White Paper

Guidance for Accurately Benchmarking FPGAs

Introduction

This paper presents a rigorous methodology for accurately benchmarking the capabilities of an FPGA architecture. The goal of benchmarking is to compare the capabilities of one FPGA architecture versus another. Since the FPGA industry does not conform to a standard benchmarking methodology, this paper describes in detail the methodology employed by Altera and shows how a poor methodology can skew results and lead to false conclusions.

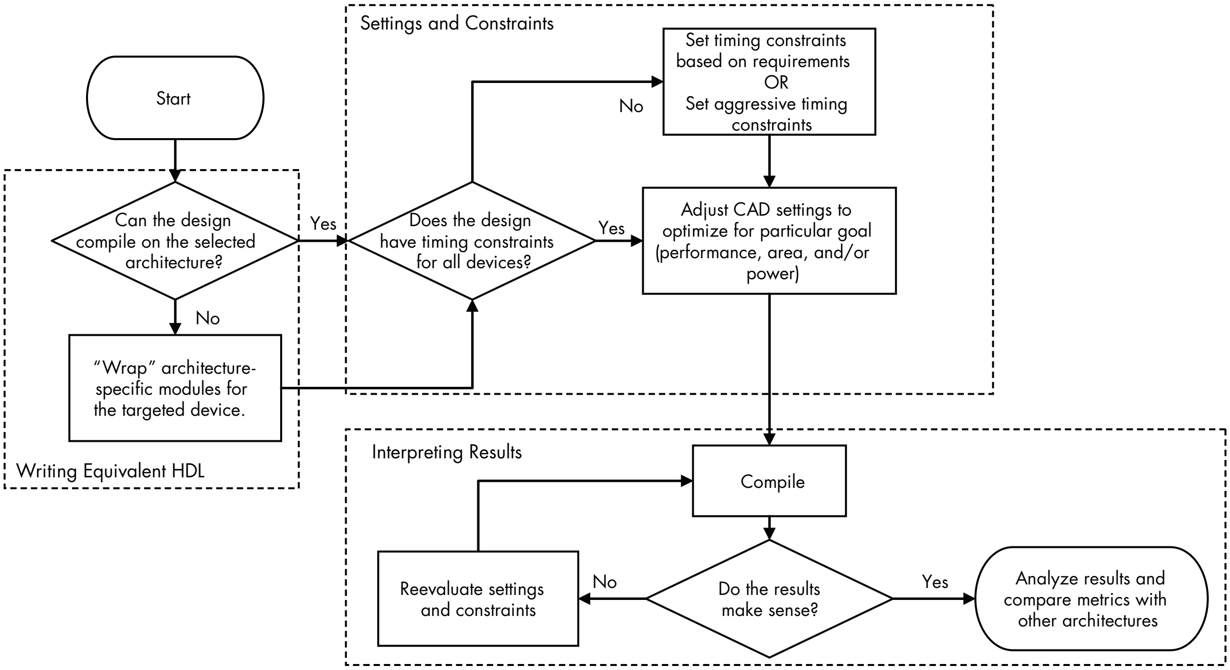
The complexity of today's designs together with the wealth of FPGA and computer-aided design (CAD) tool features available make benchmarking a difficult and expensive task. To obtain meaningful benchmarking results, a detailed understanding of the designs used in comparisons as well as an intimate knowledge of FPGA device features and CAD tools is required. A poor benchmarking process can easily result in inconclusive and, even worse, incorrect results. Altera invests significant resources to ensure the accuracy of its benchmarking results. This document guides the designer through performing a fair, accurate, and meaningful benchmarking analysis.

Benchmarking a Design

Customers benchmark to evaluate the best FPGA architecture for their design. As they gather data and form conclusions, there are considerations that must be made during the evaluation that can be difficult to determine. This paper describes how to benchmark a single design with confidence. It begins by ensuring an equivalent design is used in different FPGA architectures. Next, a methodology is presented that targets a balance of performance, area, and power, followed by guidance on how to evaluate each of those metrics. The last section presents ways to trade between performance, area, and power.

There are many factors that can affect benchmarking results, including how the design was written, CAD tool settings, design constraints, and interpretation of the results. This paper describes a methodology for benchmarking FPGA designs (shown as a flow chart in Figure 1) that takes these complex factors into consideration.

Figure 1. FPGA Benchmark Methodology Flow Chart


Settings and Constraints
Start
Set timing constraints based on requirements OR Set aggressive timing constraints
No
Can the design compile on the selected architecture?
Yes
Does the design have timing constraints for all devices?
Yes
Adjust CAD settings to optimize for particular goal (performance, area, and/or power)
No
"Wrap" architecture-specific modules for the targeted device.
Writing Equivalent HDL
Interpreting Results
Compile
Reevaluate settings and constraints
No
Do the results make sense?
Yes
Analyze results and compare metrics with other architectures

Writing Equivalent HDL for Different FPGA Architectures

Each FPGA architecture offers a suite of functions, including intellectual property (IP), as solutions to common applications embedded in a design. Customer designs may have taken advantage of such specific functions; however, they cannot be compiled in other FPGA architectures. A customer must pay attention when setting up their design for different architectures to ensure equivalent functionality for a fair comparison. Altera recommends that “wrappers” be introduced in the HDL written for the architecture-specific function, allowing them to be exchanged easily with functions supporting other FPGA architectures. The method of making wrappers is explained in this section and in the examples that follow.

