

Introduction

Static timing analysis is a method for analyzing, debugging, and validating the timing performance of a design. The classic timing analyzer analyzes the delay of every design path and analyzes all timing requirements to ensure correct circuit operation. Static timing analysis, used in conjunction with functional simulation, allows you to verify overall design operation.



For information about switching to the Quartus® II TimeQuest Timing Analyzer, refer to the *Switching to the Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*.

As part of the compilation flow, the Quartus II software automatically performs a static timing analysis so that you do not need to launch a separate timing analysis tool. The Quartus II Classic Timing Analyzer checks every path in the design against your timing constraints for timing violations and reports results in the Timing Analysis reports, giving you immediate access to the data.

This chapter assumes you have some Tcl expertise; Tcl commands are used throughout this chapter to describe alternative methods for making timing analysis assignments. Refer to “*Timing Analysis Using the Quartus II GUI*” on page 10–34 for GUI-equivalent timing constraints.

This chapter details the following aspects of timing analysis:

- “Timing Analysis Tool Setup” on page 10–2
- “Static Timing Analysis Overview” on page 10–2
- “Clock Settings” on page 10–7
- “Clock Types” on page 10–8
- “Clock Uncertainty” on page 10–10
- “Clock Latency” on page 10–11
- “Timing Exceptions” on page 10–13
- “I/O Analysis” on page 10–21
- “Asynchronous Paths” on page 10–24
- “Skew Management” on page 10–28
- “Generating Timing Analysis Reports with report_timing” on page 10–30
- “Other Timing Analyzer Features” on page 10–31
- “Timing Analysis Using the Quartus II GUI” on page 10–34
- “Scripting Support” on page 10–38
- “MAX+PLUS II Timing Analysis Methodology” on page 10–43

Timing Analysis Tool Setup

The Quartus II software version 6.0 and above supports two static timing analysis tools namely, the classic timing analyzer and the Quartus II TimeQuest Timing Analyzer. Use the **Timing Analysis** option under the Settings menu to set the Timing Analyzer that is used in the compilation process.



Arria® GX is not supported by the Quartus II Classic Timing Analyzer. To perform a static timing analysis for Arria GX, the Quartus II TimeQuest Timing Analyzer must be enabled.

The following steps set the classic timing analyzer as the default timing analysis tool in the Quartus II software.

1. On the Assignments menu, click **Settings**. The **Settings** dialog box appears.
2. In the **Category** list, click the “+” icon to expand **Timing Analysis Settings** and select the **Use Classic Timing Analyzer during compilation** radio button.



Refer to the *Quartus II TimeQuest Timing Analyzer* chapter of the *Quartus II Handbook* for more information about the Quartus II TimeQuest Timing Analyzer.

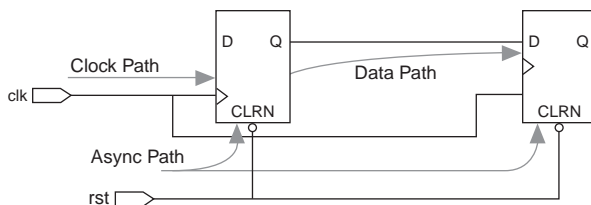
Static Timing Analysis Overview

This section provides information about static timing analysis concepts used throughout this chapter and used by the Quartus II Classic Timing Analyzer. A complete understanding of the concepts presented in this section allows you to take advantage of the powerful static timing analysis features available in the Quartus II software.

Various paths exist within any given design which connect design elements together, including the path from an output of a register to the input of another register. Timing paths play a significant role during a static timing analysis. Understanding the types of timing paths is important for timing closure and optimization. Some of the commonly analyzed paths are described in this section and are shown in Figure 10-1.

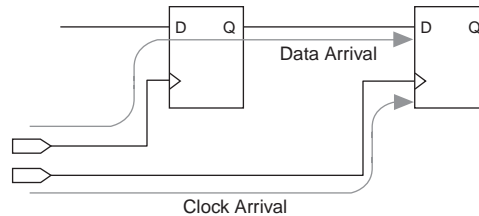
- *Clock paths*—Clock paths are the paths from device pins or internally generated clocks (nodes designated as a clock via a clock setting) to the clock ports of sequential elements such as registers.
- *Data paths*—Data paths are the paths from the data output port of a sequential element to the data input port of another sequential element.
- *Asynchronous paths*—Asynchronous paths are paths from a node to the asynchronous set or clear port of a sequential element.

Figure 10-1. Path Types



After the path types are identified, the classic timing analyzer computes data and clock arrival times for all valid register-to-register paths. Data arrival time is the delay from the source clock to the destination register. The Quartus II Classic Timing Analyzer calculates this delay by adding the clock path delay to the source register, the micro clock-to-out (t_{CO}) of the source register, and the data path delay from the source register to the destination register. Clock arrival time is the delay from the destination clock node to the destination register. **Figure 10-2** shows a data arrival path and a clock arrival path.

Figure 10-2. Data Arrival and Clock Arrival

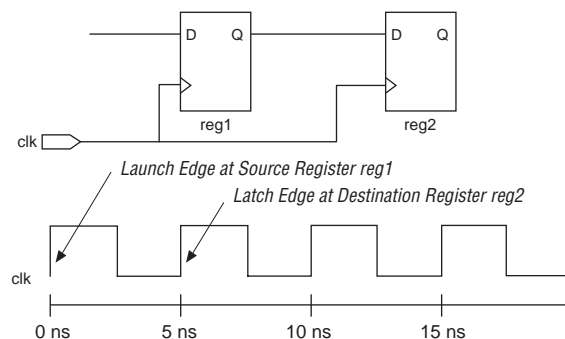


In addition to identifying various paths within a design, the Quartus II Classic Timing Analyzer analyzes clock characteristics to compute the worst-case requirement between any two registers in a single register-to-register path. You must use timing constraints to specify the characteristics of all clock signals in the design before this analysis occurs.

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

Figure 10-3 shows a single-cycle system that uses consecutive clock edges to transmit and capture data, a register-to-register path, and the corresponding launch and latch edges timing diagram. In this example, the launch edge sends the data out of register reg1 at 0 ns, and register reg2 latch edge captures the data at 5 ns.

Figure 10-3. Launch Edge and Latch Edge



By analyzing specific paths relative to the launch and latch edges, the Quartus II Classic Timing Analyzer performs clock setup and clock hold checks, validating them against your timing assignments.

Clock Analysis

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

Clock Setup Check

To determine if a design meets performance, the Quartus II Classic Timing Analyzer calculates clock timing, timing requirements, and timing exceptions to perform a clock setup check at each destination register based on the source and destination clocks and timing constraints, or exceptions that are applicable to those paths. A clock setup check ensures that data launched by a source register is latched correctly by the destination register. To perform a clock setup check, the Quartus II Classic Timing Analyzer determines the clock arrival time and data arrival time at the destination register by using the longest path for the data arrival time and the shortest path for the clock arrival time. The Quartus II Classic Timing Analyzer then checks that the difference is greater than or equal to the micro setup (t_{SU}) of the destination register as shown in [Equation 10-1](#).

Equation 10-1.

$$\text{Clock Arrival Time} - \text{Data Arrival Time} \geq \text{micro } t_{SU}$$



By default, the Quartus II Classic Timing Analyzer assumes the launched and latched edges happen on consecutive active clock edges.

The results of clock setup checks are reported in terms of slack. Slack is the margin by which a timing requirement is met or not met. Positive slack indicates the margin by which a requirement is met, and negative slack indicates the margin by which a requirement is not met. The Quartus II Classic Timing Analyzer determines clock setup slack using [Equation 10-2](#) through [Equation 10-5](#).

Equation 10-2.

$$\text{Clock Setup Slack} = \text{Data Required Time} - \text{Data Arrival Time}$$

Equation 10-3.

$$\text{Data Required} = \text{Clock Arrival Time} - \text{micro } t_{SU} - \text{Setup Uncertainty}$$

Equation 10-4.

$$\text{Clock Arrival Time} = \text{Latch Edge} + \text{Shortest Clock Path to Destination Register}$$

Equation 10-5.

$$\begin{aligned} \text{Data Arrival Time} = & \text{Launch Edge} + \text{Longest Clock Path to Source Register} \\ & + \text{micro } t_{CO} + \text{Longest Data Delay} \end{aligned}$$

The Quartus II Classic Timing Analyzer reports clock setup slack using Equation 10-6 through Equation 10-9 (which are equivalent to Equation 10-2 through Equation 10-5).

Equation 10-6.

Clock Setup Slack = Largest Register-to-Register Requirement – Longest Register-to-Register Delay

Equation 10-7.

Largest Register-to-Register Requirement = Setup Relationship between Source and Destination
+ largest clock skew – micro t_{CO} of Source Register – micro t_{SU} of Destination Register

Equation 10-8.

Setup Relationship between Source & Destination Register =
Latch Edge – Launch Edge Setup Uncertainty

Equation 10-9.

Largest Clock Skew = Shortest Clock Path to Destination Register
– Longest Clock Path to Source Register

Both sets of equations can be used to determine the slack value of any path.

Clock Hold Check

To prevent hold violations, the Quartus II Classic Timing Analyzer calculates clock timing, timing requirements, and timing exceptions to perform a clock hold check at each destination register. A clock hold check ensures data launched from the source register is not captured by an active clock edge earlier than the setup latch edge, and that the destination register does not capture data launched from the next active launch edge. To perform a clock hold check, the Quartus II Classic Timing Analyzer determines the clock arrival time and data arrival time at the destination register using the shortest path for the data arrival time and the longest path for the clock arrival time. The Quartus II Classic Timing Analyzer checks that the difference is greater than or equal to the micro hold time (t_H) of the destination register, as shown in Equation 10-10.

Equation 10-10.

Data Arrival Time – Clock Arrival Time $\geq t_H$

The Quartus II Classic Timing Analyzer determines clock hold slack using Equation 10-11 through Equation 10-14.

Equation 10-11.

Clock Hold Slack = Data Arrival Time – Data Required Time

Equation 10-12.

$$\text{Data Required Time} = \text{Clock Arrival Time} + \text{micro } t_H + \text{Hold Uncertainty}$$

Equation 10-13.

$$\text{Clock Arrival Time} = \text{Latch Edge} + \text{Longest Clock Path to Destination Register}$$

Equation 10-14.

$$\begin{aligned} \text{Data Arrival Time} = & \text{Launch Edge} + \text{Shortest Clock Path to Source Register} \\ & + \text{micro } t_{CO} + \text{Shortest Data Delay} \end{aligned}$$

The Quartus II Classic Timing Analyzer reports clock hold slack using [Equation 10-15](#) through [Equation 10-18](#).

Equation 10-15.

$$\begin{aligned} \text{Clock Hold Slack} = & \text{Shortest Register-to-Register Delay} \\ & - \text{Smallest Register-to-Register Requirement} \end{aligned}$$

Equation 10-16.

$$\begin{aligned} \text{Smallest Register-to-Register Requirement} = & \text{Hold Relationship between Source \& Destination} + \\ & \text{Smallest Clock Skew} - \text{micro } t_{CO} \text{ of Source Register} + \text{micro } t_H \text{ of Destination Register} \end{aligned}$$

Equation 10-17.

$$\text{Hold Relationship between Source and Destination Register} = \text{Latch} - \text{Launch} + \text{Hold Uncertainty}$$

Equation 10-18.

$$\begin{aligned} \text{Smallest Clock Skew} = & \text{Longest Clock Path from Clock to Destination Register} \\ & - \text{Shortest Clock Path from Clock to Source Register} \end{aligned}$$

These equations can be used to determine the slack value of any path.

Multicycle Paths

Multicycle paths are data paths that require more than one clock cycle to latch data at the destination register. For example, a register may be required to capture data on every second or third rising clock edge. [Figure 10-4](#) shows an example of a multicycle path between a multiplier's input registers and output register where the destination latches data on every other clock edge.

Refer to ["Multicycle" on page 10-13](#) for more information about multicycle exceptions.

Figure 10-4. Example Diagram of a Multicycle Path

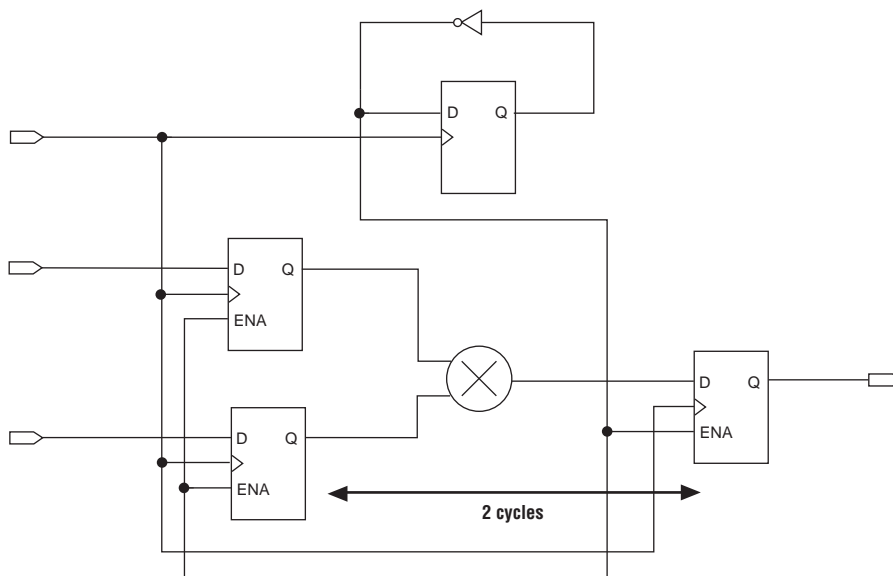


Figure 10-5 shows the default clock setup analysis launch and latch edges where multicycle assignment is equal to 1.

