

---

## Performing Equivalent Timing Analysis Between Altera Classic Timing Analyzer and Xilinx Trace

### Introduction

Most hardware designers who are qualifying FPGA performance normally run “bake-off”-style software benchmark comparisons of FPGAs from different vendors to determine which vendor provides the largest margin for their timing requirements. Unfortunately, out-of-the-box design compilations do not produce equivalent or fair performance comparisons because different software tools have different default timing analysis and optimization behaviors.

Altera’s Classic Timing Analyzer, in general, calculates all possible register-to-register and complex clock structures using the worst possible assumptions. Xilinx’s Trace timing analyzer does not analyze many of these complex structures, and comparing the software tools on this basis unfairly penalizes Quartus® II performance. The Classic Timing Analyzer makes the most stringent assumptions, allowing users to see possible problems by default. This can cause users comparing the two tools to perceive, initially, that Quartus II performance is inferior, which is not the case in actual practice. For instance, if designers determine that the reported problems are not significant, they can adjust the Classic Timing Analyzer to not report these problems, changing the reported performance of the design.

This document covers the differences in timing analysis between Altera’s Classic Timing Analyzer and Xilinx’s Trace, and explains how to configure the tools to provide equivalent performance comparison. Quartus II system-level performance is improved when the settings for the Classic Timing Analyzer are adjusted to perform timing analysis equivalent to the timing analysis performed by Trace.

### Differences in Philosophy

Altera’s Quartus II software package offers the Classic Timing Analyzer as the default timing analyzer for Altera® device families that precedes and includes Stratix® II, Stratix II GX, and Cyclone® II FPGA, as well as MAX II CPLDs. Xilinx’s ISE software package offers Trace as its timing analyzer. Classic Timing Analyzer and Trace are fundamentally different in their constraint and timing analysis philosophies. The following sections outline major differences that affect out-of-the-box experience.

#### *Full Analysis vs. Constrained Analysis*

The Classic Timing Analyzer analyzes all possible paths whether they are constrained or not. By default, the Classic Timing Analyzer analyzes everything, including gated clocks, registered clocks, and combinatorial loops. The user can cut these paths. In the absence of timing constraints, the Quartus II software attempts to optimize all clock paths in the design. This accounts for worst-case scenarios and identifies all potential problems.

By default, Xilinx’s Trace only analyzes constrained paths and does not optimize or report unconstrained paths. The Trace no-constraint or “advanced” analysis (the `trace -a` option) has limited use in competitive comparisons because the design must be fully constrained in ISE to produce good results. However, this mode is useful in identifying all clocks in the design. Trace also does not account for certain complex clock or data structures, such as registered clocks and combinatorial loops. These structures are discussed in detail in later sections.

#### *Clock Relation*

The Classic Timing Analyzer assumes clock signals coming from pins and their derivative clocks are related by default. Timing constraints can be declared related or unrelated. Xilinx’s Trace requires timing constraints to relate clocks with their derivatives. Trace provides an advanced timing analysis mode (see below) for unconstrained designs but it does not assume any clock relationships. To ensure a fair comparison, the same constraint, assumed or declared, must be made in both tools.


### Cross-Domain Clock Analysis

Because the Classic Timing Analyzer assumes all clock pins and their derivatives are related by default, it also analyzes paths crossing multiple clock domains. Xilinx's Trace does not perform this analysis by default and requires combinatorial delay constraints for performing cross-domain analysis. Neither tool analyzes unrelated clock domains.

In general, the Classic Timing Analyzer makes strict assumptions about any unconstrained paths and always performs analysis based on the worst possible scenarios. This makes designers aware of any potential problems. If there are no problems, users can loosen the strictness of Classic Timing Analyzer by applying timing constraints. Xilinx's Trace generally analyzes timing based on relaxed assumptions unless users make tighter constraints. While this causes Classic Timing Analyzer's default results to appear worse, this is a thorough and complete timing analysis.

### Differences in the Classic Timing Analyzer and Trace Timing Analyzers

The main difference between the Classic Timing Analyzer and Xilinx's Trace timing analyzers is the types of paths or clock structures that each tool analyzes. The Classic Timing Analyzer and Trace differ significantly in their analysis of the commonly used structures described in the following sections.