The process of writing equivalent HDL for different FPGA architectures takes time so that each design can take full advantage of the dedicated features present in different architectures by:

1. Determining which modules of the design are architecture specific
2. Regenerating the modules to work for the other architectures being compared
3. Creating wrapper files for the modules to ensure that only architecture-specific modules of the design are different
4. Verifying that the functionality of the regenerated module is identical to the original one

Careful attention is required to make sure the HDL for the different FPGA architectures is functionally equivalent. After making architecture-specific updates to the HDL, the updates must be verified that they are equivalent to the original design. There are three ways to verify that the HDL of a design is set up in an equivalent manner for different FPGA architectures:

- Have testbenches exercise the updated areas of the design, run functional simulations with them, and verify that the simulations’ results are identical.
- If the testbenches are not yet available, compare the hierarchical resource counts for the different architectures that help catch problems with parts of a design being optimized away due to mistakes while making conversions. The hierarchy tab in Altera® Quartus® II design software’s Project Navigator provides a summary of resource information about the entities in the project. While this doesn’t examine any functionality, it can be helpful with debugging.
- If hierarchical summaries are not offered, verify that the architecture-specific updates give equivalent logic structures by using register transfer level (RTL) viewers.

Figure 2 is a simple example which shows a wrapper file for Altera (left) and one for Xilinx (right). Altera’s ROM block is generated by the Altera MegaWizard® Plug-In Manager. Xilinx’s ROM block is generated by Xilinx Core Generator tool. The dotted lines represent connections to an identical source module of a design. The wrapper file, represented by the dotted boxes, maps the vendor-specific port names of the ROM to the common port names of the source module in the main body of the design. Figure 3 presents the same example in HDL.

Figure 2. Using a Wrapper to Design for Different Vendors’ ROM Blocks

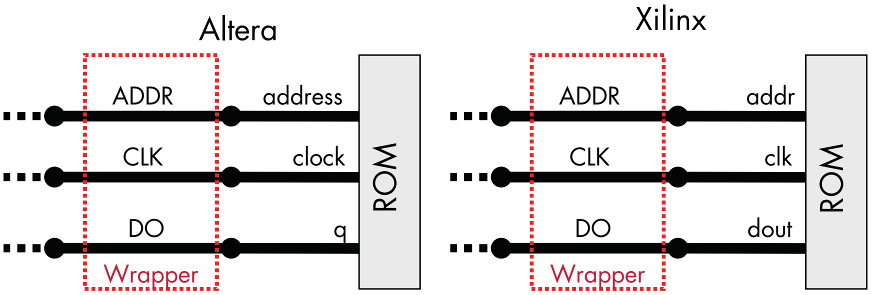
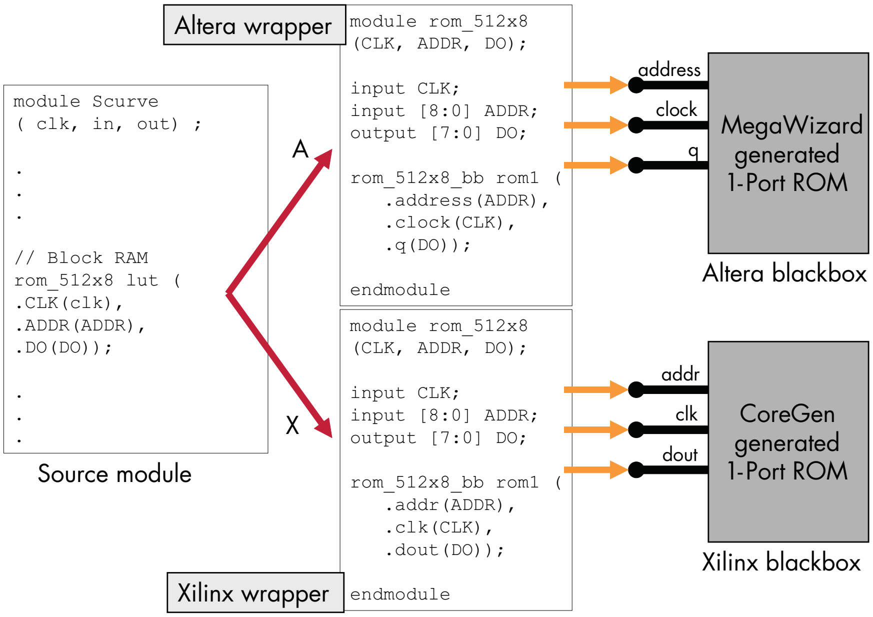


Figure 3. HDL of Wrappers Illustrated in Figure 2



```
module Scurve
( clk, in, out) ;

.
.
.

// Block RAM
rom_512x8 lut (
.CLK(clk),
.ADDR(ADDR),
.DO(DO));

.
.
.
```

```
module rom_512x8
(CLK, ADDR, DO);

input CLK;
input [8:0] ADDR;
output [7:0] DO;

rom_512x8_bb rom1 (
   .address(ADDR),
   .clock(CLK),
   .q(DO));

endmodule
```

```
module rom_512x8
(CLK, ADDR, DO);

input CLK;
input [8:0] ADDR;
output [7:0] DO;

rom_512x8_bb rom1 (
   .addr(ADDR),
   .clk(CLK),
   .dout(DO));

endmodule
```

If the design implements a memory block with an enable, a register may need to be implemented on Xilinx's memory block enable port. This additional register is used to resolve the enable latency differences between Altera and Xilinx, which can also be designed into the wrapper file. The next example illustrates how to add additional logic to make different vendor-specific blocks equivalent.