Figure 10-5. Default Clock Setup Analysis

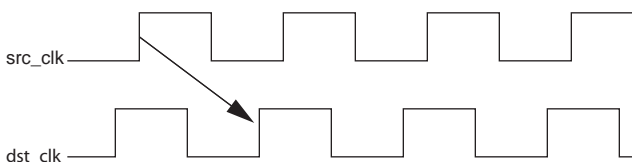
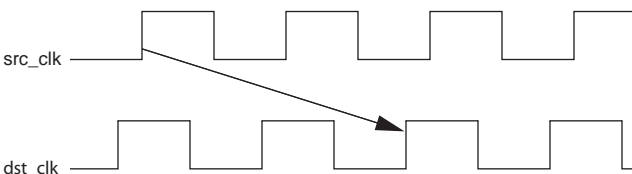


Figure 10-6 shows an analysis similar to Figure 10-5, but with a multicycle of 2.

Figure 10-6. Multicycle = 2 Clock Setup Analysis



Clock Settings

You can use individual and default clock settings to define the clocks in your design. You can base these clock settings on other clock settings already defined in your design.



To ensure the Quartus II Fitter achieves the desired performance requirements and the Quartus II Classic Timing Analyzer performs a thorough static timing analysis, you must specify all timing assignments prior to compiling the design.

Individual Clock Settings

Individual clock settings allow you to specify clock properties including performance requirements, offsets, duty cycles, and other properties for individual clock signals in your design.

You can define individual clock settings using the `create_base_clock` Tcl command. The following example defines an individual clock setting named `sys_clk` with a requirement of 100 MHz (10 ns), and assigns it to clock node `clk`.

```
create_base_clock -fmax 100MHz -target clk sys_clk
```

Default Clock Settings

You can assign a project-wide clock requirement to constrain all detected clocks in a design that do not have individual clock settings.

The `set_global_assignment -name FMAX_REQUIREMENT` Tcl command specifies a global default requirement assignment. The following example defines a 100 MHz default clock requirement:

```
set_global_assignment -name FMAX_REQUIREMENT "100.0 MHz"
```



For best placement and routing results, apply individual clock settings to all clocks in your design. All clocks adopting the default f_{MAX} are by default unrelated.

Clock Types

This section describes the types of clocks recognized by the Timing Analyzer:

- Base clocks
- Derived clocks
- Undefined clocks
- PLL clocks

Base Clocks

A base clock is independent of other clocks in a design. For example, a base clock is typically a clock signal driven directly by a device pin. A base clock is defined by individual clock settings, or automatically detected using the default clock setting.

You can use the `create_base_clock` Tcl command to define a base clock setting and assign the clock setting to a clock node. The following Tcl command creates a clock setting called `sys_clk` with a requirement of 5 ns (200 MHz) and applies the clock setting to clock node `main_clk`:

```
create_base_clock -fmax 5ns -target main_clk sys_clk
```

Derived Clocks

A derived clock is based on a previously defined base clock. For a derived clock, you can specify the phase shift, offset, multiplication and division factors, and duty cycle relative to the base clock.

You can use the `create_relative_clock` Tcl command to define and assign a derived clock setting. The following example creates a derived clock setting named `system_clockx2` that is twice as fast as the base clock `system_clock` applied to clock node `clkx2`.

```
create_relative_clock -base_clock system_clock -duty_cycle 50 \
-multiply 2 -target clkx2 system_clockx2
```

Undefined Clocks

The Quartus II Classic Timing Analyzer detects undeclared clocks in your design and displays a warning similar to the following:

```
Warning: Found pins functioning as undefined clocks and/or memory
enables
  Info: Assuming node "clk_src" is an undefined clock
  Info: Assuming node "clk_dst" is an undefined clock
```

The Quartus II Classic Timing Analyzer reports actual data delay for undefined clocks, but because no clock requirements exist for undefined clocks, the Quartus II Classic Timing Analyzer does not report slack for any register-to-register paths driven by an undefined clock.

PLL Clocks

Phase-locked loops (PLLs) are used for clock synthesis in Altera® devices. This device feature is configured and connected to your design using the `altpll` megafunction included with the Quartus II software. Using the MegaWizard™ Plug-In Manager, you can customize the input clock frequency, multiplication factors, division factors, and other parameters of the `altpll` megafunction.



For more information about using the PLL feature in your design, refer to the [ALTPLL Megafunction User Guide](#) or the handbook for the targeted device family.

For PLLs, the Quartus II Classic Timing Analyzer automatically creates derived clock settings based on the parameterization of the PLL and automatically creates a base clock setting for the input clock pin. For example, if the input clock frequency to a PLL is 100 MHz and the multiplication and division ratio is 5:2, the clock period of the PLL clock is set to 4.0 ns and is automatically calculated by the Quartus II Classic Timing Analyzer.

For the Stratix® and Cyclone® device families, you can override the PLL input clock frequency by applying a clock setting to the input clock pin of the PLL. For example, if the PLL input clock period is set to 10 ns (100 MHz) with a multiplication and division ratio of 5:2, but a clock setting of 20 ns (50 MHz) is applied to the input clock pin of the PLL, the setup relationship is 8.0 ns (125 MHz) and not 4.0 ns (250 MHz). The Quartus II Classic Timing Analyzer issues a message similar to the following:

```
Warning: ClockLock PLL "mypll_test:inst|altpll:altpll_component|_clk1"
input frequency requirement of 200.0 MHz overrides default required fmax
of 100.0 MHz -- Slack information will be reported
```



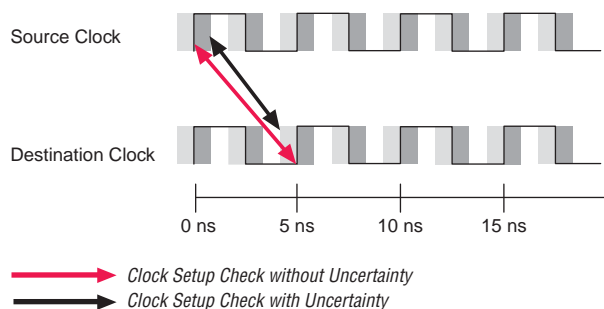
You cannot override the PLL output clock frequency with a clock setting in the Quartus II Classic Timing Analyzer.

Clock Uncertainty

You can use Clock Setup Uncertainty and Clock Hold Uncertainty assignments to model jitter, skew, or add a guard band associated with clock signals.

When a clock uncertainty assignment exists for a clock signal, the Timing Analyzer performs the most conservative setup and hold checks. For clock setup check, the setup uncertainty is subtracted from the data required time. Figure 10-7 shows an example of clock sources with a clock setup uncertainty applied.

Figure 10-7. Clock Setup Uncertainty

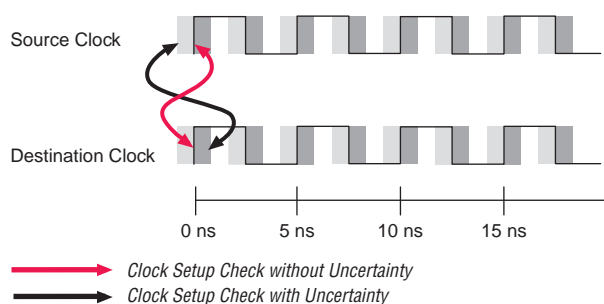


You can create clock uncertainty assignments using the Tcl command `set_clock_uncertainty`. The `set_clock_uncertainty` assignment used with the switch `-setup` specifies a clock setup uncertainty assignment. The following example creates a Clock Setup Uncertainty assignment with a value of 2 ns applied to clock signal `clk`:

```
set_clock_uncertainty -to clk -setup 2ns
```

For the clock hold check, the hold uncertainty is added to the data required time. Figure 10-8 shows an example of clock setup check with a clock setup uncertainty and clock hold uncertainty applied.

Figure 10-8. Clock Hold Uncertainty



You can use the `set_clock_uncertainty` Tcl command with the option `-hold` to specify a Clock Hold Uncertainty assignment. The following example creates a Clock Hold Uncertainty assignment with a value of 2 ns for clock signal `clk`.

```
set_clock_uncertainty -to clk -hold 2ns
```

You can also apply the clock uncertainty assignments between two clock sources. The following example creates a Clock Setup Uncertainty assignment for clock setup checks where `clk1` is the source clock and `clk2` is the destination clock:

```
set_clock_uncertainty -from clk1 -to clk2 -setup 2ns
```

Clock Latency

You can use clock latency assignments to model delays from the clock source. For example, you can use clock latency to model an external delay from an ideal clock source, such as an oscillator, to the clock pin or port of the device.

The Early Clock Latency assignment allows you to specify the shortest or earliest delay of the clock source. Conversely, the Late Clock Latency assignment allows you to specify the longest or latest delay of the clock source.

During setup analysis, the Quartus II Classic Timing Analyzer adds the Late Clock Latency assignment value to the source clock path delay and adds the Early Clock Latency assignment value to the destination clock path delay when determining clock skew for the path. During clock hold analysis, the Quartus II Classic Timing Analyzer adds the Early Clock Latency assignment value to the source clock path delay and adds the Late Clock Latency assignment value to the destination clock path delay when determining clock skew for the path.

The Early Clock Latency and Late Clock Latency assignments do not change the latch and launch edges defined by the clock setting and therefore does not change the setup or hold relationships between source and destination clocks. The clock latency assignments add only delay to the clock network and therefore only affects clock skew.

Figure 10-9 shows the clock edges used to calculate clock skew for a setup check when the Early Clock Latency and Late Clock Latency assignments are used.

Figure 10-9. Clock Setup Check Clock Skew

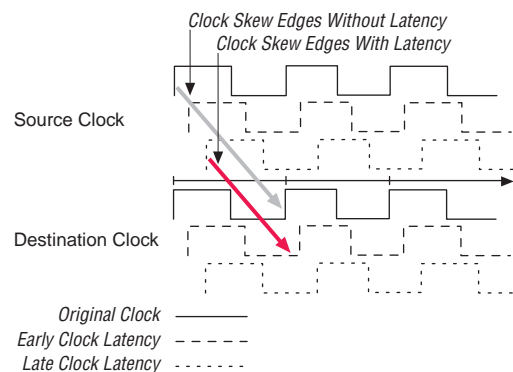
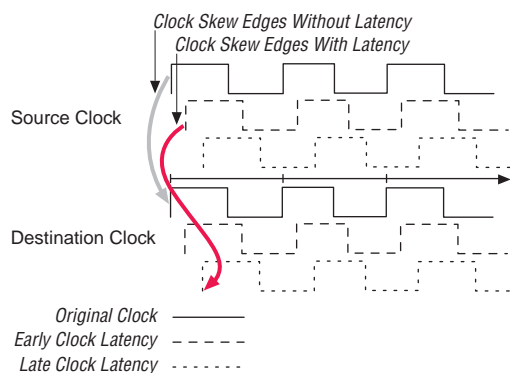



Figure 10-10 shows the clock edges used to calculate clock skew for a hold check when the Early Clock Latency and Late Clock Latency assignments are used.


Figure 10-10. Clock Hold Check Clock Skew

 The Quartus II Classic Timing Analyzer ignores clock latency if the clock signal at the source and destination registers are the same.

You can use the `set_clock_latency` Tcl command with the switches `-early` or `-late` to specify an Early Clock Latency assignment or Late Clock Latency assignment, respectively. [Example 10-1](#) specifies that the clock signal at `clk2` arrives as early as 1.8 ns and as late as 2.0 ns.

Example 10-1. Specifying Early or Late Clock Latency at `clk2`


```
set_clock_latency -early -to clk2 1.8ns
set_clock_latency -late -to clk2 2ns
```

 The early clock latency default value is the same as the late clock latency delay, and the late clock latency default value is the same as the early clock latency delay, if only one is specified.

The **Enable Clock Latency** option must be set to **ON** for the Quartus II Classic Timing Analyzer to analyze clock latency. When this option is set to **ON**, the Quartus II Classic Timing Analyzer reports clock latency as part of the clock skew calculation for either the source or destination clock path depending upon the analysis performed. To set the **Enable Clock Latency** option to **ON**, you can use the following Tcl command:

```
set_global_assignment -name ENABLE_CLOCK_LATENCY ON
```

When the **Enable Clock Latency** option is enabled, the Quartus II Classic Timing Analyzer automatically calculates latencies for derived clocks instead of automatically calculating offsets; for example, PLL compensation delays. These clock path delays are accounted for as clock skew instead of part of the setup or hold relationship as done with offsets.

 For more information about clock latency, refer to [AN 411: Understanding PLL Timing for Stratix II Devices](#).

Timing Exceptions

Timing exceptions allow you to modify the default behavior of the Quartus II Classic Timing Analyzer. This section describes the following timing exceptions:

- Multicycle
- Setup relationship and hold relationship
- Maximum delay and minimum delay
- False paths



Not all timing exceptions presented in this chapter are applicable to the HardCopy® II devices. If you are designing for the HardCopy II device family, refer to the *Timing Constraints for HardCopy II Devices* chapter in the *HardCopy II Handbook*.

Multicycle

By default, the Quartus II Classic Timing Analyzer performs a single-cycle analysis for all valid register-to-register paths in the design. Multicycle assignments specify the number of clock periods before a source register launches the data or a destination register latches the data. Multicycle assignments adjust the latch or launch edges, which relaxes the required clock setup check or clock hold check between the source and destination register pairs. You can specify multicycles separately for setup and hold, and multicycles can be based on the source clock or destination clock. Apply Multicycle exception to time groups, clock nodes, or common clock enables.

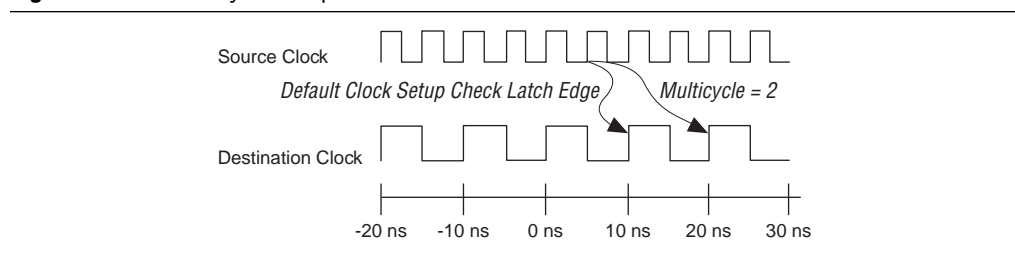
Destination Multicycle Setup Exception

A destination multicycle setup, referred to as a Multicycle exception, specifies the minimum number of clock cycles required before a register should latch a value. A Multicycle exception changes the latch edge by relaxing the required setup relationship. Figure 10-11 shows a timing diagram for a multicycle path that exists in a design with related clocks, and with the latch edge label for a clock setup check.



