns a collection of partitions matching the <*filter*>.
`get_registers <filter>`	Returns a collection of registers in the design.
`get_fanins <filter>`	Returns a collection of fan-in nodes starting from <*filter*>.
`derive_pll_clocks`	Automatically creates generated clocks on the outputs of the PLL. The generated clock properties reflect the PLL properties that have been specified by the MegaWizard® Plug-In Manager.
`get_assignment_groups <filter>`	Returns either a collection of keepers, ports, or registers that have been saved to the Quartus Settings File (**.qsf**) with the **Assignment (Time) Groups** option.
`remove_clock <clock list>`	Removes the list of clocks specified by <*clock list*>.
`set_scc_mode <size>`	Allows you to set the maximum Strongly Connected Components (SCC) loop size or force the Quartus II TimeQuest Timing Analyzer to always estimate delays through SCCs.
`set_time_format`	Sets time format, including time unit and decimal places.

For more information about collections, refer to the **.sdc** file and the *SDC and TimeQuest API Reference Manual*.

Application Examples

Example 7–1 shows various uses of the `create_clock` and `create_generated_clock` commands and specific design structures.

Example 7–1. create_clock and set_multicycle_path Commands and Specific Design Structures

```
# Create a simple 10 ns with clock with a 60 % duty cycle
create_clock -period 10 -waveform {0 6} -name clk [get_ports clk]
# The following multicycle applies to all paths ending at registers
# clocked by clk
set_multicycle_path -to [get_clocks clk] 2
```

SDC Constraint Files

The Quartus II TimeQuest Timing Analyzer stores all timing constraints in an **.sdc** file. You can create an **.sdc** file with different constraints for place-and-route and for timing analysis.

☞ The **.sdc** file should contain only SDC and Tcl commands. Commands to manipulate the timing netlist or control the compilation flow should not be included in the **.sdc** file.

The Quartus II software does not automatically update **.sdc** files. You must explicitly write new or updated constraints in the TimeQuest Timing Analyzer GUI. Use the write_sdc command, or, in the Quartus II TimeQuest Timing Analyzer, on the Constraints menu, click **Write SDC File** to write your constraints to an **.sdc** file.

☞ The constraints in the **.sdc** file are order-sensitive. A constraint must first be declared before any references are made to that constraint. For example, if a generated clock references a base clock with a name clk, the base clock constraint must be declared before the generated clock constraint.

Fitter and Timing Analysis with SDC Files

You can specify the same or different **.sdc** files for the Quartus II Fitter for place-and-route, and the Quartus II TimeQuest Timing Analyzer for static timing analysis. Using different **.sdc** files allows you to have one set of constraints for place-and-route and another set of constraints for final timing sign-off in the Quartus II TimeQuest Timing Analyzer.

Specifying SDC Files for Place-and-Route

To specify an **.sdc** file for the Fitter, you must add the **.sdc** file to your Quartus II project. To add the file to your project, use the following command in the Tcl console:

```
set_global_assignment -name SDC_FILE <SDC file name> ↵
```

Or, in the Quartus II software GUI, on the Project menu, click Add/Remove Files in Project.

The Fitter optimizes your design based on the requirements in the **.sdc** files in your project.

The results shown in the timing analysis report located in the Compilation Report are based on the **.sdc** files added to the project.

☞ You must specify the Quartus II TimeQuest Timing Analyzer as the default timing analyzer to make the Fitter read the **.sdc** file.

Specifying SDC Files for Static Timing Analysis

After you create a timing netlist in the Quartus II TimeQuest Timing Analyzer, you must specify timing constraints and exceptions before you can perform a timing analysis. The timing requirements do not have to be identical to those provided to the Fitter. You can specify your timing requirements manually or you can read a previously created **.sdc** file.

To manually enter your timing requirements, you can use constraint entry dialog boxes or SDC commands. If you have an **.sdc** file that contains your timing requirements, use this file to apply your timing requirements. To specify the **.sdc** file for timing analysis in the Quartus II TimeQuest Timing Analyzer, use the following command:

```
read_sdc [<SDC file name>] ↵
```

If you use the TimeQuest GUI to apply the **.sdc** file for timing analysis, in the Quartus II TimeQuest Timing Analyzer, on the Constraints menu, click **Read SDC File**.

The `read_sdc` command has the `-hdl` option, allowing `read_sdc` to read SDC commands embedded in HDL that uses that `ALTERA_ATTRIBUTE` attribute.

☞ The `read_sdc` command without any options reads both your **.sdc** files and any HDL-embedded commands. By default, the **Read SDC File** command in the Tasks pane reads the **.sdc** files specified in the Quartus II Settings File (**.qsf**), which are the same **.sdc** files used by the Fitter.

Synopsys Design Constraints File Precedence

The Quartus II Fitter and the Quartus II TimeQuest Timing Analyzer reads the **.sdc** files from the files list in the QSF file in the order they are listed, from top to bottom.

The Quartus II software searches for an **.sdc** file, as shown in Figure 7–22.

Figure 7–22. Synopsys Design Constraints File Order of Precedence
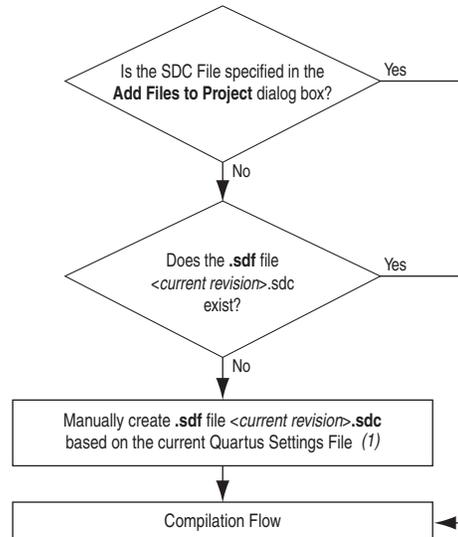
Note to Figure 7–22:

(1) This occurs only in the Quartus II TimeQuest Timing Analyzer and not during compilation in the Quartus II software. The Quartus II TimeQuest Timing Analyzer has the ability to automate the conversion of the QSF timing assignments to SDC if no **.sdc** file exists when the Quartus II TimeQuest Timing Analyzer is opened.

☞ If you type the `read_sdc` command at the command line without any arguments, the precedence order shown in Figure 7–22 is followed.

Clock Specification

The specification of all clocks and any associated clock characteristics in your design is essential for accurate static timing analysis results. The Quartus II TimeQuest Timing Analyzer supports many SDC commands that accommodate various clocking schemes and any clock characteristics.

This section describes the **.sdc** file API available to create and specify clock characteristics.

Clocks

Use the `create_clock` command to create a clock at any register, port, or pin. You can create each clock with unique characteristics. Example 7–2 shows the `create_clock` command and options.

Example 7–2. create_clock Command

```
create_clock
-period <period value>
[-name <clock name>]
[-waveform <edge list>]
[-add]
<targets>
```

Table 7–6 describes the options for the `create_clock` command.

Table 7–6. create_clock Command Options

Option	Description
`-period <period value>`	Specifies the clock period. You can also specify the clock period in units of frequency, such as `-period <num>MHz`. *(1)*
`-name <clock name>`	Name of the specific clock; for example, `sysclock`. If you do not specify the clock name, the clock name is the same as the node to which it is assigned.
`-waveform <edge list>`	Specifies the clock's rising and falling edges. The edge list alternates between rising edge and falling edge. For example, a 10 ns period where the first rising edge occurs at 0 ns and the first falling edge occurs at 5 ns would be written as `-waveform {0 5}`. The difference must be within one period unit, and the rise edge must come before the fall edge. The default edge list is `{0 <period>/2}`, or a 50% duty cycle.
`-add`	Allows you to specify more than one clock to the same port or pin.
`<targets>`	Specifies the port(s) or pin(s) to which the assignment applies. If source objects are not specified, the clock is a virtual clock. Refer to "Virtual Clocks" on page 7–29 for more information.

Note to Table 7–6:

(1) The default time unit in the Quartus II TimeQuest Timing Analyzer is nanoseconds (ns).

Example 7–3 shows how to create a 10 ns clock with a 50% duty cycle, where the first rising edge occurs at 0 ns applied to port `clk`.

Example 7–3. 100MHz Clock Creation

```
create_clock -period 10 -waveform { 0 5 } clk
```

Example 7–4 shows how to create a 10 ns clock with a 50% duty cycle that is phase shifted by 90 degrees applied to port clk_sys.

Example 7–4. 100MHz Shifted by 90 Degrees Clock Creation

```
create_clock -period 10 -waveform { 2.5 7.5 } clk_sys
```

Clocks defined with the create_clock command have a default source latency value of zero. The Quartus II TimeQuest Timing Analyzer automatically computes the clock's network latency for non-virtual clocks.

Generated Clocks

The Quartus II TimeQuest Timing Analyzer considers clock dividers, ripple clocks, or circuits that modify or change the characteristics of the incoming or master clock as generated clocks. You should define the output of these circuits as generated clocks. This definition allows the Quartus II TimeQuest Timing Analyzer to analyze these clocks and account for any network latency associated with them.

Use the create_generated_clock command to create generated clocks. Example 7–5 shows the create_generated_clock command and the available options.

Example 7–5. create_generated_clock Command

```
create_generated_clock
[-name <clock name>]
-source <master pin>
[-edges <edge list>]
[-edge_shift <shift list>]
[-divide_by <factor>]
[-multiply_by <factor>]
[-duty_cycle <percent>]
[-add]
[-invert]
[-master_clock <clock>]
[-phase <phase>]
[-offset <offset>]
<targets>
```

Table 7–7 describes the options for the create_generated_clock command.

Table 7–7. create_generated_clock Command Options (Part 1 of 2)

Option	Description
-name <clock name>	Name of the generated clock; for example, clk_x2. If you do not specify the clock name, the clock name is the same as the first node to which it is assigned.
-source <master pin>	The <master pin> specifies the node in the design from which the clock settings derive.

Table 7–7. create_generated_clock Command Options (Part 2 of 2)

Option	Description
`-edges <edge list>` | `-edge_shift <shift list>`	The `-edges` option specifies the new rising and falling edges with respect to the master clock's rising and falling edges. The master clock's rising and falling edges are numbered `1..<n>` starting with the first rising edge; for example, edge 1. The first falling edge after that is edge number 2, the next rising edge number 3, and so on. The *<edge list>* must be in ascending order. The same edge may be used for two entries to indicate a clock pulse independent of the original waveform's duty cycle.
	`-edge_shift` specifies the amount of shift for each edge in the *<edge list>*. The `-invert` option can be used to invert the clock after the `-edges` and `-edge_shifts` are applied. *(1)*
`-divide_by <factor>` | `-multiply_by <factor>`	The `-divide_by` and `-multiply_by` factors are based on the first rising edge of the clock, and extend or contract the waveform by the specified factors. For example, a `-divide_by 2` is equivalent to `-edges {1 3 5}`. For multiplied clocks, the duty cycle can also be specified. The Quartus II TimeQuest Timing Analyzer supports specifying multiply and divide factors at the same time.
`-duty_cycle <percent>`	Specifies the duty cycle of the generated clock. The duty cycle is applied last.
`-add`	Allows you to specify more than one clock to the same pin.
`-invert`	Inversion is applied at the output of the clock after all other modifications are applied, except duty cycle.
`-master_clock <clock>`	`-master_clock` is used to specify the clock if multiple clocks exist at the master pin.
`-phase <phase>`	Specifies the phase of the generated clock.
`-offset <offset>`	Specifies the offset of the generated clock.
`<targets>`	Specifies the port(s) or pin(s) to which the assignment applies.

Note to Table 7–7:

(1)   The Quartus II TimeQuest Timing Analyzer supports a maximum of three edges in the edge list.

Source latencies are based on clock network delays from the master clock (not necessarily the master pin). You can use the `set_clock_latency -source` command to override source latency.

Figure 7–23 shows how to create an inverted generated clock based on a 10 ns clock.

Figure 7–23. Generating an Inverted Clock

```
create_clock -period 10 [get_ports clk]
create_generated_clock -divide_by 1 -invert -source [get_registers clk] \
   [get_registers gen|clkreg]
```
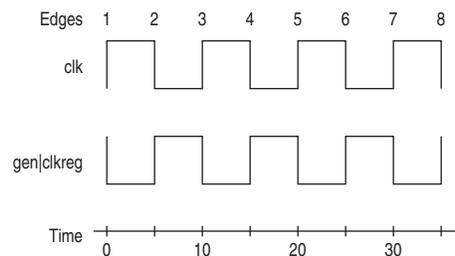
Figure 7–24 shows how to modify the generated clock using the -edges and -edge_shift options.

Figure 7–24. Edges and Edge Shifting a Generated Clock

```
create_clock -period 10 -waveform { 0 5} [get_ports clk]
# Creates a divide-by-t clock
create_generated_clock -source [get_ports clk] -edges {1 3 5 } [get_registers \
clkdivA|clkreg]
# Creates a divide-by-2 clock independent of the master clocks' duty cycle (now 50%)
create_generated_clock -source [get_ports clk] -edges {1 1 5} -edge_shift { 0 2.5 0 } \
[get_registers clkdivB|clkreg]
```
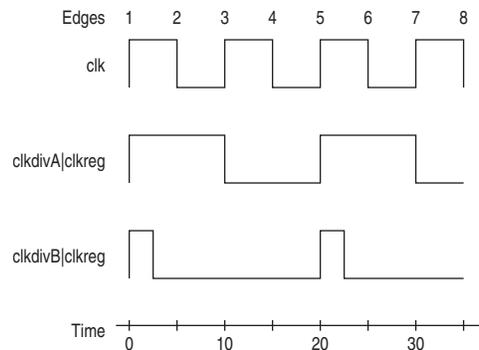
Figure 7–25 shows the effect of the -multiply_by option on the generated clock.

Figure 7–25. Multiplying a Generated Clock

```
create_clock -period 10 -waveform { 0 5 } [get_ports clk]
# Creates a multiply-by-2 clock
create_generated_clock -source [get_ports clk] -multiply_by 2 [get_registers \
clkmult|clkreg]
```
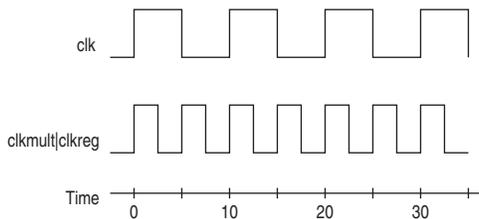
Virtual Clocks

A virtual clock is a clock that does not have a real source in the design or that does not interact directly with the design. For example, if a clock feeds only an external device's clock port and not a clock port in the design, and the external device then feeds (or is fed by) a port in the design, it is considered a virtual clock.

Use the create_clock command to create virtual clocks, with no value specified for the source option.

☞ You can use virtual clocks for set_input_delay and set_output_delay constraints.

Figure 7–26 shows an example where a virtual clock is required for the Quartus II TimeQuest Timing Analyzer to properly analyze the relationship between the external register and those in the design. Because the oscillator labeled virt_clk does not interact with the Altera device, but acts as the clock source for the external register, the clock virt_clk must be declared. Example 7–6 shows the command to create a 10 ns virtual clock named virt_clk with a 50% duty cycle where the first rising edge occurs at 0 ns. The virtual clock is then used as the clock source for an output delay constraint.

Figure 7–26. Virtual Clock Board Topology
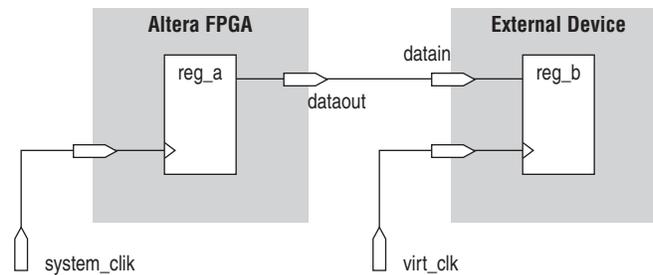
After you create the virtual clock, you can perform register-to-register analysis between the register in the Altera device and the register in the external device.