An Altera FPGA-specific structure that frequently occurs in customer designs is the phase-locked loop (PLL). To obtain equivalent functionality with Xilinx, the PLL is replaced with a module that sets up Xilinx's digital clock managers (DCM). To do this, Xilinx recommends instantiating global buffers with its DCM, which is equivalent to Altera routing its PLL clock output on global clock networks. The example in Figure 4 illustrates dotted lines to the same source module of a design that connects one of two wrapper files, depending on whether the target family is Altera or Xilinx. To the right of the Altera wrapper file, the PLL block and its ports are generated by the Altera MegaWizard Plug-In Manager. Core Generator is Xilinx's tool for its proprietary design modules (i.e., building RAM, ROM, and DSP functions); however, the DCM is directly instantiated in their HDL as shown below. In addition, the discrepancy of Xilinx not automatically placing DCM clocks on global networks is also handled in the wrapper file by having buffers surrounding the DCM. To manage the different port names, the wrapper file maps them to the common port names from the main body of the design. Figure 5 presents the same example in HDL.

Figure 4. Using a Wrapper to Design for Different Vendors' PLL/DCM Blocks

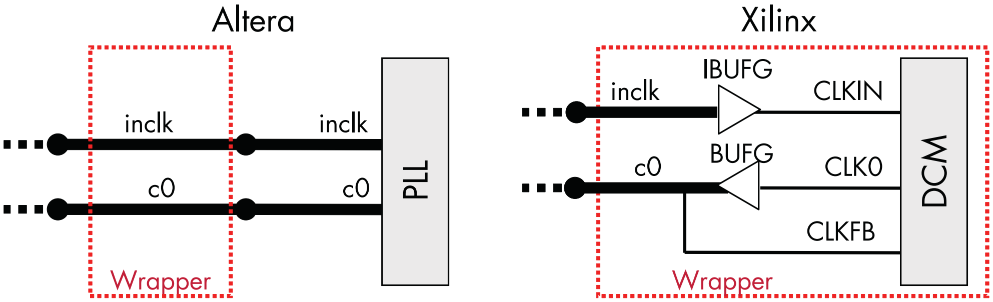
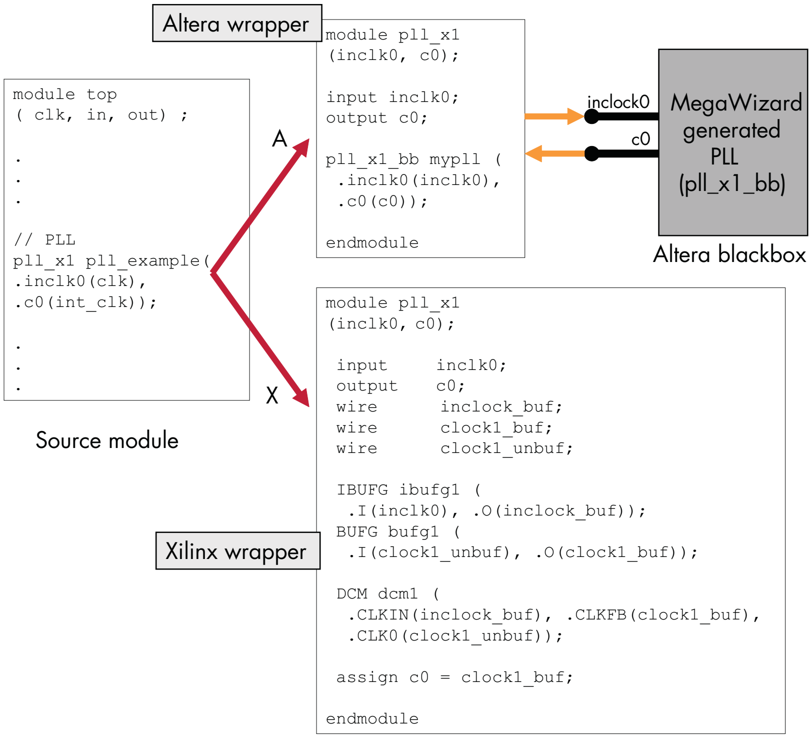


