

This chapter describes the benefits of switching to the Quartus® II TimeQuest Timing Analyzer, the differences between the TimeQuest analyzer and the Classic Timing Analyzer, and the process you should follow to switch a design from using the Classic Timing Analyzer to the TimeQuest analyzer.

Benefits of Switching to the TimeQuest Analyzer

Increasing design complexity requires a timing analysis tool with greater capabilities and flexibility. The TimeQuest analyzer offers the following benefits:

- Industry-standard Synopsys Design Constraint (SDC) support increases productivity.
- Simple, flexible reporting uses industry-standard terminology and makes timing sign-off faster.

These features ease constraint and analysis of modern, complex designs. SDC constraints support complex clocking schemes and high-speed interfaces. An example includes designs that have multiplexed clocks, regardless of whether they are switched on or off chip. Designs with source-synchronous interfaces, such as double data rate (DDR) memory interfaces, are much simpler to constrain and analyze with the TimeQuest analyzer.

There are three main differences between the Classic Timing Analyzer and the TimeQuest analyzer. Unlike the Classic Timing Analyzer, the TimeQuest analyzer has the following benefits:

- All clocks are related by default.
- The default hold multicycle value is zero.
- You must constrain all ports and ripple clocks.

This chapter contains the following sections:

- [“Switching Your Design” on page 8-2](#)
- [“Differences Between the Quartus II Timing Analyzers” on page 8-3](#)
- [“Timing Assignment Conversion” on page 8-24](#)
- [“Conversion Utility” on page 8-43](#)
- [“Notes” on page 8-53](#)

Switching Your Design

To switch a design from the Classic Timing Analyzer to the TimeQuest analyzer, perform the steps:

1. Open your compiled design in the Quartus II software.
2. Create a Synopsys Design Constraints File (.sdc) that contains timing constraints.
3. Perform timing analysis with the TimeQuest analyzer and examine the reports.

Open Your Compiled Design

To begin, open a design you previously compiled with the Quartus II software.

Create an SDC Constraints

Create SDC Constraints Manually

If you are familiar with SDC commands and syntax, you can create an .sdc with any text editor and skip to [“Start the TimeQuest Analyzer” on page 8-3](#). Name the .sdc `<revision>.sdc`, where `<revision>` is the current revision of your project, and save it in your project directory.



For more information about TimeQuest analyzer SDC constraints, refer to the [Quartus II TimeQuest Timing Analyzer](#) chapter in volume 3 of the *Quartus II Handbook*.

Create SDC Constraints from Existing Timing Assignments

You can use the TimeQuest analyzer conversion utility to help you convert the timing assignments in an existing Quartus II Settings File (.qsf) to corresponding SDC constraints. To run the TimeQuest analyzer conversion utility, on the Constraints menu, click **Generate SDC File from QSF**. You can also run the conversion utility by typing the following command at a system command prompt:

```
quartus_sta --qsf2sdc <project name> ↵
```

The .sdc created by the conversion utility is named `<revision>.sdc`.

After conversion, review the .sdc to ensure it is correct and complete, and make changes if necessary. Refer to [“Constraint File Priority” on page 8-7](#) for recommendations.

The conversion utility cannot convert some types of Classic Timing Analyzer assignments if no corresponding SDC constraint exist, or if there is more than one potentially equivalent SDC constraint. If the conversion utility cannot convert your assignment, manually convert any ambiguous assignments. Correct conversion requires knowledge of the intended function of your design. To convert your assignments, use the guidelines in [“Timing Assignment Conversion” on page 8-24](#).

Start the TimeQuest Analyzer

To open the TimeQuest analyzer GUI, on the Tools menu, click **TimeQuest Timing Analyzer**. The TimeQuest GUI automatically opens the project open in the Quartus II software GUI.

To open the TimeQuest analyzer GUI from a system command prompt, type the following command:

```
quartus_staw ␣
```

To start the TimeQuest analyzer Tcl shell type the following command:

```
quartus_sta -s ␣
```

Differences Between the Quartus II Timing Analyzers

The differences between the TimeQuest analyzer and the Classic Timing Analyzer are described in the following sections:

- “Terminology” on page 8-3
- “Constraints” on page 8-5
- “Clocks” on page 8-10
- “Hold Multicycle” on page 8-18
- “Fitter Performance and Behavior” on page 8-19
- “Reporting” on page 8-20
- “Timing Assignment Conversion” on page 8-24

Terminology

This section introduces the industry-standard SDC terminology used by the TimeQuest analyzer.

Netlists

The TimeQuest analyzer uses SDC naming conventions for netlists. Netlists consist of cells, pins, nets, ports, and clocks.

- Cells are instances of fundamental hardware elements in Altera® FPGAs, such as logic elements, look-up tables (LUTs), and registers.
- Pins are inputs and outputs of cells.
- Nets are connections between output pins and input pins.
- Ports are top-level module inputs and outputs (device inputs and outputs).
- Clocks are abstract objects outside the netlist.



The terminology of pins and ports is opposite to that of the Classic Timing Analyzer. In the Classic Timing Analyzer, ports are inputs and outputs of cells, and pins are top-level module inputs and outputs (device inputs and outputs).

Figure 8-1 shows a sample design, and Figure 8-2 shows the TimeQuest analyzer netlist representation of the design. Netlist elements in Figure 8-2 are labeled to illustrate the SDC terminology.

Figure 8-1. Sample Design

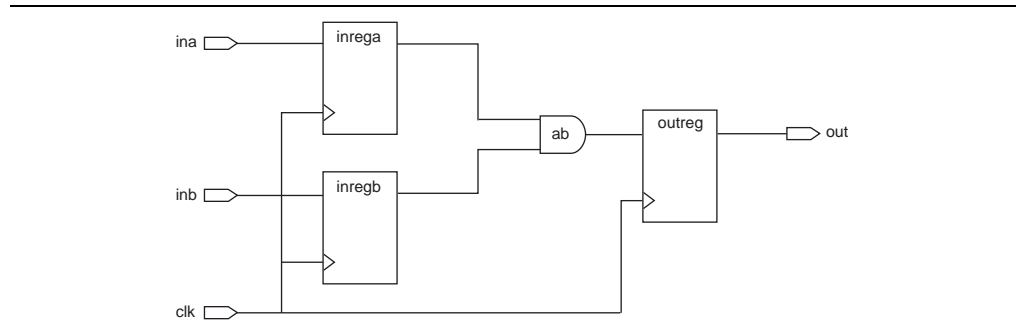
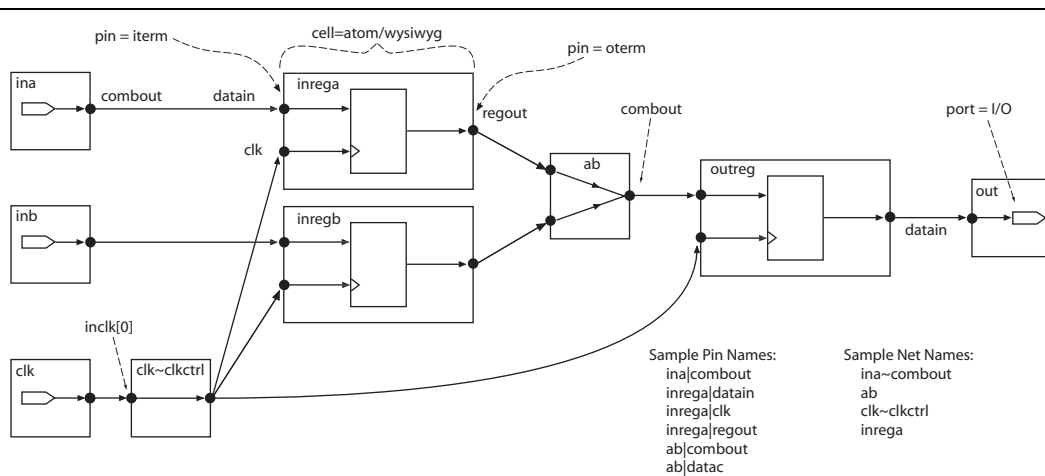


Figure 8-2. TimeQuest Analyzer Netlist



Collections

In addition to standard SDC collections, the TimeQuest analyzer supports the following Altera-specific collection types:

- **Keepers**—Noncombinational nodes in a netlist
- **Nodes**—Nodes can be combinational, registers, latches, or ports (device inputs and outputs)
- **Registers**—Registers or latches in the netlist

You can use the `get_keepers`, `get_nodes`, or `get_registers` commands to access these collections.



For more information about TimeQuest analyzer terminology, refer to the *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Constraints

The Classic Timing Analyzer and TimeQuest analyzer store constraints in different files, support different methods for constraint entry, and prioritize constraints differently. The following sections explain these differences.

Constraint Files

The TimeQuest analyzer stores all SDC timing constraints in **.sdc** files. The Classic Timing Analyzer stores all timing assignments in the **.qsf** for your project.

When you use the TimeQuest analyzer, your **.qsf** contains all assignments and settings except for SDC constraints. The TimeQuest analyzer ignores the timing assignments in your **.qsf** except when the conversion utility converts Classic Timing Analyzer timing assignments to TimeQuest analyzer SDC constraints. There is no automatic process that keeps timing constraints synchronized between your **.qsf** and **.sdc** files. You must manually keep the constraints synchronized, if so desired.

Constraint Entry

With the Classic Timing Analyzer, you enter timing assignments with the **Settings** dialog box, the Assignment Editor, or with commands in Tcl scripts. With the TimeQuest analyzer, you cannot use the Assignment Editor to enter SDC constraints. You must use one of the following methods to enter TimeQuest analyzer constraints:

- Enter constraints at the Tcl prompt in the TimeQuest analyzer
- Enter constraints in an **.sdc** with a text editor or SDC editor
- Use the constraint entry commands on the Constraints menu in the TimeQuest analyzer GUI

You can enter timing assignments for the Classic Timing Analyzer even if no timing netlist exists for your design. The TimeQuest analyzer requires that a netlist exist for interactive constraint entry. Each TimeQuest analyzer constraint is a Tcl command evaluated in real-time, if entered directly in the Tcl console. As part of this evaluation, the TimeQuest analyzer validates all names. To do this, SDC commands can only be evaluated after a netlist is created. An **.sdc** can be created at any time using the TimeQuest analyzer or any other text editor, but a netlist is required before an **.sdc** can be sourced. You must create a timing netlist in the TimeQuest analyzer before you can enter constraints with either of the following interactive methods:

- At the Tcl console of the TimeQuest analyzer
- With commands on the Constraints menu in the TimeQuest analyzer GUI

If you enter constraints with a text editor separate from the TimeQuest analyzer, no timing netlist is required.

To create a timing netlist in the TimeQuest analyzer, use the `create_timing_netlist` command, or double-click **Create Timing Netlist** in the Tasks pane of the TimeQuest analyzer GUI.

Time Units

Enter time values in default time units of nanoseconds (ns) with up to three decimal places. Note that the TimeQuest analyzer does not display the default time unit when it displays time values.

You can specify a different default time unit with the `set_time_format -unit <default time unit>` command, or specify another unit when you enter a time value, for example, `300ps`.



Specifying time units after a value is not part of the standard SDC format. Unit specification is a TimeQuest analyzer extension.

You can specify clock constraints with period or frequency in the TimeQuest analyzer. For example, you can use any of the following constraints:

- `create_clock -period 10.000`
(assuming default units and decimal places)
- `create_clock -period "100 MHz"`
- `create_clock -period "10 ns"`

MegaCore Functions

If you change any MegaCore function settings and regenerate the core after you convert your timing assignments to SDC constraints, you must manually update the SDC constraints or reconvert your assignments. You must update or reconvert, because changes to MegaCore function settings can affect timing assignments embedded in the hardware description language files of the function. The timing assignments are not converted automatically when the core settings change.



You should make a backup copy of your `.sdc` before reconverting assignments. If you make changes to the `.sdc`, you can manually copy the updated MegaCore timing constraints to your `.sdc`.

Bus Name Format

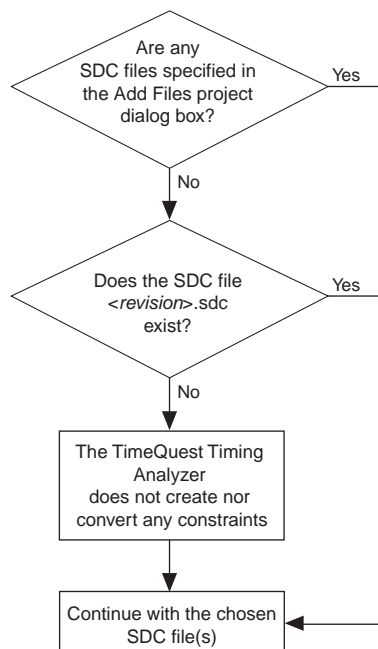
In the Classic Timing Analyzer, you can make a timing assignment to all bits in a bus with the bus name (or the bus name followed by an asterisk enclosed in square brackets) as the target. For example, to make an assignment to all bits of a bus called `address`, use `address` or `address[*]` as the target of the assignment.

In the TimeQuest analyzer, you must use the bus name followed by square brackets enclosing an asterisk, as follows: `address[*]` to make all timing assignments to all bits within a bus.

Constraint File Priority

Figure 8-3 shows the priority order in which the TimeQuest analyzer searches for .sdc files.

Figure 8-3. .sdc Search Order



If you specify constraints in multiple .sdc files, or if you use a single .sdc with a name other than <revision>.sdc, you must add the files to your project so the TimeQuest analyzer can find them.

- For more information, refer to *Managing Files in a Project* in Quartus II Help.

You can also add .sdc files to your project with the following Tcl command in your .qsf, repeated once for each .sdc:

```
set_global_assignment -name SDC_FILE <.sdc file name>
```

The TimeQuest analyzer reads constraint files from the files list in the order they are listed.

If you use an .sdc created by the conversion utility, you should place it first in the list of files. When conflicting constraints apply to the same node, the last constraint has the highest priority. Therefore, .sdc files with your additions or changes should be listed after the .sdc created by the conversion utility, so your constraints have higher priority.

When you process your design, and the TimeQuest analyzer cannot find an `.sdc` it attempts to meet a default 1 GHz constraint on all clocks in your design. The Quartus II software may prompt you to run the conversion utility if your design does not reference any `.sdc` files.

You must review the `.sdc` as you would when manually running the conversion utility. Refer to “[Reviewing Conversion Results](#)” on page 8–50 for information about reviewing the converted constraints.

Constraint Priority

The Classic Timing Analyzer prioritizes assignments based on the specificity of the nodes to which they are assigned. The more specific an assignment is, the higher its priority. The TimeQuest analyzer simplifies these precedence rules. When overlaps occur in the nodes to which the constraints apply, constraints at the bottom of the file take priority over constraints at the top of the file.