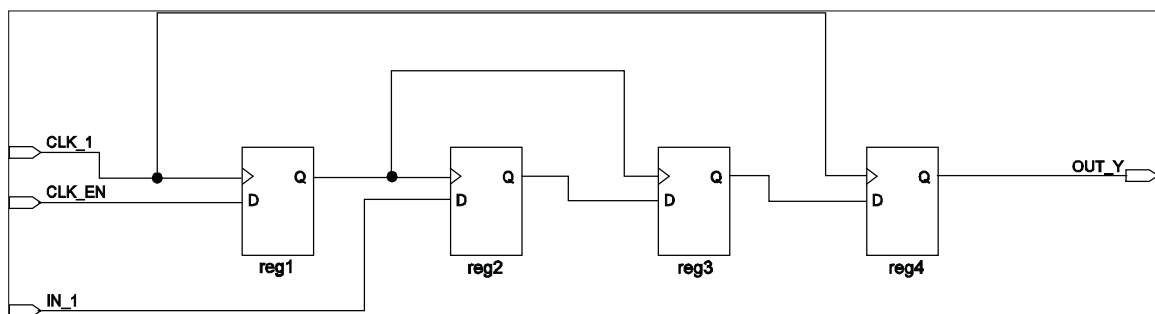
 This white paper assumes that constraints exist on all clocks of interest in the example designs described in the following sections.

### Registered-Clock Structures

Registered-clock structures are clock signals driven by the output of registers. Cascaded registered-clock structures where more than one registered clocks are connected in a chain are not uncommon. These structures are generated by synthesis tools as a function of clock-enable requirements, multicycle requirements, or user created clock divider requirements.

Classic Timing Analyzer adds the microtiming clock-to-out ( $t_{CO}$ ) delay of the registered clock as part of the clock skew and reduces the register-to-register maximum frequency. This is done to account for the worst case scenario. In a design similar to that shown in [Figure 1](#), the path between reg3 and reg4 are analyzed using reg1 as clock skew delay. Xilinx's Trace does not analyze paths using registered clocks. To analyze registered clocks, the ISE software treats the output of the registered clock as a separate domain.

Figure 1. Registered Clock Example



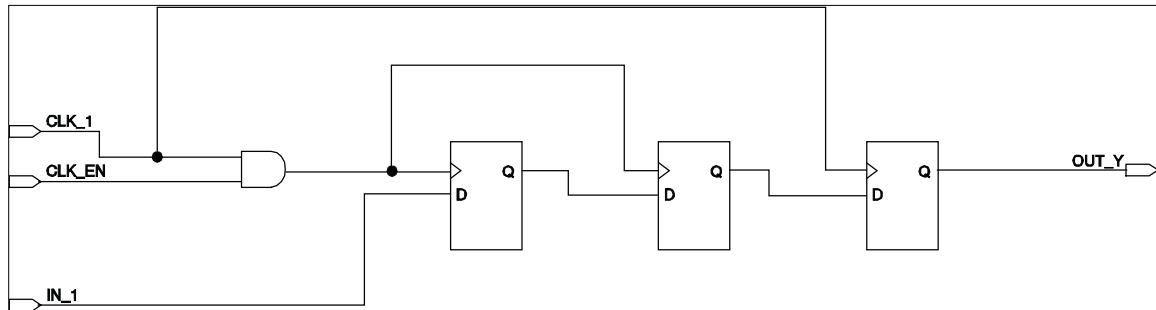
Users of the Classic Timing Analyzer must assign internal registered clock signals to their own domains for equivalent timing analysis between the two tools.

### Gated-Clock Structures

Gated-clock structures are clock paths driven by logic. Different registers can also use clock signals from different tap points of logic, effectively making use of different signals. Gated clocks cause clock skews among the affected registers and add to the minimum timing delay. The clock skews must be accounted for during timing analysis because they reduce register-to-register maximum frequency.

If the user has constrained the design, Xilinx's Trace only analyzes and reports constrained paths. In contrast, the Classic Timing Analyzer reports all possible and worst-case gated-clock paths by default. Both tools correctly measure clock skews from gated-clock logic, if users of the ISE software properly constrain their designs. When analyzing timing for the structure shown in Figure 2, the Classic Timing Analyzer calculates the  $f_{MAX}$  of the CLK\_1 and CLK\_EN input pins, but Trace reports neither unless they are constrained.