Figure 5. HDL of the Wrappers Illustrated in Figure 4



```
module top
( clk, in, out) ;

.
.
.

// PLL
pll_x1 pll_example(
.inclk0(clk),
.c0(int_clk));

.
.
.
```

```
module pll_x1
(inclk0, c0);

input inclk0;
output c0;

pll_x1_bb mypll (
 .inclk0(inclk0),
 .c0(c0));

endmodule
```

```
module pll_x1
(inclk0, c0);

 input     inclk0;
 output    c0;
 wire      inclock_buf;
 wire      clock1_buf;
 wire      clock1_unbuf;

 IBUFG ibufg1 (
  .I(inclk0), .O(inclock_buf));
 BUFG bufg1 (
  .I(clock1_unbuf), .O(clock1_buf));

 DCM dcm1 (
  .CLKIN(inclock_buf), .CLKFB(clock1_buf),
  .CLK0(clock1_unbuf));

 assign c0 = clock1_buf;

endmodule
```

If the design requires vendor-specific IP, and the equivalent IP for other vendors is not readily available, a possible work-around for benchmarking the rest of the design is to replace the IP module with a module that acts as a placeholder. This placeholder should be designed so that no part of the design is optimized away and it is not on a critical path.

Contact an Altera FAE for a parameter-enabled, port-compatible module that can be set up as a placeholder for the IP module.

Software Settings and Timing Constraints

All CAD tools offer settings that provide trade-offs among design performance, logic resource consumption, power requirements, compile time, and memory usage. The settings that produce the best results for one design may not produce the best results for another. In addition, results can be improved by providing user constraints to guide the CAD tool. In this section, device settings are discussed, and then two types of setting and constraints guidance are provided: one for a design that already has settings and constraints, and another for a design that does not yet have settings and constraints.

Device selection is critical in ensuring fair and accurate benchmarking results. For each device family being compared, choose the smallest device in which the design can be successfully implemented. Customers typically do this to minimize the FPGA cost. The smallest device into which a design can fit is selected based on the logic resources and availability of dedicated features, such as memory and multipliers, that a design needs. Arbitrarily picking a larger device can affect CAD optimizations, rendering results inconclusive.

If the design is a small sample one, occupying a fraction of the smallest device available in a given FPGA family, use the device size that is being evaluated. Be sure to select equivalent-sized parts from the other FPGAs being examined.

Refer to Altera's white paper, *Stratix III FPGAs vs. Xilinx Virtex-5 Devices: Architecture and Performance Comparison*, to select equivalent density devices.

Designs With Existing Settings and Constraints

If the design being evaluated already has design settings and constraints, then apply them consistently for all architectures being compared. For example, consider a design with its original version being optimized in a best performance mode. It is important to ensure that when this design is compiled in different architectures, it is similarly constrained in a best performance mode.

Be sure that if optimization options are set, then compilations for other architectures similarly have those options set. If the design has timing constraints, then be aware that evaluating the results of the constrained paths would just be examining slack results to check whether the design meets the timing constraints. CAD tools will not try to build extra margin when timing constraints are provided, hence comparing the numeric values of timing results that pass their constraints would be inconclusive.

Refer to Altera's white paper, *Performing Equivalent Timing Analysis Between the Classic Timing Analyzer and Xilinx Trace*, for best practices in making timing settings to obtain similar timing analysis between the vendors.

Designs Without Settings or Constraints

If the design's timing specifications are not yet determined, or if maximum performance must be measured, try a timing-constrained comparison with default CAD settings and aggressive timing constraints:

1. Make clock assignments on pin clocks to keep constraining the design simple.

2. To determine aggressive timing constraints for a design, a maximum frequency (f_{MAX}) constraint is applied to each clock in a given design such that the constraint is just beyond what is achievable for each clock. The best constraint is determined by increasing the constraint until it cannot be met.

   - In Altera's Quartus II design software, this can easily be done by over-constraining the f_{MAX} of each clock to 1 GHz.
   - For Xilinx, over-constraining results in compilation failure because it recognizes that such a requirement is impossible to meet. Thus, the best method to evaluate Xilinx's performance is through iterative compiles of increasing f_{MAX} settings until all clocks results are just below the constraints. This method is valid for Altera, but not necessary.
   - If the design has a PLL, Altera uses the f_{MAX} setting made in the PLL MegaWizard Plug-In as the timing constraint. To make timing constraint changes for PLL clocks, use the MegaWizard to apply the constraint as the f_{MAX} setting of the input clock, or set it on the input of the PLL, which propagates the constraint to the PLL outputs, multiplied by the appropriate factors. Xilinx's DCM output clocks use the timing settings applied to the DCM input clock.

3. Apply I/O constraints, but beware of its trade-off with core performance. When optimizing for core performance, set loose input and output delays so that the CAD tools evaluate I/O timing. Without I/O constraints, performance results may be unrealistic, because the CAD tools maximize the core performance without factoring in any trade-offs with I/O timing requirements.

4. Beware of how different timing analyzers have different methods of tracing complex paths. To ensure only the interesting paths are analyzed, cut cross clock domain paths. Altera's TimeQuest timing analyzer requires users to make these settings explicitly. One way to cut cross clock domain paths in TimeQuest is to put each clock into a separate clock group (e.g., have this set in the SDC file: set_clock_groups -exclusive -group {clk1} -group {clk2 clk_others})

5. Verify the constraints by checking the timing analyzer's report. Refer to the "Timing Analysis Comparison of Software Tools" section to ensure that the design is fully constrained correctly. Table 1 compares the timing analyses performed by the Xilinx and Altera tools.