As an example, in the Classic Timing Analyzer, point-to-point multicycle assignments have higher priority than single-point multicycle assignments. The two assignments in [Example 8–1](#) result in a multicycle assignment of two between `A_reg` and all nodes beginning with `B`, including `B_reg`. The single-point assignment does not apply to paths from `A_reg` to `B_reg`, because the specific point-to-point assignment takes priority over the general single-point assignment.

Example 8–1. Classic Timing Analyzer Multicycle Assignments

```
set_instance_assignment -name MULTICYCLE -from A_reg -to B* 2
set_instance_assignment -name MULTICYCLE -to B_reg 3
```

[Example 8–2](#) shows SDC versions of the preceding Classic Timing Analyzer timing assignments. However, the TimeQuest analyzer evaluates the constraints from top to bottom, regardless of whether the assignments are point-to-point assignments or single-point assignments, so the path from `A_reg` to `B_reg` receives a multicycle exception of three because it is ordered second.

Example 8–2. TimeQuest Analyzer Multicycle Exceptions

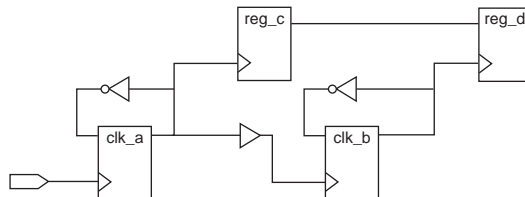
```
set_multicycle_path -from [get_keepers A_reg] -to [get_keepers B*] 2
set_multicycle_path -to [get_keepers B_reg] 3
```

Ambiguous Constraints

Because of capabilities in the TimeQuest analyzer, some Classic Timing Analyzer assignments can be ambiguous after conversion by the conversion utility. These assignments require manual updating based on your knowledge of your design.

Figure 8-4 shows a ripple clock circuit. The explanation that follows shows an ambiguous constraint for that circuit, and how to edit the constraint to remove the ambiguity in the TimeQuest analyzer.

Figure 8-4. Ripple Clock Circuit



In the Classic Timing Analyzer, the following QSF multicycle assignment from `clk_a` to `clk_b` with a value of two applies to paths transferring data from the `clk_a` domain to the `clk_b` domain:

```
set_instance_assignment -name MULTICYCLE -from clk_a -to clk_b 2
```

In Figure 8-4, this assignment applies to the path from `reg_c` to `reg_d`. In the TimeQuest analyzer, the use of the clock node names in the following equivalent multicycle exception is ambiguous:

```
set_multicycle_path -setup -from clk_a -to clk_b 2
```

The exception could apply to the path between `clk_a` and `clk_b`, or it could apply to paths from one ripple clock domain to the other ripple clock domain (`reg_c` to `reg_d`).

The TimeQuest analyzer exceptions shown in Example 8-3 are not ambiguous because they use collections to explicitly specify the targets of the exception.

Example 8-3. Unambiguous TimeQuest Analyzer Exceptions

```
# Applies to path from reg_c to reg_d
set_multicycle_path -setup -from [get_clocks clk_a] \
  -to [get_clocks clk_b] 2
# Applies to path from clk_a to clk_b
set_multicycle_path -setup -from [get_registers clk_a] \
  -to [get_registers clk_b] 2
```

Clocks

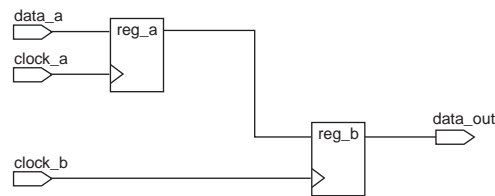
The Classic Timing Analyzer and TimeQuest analyzer detect, analyze, and report clocks differently. The following sections describe these differences.

Related and Unrelated Clocks

In the TimeQuest analyzer, all clocks are related by default, and you must add assignments to indicate unrelated clocks. However, in the Classic Timing Analyzer, all base clocks are unrelated by default. All derived clocks of a base clock are related to each other, but are unrelated to other base clocks and their derived clocks.

Figure 8-5 shows a circuit with a path between two clock domains. The TimeQuest analyzer analyzes the path from `reg_a` to `reg_b` because all clocks are related by default. The Classic Timing Analyzer does not analyze the path from `reg_a` to `reg_b` by default.

Figure 8-5. Cross Clock Domain Path



To make clocks unrelated in the TimeQuest analyzer, use the `set_clock_groups` command with the `-exclusive` option. The TimeQuest analyzer does not analyze paths between the two clock domains. For example, the following command renders `clock_a` and `clock_b` unrelated:

```
set_clock_groups -exclusive -group {clock_a} -group {clock_b}
```

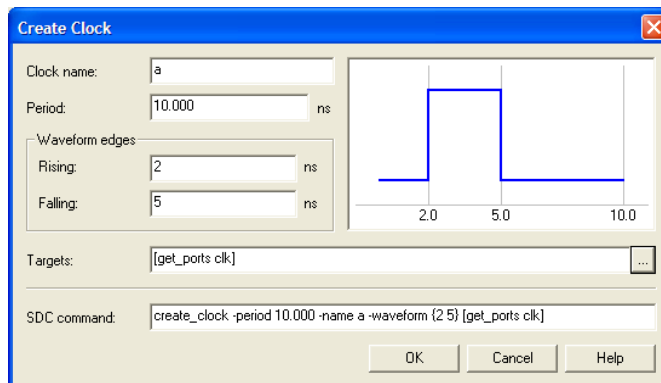
Clock Offset

In the TimeQuest analyzer, clocks can have nonzero values for the rising edge of the waveform, a feature that the Classic Timing Analyzer does not support. To specify an offset for your clock, use the `create_clock` command with the `-waveform` option to specify the rising and falling edge times, as shown in this example:

```
-waveform {<rising edge time> <falling edge time>}
```

Figure 8-6 shows a clock constraint with an offset in the TimeQuest analyzer GUI.

Figure 8-6. Create Clock Dialog Box



Clock offset affects setup and hold relationships. Launch and latch edges are evaluated after offsets are applied. Depending on the offset, the setup relationship can be the offset value, or the difference between the period and offset. You should use the clock latency constraint, instead of clock offset to emulate latency. Refer to “[Offset and Latency Example](#)” on page 8-11 for an example that illustrates the different effects of offset and latency.

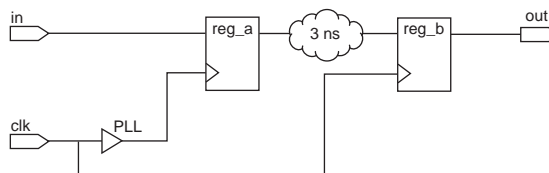
Clock Latency

The TimeQuest analyzer does not ignore early clock latency and late clock latency differences when the clock source is the same, as the Classic Timing Analyzer does. When you specify latencies, you should take common clock path pessimism into account and use uncertainty to control pessimism differences for clock-to-clock data transfers. Unlike clock offset, clock latency affects skew, and launch and latch edges are evaluated before latencies are applied, so the setup relationship is always equal to the period.

Offset and Latency Example

Figure 8-7 shows a simple register-to-register circuit used to illustrate the different effects of offset and latency. The examples show why you should not use clock offset to emulate clock latency.

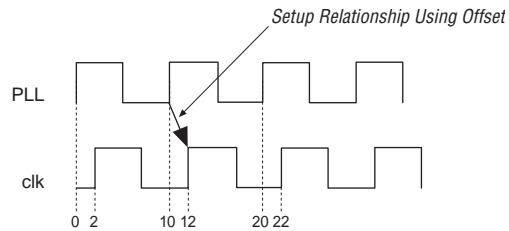
Figure 8-7. Offset and Latency Example Circuit



The period for `clk` is 10 ns, and the period for the phase-locked loop (PLL) output is 10 ns. The PLL compensation value is -2 ns. The network delay from the PLL to `reg_a` equals the network delay from `clk` to `reg_b`. Finally, the delay from `reg_a` to `reg_b` is 3 ns.

Clock Offset Scenario

Treat the PLL compensation value of -2 ns as a clock offset of -2 ns with a clock skew of 0 ns. Launch and latch edges are evaluated after offsets are applied, so the setup relationship is 2 ns (Figure 8-8).

Figure 8-8. Setup Relationship Using Offset

Equation 8-1 shows how to calculate the slack value for the path from `reg_a` to `reg_b`.

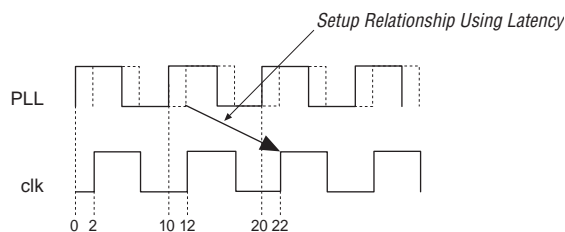
Equation 8-1.

$$\begin{aligned} \text{slack} &= \text{clock arrival} - \text{data arrival} \\ \text{slack} &= \text{setup relationship} + \text{clock skew} - \text{reg_to_reg delay} \\ \text{slack} &= 2\text{ns} + 0\text{ns} - 3\text{ns} \\ \text{slack} &= -1\text{ns} \end{aligned}$$

The negative slack requires a multicycle assignment with a value of two and a hold multicycle assignment with a value of one to correct. With these assignments from `reg_a` to `reg_b`, the setup relationship is then 12 ns, resulting in a slack of 9 ns.

Clock Latency Scenario

Treat the PLL compensation value of -2 ns as latency with a clock skew of 2 ns. Because launch and latch edges are evaluated before latencies are applied, the setup relationship is 10 ns (the period of `clk` and the PLL) (Figure 8-9).

Figure 8-9. Setup Relationship Using Latency

Equation 8-2 shows how to calculate the slack value for the path from reg_a to reg_b.

Equation 8-2.

$$\begin{aligned} \text{slack} &= \text{clock arrival} - \text{data arrival} \\ \text{slack} &= \text{setup relationship} + \text{clock skew} - \text{reg_to_reg delay} \\ \text{slack} &= 10\text{ns} + 2\text{ns} - 3\text{ns} \\ \text{slack} &= 9\text{ns} \end{aligned}$$

The slack of 9 ns is identical to the slack computed in the previous example, but because this example uses latency instead of offset, no multicycle assignment is required.

Clock Uncertainty

The Classic Timing Analyzer ignores **Clock Setup Uncertainty** and **Clock Hold Uncertainty** assignments when you specify a setup or hold relationship between two clocks. However, the TimeQuest analyzer does not ignore clock uncertainty when you specify a setup or hold relationship between two clocks. Figure 8-10 and Figure 8-11 illustrate the different behavior between the TimeQuest analyzer and the Classic Timing Analyzer.

In both figures, the constraints are identical. There is a 20-ns period for clk_a and clk_b. There is a setup relationship (a set_max_delay exception in the TimeQuest analyzer) of 7 ns from clk_a to clk_b, and a clock setup uncertainty constraint of 1 ns from clk_a to clk_b. The actual setup relationship in the TimeQuest analyzer is 1 ns less than in the Classic Timing Analyzer because of the way they analyze clock uncertainty.

Figure 8-10. Classic Timing Analyzer Behavior

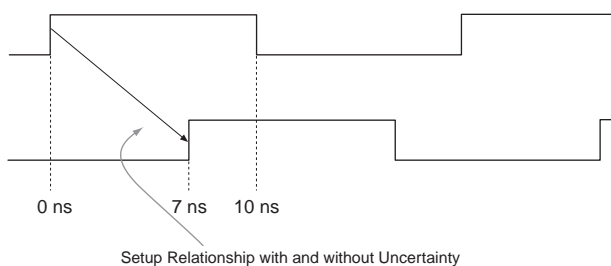
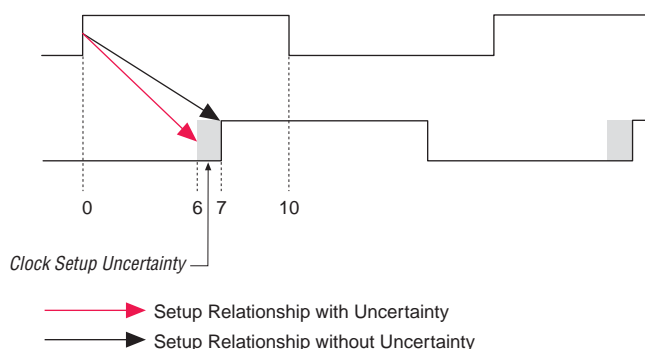


Figure 8-11. TimeQuest Analyzer Behavior



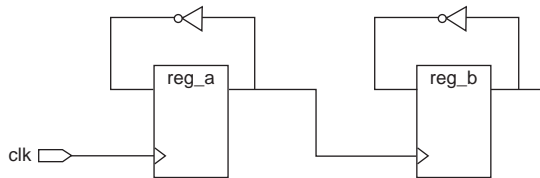
Derived and Generated Clocks

Generated clocks in the TimeQuest analyzer are different than derived clocks in the Classic Timing Analyzer. In the Classic Timing Analyzer, the source of a derived clock must be a base clock. However, in the TimeQuest analyzer, the source of a generated clock can be any other clock in the design (including virtual clocks), or any node to which a clock propagates through the clock network. Because generated clocks are related through the clock network, you can specify generated clocks for isolated modules, such as IP, without knowing the details of the clocks outside of the module.

In the TimeQuest analyzer, you can specify generated clocks relative to specific edges and edge shifts of a master clock, a feature that the Classic Timing Analyzer does not support.

Figure 8-12 shows a simple ripple clock that you should constrain with generated clocks in the TimeQuest analyzer.

Figure 8-12. Generated Clocks Circuit



The TimeQuest analyzer constraints shown in Example 8-4 constrain the clocks in the circuit above. Note that the source of each generated clock can be the input pin of the register itself, not the name of another clock.

Example 8-4. Generated Clock Constraints

```
create_clock -period 10 -name clk clk
create_generated_clock -divide_by 2 -source reg_a|CLK -name reg_a reg_a
create_generated_clock -divide_by 2 -source reg_b|CLK -name reg_b reg_b
```

Automatic Clock Detection

The Classic Timing Analyzer and TimeQuest analyzer identify clock sources of registers that do not have a defined clock source. The Classic Timing Analyzer traces back along the clock network, through registers and logic, until it reaches a top-level port in your design. The TimeQuest analyzer also traces back along the clock network, but it stops at registers.

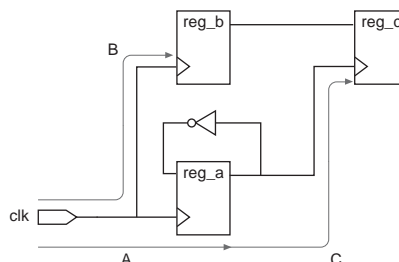
You can use two SDC commands in the TimeQuest analyzer to automatically detect and create clocks for unconstrained clock sources:

- **derive_clocks**—creates clocks on sources of clock pins that do not already have at least one clock sourcing the clock pin
- **derive_pll_clocks**—identifies PLLs and creates generated clocks on the clock output pins

derive_clocks Command

Figure 8-13 shows a circuit with a divide-by-two register and indicates the clock network delays as A, B, and C. The following explanation describes how the Classic Timing Analyzer and TimeQuest analyzer detect the clocks in Figure 8-13.