Example 7–6. Virtual Clock Example 1

```
#create base clock for the design
create_clock -period 5 [get_ports system_clk]
#create the virtual clock for the external register
create_clock -period 10 -name virt_clk -waveform { 0 5 }
#set the output delay referencing the virtual clock
set_output_delay -clock virt_clk -max 1.5 [get_ports dataout]
```

Example 7–7 shows the command to create a 10 ns virtual clock with a 50% duty cycle that is phase shifted by 90°.

Example 7–7. Virtual Clock Example 2

```
create_clock -name virt_clk -period 10 -waveform { 2.5 7.5 }
```

Multi-Frequency Clocks

Certain designs have more than one clock source feeding a single clock port in the design. The additional clock may act as a low-power clock, with a lower frequency than the primary clock. To analyze this type of design, the create_clock command supports the -add option that allows you to add more than one clock to a clock node.

Example 7–8 shows the command to create a 10 ns clock applied to clock port clk, and then add an additional 15 ns clock to the same clock port. The Quartus II TimeQuest Timing Analyzer uses both clocks when it performs timing analysis.

Example 7–8. Multi-Frequency Example

```
create_clock –period 10 –name clock_primary –waveform { 0 5 } [get_ports
clk]
create_clock –period 15 –name clock_secondary –waveform { 0 7.5 }
[get_ports clk] -add
```

Automatic Clock Detection

To create clocks for all clock nodes in your design automatically, use the `derive_clocks` command. This command creates clocks on ports or registers to ensure every register in the design has a clock.

Example 7–9 shows the `derive_clocks` command and options.

Example 7–9. derive_clocks Command

```
derive_clocks
[-period <period value>]
[-waveform <edge list>]
```

Table 7–8 describes the options for the `derive_clocks` command.

Table 7–8. derive_clocks Command Options

Option	Description
–period <period value>	Creates the clock period. You can also specify the frequency as –period <num>MHz. (1)
–waveform <edge list>	Creates the clock's rising and falling edges. The edge list alternates between the rising edge and falling edge. For example, for a 10 ns period where the first rising edge occurs at 0 ns and the first falling edge occurs at 5 ns, the edge list is waveform { 0 5 }. The difference must be within one period unit, and the rising edge must come before the falling edge. The default edge list is { 0 period/2 }, or a 50% duty cycle.

Note to Table 7–8:

(1) This option uses the default time unit nanoseconds (ns).

☞ The `derive_clocks` command does not create clocks for the output of the PLLs.

The `derive_clocks` command is equivalent to using `create_clock` for each register or port feeding the clock pin of a register.

☞ Using the `derive_clocks` command for final timing sign-off is not recommended. You should create clocks for all clock sources using the `create_clock` and `create_generated_clock` commands.

Derive PLL Clocks

PLLs are used for clock management and synthesis in Altera devices. You can customize the clocks generated from the outputs of the PLL based on design requirements. Because a clock should be created for all clock nodes, all outputs of the PLL should have an associated clock.

You can manually create a clock for each output of the PLL with the
`create_generated_clock` command, or you can use the `derive_pll_clocks`
command, which automatically searches the timing netlist and creates generated
clocks for all PLL outputs according to the settings specified for each PLL output.

Use the `derive_pll_clocks` command to automatically create a clock for each
output of the PLL. Example 7–10 shows the `derive_pll_clocks` command and
options.

Example 7–10. derive_pll_clocks Command

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

Table 7–9 describes the options for the `derive_pll_clocks` command.

Table 7–9. derive_pll_clocks Command Options

Option	Description
-use_tan_name	By default, the clock name is the output clock name. This option uses the net name similar to the names used by the Quartus II Classic Timing Analyzer.
-create_base_clocks	Creates the base clocks on input clock ports of the design that are feeding the PLL.

The `derive_pll_clocks` command calls the `create_generated_clock`
command to create generated clocks on the outputs of the PLL. The source for the
`create_generated_clock` command is the input clock pin of the PLL. Before or
after the `derive_pll_clocks` command has been issued, you must manually create
a base clock for the input clock port of the PLL. If a clock is not defined for the input
clock node of the PLL, no clocks are reported for the PLL outputs. Instead, the
Quartus II TimeQuest Timing Analyzer issues a warning message similar to
Example 7–11 when the timing netlist is updated.

Example 7–11. Warning Message

```
Warning: The master clock for this clock assignment could not be
derived.
Clock: <name of PLL output clock pin name> was not created.
```

☞   You can use the `-create_base_clocks` option to create the input clocks for the
PLL inputs automatically.

You can include the `derive_pll_clocks` command in your **.sdc** file, which allows
the `derive_pll_clocks` command to automatically detect any changes to the PLL.
With the `derive_pll_clocks` command in your **.sdc** file, each time the file is read,
the appropriate `create_generated_clocks` command for the PLL output clock
pin is generated. If you use the `write_sdc-expand` command after the
`derive_pll_clock` command, the new **.sdc** file contains the individual
`create_generated_clock` commands for the PLL output clock pins and not the
`derive_pll_clocks` command. Any changes to the properties of the PLL are not
automatically reflected in the new **.sdc** file. You must manually update the
`create_generated_clock` commands in the new **.sdc** file written by the
`derive_pll_clocks` command to reflect the changes to the PLL.

☞ The `derive_pll_clocks` constraint will also constrain any LVDS transmitters or LVDS receivers in the design by adding the appropriate multicycle constraints to account for any deserialization factors.

For example, Figure 7–27 shows a simple PLL design with a register-to-register path.

Figure 7–27. Simple PLL Design
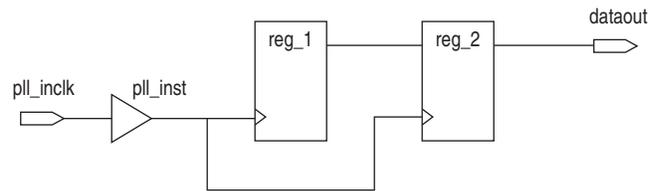
Use the `derive_pll_clocks` command to automatically constrain the PLL. When this command is issued for the design shown in Figure 7–27, the messages shown in Example 7–12 are generated.

Example 7–12. derive_pll_clocks Generated Messages

```
Info:
Info: Deriving PLL Clocks:
Info: create_generated_clock -source
pll_inst|altpll_component|pll|inclk[0] -divide_by 2 -name
pll_inst|altpll_component|pll|clk[0]
pll_inst|altpll_component|pll|clk[0]
Info:
```

The node name `pll_inst|altpll_component|pll|inclk[0]` used for the source option refers to the input clock pin of the PLL. In addition, the name of the output clock of the PLL is the name of the PLL output clock node, `pll_inst|altpll_component|pll|clk[0]`.

☞ If the PLL is in clock switchover mode, multiple clocks are created for the output clock of the PLL; one for the primary input clock (for example, `inclk[0]`), and one for the secondary input clock (for example, `inclk[1]`). In this case, you should cut the primary and secondary output clocks using the `set_clock_groups` command with the `-exclusive` option.

Before you can generate any reports for this design, you must create a base clock for the PLL input clock port. Use a the following command or one similar:

```
create_clock -period 5 [get_ports pll_inclk]
```

☞ You do not have to generate the base clock on the input clock pin of the PLL: `pll_inst|altpll_component|pll|inclk[0]`. The clock created on the PLL input clock port propagates to all fan-outs of the clock port, including the PLL input clock pin.

Default Clock Constraints

To provide a complete clock analysis, the Quartus II TimeQuest Timing Analyzer, by default, automatically creates clocks for all detected clock nodes in your design that have not be constrained, if there are no base clock constraints in the design. The Quartus II TimeQuest Timing Analyzer creates a base clock with a 1 GHz requirement to unconstrained clock nodes, using the following command:

```
derive_clocks -period 1 ↵
```

☞ Individual clock constraints (for example, `create_clock` and `create_generated_clock`) should be made for all clocks in the design. This results in a thorough and realistic analysis of the design's timing requirements. Avoid using `derive_clocks` for final timing sign-off.

The default clock constraint is only applied when the Quartus II TimeQuest Timing Analyzer detects that all synchronous elements have no clocks associated with them. For example, if a design contains two clocks and only one clock has constraints, the default clock constraint is not applied. However, if both clocks have not been constrained, the default clock constraint is applied.

Clock Groups

Many clocks can exist in a design; however, not all of the clocks interact with one another and certain clock interactions are not possible.

Use the `set_clock_groups` command to specify clocks that are exclusive or asynchronous. Example 7–13 shows the `set_clock_groups` command and options.

Example 7–13. set_clock_groups Command

```
set_clock_groups
[-asynchronous | -exclusive]
-group <clock name>
[-group <clock name>]
[-group <clock name>] ...
```

Table 7–10 describes the options for the `set_clock_groups` command.

Table 7–10. set_clock_groups Command Options

Option	Description
`-asynchronous`	Asynchronous clocks—when the two clocks have no phase relationship and are active at the same time.
`-exclusive`	Exclusive clocks—when only one of the two clocks is active at any given time. An example of an exclusive clock group is when two clocks feed a 2-to-1 MUX.
`-group <clock name>`	Specifies valid destination clock names that are mutually exclusive. <clock name> is used to specify the clock names.

The exclusive option is used to declare when two clocks are mutually exclusive to each other and cannot coexist in the design at the same time. This can happen when multiple clocks are created on the same node or for multiplexed clocks. For example, a port can be clocked by either a 25-MHz or a 50-MHz clock. To constrain this port, two clocks should be created on the port with the create_clock command, then use set_clock_groups -exclusive to declare that they cannot coexist in the design at the same time.

This eliminates any clock transfers that may be derived between the 25-MHz clock and the 50-MHz clock. Example 7–14 shows the constraints for this.

Example 7–14. Exclusive Option

```
create_clock -period 40 -name clk_A [get_ports {port_A}]
create_clock -add -period 20 -name clk_B [get_ports {port_A}]
set_clock_groups -exclusive -group {clk_A} -group {clk_B}
```

A group is defined with the -group option. The TimeQuest Timing Analyzer cuts the timing paths between clocks each of the separate -groups groups.

The asynchronous option is used to group related and unrelated clocks. With the asynchronous option, clocks that are contained in groups are considered asynchronous to each other. Any clocks within each group are considered synchronous to each other.

For example, suppose you have three clocks: clk_A, clk_B, and clk_C. The clocks clk_A and clk_B are related to each other, but clock clk_C operates completely asynchronous with clk_A or clk_B. Example 7–15 makes clk_A and clk_B related in the same group and unrelated with the second group which contains clk_C.

Example 7–15. Asynchronous Option Example 1

```
set_clock_groups -asynchronous -group {clk_A clk_B} -group {clk_C}
```

Example 7–16 shows an alternative method of specifying the same constraint as Example 7–15.

Example 7–16. Asynchronous Option Example 2

```
set_clock_groups -asynchronous -group {clk_C}
```

This makes clk_C unrelated with every other clock in the design because clk_C is the only group in the constraint.

☞ The TimeQuest Timing Analyzer assumes all clocks are related by default, unless constrained otherwise.

Example 7–17 shows a set_clock_groups command and the equivalent set_false_path commands.

Example 7–17. set_clock_groups

```
# Clocks A and C are never active when clocks B and D are active
set_clock_groups -exclusive -group {A C} -group {B D}

# Equivalent specification using false paths
set_false_path -from [get_clocks A] -to [get_clocks B]
set_false_path -from [get_clocks A] -to [get_clocks D]
set_false_path -from [get_clocks C] -to [get_clocks B]
set_false_path -from [get_clocks C] -to [get_clocks D]
set_false_path -from [get_clocks B] -to [get_clocks A]
set_false_path -from [get_clocks B] -to [get_clocks C]
set_false_path -from [get_clocks D] -to [get_clocks A]
set_false_path -from [get_clocks D] -to [get_clocks C]
```

Clock Effect Characteristics

The `create_clock` and `create_generated_clock` commands create ideal clocks that do not account for any board effects. This section describes how to account for clock effect characteristics with clock latency and clock uncertainty.

Clock Latency

There are two forms of clock latency: source and network. Source latency is the propagation delay from the origin of the clock to the clock definition point (for example, a clock port). Network latency is the propagation delay from a clock definition point to a register's clock pin. The total latency (or clock propagation delay) at a register's clock pin is the sum of the source and network latencies in the clock path.

☞ The `set_clock_latency` command supports only source latency. The `-source` option must be specified when using the command.

Use the `set_clock_latency` command to specify source latency to any clock ports in the design. Example 7–18 shows the `set_clock_latency` command and options.

Example 7–18. set_clock_latency Command

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

Table 7–11 describes the options for the `set_clock_latency` command.

Table 7–11. set_clock_latency Command Options (Part 1 of 2)

Option	Description
`-source`	Specifies a source latency.
`-clock <clock list>`	Specifies the clock to use if the target has more than one clock assigned to it.
`-rise	-fall`	Specifies the rising or falling delays.
`-late	-early`	Specifies the earliest or the latest arrival times to the clock.

Table 7–11. set_clock_latency Command Options (Part 2 of 2)

Option	Description
`<delay>`	Specifies the delay value.
`<targets>`	Specifies the clocks or clock sources if a clock is clocked by more than one clock.

The Quartus II TimeQuest Timing Analyzer automatically computes network latencies; therefore, the `set_clock_latency` command specifies only source latencies.

Clock Uncertainty

The `set_clock_uncertainty` command specifies clock uncertainty or skew for clocks or clock-to-clock transfers. Specify the uncertainty separately for setup and hold. Specify separate rising and falling clock transitions. The Quartus II TimeQuest Timing Analyzer subtracts setup uncertainty from the data required time for each applicable path and adds the hold uncertainty to the data required time for each applicable path.

Use the `set_clock_uncertainty` command to specify any clock uncertainty to the clock port. Example 7–19 shows the `set_clock_uncertainty` command and options.

Example 7–19. set_clock_uncertainty Command and Options

```
set_clock_uncertainty
[-rise_from <rise from clock> | -fall_from <fall from clock> |
    -from <from clock>]
[-rise_to <rise to clock> | -fall_to <fall to clock> | -to <to clock>]
[-setup | -hold]
<value>
-add
```

Table 7–12 describes the options for the `set_clock_uncertainty` command.

Table 7–12. set_clock_uncertainty Command Options

Option	Description
`-from <from clock>`	Specifies the `from` clock.
`-rise_from <rise from clock>`	Specifies the `rise-from` clock.
`-fall_from <fall from clock>`	Specifies the `fall-from` clock.
`-to <to clock>`	Specifies the `to` clock.
`-rise_to <rise to clock>`	Specifies the `rise-to` clock.
`-fall_to <fall to clock>`	Specifies the `fall-to` clock.
`-setup	-hold`	Specifies setup or hold.
`<value>`	Uncertainty value.
`-add`	Specifies that the uncertainty *<value>* should be added to the uncertainty value derived by the `derive_clock_uncertainty` command.

Derive Clock Uncertainty

Use the `derive_clock_uncertainty` command to automatically apply inter-clock, intra-clock, and I/O interface uncertainties. Both setup and hold uncertainties are calculated for each clock-to-clock transfer. Example 7–20 shows the `derive_clock_uncertainty` command and options.

Example 7–20. derive_clock_uncertainty Command

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

Table 7–13 describes the options for the `derive_clock_uncertainty` command.

Table 7–13. derive_clock_uncertainty Command Options

Option	Description
-overwrite	Overwrites previously performed clock uncertainty assignments.
-add	Adds derived uncertainty results to any user-defined clock uncertainty assignments.

The Quartus II TimeQuest Timing Analyzer automatically applies clock uncertainties to clock-to-clock transfers in the design.

Any clock uncertainty constraints that have been applied to source and destination clock pairs with the `set_clock_uncertainty` command have a higher precedence than the clock uncertainties derived from the `derive_clock_uncertainty` command for the same source and destination clock pairs. For example, if `set_clock_uncertainty` is applied between `clka` and `clkb`, the `derive_clock_uncertainty` values for the clock transfer is ignored by default. The `set_clock_uncertainty` constraint has priority over the `derive_clock_uncertainty` constraint.

The clock uncertainty value that would have been used; however, is still reported for informational purposes. You can use the `-overwrite` command to overwrite previous clock uncertainty assignments, or remove them manually with the `remove_clock_uncertainty` command. You can also use the `-add` option to add clock uncertainty determined by the `derive_clock_uncertainty` command to any previously defined clock uncertainty value.