Figure 2. Gated Clock Example

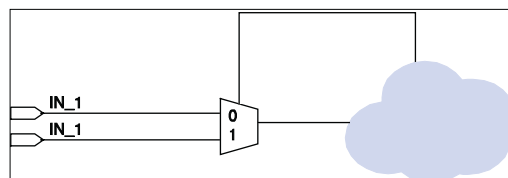


In designs with multiple inputs to logic that result in a clock signal, the Classic Timing Analyzer analyzes and reports all inputs as clocks unless the user specifies otherwise. Trace does not report any of these signals as clocks unless directed to do so by the user.

### Combinatorial Loop Structures

Combinatorial loops are logic structures designed to utilize outputs from the structure as partial input to the same structure. The total combinatorial delay from the source to the destination register is theoretically increased because of this extra logic path. However, the majority of combinatorial loops are false paths or “don’t care” paths. As shown in Figure 3, they are most often caused by synthesis of incomplete if-else or case statements.

Figure 3. Combinatorial Loop Example

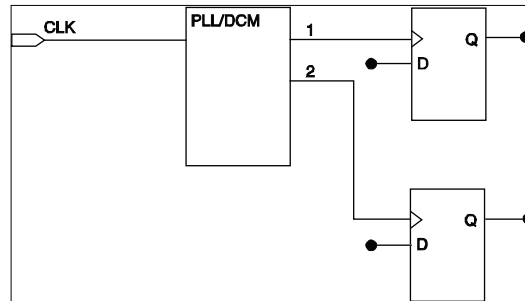


Xilinx's Trace timing analyzer does not have the capability to account for combinatorial loops. The ISE software ignores combinatorial loops and a warning is issued. Classic Timing Analyzer automatically accounts for all combinatorial loops by default and adds the delays to the total register-to-register  $f_{MAX}$ . If the user wants to remove undesired combinatorial loop paths from timing analysis, the Classic Timing Analyzer provides the option for timing analysis to ignore the false path.

### Designs With PLL/DCM

A Xilinx DCM or Altera PLL is used to provide signal de-skew or clock synthesis, such as clock multiplication, division or phase shifting. Clock synthesis affects the timing constraints placed on an FPGA because of different clock rates, clock relationships, or phases. There are necessary differences in constraining and reporting designs that use DCM/PLL (Figure 4), especially when using multiple outputs.

Figure 4. PLL/DCM Example



The Xilinx ISE software has only one way of applying timing constraints to a DCM, which is by doing so on its input. PLL constraints are normally set in the Quartus II software using the MegaWizard® Plug-In Manager for the `altpll` megafunction. Because the MegaWizard settings use the PLL input timing constraints by default, the user must be aware of them. It is important to have the same parameters in DCMs and PLLs such that the relationships between output clocks and input clocks are set the same way for both vendors. When applying constraints on the inputs of DCMs and PLLs, verify that their respective output clocks have equivalent constraints resulting from the parameters.

**Setup ( $t_{SU}$ ) and Clock-to-Out ( $t_{CO}$ ) Timing Analysis**

Because the Classic Timing Analyzer accounts for registered clock structures and Xilinx’s Trace does not, setup ( $t_{SU}$ ) and clock-to-out ( $t_{CO}$ ) timing analysis results are also different between the two tools. If the input or output register has a registered-clock structure preceding it, the Classic Timing Analyzer adds the microtiming parameter of the register to the external  $t_{SU}$  and  $t_{CO}$  timing measurement. Trace does not report  $t_{SU}$  or  $t_{CO}$  for this type of structure. For the example shown in Figure 5, the Classic Timing Analyzer will report  $t_{CO}$  for reg4 through the microtiming delay of the gate and reg1. Trace will not report the  $t_{CO}$  value.