Table 1. Timing Analysis Differences in Xilinx Trace and Altera TimeQuest Timing Analyzers

Design Structure	Xilinx Trace Timing Analyzer	Altera TimeQuest Timing Analyzer
Default Cross-Domain Clock Analysis	Register-to-register paths clocked by different clocks are not analyzed. (Exception: clocks from the same DCM)	Identified clocks are related and all paths of registers fed by them are analyzed.
Combinational Loop	Loops are not guaranteed to be analyzed. Warning reported.	The longest and shortest paths of all loops are analyzed for all inputs to all outputs. Optionally, such paths can be ignored.
Designs With DCMs/PLLs	Analyzed if the constraint is applied to the input of a DCM.	Analyzed when the derive_pll_clocks SDC setting is used (done by default) to obtain PLL clock settings made in the PLL function.*(1)*

Note:

(1) To use the PLL input frequency setting made in the MegaWizard Plug-In Manager for the altpll function as the PLL input clock constraint, use derive_pll_clocks -create_base_clocks option.

More details on the design structures listed in Table 1 are discussed in Altera's white paper, *Performing Equivalent Timing Analysis Between Altera TimeQuest and Xilinx Trace*.

Interpreting Results

Successfully compiling a design in different architectures is only the beginning of benchmarking. It is the verification of the results that dictates how meaningful the benchmark is. This section covers how to examine timing, area, and power results.

Timing Analysis Comparison of Software Tools

Interpretation of results is critical. Simply comparing the results from Quartus II and ISE software without paying detailed attention to the differences in timing analysis of each tool can easily produce misleading results. This subsection presents how to correctly interpret timing analysis results.

When evaluating timing analysis results, special consideration must be taken to ensure that timing metrics are extracted and compared correctly. Important timing metrics are:

- f_{MAX} or inversely minimum period
- Input pin setup delay
- Clock to output pin delay
- If timing specifications were made, the slacks of all the metrics listed above are useful.

When comparing timing results of different architectures, it is imperative to verify that the reporting is the same in both architectures:

1. *Timing constraints* - Begin by checking if the timing constraints applied are being reflected in the timing reports.

2. *Unconstrained paths* - Timing analysis tools are capable of producing report summaries that indicate which paths are unconstrained. The TimeQuest Tcl command is report_ucp -panel_name "Unconstrained Paths." Review these reports and make sure they are comparable between the architectures.

3. *Special clock structures* - If there are special clock structures in the design, check that the analyses for paths using those clocks are the same for both architectures. If not, revisit the previous section on equivalently constraining different architectures.

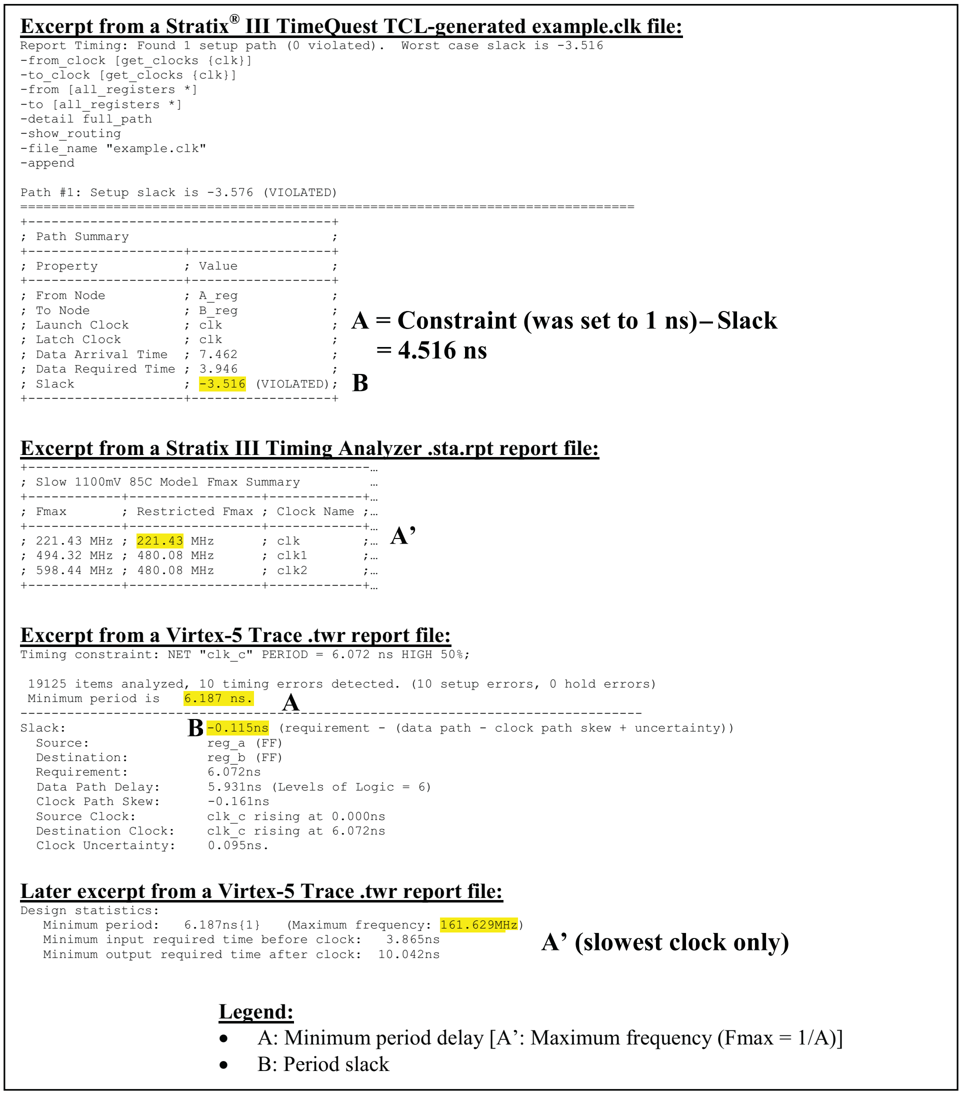
4. *Verify setup analysis* - Check that setup analysis was done the same way for both architectures as it has an impact on how the CAD optimizes timing delays.