The following list shows the types of clock-to-clock transfers in which clock certainties can arise. They are modeled by the `derive_clock_uncertainty` command automatically.

■ Inter-clock

■ Intra-clock

■ I/O Interface

☞ Altera recommends using the `derive_clock_uncertainty` command.

Intra-Clock Transfers

Intra-clock transfers occur when the register-to-register transfer happens in the core of the FPGA and source and destination clocks come from the same PLL output pin or clock port. An example of an intra-clock transfer is shown in Figure 7–28.

Figure 7–28. Intra-Clock Transfer
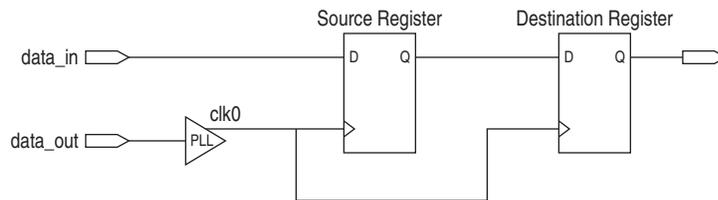
Inter-Clock Transfers

Inter-clock transfers occur when a register-to-register transfer happens in the core of the FPGA and source and destination clocks come from a different PLL output pin or clock port. An example of an inter-clock transfer is shown in Figure 7–29.

Figure 7–29. Inter-Clock Transfer
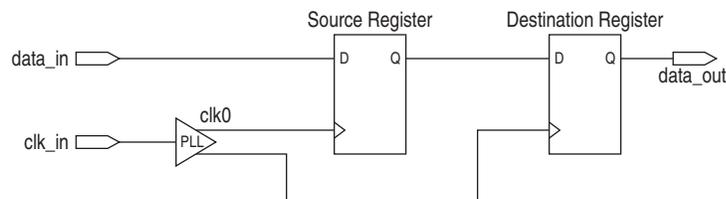
I/O Interface Clock Transfers

I/O interface clock transfers occur when data transfers from an I/O port to the core of the FPGA (input) or from the core of the FPGA to the I/O port (output). An example of an I/O interface clock transfer is shown in Figure 7–30.

Figure 7–30. I/O Interface-Clock Transfer
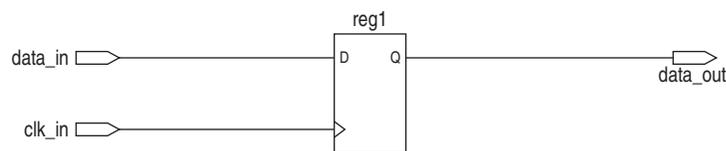
For I/O interface uncertainty, you must create a virtual clock and constrain the input and output ports with the set_input_delay and set_output_delay commands that reference the virtual clock. The virtual clock is required to prevent the derive_clock_uncertainty command from applying clock uncertainties for either intra- or inter-clock transfers on an I/O interface clock transfer when the set_input_delay or set_output_delay commands reference a clock port or PLL output. If a virtual clock is not referenced in the set_input_delay or set_output_delay commands, the derive_clock_uncertainty command calculates intra- or inter-clock uncertainty value for the I/O interface.

Create the virtual clock with the same properties as the original clock that is driving the I/O port. For example, Figure 7–31 shows a typical input I/O interface with the clock specifications.

Figure 7–31. I/O Interface Specifications
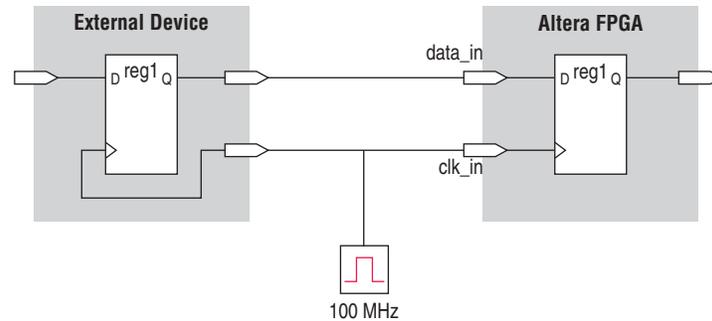
Example 7–21 shows the SDC commands to constrain the I/O interface shown in Figure 7–31.

Example 7–21. SDC Commands to Constrain the I/O Interface

```
# Create the base clock for the clock port
create_clock –period 10 –name clk_in [get_ports clk_in]
# Create a virtual clock with the same properties of the base clock
# driving the source register
create_clock –period 10 –name virt_clk_in
# Create the input delay referencing the virtual clock and not the base
# clock
# DO NOT use set_input_delay –clock clk_in <delay_value>
# [get_ports data_in]
set_input_delay –clock virt_clk_in <delay value> [get_ports data_in]
```

I/O Specifications

The Quartus II TimeQuest Timing Analyzer supports Synopsys Design Constraints that constrain the ports in your design. These constraints allow the Quartus II TimeQuest Timing Analyzer to perform a system static timing analysis that includes not only the FPGA internal timing, but also any external device timing and board timing parameters.

Input and Output Delay

Use input and output delay constraints to specify any external device or board timing parameters. When you apply these constraints, the Quartus II TimeQuest Timing Analyzer performs static timing analysis on the entire system.

Set Input Delay

The `set_input_delay` constraint specifies the data arrival time at a port (a device I/O) with respect to a given clock. Figure 7–32 shows an input delay path.

Figure 7–32. Set Input Delay
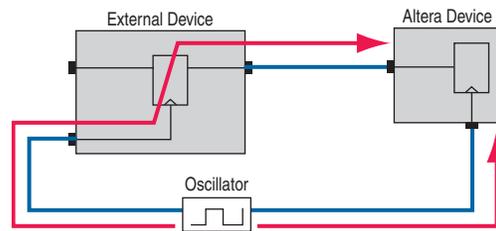
Use the `set_input_delay` command to specify input delay constraints to ports in the design. Example 7–22 shows the `set_input_delay` command and options.

Example 7–22. set_input_delay Command

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

Table 7–14 describes the options for the `set_input_delay` command.

Table 7–14. set_input_delay Command Options

Option	Description
`-clock <clock name>`	Specifies the source clock.
`-clock_fall`	Specifies the arrival time with respect to the falling edge of the clock.
`-rise	-fall`	Specifies either the rise or fall delay at the port.
`-max	-min`	Specifies the minimum or maximum data arrival time.
`-add_delay`	Adds another delay, but does not replace the existing delays assigned to the port.
`-reference_pin <target>`	Specifies a pin or port in the design from which to determine source and network latencies. This is useful to specify input delays relative to an output port fed by a clock.
`-source_latency_ included`	Specifies that the input delay value includes the source latency delay value; therefore, any source clock latency assigned to the clock is ignored.
`<delay value>`	Specifies the delay value.
`<targets>`	Specifies the destination ports or pins.

☞ A warning message appears if you specify only a `-max` or `-min` value for the input delay value. The input minimum delay default value is equal to the input maximum delay; the input maximum delay default value is equal to the input minimum delay, if only one is specified. Similarly, a warning message appears if you specify only a `-rise` or `-fall` value for the delay value, and the delay defaults in the same manner as the input minimum and input maximum delays.

The maximum value is used for setup checks; the minimum value is used for hold checks.

By default, a set of input delays (min/max, rise/fall) is allowed for only one -clock, -clock_fall, -reference_pin combination. Specifying an input delay on the same port for a different clock, -clock_fall or -reference_pin removes any previously set input delays, unless you specify the -add_delay option. When you specify the -add_delay option, the worst-case values are used.

The -rise and -fall options are mutually exclusive, as are the -min and -max options.

Set Output Delay

The set_output_delay command specifies the data required time at a port (the device pin) with respect to a given clock.

Use the set_output_delay command to specify output delay constraints to ports in the design. Figure 7–33 shows an output delay path.

Figure 7–33. Output Delay
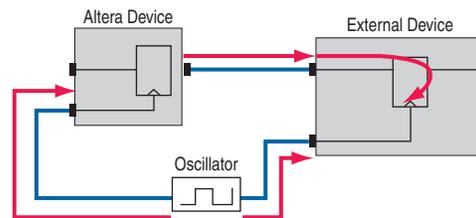
Example 7–23 shows the set_output_delay command and options.

Example 7–23. set_output_delay Command

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

Table 7–15 describes the options for the set_output_delay command.

Table 7–15. set_output_delay Command Options (Part 1 of 2)

Option	Description
-clock <clock name>	Specifies the source clock.
-clock_fall	Specifies the required time with respect to the falling edge of the clock.
-rise	-fall	Specifies either the rise or fall delay at the port.
-max	-min	Specifies the minimum or maximum data arrival time.
-add_delay	Adds another delay, but does not replace the existing delays assigned to the port.

Table 7–15. set_output_delay Command Options (Part 2 of 2)

Option	Description
-reference_pin <target>	Specifies a pin or port in the design from which to determine source and network latencies. Use this option to specify input delays relative to an output port fed by a clock.
-source_latency_included	Specifies that the input delay value includes the source latency delay value; therefore, any source clock latency assigned to the clock will subsequently be ignored.
<delay value>	Specifies the delay value.
<targets>	Specifies the destination ports or pins.

☞ A warning message appears if you specify only a -max or -min value for the output delay value. The output minimum delay default value is the output maximum delay; the output maximum delay default value is the output minimum delay, if only one is specified.

The maximum value is used for setup checks; the minimum value is used for hold checks.

By default, a set of output delays (min/max, rise/fall) is allowed for only one clock, -clock_fall, port combination. Specifying an output delay on the same port for a different clock or -clock_fall removes any previously set output delays, unless you specify the -add_delay option. When you specify the -add_delay option, the worst-case values are used.

The -rise and -fall options are mutually exclusive, as are the -min and -max options.

Skew Specifications

The TimeQuest Timing Analyzer supports the ability to specify and report maximum, minimum, and skew delays between a source and destination points.

set_net_delay

Use the set_net_delay command in conjunction with the report_net_delay command to report the net delays and perform minimum, maximum, or skew timing analysis across nets. Example 7–24 shows the set_net_delay command and options.

The set_net_delay and report_net_delay commands can be used when verifying timing-critical delays and skews for high-speed interfaces. For example, the command can be used to report the skew across a high-speed data bus for each bit.

☞ Both of these commands are used for reporting purposes only. The set_net_delay command is not used to optimize the path or skew for the specified nets.

Example 7–24. set_net_delay Command

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

Table 7–16 describes the options for the set_net_delay command.

Table 7–16. set_net_delay Command Options

Option	Description
-from <names>	Valid source pins or ports (string patterns are matched using Tcl string matching).
-max	Specifies maximum delay.
-min	Specifies minimum delay.
-skew	Specifies skew across edges.
-to <names>	Valid destination pins or ports (string patterns are matched using Tcl string matching). If -to is left unspecified, the missing value or values are substituted by an "*" character.
<delay>	Required delay.

When the -min option is specified, the slack is calculated with the minimum edge delay. When the -max option is specified, the slack is calculated with the maximum edge delay. When the -skew option is specified, the slack is calculated across all the valid edges that satisfy the -from and -to filters.

Timing Exceptions

Timing exceptions modify the default analysis performed by the Quartus II TimeQuest Timing Analyzer. This section describes the following available timing exceptions:

■ "False Path" on page 7–45

■ "Minimum Delay" on page 7–46

■ "Maximum Delay" on page 7–47

■ "Multicycle Path" on page 7–48

Precedence

If a conflict of node names occurs between timing exceptions, the following order of precedence applies:

1. False path

2. Minimum delays and maximum delays

3. Multicycle path

The false path timing exception has the highest precedence. Within each category, assignments to individual nodes have precedence over assignments to clocks. Finally, the remaining precedence for additional conflicts is order-dependent, such that the last assignments overwrite (or partially overwrite) earlier assignments.

False Path

False paths are paths that can be ignored during timing analysis.

Use the `set_false_path` command to specify false paths in the design. Example 7–25 shows the `set_false_path` command and options.

Example 7–25. set_false_path Command

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

Table 7–17 describes the options for the `set_false_path` command.

Table 7–17. set_false_path Command Options

Option	Description
`-fall_from <clocks>`	The *<names>* is a collection or list of objects in the design. Specifies false path begins at the fall from *<clocks>*.
`-fall_to <clocks>`	The <names> is a collection or list of objects in the design. Specifies false path ends at the fall to *<clocks>*.
`-from <names>`	The *<names>* is a collection or list of objects in the design. Specifies false path begins at the *<names>*.
`-hold`	Specifies the false path is valid during the hold analysis only.
`-rise_from <clocks>`	The *<names>* is a collection or list of objects in the design. Specifies false path begins at the rise from *<clocks>*.
`-rise_to <clocks>`	The *<names>* is a collection or list of objects in the design. Specifies false path ends at the rise to *<clocks>*.
`-setup`	Specifies the false path is valid during the setup analysis only.
`-through <names>`	The *<names>* is a collection or list of objects in the design. Specifies false path passes through *<names>*.
`-to <names>`	The *<names>* is a collection or list of objects in the design. Specifies false path ends at *<names>*.
`<delay>`	Specifies the delay value.

When the objects are timing nodes, the false path only applies to the path between the two nodes. When an object is a clock, the false path applies to all paths where the source node (`-from`) or destination node (`-to`) is clocked by the clock.

Minimum Delay

Use the `set_min_delay` command to specify an absolute minimum delay for a given path. Example 7–26 shows the `set_min_delay` command and options.

Example 7–26. set_min_delay Command

```
set_min_delay
[-fall_from <clocks> | -rise_from <clocks> | -from <names>]
[-fall_to <clocks> | -rise_to <clocks> | -to <names>]
[-through <names>]
<delay>
```

Table 7–18 describes the options for the `set_min_delay` command.

Table 7–18. set_min_delay Command Options

Option	Description
-fall_from <clocks>	The <names> is a collection or list of objects in the design. Specifies the minimum delay begins at the falling edge of <clocks>.
-fall_to <clocks>	The <names> is a collection or list of objects in the design. Specifies the minimum delay ends at the falling of <clocks>.
-from <names>	The <names> is a collection or list of objects in the design. The <names> acts as the start point of the path.
-rise_from <clocks>	The <names> is a collection or list of objects in the design. Specifies the minimum delay at the rising edge of <clocks>.
rise_to <clocks>	The <names> is a collection or list of objects in the design. Specifies the minimum delay at the rising edge of <clocks>.
-through <names>	The <names> is a collection or list of objects in the design. The <names> acts as the through point of the path.
-to <names>	The <names> is a collection or list of objects in the design. The <names> acts as the end point of the path.
<delay>	Specifies the delay value.

If the source or destination node is clocked, the clock paths are taken into account, allowing more or less delay on the data path. If the source or destination node has an input or output delay, that delay is also included in the minimum delay check.

When the objects are timing nodes, the minimum delay applies only to the path between the two nodes. When an object is a clock, the minimum delay applies to all paths where the source node (`-from`) or destination node (`-to`) is clocked by the clock.

You can apply the `set_min_delay` command exception to an output port that does not use a `set_output_delay` constraint. In this case, the setup summary and hold summary report the slack for these paths. Because there is no clock associated with the output port, no clock is reported for these paths and the Clock column is empty. In this case, you cannot report timing for these paths.

☞ To report timing using clock filters for output paths with the `set_min_delay` command, you can use the `set_output_delay` command for the output port with a value of 0. You can use an existing clock from the design or a virtual clock as the clock reference in the `set_output_delay` command.

Maximum Delay

Use the `set_max_delay` command to specify an absolute maximum delay for a given path. Example 7–27 shows the `set_max_delay` command and options.

Example 7–27. set_max_delay Command

```
set_max_delay
[-fall_from <clocks> | -rise_from <clocks> | -from <names>]
[-fall_to <clocks> | -rise_to <clocks> | -to <names>]
[-through <names>]
<delay>
```

Table 7–19 describes the options for the `set_max_delay` command.