Figure 5. I/O Timing Issue Example

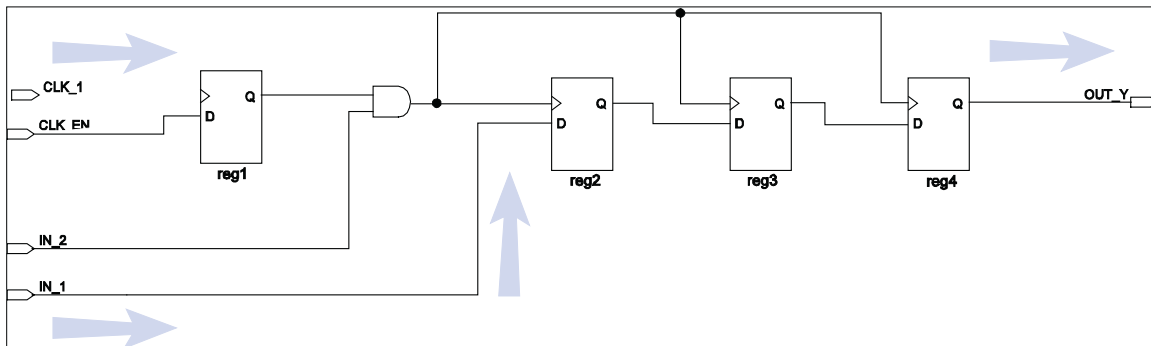


Table 1 summarizes the timing analysis differences between Xilinx Trace and Altera Classic Timing Analyzer.

Table 1. Timing Analysis Differences in Xilinx Trace and Altera Classic Timing Analyzer

Design Structure	Xilinx's Trace Timing Analyzer	Altera's Classic Timing Analyzer
Registered Clock	Not analyzed. Output can be a separate clock domain if properly constrained.	Always analyzed as part of the clock skew but can be turned off or analyzed as a separate clock domain if constrained the same way as in the ISE software.
Gated Clock	Analyzed only when constrained.	All signals related to clock ports are assumed to be clock signals.
Combinational Loop	Cannot be analyzed. Warning reported.	Analyzed by default but can be cut off by user if desired.
Designs with DCM/PLL	Analyzed if constraint assigned to input of DCM.	Constraint assigned in the PLL MegaWizard or to input of PLL.
Setup ( $t_{su}$ ) and Clock-to-Out ( $t_{co}$ )	Registered-clock preceding input or output registers not analyzed.	All worst-case structures analyzed.

## How to Perform Equivalent Timing Analysis

To achieve a fair comparison between the Classic Timing Analyzer and Xilinx's Trace, adjustments must be made to make the tools analyze the same paths in a design because the tools are based on different philosophies for optimizing performance and performing timing analysis. The user must constrain all relevant clocks in both tools. Constraining all clocks as separate domains is the approach taken for the purposes of this document, although a subset of clocks may also be constrained, as appropriate.

To gather information on clocks in the design, perform the following steps:

1. Perform an initial compilation using default values for both the ISE software and the Quartus II software.
2. Because the Quartus II software reports all possible paths by default, extract the clock names from the fitter report file by clicking on the Usage column of the Control Signals table (see [Figure 6](#)). Note that all signals that are used by registers as clock signals are listed in this table. Also, perform advanced timing analysis in Xilinx's Trace (`trace -a`) and extract the clock names from the report file.
3. Verify and correlate that the clock names used in the Quartus II software and ISE initial report file are the same.
4. After the clocks have been identified, apply  $f_{MAX}$  constraints to the clocks in both tools.

There may be clocks listed in the Fitter report file that do not appear in the Trace report file. These clocks must be constrained with a default value to ensure that the Classic Timing Analyzer does not automatically relate them.