5. *Verify hold analysis* - As with setup analysis, hold analysis also impacts how the CAD optimizes timing delays, so make sure that the hold analysis was done the same way for both architectures.

Timing reports from Altera and Xilinx are illustrated in Figure 6, Figure 7, and Figure 8, with common metrics labeled with the same letter. The Altera reports are generated by running TCL scripts in TimeQuest. The TimeQuest TCL commands are included near the top of each figure. Xilinx's reports are generated with Trace timing analysis tool. The important timing metrics are highlighted in each of these figures.

Altera introduced TimeQuest in Quartus II software v.6.0. The TimeQuest timing analyzer is an ASIC-strength timing analyzer that supports the industry-standard Synopsys Design Constraints (SDC) format. All discussions in this section are based on TimeQuest since this is Altera's timing analyzer for new device families.

Figure 6. Stratix III TimeQuest and Virtex-5 Trace f_{MAX} Timing Reports Compared



Excerpt from a Stratix® III TimeQuest TCL-generated example.clk file:

```
Report Timing: Found 1 setup path (0 violated).  Worst case slack is -3.516
-from_clock [get_clocks {clk}]
-to_clock [get_clocks {clk}]
-from [all_registers *]
-to [all_registers *]
-detail full_path
-show_routing
-file_name "example.clk"
-append

Path #1: Setup slack is -3.576 (VIOLATED)
===============================================================================
+---------------------------------------+
; Path Summary                          ;
+--------------------+------------------+
; Property           ; Value            ;
+--------------------+------------------+
; From Node          ; A_reg            ;
; To Node            ; B_reg            ;
; Launch Clock       ; clk              ;  A = Constraint (was set to 1 ns)– Slack
; Latch Clock        ; clk              ;      = 4.516 ns
; Data Arrival Time  ; 7.462            ;
; Data Required Time ; 3.946            ;
; Slack              ; -3.516 (VIOLATED);  B
+--------------------+------------------+
```

Excerpt from a Stratix III Timing Analyzer .sta.rpt report file:

```
+--------------------------------------------…
; Slow 1100mV 85C Model Fmax Summary         …
+------------+-----------------+------------+…
; Fmax       ; Restricted Fmax ; Clock Name ;…
+------------+-----------------+------------+…  A'
; 221.43 MHz ; 221.43 MHz      ; clk        ;…
; 494.32 MHz ; 480.08 MHz      ; clk1       ;…
; 598.44 MHz ; 480.08 MHz      ; clk2       ;…
+------------+-----------------+------------+…
```

Excerpt from a Virtex-5 Trace .twr report file:

```
Timing constraint: NET "clk_c" PERIOD = 6.072 ns HIGH 50%;

 19125 items analyzed, 10 timing errors detected. (10 setup errors, 0 hold errors)
 Minimum period is   6.187 ns.   A
------------------------------------------------------------------------------
Slack:                B -0.115ns (requirement - (data path - clock path skew + uncertainty))
  Source:               reg_a (FF)
  Destination:          reg_b (FF)
  Requirement:          6.072ns
  Data Path Delay:      5.931ns (Levels of Logic = 6)
  Clock Path Skew:      -0.161ns
  Source Clock:         clk_c rising at 0.000ns
  Destination Clock:    clk_c rising at 6.072ns
  Clock Uncertainty:    0.095ns.
```