Table 7–19. or set_max_delay Command Options

Option	Description
`-fall_from <clocks>`	The <*names*> is a collection or list of objects in the design. Specifies the maximum delay begins at the falling edge of <*clocks*>.
`-fall_to <clocks>`	The <*names*> is a collection or list of objects in the design. Specifies the maximum delay ends at the falling of <*clocks*>.
`-from <names>`	The <*names*> is a collection or list of objects in the design. The <*names*> acts as the start point of the path.
`-rise_from <clocks>`	The <*names*> is a collection or list of objects in the design. Specifies the maximum delay at the rising edge of <*clocks*>.
`rise_to <clocks>`	The <*names*> is a collection or list of objects in the design. Specifies the maximum delay at the rising edge of <*clocks*>.
`-through <names>`	The <*names*> is a collection or list of objects in the design. The <*names*> acts as the through point of the path.
`-to <names>`	The <*names*> is a collection or list of objects in the design. The <*names*> acts as the end point of the path.
`<delay>`	Specifies the delay value.

If the source or destination node is clocked, the clock paths are taken into account, allowing more or less delay on the data path. If the source or destination node has an input or output delay, that delay is also included in the maximum delay check.

When the objects are timing nodes, the maximum delay only applies to the path between the two nodes. When an object is a clock, the maximum delay applies to all paths where the source node (`-from`) or destination node (`-to`) is clocked by the clock.

You can apply the `set_max_delay` command exception to an output port that does not use a `set_output_delay` constraint. In this case, the setup summary and hold summary report the slack for these paths. Because there is no clock associated with the output port, no clock is reported for these paths and the Clock column is empty. In this case, you cannot report timing for these paths.

☞ To report timing using clock filters for output paths with the `set_max_delay` command, you can use the `set_output_delay` command for the output port with a value of 0. You can use an existing clock from the design or a virtual clock as the clock reference in the `set_output_delay` command.

Multicycle Path

By default, the Quartus II TimeQuest Timing Analyzer uses a single-cycle analysis. When analyzing a path, the setup launch and latch edge times are determined by finding the closest two active edges in the respective waveforms. For a hold analysis, the timing analyzer analyzes the path against two timing conditions for every possible setup relationship, not just the worst-case setup relationship. Therefore, the hold launch and latch times may be completely unrelated to the setup launch and latch edges.

A multicycle constraint relaxes setup or hold relationships by the specified number of clock cycles based on the source (-start) or destination (-end) clock. An end multicycle constraint of 2 extends the worst-case setup latch edge by one destination clock period.

Hold multicycle constraints are based on the default hold position (the default value is 0). An end hold multicycle constraint of 1 effectively subtracts one destination clock period from the default hold latch edge.

Use the set_multicycle_path command to specify the multicycle constraints in the design. Example 7–28 shows the set_multicycle_path command and options.

Example 7–28. set_multicycle_path Command

```
set_multicycle_path
[-end]
[-fall_from <clocks> | -rise_from <clocks> | -from <names>]
[-fall_to <clocks> | -rise_to <clocks> | -to <names>]
[-hold]
[-setup]
[-start]
[-through <names>]
<path multiplier>
```

Table 7–20 describes the options for the set_multicycle_path command.

Table 7–20. set_multicycle_path Command Options (Part 1 of 2)

Option	Description
fall_from <clocks>	The <names> is a collection or list of objects in the design. Specifies the multicycle begins at the falling edge of <clocks>.
fall_to <clocks>	The <names> is a collection or list of objects in the design. Specifies the multicycle ends at the falling of <clocks>.
-from <names>	The <names> is a collection or list of objects in the design. The <names> acts as the start point of the path.
-hold | -setup	Specifies the type of multicycle to be applied.
-rise_from <clocks>	The <names> is a collection or list of objects in the design. Specifies the multicycle at the rising edge of <clocks>.
-rise_to <clocks>	The <names> is a collection or list of objects in the design. Specifies the multicycle ends at the rising edge of <clocks>.
-start | -end	Specifies whether the start or end clock acts as the source or destination for the multicycle.
-through <names>	The <names> is a collection or list of objects in the design. Specifies multicycle passes through <names>.

Table 7–20. set_multicycle_path Command Options (Part 2 of 2)

Option	Description
`-to <names>`	The <*names*> is a collection or list of objects in the design. The <*names*> acts as the end point of the path.
`<path multiplier>`	Specifies the multicycle multiplier value.

When the objects are timing nodes, the multicycle constraint only applies to the path between the two nodes. When an object is a clock, the multicycle constraint applies to all paths where the source node (`-from`) or destination node (`-to`) is clocked by the clock.

Annotated Delay

Use the `set_annotated_delay` command to annotate 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. The annotated delay can be specified for specific transition edges: rise-rise, fall-rise, rise-fall, and fall-fall, and can also set different minimum and maximum values.

☞ If no transition is specified, the given delay is assigned to all four values. Options `-max` and `-min` allow users to specify maximum or minimum delay.

Example 7–29 shows the `set_annotated` command and options.

Example 7–29. set_annotated_delay Command

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

Table 7–21 describes the options for the `set_annotated_delay` command.

Table 7–21. set_annotated_delay Command Options (Part 1 of 2)

Options	Description
`-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). If `-from` value is left unspecified, the "*" character is used.
`-max`	Specifies that only the maximum delay should be set.
`-min`	Specifies that only the minimum delay should be set.
`-net`	Specifies that net delay must be set.
`-operating_conditions <operating_conditions>`	Specifies the operating conditions Tcl object. Refer to Table 7–49 on page 7–74 for the operating conditions Tcl object.
`-rf`	Specifies that RF delay must be set.

Table 7–21. set_annotated_delay Command Options (Part 2 of 2)

Options	Description
-rr	Specifies that RR delay must be set.
-to <names>	Valid destination pins or ports (string patterns are matched using Tcl string matching). If –to value is left unspecified, the "*" character is used.
<delay>	The delay value in default time units.

With the -operation_conditions option, different operating conditions can be specified in a single **.sdc** file, removing the requirement of having multiple **.sdc** files that specify different operating conditions.

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

Application Examples

This section describes specific examples for the set_multicycle_path command.

Figure 7–34 shows a register-to-register path where the source clock, src_clk, has a period of 10 ns and the destination clock, dst_clk, has a period of 5 ns.

Figure 7–34. Register-to-Register Path
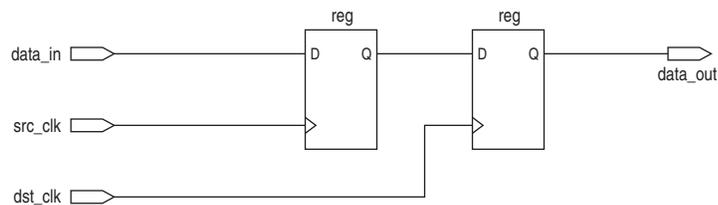
Figure 7–35 shows the respective timing diagrams for the source and destination clocks and the default setup and hold relationships. The default setup relationship is 5 ns; the default hold relationship is 0 ns.

Figure 7–35. Default Setup and Hold Timing Diagram
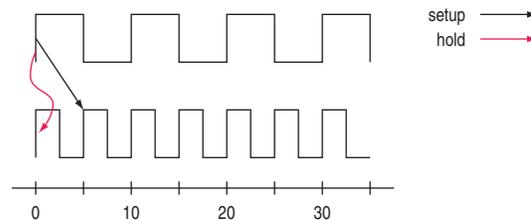
The default setup and hold relationships can be modified with the set_multicycle_path command to accommodate system requirements.

Table 7–22 shows the commands used to modify either the launch or latch edge times that the TimeQuest Timing Analyzer uses to determine a setup relationship or hold relationship.

Table 7–22. Commands to Modify Edge Times

Command	Description of Modification
set_multicycle_path -setup -end	Latch edge time of the setup relationship
set_multicycle_path -setup -start	Launch edge time of the setup relationship
set_multicycle_path -hold -end	Latch edge time of the hold relationship
set_multicycle_path -hold -start	Launch edge time of the hold relationship

Figure 7–36 shows the command used to modify the setup latch edge and the resulting timing diagram. The command moves the latch edge time to 10 ns from the default 5 ns.

Figure 7–36. Modified Setup Diagram

```
# latch every 2nd edge
set_multicycle_path -from [get_clocks src_clk] -to [get_clocks dst_clk] -setup -end 2
```
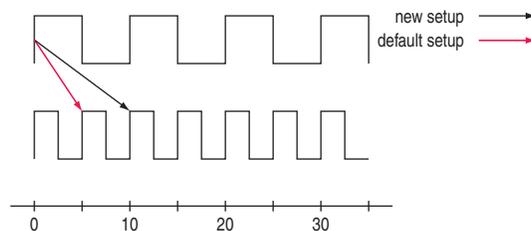

Constraint and Exception Removal

When using the Quartus II TimeQuest Timing Analyzer interactively, it is usually necessary to remove a constraint or exception. In cases where constraints and exceptions either become outdated or have been erroneously entered, the Quartus II TimeQuest Timing Analyzer provides a convenient way to remove them.

Table 7–23 lists commands that allow you to remove constraints and exceptions from a design.

Table 7–23. Constraint and Exception Removal

| Command	Description
remove_clock [-all] [<clock list>]	Removes any clocks specified by <clock list> that have been previously created. The -all option removes all declared clocks.
remove_clock_groups -all	Removes all clock groups previously created. Specific clock groups cannot be removed.
remove_clock_latency -source <targets>	Removes the clock latency constraint from the clock specified by <targets>.
remove_clock_uncertainty -from <from clock> -to <to clock>	Removes the clock uncertainty constraint from <from clock> to <to clock>.
remove_input_delay <targets>	Removes the input delay constraint from <targets>.
remove_output_delay <targets>	Removes the output delay constraint from <targets>.
remove_annotated_delay -all	Removes any annotated delay specified with the set_annotated_delay command.
reset_design	Removes all constraints and exceptions in the design.

Timing Reports

The Quartus II TimeQuest Timing Analyzer provides real-time static timing analysis result reports. Reports are generated only when requested. You can customize which report to display specific timing information, excluding those fields not required.

This section describes various report generation commands supported by the Quartus II TimeQuest Timing Analyzer.

report_timing

Use the `report_timing` command to generate a setup, hold, recovery, or removal report. Example 7–30 shows the `report_timing` command and options.

Example 7–30. report_timing Command

```
report_timing
[-append]
[-detail <summary|path_only|path_and_clock|full_path>]
[-fall_to_clock <names>|-rise_to_clock <names>]
[-to <names>|-to_clock <names>]
[-false_path]
[-file <name>]
[-from <names>]
[-from_clock <names>|-rise_from_clock <names>|-fall_from_clock <names>]
[-less_than_slack <slack limit>]
[-npaths <number>]
[-nworst <number>]
[-pairs_only]
[-panel_name <name>]
[-setup|-hold|-recovery|-removal]
[-show_routing]
[-stdout]
[-through <names>]
```

Table 7–24 describes the options for the `report_timing` command.

Table 7–24. report_timing Command Options (Part 1 of 2)

Option	Description
-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file is overwritten.
-detail <summary|path_only|path_and_clock|full_path>	Specifies whether or not the clock path detail is reported: `Path Only`: Clock network delay is lumped together `Summary`: Lists each individual path `Path and Clock`: Clock network delay is shown in detail `Full Path`: More clock network details, in particular for generated clocks
-fall_from_clock <names>	Specifies the falling edge of the <names> from the source register to be analyzed. The options `from_clock`, `fall_from_clock`, and `rise_from_clock` are mutually exclusive.
-fall_to_clock <names>	Specifies the falling edge of the <names> to the destination register to be analyzed; the options `to_clock`, `fall_to_clock`, and `rise_to_clock` are mutually exclusive.
-false_path	Reports only paths that are cut by a false path assignment.

Table 7–24. report_timing Command Options (Part 2 of 2)

Option	Description
-file <*names*>	Sends the results to an ASCII or HTML file. The extension specified in the file name determines the file type either ***.rpt**, ***.txt**, or ***.html**.
-hold	Specifies a clock hold analysis.
-less_than_slack <*slack limit*>	Limits the paths to be reported to the <*slack limit*> value.
-npaths <*number*>	Specifies the number of paths to report.
-nworst <*number*>	Restricts the number of paths per endpoint.
-panel_name <*names*>	Specifies the name of the panel in the **Reports** pane.
-panel_name <*names*>	Sends the results to the panel and specifies the name of the new panel.
-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	Specifies a recovery analysis.
-removal	Specifies a removal analysis.
-rise_from_clock <*names*>	Specifies the rising edge of the <*names*> from the source register to be analyzed; the options from_clock, fall_from_clock, and rise_from_clock are mutually exclusive.
-rise_to_clock <*names*>	Specifies the rising edge of the <*names*> to the destination register to be analyzed; the options to_clock, fall_to_clock, and rise_to_clock are mutually exclusive.
-setup	Specifies a clock setup analysis.
-show_routing	Option to display detailed routing in the path.
-stdout	Indicates the report be sent to stdout.
-through <*names*>	Specifies the through node for analysis.
-to <*names*>	Specifies the to node for analysis.
-to_clock <*names*>	Specifies the destination clock for analysis.

Example 7–31 shows a sample report that results from typing the following command:

```
report_timing -from_clock clk_async -to_clock clk_async -setup -npaths 1 ↵
```

Example 7–31. Sample report_timing Report (Part 1 of 2)

```
 Info:
 ================================================================
Info: To Node      : dst_reg
Info: From Node    : src_reg
Info: Latch Clock  : clk_async
Info: Launch Clock : clk_async
Info:
Info: Data Arrival Path:
Info:
```

Example 7–31. Sample report_timing Report (Part 2 of 2)

```
Info: Total (ns)  Incr (ns)     Type  Node
Info: =========  ========= ==  ====  =========================
Info:     0.000      0.000                launch edge time
Info:     2.237      2.237  R             clock network delay
Info:     2.410      0.173        uTco    src_reg
Info:     2.410      0.000 RR  CELL       src_reg|regout
Info:     3.407      0.997 RR    IC       dataout|datain
Info:     3.561      0.154 RR  CELL       dst_reg
Info:
Info: Data Required Path:
Info:
Info: Total (ns)  Incr (ns)     Type  Node
Info: =========  ========= ==  ====  =========================
Info:    10.000     10.000                latch edge time
Info:    11.958      1.958  R             clock network delay
Info:    11.610     -0.348        uTsu    dst_reg
Info:
Info: Data Arrival Time  :     3.561
Info: Data Required Time :    11.610
Info: Slack              :     8.049
Info: ==========================================================
```

The `report_timing` command generates a report of the specified analysis type—either setup, hold, recovery, or removal. Each of the column descriptions are described in the Table 7–25.

☞ All columns appear only when a report panel is created. If the `report_timing` output is directed to a file or the console, only the Total, Incr, RF, Type, and Node columns appear.

Table 7–25. Timing Report Data

Column Name	Description
Total	Shows the accumulated time delay.
Incr	Shows the increment in delay.
RF	Shows the input and output transition of the element; this can be one of the following: R, F, RR, RF, FR, FF.
Type	Shows the node type; refer to Table 7–26 of a description of the various node types.
Fanout	Shows the number of fan-outs of the element.
Location	Shows the location of the element in the FPGA.
Element	Shows the name of the element.

Table 7–26 provides a description of the possible node type in the `report_timing` reports.

Table 7–26. Type Description (Part 1 of 2)

Type Name	Description
CELL	Indicates the element is either a register or a combinational element in the FPGA; t.he CELL can be a register in the ALM, memory blocks, DSP blocks, or I/O block
COMP	Indicates the PLL clock network compensation delay.

Table 7–26. Type Description (Part 2 of 2)

Type Name	Description
IC	Indicates the element is an interconnect delay .
utco	Indicates the element's micro clock-to-out time.
utsu	Indicates the element's micro setup time.
uth	Indicates the element's micro hold time.
iext	Indicates the element's external input delay time.
oext	Indicates the element's external output delay time.
LOOP	Indicates a lumped delay bypassing combinational loops.
RE	Indicates a specified routing delay.

report_metastability

Use the `report_metastability` function to estimate the robustness of asynchronous transfers in your design, and to generate a report that details the MTBF for all detected synchronization register chains.

Example 7–32. report_metastability command

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

Table 7–27 describes the options for the `report_metastability` command.