Figure 6. Quartus II Fitter Report

Name	Location	Fan-Out	Usage
1 TopRx:TopRxPIIRx53m:PIRx53mPD_3PDRx53mlun1_PDRst_x	LCCOMB_X15_Y2_N16	2	Async. clear
2 TopRx:TopRxPIIRx77m:PIRx77mPD_2PDRx77mlun1_PDRst_x	LCCOMB_X15_Y2_N22	2	Async. clear
3 TopTx:TopTxPIITx103m:PIITx103mPD_1PDTx103mlun1_PDRst_x	LCCOMB_X12_Y7_N24	2	Async. clear
4 TopTx:TopTxPIITx77m:PIITx77mPDt_xlun1_PDRst_x	LCCOMB_X21_Y1_N0	2	Async. clear
5 mpint:MPIntreset_i	LCCOMB_X20_Y7_N28	21	Async. clear
6 ARsIF	PIN_87	1815	Async. clear
7 Clk26m	PIN_138	40	Clock
8 ClkRBC0	PIN_28	143	Clock
9 ClkRBC1	PIN_24	215	Clock
10 ClkRx53m	PIN_132	204	Clock
11 ClkRx77m	PIN_130	113	Clock
12 ClkTx103m	PIN_23	308	Clock
13 ClkTx53m	PIN_129	64	Clock
14 ClkTx77m	PIN_27	189	Clock
15 LineRefTx	PIN_164	1	Clock
16 SClk	PIN_131	173	Clock
17 SelClk:SelClk77mClkSel_x	LCCOMB_X15_Y8_N0	219	Clock
18 SelClk_1:SelClk53mTxClkSel_x	LCCOMB_X18_Y12_N6	13	Clock
19 SelClk_2:SelClk53mRxClkSel_x	LCCOMB_X20_Y11_N30	167	Clock
20 SerEn	PIN_92	14	Clock
21 TopRx:TopRxPIIRx53m:PIRx53mCount31Rx53m:Count31Rx53mClkL0L_Z	LCFF_X14_Y1_N9	1	Clock
22 TopRx:TopRxPIIRx53m:PIRx53mCount31Rx53m:Count31Rx53mPDRIFb53m	LCFF_X15_Y1_N5	5	Clock
23 TopRx:TopRxPIIRx53m:PIRx53mRfISlct_3:RfISlctRx53mPDRISlct_x	LCCOMB_X15_Y2_N28	1	Clock
24 TopRx:TopRxPIIRx77m:PIRx77mCount8Rx77m:Count8Rx77mClkL0L_Z	LCFF_X12_Y1_N17	1	Clock
25 TopRx:TopRxPIIRx77m:PIRx77mCount8Rx77m:Count8Rx77mPDRIFb77m_Z	LCFF_X13_Y1_N25	4	Clock
26 TopRx:TopRxPIIRx77m:PIRx77mRfISlct_2:RfISlctRx77mPDRISlct_x	LCCOMB_X15_Y2_N26	2	Clock
27 TopTx:TopTxPIITx103m:PIITx103mCount11Tx103m:Count11Tx103mClkL0L_Z	LCFF_X13_Y12_N13	1	Clock
28 TopTx:TopTxPIITx103m:PIITx103mCount11Tx103m:Count11Tx103mPDRIFb103m_Z	LCFF_X9_Y7_N23	4	Clock
29 TopTx:TopTxPIITx103m:PIITx103mRfISlct_1:RfISlctTx103mPDRISlct_x	LCCOMB_X12_Y7_N26	2	Clock
30 TopTx:TopTxPIITx77m:PIITx77mFracDivTx77m:FracDivTx77mClkL0L_Z	LCFF_X19_Y1_N25	1	Clock
31 TopTx:TopTxPIITx77m:PIITx77mFracDivTx77m:FracDivTx77mPDRIFb77m_Z	LCFF_X20_Y1_N21	4	Clock
32 TopTx:TopTxPIITx77m:PIITx77mRfISlct:RfISlctTx77mPDRISlct_x	LCCOMB_X21_Y1_N20	2	Clock
33 G_517	LCCOMB_X25_Y10_N4	8	Clock enable
34 TopRx:TopRxPIIRx53m:PIRx53mFracDivRx53m:FracDivRx53mIN_84_i	LCCOMB_X14_Y3_N28	1	Clock enable
35 TopRx:TopRxPIIRx77m:PIRx77mCount5Rx77m:Count5Rx77mSyncRf20	LCCOMB_X6_Y1_N12	1	Clock enable
36 TopRx:TopRxStm4Rx:Stm4RxGenFifoPtr_Generic_512s_9s_400s_512s_9s_0s:GenFifoPtrRxN_379_i	LCCOMB_X12_Y6_N28	9	Clock enable
37 TopRx:TopRxStm4Rx:Stm4RxIpbip12det:bp12detbeginframe	LCCOMB_X14_Y6_N18	24	Clock enable
38 TopRx:TopRxStm4Rx:Stm4RxIstm4ovhdext:stm4ovhdexticldloadsync_0_a2	LCCOMB_X14_Y9_N20	38	Clock enable
39 TopRx:TopRxStm4Rx:Stm4RxIstm4ovhdext:stm4ovhdexticldword_reg_1_sqmuxa~_DUP_COMB_4	LCCOMB_X17_Y9_N4	1	Clock enable

The following sections outline the steps needed to ensure that the tools analyze the same paths. These steps involve making constraints in the Quartus II software to match default ISE behavior and equivalent settings in the ISE software. The effectiveness of these recommendations may vary with the characteristics of the design used to measure benchmarks. An intimate knowledge of the design structure is necessary to reduce inequivalency.