By default, the Multicycle exception value is 1.

Figure 10-11. Multicycle Setup



You can apply Multicycle exception between any two registers or between any two clock domains. Use the Tcl command `set_multicycle_assignment`, and the switch `-setup` and `-end`. For example, to apply a Multicycle exception of 2 between all registers clocked by source clock `clk_src`, and all registers clocked by destination clock `clk_dst`, enter the following Tcl command:

```
set_multicycle_assignment -setup -end -from clk_src -to clk_dst 2
```

To apply a Multicycle exception of 2 between the source register `reg1` and the destination register `reg2`, enter the following Tcl command:

```
set_multicycle_assignment -setup -end -from reg1 -to reg2 2
```

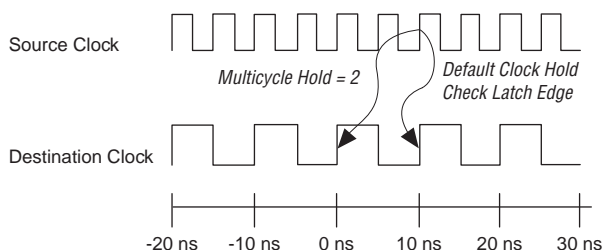
Destination Multicycle Hold Exception

A destination multicycle hold, referred to as a Multicycle Hold exception, modifies the latch edge used for a clock hold check for the register-to-register path based on the destination clock. A Multicycle Hold exception changes the latch edge by relaxing the required hold relationship. Figure 10-12 shows a timing diagram labeling the latching edge for a clock setup check.



If no Multicycle Hold value is specified, the Multicycle Hold value defaults to the value of the multicycle exception.

Figure 10-12. Multicycle Hold



You can create Multicycle Hold exceptions with the Tcl command `set_multicycle_assignment` and the switch `-hold` and `-end`. The following example specifies a Multicycle Hold exception of 3 from register `reg1` to register `reg2`:

```
set_multicycle_assignment -hold -end -from reg1 -to reg2 3
```

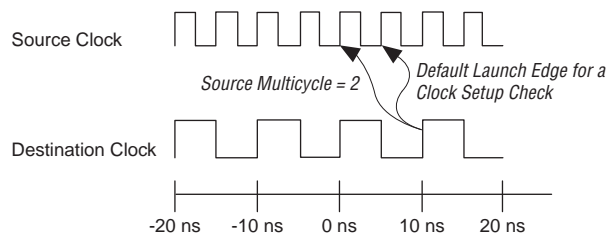
By default, the hold multicycle is set to equal that of the setup multicycle value along the same path. For example, if a setup multicycle of 2 has been applied to a register-to-register path without a separate hold multicycle, the hold multicycle value would be set to 2. The default hold multicycle value can also be changed to a value of 1. This forces all paths with a setup multicycle assignment to have a default hold multicycle of 1. To change the default hold multicycle value, in the **Settings** dialog box, click the **More Timing Settings** option.

If your design requires a hold multicycle value not equal to the setup multicycle or 1, you must explicitly apply a hold multicycle assignment to the path.

Source Multicycle Setup Exception

A source multicycle setup, referred to as Source Multicycle Setup exception, is used to extend the required delay by adjusting the source clock's launch edge rather than the destination clock's latch edge; for example, multicycle setup. Source Multicycle exceptions are useful when the source and destination registers are clocked by related clocks at different frequencies. Figure 10-13 shows an example of a Source Multicycle exception with the launch edge labeled for a clock setup check.

Figure 10-13. Source Multicycle



You can create Source Multicycle Setup exceptions with the Tcl command `set_multicycle_assignment` and the switches `-setup` and `-start`. The following example specifies a Source Multicycle exception of 3 from register `reg1` to register `reg2`:

```
set_multicycle_assignment -setup -start -from reg1 -to reg2 3
```

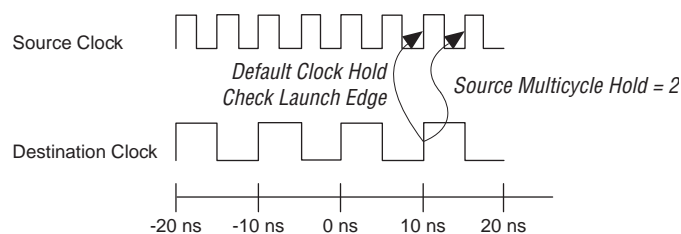
By default, the hold multicycle is set to equal that of the setup multicycle value along the same path. For example, if a setup multicycle of 2 has been applied to a register-to-register path without a separate hold multicycle, the hold multicycle value would be set to 2. The default hold multicycle value can also be changed to a value of 1. This forces all paths with a setup multicycle assignment to have a default hold multicycle of 1. To change the default hold multicycle value, in the **Settings** dialog box, click the **More Timing Settings** option.

If your design requires a hold multicycle value not equal to the setup multicycle or 1, you must explicitly apply a hold multicycle assignment to the path.

Source Multicycle Hold Exception

The Source Multicycle Hold exception modifies the latch edge used for a clock hold check for the register-to-register path based on the source clock. Source Multicycle Hold exceptions increase the required hold delay by adding source clock cycles. [Figure 10-14](#) shows an example of a source multicycle hold with launch edge labeled for a clock hold check.

Figure 10-14. Source Multicycle Hold



You can create Source Multicycle Hold exceptions with the Tcl command `set_multicycle_assignment` and the switch `-setup` and `-start`. The following example specifies a Source Multicycle Hold exception of 3 from register `reg1` to register `reg2`:

```
set_multicycle_assignment -hold -start -from reg1 -to reg2 3
```

Default Hold Multicycle

The Quartus II Classic Timing Analyzer sets the hold multicycle value to equal the multicycle value when a multicycle exception has been entered without a corresponding hold multicycle. You can change the behavior with the `DEFAULT_HOLD_MULTICYCLE` assignment. The value of the assignment can either be "ONE" or "SAME AS MULTICYCLE".

The assignment has the following syntax:

```
set_global_assignment -name DEFAULT_HOLD_MULTICYCLE "<value>"
```

Clock Enable Multicycle

For all enable-driven registers, the setup relationship or hold relationship can be modified with the Clock Enable Multicycle, Clock Enable Multicycle Hold, Clock Enable Source Multicycle, or Clock Enable Multicycle Source Hold.

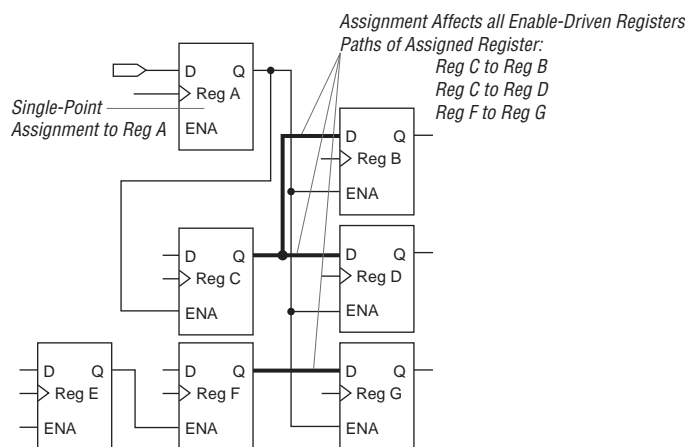
The **Clock Enable Multicycle** modifies the latching edge when a clock setup check is performed for all registers driven by the specified clock enables, and the **Clock Enable Multicycle Hold** modifies the latching edge when a clock hold check is performed for all registers driven by the specified clock enable. The **Clock Enable Source Multicycle** modifies the launching edge when a clock setup check is performed for all enabled driven registers, and the **Clock Enable Source Multicycle Hold** modifies the launching edge when a clock hold check is performed for all enabled driven registers.



Clock enable-based multicycle exceptions apply only to registers using dedicated clock enable circuitry. If the enable is synthesized into a logic cell; for example, due to logic prioritization, the multicycle does not apply.

The Clock Enable Multicycle, Clock Enable Multicycle Hold, Clock Enable Source Multicycle, and Clock Enable Multicycle Source Hold can be either a single-point or a point-to-point assignment. **Figure 10-15** shows an example of a single-point assignment. In this example, register Reg A has the single-point assignment applied. This has the affect of modifying a register-to-register latching edge whose enable port is driven by register Reg A. All register-to-register paths with enables driven by the single-point assignment are affected, even those driven by different clock sources.

Figure 10-15. Single-Point Clock Enable Multicycle



Point-to-point assignments apply to all paths where the source registers' enable ports are driven by the source (*from*) node and the destination registers' enable ports are driven by the destination (*to*) node. **Figure 10-16** shows an example of a point-to-point assignment made to different source and destination registers. In this example, register *Reg A* is specified as the source, and register *Reg B* is specified as the destination for the assignment. Only register-to-register paths that have their enables driven by the assigned point-to-point registers have their latching edges modified.

Figure 10-16. Different Source and Destination Point-to-Point Assignment Clock Enable Multicycle

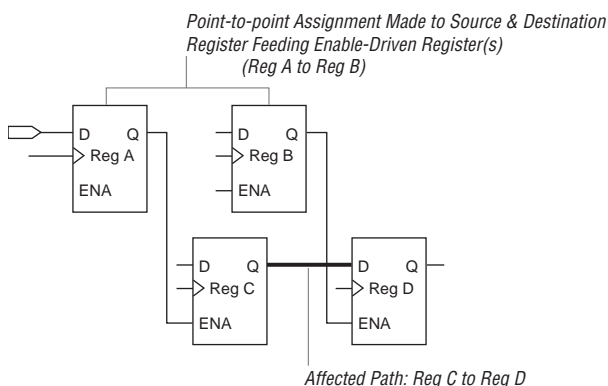
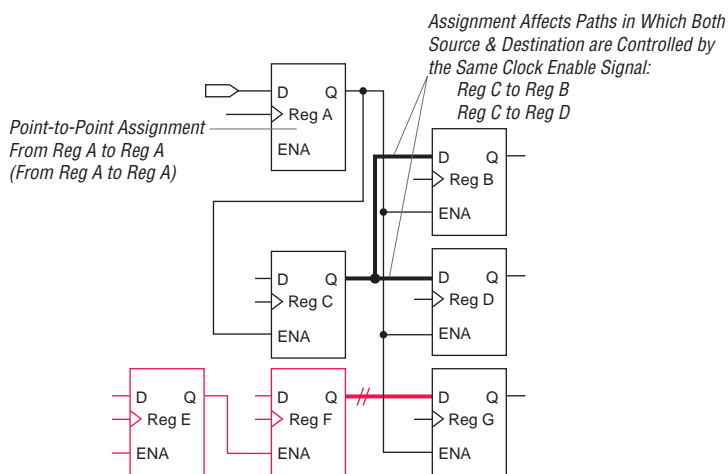


Figure 10-17 shows an example of a point-to-point assignment made to the same source and destination register. In this example, register *Reg A* has been specified as both the source and register for the assignment. Only register-to-register paths that have both the source-enable port and destination-enable port has the latching edge modified by the assigned point-to-point assignment.

Figure 10-17. Same Source and Destination Point-to-Point Assignment Clock Enable Multicycle



You can use the `set_instance_assignment -name CLOCK_ENABLE_MULTICYCLE` and `set_instance_assignment -name CLOCK_ENABLE_MULTICYCLE_HOLD Td` commands to specify either a Clock Enable Multicycle or a Clock Enable Multicycle Hold assignment, respectively. The following example specifies a single-point Clock Enable Multicycle assignment of 2 ns to `reg1`:

```
set_instance_assignment -name CLOCK_ENABLE_MULTICYCLE 2 -to reg1
```

The following example specifies a point-to-point Clock Enable Multicycle Hold assignment of 2 from register reg1 to register reg2:

```
set_instance_assignment -name CLOCK_ENABLE_MULTICYCLE_HOLD 2 \
-from reg1 -to reg2
```

You can use the `set_instance_assignment -name CLOCK_ENABLE_SOURCE_MULTICYCLE` and `set_instance_assignment -name CLOCK_ENABLE_MULTICYCLE_SOURCE_HOLD` Tcl commands to specify either a Clock Enable Multicycle or Clock Enable Multicycle Hold assignment, respectively. The following example specifies a single-point Clock Enable Multicycle assignment of 2 ns to reg1:

```
set_instance_assignment -name CLOCK_ENABLE_SOURCE_MULTICYCLE 2 \
-to reg1
```

The following example specifies a point-to-point Clock Enable Multicycle Hold assignment of 2 from register reg1 to register reg2:

```
set_instance_assignment -name \
CLOCK_ENABLE_SOURCE_MULTICYCLE_HOLD 2 -from reg1 -to reg2
```

Setup Relationship and Hold Relationship

By default, the Quartus II Classic Timing Analyzer determines all setup and hold relationships based on clock settings. The Setup Relationship and Hold Relationship exceptions allow you to override any default setup or hold relationships.

[Example 10-2](#) shows the path details of a register-to-register path that has a 10 ns clock setting applied to the clock signal driving the 2 registers.

Example 10-2. Default Setup Relationship with 10 ns Clock Setting

```
Info: Slack time is 9.405 ns for clock "data_clk" between source register "reg9" and
destination register "reg10"
Info: Fmax is restricted to 500.0 MHz due to tcl and tch limits
Info: + Largest register to register requirement is 9.816 ns
Info: + Setup relationship between source and destination is 10.000 ns
Info: + Latch edge is 10.000 ns
Info: - Launch edge is 0.000 ns
Info: + Largest clock skew is 0.000 ns
Info: - Micro clock to output delay of source is 0.094 ns
Info: - Micro setup delay of destination is 0.090 ns
Info: - Longest register to register delay is 0.411 ns
```

In [Example 10-3](#), a 15 ns Setup Relationship exception is applied to the register-to-register path, overriding the default 10 ns setup relationship.