Figure 8-13. Example Circuit for derive_clocks



The Classic Timing Analyzer detects that `clk` is the clock source for registers `reg_a`, `reg_b`, and `reg_c`. It detects that `clk` is the clock source for `reg_c` because it traces back along the clock network for `reg_c` through `reg_a`, until it finds the `clk` port. The Classic Timing Analyzer computes the clock arrival time for `reg_c` as $A + C$.

The `derive_clocks` command in the TimeQuest analyzer creates two base clocks, one on the `clk` port and one on `reg_a`, because the command does not trace through registers on the clock network. The clock arrival time for `reg_c` is C because the clock starts at `reg_a`.

To make the TimeQuest analyzer compute the same clock arrival time ($A + C$) as the Classic Timing Analyzer for `reg_c`, make the following modifications to the clock constraints created by the `derive_clocks` command:

1. Change the base clock named `reg_a` to a generated clock
2. Make the source the clock pin of `reg_a` (`reg_a | clk`) or the port `clk`
3. Add a `-divide_by 2` option

These modifications cause the clock arrival times to `reg_c` to match between the Classic Timing Analyzer and the TimeQuest analyzer. However, the clock for `reg_c` is shown as `reg_a` instead of `clk`, and the launch and latch edges may change for some paths due to the `-divide_by` option.

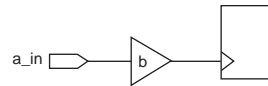
You can use the `derive_clocks` command at the beginning of your design cycle when you do not know all of the clock constraints for your design, but you should not use it during timing sign-off. Instead, you should constrain each clock in your design with the `create_clock` or `create_generated_clocks` commands.

The `derive_clocks` command detects clocks in your design using the following rules:

1. An input clock port is detected as a clock only if there are no other clocks feeding the destination registers.
 - a. Input clock ports are not detected automatically if they feed only other base clocks.
 - b. If other clocks feed the port's register destinations, the port is assumed to be an enable or data port for a gated clock.
 - c. When no clocks are defined, and multiple clocks feed a destination register, the auto-detected clock is selected arbitrarily.
2. All ripple clocks (registers in a clock path) are detected as clocks automatically using the same rules for input clock ports. If both an input port and a register feed register clock pins, the input port is selected as the clock.

The following examples show how the `derive_clocks` command detects clocks in the circuit shown in [Figure 8-14](#).

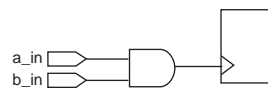
Figure 8-14. Detectable Clock



- If you do not make any clock settings, and then you use the `derive_clocks` command, it detects `a_in` as a clock according to rule 1, because there are no other clocks feeding the register.
- If you create a clock with `b` as its target, and then you run the `derive_clocks` command, it does not detect `a_in` as a clock according to rule 1a, because `a_in` feeds only another clock.

The following examples show how the `derive_clocks` command detects clocks in the circuit shown in [Figure 8-15](#).

Figure 8-15. Two Detectable Clocks



- If you do not make any clock settings and then you use the `derive_clocks` command, it selects a clock arbitrarily, according to rule 1c.
- If you create a clock with `a_in` as its target and then you use the `derive_clocks` command, it does not detect `b_in` as a clock according to rule 1b, because another clock (`a_in`) feeds the register.

derive_pll_clocks Command

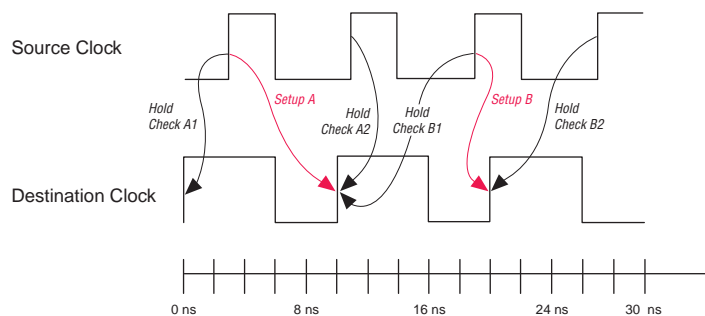
The `derive_pll_clocks` command names the generated clocks according to the names of the PLL output pins by default, and you cannot change these generated clock names. If you want to use your own clock names, you must use the `create_generated_clock` command for each PLL output clock and specify the names with the `-name` option.

If you use the PLL clock-switchover feature, the `derive_pll_clocks` command creates additional generated clocks on each output clock pin of the PLL based on the secondary clock input to the PLL. This may require the `set_clock_groups` or `set_false_path` commands to cut the primary and secondary clock outputs. For information about how to make clocks unrelated, refer to “[Related and Unrelated Clocks](#)” on page 8-10.

Hold Relationship

The TimeQuest analyzer and Classic Timing Analyzer choose the worst-case hold relationship differently. Refer to [Figure 8-16](#) for sample waveforms to illustrate the different effects.

Figure 8-16. Worst-Case Hold



The Classic Timing Analyzer first identifies the worst-case setup relationship. The worst-case setup relationship is Setup B. Then the Classic Timing Analyzer chooses the worst-case hold relationship (Hold Check B1 or Hold Check B2) for that specific setup relationship, Setup B. The Classic Timing Analyzer chooses Hold Check B2 for the worst-case hold relationship.

However, the TimeQuest analyzer calculates worst-case hold relationships for all possible setup relationships and chooses the absolute worst-case hold relationship. The TimeQuest analyzer checks two hold relationships for every setup relationship:

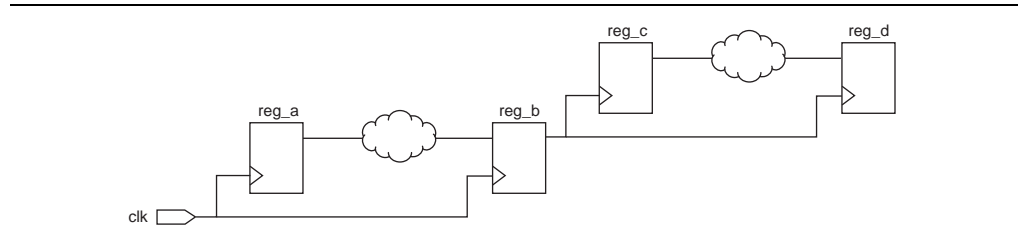
- Data launched by the current launch edge not captured by the previous latch edge (Hold Check A1 and Hold Check B1)
- Data launched by the next launch edge not captured by the current latch edge (Hold Check A2 and Hold Check B2)

The TimeQuest analyzer chooses Hold Check A2 as the absolute worst-case hold relationship.

Clock Objects

The Classic Timing Analyzer treats nodes with clock settings assigned to them as special objects in the timing netlist. Any node in the timing netlist with a clock setting is treated as a clock object, regardless of its actual type, such as a register. When a register has a clock setting assigned to it, the Classic Timing Analyzer does not analyze register-to-register paths beginning or ending at that register. Figure 8-17 shows a circuit that illustrates this situation.

Figure 8-17. Clock Objects



With no clock assignments on any of the registers, the Classic Timing Analyzer analyzes timing on the path from reg_a to reg_b, and from reg_c to reg_d. If you make a clock setting assignment to reg_b, reg_b is no longer considered a register node in the netlist, and the path from reg_a to reg_b is no longer analyzed.

In the TimeQuest analyzer, clocks are abstract objects that are associated with nodes in the timing netlist. The TimeQuest analyzer analyzes the path from reg_a to reg_b even if there is a clock assigned to reg_b.

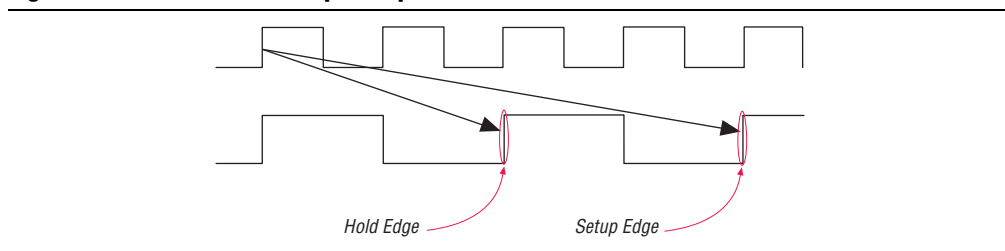
Hold Multicycle

The hold multicycle value numbering scheme is different in the Classic Timing Analyzer and TimeQuest analyzer. Also, you can choose between two values for the default hold multicycle value in the Classic Timing Analyzer but you cannot change the default value in the TimeQuest analyzer. The hold multicycle value specifies which clock edge is used for hold analysis when you change the latch edge with a multicycle assignment.

In the Classic Timing Analyzer, the hold multicycle value is based on one, and is the number of clock cycles away from the setup edge. In the TimeQuest analyzer, the hold multicycle value is based on zero, and is the number of clock cycles away from the default hold edge. In the TimeQuest analyzer, the default hold edge is one edge before or after the setup edges. Subtract one from any hold multicycle value in the Classic Timing Analyzer to compute the equivalent value for the TimeQuest analyzer.

Figure 8-18 shows simple waveforms for a cross-clock domain transfer with the indicated setup and hold edges.

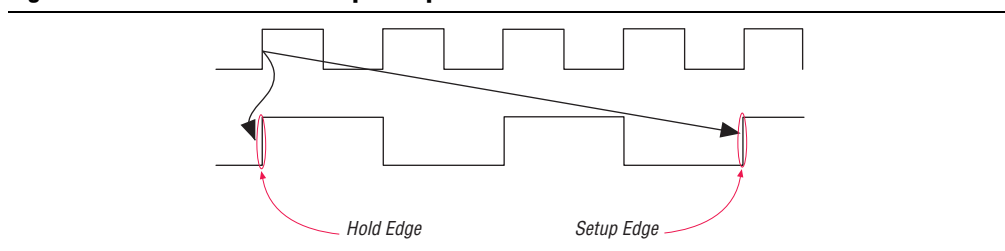
Figure 8-18. First Relationship Example



In the TimeQuest analyzer, only a multicycle exception of two is required to constrain the design for the indicated setup and hold relationships.

Figure 8-19 shows simple waveforms for a different cross-clock domain transfer with indicated setup and hold edges. The following explanation shows what exceptions to apply to achieve the desired setup and hold relationships.

Figure 8-19. Second Relationship Example



In the TimeQuest analyzer, you must use the following two exceptions:

- A multicycle exception of two
- A hold multicycle exception of one, because the hold edge is one edge behind the default hold edge, which is one edge after the setup edge.

You should always add a hold multicycle assignment for every multicycle assignment to ensure the correct exceptions are applied regardless of the timing analyzer you use.

Fitter Performance and Behavior

When you analyze a design with the TimeQuest analyzer, the Fitter memory use and compilation time may increase as compared to the Classic Timing Analyzer, however the timing analysis time may decrease.

The behavior for one value of the **Optimize hold time** Fitter assignment differs between the TimeQuest analyzer and the Classic Timing Analyzer. In the TimeQuest analyzer, the **I/O Paths and Minimum TPD Paths** setting and the **All Paths** setting are equivalent, whereas in the Classic Timing Analyzer the settings directed the Fitter to optimize hold times differently.

Reporting

The TimeQuest analyzer provides a more flexible and powerful interface for reporting timing analysis results than the Classic Timing Analyzer. Although the interface and constraints are more flexible and powerful, both analyzers use the same device timing models, except for device families that support rise/fall analysis. The Classic Timing Analyzer does not support rise/fall analysis, but the TimeQuest analyzer does. Therefore, you may see slightly different delays on identical paths in device families that support rise/fall analysis if you analyze timing in both analyzers.

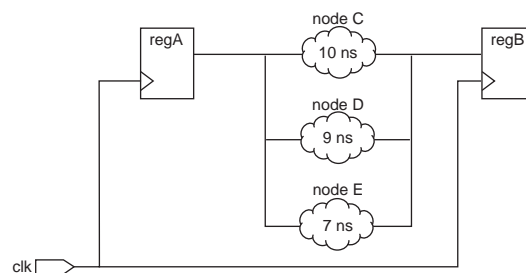
Both analyzers report identical delays along identically constrained paths in your design. The TimeQuest analyzer allows you to constrain some paths that you could not constrain with the Classic Timing Analyzer. Differently constrained paths result in different reported values, but for identical paths in your design that are constrained identically, the delays are exactly the same. Both timing analyzers use the same timing models.

Paths and Pairs

In reporting, the most significant difference between the two analyzers is that the TimeQuest analyzer reports paths, while the Classic Timing Analyzer reports pairs. Path reporting means that the analyzer separately reports every path between two registers. Pair reporting means that the analyzer reports only the worst-case path between two registers. One benefit of path reporting over pair reporting is that you can more easily identify common points in failing paths that may be good targets for optimization.

If your design does not meet timing constraints, this reporting difference can give the impression that there are many more timing failures when you use the TimeQuest analyzer. Figure 8-20 shows a sample circuit followed by a description of the differences between path and pair reporting.

Figure 8-20. Failing Paths



There is an 8-ns period constraint on the clk pin, resulting in two paths that fail timing: $\text{regA} \rightarrow \text{C} \rightarrow \text{regB}$ and $\text{regA} \rightarrow \text{D} \rightarrow \text{regB}$. The Classic Timing Analyzer reports only worst-case path $\text{regA} \rightarrow \text{C} \rightarrow \text{regB}$. The TimeQuest analyzer reports both failing paths $\text{regA} \rightarrow \text{C} \rightarrow \text{regB}$ and $\text{regA} \rightarrow \text{D} \rightarrow \text{regB}$. It also reports path $\text{regA} \rightarrow \text{E} \rightarrow \text{regB}$ with positive slack.

Default Reports

The TimeQuest analyzer generates only a small number of reports by default, as compared to the Classic Timing Analyzer, which generates every available report by default. With the TimeQuest analyzer, you generate reports on demand.

Netlist Names

The Classic Timing Analyzer uses register names in reporting, but the TimeQuest analyzer uses register pin names (with the exception of port names of the top-level module). Buried nodes or register names are used when necessary.

Example 8-5 shows how register names are used in Classic Timing Analyzer reports.