### Correcting Registered Clock Analysis Differences

Xilinx’s ISE software accounts for registered-clock structures by treating original and derived clocks as separate clock domains. In addition, constraints applied to the original clocks do not constrain paths using these registered clocks. The Classic Timing Analyzer analyzes the relationships between the clocks by default, applying the  $f_{MAX}$  constraints to all registered versions of the clock, and analyzing the paths between them.

One of two approaches can be used to ensure equivalent timing analysis. One approach involves making additional constraints in the Quartus II software to break the relationships between clocks that the Classic Timing Analyzer automatically infers. The other approach involves explicitly declaring relationships between clocks in the ISE software to match the default timing analysis behavior in the Classic Timing Analyzer.

In the Quartus II software, use the Assignment Editor to create timing constraints for both the original and registered clock signals. For the example shown in Figure 1, apply the constraint to the CLK\_1 pin and the output of reg1.

Making these two clock settings breaks the relationship between the original clock and its derivative or registered version because each is treated as an independent clock. In the ISE software, use the Constraints Editor to assign timing constraints to both the original and registered clock signals to ensure analysis of the appropriate paths.

### *Correcting Gated Clock Analysis Differences*

For gated clocks, only constrain the original clock, which is the input to the gate. For the gated-clock example shown in [Figure 2](#), the constraint is applied to the CLK\_1 input pin. If that constraint is applied, both tools analyze the paths using both the original clock and its gated version as well as paths between the clocks.

To simplify the necessary changes, the user can constrain the original and gated clocks so that they are independent. This simplifies the procedure because the user does not need to understand the relationships between the original and derived clocks to make the appropriate settings. The same solution is sufficient for both registered and gated clocks. Because the ordering of constraints to the same paths is important when using the ISE software, the user must make the timing settings to the derived clock first.

### *Correcting Combinatorial Loop Analysis Differences*

Since the ISE software does not analyze paths for combinatorial loop structures, these paths need to be explicitly cut in the Quartus II software to perform equivalent analysis. To cut these paths, first identify the combinatorial loop structures, then choose the paths in the Assignment Editor and choose “cut timing path” for each one. The Quartus II timing analyzer ignores these paths.

### *Correcting Analysis for Designs With PLL/DCM*

To perform equivalent timing analysis for clocks generated by DCMs in ISE and PLLs in the Quartus II software, perform the following procedure. For ISE, assign a timing constraint to the input of the DCM. The ISE software applies this constraint to the output clocks and sets values according to the DCM clock frequency. In the Quartus II software, either assign the same constraint to the input of the PLL or verify that the input and output clock frequency parameters in the `altpll` MegaWizard causes the Quartus II software to constrain each output clock frequency to match the settings made in the ISE software. By constraining the design in this way, both tools analyze the paths between the PLL/DCM generated clocks.

### *Correcting ( $t_{SU}$ ) and Clock-to-Out ( $t_{CO}$ ) Analysis Differences*

Since the ISE software does not automatically determine the relationship between registered clocks and their original source, it does not calculate setup or clock-to-out time for any paths that use registered clock signals. To exclude the same paths in the Quartus II software, the solution is similar to the registered clock solution. Make the original and the derived clock independent by specifying a false path between them in the Quartus II software. Since gated clocks are handled in the same way for both tools, constraining setup and clock-to-out timing is all that is necessary.

## **Timing Analysis Correction Results**

Below are examples of before and after results of configuring the Quartus II software to perform equivalent timing analysis to the ISE software. In each of these cases the customer has constrained all clock signals in the Xilinx ISE tool. For the Quartus II software, there are two compilations:

- Default compilation, where the Quartus II software performs conservative worst-case analysis on all paths
- Modified timing analysis where the corrections described in the previous section of this document are applied to achieve a fair comparison of the two tools.



## Design 1

Table 2 shows Quartus II data for Design 1 after default compilation and after corrections are applied.