Example 10-3. Setup Relationship Assignment of 15 ns

```
Info: Slack time is 14.405 ns for clock "data_clk" between source register "reg9" and
destination register "reg10"
Info: Fmax is restricted to 500.0 MHz due to tcl and tch limits
Info: + Largest register to register requirement is 14.816 ns
Info: + Setup relationship between source and destination is 15.000 ns
Info: Setup Relationship assignment value is 15.000 ns between source "reg9" and
destination "reg10"
Info: + Largest clock skew is 0.000 ns
Info: Total interconnect delay = 1.583 ns ( 51.31 % )
Info: - Micro clock to output delay of source is 0.094 ns
Info: - Micro setup delay of destination is 0.090 ns
Info: - Longest register to register delay is 0.411 ns
```

You can create a Setup Relationship exception with the Tcl command `set_instance_assignment -name SETUP_RELATIONSHIP`. The following example specifies a Setup Relationship exception of 5 ns from register `reg1` to register `reg2`:

```
set_instance_assignment -name SETUP_RELATIONSHIP 5ns -from reg1 \
-to reg2
```

You can use Hold Relationship exception to override the default hold relationship of any register-to-register paths.

You can use the `set_instance_assignment -name HOLD_RELATIONSHIP` Tcl command to specify a hold relationship assignment. The following example specifies a Hold Relationship exception of 1 ns from register `reg1` to register `reg2`:

```
set_instance_assignment -name HOLD_RELATIONSHIP 1ns -from reg1 \
-to reg2
```

Maximum Delay and Minimum Delay

You can use Maximum Delay and Minimum Delay assignments to specify delay requirements for pin-to-register, register-to-register, and register-to-pin paths. The Maximum Delay assignment overrides any setup relationship for any path. The Minimum Delay assignment overrides any hold relationship for any path.



The Quartus II Classic Timing Analyzer ignores the effects of clock skew when checking a design against Maximum Delay and Minimum Delay assignments.

You can use the `set_instance_assignment -name MAX_DELAY` and `set_instance_assignment -name MIN_DELAY` Tcl commands to specify a Maximum Delay assignment or a Minimum Delay assignment, respectively. The following example specifies a maximum delay of 2 ns between source register `reg1` and destination register `reg2`:

```
set_instance_assignment -name MAX_DELAY 2ns -from reg1 -to reg2
```

The following example specifies a minimum delay of 1 ns between input pin `data_in` to destination register `dst_reg`:

```
set_instance_assignment -name MIN_DELAY 1ns -from data_in -to dst_reg
```

False Paths

A false path is any path that is not relevant to a circuit's operation, such as test logic. There are several global assignments to cut different classes of paths, such as unrelated clock domains and paths through bidirectional pins, but you can also cut an individual timing path to a specific false path.

The Timing Analyzer provides the following three global options that allow you to remove false paths from your design:

- Cut off feedback from I/O pins
- Cut off read-during-write signal paths
- Cut paths between unrelated clock domains

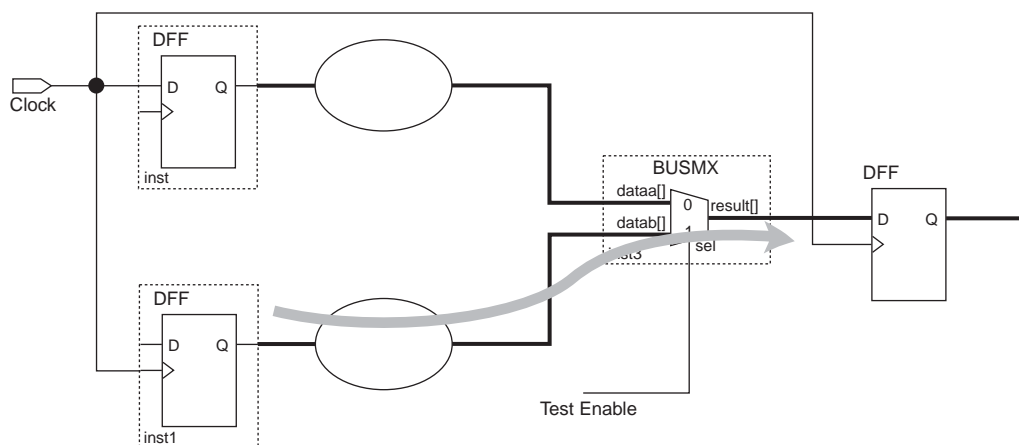
You can use the `set_global_assignment -name CUT_OFF_IO_PIN_FEEDBACK ON` Tcl command to cut the feedback path when a bidirectional I/O pin is connected directly or indirectly to both the input and output of a latch.

You can use the `set_global_assignment -name CUT_OFF_READ_DURING_WRITE_PATHS ON` Tcl command to cut the path from the write-enable register through memory element to a destination register.

You can use the `set_global_assignment -name CUT_OFF_PATHS_BETWEEN_CLOCK_DOMAINS ON` Tcl command to cut paths between register-to-register where the source and destination clocks are different.

You can use the `set_timing_cut_assignment` Tcl command to cut specific timing paths. In [Figure 10-18](#), the path from `inst1` through the multiplexer to `inst2` is used only for design testing. This false path is not required under normal operation and does not need to be analyzed during static timing analysis. [Figure 10-18](#) shows an example of a false path.

Figure 10-18. False Path Signal



To cut the timing path from source register `inst1` to destination register `inst2`, enter the following Tcl command:

```
set_timing_cut_assignment -from inst1 -to inst2
```

You can also use the `set_timing_cut_assignment` Tcl command as a single point assignment. When you use the single point assignment, all fanout of the node is cut. For example, the following Tcl command cuts all timing paths originating for node `src_reg`:

```
set_timing_cut_assignment -to src_reg
```

I/O Analysis

The I/O analysis performed by the Quartus II Classic Timing Analyzer ensures your Altera FPGA design meets all timing specifications for interfacing with external devices. This section describes assignments relevant to I/O analysis and other I/O analysis features and options available with the Quartus II Classic Timing Analyzer.

External Input Delay and Output Delay Assignments

External input and output delays represent delays from or to external devices or boards traces. You can make **Input Delay** and **Output Delay** assignments to ensure the Quartus II Classic Timing Analyzer can perform a full system analysis. By providing **Input Delays** and **Output Delays**, the Quartus II Classic Timing Analyzer is able to perform clock setup and clock hold checks for these paths. This also allows other timing assignments, such as multicycle or clock uncertainty, to be applied to input and output paths.



Do not combine individual or global t_{SU} , t_H , t_{PD} , t_{CO} , minimum t_{CO} , or minimum t_{PD} assignments with **Input Delay** or **Output Delay** assignments.

Input Delay Assignment

External input delays are specified with either Input Maximum Delay or Input Minimum Delay assignments. Make Input Maximum Delay assignments to specify the maximum delay of a signal from an external register to a specified input or bidirectional pin on the FPGA relative to a specified clock source. Make Input Minimum Delay assignments to specify the minimum delay of a signal from an external register to a specified input or bidirectional pin on the FPGA relative to a specified clock source.

When performing a clock setup check, the Quartus II Classic Timing Analyzer adds the Input Maximum Delay assignment value to the data arrival time (or subtracts the assignment value from the point-to-point requirement).

When performing a clock hold check, the Quartus II Classic Timing Analyzer adds the Input Minimum Delay assignment value to the data arrival time (or subtracts the assignment value from the point-to-point requirement).

The value of the input delay assignment usually represents the sum of the t_{CO} of the external device, the actual board delay to the input pin of the Altera device, and the board clock skew.



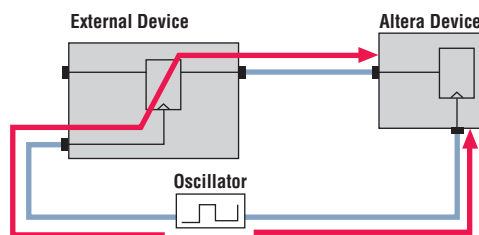
The **Input Minimum Delay** defaults to the **Input Maximum Delay** and the **Input Maximum Delay** defaults to the **Input Minimum Delay** if only one is specified.

For example, the Input Maximum Delay and Input Minimum Delay can be used to model the delay associated with an external device driving into an Altera FPGA. Figure 10-19 shows an example of the input delay path. For Figure 10-19, the Input Maximum Delay can be calculated as shown in Equation 10-19.

Equation 10-19.

$$\text{Input Maximum Delay} = \text{External Device Board Clock Path} + \text{External Device } t_{CO} + \text{External Device to Altera Device Board Delay} - \text{External Clock Path to Altera Device}$$

Figure 10-19. Input Delay



Use the Tcl command `set_input_delay` to specify an input delay. The following example specifies an **Input Maximum** Delay assignment of 1.5 ns from clock node `clk` to input pin `data_in`:

```
set_input_delay -clk_ref clk -to "data_in" -max 1.5ns
```

The following example specifies an Input Minimum Delay assignment of 1 ns from clock node `clk` to input pin `data_in`:

```
set_input_delay -clk_ref clk -to "data_in" -min 1ns
```

When using **Input Delay** assignments, specify a particular clock reference. The clock reference is the clock that feeds the external register's clock port that feeds the Altera device. This allows the Quartus II Classic Timing Analyzer to perform the proper analysis for the input path.



The t_{SU} , t_{H} , t_{PD} , and min t_{PD} timing paths reported for input pins, where input delay internal to the Altera FPGA assignments has been applied, include only the data delay from these pins and do not account for any clock setup relationships, clock hold relationships, or slack.

Output Delay Assignment

You can specify external output delays with either Output Maximum Delay or Output Minimum Delay assignments. Make Output Maximum Delay assignments to specify the maximum delay of a signal from the specified FPGA output pin to an external register, relative to a specified clock source. Make Output Minimum Delay assignments to specify the minimum delay of a signal from the specified FPGA output pin to an external register relative to a specified clock source.

When performing a clock setup check, the Quartus II Classic Timing Analyzer subtracts the Output Maximum Delay assignment value from the data required time (or subtracts the assignment value from the point-to-point requirement).

When performing a clock hold check, the Quartus II Classic Timing Analyzer subtracts the Output Minimum Delay assignment value from the data required time (or subtracts the assignment value from the point-to-point requirement).

The value of this assignment usually represents the sum of the t_{SU} of the external device, the actual board delay from the output pin of the Altera device, and the board clock skew.

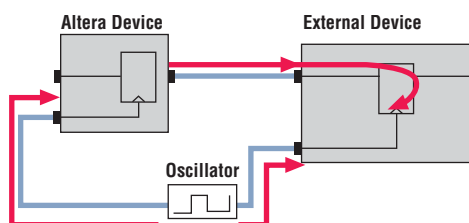


The Output Minimum Delay default value is the same as the Output Maximum Delay, and the Output Maximum Delay default value is the same as the Output Minimum Delay if only one is specified.

For example, use the Output Maximum Delay and Output Minimum Delay to model the delay associated with outputs for an Altera FPGA driving into an external device. Figure 10-20 shows an example of an output delay path. For Figure 10-20 the Output Maximum Delay can be calculated, as shown in Equation 10-20.

Equation 10-20.

$$\text{Output Maximum Delay} = \text{Altera Device to External Device Board Delay} + \text{External Device } t_{SU} + \text{External Clock Path to Altera Device} - \text{External Device Board Clock Path}$$

Figure 10-20. Output Delay

The Tcl command `set_output_delay` specifies an Output Delay assignment. The following example specifies an Output Maximum Delay assignment of 2 ns from clock `clk` to output pin `data_out`:

```
set_output_delay -clk_ref clk -to data_out -max 2ns
```

The following example specifies an Output Minimum Delay assignment of 1 ns from clock `clk` to output pin `data_out`:

```
set_output_delay -clk_ref clk -to data_out -min 1ns
```

When using output delay assignments, specify a specific clock reference. The clock reference is the clock that feeds the external register's clock port that is fed by the Altera device. This allows the Quartus II Classic Timing Analyzer to perform the correct static timing analysis on the output path.



The t_{CO} , minimum t_{CO} , t_{PD} , and minimum t_{PD} timing paths reported for output pins, where output delay assignments have been applied include only the data delay internal to the Altera FPGA to those pins, and do not account for any clock setup relationships, clock hold relationships, or slack.

Virtual Clocks

You can use virtual clocks to model clock signals outside of the Altera FPGA, that is, clocks that do not directly drive anything within the Altera FPGA. For example, you can use a virtual clock to model a clock signal feeding an external output register that feeds the Altera FPGA.

Using the `-virtual` option of the `create_base_clock` Tcl command specifies a virtual clock assignment.



Before you can use virtual clock for either an input or output delay assignment, the virtual clock must have the Virtual Clock Reference assignment enabled for the virtual clock setting.

The code in [Example 10-4](#) creates a virtual clock named `virt_clk`, with a 200 MHz requirement, and uses the virtual clock setting as the clock reference for the input delay assignment.

Example 10-4. Creating a Virtual Clock Named `virt_clk`

```
#create the virtual clock setting
create_base_clock -fmax 200MHz -virtual virt_clk

#enable the virtual clock reference for the virtual clock setting
set_instance_assignment -name VIRTUAL_CLOCK_REFERENCE On -to virt_clk

#use the virtual clock setting as the clock reference for the input delay assignment
set_input_delay -clk_ref virt_clk -to data_in -max 2ns
```

Asynchronous Paths

The Quartus II Classic Timing Analyzer can analyze asynchronous signals that connect to the clear, preset, or load ports of a register. This section explains how the Quartus II Classic Timing Analyzer analyzes asynchronous paths.

Recovery and Removal

Recovery time is the minimum length of time an asynchronous control signal; for example, clear and preset, must be stable before the active clock edge. Removal time is the minimum length of time an asynchronous control signal must be stable after the active clock edge. The Enable Recovery/Removal analysis option reports the results of recovery and removal checks for paths that end at an asynchronous clear, preset, or load signal of a register.