Example 8-5. Netlist Names in the Classic Timing Analyzer

```
Info: + Shortest register to register delay is 0.538 ns
      Info: 1: + IC(0.000 ns) + CELL(0.000 ns) = 0.000 ns; Loc. =
LCFF_X1_Y5_N1;
Fanout = 1; REG Node = 'inst'
      Info: 2: + IC(0.305 ns) + CELL(0.149 ns) = 0.454 ns; Loc. =
LCCOMB_X1_Y5_N20; Fanout = 1; COMB Node = 'inst3~feeder'
      Info: 3: + IC(0.000 ns) + CELL(0.084 ns) = 0.538 ns; Loc. =
LCFF_X1_Y5_N21; Fanout = 1; REG Node = 'inst3'
      Info: Total cell delay = 0.233 ns ( 43.31 % )
      Info: Total interconnect delay = 0.305 ns ( 56.69 % )
```

Example 8-6 shows the same information as presented in a TimeQuest analyzer report. In this example, register pin names are used in place of register names.

Example 8-6. Netlist Names in the TimeQuest Analyzer

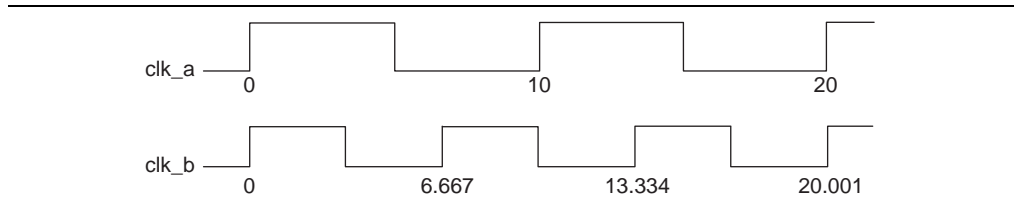
```
Info:      3.788      0.250      uTco      inst
Info:      3.788      0.000 RR      CELL      inst|regout
Info:      4.093      0.305 RR      IC       inst3~feeder|datad
Info:      4.242      0.149 RR      CELL      inst3~feeder|combout
Info:      4.242      0.000 RR      IC       inst3|datain
Info:      4.326      0.084 RR      CELL      inst3
```

Non-Integer Clock Periods

In some cases when related clock periods are not integer multiples of each other, a lack of precision in clock period definition in the TimeQuest analyzer can result in reported setup or hold relationships of a few picoseconds. In addition, launch and latch times for the relationships can be very large. If you experience this, use the `set_max_delay` and `set_min_delay` commands to specify the correct relationships. The Classic Timing Analyzer can maintain additional information about clock frequency that mitigates the lack of precision in clock period definition.

When the clock period cannot be expressed as an integer in terms of picoseconds, you have the situation shown in [Figure 8-21](#). This figure shows two clocks: `clk_a` has a 10 ns period, and `clk_b` has a 6.667 ns period.

Figure 8-21. Very Small Setup Relationship



There is a 1 ps setup relationship at 20 ns because you cannot specify the 6.667 ns period beyond picosecond precision. You should apply the maximum and minimum delay exceptions shown in [Example 8-7](#) between the two clocks to specify the correct relationships.

Example 8-7. Minimum and Maximum Delay Exceptions

```
set_max_delay -from [get_clocks clk_a] -to [get_clocks clk_b] 3.333
set_min_delay -from [get_clocks clk_a] -to [get_clocks clk_b] 0
```

Other Features

The TimeQuest analyzer reports time values without units. By default, the units are nanoseconds, and three decimal places are displayed. You can change the default time unit and decimal places with the `set_time_format` command.

When you use the TimeQuest analyzer in a Tcl shell, output is ASCII-formatted, and columns are aligned for easy reading on 80-column consoles. [Example 8-8](#) shows sample output from the TimeQuest analyzer `report_timing` command.

Example 8-8. ASCII-Formatted TimeQuest Analyzer Report

```
tcl> report_timing -from inst -to inst5
Info: Report Timing: Found 1 setup paths (0 violated). Worst case slack
is 3.634
Info: -from [get_keepers inst]
Info: -to [get_keepers inst5]
Info: Path #1: Slack is 3.634
Info:
=====
Info: From Node      : inst
Info: To Node        : inst5
Info: Launch Clock   : clk_a
Info: Latch Clock    : clk_b
Info:
Info: Data Arrival Path:
Info:
Info: Total (ns)  Incr (ns)      Type  Node
Info: =====  =====  ==  ====
=====
Info:          0.000      0.000           launch edge time
Info:          2.347      2.347  R           clock network delay
Info:          2.597      0.250   uTco   inst
Info:          2.597      0.000  RR   CELL  inst|regout
Info:          3.088      0.491  RR    IC   inst6|datac
Info:          3.359      0.271  RR   CELL  inst6|combout
Info:          3.359      0.000  RR    IC   inst5|datain
Info:          3.443      0.084  RR   CELL  inst5
Info:
Info: Data Required Path:
Info:
Info: Total (ns)  Incr (ns)      Type  Node
Info: =====  =====  ==  ====
=====
Info:          4.000      4.000           latch edge time
Info:          7.041      3.041  R           clock network delay
Info:          7.077      0.036   uTsu   inst5
Info:
Info: Data Arrival Time   :      3.443
Info: Data Required Time  :      7.077
Info: Slack               :      3.634
Info:
=====
Info:
1 3.634
```

Timing Assignment Conversion

This section describes Classic Timing Analyzer QSF timing assignments and the equivalent TimeQuest analyzer constraints. In some cases, there is more than one potentially equivalent SDC constraint. In these cases, correct conversion requires knowledge of the intended function of your design.

This section includes the following topics:

- “Clock Enable Multicycle” on page 8–28
- “Clock Latency” on page 8–25
- “Clock Uncertainty” on page 8–25
- “Cut Timing Path” on page 8–40
- “Default Required f_{MAX} Assignment” on page 8–26
- “Hold Relationship” on page 8–25
- “Input and Output Delay” on page 8–29
- “Inverted Clock” on page 8–25
- “Maximum Clock Arrival Skew” on page 8–41
- “Maximum Data Arrival Skew” on page 8–41
- “Maximum Delay” on page 8–40
- “Minimum Delay” on page 8–41
- “Minimum t_{CO} Requirement” on page 8–36
- “Minimum t_{PD} Requirement” on page 8–40
- “Multicycle” on page 8–27
- “Not a Clock” on page 8–26
- “Setup Relationship” on page 8–24
- “ t_{CO} Requirement” on page 8–34
- “ t_H Requirement” on page 8–32
- “ t_{PD} Requirement” on page 8–38
- “ t_{SU} Requirement” on page 8–30
- “Virtual Clock Reference” on page 8–26

Setup Relationship

The **Setup Relationship** assignment overrides the setup relationship between two clocks. By default, the Classic Timing Analyzer automatically calculates the setup relationship based on your clock settings. The QSF variable for the **Setup Relationship** assignment is `SETUP_RELATIONSHIP`. In the TimeQuest analyzer, use the `set_max_delay` command to specify the maximum setup relationship for a path.

The setup relationship value is the time between latch and launch edges before the TimeQuest analyzer accounts for clock latency, source μt_{CO} , or destination μt_{SU} .

Hold Relationship

The **Hold Relationship** assignment overrides the hold relationship between two clocks. By default, the Classic Timing Analyzer automatically calculates the hold relationship based on your clock settings. The QSF variable for the **Hold Relationship** assignment is `HOLD_RELATIONSHIP`. In the TimeQuest analyzer, use the `set_min_delay` command to specify the minimum hold relationship for a path.

Clock Latency

Table 8-1 shows the equivalent SDC constraints for the **Early Clock Latency** and **Late Clock Latency** Classic Timing Analyzer assignments.

Table 8-1. Classic Timing Analyzer Assignments and SDC Equivalent Constraints for Clock Latency Assignments

Classic Timing Analyzer Assignment		SDC Constraint
Assignment Name	QSF Variable	
Early Clock Latency	<code>EARLY_CLOCK_LATENCY</code>	<code>set_clock_latency -source -late</code>
Late Clock Latency	<code>LATE_CLOCK_LATENCY</code>	<code>set_clock_latency -source -early</code>

For more information about clock latency support in the TimeQuest analyzer, refer to “[Clock Latency](#)” on page 8-11.

Clock Uncertainty

Table 8-2 shows the equivalent SDC constraints for the **Clock Setup Uncertainty** and **Clock Hold Uncertainty** Classic Timing Analyzer assignments.

Table 8-2. Classic Timing Analyzer Assignments and SDC Equivalent Constraints for Clock Uncertainty Assignments

Classic Timing Analyzer Timing Assignment		SDC Constraint
Assignment Name	QSF Variable	
Clock Setup Uncertainty	<code>CLOCK_SETUP_UNCERTAINTY</code>	<code>set_clock_uncertainty -setup</code>
Clock Hold Uncertainty	<code>CLOCK_HOLD_UNCERTAINTY</code>	<code>set_clock_uncertainty -hold</code>

Inverted Clock

The Classic Timing Analyzer detects inverted clocks automatically when the clock inversion occurs at the input of the LCELL that contains the register specified in the assignment. You must make an **Inverted Clock** assignment in all other situations for Classic Timing Analyzer analysis. The QSF variable for the **Inverted Clock** assignment is `INVERTED_CLOCK`. The TimeQuest analyzer detects inverted clocks automatically, regardless of the type of inversion circuit, in designs that target device families that support unateness. For designs that target any other device family, you must create a generated clock with the `-invert` option on the output of the cell that inverts the clock.

Not a Clock

The **Not a Clock** assignment directs the Classic Timing Analyzer to identify specified node as not a clock source when it would normally be detected as a clock because of a global f_{MAX} requirement. The QSF variable for the **Not a Clock** assignment is `NOT_A_CLOCK`. This assignment is not supported in the TimeQuest analyzer and there is no equivalent constraint. Appropriate clock constraints are created in the TimeQuest analyzer only.

Default Required f_{MAX} Assignment

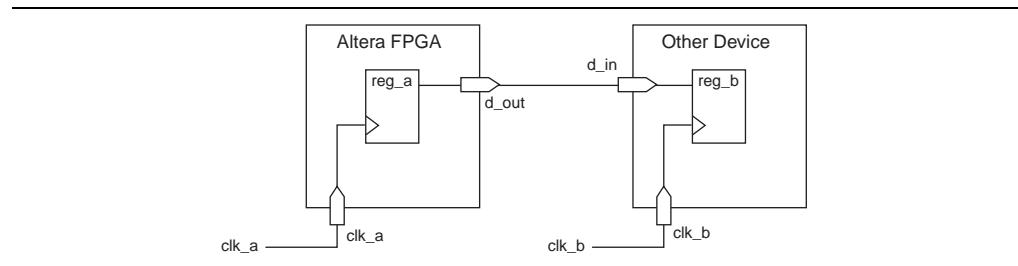
The **Default Required f_{MAX}** assignment allows you to specify an f_{MAX} requirement for the Classic Timing Analyzer for all unconstrained clocks in your design. The QSF variable for the **Default Required f_{MAX}** assignment is `FMAX_REQUIREMENT`. You can use the `derive_clocks` command to create clocks on sources of clock pins in your design that do not already have clocks assigned to them. You should constrain each individual clock in your design with the `create_clock` or `created_generated_clock` command, instead of the `derive_clocks` command. Refer to “[Automatic Clock Detection](#)” on page 8-14 to learn why you should constrain individual clocks in your design.

Virtual Clock Reference

The **Virtual Clock Reference** assignment allows you to define timing characteristics of a reference clock outside the FPGA. The QSF variable for the **Virtual Clock Reference** assignment is `VIRTUAL_CLOCK_REFERENCE`. The TimeQuest analyzer supports virtual clocks by default, while the Classic Timing Analyzer requires the **Virtual Clock Reference** assignment to indicate that a clock setting is for a virtual clock. To create a virtual clock in the TimeQuest analyzer, use the `create_clock` or `create_generated_clock` commands with the `-name` option and no targets.

[Figure 8-22](#) shows a circuit that requires a virtual clock, and the following example shows how to constrain the circuit. The circuit shows data transfer between an Altera FPGA and another device, and the clocks for the two devices are not related. You can constrain the path with an output delay assignment, but that assignment requires a virtual clock that defines the clock characteristics of the destination device.

Figure 8-22. Virtual Clock Circuit



Assume the circuit has the following assignments in the Classic Timing Analyzer:

- Clock period of 10 ns on `system_clk` (clock for the Altera FPGA)
- Clock period of 8 ns on `virt_clk` (clock for the other device)
- The **Virtual Clock Reference** setting for `virt_clk` is **On** (indicates that `virt_clk` is a virtual clock)
- The **Output Maximum Delay** setting of 5 ns on `dataout` with respect to `virt_clk` (constrains the path between the two devices)

The SDC commands shown in [Example 8-9](#) constrain the circuit the same way.

Example 8-9. SDC Constraints

```
# Clock for the Altera FPGA
create_clock -period 10 -name system_clk [get_ports system_clk]
# Virtual clock for the other device, with no targets
create_clock -period 8 -name virt_clk
# Constrains the path between the two devices
set_output_delay -clock virt_clk 5 [get_ports dataout]
```

Clock Settings

The Classic Timing Analyzer includes a variety of assignments to describe clock settings. These include duty cycle, f_{MAX} , offset, and others. In the TimeQuest analyzer, use the `create_clock` and `create_generated_clock` commands to constrain clocks.

Multicycle

[Table 8-3](#) shows the equivalent SDC exceptions for each of these Classic Timing Analyzer timing assignments.

Table 8-3. Classic Timing Analyzer Multicycle Assignments and SDC Equivalent Exceptions

Classic Timing Analyzer Multicycle Assignment		SDC Exception
Assignment Name	QSF Variable	
Multicycle (1)	MULTICYCLE	<code>set_multicycle_path -setup -end</code>
Source Multicycle (2)	SRC_MULTICYCLE	<code>set_multicycle_path -setup -start</code>
Multicycle Hold (3)	HOLD_MULTICYCLE	<code>set_multicycle_path -hold -end</code>
Source Multicycle Hold	SRC_HOLD_MULTICYCLE	<code>set_multicycle_path -hold -start</code>

Notes to Table 8-3:

- (1) A multicycle assignment is also known as a “destination multicycle setup” assignment.
- (2) A source multicycle assignment is also known as a “source multicycle setup” assignment.
- (3) A multicycle hold is also known as a “destination multicycle hold” assignment.

The default value and numbering scheme for the hold multicycle value is different in the Classic Timing Analyzer and TimeQuest analyzer. Refer to “[Hold Multicycle](#)” on [page 8–18](#) for more information about the difference between the default value and numbering scheme for the hold multicycle value in the Classic Timing Analyzer and TimeQuest analyzer.