Table 2. Design 1 Timing Analysis Correction Example

Characteristic	Default Compilation	After Corrections Are Applied
Size	6,066 LEs	6,066 LEs
Clock 1 Maximum Frequency ( $f_{MAX}$ )	63.75 MHz	91.32 MHz
Clock 2 Maximum Frequency ( $f_{MAX}$ )	52.54 MHz	242.31 MHz

After the user configures the Quartus II software to perform timing analysis equivalent to the ISE software, both reported clock frequencies go up significantly. This design has many paths that go across derived clock domains and the Quartus II software analyzes them all by default, hence the lower  $f_{MAX}$ . The  $f_{MAX}$  measurement increases after the cross-clock paths are cut off from analysis. The critical path of Clock 2 after default compilation is 19.033 ns, of which 5.226 ns is clock skew. This skew is a result of a clock structure similar to that shown in Figure 2. After the clock relationships are broken, the critical path has two registers directly driven by Clock 2 with almost no clock skew.

## Design 2

By default, the Quartus II software accounts for all cross-domain analysis, assumes worst-case scenarios for gated-clock signals, and analyzes all paths that exist in between. This design also contains registered clocks. After the user removes cross domain analysis and assigns an internal node clock frequency to mimic ISE timing analysis, the clock frequencies increase and two internal clock signals are introduced (Table 3). Clocks 1, 4, and 8 are not reported because they are inputs to the same logic that produces Internal Clocks 1 and 2. Internal Clock 2 is generated by Clock 6 gated with Clock 5. Clock 6 is the pin clock and Clock 5 is no longer reported as a clock because its analysis is done using Internal Clock 2.

Table 3. Design 2 Timing Analysis Fix Example

Characteristic	Default Compilation	After Fixes Are Applied
Size	16,362 LEs	16,362 LEs
Clock 1 Maximum Frequency ( $f_{MAX}$ )	133.12 MHz	-
Clock 2 Maximum Frequency ( $f_{MAX}$ )	130.23 MHz	242.31 MHz
Clock 3 Maximum Frequency ( $f_{MAX}$ )	181.55 MHz	266.24 MHz
Clock 4 Maximum Frequency ( $f_{MAX}$ )	133.12 MHz	-
Clock 5 Maximum Frequency ( $f_{MAX}$ )	161.84 MHz	-
Clock 6 Maximum Frequency ( $f_{MAX}$ )	84.79 MHz	156.25 MHz
Clock 7 Maximum Frequency ( $f_{MAX}$ )	161.84 MHz	422.12 MHz
Clock 8 Maximum Frequency ( $f_{MAX}$ )	133.12 MHz	-
Internal Clock 1 Frequency ( $f_{MAX}$ )	-	132.8 MHz
Internal Clock 2 Frequency ( $f_{MAX}$ )	-	415.11 MHz
Number of Setup/Hold Time Violations	15	0

## Design 3

Design 3 contains registered clocks. Breaking the clock relationships eliminates reported clock skew. The netlist holds gated clocks and paths crossing between clock domains. After the user assigns individual clock signals, the performance of each clock signal increases, making fair performance comparison with ISE timing analysis possible (Table 4).



Table 4. Design 3 Timing Analysis Correction Example

Characteristic	Default Compilation	After Corrections Are Applied
Size	44,811 LEs	44,811 LEs
Clock 1 Maximum Frequency ( $f_{MAX}$ )	49.45 MHz	51.27 MHz
Clock 2 Maximum Frequency ( $f_{MAX}$ )	90.95 MHz	92.91 MHz
Clock 3 Maximum Frequency ( $f_{MAX}$ )	87.32 MHz	89.94 MHz

### Achieving Best Results Through Quartus II Automation

The Quartus II software includes a Tcl/Tk utility called the Design Space Explorer (DSE). DSE allows users to run multiple compilations with different seeds, physical synthesis settings, and other optimization options. DSE then reports the settings that provide the optimal results. DSE is ideal for running overnight benchmark tests to obtain potentially higher performance results. This utility is recommended for benchmarking because it requires very little user effort and in general produces higher performance results.


 For more information, refer to the *Design Space Explorer* chapter in Volume 2 of the *Quartus II Handbook*: [www.altera.com/literature/hb/qts/qts\\_qii52008.pdf](http://www.altera.com/literature/hb/qts/qts_qii52008.pdf).