Table 7–27. report_metastability Command Options

Option	Description
`-append`	If output is sent to a file, this option appends the result to that file. Otherwise, the file is overwritten.
`-file <name>`	Sends the results to an ASCII or HTML file. The extension specified in the file name determines the file type—either ***.txt** or ***.html**.
`-panel_name <name>`	Sends the results to the panel and specifies the name of the new panel.
`-stdout`	Indicates the report be sent to `stdout`.

The `report_metastability` function generates a list of synchronization register chains found in the design, and estimates the MTBF of each chain. The design MTBF is an estimate of the overall robustness of the design, computed from the MTBF results from all of the synchronization chains.

The MTBF estimates assume typical conditions, defined as nominal silicon characteristics for the selected device speed grade, as well as nominal voltage and temperature (25°C).

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 are made under the assumption that the arriving data switches once every eight 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.

The design must be fully constrained and pass timing for the features to be available in the TimeQuest Timing Analyzer. If an input port fans out to registers that are not acting as synchronization registers, it is important that a `set_input_delay` constraint is applied to the input port. Without the constraint, the TimeQuest Timing Analyzer views the input port as an asynchronous input, and therefore these registers are reported as synchronization registers. Likewise, registers that are at the end of false paths are also considered synchronization registers.

If you constrain a synchronous input port with a `set_max_delay` constraint, this does not prevent synchronizer identification. Because the `set_max_delay` constraint does not associate the input port with a clock domain, this input port is still considered to be asynchronous. Instead of using the `set_max_delay` constraint, use a `set_input_delay -max` constraint.

For example, instead of specifying setup requirements as:

`set_max_delay <`t_{SU}` requirement>`

use the following syntax:

`set_input_delay -max <latch – launch – `t_{SU}` requirement>`

Settings

Before the TimeQuest Timing Analyzer can analyze a design for metastability, both the Metastability Analysis and Synchronization Register Chain Length options must be set.

Metastability Analysis

This option controls how aggressively the TimeQuest Timing Analyzer and the Fitter identifies registers as being part of a synchronization register chain. The recommended setting for the **Metastability Analysis** option is **AUTO**.

The definitions of various settings of the **Metastability Analysis** option are shown in Table 7–28.

Table 7–28. Settings for the Metastability Analysis Option

Settings	Description
AUTO	■ Identify synchronization chains that does not contain logic.
	■ Report only synchronization register chains of length greater than 1.
OFF	■ No synchronization registers is identified.
FORCED_IF_ASYNCHRONOUS	■ Identify synchronization chains only if it appears like an asynchronous transfer, reported regardless of length, or the presence of logic.
FORCED	■ Identify synchronization chains even if the chain appears synchronous.
	■ This option should not be used on a global basis.

To adjust the **Metastability Analysis** option, in the Quartus II software, on the Settings menu, point to the **TimeQuest Timing Analyzer** page under **Timing Analysis Settings**. From the pull-down menu next to **Metastability Analysis**, select your desired setting.

☞ If you change the **Metastability Analysis** setting, you must rerun the Fitter to account for the new setting.

In addition to the global assignment, the Assignment Editor supports the **Metastability Analysis** option on a node- and entity-level. This option allows you to specify a node or entity and turn off metastability analysis for specific parts of the design. If some synchronization chains are misidentified and you remove them from the report, you can turn off analysis of these paths. To do this, on the Assignment Editor, set **Analyze Metastability** to **OFF** for the first register in these synchronization chains. When this option is turned off, metastability analysis is not performed for these synchronization chains.

Conversely, if there are synchronization chains in your design that were not detected, you can turn on the **Analyze Metastability** option for the first register in this chain, and it is 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 default mode of metastability analysis, these structures are not considered to be synchronizers.

Synchronization Register Chain Length

The **Synchronization Register Chain Length** option prevents the Quartus II software optimization algorithms from being performed on the registers contained in synchronization chain. For example, if the **Synchronization Register Chain Length** is set to 2, optimizations such as register duplication or logic retiming is prevented from being performed on the first two registers in all identified synchronization chains. The default setting for the **Synchronization Register Chain Length** option is 2.

To change the **Synchronization Register Chain Length** option, on the Settings menu, point to **Analysis & Synthesis**. Click **More Settings**. The **More Settings** dialog box appears. From this dialog box, you can set the Synchronization Register Chain Length.

You can also set the Synchronization Register Chain Length on a node or an entity via the Assignment Editor. You can set this value on the head of a synchronization chain to protect only registers in this chain, up to the value you specify. This is useful if you protect a specific synchronization chain up to the third register, but want a global default protection level of two.

Altera recommends that you set this option to the maximum length of synchronization chains you have in your design so that all synchronization registers are preserved.

report_clock_transfers

Use the `report_clock_transfers` command to generate a report that details all clock-to-clock transfers in the design. A clock-to-clock transfer is reported if a path exists between two registers that are clocked by two different clocks. Information such as the number of destinations and sources is also reported.

Use the `report_clock_transfers` command to generate a setup, hold, recovery, or removal report.

Example 7–33 shows the `report_clock_transfers` command and options.

Example 7–33. report_clock_transfers Command

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

Table 7–29 describes the options for the `report_clock_transfers` command.

Table 7–29. report_clock_transfers Command Options

Option	Description
-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file is overwritten.
-file <name>	Sends the results to an ASCII or HTML file. The extension specified in the file name determines the file type either **.txt** or **.html**.
-hold	Creates a clock transfer summary for hold analysis.
-setup	Creates a clock transfer summary for setup analysis.
-stdout	Indicates the report be sent to stdout.
-recovery	Creates a clock transfer summary for recovery analysis.
-removal	Creates a clock transfer summary for removal analysis.
-panel_name <name>	Sends the results to the panel and specifies the name of the new panel.

report_clocks

Use the `report_clocks` command to generate a report that details all clocks in the design. The report contains information such as type, period, waveform (rise and fall), and targets for all clocks in the design.

Example 7–34 shows the `report_clocks` command and options.

Example 7–34. report_clocks Command

```
report_clocks
[-append]
[-desc]
[-file <name>]
[-stdout]
[-panel_name <name>]
```

Table 7–30 describes the options for the `report_clocks` command.

Table 7–30. report_clocks Command Options

Option	Description
-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file is overwritten.
-file <name>	Sends the results to an ASCII or HTML file. The extension specified in the file name determines the file type—either **.txt** or **.html**.
-desc	Specifies the clock names to sort in descending order. The default is ascending order.
-stdout	Indicates the report be sent to stdout.
-panel_name <name>	Sends the results to the panel and specifies the name of the new panel.

report_min_pulse_width

The report_min_pulse_width command checks that a clock high or low pulse is sustained long enough to be recognized as an actual change in the clock signal. A failed minimum pulse width check indicates that the register may not recognize the clock transition. Use the report_min_pulse_width command to generate a report that details the minimum pulse width for all clocks in the design. The report contains information for high and low pulses for all clocks in the design.

The report_min_pulse_width command also reports minimum period checks for RAM and DSP, as well as I/O edge rate limits for input and output clock ports. For output ports, the port must either have a clock (or generated clock) assigned to it or used as the -reference_pin for input/output delays.

The report_min_pulse_width command checks the I/O edge rate limits, but does not always perform the check for output clock ports. For the report_min_pulse_width command to check the I/O edge rate limits for output clock ports, the output clock port must fall into one of the following categories:

■ Have a clock or generated clock constraint assigned to it

or

■ Use a -reference_pin for an input or output delay constraint

Each register in the design is reported twice per clock that clocks the register: once for the high pulse and once for the low pulse. Example 7–35 shows the report_min_pulse_width command and options.

Example 7–35. report_min_pulse_width Command

```
report_min_pulse_width
[-append]
[-file <name>]
[-nworst <number>]
[-stdout]
[<targets>]
[-panel_name <name>]
```

Table 7–31 describes the options for the report_min_pulse_width command.

Table 7–31. report_min_pulse_width Command Options

Option	Description
-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file is overwritten.
-file <*name*>	Sends the results to an ASCII or HTML file. The extension specified in the file name determines the file type—either ***.txt** or ***.html**.
-nworst <*number*>	Specifies the number of pulse width checks to report. The default is 1.
-stdout	Redirects the output to stdout via messages; only required if another output format, such as a file, has been selected and is also to receive messages.
<*targets*>	Specifies registers.
-panel_name <*name*>	Sends the results to the panel and specifies the name of the new panel.

report_net_timing

Use the report_net_timing command to generate a report that details the delay and fan-out information about a net in the design. A net corresponds to a cell's output pin.

Example 7–36 shows the report_net_timing command and options.

Example 7–36. report_net_timing Command

```
report_net_timing
[-append]
[-file <name>]
[-nworst_delay <number>]
[-nworst_fanout <number>]
[-stdout]
[-panel_name <name>]
```

Table 7–32 describes the options for the report_net_timing command.

Table 7–32. report_net_timing Command Options

Option	Description
-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file is overwritten.
-file <*name*>	Sends the results to an ASCII or HTML file. The extension specified in the file name determines the file type—either ***.txt** or ***.html**.
-nworst_delay <*number*>	Specifies that <*number*> worst net delays be reported.
-nworst_fanout <*number*>	Specifies that <*number*> worst net fan-outs be reported.
-stdout	Indicates the report be sent to stdout.
-panel_name <*name*>	Sends the results to the panel and specifies the name of the new panel.

report_sdc

Use the `report_sdc` command to generate a report of all the Synopsys design constraints in the project.

Example 7–37 shows the `report_sdc` command and options.

Example 7–37. report_sdc Command

```
report_sdc
[-ignored]
[-append]
[-file]
[-stdout]
[-panel_name <name>]
```

Table 7–33 describes the options for the `report_sdc` command.

Table 7–33. report_sdc Command Options

Option	Description
-ignored	Reports assignments that were ignored.
-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file is overwritten.
-file	Sends the results to an ASCII or HTML file. The extension specified in the file name determines the file type—either ***.txt** or ***.html**.
-stdout	Indicates that the report is sent to `stdout`.
-panel_name <name>	Sends the results to the panel and specifies the name of the new panel.

report_ucp

Use the `report_ucp` command to generate a report of all unconstrained paths in the design.

Example 7–38 shows the `report_ucp` command and options.

Example 7–38. report_ucp Command

```
report_ucp
[-append]
[-file <name>]
[-hold]
[-setup]
[-stdout]
[-summary]
[-panel_name <name>]
```

Table 7–34 describes the options for the `report_ucp` command.

Table 7–34. Option Descriptions for report_ucp (Part 1 of 2)

Option	Description
-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file is overwritten.
-file <name>	Sends the results to an ASCII or HTML file. The extension specified in the file name determines the file type—either ***.txt** or ***.html**.

Table 7–34. Option Descriptions for report_ucp (Part 2 of 2)

Option	Description
-hold	Reports all unconstrained hold paths.
-setup	Reports all unconstrained setup paths.
-stdout	Indicates the report be sent to stdout.
-summary	Generates only the summary panel.
-panel_name <name>	Sends the results to the panel and specifies the name of the new panel.

Table 7–35 summarizes all reporting commands available in the Quartus II TimeQuest Timing Analyzer.

Table 7–35. Reports from the Tasks Pane and Tcl Commands

Task Pane Report	Tcl Command	Description
Report Setup Summary	create_timing_summary -setup	Generates a clock setup summary for all defined clocks.
Report Hold Summary	create_timing_summary -hold	Generates a clock hold summary for all defined clocks.
Report Recovery Summary	create_timing_summary -recovery	Generates a clock recovery summary for all defined clocks.
Report Removal Summary	create_timing_summary -removal	Generates a clock removal summary for all defined clocks.
Report Clocks	report_clocks	Generates a clock summary for all defined clocks.
Report Clock Transfers	report_clock_transfers	Generates a clock transfer summary for all clock-to-clock transfers in the design.
Report SDC	report_sdc	Generates a summary of all **.sdc** file commands read.
Report Unconstrained Paths	report_ucp	Generates a summary of all unconstrained paths in the design.
Report Timing	report_timing	Generates a detailed summary for specific paths in the design.
Report Net Timing	report_net_timing	Generates a detailed summary for specific nets in the design.
Report Minimum Pulse Width	report_min_pulse_width	Generates a detailed summary for specific registers in the design.
Create Slack Histogram	create_slack_histogram	Generates a detailed histogram for a specific clock in the design.

report_path

Use the `report_path` command to report the longest delay paths and corresponding delay value.

Example 7–39 shows the `report_path` command and options.

Example 7–39. report_path Command

```
report_path
[-append]
[-file <name>]
[-from <names>]
[-npaths <number>]
[-stdout]
[-through]
[-to <names>]
[-panel_name <name>]
```

Table 7–36 describes the options for the `report_path` command.

Table 7–36. report_path Command Options

Option	Description
`-append`	If output is sent to a file, this option appends the result to that file. Otherwise, the file is overwritten.
`-file <name>`	Sends the results to an ASCII or HTML file. The extension specified in the file name determines the file type—either ***.txt** or ***.html**.
`-from <names>`	Specifies the source node for analysis.
`-npaths <number>`	Specifies the number of paths to report.
`-stdout`	Indicates the report be sent to `stdout`.
`-through <name>`	Specifies the through node for analysis.
`-to <names>`	Specifies the destination node for analysis.
`-panel_name <name>`	Sends the results to the panel and specifies the name of the new panel.

report_bottleneck

Use the `report_bottleneck` command to report a rating per node based on the number of failing paths through each node for the worst 1,000 setup paths. Example 7–40 shows the `report_bottleneck` command and options.

Example 7–40. report_bottleneck Command

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

By default, the `report_bottleneck` command reports a rating for the worst 1,000 setup paths.

In addition to the default metric, there are a few additional "standard" metrics to choose from, such as:

■ `-metric num_fanouts`

■ `-metric tns`

You can also create a custom metric to evaluate the nodes based on the combination of the number of fanouts, fanins, failing paths, total paths, and other keepers. 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`.

Table 7–37 describes the options for the `report_bottleneck` command.

Table 7–37. report_bottleneck Command

Option	Description
`-cmetric <cmetric_name>`	Custom metric function to evaluate individual nodes.
`-details`	Show the detailed information (number of failing edges, number of fanins, and so forth).
`-metric <default|tns|num_paths|num_fpaths|num_fanins|num_fanouts>`	Indicate the metric to use to rate individual nodes.
`-panel <panel_name>`	Sends the results to the panel and specifies the name of the new panel.
`-stdout`	Indicates the report be sent to stdout.
`<paths>`	Paths to be analyzed.

Example 7–41 shows how to create a custom metric with the `report_bottleneck` command.

Example 7–41. report_bottleneck Custom Metric

```
#set the number of paths to be reported
set paths [ get_timing_paths -npaths 1000 -setup ]

#create the custom metric
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
}
#reporting the results of the custom metric
report_bottleneck -cmetric report_bottleneck_custom_metric -panel
"Timing Analysis Bottleneck Report - Custom" $paths
```

report_datasheet

Use the `report_datasheet` command to generate a datasheet report which summarizes the timing characteristics of the entire design. It reports setup (`tsu`), hold (`th`), clock-to-output (`tco`), minimum clock-to-output (`mintco`), propagation delay (`tpd`), and minimum propagation delay (`mintpd`) times. Example 7–42 shows the `report_datasheet` command and options.

Example 7–42. report_datasheet Command

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

Table 7–38 describes the options for the `report_datasheet` command.

Table 7–38. report_datasheet Command Options

Option	Description
-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file is overwritten.
-file <name>	Sends the results to an ASCII or HTML file. The extension specified in the file name determines the file type—either ***.txt** or ***.html**.
-stdout	Indicates the report be sent to stdout.
-panel_name <name>	Sends the results to the panel and specifies the name of the new panel.

The delays are reported with respect to a base clock or port for which they are relevant. If there is a case where there are multiple paths for a clock, the maximum delay of the longest path is taken for the t_{SU}, t_H, t_{CO}, and t_{PD}, and the minimum delay of the shortest path is taken for $mint_{CO}$ and $mint_{PD}$.

report_rskm

Use the `report_rskm` command to generate a report that details the receiver skew margin for LVDS receivers.

Example 7–43 shows the `report_rskm` command and options.

Example 7–43. report_rskm Command

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

Table 7–39 describes the options for the report_rskm command.