For more information about how to convert the hold multicycle value, refer to the *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Clock Enable Multicycle

The Classic Timing Analyzer supports the following clock enable multicycle assignments:

- **Clock Enable Multicycle**
- **Clock Enable Source Multicycle**
- **Clock Enable Multicycle Hold**
- **Clock Enable Source Multicycle Hold**

Corresponding types of multicycle assignments are applied to all registers enabled by the targets of the specified clock. The TimeQuest analyzer supports clock-enabled multicycle constraints with the `get_fanouts` command. Use the `get_fanouts` command to create a collection of nodes that have a common source signal, such as a clock enable.

I/O Constraints

FPGA I/O timing assignments have typically been made with FPGA-centric t_{SU} and t_{CO} requirements for the Classic Timing Analyzer. However, the Classic Timing Analyzer also supports input and output delay assignments to accommodate industry-standard, system-centric timing constraints. Where possible, you should use system-centric constraints to constrain your designs for the TimeQuest analyzer. [Table 8–4](#) includes Classic Timing Analyzer I/O assignments, the equivalent FPGA-centric SDC constraints, and recommended system-centric SDC constraints.

For setup checks (t_{SU} and t_{CO}), $\langle latch - launch \rangle$ equals the clock period for same-clock transfers. For hold checks (t_H and Minimum t_{CO}), $\langle latch - launch \rangle$ equals 0 for same-clock transfers. Conversions from Classic Timing Analyzer assignments to `set_input_delay` and `set_output_delay` constraints work well only when the source and destination registers' clocks are the same (same clock and polarity). If the source and destination registers' clocks are different, the conversion may not be straightforward and you should take extra care when converting to `set_input_delay` and `set_output_delay` constraints.

Table 8-4. Classic Timing Analyzer and TimeQuest Analyzer Equivalent I/O Constraints

Classic Timing Analyzer Assignment	FPGA-centric SDC	System-centric SDC
t_{SU} Requirement	<code>set_max_delay <t_{SU} requirement></code>	<code>set_input_delay -max <latch - launch -- t_{SU} requirement></code>
t_H Requirement	<code>set_min_delay - <t_H requirement> (1)</code>	<code>set_input_delay -min <latch -- launch + t_H requirement></code>
t_{CO} Requirement	<code>set_max_delay <t_{CO} requirement></code>	<code>set_output_delay -max <latch -- launch - t_{CO} requirement></code>
Minimum t_{CO} Requirement	<code>set_min_delay <minimum t_{CO} requirement></code>	<code>set_output_delay -min <latch -- launch - minimum t_{CO} requirement></code>
t_{PD} Requirement	<code>set_max_delay <t_{PD} requirement></code>	(2)
Minimum t_{PD} Requirement	<code>set_min_delay <minimum t_{PD} requirement></code>	(2)

Notes to Table 8-4:

- (1) Refer to “ t_H Requirement” on page 8-32 for an explanation about why this exception uses the negative t_H requirement.
- (2) The input and output delays can be used for t_{PD} paths, such that they will be analyzed as a system f_{MAX} path. This is a feature unique to the TimeQuest analyzer.

Input and Output Delay

Table 8-5 shows the equivalent SDC exceptions for Classic Timing Analyzer timing assignments.

Table 8-5. Classic Timing Analyzer Assignments and SDC Equivalent Exceptions

Classic Timing Analyzer Assignment		SDC Exception
Assignment Name	QSF Variable	
Input Maximum Delay	INPUT_MAX_DELAY	<code>set_input_delay -max</code>
Input Minimum Delay	INPUT_MIN_DELAY	<code>set_input_delay -min</code>
Output Maximum Delay	OUTPUT_MAX_DELAY	<code>set_output_delay -max</code>
Output Minimum Delay	OUTPUT_MIN_DELAY	<code>set_output_delay -min</code>

In some circumstances, you may receive the following warning message when you update the SDC netlist:

```
Warning: For set_input_delay/set_output_delay, port "<port>" does not have delay for flag (<rise|fall>, <min|max>)
```

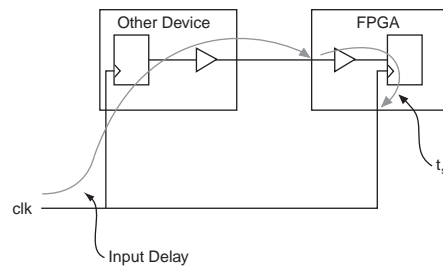

Equation 8-3 shows the derivation of this conversion.

Equation 8-3.

$$\begin{aligned} \text{required} - \text{arrival} &> 0 \\ \text{required} &= \text{latch} + \text{board.dstclk} + \text{dst.clk} - \text{dst.utsu} \\ \text{arrival} &= \text{launch} + \text{board.srcclk} + \text{src.clk} + \text{src.utco} + \text{src.out} + \text{srctodst} + \text{dst.in} \\ \text{input_delay} &= \text{board.srcclk} + \text{src.clk} + \text{src.utcu} + \text{src.out} + \text{srctodst} - \text{board.dstclk} \\ \text{required} &= \text{latch} + \text{dst.clk} - \text{dst.utsu} \\ \text{arrival} &= \text{launch} + \text{input_delay} + \text{dst.in} \\ (\text{latch} + \text{dst.clk} - \text{dst.utsu}) - (\text{launch} + \text{input_delay} + \text{dst.in}) &> 0 \\ t_{su} \text{ requirement} - \text{actual } t_{su} &> 0 \\ \text{actual } t_{su} &= \text{dst.in} + \text{dst.utsu} - \text{dst.clk} \\ t_{su} \text{ requirement} - (\text{dst.in} + \text{dst.utsu} - \text{dst.clk}) &> 0 \\ t_{su} \text{ requirement} &= \text{latch} - \text{launch} - \text{input_delay} \\ \text{input_delay} &= \text{latch} - \text{launch} - t_{su} \text{ requirement} \end{aligned}$$

The delay value is the difference between the period of the clock source of the register and the t_{su} Requirement value, as shown in Figure 8-24.

Figure 8-24. t_{su} Requirement



The delay value for the set_max_delay command is the t_{su} Requirement value. Equation 8-4 shows the derivation of this conversion.

Equation 8-4.

$$\begin{aligned} \text{required} - \text{arrival} &> 0 \\ \text{required} &= \text{latch} + \text{board.dstclk} + \text{dst.clk} - \text{dst.utsu} \\ \text{arrival} &= \text{launch} + \text{board.srcclk} + \text{src.clk} + \text{src.utco} + \text{src.out} + \text{srctodst} + \text{dst.in} \\ \text{max_delay} &= \text{latch} + \text{board.dstclk} + - \text{launch} - \text{board.srcclk} - \text{src.clk} - \text{src.out} - \text{srctodst} \\ \text{required} &= \text{max_delay} + \text{dst.clk} - \text{dst.utsu} \\ \text{arrival} &= \text{dst.in} \\ (\text{max_delay} + \text{dst.clk} - \text{dst.utsu}) - (\text{dst.in}) &> 0 \\ t_{su} \text{ requirement} - t_{su} &> 0 \\ \text{actual } t_{su} &= \text{dst.in} + \text{dst.utsu} - \text{dst.clk} \\ t_{su} \text{ requirement} - (\text{dst.in} + \text{dst.utsu} - \text{dst.clk}) &> 0 \\ \text{set_max_delay} &= t_{su} \text{ requirement} \end{aligned}$$

Table 8-6 shows the different ways you can make t_{SU} assignments in the Classic Timing Analyzer, and the corresponding options for the `set_max_delay` exception.

Table 8-6. t_{SU} Requirement and `set_max_delay` Equivalence

t_{SU} Requirement Options	<code>set_max_delay</code> Options
-to <pin>	-from [get_ports <pin>] -to [get_registers *]
-to <clock>	-from [get_ports *] -to [get_clocks <clock>]
-to <register>	-from [get_ports *] -to [get_registers <register>]
-from <pin> -to <register>	-from [get_ports <pin>] -to [get_registers <register>]
-from <clock> -to <pin>	-from [get_ports <pin>] -to [get_clocks <clock>] (1)

Note to Table 8-6:

- (1) If the pin in this assignment feeds registers clocked by the same clock, it is equivalent to the first option, -to <pin>. If the pin feeds registers clocked by different clocks, use `set_input_delay` to constrain the paths properly.

To convert a global t_{SU} assignment to an equivalent SDC exception, use the command shown in Example 8-10.

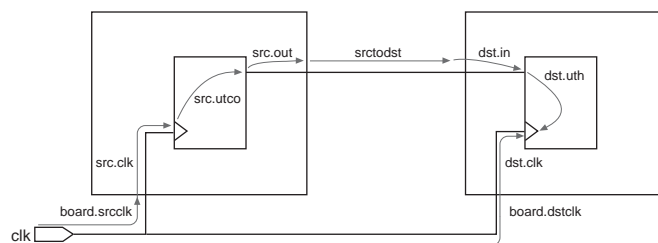
Example 8-10. Converting a Global t_{SU} Assignment to an Equivalent SDC Exception

```
set_max_delay -from [all_inputs] -to [all_registers] < $t_{SU}$  value>
```

t_H Requirement

The t_H Requirement specifies the maximum acceptable clock hold time for the input (data) pin. The QSF variable for the t_H Requirement assignment is `TH_REQUIREMENT`. You can convert the t_H Requirement assignment to the `set_min_delay` command, or the `set_input_delay` command with the `-min` option. The delay value for the `set_input_delay` command is $\langle \text{latch} - \text{launch} + t_H \text{ requirement} \rangle$. Refer to the labeled paths in Figure 8-25 to understand the names in Equation 8-5 and Equation 8-6.

Figure 8-25. Path Names



Equation 8-5 shows the derivation of this calculation.

Equation 8-5.

$$\begin{aligned}
 & \text{arrival} - \text{required} > 0 \\
 & \text{arrival} = \text{launch} + \text{board.srcclk} + \text{src.clk} + \text{src.utco} + \text{src.out} + \text{srctodst} + \text{dst.in} \\
 & \text{required} = \text{latch} + \text{board.dstclk} + \text{dst.clk} + \text{dst.uth} \\
 & \text{input_delay} = \text{board.srcclk} + \text{src.clk} + \text{srcutcu} + \text{src.out} + \text{srctodst} - \text{board.dstclk} \\
 & \text{arrival} = \text{launch} + \text{input_delay} + \text{dst.in} \\
 & \text{required} = \text{latch} + \text{dst.clk} + \text{dst.uth} \\
 & (\text{launch} + \text{input_delay} + \text{dst.in}) - (\text{latch} + \text{dst.clk} + \text{dst.uth}) > 0 \\
 & t_H \text{ requirement} - \text{actual } t_H > 0 \\
 & \text{actual } t_H = \text{dst.clk} + \text{dst.uth} - \text{dst.in} \\
 & t_H \text{ requirement} - (\text{dst.clk} + \text{dst.uth} - \text{dst.in}) > 0 \\
 & t_H \text{ requirement} = \text{launch} - \text{latch} + \text{input_delay} \\
 & \text{input_delay} = \text{latch} - \text{launch} + t_H \text{ requirement}
 \end{aligned}$$

The delay value for the `set_min_delay` command is the **t_H Requirement** value. Equation 8-6 shows the derivation of this conversion.

Equation 8-6.

$$\begin{aligned}
 & \text{arrival} - \text{required} > 0 \\
 & \text{arrival} = \text{dst.in} \\
 & \text{required} = \text{min_delay} + \text{dst.clk} + \text{dst.uth} \\
 & \text{dst.in} - (\text{min_delay} + \text{dst.clk} + \text{dst.uth}) \\
 & t_H \text{ requirement} - \text{actual } t_H > 0 \\
 & \text{actual } t_H = \text{dst.clk} + \text{dst.uth} - \text{dst.in} \\
 & t_H \text{ requirement} - (\text{dst.clk} + \text{dst.uth} - \text{dst.in}) > 0 \\
 & \text{set_min_delay} = -t_H \text{ requirement}
 \end{aligned}$$

Table 8-7 shows the different ways you can make t_H assignments in the Classic Timing Analyzer, and the corresponding options for the `set_min_delay` command.

Table 8-7. t_H Requirement and `set_min_delay` Equivalence

t_H Requirement Options	<code>set_min_delay</code> Options
-to <pin>	-from [get_ports <pin>] -to [get_registers *]
-to <clock>	-from [get_ports *] -to [get_clocks <clock>]
-to <register>	-from [get_ports *] -to [get_registers <register>]
-from <pin> -to <register>	-from [get_ports <pin>] -to [get_registers <register>]
-from <clock> -to <pin>	-from [get_ports <pin>] -to [get_clocks <clock>] (1)

Note to Table 8-7:

- (1) If the pin in this assignment feeds registers clocked by the same clock, it is equivalent to the first option, -to <pin>. If the pin feeds registers clocked by different clocks, use `set_input_delay` to constrain the paths properly. Refer to “Input and Output Delay” on page 8-29 for additional information.

To convert a global t_H assignment to an equivalent SDC exception, use the command shown in Example 8-11.

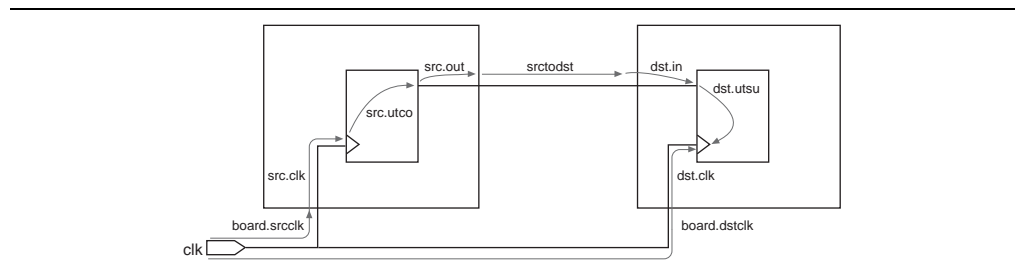
Example 8-11. Converting a Global t_H Assignment to an Equivalent SDC Exception

```
set_min_delay -from [all_inputs] -to [all registers] <negative  $t_H$  value>
```

t_{CO} Requirement

The t_{CO} Requirement assignment specifies the maximum acceptable clock to output delay to the output pin. The QSF variable for the t_{CO} Requirement assignment is `TCO_REQUIREMENT`. You can convert the t_{CO} Requirement assignment to the `set_max_delay` command or the `set_output_delay` command with the `-max` option. The delay value for the `set_output_delay` command is `<latch - launch - t_{CO} requirement>`. Refer to the labeled paths in Figure 8-26 to understand the names in Equation 8-7 and Equation 8-8.

Figure 8-26. Path Names



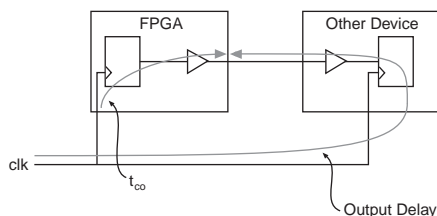
Equation 8-7 shows the derivation of this conversion.