### Synthesis EDA Tools Estimation

Most EDA synthesis tools today used in FPGA design provide area and performance estimation capabilities to allow users to quickly verify their design resources without having to compile in the FPGA place and route tools, which normally take longer to complete. Although Altera works very closely with EDA vendors, there are some inherent inaccuracies that make this comparison method unsuitable for performing resource and performance benchmark tests.

To obtain accurate comparisons, the design **must** be compiled using the FPGA vendor's place and route tool (Quartus II software or the ISE software) because:

- EDA tools only provide estimates, not actual data. Actual performance and area results can vary significantly after remapping and place-and-route. Having register packing performed by the Quartus II software, for example, reduces area utilization of not register packing by 11 percent. The ISE software also produces actual performance and device utilization data that differs from that provided by EDA synthesis tools.
- The accuracy of the estimates also depends on the existence of black boxes (containing Altera megafunctions or Xilinx CoreGen cores). EDA synthesis tools do not synthesize the black-boxed functions and cannot perform timing estimates on them.
- Xilinx and Altera resource estimations are not equivalent. Virtex-4, Virtex-II, Virtex-II Pro, and Spartan-3 have slices as a unit for logic blocks. Stratix, Stratix GX, Cyclone, and Cyclone II have logic elements (LEs), which have different configurations. Stratix II and Stratix II GX devices have ALUTs, which are different from slices and LEs. These three configurations are not equivalent, and the user must compile the design in either the Quartus II or ISE software to determine the actual performance of the design on the targeted device.
- Most FPGA EDA synthesis tools provide system  $f_{MAX}$  estimates instead of core  $f_{MAX}$ . The Quartus II software and the ISE software report core frequency.

 For more information, see *TB 84: Differences in Logic Utilization between Quartus II & Synplify Report Files*: [www.altera.com/literature/tb/tb84.pdf](http://www.altera.com/literature/tb/tb84.pdf).

### Summary

Each FPGA software tool has its own set of parameters and environment determined by the FPGA vendor to be optimum for their products. The Altera Quartus II software and the Xilinx ISE software have different default environments, causing out-of-the-box software benchmarking to occasionally produce non-equivalent comparisons, especially in large-density designs. Complex structures, such as gated clocks, registered clocks, combinational loops, and DLL/PLL usage are analyzed differently in the ISE software from the way they are analyzed in the Quartus II software.

In order to perform fair cross-vendor software benchmarking, some constraints and settings adjustments need to be made where applicable. The Quartus II timing analyzer performs complete and thorough analysis of all permutations of paths and assumes the worst-possible case when reporting timing analysis. The ISE software, by contrast, omits certain structures during timing analysis, reducing the total net delay. If Quartus II constraints are set to mimic those of the ISE software, performance improvement is seen for the Quartus II software.

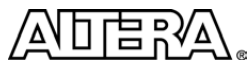
The Quartus II software and ISE software must be used to completely compile a design for accurate timing analysis because EDA synthesis tools alone do not provide a complete picture of timing in the device. To facilitate the performance of multiple full compilations, the Quartus II Design Space Explorer utility makes it easy to perform multiple compilations with different settings to improve performance beyond default values.

### Further Information

- *Design Space Explorer* chapter in Volume 2 of the *Quartus II Handbook*:  
[www.altera.com/literature/hb/qts/qts\\_qii52008.pdf](http://www.altera.com/literature/hb/qts/qts_qii52008.pdf)
- *TB 84: Differences in Logic Utilization between Quartus II & Synplify Report Files*:  
[www.altera.com/literature/tb/tb84.pdf](http://www.altera.com/literature/tb/tb84.pdf)

### Acknowledgements

- Kalen Brunham, Member of Technical Staff, Software and System Engineering, Altera Corporation
- Alexander Grbic, Manager, Software and System Engineering, Altera Corporation
- Carolyn Lam-Leventis, Sr. Software Engineer, Software and System Engineering, Altera Corporation
- Subianto Windoro, Product Marketing Manager, Technical Marketing, Altera Corporation



101 Innovation Drive  
San Jose, CA 95134  
[www.altera.com](http://www.altera.com)

Copyright © 2007 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries. All other product or service names are the property of their respective holders. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.