Enable the recovery and removal analysis with the following Tcl command:

```
set_global_assignment -name ENABLE_RECOVERY_REMOVAL_ANALYSIS ON
```

With this option enabled, the Quartus II Classic Timing Analyzer reports the result of the recovery analysis and removal analysis.



By default, the recovery and removal analysis is disabled. You should enable this option for all designs that contain asynchronous control signals.

Recovery Report

When you set `ENABLE_RECOVERY_REMOVAL_ANALYSIS` to **ON**, the Quartus II Classic Timing Analyzer determines the recovery time as the minimum amount of time required between an asynchronous control signal becoming inactive and the next active clock edge, compares this to your design, and reports the results as slack. The Recovery report alerts you to conditions where an active clock edge occurs too soon after the asynchronous input becomes inactive, rendering the register's data uncertain.

The recovery slack time calculation is similar to the calculation for clock setup slack, which is based on data arrival time and data required time except for asynchronous control signals. If the asynchronous control is registered, the Quartus II Classic Timing Analyzer calculates the recovery slack time using Equation 10-21 through Equation 10-23.

Equation 10-21.

$$\text{Recovery Slack Time} = \text{Data Required Time} - \text{Data Arrival Time}$$

Equation 10-22.

$$\text{Data Arrival Time} = \text{Launch Edge} + \text{Longest Clock Path to Source Register} + \text{micro } t_{CO} \text{ of Source Register} + \text{Longest Register-to-Register Delay}$$

Equation 10-23.

$$\text{Data Required Time} = \text{Latch Edge} + \text{Longest Clock Path to Source Register} + \text{micro } t_{SU} \text{ of Destination Register}$$

Example 10-5 shows recovery time as reported by the Timing Analyzer.

Example 10-5. Recovery Time Reporting for a Registered Asynchronous Reset Signal

```
Info: Slack time is 8.947 ns for clock "a_clk" between source register "async_reg1" and
destination register "reg_1"
Info: Requirement is of type recovery
Info: - Data arrival time is 4.028 ns
Info: + Launch edge is 0.000 ns
Info: + Longest clock path from clock "a_clk" to source register is 3.067 ns
Info: + Micro clock to output delay of source is 0.094 ns
Info: + Longest register to register delay is 0.867 ns
Info: + Data required time is 12.975 ns
Info: + Latch edge is 10.000 ns
Info: + Shortest clock path from clock "a_clk" to destination register is 3.065 ns
Info: - Micro setup delay of destination is 0.090 ns
```

If the asynchronous control is not registered, the Quartus II Classic Timing Analyzer uses Equation 10-24 through Equation 10-26 to calculate the recovery slack time.

Equation 10-24.

$$\text{Recovery Slack Time} = \text{Data Required Time} - \text{Data Arrival Time}$$

Equation 10-25.

$$\text{Data Arrival Time} = \text{Launch Edge} + \text{Maximum Input Delay} + \text{Maximum Pin-to-Register Delay}$$

Equation 10-26.

$$\text{Data Required Time} = \text{Latch Edge} + \text{Shortest Clock Path to Destination Register Delay} - \text{micro } t_{\text{SU}} \text{ of Destination Register}$$

Example 10-6 shows recovery time as reported by the Timing Analyzer.

Example 10-6. Recovery Time Reporting for a Non-Registered Asynchronous Reset Signal

```
Info: Slack time is 8.744 ns for clock "a_clk15" between source pin "a_arst2" and
destination register "inst5"
Info: Requirement is of type recovery
Info: - Data arrival time is 4.787 ns
Info: + Launch edge is 0.000 ns
Info: + Max Input delay of pin is 1.500 ns
Info: + Max pin to register delay is 3.287 ns
Info: + Data required time is 13.531 ns
Info: + Latch edge is 10.000 ns
Info: + Shortest clock path from clock "a_clk15" to destination register is 3.542 ns
Info: - Micro setup delay of destination is 0.011 ns
```



If the asynchronous reset signal is from a device pin, an Input Maximum Delay assignment must be made to the asynchronous reset pin for the Quartus II Classic Timing Analyzer to perform recovery analysis on that path.

Removal Report

When you set `ENABLE_RECOVERY_REMOVAL_ANALYSIS` to **ON**, the Quartus II Classic Timing Analyzer determines the removal time as the minimum amount of time required between an active clock edge that occurs while an asynchronous input is active, and the deassertion of the asynchronous control signal. The Quartus II Classic Timing Analyzer then compares this to your design and reports the results as slack. The Removal report alerts you to a condition in which an asynchronous input signal goes inactive too soon after a clock edge, thus rendering the register's data uncertain.

The removal time slack calculation is similar to the one used to calculate clock hold slack, which is based on data arrival time and data required time except for asynchronous control signals. If the asynchronous control is registered, the Quartus II Classic Timing Analyzer uses Equation 10-27 through Equation 10-29 to calculate the removal slack time.

Equation 10-27.

$$\text{Removal Slack Time} = \text{Data Arrival Time} - \text{Data Required Time}$$

Equation 10-28.

$$\text{Data Arrival Time} = \text{Launch Edge} + \text{Shortest Clock Path from Source Register Delay} + \text{micro } t_{\text{CO}} \text{ of Source Register} + \text{Shortest Register-to-Register Delay}$$

Equation 10-29.

$$\text{Data Required Time} = \text{Latch Edge} + \text{Longest Clock Path to Destination Register Delay} + \text{micro } t_{\text{H}} \text{ of Destination Register}$$

Example 10-7 shows removal time as reported by the Quartus II Classic Timing Analyzer.

Example 10-7. Removal Time Reporting for a Registered Asynchronous Reset Signal

```
Info: Minimum slack time is 814 ps for clock "a_clk" between source register "async_reg1"
and destination register "reg_1"
Info: Requirement is of type removal
Info: + Data arrival time is 4.028 ns
      Info: + Launch edge is 0.000 ns
      Info: + Shortest clock path from clock "a_clk" to source register is 3.067 ns
      Info: + Micro clock to output delay of source is 0.094 ns
      Info: + Shortest register to register delay is 0.867 ns
Info: - Data required time is 3.214 ns
      Info: + Latch edge is 0.000 ns
      Info: + Longest clock path from clock "a_clk" to destination register is 3.065 ns
      Info: + Micro hold delay of destination is 0.149 ns
```

If the asynchronous control is not registered, the Quartus II Classic Timing Analyzer uses [Equation 10-30](#) through [Equation 10-32](#) to calculate the removal slack time.

Equation 10-30.

$$\text{Removal Slack Time} = \text{Data Arrival Time} - \text{Data Required Time}$$

Equation 10-31.

$$\text{Data Arrival Time} = \text{Launch Edge} + \text{Input Minimum Delay of Pin} + \text{Minimum Pin-to-Register Delay}$$

Equation 10-32.

$$\text{Data Required Time} = \text{Latch Edge} + \text{Longest Clock Path to Destination Register Delay} + \text{micro } t_{\text{H}} \text{ of Destination Register}$$

Example 10-8 shows removal time as reported by the Quartus II Classic Timing Analyzer.

Example 10-8. Removal Time Reporting for a Non-Registered Asynchronous Reset Signal

```

Info: Minimum slack time is 1.131 ns for clock "a_clk15" between source pin "a_arst2"
and destination register "inst5"
    Info: Requirement is of type removal
    Info: + Data arrival time is 4.787 ns
Info: + Launch edge is 0.000 ns
Info: + Min Input delay of pin is 1.500 ns
Info: + Min pin to register delay is 3.287 ns
    Info: - Data required time is 3.656 ns
Info: + Latch edge is 0.000 ns
Info: + Longest clock path from clock "a_clk15" to destination register is 3.542 ns
    Info: + Micro hold delay of destination is 0.114 ns

```



If the asynchronous reset signal is from a device pin, an Input Minimum Delay assignment must be made to the asynchronous reset pin for the Quartus II Classic Timing Analyzer to perform a removal analysis on this path.

Skew Management

Clock skew is the difference in the arrival times of a clock signal at two different registers, which can be caused by path length differences between two clock paths, or by using gated or rippled clocks. As clock periods become shorter and shorter, the skew between data arrival times and clock arrival times becomes more significant. The Quartus II Classic Timing Analyzer provides two assignments for analyzing and constraining skew for data and clock signals.

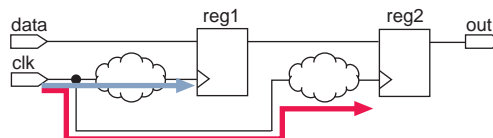
Maximum Clock Arrival Skew

Make Maximum Clock Arrival Skew assignments to specify the maximum allowable clock arrival skew between a clock signal and various destination registers. The Quartus II Classic Timing Analyzer compares the longest clock path to the registers' clock port and the shortest clock path to the registers' clock port to determine if your maximum clock arrival skew is achieved. Maximum clock arrival skew is calculated using [Equation 10-33](#).

Equation 10-33.

$$\text{Maximum Clock Arrival Skew} = \text{Longest Clock Path} - \text{Shortest Clock Path}$$

For example, if the delay from clock pin `clk` to the clock port of register `reg1` is 1.0 ns, and the delay from clock pin `clk` to the clock port of register `reg2` is 3.0 ns, as shown in [Figure 10-21](#), the Quartus II Classic Timing Analyzer provides a clock skew slack time of 2.0 ns.

Figure 10-21. Clock Arrival Paths


 You should apply the Maximum Clock Arrival Skew assignment to a clock node and a group of registers. When you make a Maximum Clock Arrival Skew assignment, the Fitter attempts to satisfy the skew requirement.

You can use the `set_instance_assignment -name max_clock_arrival_skew` Tcl command to specify a Maximum Clock Arrival Skew assignment. The following example specifies a maximum clock arrival skew of 1 ns from clock signal `clk` to the bank of registers matching `reg*`:

```
set_instance_assignment -name max_clock_arrival_skew 1ns -from clk \
-to reg*
```

Maximum Data Arrival Skew

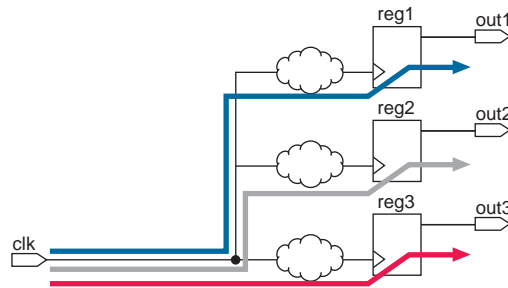
Make Maximum Data Arrival Skew assignments to specify the maximum allowable data arrival skew to various destination registers or pins. The Quartus II Classic Timing Analyzer compares the longest data arrival path to the shortest data arrival path to determine if your specified maximum data arrival skew is achieved. Maximum data arrival skew is calculated using [Equation 10-34](#).

Equation 10-34.

$$\text{Maximum Data Arrival Skew} = \text{Longest Data Arrival Path} - \text{Shortest Data Arrival Path}$$

For example, if the data arrival time to output pin `out1` is 2.0 ns, the data arrival time to output pin `out2` is 1.5 ns, and the data arrival time to output pin `out3` is 1.0 ns, as shown in [Figure 10-22](#), the Quartus II Classic Timing Analyzer provides a maximum data arrival skew slack time of 1.0 ns.

Figure 10-22. Data Arrival Paths



 When you make a Maximum Data Arrival Skew assignment, the Fitter attempts to satisfy the skew requirement.

You can use the `set_instance_assignment -name max_data_arrival_skew` Tcl command to specify a maximum data arrival skew value. The following example specifies a maximum data arrival skew of 1 ns from clock signal `clk` to the bank of output pins `dout`:

```
set_instance_assignment -name max_data_arrival_skew 1ns -from clk \
-to dout[*]
```

Generating Timing Analysis Reports with report_timing

The Quartus II Classic Timing Analyzer includes the `report_timing` Tcl command for generating text-based timing analysis reports. You can customize the output of `report_timing` using multiple switches that allow the generation of both detailed and general timing reports on any path in the design.



The `report_timing` Tcl command is available in the `quartus_tan` executable.

Prior to using the `report_timing` Tcl command, you must open a Quartus II project and create a timing netlist. For example, the following two Tcl commands accomplish this:

```
project_open my_project
create_timing_netlist
```

The `report_timing` Tcl command provides `-from` and `-to` switches for filtering specific source and destination nodes. For example, the following `report_timing` Tcl command reports all clock setup paths, with the switch `-clock_setup`, between registers `src_reg*` and `dst_reg*`. The `-npaths 20` switch limits the report to 20 paths.

```
report_timing -clock_setup -from src_reg* -to dst_reg* -npaths 20
```

The switches `-clock_filter` and `-src_clock_filter` are also available for filtering based on specific clock sources. For example, the following `report_timing` Tcl command reports all clock setup paths where the destination registers are clocked by `clk`:

```
report_timing -clock_setup -clock_filter clk
```

The following example reports clock setup paths where the destination registers are clocked by `clk`, and the source registers are clocked by `src_clk`.

```
report_timing -clock_setup -clock_filter clk -src_clock_filter src_clk
```

Example 10-9 is an example script that can be sourced by the `quartus_tan` executable:

Example 10-9. Source for the quartus_tan Executable

```
# Open a project
project_open my_project
# Always create the netlist first
create_timing_netlist
# List clock setup paths for clock clk
# from registers abc* to registers xyz*
report_timing -clock_setup -clock_filter clk -from abc* -to xyz*
# List the top 5 pin-to-pin combinational paths
report_timing -tpd -npaths 5
# List the top 5 pin-to-pin combinational paths and
# write output to an out.tao file
report_timing -tpd -npaths 5 -file out.tao
# Compute min tpd and append results to existing out.tao
report_timing -min_tpd -npaths 5 -file out.tao -append
# Show longest path (register to register data path) between a* and b*
report_timing -longest_paths -npaths 1
delete_timing_netlist
project close
```

Other Timing Analyzer Features

The Quartus II Classic Timing Analyzer provides many features for customizing and increasing the efficiency of static timing analysis, including:

- Wildcard assignments
- Assignment groups
- Fast corner analysis
- Early timing estimation
- Timing constraint checker
- Latch analysis

Wildcard Assignments

To simplify the tasks of making assignments to many node assignments, the Quartus II software accepts the * and ? wildcard characters. Use these wildcard characters to reduce the number of individual assignments you need to make for your design.