Equation 8-7.

$$\begin{aligned} \text{required} - \text{arrival} &> 0 \\ \text{required} &= \text{latch} - \text{output_delay} \\ \text{arrival} &= \text{launch} + \text{src.clk} + \text{src.utco} + \text{src.out} \\ \text{output_delay} &= \text{srctodst} + \text{dst.in} + \text{dst.utsu} - \text{dst.clk} - \text{board.dstc.k} + \text{board.srcclk} \\ (\text{latch} - \text{output_delay}) - (\text{launch} + \text{src.clk} + \text{src.utco} + \text{src.out}) &> 0 \\ t_{\text{co}} \text{ requirement} - \text{actual } t_{\text{co}} &> 0 \\ \text{actual } t_{\text{co}} &= \text{launch} + \text{src.clk} + \text{src.utco} + \text{src.out} \\ t_{\text{co}} \text{ requirement} - (\text{src.clk} + \text{src.utco} + \text{src.out}) &> 0 \\ t_{\text{co}} \text{ requirement} &= \text{latch} - \text{launch} - \text{output_delay} \\ \text{output_delay} &= \text{latch} - \text{launch} - t_{\text{co}} \text{ requirement} \end{aligned}$$

The delay value is the difference between the period of the clock source of the register and the t_{CO} Requirement value, as illustrated in Figure 8-27.

Figure 8-27. t_{CO} Requirement



The delay value for the set_max_delay command is the t_{CO} Requirement value. Equation 8-8 shows the derivation of this conversion.

Equation 8-8.

$$\begin{aligned} \text{required} - \text{arrival} &> 0 \\ \text{required} &= \text{set_max_delay} \\ \text{arrival} &= \text{src.clk} + \text{src.utco} + \text{src.out} \\ \text{set_max_delay} - (\text{src.clk} + \text{src.utco} + \text{src.out}) &> 0 \\ t_{\text{co}} \text{ requirement} - \text{actual } t_{\text{co}} &> 0 \\ \text{actual } t_{\text{co}} &= \text{src.clk} + \text{src.utco} + \text{src.out} \\ t_{\text{co}} \text{ requirement} - (\text{src.clk} + \text{src.utco} + \text{src.out}) &> 0 \\ \text{set_max_delay} &= t_{\text{co}} \text{ requirement} \end{aligned}$$

Table 8-8 shows the different ways you can make t_{CO} assignments in the Classic Timing Analyzer, and the corresponding options for the `set_max_delay` exception.

Table 8-8. t_{CO} Requirement and `set_max_delay` Equivalence

t_{CO} Requirement Options	<code>set_max_delay</code> Options
-to <pin>	-from [get_registers *] -to [get_ports <pin>]
-to <clock>	-from [get_clocks <clock>] -to [get_ports *]
-to <register>	-from [get_registers <register>] -to [get_ports *]
-from <register> -to <pin>	-from [get_registers <register>] -to [get_ports <pin>]
-from <clock> -to <pin>	-from [get_clocks <clock>] -to [get_ports <pin>] (1)

Note to Table 8-8:

- (1) If the pin in this assignment feeds registers clocked by the same clock, it is equivalent to the first option, -to <pin>. If the pin feeds registers clocked by different clocks, you should use `set_output_delay` to constrain the paths properly.

To convert a global t_{CO} assignment to an equivalent SDC exception, use the command in Example 8-12.

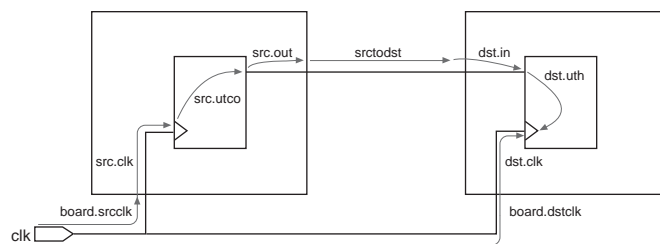
Example 8-12. Converting a Global t_{CO} Assignment to an Equivalent SDC Exception

```
set_max_delay -from [all registers] -to [all_outputs] < $t_{CO}$  value>
```

Minimum t_{CO} Requirement

The **Minimum t_{CO} Requirement** assignment specifies the minimum acceptable clock to output delay to the output pin. The QSF variable for the **Minimum t_{CO} Requirement** assignment is `MIN_TCO_REQUIREMENT`. You can convert the **Minimum t_{CO} Requirement** assignment to the `set_min_delay` command or the `set_output_delay` command with the `-min` option. The delay value for the `set_output_delay` command is <latch – launch – minimum t_{CO} requirement>. Refer to the labeled paths in Figure 8-28 to understand the names in Equation 8-9 and Equation 8-10.

Figure 8-28. Path Names



Equation 8-9 shows the derivation of this conversion.

Equation 8-9.

$$\begin{aligned}
 & \text{arrival} + \text{required} > 0 \\
 & \text{arrival} = \text{launch} + \text{src.clk} + \text{src.utco} + \text{src.out} \\
 & \text{required} = \text{latch} - \text{output_delay} \\
 & \text{output_delay} = \text{srctodst} + \text{dst.in} - \text{dst.uth} - \text{dst.clk} - \text{board.dstclk} + \text{board.srcclk} \\
 & (\text{launch} + \text{src.clk} + \text{src.utco} + \text{src.out}) - (\text{latch} - \text{output_delay}) > 0 \\
 & \text{minimum } t_{\text{co}} - \text{minimum } t_{\text{co}} \text{ requirement} > 0 \\
 & \text{minimum } t_{\text{co}} = \text{launch} + \text{src.clk} + \text{src.utco} + \text{src.out} \\
 & (\text{launch} + \text{src.clk} + \text{src.utco} + \text{src.out}) - \text{minimum } t_{\text{co}} \text{ requirement} > 0 \\
 & \text{minimum } t_{\text{co}} \text{ requirement} = \text{latch} - \text{launch} - \text{output_delay} \\
 & \text{output_delay} = \text{latch} - \text{launch} - \text{minimum } t_{\text{co}} \text{ requirement}
 \end{aligned}$$

The delay value for the `set_min_delay` command is the **Minimum t_{co} Requirement**. Equation 8-10 shows the derivation of this conversion.

Equation 8-10.

$$\begin{aligned}
 & \text{arrival} - \text{required} > 0 \\
 & \text{arrival} = \text{src.clk} + \text{src.utco} + \text{src.out} \\
 & \text{required} = \text{min_delay} \\
 & (\text{src.clk} + \text{src.utco} + \text{src.out}) - (\text{set_min_delay}) > 0 \\
 & \text{minimum } t_{\text{co}} - \text{minimum } t_{\text{co}} \text{ requirement} > 0 \\
 & \text{minimum } t_{\text{co}} = \text{src.clk} + \text{src.utco} + \text{src.out} \\
 & (\text{src.clk} + \text{src.utco} + \text{src.out}) - \text{minimum } t_{\text{co}} \text{ requirement} > 0 \\
 & \text{set_min_delay} = \text{minimum } t_{\text{co}} \text{ requirement}
 \end{aligned}$$

Table 8-9 shows the different ways you can make minimum t_{CO} assignments in the Classic Timing Analyzer, and the corresponding options for the `set_min_delay` exception.

Table 8-9. Minimum t_{CO} Requirement and `set_min_delay` Equivalence

Minimum t_{CO} Requirement Options	<code>set_min_delay</code> Options
-to <pin>	-from [get_registers *] -to [get_ports <pin>]
-to <clock>	-from [get_clocks <clock>] -to [get_ports *]
-to <register>	-from [get_registers <register>] -to [get_ports *]
-from <register> -to <pin>	-from [get_registers <register>] -to [get_ports <pin>]
-from <clock> -to <pin>	-from [get_clocks <clock>] -to [get_ports <pin>] (1)

Note to Table 8-9:

- (1) If the pin in this assignment feeds registers clocked by the same clock, it is equivalent to the first option, -to <pin>. If the pin feeds registers clocked by different clocks, use `set_output_delay` to constrain the paths properly.

To convert a global **Minimum t_{CO} Requirement** to an equivalent SDC exception, use the command shown in Example 8-13.

Example 8-13. Converting a Global Minimum t_{CO} Requirement to an Equivalent SDC Exception

```
set_min_delay -from [all_registers] -to [all_outputs] <minimum tCO value>
```

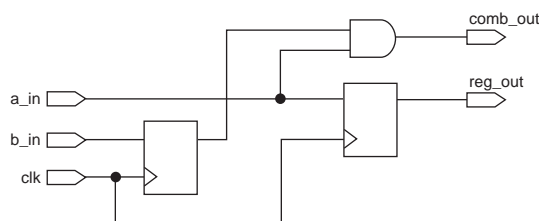
t_{PD} Requirement

The t_{PD} **Requirement** assignment specifies the maximum acceptable input to nonregistered output delay, that is, the time required for a signal from an input pin to propagate through combinational logic and appear at an output pin. The QSF variable for the t_{PD} **Requirement** assignment is `TPD_REQUIREMENT`. You can use the `set_max_delay` command in the TimeQuest analyzer as an equivalent constraint as long as you account for input and output delays. The t_{PD} **Requirement** assignment does not take into account input and output delays, but the `set_max_delay` exception does, so you must modify the `set_max_delay` value to take into account input and output delays.

Combinational Path Delay Scenario

Figure 8-29 shows a circuit to illustrate the following example of converting a t_{PD} **Requirement** to a `set_max_delay` constraint.

Figure 8-29. t_{PD} Example



Assume the circuit has the following assignments in the Classic Timing Analyzer:

- Clock period of 10 ns
- **t_{PD} Requirement** from a_in to comb_out of 10 ns
- **Input Max Delay** on a_in relative to clk of 2 ns
- **Output Max Delay** on comb_out relative to clk of 2 ns

The path from a_in to comb_out is not affected by the input and output delays. The slack is equal to the $\langle t_{PD} \text{ Requirement from a_in to comb_out} \rangle - \langle \text{path delay from a_in to comb_out} \rangle$.

Assume the circuit has the SDC constraints shown in [Example 8-14](#) in the TimeQuest analyzer.

Example 8-14. SDC Constraints

```
create_clock -period 10 -name clk [get_ports clk]
set_max_delay -from a_in -to comb_out 10
set_input_delay -clk clk 2 [get_ports a_in]
set_output_delay -clk clk 2 [get_ports comb_out]
```

The path from a_in to comb_out is affected by the input and output delays. The slack is equal to:

$\langle \text{set_max_delay value from a_in to comb_out} \rangle - \langle \text{input delay} \rangle - \langle \text{output delay} \rangle - \langle \text{path delay from a_in to comb_out} \rangle$

To convert a global **t_{PD} Requirement** assignment to an equivalent SDC exception, use the command shown in [Example 8-15](#). You should add the input and output delays to the value of your converted **t_{PD} Requirement** (**set_max_delay** exception value) to achieve an equivalent SDC exception.

Example 8-15. Converting a Global t_{PD} Requirement Assignment to an Equivalent SDC Exception

```
set_max_delay -from [all_inputs] -to [all_outputs] <value>
```

Minimum t_{PD} Requirement

The **Minimum t_{PD} Requirement** assignment specifies the minimum acceptable input to nonregistered output delay, that is, the minimum time required for a signal from an input pin to propagate through combinational logic and appear at an output pin. The QSF variable for the **Minimum t_{PD} Requirement** assignment is `MIN_TPD_REQUIREMENT`. You can use the `set_min_delay` command in the TimeQuest analyzer as an equivalent constraint as long as you account for input and output delays. The **Minimum t_{PD} Requirement** assignment does not take into account input and output delays, but the `set_min_delay` exception does.

Refer to “[Combinational Path Delay Scenario](#)” on page 8-38 to see how input and output delays affect minimum and maximum delay exceptions.

To convert a global **Minimum t_{PD} Requirement** assignment to an equivalent SDC exception, use the command shown in [Example 8-16](#).

Example 8-16. Converting a Global Minimum t_{PD} Requirement Assignment to an Equivalent SDC Exception

```
set_min_delay -from [all_inputs] -to [all_outputs] <value>
```

Cut Timing Path

The **Cut Timing Path** assignment in the Classic Timing Analyzer is equivalent to the `set_false_path` command in the TimeQuest analyzer. The QSF variable for the **Cut Timing Path** assignment is `CUT`.

Maximum Delay

The **Maximum Delay** assignment specifies the maximum required delay for the following types of paths:

- Pins to registers
- Registers to registers
- Registers to pins

The QSF variable for the **Maximum Delay** assignment is `MAX_DELAY`. This overwrites the requirement computed from the clock setup relationship and clock skew. There is no equivalent constraint in the TimeQuest analyzer.



The **Maximum Delay** assignment for the Classic Timing Analyzer is not related to the `set_max_delay` exception in the TimeQuest analyzer.

Minimum Delay

The **Minimum Delay** assignment specifies the minimum required delay for the following types of paths:

- Pins to registers
- Registers to registers
- Registers to pins

The QSF variable for the **Minimum Delay** assignment is `MIN_DELAY`. This overwrites the requirement computed from the clock hold relationship and clock skew. There is no equivalent constraint in the TimeQuest analyzer.



The **Minimum Delay** assignment for the Classic Timing Analyzer is not related to the `set_min_delay` exception in the TimeQuest analyzer.

Maximum Clock Arrival Skew

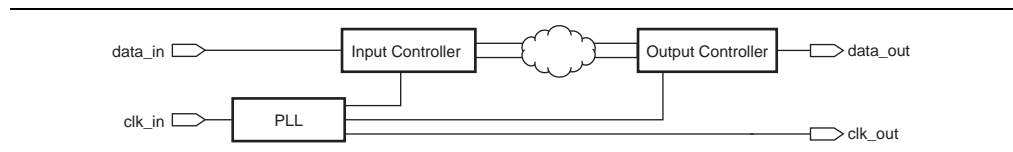
The **Maximum Clock Arrival Skew** assignment specifies the maximum clock skew between a set of registers. The QSF variable for the **Maximum Clock Arrival Skew** assignment is `MAX_CLOCK_ARRIVAL_SKEW`. In the Classic Timing Analyzer, this assignment is specified between a clock node name and a set of registers. **Maximum Clock Arrival Skew** is not supported in the TimeQuest analyzer.

Maximum Data Arrival Skew

The **Maximum Data Arrival Skew** assignment specifies the maximum data arrival skew between a set of registers, pins, or both. The QSF variable for the **Maximum Data Arrival Skew** assignment is `MAX_DATA_ARRIVAL_SKEW`. In this case, the data arrival delay represents the t_{CO} from the clock to the given register, pin, or both. This assignment is specified between a clock node and a set of registers, pins, or both.

The TimeQuest analyzer does not support a constraint to specify maximum data arrival skew, but you can specify setup and hold times relative to a clock port to constrain an interface like this. [Figure 8-30](#) shows a simplified source-synchronous interface used in the following example.