Table 7–39. report_rskm Command Options

Type Name	Description
-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file is overwritten.
-file <*name*>	Sends the results to an ASCII or HTML file. The extension specified in the file name determines the file type—either ***.txt** or ***.html**.
-panel_name <*name*>	Sends the results to the panel and specifies the name of the new panel.
-stdout	Indicates the report be sent to stdout.

The 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 formula shown in Equation 7–12:

Equation 7–12.

$$RSKM = \frac{(TUI - SW - RCCS)}{2}$$

The time unit interval is the LVDS clock period ($1/f_{MAX}$). 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 reflects channel-to-channel skew seen by the LVDS receiver. This RCCS includes transmitter channel-to-channel skew (TCCS) of the upstream transmitter and maximum channel-to-channel skew between the transmitter and receiver. RCCS is equal to the difference between the maximum input delay and minimum input delay. If no input delay is set, RCCS defaults to zero.

report_tccs

Use the report_tccs command to generate a report that details the channel-to-channel skew margin for LVDS transmitters.

Example 7–44 shows the report_tccs command and options.

Example 7–44. report_tccs Command

```
report_tccs
[-append]
[-file <name>]
[-panel_name <name>]
[-quiet]
[-stdout]
```

Table 7–40 describes the options for the report_tccs command.

Table 7–40. report_tccs Command Options

Type Name	Description
-append	Specifies that the current report be appended to the file specified by the -file option.
-file <name>	Indicates that the current report is written to the file <name>.
-panel_name <name>	Sends the results to the panel and specifies the name of the new panel.
-quiet	Specifies that nothing is printed if there are no LVDS receivers in the design.
-stdout	Indicates the report be sent to stdout.

The TCCS is the timing difference between the fastest and slowest output transitions, including t_{CO} variations and clock skew.

report_path

Use the report_path command to generate a report that details the longest delay paths between any two arbitrary keeper nodes.

Example 7–45 shows the report_path command and options.

Example 7–45. report_path Command

```
report_path
[-append]
[-file <name>]
[-from <names>]
[-min_path]
[-npaths <number>]
[-nworst <number>]
[-panel_name <name>]
[-stdout]
[-summary]
[-through <names>]
[-to <names>]
```

Table 7–41 describes the options for the report_path command.

Table 7–41. report_path Command Options (Part 1 of 2)

| Type Name	Description
-append	Specifies that the current report be appended to the file specified by the -file option.
-file <name>	Indicates that the current report is written to the file <name>.
-from <names>	The <names> is a collection or list of objects in the design. The <names> acts as the start point of the path.
-min_path	Displays the minimum delay paths.
-npaths <number>	Specifies the number of paths to report.
-nworst <number>	Specifies the maximum number of paths to report for each endpoint.
-panel_name <name>	Sends the results to the panel and specifies the name of the new panel.
-stdout	Indicates the report be sent to stdout.
-summary	Creates a single table with a summary of each path found.

Table 7–41. report_path Command Options (Part 2 of 2)

Type Name	Description
`-through <names>`	The <*names*> is a collection or list of objects in the design. Specifies false path passes through <*names*>.
`-to <names>`	The <*names*> is a collection or list of objects in the design. The <*names*> acts as the end point of the path.

☞ The delay path reported cannot pass through a keeper node; for example, a register or port. Instead, the delay path must be from the output pin of a keeper node to the input pin of a keeper node.

Figure 7–37 shows a simple design with a register-to-register path.

Figure 7–37. Simple Register-to-Register Path
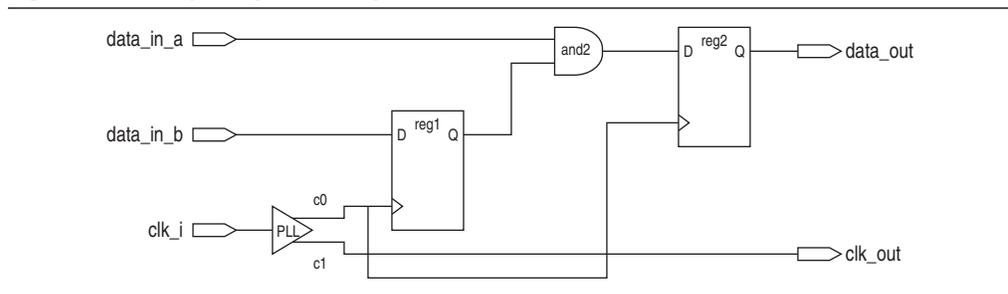
Example 7–46 shows the report generated from the following command:

```
report_path -from [get_pins {reg1|regout}] -to [get_pins \
{reg2|datain}] -npaths 1 -panel_name "Report Path" -stdout
```

Example 7–46. report_path from Keeper Output Pin to Keeper Input Pin

```
 Info: =============================================================
Info: From Node : reg1|regout
Info: To Node : reg2|datain
Info:
Info: Path:
Info:
Info: Total (ns) Incr (ns) Type Element
Info: ========== ========= == ==== ==================
Info: 0.000 0.000 reg1|regout
Info: 0.206 0.206 RR IC and2|datae
Info: 0.360 0.154 RR CELL and2|combout
Info: 0.360 0.000 RR IC reg2|datain
Info:
Info: Total Path Delay : 0.360
Info: =============================================================
```

Example 7–47 shows the report generated from the following command:

```
report_path -from [get_ports data_in_a] -to [get_pins {reg2|regout}] \
-npaths 1
```

Example 7–47. report_path from Keeper-to-Keeper Output Pin

```
Info: Report Path: No paths were found
0 0.000
```

No paths were reported in Example 7–47 because the destination passes through an input pin of a keeper node.

report_net_delay

Use the report_net_delay to generate a slack report for paths constrained with the set_net_delay command. The report_net_delay command 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. Example 7–48 shows the report_net_delay command and options.

Example 7–48. report_net_delay Command

```
report_net_delay
    [-append]
    [-file <name>]
    [-nworst <number>]
    [-panel_name <name>]
    [-stdout]
```

Table 7–42 describes the options for the report_net_delay command.

Table 7–42. report_net_delay Command Options

Options	Description
-append	If output is sent to a file, this option appends the result to that file. Otherwise, the file is overwritten.
-file <name>	Sends the results to an ASCII or HTML file. The extension specified in the file name determines the file type—either ***.txt** or ***.html**.
-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 have to use this option if you have selected another output format, such as a file, and would also like to receive messages.

check_timing

Use the check_timing command to generate a report on any potential problem with the design or applied constraints. Not all check_timing results are serious issues. The results should be examined to see if the results are desired. Example 7–49 shows the check_timing command and options.

Example 7–49. check_timing Command

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

Table 7–43 describes the options for the check_timing command.

Table 7–43. check_timing Command Options

Option	Description
-append	Specifies that the current report be appended to the file specified by the -file option.
-file <*name*>	Indicates that the current report is written to the file <*name*>.
-include	Indicates that a check is to be performed with the <*check_list*>. Refer to Table 7–44 for a list of checks.
-stdout	Indicates the report be sent to stdout.
-panel_name <*name*>	Sends the results to the panel and specifies the name of the new panel.

Table 7–44 describes the possible checks.

Table 7–44. Possible Checks (Part 1 of 2)

Option	Description
no_clock	Checks that registers have at least one clock at their clock pin and ensures that ports determined to be clocks have a clock assigned to them.
multiple_clock	Checks that registers have at most one clock at their clock pin. When multiple clocks reach a register's clock pin, both clocks are used for analysis.
generated_clock	Checks that generated clocks are valid. Generated clocks must have a source that is clocked by a valid clock. They must also not depend on each other in a loop (clk1 cannot have clk2 as a source if clk2 already uses clk1 as a source).
no_input_delay	Checks that every input port that is not determined to be a clock has an input delay set on it.
no_output_delay	Checks that every output port has an output delay set on it.
partial_input_delay	Checks that input delays are complete. Makes sure that input delays have a rise-min, fall-min, rise-max, and fall-max portion set.
partial_output_delay	Checks that output delays are complete. Makes sure that output delays have a rise-min, fall-min, rise-max, and fall-max portion set.
reference_pin	Checks if 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 fan-in of the reference_pin. Being in the direct fan-in of the reference_pin means that there must be no keepers between the clock and reference_pin.
latency_override	Ensures that the clock latency constraint applied on a port or pin overrides the more generic clock latency set on a clock. Clock latency set to a clock applies to all keepers clocked by the clock. Clock latency set to a pin or a port applies to registers in the fan-out of the port or pin.
loops	Checks that there are no strongly connected components in the design. These loops prevent a design from being properly analyzed. Indicates that loops exist but were marked so that they would not be traversed.
latches	Checks whether there are latches in the design. The Quartus II TimeQuest Timing Analyzer warns the user that the latches exist and cannot be properly analyzed.
pos_neg_clock_domain	Checks whether 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.

Table 7–44. Possible Checks (Part 2 of 2)

Option	Description
`pll_cross_check`	Checks the clocks that are assigned to a PLL against the PLL setting definted in the user's design files. Inconsistent setting or an unmatched number of clocks associated with the PLL are reported to the user.
`no_uncertainty`	Checks that each clock-to-clock transfer has a clock uncertainty assignment set between the two clocks.
`virtual_clock`	Checks that each virtual clock is referenced.
`partial_multicycle`	Checks that each setup multicycle assignment has a corresponding hold multicycle assignment, and each hold multicycle assignment has a corresponding setup multicyle assignment.
`multicycle_consistency`	Checks that all of the multicycle cases in which a setup multicycle does not equal one greater than the hold multicycle. Hold multicycle assignments are usually one cycle fewer than setup multicycle assignments.
`partial_min_max_delay`	Verifies that each minimum delay assignment has a corresponding maximum delay assignment, and that each maximum delay assignment has a corresponding minimum delay assignment.
`clock_assignments_` `on_output_ports`	Checks that reports all of the clock assignments that have been applied to output ports.
`generated_io_delay`	Checks that all of the I/O delays with no reference pin, generated `-clock`, or no `-source_atencey_included`.

Example 7–50 shows how the `check_timing` command can be used.

Example 7–50. The check_timing Command

```
# 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 for combinational loops, latches, or
# misssing or incomplete input delays
check_timing -include {loops latches no_input_delay
partial_input_delay}
```

report_clock_fmax_summary

Use the `report_clock_fmax_summary` to report potential f_{MAX} for every clock in the design, regardless of the user-specified clock periods. f_{MAX} 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, f_{MAX} is computed as if the rising and falling edges are scaled along with f_{MAX}, such that the duty cycle (in terms of a percentage) is maintained.

Example 7–51 shows the `report_clock_fmax_summary` command and options.

Example 7–51. report_clock_fmax_summary Command

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

Table 7–45 describes the options for the report_clock_fmax_summary command.

Table 7–45. report_clock_fmax_summary Command Options

Option	Description
-append	Specifies that the current report be appended to the file specified by the -file option.
-file <name>	Indicates that the current report is written to the file <name>.
-stdout	Indicates the report be sent to stdout.
-panel_name <name>	Sends the results to the panel and specifies the name of the new panel.

The f_{MAX} Summary report contains four columns: f_{MAX}, Restricted f_{MAX}, Clock Name, and Note. The description of each column is given in Table 7–46.

Table 7–46. f_{MAX} Summary Report Column

Column Name	Description
f_{MAX}	Shows the fastest possible frequency at which the internal register-to-register can run. This is not the fastest the clock port can be driven.
Restricted f_{MAX}	Shows fastest possible frequency at which the clock port can run. This number may be lower than the f_{MAX} column for various reasons, including hold timing requirements, I/O edge rate limits for clocks (which also depends on I/O standards), minimum pulse width checks for registers, and minimum period checks for RAM and DSP registers.
Clock Name	Shows the clock name.
Note	Shows any notes related to the clock.

create_timing_summary

Reports the worst-case clock setup and clock hold slacks and endpoint total negative slack (TNS) per clock domain. Total negative slack is the sum of all slacks less than zero for each destination register or port in the clock domain.

Example 7–52 shows the create_timing_summary command and options.

Example 7–52. create_timing_summary Command

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

Table 7–47 describes the options for the create_timing_summary command.

Table 7–47. create_timing_summary Command Options

Option	Description
-append	Specifies that the current report be appended to the file specified by the -file option.
-file <name>	Indicates that the current report is written to the file <name>.
-hold	Generates a clock hold check summary report.
-panel_name <name>	Sends the results to the panel and specifies the name of the new panel.
-recovery	Generates a recovery check summary report.
-removal	Generates a removal check summary report.
-setup	Generates a clock setup check summary report.
-stdout	Indicates the report be sent to stdout.

Timing Analysis Features

The TimeQuest Timing Analyzer supports many features that enhances and provides a through analysis of your designs. This section covers the many features available in the TimeQuest Timing Analyzer.

Multi-Corner Analysis

Multi-corner analysis allows a design to be verified under a variety of operating conditions (voltage, process, and temperature) while performing a static timing analysis on the design.

Use the set_operating_conditions command to change the operating conditions or speed grade of the device used for static timing analysis.

Example 7–53 shows the set_operating_conditions command and options.

Example 7–53. set_operating_conditions Command

```
set_operating_conditions
[-model <fast|slow>]
[-speed <speed grade>]
[-temperature <value in °C>]
[-voltage <value in mV>]
[<operating condition Tcl object>]
```

Table 7–48 describes the options for the set_operating_conditions command.

Table 7–48. set_operating_conditions Command Options

Option	Description
-model <fast|slow>	Specifies the timing model.
-speed <speed grade>	Specifies the device speed grade.
-temperature <value in °C>	Specifies the operating temperature.
-voltage <value in mV>	Specifies the operating voltage.
<operating condition Tcl object>	Specifies the operating condition Tcl object that specifies the operating conditions.

☞ If an operating condition Tcl object *is* used, the model, speed, temperature, and voltage options are not required. If an operating condition Tcl object *is not* used, the model must be specified, and the -speed, -temperature, and -voltage options are optional, using the appropriate defaults for the device where applicable.

Table 7–49 shows a few of the available operating conditions that can be set for each device family.

Table 7–49. Device Family Operating Conditions

| Device Family | Available Conditions | | | | Operating Condition Tcl Objects |
	Speed Grade	Model	Voltage (mV)	Temp (°C)	
Stratix III	4	Slow	1100	85	`4_slow_1100mv_85c  4_slow_1100mv_0c`
		Slow	1100	0	`MIN_fast_1100mv_0c`
		Fast	1100	0	
Cyclone® III	7	Slow	1200	85	`7_slow_1200mv_85c  7_slow_1200mv_0c`
		Slow	1200	0	`MIN_fast_1200mv_0c`
		Fast	1200	0	
Stratix II	4	Slow	N/A	N/A	`4_slow`
		Fast			`MIN_fast`
Cyclone II	6	Slow	N/A	N/A	`6_slow`
		Fast			`MIN_fast`

☞ Use the `get_available_operating_conditions-all` command to obtain a list of available operating conditions for the target device.

Example 7–54 shows how to set the operating conditions for a Stratix III design to the slow model, 1100 mV, and 85° C.

Example 7–54. Setting Operating Conditions with Individual Values

```
set_operating_conditions -model slow -temperature 85 -voltage 1100
```

Alternatively, you can set the operating conditions in Example 7–54 with the Tcl object as shown in Example 7–55.

Example 7–55. Setting Operating Conditions with a Tcl Object

```
set_operating_conditions 4_slow_1100mv_85c
```

Advanced I/O Timing and Board Trace Model Assignments

The Quartus II TimeQuest Timing Analyzer is able to use Advanced I/O Timing and Board Trace Model assignments to model I/O buffer delays in your design.

To turn the Advanced I/O feature on or off, in the **Settings** dialog box, under the **TimeQuest Timing Analyzer** option, choose **on** or **off**.

If you turn the **Advanced I/O Timing** on or offor change Board Trace Model assignments and do not recompile before you analyze timing, you must use the `-force_dat` command when you create the timing netlist. Type the following command in the Tcl console of the Quartus II TimeQuest Timing Analyzer:

```
create_timing_netlist -force_dat  ↵
```

If you turn the **Advanced I/O Timing** or change Board Trace Model assignments on or off and recompile before you analyze timing, you do not have to use the `-force_dat` command when you create the timing netlist. You can create the timing netlist with the `create_timing_netlist` command, or with the **Create Timing Netlist** task in the **Tasks** pane.