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

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

Assignment Groups

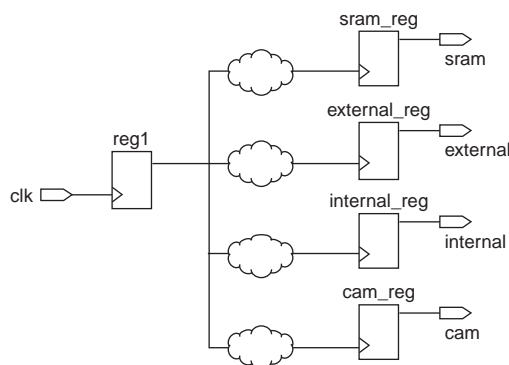
Assignment groups, also known as time groups, allow you to define a custom group of nodes to which you can assign timing assignments. You can also exclude specific nodes, wildcards, and time groups from a time group.

Use the `timegroup Tcl` command to create an assignment group. The following example creates an assignment group `srcgrp` and adds nodes with names that match `src1*` to the group:

```
timegroup srcgrp -add_member src1*
```

For example, [Figure 10-23](#) has false paths between source register `reg1` and destination register bank `sram_reg`, `external_reg`, `internal_reg`, and `cam_reg` that need to be cut. Without the use of assignment groups, the assignments required are:

```
set_timing_cut_assignment -from reg1 to sram_reg  
set_timing_cut_assignment -from reg1 to external_reg  
set_timing_cut_assignment -from reg1 to internal_reg  
set_timing_cut_assignment -from reg1 to cam_reg
```

Figure 10-23. False Path

With an assignment group called `dst_reg_bank`, the assignments required are:

```
#create a time group called dst_reg
timegroup dst_reg_bank -add_member sram_reg
timegroup dst_reg_bank -add_member external_reg
timegroup dst_reg_bank -add_member internal_reg
timegroup dst_reg_bank -add_member cam_reg
#cut timing paths
set_timing_cut_assignment -from reg1 to dst_reg_bank
```

Once an assignment group has been defined, applicable timing assignment can be made to the time group without redefining the assignment group.



Assigning individual nodes to time groups and applying timing assignments to these time groups can improve the performance of the Quartus II Classic Timing Analyzer.

Fast Corner Analysis

Fast Corner Analysis uses timing models generated under best-case conditions (voltage, process, and temperature) for the fastest speed-grade device.



Both Fast Corner and Slow Corner static timing analysis reports are saved to the `<project name>.tan.rpt` file, potentially overwriting previous timing analysis reports. To preserve a copy of your reports, save the file with a new name before the next compilation or static timing analysis, or use the Combined Fast/Slow Analysis report feature.

The Quartus II software also reports minimum delay checks after a slow corner (default) analysis. These results are generated by reporting minimum delay checks using worst-case timing models.

To perform fast corner static timing analysis with the best-case timing models, you can use the switch `--fast_model=on` with the `quartus_tan` executable. The following Tcl command enables the fast timing models:

```
quartus_tan <project_name> --fast_model=on
```


Early Timing Estimation

The majority of Quartus II software compilation time is consumed by the place-and-route process used to obtain optimal design results. To accelerate the design process for large designs, the Quartus II software provides **Early Timing Estimation**. This feature provides a quick static timing analysis in a fraction of the time required for a full compilation by performing a preliminary place-and-route on the design without full optimizations, which reduces total compile time by up to five times compared to a fully fitted design.



An Early Timing Estimate fit is not fully optimized or legally routed. The timing delay report is only an estimate. Typically, the estimated delays are within 10% of those obtained with a full fit when the realistic setting is used.

The Early Timing Estimate has three settings for generating timing estimates: Realistic, Optimistic, and Pessimistic. [Table 10-1](#) describes these settings.

Table 10-1. Early Timing Estimate Setting Options

Setting	Description
Realistic (default setting: estimates final timing using standard fitting)	Generates timing estimates that are likely to be closest to full compilation results.
Optimistic (estimates best-case final timing)	Generates timing estimates that are unlikely to be exceeded by full compilation.
Pessimistic (estimates worst-case final timing)	Generates timing estimates that are likely to be exceeded by full compilation.

To use the **Early Timing Estimate** feature, enter the following Tcl command when performing a fit:

```
quartus_fit \
--early_timing_estimate[=<realistic|optimistic|pessimistic>]
```

After **Early Timing Estimate** is complete, a full timing report is generated based on the early placement and routing delays. In addition, you can view the preliminary logic placement in the Timing Closure floorplan. The early timing placement allows you to perform initial placement and view the timing interaction of various placement topology.

Timing Constraint Checker

Altera recommends that you enter all timing constraints into the Quartus II software prior to performing a full compilation. This ensures that the Fitter targets the correct timing requirements and ensures that the Quartus II Classic Timing Analyzer reports the correct violations for all timing paths in the design. To ensure that all constraints have been applied to design nodes, the **Timing Constraint Check** feature reports all unconstraint paths in your design. [Example 10-10](#) shows the timing constraint check summary generated after a full compilation.

Example 10-10. Timing Constraint Check Summary

```

+-----+
; Timing Constraint Check Summary                                     ;
+-----+
; Timing Constraint Check Status           ; Analyzed - Tue Feb 28 11:42:31 2006      ;
; Quartus II Version                       ; 6.1 Internal Build 143 02/20/2006 SJ Full Version ;
; Revision Name                           ; test                                     ;
; Top-level Entity Name                   ; Block1                                   ;
; Unconstrained Clocks                    ; 0                                       ;
; Unconstrained Paths (Setup)              ; 22                                      ;
; Unconstrained Reg-to-Reg Paths (Setup)   ; 0                                       ;
; Unconstrained I/O Paths (Setup)          ; 22                                      ;
; Unconstrained Paths (Hold)               ; 12                                      ;
; Unconstrained Reg-to-Reg Paths (Hold)    ; 0                                       ;
; Unconstrained I/O Paths (Hold)           ; 12                                      ;
+-----+

```

To perform a timing constraint check, use the switch `--check_constraints` with the `quartus_tan` executable. The following Tcl command performs a timing constraint check on both setup and hold on the design system:

```
quartus_tan block1 --check_constraints=both
```

Latch Analysis

Latches are implemented in the Quartus II software as look-up-tables (LUTs) feeding back onto themselves. The Quartus II Classic Timing Analyzer can analyze these latches as synchronous elements rather than as combinational elements. The clock enables are analyzed as inverted clocks. The Quartus II Classic Timing Analyzer reports the results of setup and hold analysis on these latches.

You can turn on the Analyze Latches As Synchronous Elements option with the following Tcl command:

```
set_global_assignment -name ANALYZE_LATCHES_AS_SYNCHRONOUS_ELEMENTS ON
```

Timing Analysis Using the Quartus II GUI

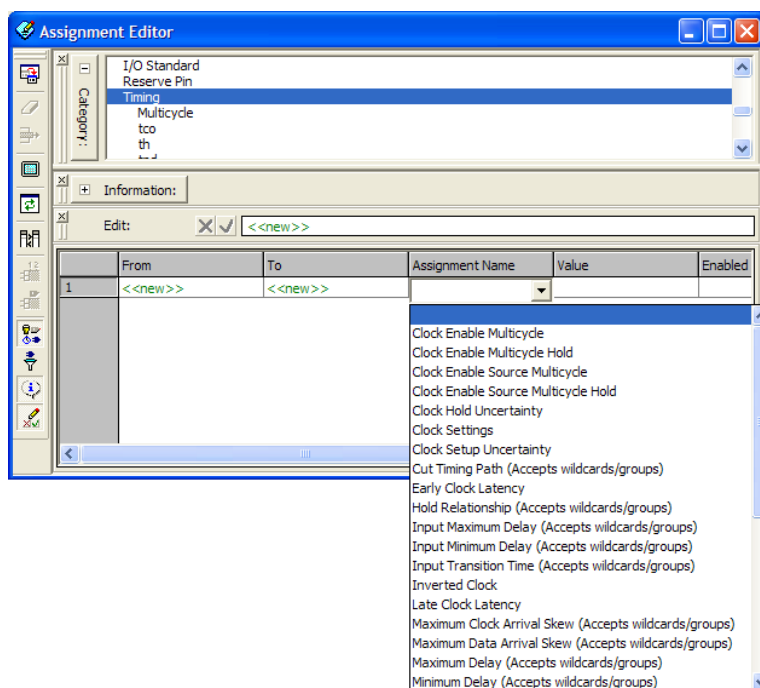
In addition to the extensive scripting support available in the Quartus II Classic Timing Analyzer, the Quartus II software provides the Assignment Editor and other user interface tools, giving you access to the Quartus II Classic Timing Analyzer features and assignments.

Assignment Editor

The Assignment Editor is a spreadsheet-style interface used for adding, modifying, and deleting timing assignments.

To make timing assignments in the Assignment Editor, choose **Timing** from the category list to cause the Assignment Name column to display only timing assignments. Double-click <<new>> in the Assignment Name field, the Assignment Name list displays. [Figure 10-24](#) shows the Assignment Editor with the Assignment Name list displaying timing assignment types.

Figure 10-24. Assignment Editor



For more information about the Assignment Editor, refer to the *Assignment Editor* chapter in volume 2 of the *Quartus II Handbook*.

Timing Settings

You can specify delay requirements and clock settings with the **Timing Analysis Settings** page of the **Settings** dialog box.

To access this page, on the Assignments menu, click **Settings**. In the **Category** list, click the “+” icon next to **Timing Analysis Settings** to expand the folder. (Be sure that the **Use Classic Timing Analyzer during compilation** radio button is turned on.) Select **Classic Timing Analyzer Settings**. The **Classic Timing Analysis Settings** page appears.

Clock Settings Dialog Box

You can create or modify base clock settings or derived clock settings using the **Clock Settings** dialog box. To access this page, on the Assignments menu, click **Settings**. In the **Category** list, click the “+” icon next to **Timing Analysis Settings** to expand the folder. (Be sure that the **Use Classic Timing Analyzer during compilation** radio button is turned on.) Click on **Classic Timing Analyzer Settings**. The **Timing Analysis Settings** page displays. Under **Clock Settings**, click **Individual Clocks**. The **Individual Clock** dialog box appears.



Click the **New** button in the **Individual Clocks** dialog box to access the **New Clock Settings** dialog box and create a base or derived clock setting.

More Timing Settings Dialog Box

On the **Timing Analysis Settings** page of the **Settings** dialog box, click **More Settings** to display the **More Timing Settings** dialog box. The **More Timing Settings** dialog box provides access to many global timing analysis options.

Timing Reports

The Quartus II Classic Timing Analyzer report is a section of the Compilation Report containing the static timing analysis results. The Quartus II Classic Timing Analyzer report includes clock setup and clock hold measurements for all clock sources. The report also shows t_{CO} for all output pins, t_{SU} and t_H for all input pins, and t_{PD} for any pin-to-pin combinational paths in the design. Other reports are created for different analyses and device features.

In the **Settings** dialog box, you can specify the range of information to be reported in the timing analysis of the Compilation Report. To access this page, on the Assignments menu, click **Settings**. In the **Category** list, click the  icon next to **Timing Analysis Settings** to expand the folder. (Be sure that the **Use Classic Timing Analyzer during compilation** radio button is turned on.) Click the  icon next to **Classic Timing Analyzer Settings** to expand the folder. Click **Classic Timing Analyzer Reporting**. The **Classic Timing Analyzer Reporting** dialog box appears.

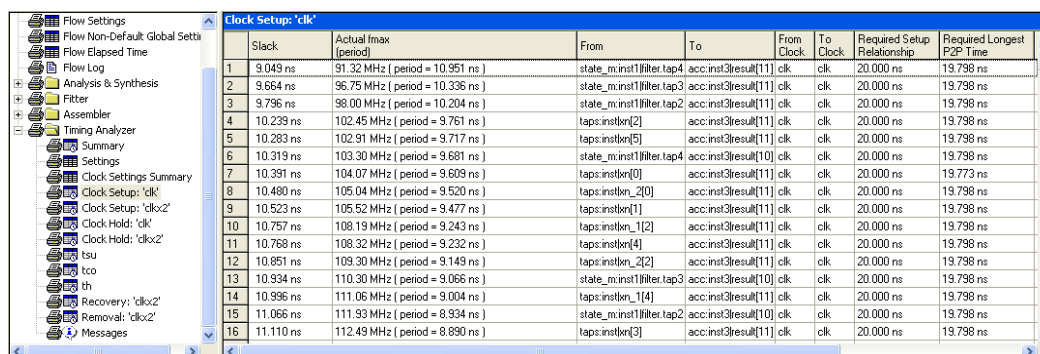
If there are no timing assignments for the design, the Quartus II Classic Timing Analyzer does not generate slack reports for any detected clock nodes. The Quartus II Classic Timing Analyzer only reports slack measurements for pins with individual or global t_{SU} , t_H , or t_{CO} assignments. A positive slack indicates the margin by which the path surpasses the clock timing requirements. A negative slack indicates the margin by which the path fails the clock timing requirements.



This Timing Analysis report is also available in text format located in the design directory with the file name *<revision name>.tan.rpt*.

In the Compilation Report, select an analysis type under the Timing Analyzer folder to display the analysis report; for example, Clock Setup or Clock Hold. [Figure 10-25](#) shows an example of a Clock Setup report for clock signal *clk*.