Figure 8-30. Source-Synchronous Interface Diagram

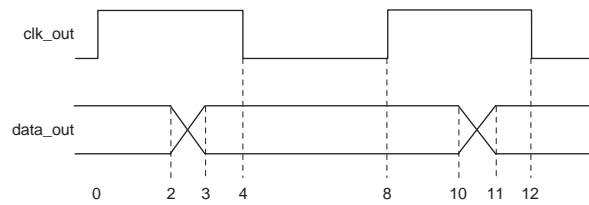


Constraining Skew on an Output Bus

This example constrains the interface so that all bits of the `data_out` bus go off-chip between 2 ns and 3 ns after the `clk_out` signal. Assume that `clk_in` and `clk_out` have a period of 8 ns.

The following equations and example show how to create timing requirements that satisfy the timing relationships shown in [Figure 8-31](#).

Figure 8-31. Source-Synchronous Timing Diagram



[Equation 8-11](#) shows how to compute the value for the `set_output_delay -min` command that creates the 2 ns hold requirement on the destination. For hold requirement calculations in which source and destination clocks are the same, $\langle \text{latch} \rangle - \langle \text{launch} \rangle = 0$.

Equation 8-11.

$$\begin{aligned} \text{latch} - \text{launch} &= 0\text{ns} \\ \text{output delay} &= \text{latch} - \text{launch} - 2\text{ns} \\ \text{output delay} &= -2\text{ns} \end{aligned}$$

[Equation 8-12](#) shows how to compute the value for the `set_output_delay` command that creates the 3 ns setup requirement on the destination. For setup requirement calculations in which source and destination clocks are the same, $\langle \text{latch} \rangle - \langle \text{launch} \rangle = \text{clock period}$.

Equation 8-12.

$$\begin{aligned} \text{latch} - \text{launch} &= 8\text{ns} \\ \text{output delay} &= \text{latch} - \text{launch} - 3\text{ns} \\ \text{output delay} &= 5\text{ns} \end{aligned}$$

Refer to [“I/O Constraints” on page 8-28](#) for an explanation of the above equations.

[Example 8-17](#) shows the three constraints together.

Example 8-17. Constraining a DDR Interface

```
set period 8.000
create_clock -period $period \
             -name clk_in \
             [get_ports data_out*]
derive_pll_clocks
set_output_delay -add_delay \
                 -clock ddr_pll_1_inst|altpll_component|pll|CLK[0] \
                 -reference_pin [get_ports clk_out] \
                 -min -2.000 \
                 [get_ports data_out*]
set_output_delay -add_delay \
                 -clock ddr_pll_1_inst|altpll_component|pll|CLK[0] \
                 -reference_pin [get_ports clk_out] \
                 -max [expr $period - 3.000] \
                 [get_ports data_out*]
```

Conversion Utility

The TimeQuest analyzer includes a conversion utility to help you convert Classic Timing Analyzer assignments in a .qsf to SDC constraints in an .sdc. The utility can use information from your project report database (in the \ldb folder), if it exists, so you should compile your design before the conversion.



The conversion utility ignores all disabled QSF assignments. Disabled assignments show **No** in the **Enabled** column of the Assignment Editor, and include the -disable option in the .qsf.

Refer to “[Create SDC Constraints from Existing Timing Assignments](#)” on page 8-2 to learn how to run the conversion utility.

Unsupported Global Assignments

The conversion utility checks whether any of the global timing assignments in [Table 8-10](#) exist in your project. Any global assignments not supported by the conversion utility are ignored during the conversion. Refer to the indicated page for information about each assignment, and how to manually convert these global assignments to SDC commands.

Table 8-10. Global Timing Assignments

Assignment Name	QSF Variable	More Information
t_{SU} Requirement	TSU_REQUIREMENT	page 8-30
t_H Requirement	TH_REQUIREMENT	page 8-32
t_{CO} Requirement	TCO_REQUIREMENT	page 8-34
Minimum t_{CO} Requirement	MIN_TCO_REQUIREMENT	page 8-36
t_{PD} Requirement	TPD_REQUIREMENT	page 8-38
Minimum t_{PD} Requirement	MIN_TPD_REQUIREMENT	page 8-40

Recommended Global Assignments

When any unsupported assignments have been identified, the conversion utility checks the global assignments in [Table 8-11](#) to ensure they match the specified values.

Table 8-11. Recommended Global Assignments

Classic Timing Analyzer Assignment Name	QSF Variable	Value
Cut off clear and preset signal paths	CUT_OFF_CLEAR_AND_PRESET_PATHS	ON
Cut off feedback from I/O pins	CUT_OFF_IO_PIN_FEEDBACK	ON
Cut off read during write signal paths	CUT_OFF_READ_DURING_WRITE_PATHS	ON
Analyze latches as synchronous elements	ANALYZE_LATCHES_AS_SYNCHRONOUS_ELEMENTS	ON
Enable Clock Latency	ENABLE_CLOCK_LATENCY	ON
Display Entity Name	PROJECT_SHOW_ENTITY_NAME	ON

The following assignments are checked to ensure the functionality of the Classic Timing Analyzer with the specified values corresponding to the behavior of the TimeQuest analyzer.

- **Cut off clear and preset signal paths**—The TimeQuest analyzer does not support this functionality. You should use Recovery and Removal analysis instead to analyze register control paths. The Classic Timing Analyzer does not support this option.
- **Cut off feedback from I/O pins**—The TimeQuest analyzer does not match the functionality of the Classic Timing Analyzer when this assignment is set to OFF.
- **Cut off read during write signal paths**—The TimeQuest analyzer does not match the functionality of the Classic Timing Analyzer when this assignment is set to OFF.
- **Analyze latches as synchronous elements**—The TimeQuest analyzer analyzes latches as synchronous elements by default and does not match the functionality of the Classic Timing Analyzer when this assignment is set to OFF. The Classic Timing Analyzer analyzes latches as synchronous elements by default.
- **Enable Clock Latency**—The TimeQuest analyzer includes clock latency in its calculations. The TimeQuest analyzer does not match the functionality of the Classic Timing Analyzer when this assignment is set to OFF. Latency on a clock can be viewed as a simple delay on the clock path, and affects clock skew. This is in contrast to an offset, which alters the setup and hold relationship between two clocks. Refer to [“Offset and Latency Example” on page 8-11](#) for an example of the different effects of offset and latency. When you turn on **Enable Clock Latency** in the Classic Timing Analyzer, it affects the following aspects of timing analysis:
 - **Early Clock Latency** and **Late Clock Latency** assignments are honored
 - The compensation delay of a PLL is analyzed as latency
 - For clock settings where you do not specify an offset, the automatically computed offset is treated as latency
- **Display Entity Name**—Any entity-specific assignments are ignored in the TimeQuest analyzer because they do not include the entity name when this option is set to ON.

If your design meets timing requirements in the Classic Timing Analyzer without all of the settings recommended in [Table 8-11 on page 8-43](#), you should perform one of the following actions:

- Change the settings and reconstrain and reverify as necessary.
or
- Add or modify SDC constraints as appropriate because analysis in the TimeQuest analyzer may be different after conversion.

Clock Conversion

Next, the conversion utility adds the `derive_pll_clocks` command to the `.sdc`. This command creates generated clocks on all PLL outputs in your design each time the `.sdc` is read. The command does not add a clock on the FPGA port that drives the PLL input.

The conversion utility includes the `derive_pll_clocks -use_net_name` command in the `.sdc` it creates. The `-use_net_name` option overrides the default clock naming behavior (the PLL pin name) so the clock name is the same as the net name in the Classic Timing Analyzer.

Including the `-use_net_name` option ensures that the conversion utility converts clock-to-clock exceptions properly. If you remove the `-use_net_name` option, you must also edit references to the clock names in other SDC commands so they match the PLL pin names.

If your design includes a global f_{MAX} assignment, the assignment is converted to a `derive_clocks` command. The behavior of a global f_{MAX} assignment is different from the behavior of clocks created with the `derive_clocks` command, and you should use the `report_clocks` command when you review conversion results to evaluate the clock settings. Refer to [“Automatic Clock Detection” on page 8-14](#) for an explanation of the differences. As soon as you know the appropriate clock settings, you should use the `create_clock` or `create_generated_clock` command instead of the `derive_clocks` command.



The conversion utility adds a `post_message` command before the `derive_clocks` command to remind you that the clocks are derived automatically. The TimeQuest analyzer displays the reminder the first time it reads the `.sdc`. Remove or comment out the `post_message` command to prevent the message from displaying.

Next, the conversion utility identifies and converts clock settings in the `.qsf`. If a project database exists, the utility also identifies and converts any additional clocks in the report file that are not in the `.qsf`, such as PLL base clocks.



If you change the PLL input frequency, you must modify the SDC constraint manually.

The conversion utility ignores clock offsets on generated clocks. Refer to [“Clock Offset” on page 8-10](#) for information about how to use offset values in the TimeQuest analyzer.

Instance Assignment Conversion

Next, the conversion utility converts the instance assignments shown in [Table 8-12](#). Refer to the indicated page for information about each assignment.

Table 8-12. Instance Timing Assignments

Assignment Name	QSF Variable	More Information
Late Clock Latency	LATE_CLOCK_LATENCY	page 8-25
Early Clock Latency	EARLY_CLOCK_LATENCY	
Clock Setup Uncertainty	CLOCK_SETUP_UNCERTAINTY	page 8-25
Clock Hold Uncertainty	CLOCK_HOLD_UNCERTAINTY	
Multicycle (1)	MULTICYCLE	page 8-27
Source Multicycle (2)	SRC_MULTICYCLE	
Multicycle Hold (3)	HOLD_MULTICYCLE	
Source Multicycle Hold	SRC_HOLD_MULTICYCLE	
Clock Enable Multicycle	CLOCK_ENABLE_MULTICYCLE	page 8-28
Clock Enable Source Multicycle	CLOCK_ENABLE_SOURCE_MULTICYCLE	
Clock Enable Multicycle Hold	CLOCK_ENABLE_MULTICYCLE_HOLD	
Clock Enable Source Multicycle Hold	CLOCK_ENABLE_SOURCE_MULTICYCLE_HOLD	
Cut Timing Path	CUT	page 8-40
Input Maximum Delay	INPUT_MAX_DELAY	page 8-29
Input Minimum Delay	INPUT_MIN_DELAY	
Output Maximum Delay	OUTPUT_MAX_DELAY	
Output Minimum Delay	OUTPUT_MIN_DELAY	

Notes to [Table 8-12](#):

- (1) A multicycle assignment can also be known as a “destination multicycle setup” assignment.
- (2) A source multicycle assignment can also be known as a “source multicycle setup” assignment.
- (3) A multicycle hold can also be known as a “destination multicycle hold” assignment.

Depending on input and output delay assignments, you may receive a warning message when the `.sdc` is read. The message warns that the `set_input_delay` command, `set_output_delay` command, or both are missing the `-max` option, `-min` option, or both. Refer to [“Input and Output Delay” on page 8-29](#) for an explanation of why the warning occurs and how to avoid it.

The conversion utility automatically adds multicycle hold exceptions for each multicycle setup assignment. The value of each multicycle hold exception depends on the **Default hold multicycle** assignment value in your project. If the value is **One**, the conversion utility uses a value of 0 (zero) for each multicycle hold exception it adds. If the value is **Same as multicycle**, the conversion utility uses a value one less than the corresponding multicycle setup assignment value for each multicycle hold exception it adds. Refer to [“Hold Multicycle” on page 8-18](#) for more information on hold multicycle differences between the Classic Timing Analyzer and the TimeQuest analyzer.

Next, the conversion utility converts the instance assignments shown in [Table 8-13](#). Refer to the indicated page for information about each assignment. If the t_{PD} and minimum t_{PD} assignment targets also have input or output delays that apply to them, the t_{PD} and minimum t_{PD} conversion values may be incorrect. This is described in more detail on the indicated pages for the appropriate assignments.

Table 8-13. Instance Timing Assignments

Assignment Name	QSF Variable	More Information
t_{PD} Requirement (1)	TPD_REQUIREMENT	page 8-38
Minimum t_{PD} Requirement (1)	MIN_TPD_REQUIREMENT	page 8-40
Setup Relationship	SETUP_RELATIONSHIP	page 8-24
Hold Relationship	HOLD_RELATIONSHIP	page 8-25

Note to [Table 8-13](#):

- (1) Refer to " [\$t_{PD}\$ and Minimum \$t_{PD}\$ Requirement Conversion](#)" on [page 8-55](#) for more information about how the conversion utility converts single-point t_{PD} and minimum t_{PD} assignments.

The conversion utility converts Classic Timing Analyzer I/O timing assignments to FPGA-centric SDC constraints. [Table 8-14](#) includes Classic Timing Analyzer timing assignments, the equivalent FPGA-centric SDC constraints, and recommended system-centric SDC constraints.

Table 8-14. Classic Timing Analyzer and TimeQuest Analyzer Equivalent Constraints

Classic Timing Analyzer Assignment	FPGA-Centric SDC	System-Centric SDC	More Information
t_{SU} Requirement (1)	set_max_delay	set_input_delay -max	page 8-30
t_H Requirement (1)	set_min_delay	set_input_delay -min	page 8-32
t_{CO} Requirement (2)	set_max_delay	set_output_delay -max	page 8-34
Minimum t_{CO} Requirement (2)	set_min_delay	set_output_delay -min	page 8-36

Notes to [Table 8-14](#):

- (1) Refer to " [\$t_{PD}\$ and Minimum \$t_{PD}\$ Requirement Conversion](#)" on [page 8-55](#) for more information about how the conversion utility converts this type of assignment.
 (2) Refer to " [\$t_{CO}\$ Requirement Conversion](#)" on [page 8-49](#) for more information about how the conversion utility converts this type of assignment.

The conversion utility can convert Classic Timing Analyzer I/O timing assignments only to the FPGA-centric constraints without additional information about your design. Making system-centric constraints requires information about the external circuitry interfacing with the FPGA such as external clocks, clock latency, board delay, and clocking exceptions. You cannot convert Classic Timing Analyzer timing assignments to system-centric constraints without that information. If you use the conversion utility, you can modify the SDC constraints to change the FPGA-centric constraints to system-centric constraints as appropriate.

PLL Phase Shift Conversion

The conversion utility does not account for PLL phase shifts when it converts values of the following FPGA-centric I/O timing assignments:

- **t_{SU} Requirement**
- **t_H Requirement**
- **t_{CO} Requirement**
- **Minimum t_{CO} Requirement**

If any of your paths go through PLLs with a phase shift, you must correct the converted values for those paths according to the formula in [Equation 8-13](#):

Equation 8-13.

$$\langle \text{correct value} \rangle = \langle \text{converted value} \rangle - \frac{(\langle \text{pll output period} \rangle \times \langle \text{phase shift} \rangle)}{360}$$

[Example 8-18](#) shows the incorrect conversion result for a t_{CO} assignment and how to correct it. For the example, assume the PLL output frequency is 200 MHz (period is 5 ns), the phase shift is 90 degrees, the t_{CO} **Requirement** value is 1 ns, and it is made to data[0]. The .qsf contains the following assignment:

Example 8-18. Assignment

```
set_instance_assignment -name TCO_REQUIREMENT -to data[0] 1.0
```

The conversion utility generates the SDC command shown in [Example 8-19](#).