For more information about the Advanced I/O Timing feature, refer to the *I/O Management* chapter in volume 2 of the *Quartus II Handbook*.

Wildcard Assignments and Collections

To simplify the task of applying constraints to many nodes in a design, the Quartus II TimeQuest Timing Analyzer accepts the "*" and "?" wildcard characters. Use these wildcard characters to reduce the number of individual constraints you must specify in your design.

The "*" wildcard character matches any string. For example, given an assignment made to a node specified as `reg*`, the Quartus II TimeQuest Timing Analyzer searches for and applies the assignment to all design nodes that match the prefix `reg` with none, one, or several characters following, such as `reg1`, `reg[2]`, `regbank`, and `reg12bank`.

The "?" wildcard character matches any single character. For example, given an assignment made to a node specified as `reg?`, the Quartus II TimeQuest Timing Analyzer searches and applies the assignment to all design nodes that match the prefix `reg` and any single character following; for example, `reg1`, `rega`, and `reg4`.

Both the collection commands `get_cells` and `get_pins` have three options that allow you to refine searches that include the wildcard character. To refine your search results, select the default behavior, the `-hierarchical` option, or the `-compatibility` option.

☞ The pipe character is used to separate one hierarchy level from the next in the Quartus II TimeQuest Timing Analyzer. For example, *<absolute full cell name>*|*<pin suffix>* represents a hierarchical pin name with the "|" separating the hierarchy from the pin name.

When you use the collection commands `get_cells` and `get_pins` without an option, the default search behavior is performed on a per-hierarchical level of the pin name; that is, the search is performed level by level. A full hierarchical name may contain multiple hierarchical levels where a "|" is used to separate the hierarchical levels, and each wildcard character represents only one hierarchical level. For example,"*" represents the first hierarchical level and "*|*" represents the first and second hierarchical levels.

When you use the collection commands `get_cells` and `get_pins` with the `-hierarchical` option, a recursive match is performed on the relative hierarchical path name of the form *<short cell name>|<pin name>*. The search is performed on the node name; for example, the last hierarchy of the name and not the hierarchy path. Unlike the default behavior, this option does not limit the search to each hierarchy level represented by the pipe character.

☞ The pipe character cannot be used in the search with the `get_cells` `-hierarchical` option. However, the pipe character can be used with the `get_pins` collection search.

When you use the collection commands `get_cells` and `get_pins` with the `-compatibility` option, the search performed is similar to that of the Quartus II Classic Timing Analyzer. This option searches the entire hierarchical path and pipe characters are not treated as special characters.

Assuming the following cells exist in a design:

```
foo
foo|bar
```

and the following pin names:

```
foo|dataa
foo|datab
foo|bar|datac
foo|bar|datad
```

Table 7–50 shows the results of using these search strings.

Table 7–50. Sample Search Strings and Search Results

Search String	Search Result
`get_pins *|dataa`	`foo|dataa`
`get_pins *|datac`	*<empty>*
`get_pins *|*|datac`	`foo|bar|datac`
`get_pins foo*|*`	`foo|dataa, foo|datab`
`get_pins -hierarchical *|*|datac`	*<empty>* *(1)*
`get_pins -hierarchical foo|*`	`foo|dataa, foo|datab`
`get_pins -hierarchical *|datac`	`foo|bar|datac`
`get_pins -hierarchical foo|*|datac`	*<empty>* *(1)*
`get_pins -compatibility *|datac`	`foo|bar|datac`
`get_pins -compatibility *|*|datac`	`foo|bar|datac`

Note to Table 7–50:

(1)   Due to the additional `*|*|` in the search string, the search result is *<empty>*.

Resetting a Design

Use the `reset_design` command to remove all timing constraints and exceptions from the design under analysis. The command removes all clocks, generated clocks, derived clocks, input delays, output delays, clock latency, clock uncertainty, clock groups, false paths, multicycle paths, min delays, and max delays.

This command provides a convenient way to return to the initial state of analysis without the need to delete and re-create a new timing netlist.

Cross-Probing

The cross-probing feature allows you to locate paths and elements from the TimeQuest Timing Analyzer to various tools available in the Quartus II software (and vice versa).

From the TimeQuest GUI, you can right-click any path in the **View** pane and select either **Locate Path** or **Locate**.

The source is the element in the **From Node** column and the destination is the element in the **To Node** column.

The **Locate Path** option allows you to located the data arrival path, default, of the currently selected row. To locate the data required time path select a row in the data required path panel.

☞ The **Locate Required Path** command is available only when there is a path to show; unless the user reports the clock path as well, there is probably only a single node in the required path. In this case, the command is not available.

The **Locate** option allows you to locate the highlighted element.

The **Locate Path** and **Locate** commands can cross-probe to either the Chip Planner, Technology Map Viewer, or Resource Property Editor. Additionally, the **Locate Path** option can cross-probe to Critical Path Settings.

From the **Critical Path Settings** dialog box in the Chip Planner, you can cross-probe to the TimeQuest Timing Analyzer to report critical paths in the design.

locate

Use the `locate` command in the Console pane to cross-probe to the Chip Editor, Critical Path Settings, Resource Property Editor, and the Technology Map Viewer.

Example 7–56 shows the `locate` command and options.

Example 7–56. locate Command

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

Table 7–51 describes the options for the `locate` command.

Table 7–51. locate Options

Option	Description
-chip	Locates the object in the Chip Planner.
-color <black|blue|brown|green| grey|light_grey|orange| purple|red|white>	Identifies the objects you are locating.
-cps	Locates the object in the Critical Path Settings dialog of the Chip Planner.
-label <label>	Specifies a label used to identify the objects you are locating.
-rpe	Locates in the Resource Property Editor.
-tmv	Locates 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.

Example 7–57 shows how to cross-probe ten paths from TimeQuest Timing Analyzer to the Chip Editor and locate all data ports in the Technology Map Viewer.