Figure 10-25. Timing Analysis Report



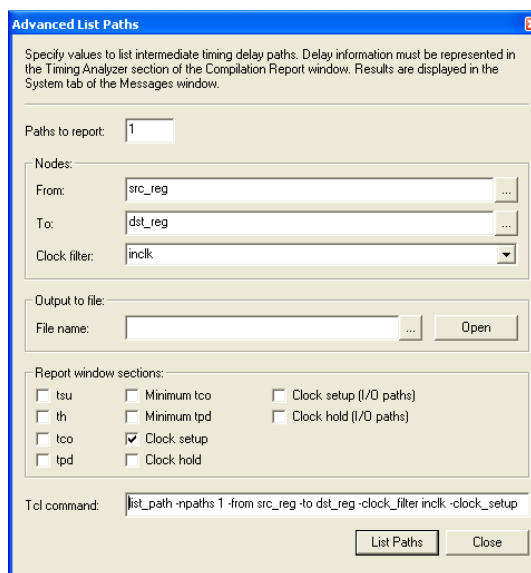
	Slack	Actual fmax (period)	From	To	From Clock	To Clock	Required Setup Relationship	Required Longest P2P Time
1	9.049 ns	91.32 MHz (period = 10.951 ns)	state_minst1filter.tap4	acc:inst3result[11] clk	clk	clk	20.000 ns	19.798 ns
2	9.664 ns	96.75 MHz (period = 10.336 ns)	state_minst1filter.tap3	acc:inst3result[11] clk	clk	clk	20.000 ns	19.798 ns
3	9.796 ns	98.00 MHz (period = 10.204 ns)	state_minst1filter.tap2	acc:inst3result[11] clk	clk	clk	20.000 ns	19.798 ns
4	10.239 ns	102.45 MHz (period = 9.761 ns)	taps:inst0b[2]	acc:inst3result[11] clk	clk	clk	20.000 ns	19.798 ns
5	10.283 ns	102.91 MHz (period = 9.717 ns)	taps:inst0b[5]	acc:inst3result[11] clk	clk	clk	20.000 ns	19.798 ns
6	10.319 ns	103.30 MHz (period = 9.681 ns)	state_minst1filter.tap4	acc:inst3result[10] clk	clk	clk	20.000 ns	19.798 ns
7	10.391 ns	104.07 MHz (period = 9.609 ns)	taps:inst0b[0]	acc:inst3result[11] clk	clk	clk	20.000 ns	19.773 ns
8	10.480 ns	105.04 MHz (period = 9.520 ns)	taps:inst0b[20]	acc:inst3result[11] clk	clk	clk	20.000 ns	19.798 ns
9	10.523 ns	105.52 MHz (period = 9.477 ns)	taps:inst0b[1]	acc:inst3result[11] clk	clk	clk	20.000 ns	19.798 ns
10	10.757 ns	108.19 MHz (period = 9.243 ns)	taps:inst0b[12]	acc:inst3result[11] clk	clk	clk	20.000 ns	19.798 ns
11	10.768 ns	108.32 MHz (period = 9.232 ns)	taps:inst0b[4]	acc:inst3result[11] clk	clk	clk	20.000 ns	19.798 ns
12	10.951 ns	109.30 MHz (period = 9.149 ns)	taps:inst0b[22]	acc:inst3result[11] clk	clk	clk	20.000 ns	19.798 ns
13	10.934 ns	110.30 MHz (period = 9.066 ns)	state_minst1filter.tap3	acc:inst3result[10] clk	clk	clk	20.000 ns	19.798 ns
14	10.996 ns	111.06 MHz (period = 9.004 ns)	taps:inst0b[14]	acc:inst3result[11] clk	clk	clk	20.000 ns	19.798 ns
15	11.066 ns	111.93 MHz (period = 8.934 ns)	state_minst1filter.tap2	acc:inst3result[10] clk	clk	clk	20.000 ns	19.798 ns
16	11.110 ns	112.49 MHz (period = 8.890 ns)	taps:inst0b[3]	acc:inst3result[11] clk	clk	clk	20.000 ns	19.798 ns

Advanced List Path

The **Advanced List Paths** dialog box provides detailed information about a specific path, such as interconnect and cell delays between any two valid register-to-register paths (Figure 10-26).

The **Advanced List Paths** dialog box allows you to select the type of paths you want listed. For example, you can obtain detailed information for Clock Setup and Clock Hold for a specific clock. In addition, the Tcl command field in the window matches the equivalent Tcl command you can use in either a custom Tcl script or in the Tcl console.

Figure 10-26. Advanced List Paths Dialog Box

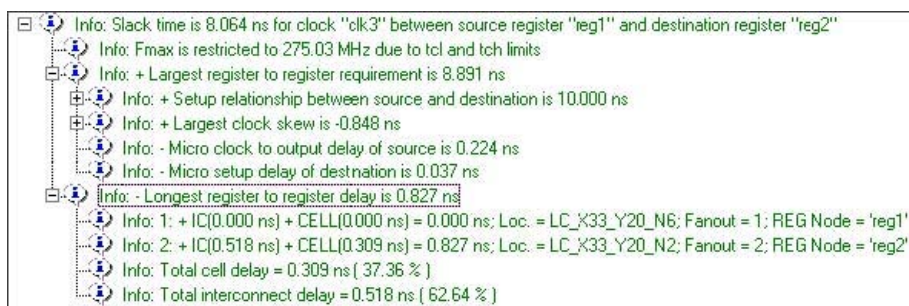


You can perform a list path command directly from the Timing Analysis report. To do this, right click a path and click **List Path** (Figure 10-27). To launch the **Advanced List Paths** dialog box, right-click a path and in the menu that appears, and select **Advanced List Paths**.

The **Advanced List Paths** dialog box displays only paths that are visible in the Timing Analysis report. To increase the amount of paths reported by the Quartus II Classic Timing Analyzer, on the Assignments menu, click **Timing Analysis Settings**. In the **Category** list, expand **Timing Analysis Settings** and select **Timing Analyzer Reporting**. In the **Timing Analyzer Reporting page**, specify the range of information to be reported by the Quartus II Classic Timing Analyzer.



Both the **Advanced List Paths** and the **List Path** commands display the path information in the **System** message window.

Figure 10-27. List Path in the Message Window

If the **Combined Fast/Slow Timing** option is enabled, the **List Path** Tcl command displays only path delays reported in the Slow Model section.

Early Timing Estimate

To start an Early Timing Estimate, on the Processing menu, point to **Start** and click **Start Early Timing Estimate**. To specify the Early Timing Estimate mode, on the Assignments menu, click **Settings**. In the **Category** list, select **Compilation Processes Settings**, select **Early Timing Estimate** and click the desired timing estimate mode. For more information about the Early Timing Estimate feature, refer to “[Early Timing Estimation](#)” on page 10-33.

Assignment Groups

To define, modify, and delete assignment groups, also known as time groups, from a single dialog box, on the Assignments menu, click **Assignment (Time) Groups**. The **Assignment Groups** dialog box appears.

Scripting Support

You can run procedures and make settings described in this chapter in a Tcl script. You can also run some procedures at a command prompt. For detailed information about scripting command options, refer to the Quartus II Command-Line and Tcl API Help browser. To run the Help browser, type the following command at the command prompt:

```
quartus_sh --qhelp
```



For more information in PDF form, refer to the [Quartus II Scripting Reference Manual](#).



For more information about Tcl scripting, refer to the [Tcl Scripting](#) chapter in volume 2 of the [Quartus II Handbook](#).



For information about all settings and constraints in the Quartus II software, refer to the [Quartus II Settings File Reference Manual](#).



For more information about command-line scripting, refer to the [Command-Line Scripting](#) chapter in volume 2 of the [Quartus II Handbook](#).

Creating Clocks

There are two Tcl commands that allow you to define clocks in a design, `create_base_clock` and `create_relative_clock`.

Base Clocks

Use the `create_base_clock` Tcl command to define a base clock:

```
create_base_clock [-h | -help] [-long_help] -fmax <fmax> \
[-duty_cycle <integer>] [-virtual] [-target <name>] [-no_target] \
[-entity <entity>] [-disable] [-comment <comment>] <clock_name>
```

To define a base clock setting named `sys_clk` with a 100 MHz requirement applied to node `clk_src`, enter the following Tcl command:

```
create_base_clock -fmax 100MHz -target clk_src sys_clk
```

Derived Clocks

Use the `create_relative_clock` Tcl command to define a relative clock:

```
create_relative_clock [-h | -help] [-long_help] \
-base_clock <Base clock> [-duty_cycle <integer>] \
[-multiply <integer>] [-divide <integer>] [-offset <offset>] \
[-phase_shift <integer>] [-invert] [-virtual] [-target <name>] \
[-no_target] [-entity <entity>] [-disable] \
[-comment <comment>] <clock_name>
```

To define a relative clock named `aux_clk` based upon base clock setting `sys_clk` with a multiplication factor of 2 applied to node `rel_clk`, enter the following Tcl command:

```
create_relative_clock -base_clock sys_clk -multiply 2 \
-target rel_clk aux_clk
```

Clock Latency

You can use the `set_clock_latency` Tcl command to create either an early or late clock latency assignment:

```
set_clock_latency [-h | -help] [-long_help] [-early] [-late] \
-to <to> [<value>]
```

To apply an early clock latency of 1 ns and a late clock latency of 2 ns to clock node `clk`, enter the following Tcl commands:

```
set_clock_latency -early -to clk 2ns
```

Clock Uncertainty

You can use the `set_clock_uncertainty` Tcl command to create clock uncertainty assignments as shown in the following example:

```
set_clock_uncertainty [-h] [-help] [-long_help] [-from \
<source clock name> ] -to <destination clock name> [-setup] [-hold] \
[-remove] [-disable] [-comment <comment>] <value>
```

To apply a clock setup uncertainty of 50 ps between source clock node `clk_src` and destination clock node `clk_dst`, enter the following Tcl command:

```
set_clock_uncertainty -from clk_src -to clk_dst -setup 50ps
```


To apply a clock hold uncertainty of 25 ps between to clock node `clk_sys`, enter the following Tcl command:

```
set_clock_uncertainty -to clk_sys -setup 25ps
```

Cut Timing Paths

You can use the `set_timing_cut_assignment` Tcl command to create cut timing assignments:

```
set_timing_cut_assignment [-h | -help] [-long_help] \
[-from <from_node_list>] [-to <to_node_list>] [-remove] [-disable] \
[-comment <comment>]
```

To cut the timing path from source register `reg1` to destination register `reg2`, enter the following Tcl command:

```
set_timing_cut_assignment -from reg1 -to reg2
```

Input Delay Assignment

You can use the Tcl command `set_input_delay` to create input delay assignments:

```
set_input_delay [-h | -help] [-long_help] [-clk_ref <clock>] \
-to <input_pin> [-min] [-max] [-clock_fall] [-remove] [-disable] \
[-comment <comment>] [<value>]
```

To apply an input maximum delay of 2 ns to an input pin named `data_in` that feeds a register clocked by clock source `clk`, enter the following Tcl command:

```
set_input_delay -clk_ref clk -to data_in -max 2ns
```

Maximum and Minimum Delay

The following Tcl commands create the Maximum Delay and Minimum Relationship assignments, respectively:

```
set_instance_assignment -name MAX_delay <value> -from <node> -to <node>
set_instance_assignment -name MIN_delay <value> -from <node> -to <node>
```

To apply a Maximum Delay of 8 ns and a minimum of 5 ns between source register `reg1` and destination register `reg2`, enter the following Tcl command:

```
set_instance_assignment -name MAX_DELAY 8ns -from reg1 -to reg2
set_instance_assignment -name MIN_DELAY 5ns -from reg1 -to reg2
```

To apply a Maximum Delay of 10 ns for all paths from source clock `clk_src` to destination clock `clk_dst`, enter the following Tcl command:

```
set_instance_assignment -name MAX_DELAY 10ns -from clk_src -to clk_dst
```

Maximum Clock Arrival Skew

The following Tcl command defines the Maximum Clock Arrival Skew assignment:

```
set_instance_assignment -name max_clock_arrival_skew <value> \
-from <clock> -to <node>
```

To apply a Maximum Clock Arrival Skew of 1 ns for clock source `clk` to a predefined timegroup called `reg_group`, enter the following Tcl command:

```
set_instance_assignment -name max_clock_arrival_skew 1ns -from clk \
-to reg_group
```


Maximum Data Arrival Skew

To create Maximum Data Arrival Skew assignments, use the Tcl command `set_instance_assignment -name max_data_arrival`:

```
set_instance_assignment -name max_data_arrival_skew <value> \
-from <clock> -to <node>
```

To apply a Maximum Data Arrival Skew of 1 ns for clock source `clk` to a predefined timegroup of pins called `pin_group`, enter the following Tcl command:

```
set_instance_assignment -name max_data_arrival_skew 1ns -from clk \
-to pin_group
```

Multicycle

Use the `set_multicycle_assignment` Tcl command to create Multicycle assignments:

```
set_multicycle_assignment [-h | -help] [-long_help] [-setup] [-hold] \
[-start] [-end] [-from <from_list>] [-to <to_list>] [-remove] \
[-disable] [-comment <comment>] <path_multiplier>
```

To apply a Multicycle Setup of 2 and a Hold Multicycle of 1 between source register `reg1` and destination register `reg2`, enter the following Tcl commands:

```
set_multicycle_assignment -setup -end -from reg1 -to reg2 2
set_multicycle_assignment -hold -end -from reg1 -to reg2 1
```

To apply a Source Multicycle Setup of 2 between source register `reg1` and destination register `reg2`, enter the following Tcl command:

```
set_multicycle_assignment -setup -start -from reg1 -to reg2 1
```

To apply a multicycle setup of 2 for all paths from source clock `clk_src` to destination clock `clk_dst`, enter the following Tcl command:

```
set_multicycle_assignment -setup -end -from clk_src -to clk_dst 2
```

Output Delay Assignment

Use the Tcl command `set_output_delay` to create Output Delay assignments:

```
set_output_delay [-h | -help] [-long_help] [-clk_ref <clock>] \
-to <output_pin> [-min] [-max] [-clock_fall] [-remove] [-disable] \
[-comment <comment>] [<value>]
```