Example 8-19. SDC Command

```
set_max_delay -from [get_registers *] -to [get_ports data[0]] 1.0
```

To correct the value, use the formula and values above, as shown in the following equation:

$$1.0 - \frac{(5 \times 90)}{360} = -0.25$$

Then, change the value so the SDC command so that it looks like [Example 8-20](#).

Example 8-20. SDC Command with Correct Values

```
set_max_delay -from [get_registers *] -to [get_ports data[0]] -0.25
```

t_{CO} Requirement Conversion

The conversion utility uses a special process to convert **t_{CO} Requirement** and **Minimum t_{CO} Requirement** assignments. In addition to the `set_max_delay` or `set_min_delay` commands, the conversion utility adds a `set_output_delay` constraint relative to a virtual clock named N/C (Not a Clock). It also creates the virtual clock named N/C with a period of 10 ns. Adding the virtual clock allows you to report timing on the output paths. Without the virtual clock N/C, the clock used for reporting would be blank. [Example 8-21](#) shows how the conversion utility converts a **t_{CO} Requirement** assignment of 5.0 ns to `data[0]`.

Example 8-21. Converting a t_{CO} Requirement Assignment of 5.0 ns to data[0]


```
set_max_delay -from [get_registers *] -to [get_ports data[0]] 5.0
set_output_delay -clock "N/C" 0 [get_ports data[0]]
```

Entity-Specific Assignments

Next, the conversion utility converts the entity-specific assignments listed in [Table 8-15](#) that exist in the **Timing Analyzer Settings** report panel. This usually occurs if you have any timing assignments in your Verilog HDL or VHDL source, which can include MegaCore function files. These entity-specific assignments cannot be automatically converted unless your project is compiled and a `\db` directory exists.

Table 8-15. Entity-Specific Timing Assignments

Classic Timing Analyzer Assignment	QSF Variable	More Information
Multicycle	MULTICYCLE	page 8-27
Source Multicycle	SRC_MULTICYCLE	
Multicycle Hold	HOLD_MULTICYCLE	
Source Multicycle Hold	SRC_HOLD_MULTICYCLE	
Setup Relationship	SETUP_RELATIONSHIP	page 8-24
Hold Relationship	HOLD_RELATIONSHIP	page 8-25
Cut Timing Path	CUT	page 8-40

 You must manually convert all other entity-specific timing assignments.

Paths Between Unrelated Clock Domains

The conversion utility can create exceptions that cut paths between unrelated clock domains, which matches the default behavior of the Classic Timing Analyzer. When **Cut paths between unrelated clock domains** is turned on, the conversion utility creates clock groups with the `set_clock_groups` command and uses the `-exclusive` option to cut paths between the clock groups.

Unsupported Instance Assignments

Finally, the conversion utility checks for the unsupported instance assignments listed in [Table 8-16](#) and warns you if any exist. Refer to the indicated page for information about each assignment.



You can manually convert some of the assignments to SDC constraints.

Table 8-16. Instance Timing Assignments

Assignment Name	QSF Variable	More Information
Inverted Clock	INVERTED_CLOCK	page 8-25
Maximum Clock Arrival Skew	MAX_CLOCK_ARRIVAL_SKEW	page 8-41
Maximum Data Arrival Skew	MAX_DATA_ARRIVAL_SKEW	page 8-41
Maximum Delay	MAX_DELAY	page 8-40
Minimum Delay	MIN_DELAY	page 8-41
Virtual Clock Reference	VIRTUAL_CLOCK_REFERENCE	page 8-26

Reviewing Conversion Results

You must review the messages that are generated during the conversion process, and review the `.sdc` file for correctness and completeness. Warning and critical warning messages identify significant differences between the Classic Timing Analyzer and TimeQuest analyzer behaviors. In some cases, warning messages indicate that the conversion utility ignored assignments because it could not determine the intended functionality of your design. You must add to or modify the SDC constraints as necessary based on your knowledge of the design.

The conversion utility creates an `.sdc` with the same name as your current revision, `<revision>.sdc`, and it overwrites any existing `<revision>.sdc`. If you use the conversion utility to create an `.sdc`, you should make additions or corrections in a separate `.sdc`, or a copy of the `.sdc` created by the conversion utility. That way, you can rerun the conversion utility later without overwriting your additions and changes. If you have constraints in multiple `.sdc` files, refer to [“Constraint File Priority” on page 8-7](#) to learn how to add constraints to your project.

Warning Messages

The conversion utility may generate any of the following warning messages. Refer to the information provided for each warning message to learn what to do in that instance.

Ignored QSF Variable `<assignment>`

The conversion utility ignored the specified assignment. Determine whether an equivalent constraint is necessary and manually add one if appropriate. Refer to [“Timing Assignment Conversion” on page 8-24](#) for information about conversions for all QSF timing assignments.

Global <name> = <value>

The conversion utility ignored the global assignment <name>. Manually add an equivalent constraint if appropriate. Refer to [“Unsupported Global Assignments” on page 8-43](#) for information about conversions for these assignments.

QSF: Expected <name> to be set to <expected value> but it is set to <actual value>

The behavior of the TimeQuest analyzer is closest to the Classic Timing Analyzer when the value for the specified assignment is the expected value. Because the actual assignment value is not the expected value in your project, the TimeQuest analyzer analysis may be different from the Classic Timing Analyzer analysis. Refer to [“Recommended Global Assignments” on page 8-43](#) for an explanation about the indicated QSF variable names.

QSF: Found Global f_{MAX} Requirement. Translation will be done using derive_clocks

Your design includes a global f_{MAX} requirement, and the requirement is converted to the derive_clocks command. Refer to [“Default Required \$f_{MAX}\$ Assignment” on page 8-26](#) for information about how to convert to an SDC constraint.

TAN Report Database not found. HDL based assignments will not be migrated

You did not analyze your design with the Classic Timing Analyzer before running the conversion utility. As a result, the conversion utility did not convert any timing assignments in your HDL source code to SDC constraints. If you have timing assignments in your HDL source code, you must find and convert them manually, or analyze your design with the Classic Timing Analyzer and rerun the conversion utility.

Ignored Entity Assignment (Entity <entity>): <variable> = <value> -from <from> -to <to>

The conversion utility ignored the specified entity assignment because the utility cannot automatically convert the assignment. [Table 8-15 on page 8-49](#) lists the entity-specific assignments the script can convert automatically.

Refer to [“Timing Assignment Conversion” on page 8-24](#) for information about how to convert the entity assignment manually.

Ignoring OFFSET_FROM_BASE_CLOCK assignment for clock <clock>

In some cases, this assignment is used to work around a limitation in how the Classic Timing Analyzer handles some forms of clock inversion. The conversion script ignores the assignment because it cannot determine whether the assignment is used as a workaround. Review your clock setting and add the assignment in your .sdc if appropriate. Refer to [“Clock Offset” on page 8-10](#) for more information about converting clock offset.

Clock <clock> has no FMAX_REQUIREMENT - No clock was generated

The conversion utility did not convert the clock named <clock> because it has no f_{MAX} requirement. You should add a clock constraint with an appropriate period to your .sdc.

No Clock Settings defined in .qsf

If your .qsf has no clock settings, ignore this message. You must add clock constraints in your .sdc manually.

Clocks

Ensure that the conversion utility converted all clock assignments correctly. Run the `report_clocks` command, or double-click **Report Clocks** in the Tasks pane in the TimeQuest analyzer GUI. Make sure that the right number of clocks is reported. If any clock constraints are missing, you must add them manually with the appropriate SDC commands (`create_clock` or `create_generated_clock`). Confirm that each option for each clock is correct.

The TimeQuest analyzer can create more clocks, such as:

- `derive_clocks` selecting ripple clocks
- `derive_pll_clocks`, adding
 - Extra clocks for PLL switchover
 - Extra clocks for LVDS pulse-generated clocks (`~load_reg`)

Clock Transfers

After you confirm that all clock assignments are correct, run the `report_clock_transfers` command, or double-click **Report Clock Transfers** in the Tasks pane in the TimeQuest analyzer GUI. The command generates a summary table with the number of paths between each clock domain. If the number of cross-clock domain paths seems high, remember that all clock domains are related in the TimeQuest analyzer. You must cut unrelated clock domains. Refer to “[Related and Unrelated Clocks](#)” on page 8–10 for information about how to cut unrelated clock domains.

Path Details

If you have unexpected differences between the Classic Timing Analyzer and the TimeQuest analyzer on some paths, follow these steps to identify the cause of the difference.

1. Report timing on the path in the TimeQuest analyzer.
2. Compare slack values.
3. Compare source and destination clocks.
4. Compare the launch/latch times in the TimeQuest analyzer to the setup/hold relationship in the Classic Timing Analyzer. The times are absolute in the TimeQuest analyzer and relative in the Classic Timing Analyzer.
5. Compare clock latency values.

Unconstrained Paths

Next, run the `report_ucp` command, or double-click **Report Unconstrained Paths** in the Tasks pane in the TimeQuest analyzer GUI. This command generates a series of reports that detail any unconstrained paths in your design. If your design was completely constrained in the Classic Timing Analyzer, but there are unconstrained paths in the TimeQuest analyzer, some assignments may not have been converted properly. Also, some of the assignments could be ambiguous. The TimeQuest analyzer analyzes more paths than the Classic Timing Analyzer does, so any unconstrained paths might be paths you could not constrain in the Classic Timing Analyzer.

Bus Names

If your design includes Classic Timing Analyzer timing assignments to buses, and the bus names do not include square brackets enclosing an asterisk, such as: `address[*]`, you should review the SDC constraints to ensure the conversion is correct. Refer to [“Bus Name Format” on page 8-6](#) for more information.

Other

Review the notes listed in [“Conversion Utility” on page 8-55](#).

Rerunning the Conversion Utility

You can force the conversion utility to run even if it can find an `.sdc` according to the priority described in [“Constraint File Priority” on page 8-7](#). Any method described in [“Create SDC Constraints from Existing Timing Assignments” on page 8-2](#) forces the conversion utility to run even if it can find an `.sdc`.

Notes

This section describes notes for the TimeQuest analyzer.

Output Pin Load Assignments

The TimeQuest analyzer takes **Output Pin Load** values into account when it analyzes your design. If you change **Output Pin Load** assignments and do not recompile before you analyze timing, you must use the `-force_dat` option when you create the timing netlist. Type the following command at the Tcl console of the TimeQuest analyzer:

```
create_timing_netlist -force_dat ←
```

If you change **Output Pin Load** assignments and recompile before you analyze timing, do not use the `-force_dat` option 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.

Also note that the SDC `set_output_load` command is not supported, so you must make all output load assignments in the `.qsf`.

Constraint Target Types

The TimeQuest analyzer supports mixed exception types; you can apply an exception of any clock/node combination.

DDR Constraints with the DDR Timing Wizard

The DDR Timing Wizard creates an `.sdc` that contains constraints for a DDR interface. You can use that `.sdc` with the TimeQuest analyzer to analyze only the DDR interface part of a design.

You should use the `.sdc` created by DDR Timing Wizard for constraining a DDR interface in the TimeQuest analyzer. Additionally, your `.qsf` should not contain timing assignments for the DDR interface if you plan to use the conversion utility to create an `.sdc`. You should run the conversion utility before you use DDR Timing Wizard, and you should choose not to apply assignments to the `.qsf`.

However, if you used DDR Timing Wizard and chose to apply assignments to a `.qsf`, before you used the conversion utility, you should remove the DDR Timing Wizard-generated QSF timing assignments and rerun the conversion utility. The conversion utility creates some incompatible SDC constraints from the DDR Timing Wizard QSF assignments.

Unsupported SDC Features

Some SDC commands and features are not supported by the current version of the TimeQuest analyzer, including the following commands and features:

- The `get_designs` command, because the Quartus II software supports a single design, so this command is not necessary
- True latch analysis with time-borrowing feature; it can, however, convert latches to negative-edge-triggered registers
- The case analysis feature
- Loads, clock transitions, input transitions, and other features

Constraint Passing and Optimization

The Quartus II software can read constraints generated by other EDA software, and write constraints to be read by other EDA software.

Other synthesis software can generate constraints that target the `.qsf`. If you change timing constraints in synthesis software after creating an `.sdc` for the TimeQuest analyzer, you must update the SDC constraints. You can use the conversion utility, or update the `.sdc` manually.

If you use physical synthesis with the TimeQuest analyzer, the design may have lower performance.

Clock Network Delay Reporting

The TimeQuest analyzer reports delay on the clock network by node-to-node segments; the Classic Timing Analyzer reports delay on the clock network as a single number, rather than node-to-node segments

Project Management

If you use the `project_open` Tcl command in the TimeQuest analyzer to open a project compiled with an earlier version of the Quartus II software, the TimeQuest analyzer overwrites the compilation results (`\db` folder) without warning. Opening a project any other way results in a warning, and you can choose not to open the project.

Conversion Utility

This section describes the notes for the QSF assignment to SDC constraint conversion utility.

t_{PD} and Minimum t_{PD} Requirement Conversion


The conversion utility treats the targets of single-point t_{PD} and minimum t_{PD} assignments as device outputs. It does not correctly convert targets of single-point t_{PD} and minimum t_{PD} assignments that are device inputs. The following QSF assignment applies to an a device input named `d_in`:

```
set_instance_assignment -name TPD_REQUIREMENT -to d_in "3 ns"
```

The conversion utility creates the following SDC command, regardless of whether `d_in` is a device input or device output:

```
set_max_delay "3 ns" -from [get_ports *] -to [get_ports d_in]
```


You must update any incorrect SDC constraints manually.


 For more detailed information about the features and capabilities of the TimeQuest analyzer, refer to the *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

Document Revision History

Table 8-17. Document Revision History

Date	Version	Changes
December 2010	10.1.0	<ul style="list-style-type: none"> ■ Changed to new document template. ■ Removed deprecated Classic Timing Analyzer features. ■ Minor updates to content.
July 2010	10.0.0	Minor updates to content.
November 2009	9.1.0	No change to content.
March 2009	9.0.0	This was chapter 8 in version 8.1.
November 2008	8.1.0	Changed to 8-1/2 x 11 page size. No change to content.

 For previous versions of the *Quartus II Handbook*, refer to the [Quartus II Handbook Archive](#).

 Take an [online survey](#) to provide feedback about this handbook chapter.