Example 7–57. Cross-probing from TimeQuest

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

# locate all ports that begin with data to the Tech Map Viewer

locate [get_ports data*] -tmv
```

The TimeQuest Timing Analyzer GUI

The Quartus II TimeQuest Timing Analyzer provides an intuitive and easy-to-use GUI that allows you to efficiently constrain and analyze your designs. The GUI consists of the following panes:

■ "The Quartus II Software Interface and Options" described on page 7–79

■ "View Pane" described on page 7–80

■ "Tasks Pane" described on page 7–82

■ "Console Pane" described on page 7–84

■ "Report Pane" described on page 7–84

■ "Constraints" described on page 7–84

■ "Name Finder" described on page 7–86

■ "Target Pane" described on page 7–87

■ "SDC Editor" described on page 7–88

Each pane provides features that enhance productivity (Figure 7–38).

Figure 7–38. The TimeQuest GUI
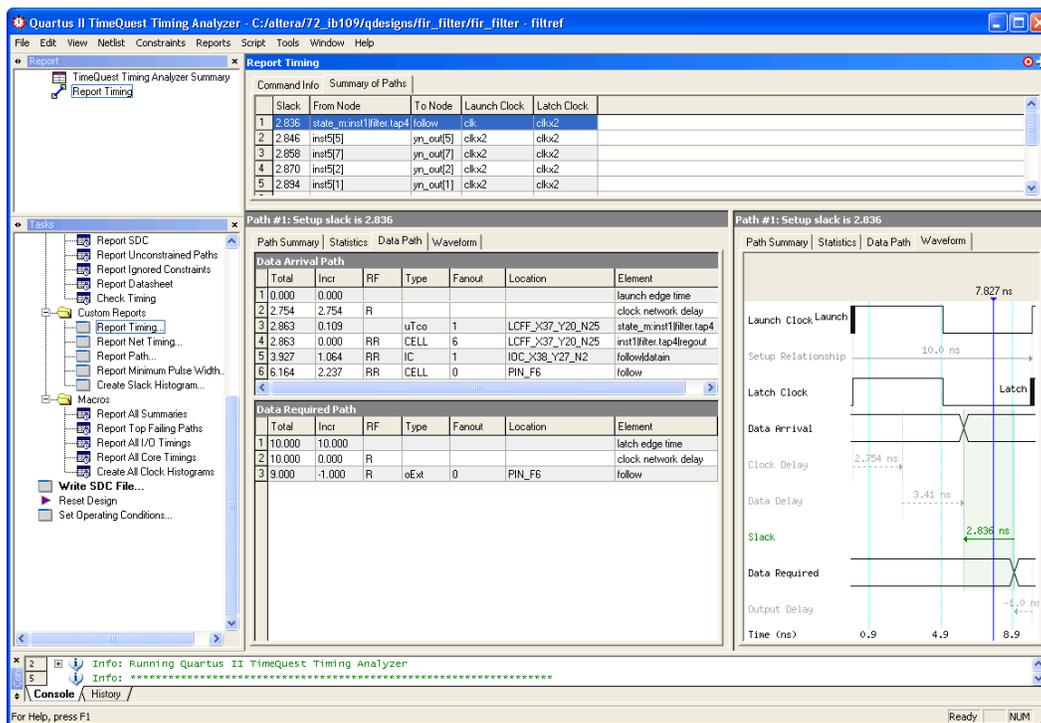
The Quartus II Software Interface and Options

The Quartus II software allows you to configure various options for the Quartus II TimeQuest Timing Analyzer report generation that are generated in the Compilation Report for the design.

The TimeQuest Timing Analyzer settings, in the **Settings** dialog box, allow you to configure the options shown in Table 7–52.

Table 7–52. The Quartus II TimeQuest Timing Analyzer Settings

Options	Description
.sdc files to include in the project	Adds and removes **.sdc** files associated with the project.
Enable Advanced I/O Timing	Generates advanced I/O timing results from board trace models specified for each pin.
Enable multicorner timing analysis during compilation	Generates multiple reports for all available operating conditions of the target device.
Report worst-case paths during compilation	Generates worst-case path reports per clock domain.
Tcl Script File for customizing report during compilation	Specifies any custom scripts to be sourced for any custom report generation.

☞   The options shown in Table 7–52 control only the reports generated in the Compilation Report, and do not control the reports generated in the Quartus II TimeQuest Timing Analyzer.

View Pane

The View pane is the main viewing area for the timing analysis results. Use the View pane to view summary reports, custom reports, or histograms. Figure 7–39 shows the View pane after you select the **Summary (Setup)** report from the Report pane.

Figure 7–39. Summary (Setup) Report
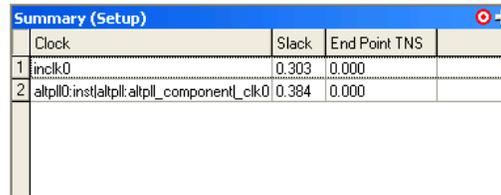
View Pane: Splitting

For analyzing the timing results properly, comparing multiple reports is extremely important. To facilitate multiple report viewing, the View pane supports window splitting. Window splitting divides the View pane into multiple windows, allowing you to view different reports side-by-side.

You can split the View pane into multiple windows using the split icon located in the upper right corner of the View pane. Drag the icon in different directions to generate additional window views in the View pane. For example, if you drag the split icon to the left, the View pane creates a new window to the right of the current window (Figure 7–40).

Figure 7–40. Splitting the View Pane to the Left (Before and After Split Left)

If you drag the split icon diagonally, the View pane creates three new windows in the View pane (Figure 7–41).

Figure 7–41. Splitting the View Pane Diagonally (Before and After Diagonal Split)
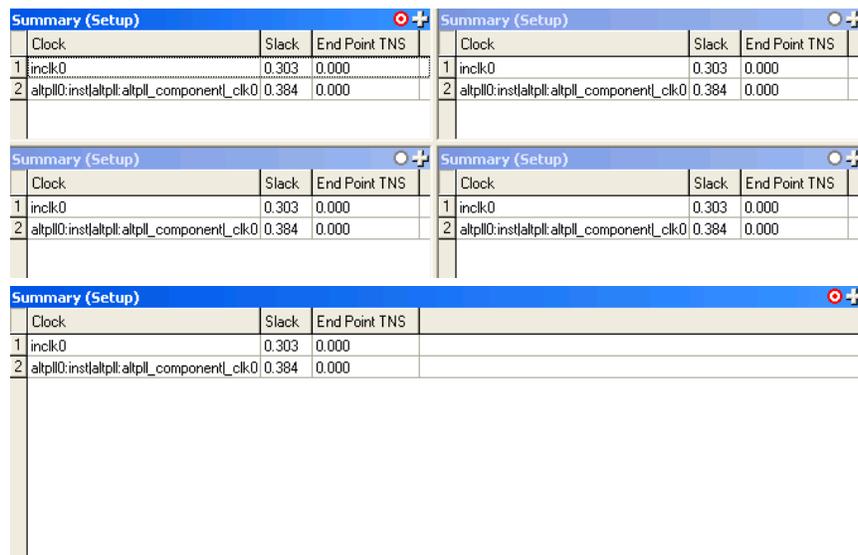
Drag the split icon downward to create a new window directly below the current window.

View Pane: Removing Split Windows

You can remove windows that you create in the View pane using the split icon by dragging the border of the window over the window you wish to remove (Figure 7–42).

Figure 7–42. Removing a Split Window (Before and After Split is Removed)
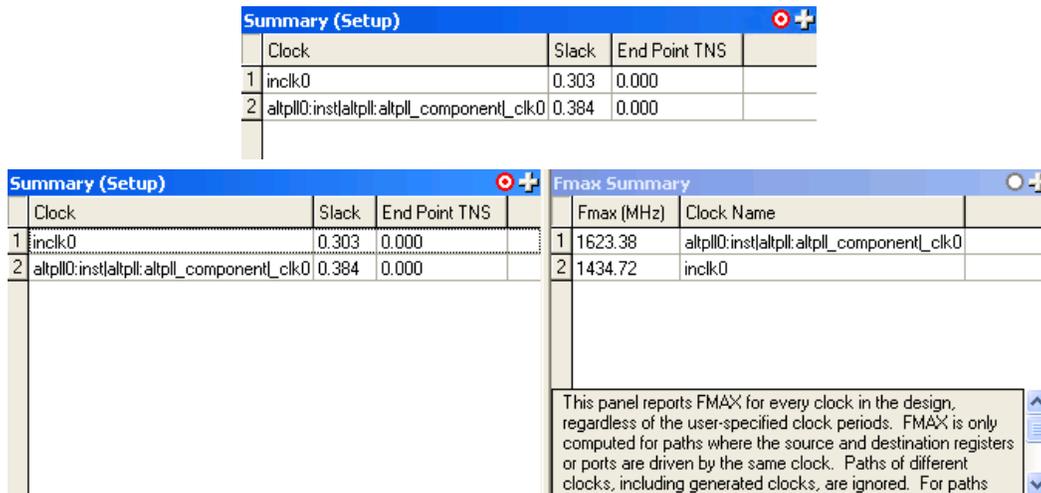
Tasks Pane

Use the Tasks pane to access common commands such as netlist setup and report generation.

The following common commands are located in the Tasks pane: Open Project, Set Operating Conditions, and Reset Design. The other commands, including timing netlist setup and report generation, are contained in the following folders:

■ Netlist Setup

■ Reports

☞ Each command in the Tasks pane has an equivalent Tcl command that is displayed in the Console pane when the command runs.

Opening a Project and Writing a Synopsys Design Constraints File

To open a project in the Quartus II TimeQuest Timing Analyzer, double-click the **Open Project** task. If you launch the Quartus II TimeQuest Timing Analyzer from the Quartus II software GUI, the project opens automatically.

You can add or remove constraints from the timing netlist after the Quartus II TimeQuest Timing Analyzer reads the initial **.sdc** file. After the file is read, the initial **.sdc** file becomes outdated compared to the constraints in the Quartus II TimeQuest Timing Analyzer. Use the **Write SDC File** command to generate an **.sdc** file that is up-to-date and reflects the current state of constraints in the Quartus II TimeQuest Timing Analyzer.

Netlist Setup Folder

The Netlist Setup folder contains tasks that are used to set up the timing netlist for timing analysis. The three tasks located in this folder are Create Timing Netlist, Read SDC File, and Update Timing Netlist.

Use the Create Timing Netlist task to create a netlist that the Quartus II TimeQuest Timing Analyzer uses to perform static timing analysis. This netlist is used only for timing analysis by the Quartus II TimeQuest Timing Analyzer.

☞ You must always create a timing netlist before you perform an analysis with the Quartus II TimeQuest Timing Analyzer.

Use the Read SDC File command to apply constraints to the timing netlist. By default, the Read SDC File command reads the *<current revision>***.sdc** file.

☞ Use the `read_sdc` command to read an **.sdc** file that is not associated with the current revision of the design.

Use the Update Timing Netlist command to update the timing netlist after you enter constraints or read an **.sdc** file. You should use this command if any constraints are added or removed from the design.

Reports Folder

The Reports folder contains commands to generate timing summary reports of the static timing analysis results. The twelve commands located in this folder are summarized in Table 7–53.

Table 7–53. Reports Folder Commands

Report Task	Description
Report Fmax Summary	Generates a f_{MAX} summary report for all clocks in the design.
Report Setup Summary	Generates a clock setup summary report for all clocks in the design.
Report Hold Summary	Generates a clock hold summary report for all clocks in the design.
Report Recovery Summary	Generates a recovery summary report for all clocks in the design.
Report Removal Summary	Generates a removal summary report for all clocks in the design.
Report Clocks	Generates a summary report of all created clocks in the design.
Report Clock Transfers	Generates a summary report of all clock transfers detected in the design.
Report Minimum Pulse Width	Generates a summary report of all minimum pulse widths in the design.
Report SDC	Generates a summary report of the constraints read from the **.sdc** file.
Report Unconstrained Paths	Generates a summary report of all unconstrained paths in the design.
Report Ignored Constraints	Generates a summary report of all ignored SDC constraints for the design.
Report Datasheet	Generates a datasheet report for the design.

Macros Folder

The Macros folder contains commands that perform custom tasks available in the Quartus II TimeQuest Timing Analyzer utility package. These commands are: Report All Summaries, Report Top Failing Paths, and Create All Clock Histograms.

Table 7–54 describes the commands available in the Macros folder.

Table 7–54. Macros Folder Commands

Macro Task	Description
Report All Summaries	This command runs the Report Setup Summary, Report Hold Summary, Report Recovery Summary, Report Removal Summary, and Minimum Pulse Width commands to generate all summary reports.
Report Top Failing Paths	This command generates a report containing a list of top failing paths.
Create All Clock Histograms	This command runs the Create Slack Histogram command to generate a clock histogram for all clocks in the design.
Report All I/O Timings	This command generates a report of all timing paths that start or end at a device port.
Report All Core Timings	This command generates a report of all timing paths that start and end at the device register.

Console Pane

The Console pane is both a message center for the Quartus II TimeQuest Timing Analyzer and an interactive Tcl console. The Console pane has two tabs: the **Console** tab and the **History** tab. The **Console** tab shows all messages, such as info and warning messages. Also, the Console tab allows you to enter and run Synopsys design constraints and Tcl commands. The Console tab shows the Tcl equivalent of all commands that you run in the Tasks pane. The History tab records all the Synopsys design constraints and Tcl commands that are run.

☞ To run the commands located in the History tab after the timing netlist has been updated, right-click the command and click **Rerun**.

You can copy Tcl commands from the Console and History tabs to easily generate Tcl scripts to perform timing analysis.

Report Pane

Use the Report pane to access all reports generated from the Tasks pane, and by any custom report commands. When you select a report in the Report pane, the report is shown in the active window in the View pane.

☞ If a report is out-of-date with respect to the current constraints, a "?" icon is shown next to the report.

Constraints

Use the Constraints menu to access commonly used constraints, exceptions, and commands. You can access this menu from the toolbar, click **Edit** and then click **Constraint menu**.

The following commands are available on the Constraints menu:

■ **Create Clock**

■ **Create Generated Clock**

■ **Set Clock Latency**

■ **Set Clock Uncertainty**

■ **Set Clock Groups**

■ **Remove Clock**

For example, use the **Create Clock** dialog box to create clocks in your design. Figure 7–43 shows the **Create Clock** dialog box.

Figure 7–43. Create Clock Dialog Box
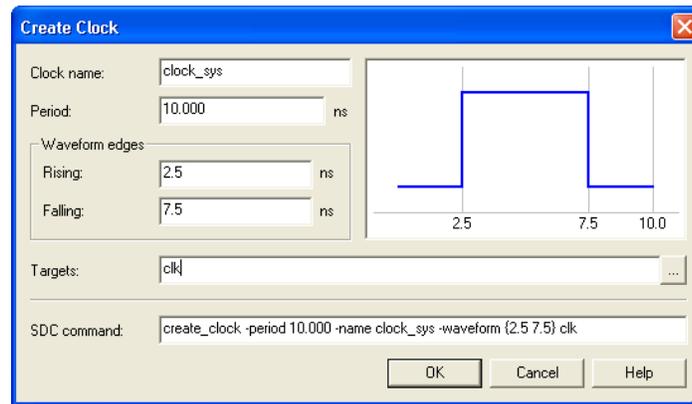
The following commands specify timing exceptions and are available on the Constraints menu:

■ **Set False Path**

■ **Set Multicycle Path**

■ **Set Maximum Delay**

■ **Set Minimum Delay**

All the dialog boxes used to specify timing constraints or exceptions from commands have an SDC command field. This field contains the SDC command that is run when you click OK.

☞ All commands and constraints created in the Quartus II TimeQuest Timing Analyzer user interface are echoed in the **Console** pane.

The constraints specified with Constraints menu commands are not saved to the current **.sdc** file automatically. You must run the **Write SDC File** command to save your constraints.

The following commands are available on the Constraints menu in the Quartus II TimeQuest Timing Analyzer:

■ **Generate SDC File from QSF**

■ **Read SDC File**

■ **Write SDC File**

The **Generate SDC File from QSF** command runs a Tcl script that converts the Quartus II Classic Timing Analyzer constraints in a QSF file to an **.sdc** file for the Quartus II TimeQuest Timing Analyzer. The file *<current revision>***.sdc** is created by this command.

👣 For information about the **Generate SDC File from QSF** command, refer to the *Switching to the Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

The Generate SDC File from QSF command attempts to convert all timing constraints and exceptions in the QSF file to their equivalent **.sdc** file constraints. However, not all QSF file constraints are convertible to **.sdc** file constraints. Review the **.sdc** file after it is created to ensure that all constraints have been successfully converted.

The **Read SDC File** command reads the *<current revision>***.sdc** file.

When you select the **Write SDC File** command, an up-to-date **.sdc** file that reflects the current state of constraints in the Quartus II TimeQuest Timing Analyzer is generated.

Name Finder

Use the **Name Finder** dialog box to select the target for any constraints or exceptions in the Quartus II TimeQuest Timing Analyzer GUI. The **Name Finder** dialog box allows you to specify collections, filters, and filter options. The Collections field in the **Name Finder** dialog box allows you to specify the type of name to select. To select the type, in the **Collection** list, select the desired collection API from the following list:

- `get_cells`
- `get_clocks`
- `get_keepers`
- `get_nets`
- `get_nodes`
- `get_pins`
- `get_ports`
- `get_registers`

For more information about the various collection APIs, refer to "Collections" on page 7–22.

The Filter field allows you to filter names based on your own criteria, including wildcard characters. You can further refine your results using the following filter options:

- Case-insensitive
- Hierarchical
- Compatibility mode

For more information about the filter options, refer to "Wildcard Assignments and Collections" on page 7–75.

The Name Finder dialog box also provides an SDC command field that displays the currently selected name search command. You can copy the value from this field and use it for other constraint target fields. The **Name Finder** dialog box is shown in Figure 7–44.

Figure 7–44. Name Finder Dialog Box
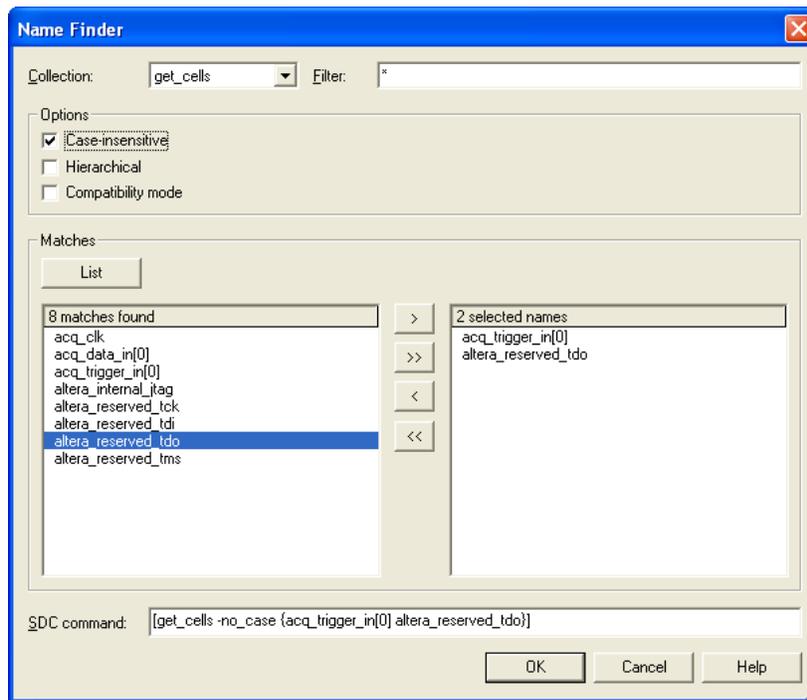
Target Pane

When using the TimeQuest GUI, you can split the View pane into multiple windows. The splitting feature allows you to display multiple reports in the View pane. After splitting the **View** pane, the last active window is updated with any new reports. You can change this behavior by changing the state of each split window. To change the window state, click the target circle in the upper right corner (Figure 7–45). Table 7–55 describes the state of each window.

Figure 7–45. Target Pane
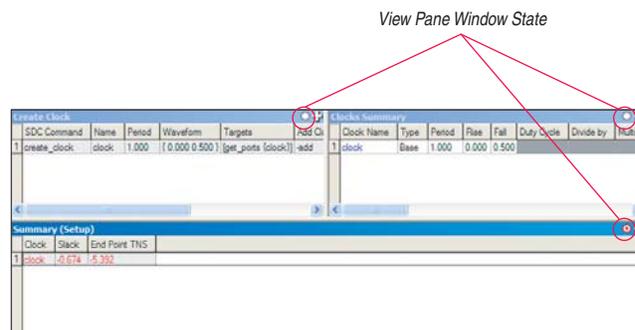
Table 7–55. View Pane Window State

State	Description
Partially Filled Red Circle	Indicates that the active window displays any new reports.
Fully Filled Red Circle	Indicates that the window, independent of it being the active window, displays any new reports.
Empty Circle	Indicates that the window does not display any new reports.

Clicking on the circle in the upper right corner of an active window changes the state of the window.

SDC Editor

The TimeQuest Timing Analyzer GUI also provides an SDC editor. The SDC editor provides an easy and convenient way to write, edit, and view **.sdc** files. The SDC editor is context sensitive. After an SDC constraint or exception has been entered, a tooltip appears that shows the options and format for the constraint or exception.

The SDC editor also provides helpful tools, including SDC templates and SDC templates for common design structures. To find these templates, when the SDC editor is active, look on the **Edit** menu.

☞ On the menu bar, the Constraints menu opens the **Constraints** dialog box. After you have finished entering all required parameters, the **.sdc** file is inserted at the current cursor position.

Conclusion

The Quartus II TimeQuest Timing Analyzer addresses the requirements of complex designs, resulting in increased productivity and efficiency through its intuitive user interface, support of industry-standard constraints format, and scripting capabilities. The Quartus II TimeQuest Timing Analyzer is a next-generation timing analysis tool that supports the industry-standard SDC format and allows designers to create, manage, and analyze complex timing constraints and to perform advanced timing verification.

Referenced Documents

This chapter references the following documents:

■ *AN 481: Applying Multicycle Exceptions in the TimeQuest Timing Analyzer*

■ *I/O Management* chapter in volume 2 of the *Quartus II Handbook*

■ *Quartus II TimeQuest Timing Analyzer Cookbook*

■ *SDC and TimeQuest API Reference Manual*

■ *Switching to the Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*

■ *TimeQuest Quick Start Tutorial*

■ *Volume 4: SOPC Builder* in the *Quartus II Handbook*

Document Revision History

Table 7–56 shows the revision history for this chapter.

Table 7–56. Document Revision History   (Part 1 of 2)

Date and Version	Changes Made	Summary of Changes
November 2008 v8.1.0	Updated for the Quartus II software version 8.1, including: ■ Added the following sections: 　■ "set_net_delay" on page 7–43 　■ "Annotated Delay" on page 7–49 　■ "report_net_delay" on page 7–69 ■ Updated the descriptions of the `-append` and `-file` *<name>* options in tables throughout the chapter ■ Updated entire chapter using 8½" × 11" chapter template ■ Minor editorial updates	Medium update for the Quartus II software version 8.1 release.
May 2008 v8.0.0	Updated for the Quartus II software version 8.0, including: ■ Changed the heading "Specify Design Timing Requirements" to "The Quartus II TimeQuest Timing Analyzer Flow Guidelines" on page 7–26 ■ In "SDC Constraint Files" on page 7–29, added information about order-sensitivity ■ Added a new section on "Metastability" on page 7–19 ■ Added a new section on "Common Clock Path Pessimism" on page 7–22 ■ Removed information about Asynchronous clocks from "Clock Groups" on page 7–43 ■ Updated information in Example 7–28 ■ Added three entries to Table 7–22 ■ Added information to Table 7–24 ■ Added information about the RSKM to "report_rskm" on page 7–80, including a formulaic equation (Equation 12) ■ Added the section "Clock Groups" on page 7–43 ■ Added Table 7–44 to "report_clock_fmax_summary" on page 7–86 ■ Added qualifier to introduction of Table 7–46 ■ Added Speed Grade information to Table 7–46 ■ Removed [-dtw] and added [-add] to information about the derive_clock_uncertainty command ("Derive Clock Uncertainty" on page 7–47) ■ Added the section "report_metastability" on page 7–68 ■ Added a new information about RSKM to "report_rskm" on page 7–80 ■ Added the section "Cross-Probing" on page 7–93 ■ Minor editorial updates ■ Added hyperlinks to referenced documents throughout chapter	Significant update for the Quartus II software version 8.0 release.

Table 7–56. Document Revision History   (Part 2 of 2)

Date and Version	Changes Made	Summary of Changes
October 2007 v7.2.0	Updated for the Quartus II software version 7.2, including: ■ Updated organization flow of the Compilation Flow with TimeQuest Guidelines, Timing Analysis Overview, and Specify Design Timing Requirements sections ■ Added new information on Clock as Data Analysis	Updated for the Quartus II software version 7.2.
May 2007 v7.1.0	Updated for the Quartus II software version 7.1, including: ■ Added support of `report_path` in "Timing Reports" on page 6–57 ■ Added `report_timing` information, especially on page 6-11 ■ Added new information under the following headings: ■ "Derive Clock Uncertainty" on page 6–40 ■ "report_rskm" on page 6–69 ■ "report_tccs" on page 6–69 ■ "report_path" on page 6–70 ■ Replaced the "Fast Timing Model Analysis" section with "Multi-Corner Analysis" on page 6–76 ■ Performed general 7.1 updates	Updated for the Quartus II software version 7.1.
March 2007 v7.0.0	Updated Quartus II software 7.0 revision and date only. No other changes made to chapter.	—
November 2006 v6.1.0	Updated for the Quartus II software version 6.1, including: ■ New "Getting Started" section, including descriptions of the Create Clock and Create Generated Clock dialog boxes/commands, sections on Specifying Clock Requirements, Specifying Input and Output Port Requirements, and Reporting ■ SDC Editor ■ Usability enhancements to the GUI ■ Updated SDC support ■ Numerous changes throughout chapter	Updated for the Quartus II software version 6.1.
July 2006 v6.0.1	Updated for the Quartus II software version 6.0.1: ■ Fixed typo in report_clock_transfers command on page 6-15.	—
May 2006 v6.0.0	Initial release.	—