To apply an Output Maximum Delay of 3 ns to an output pin named `data_out` that is fed to a register clocked by clock source `clk`, enter the following Tcl command:

```
set_output_delay -clk_ref clk -to data_out -max 3ns
```

Report Timing

Use the `report_timing` Tcl command to generate timing reports:

```
report_timing [-h | -help] [-long_help] [-npaths <number>] [-tsu] \
[-th] [-tco] [-tpd] [-min_tco] [-min_tpd] [-clock_setup] \
[-clock_hold] [-clock_setup_io] [-clock_hold_io] [-clock_setup_core] \
[-clock_hold_core] [-recovery] [-removal] [-dqs_read_capture] \
[-stdout] [-file <name>] [-append] [-table <name>] [-from <names>] \
[-to <names>] [-clock_filter <names>] [-src_clock_filter <names>] \
[-longest_paths] [-shortest_paths] [-all_failures]
```

The following example generates a list of all clock setup paths for clock source `clk` from registers `src_reg*` to registers `dst_reg*`:

```
report_timing -clock_setup -clock_filter clk -from src_reg* \
-to dst_reg*
```

Setup and Hold Relationships

The following Tcl commands create Setup Relationship and Hold Relationship assignments, respectively:

```
set_instance_assignment -name SETUP_RELATIONSHIP <value> -from <node> \
-to <node>
set_instance_assignment -name HOLD_RELATIONSHIP <value> -from <node> \
-to <node>
```

To apply a Setup Relationship of 12 ns and a Hold Relationship of 2 ns between source register `reg1` and destination registers `reg2`, enter the following Tcl command:

```
set_instance_assignment -name SETUP_RELATIONSHIP 12ns -from reg1 \
-to reg2
set_instance_assignment -name HOLD_RELATIONSHIP 2ns -from reg1 -to reg2
```

To apply a setup relationship of 10 ns for all paths from source clock `clk_src` to destination clock `clk_dst`, enter the following Tcl command:

```
set_instance_assignment -name SETUP_RELATIONSHIP 10ns -from clk_src \
-to clk_dst
```

Assignment Group

Use the `timegroup` Tcl command to create assignment groups:

```
timegroup [-h | -help] [-long_help] [-add_member <name>] \
[-add_exception <name>] [-remove_member <name>] [-remove_exception \
<name>] [-get_members] [-get_exceptions] [-overwrite] [-remove] \
[-disable] [-comment <comment>] <group_name>
```

The following example creates an assignment group called `reg_bank` with members `dst_reg*`, and excludes register `dst_reg5`.

```
timegroup reg_bank -add_member dst_reg* -add_exception dst_reg5
```

Virtual Clock

Use the `create_relative_clock` with the `-virtual` switch to create Virtual Clock assignments:

```
create_relative_clock [-h | -help] [-long_help] -base_clock \
<Base clock> [-duty_cycle <integer>] [-multiply <integer>] \
[-divide <integer>] [-offset <offset>] [-phase_shift <integer>] \
[-invert] [-virtual] [-target <name>] [-no_target] [-entity <entity>] \
[-disable] [-comment <comment>] <clock_name>
```

To define a virtual clock derived from the base clock setting `clk_aux` named `brd_sys`, enter the following Tcl command:

```
create_relative_clock -base_clock clk_aux -virtual brd_sys
```

MAX+PLUS II Timing Analysis Methodology

This section describes the basic static timing analysis and assignments available in the Quartus II software that originated in the MAX+PLUS® II design software.

f_{MAX} Relationships

Maximum clock frequency is the fastest speed at which the design clock can run without violating internal setup and hold time requirements. The Quartus II software performs static timing analysis on both single- and multiple-clock designs.



Apply clock settings to all clock nodes in a design to ensure that you meet all performance requirements. Refer to “Clock Settings” on page 10-7 for more information.

Slack

Slack is the margin by which a timing requirement such as f_{MAX} is met or not met. Positive slack indicates the margin by which a requirement is met. Negative slack indicates the margin by which a requirement is not met. The Quartus II software determines slack using Equation 10-35 through Equation 10-38.

Equation 10-35.

$$\text{Clock Setup Slack} = \text{Longest Register-to-Register Requirement} - \text{Longest Register-to-Register Delay}$$

Equation 10-36.

$$\text{Register-to-Register Requirement} = \text{Setup Relationship} + \text{Largest Clock Skew} - \text{micro } t_{CO} \text{ of Source Register} - \text{micro } t_{SU} \text{ of Destination Register}$$

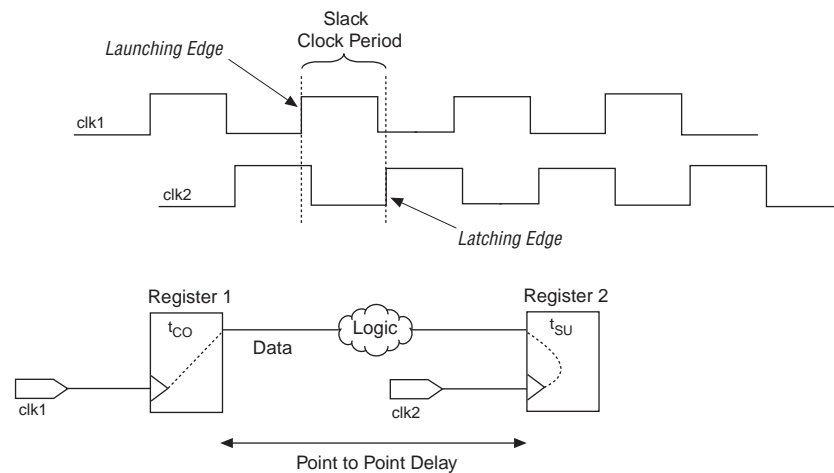
Equation 10-37.

$$\text{Clock Hold Slack} = \text{Shortest Register-to-Register Delay} - \text{Smallest Register-to-Register Requirement}$$

Equation 10-38.

$$\text{Shortest Register-to-Register Requirement} = \text{Hold Relationship} + \text{Smallest Clock Skew} - \text{micro } t_{CO} \text{ of Source Register} - \text{micro } t_H \text{ of Destination Register}$$

Figure 10-28 shows a slack calculation diagram.

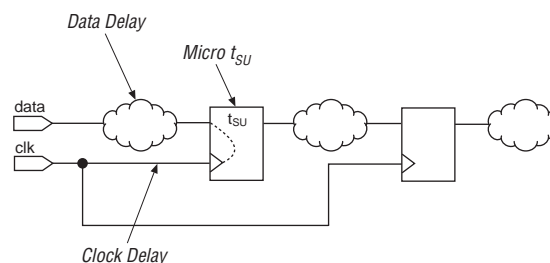
Figure 10-28. Slack Calculation Diagram

I/O Timing

This section describes the basic measurements made for I/O timing in the Quartus II software.

t_{SU} Timing

t_{SU} specifies the length of time data needs to arrive and be stable at an external input pin prior to a clock transition on an associated clock I/O pin. A t_{SU} requirement describes this relationship for an input register relative to the I/O pins of the FPGA. [Figure 10-29](#) shows a diagram of clock setup time.

Figure 10-29. Clock Setup Time (t_{SU})

Micro t_{SU} is the internal setup time of the register. It is a characteristic of the register and is unaffected by the signals feeding the register. [Equation 10-39](#) calculates the t_{SU} of data with respect to clk for the circuit shown in [Figure 10-29](#).

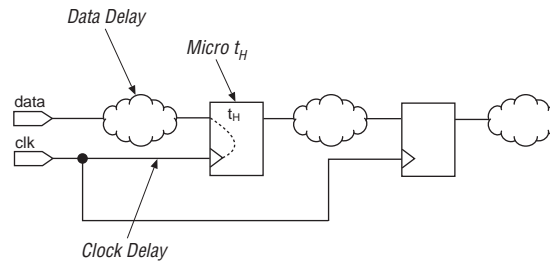
Equation 10-39.

$$t_{SU} = \text{Longest Data Delay} - \text{Shortest Clock Delay} + \text{micro } t_{SU} \text{ of Input Register}$$

t_H Timing

t_H specifies the length of time data needs to be held stable on an external input pin after a clock transition on an associated clock I/O pin. A t_H requirement describes this relationship for an input register relative to the I/O pins of the FPGA. Figure 10-30 shows a diagram of clock hold time.

Figure 10-30. Clock Hold Time (t_H)



Micro t_H is the internal hold time of the register. Equation 10-40 calculates the t_H of data with respect to clk for the circuit shown in Figure 10-30.

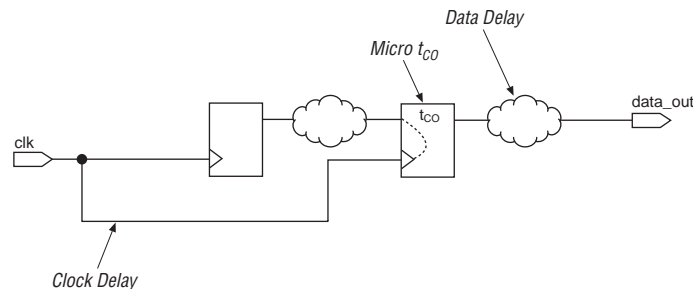
Equation 10-40.

$$t_H = \text{Longest Clock Delay} - \text{Shortest Data Delay} + \text{micro } t_H \text{ of Input Register}$$

t_{CO} Timing

Clock-to-output delay is the maximum time required to obtain a valid output at an output pin fed by a register, after a clock transition on the input pin that clocks the register. Micro t_{CO} is the internal clock-to-output delay of the register. Figure 10-31 shows a diagram of clock-to-output delay.

Figure 10-31. Clock-to-Output Delay (t_{CO})



Equation 10-41 calculates the t_{CO} for output pin `data_out` with respect to clock node `clk` for the circuit shown in Figure 10-31.

Equation 10-41.

$$t_{CO} = \text{Longest Clock Delay} + \text{micro } t_{CO} \text{ of Output Register}$$

Minimum t_{CO} (min t_{CO})

Minimum clock-to-output delay is the minimum time required to obtain a valid output at an output pin fed by a register, after a clock transition on the input pin that clocks the register. Micro t_{CO} is the internal clock-to-output delay of registers in Altera FPGAs. Unlike the t_{CO} assignment, the min t_{CO} assignment looks at the shortest delay paths (Equation 10-42).

Equation 10-42.

$$\min t_{CO} = \text{Shortest Clock Delay} + \text{Shortest Data Delay} + \text{micro } t_{CO} \text{ of Output Register}$$

 t_{PD} Timing

Pin-to-pin delay (t_{PD}) is the time required for a signal from an input pin to propagate through combinational logic and appear at an external output pin (Equation 10-43).

Equation 10-43.

$$t_{PD} = \text{Longest Pin-to-Pin Delay}$$



In the Quartus II software, you can make t_{PD} assignments between an input pin and an output pin.

Minimum t_{PD} (min t_{PD})

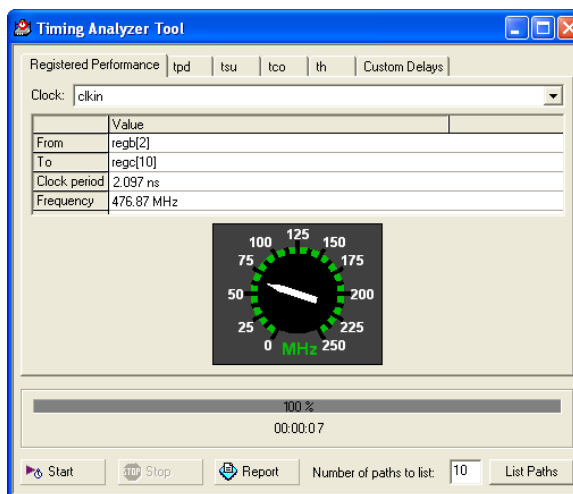
The minimum pin-to-pin delay (t_{PD}) is the time required for a signal from an input pin to propagate through combinational logic and appear at an external output pin. Unlike the t_{PD} assignment, the min t_{PD} assignment applies to the shortest pin-to-pin delay (Equation 10-44).

Equation 10-44.

$$\min t_{PD} = \text{Shortest Pin-to-Pin Delay}$$

The Timing Analyzer Tool

To facilitate the classic static timing analysis flow and constraint, the Quartus II software provides a MAX+PLUS II-style Timing Analyzer Tool available on the Tools menu. The Timing Analyzer Tool provides a simple interface, similar to the Timing Analyzer tool in MAX+PLUS II, that reports register-to-register performance, I/O timing, and custom delay values (Figure 10-32).

Figure 10-32. Timing Analyzer Tool

Conclusion

Evolving design and aggressive process technologies require larger and higher-performance FPGA designs. Increasing design complexity demands enhanced static timing analysis tools that aid designers in verifying design timing requirements. Without advanced static timing analysis tools, you risk circuit failure in complex designs. The Quartus II Classic Timing Analyzer incorporates a set of powerful static timing analysis features critical in enabling system-on-a-programmable-chip (SOPC) designs.

Referenced Documents

This chapter references the following documents:

- *ALTPLL Megafunction User Guide*
- *AN 411: Understanding PLL timing for Stratix II Devices*
- *Assignment Editor* chapter in volume 2 of the *Quartus II Handbook*
- *Command-Line Scripting* chapter in volume 2 of the *Quartus II Handbook*
- *Quartus II Scripting Reference Manual*
- *Quartus II Settings File Reference Manual*
- *Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*
- *Switching to the Quartus II TimeQuest Timing Analyzer* chapter in volume 3 of the *Quartus II Handbook*
- *Tcl Scripting* chapter in volume 2 of the *Quartus II Handbook*

Document Revision History

Table 10-2 shows the revision history for this chapter.

Table 10-2. Document Revision History

Date and Version	Changes Made	Summary of Changes
March 2009 v9.0.0	This was chapter 9 in version 8.1.	—
November 2008 v8.1.0	Changed to 8-1/2 x 11 page size. No change to content.	Updated for the Quartus II software version 8.1 release.
May 2008 v8.0.0	Added hyperlinks to referenced documents throughout the chapter. No other substantive changes were made.	—



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