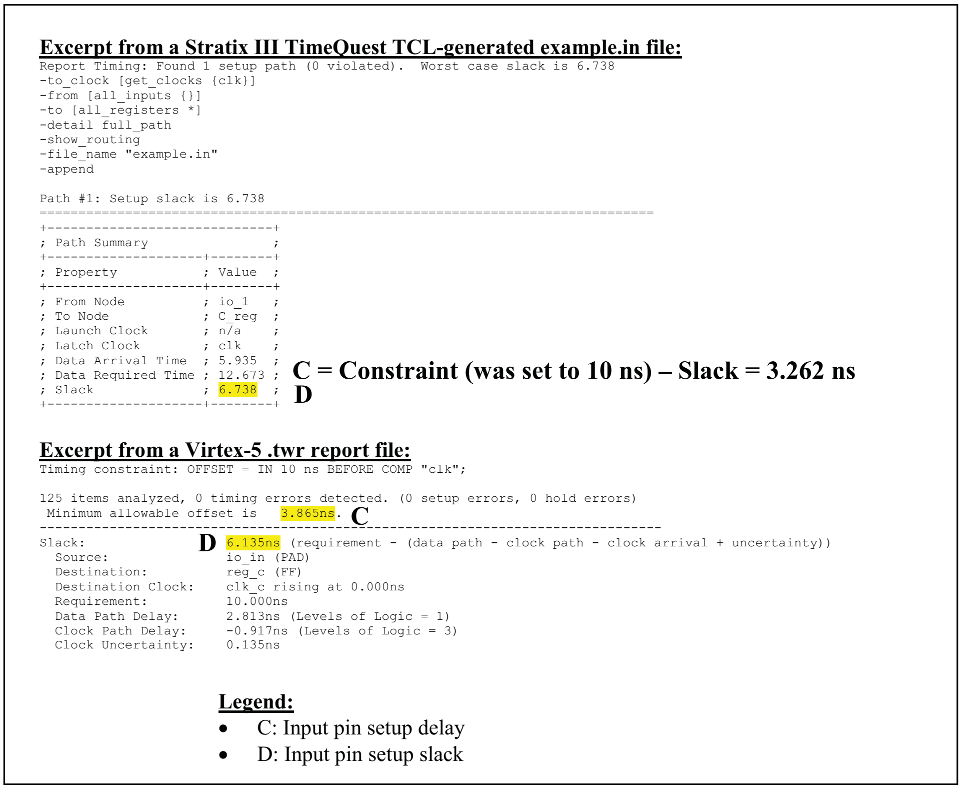
Later excerpt from a Virtex-5 Trace .twr report file:

```
Design statistics:
   Minimum period:   6.187ns{1}   (Maximum frequency: 161.629MHz)
   Minimum input required time before clock:   3.865ns           A' (slowest clock only)
   Minimum output required time after clock:  10.042ns
```

Legend:

- A: Minimum period delay [A': Maximum frequency (Fmax = 1/A)]
- B: Period slack

Figure 7. Stratix III TimeQuest and Virtex-5 Trace Input Pin Setup Delay Timing Reports Compared



Excerpt from a Stratix III TimeQuest TCL-generated example.in file:

```
Report Timing: Found 1 setup path (0 violated).  Worst case slack is 6.738
-to_clock [get_clocks {clk}]
-from [all_inputs {}]
-to [all_registers *]
-detail full_path
-show_routing
-file_name "example.in"
-append

Path #1: Setup slack is 6.738
===============================================================================
+-----------------------------+
; Path Summary                ;
+--------------------+--------+
; Property           ; Value  ;
+--------------------+--------+
; From Node          ; io_1   ;
; To Node            ; C_reg  ;
; Launch Clock       ; n/a    ;
; Latch Clock        ; clk    ;
; Data Arrival Time  ; 5.935  ;
; Data Required Time ; 12.673 ;  C = Constraint (was set to 10 ns) – Slack = 3.262 ns
; Slack              ; 6.738  ;  D
+--------------------+--------+
```

Excerpt from a Virtex-5 .twr report file:

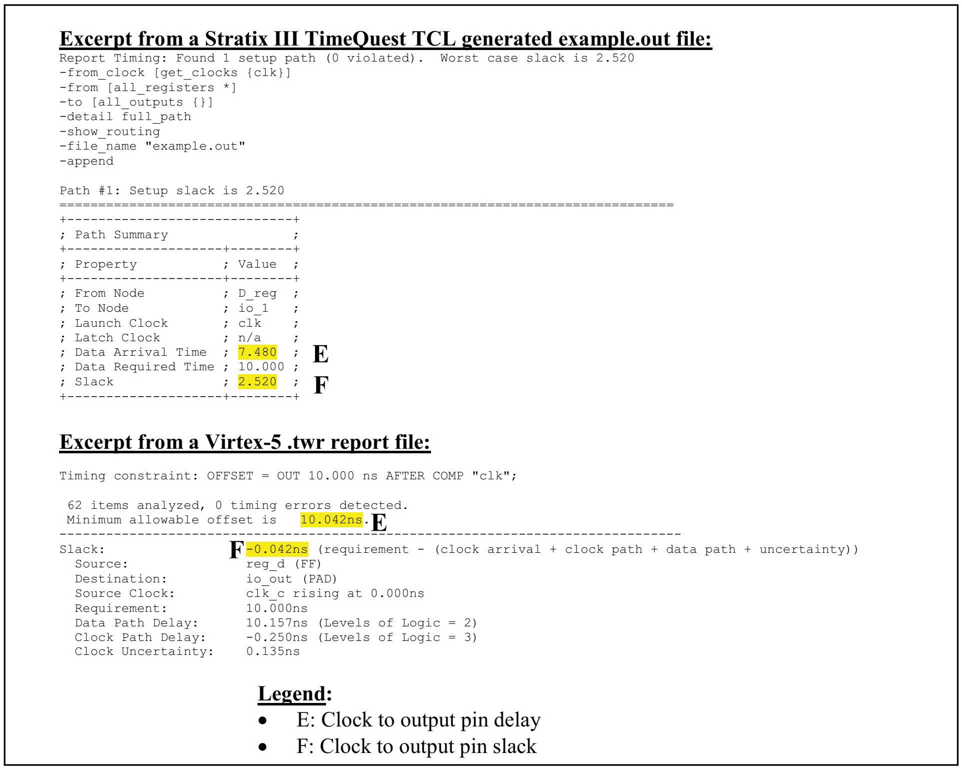
```
Timing constraint: OFFSET = IN 10 ns BEFORE COMP "clk";

125 items analyzed, 0 timing errors detected. (0 setup errors, 0 hold errors)
 Minimum allowable offset is   3.865ns. C
--------------------------------------------------------------------------------
Slack:                D 6.135ns (requirement - (data path - clock path - clock arrival + uncertainty))
  Source:               io_in (PAD)
  Destination:          reg_c (FF)
  Destination Clock:    clk_c rising at 0.000ns
  Requirement:          10.000ns
  Data Path Delay:      2.813ns (Levels of Logic = 1)
  Clock Path Delay:     -0.917ns (Levels of Logic = 3)
  Clock Uncertainty:    0.135ns
```

Legend:

- C: Input pin setup delay
- D: Input pin setup slack

Figure 8. Stratix III TimeQuest and Virtex-5 Trace Clock-to-Out Timing Reports Compared



Excerpt from a Stratix III TimeQuest TCL generated example.out file:

```
Report Timing: Found 1 setup path (0 violated).  Worst case slack is 2.520
-from_clock [get_clocks {clk}]
-from [all_registers *]
-to [all_outputs {}]
-detail full_path
-show_routing
-file_name "example.out"
-append

Path #1: Setup slack is 2.520
===============================================================================
+-----------------------------+
; Path Summary                ;
+--------------------+--------+
; Property           ; Value  ;
+--------------------+--------+
; From Node          ; D_reg  ;
; To Node            ; io_1   ;
; Launch Clock       ; clk    ;
; Latch Clock        ; n/a    ;
; Data Arrival Time  ; 7.480  ;  E
; Data Required Time ; 10.000 ;
; Slack              ; 2.520  ;  F
+--------------------+--------+
```

Excerpt from a Virtex-5 .twr report file:

```
Timing constraint: OFFSET = OUT 10.000 ns AFTER COMP "clk";

 62 items analyzed, 0 timing errors detected.
 Minimum allowable offset is   10.042ns.E
--------------------------------------------------------------------------------
Slack:                F-0.042ns (requirement - (clock arrival + clock path + data path + uncertainty))
  Source:               reg_d (FF)
  Destination:          io_out (PAD)
  Source Clock:         clk_c rising at 0.000ns
  Requirement:          10.000ns
  Data Path Delay:      10.157ns (Levels of Logic = 2)
  Clock Path Delay:     -0.250ns (Levels of Logic = 3)
  Clock Uncertainty:    0.135ns
```

Legend:

- E: Clock to output pin delay
- F: Clock to output pin slack

As discussed in the "Software Settings and Timing Constraints" section, a design may already have its timing constraints specified. If not, then settings for determining maximum performance have been outlined. The method in which timing results are evaluated differs between having a design constrained by its specifications and setting the design to do the best it can.

Designs With Timing Constraints Based on Specifications

The goal for the CAD tool is to meet the requirements. Evaluation here is based on whether all slacks are positive, and not on the value of the delays. Therefore, do not compare whether one tool gave a better result based on the f_{MAX}. One f_{MAX} result being better than the other is not an indication of the architecture capability, and is just a result of when the CAD tool decided to stop optimizing after meeting the requested constraint setting. The proper comparison is to examine any negative slacks, evaluate how difficult it is for the architecture to successfully implement a design, and examine the domains that have negative slack and the sum (total) of the negative slack (TNS).

Designs Constrained to Achieve Maximum Performance

The goal for the CAD tool is to do the best it can. In this area, evaluate the value of the delays. Since the constraints have been set to be beyond achievable in order to push CAD tools to achieve the best performance, the slacks are likely to be mostly negative and not interesting to examine. When evaluating the delays, beware of high-speed clocks and clocks with low fanout. These types of clocks can have large variance, and thus are not meaningful in comparisons.

Summarizing the results can be overwhelming, especially for designs with multiple clock domains. While it is important to look at individual clocks one by one, evaluating all of them together can help give an overview of the timing results of a design:

- Pick out the domains that are meaningful, and ignore clocks with low fanout and very high speed, which can dilute the quality of the evaluation.
- For metrics that have large positive number results, such as f_{MAX}, the geometric mean is a good method of combining the results of multiple clocks into a single value. The geometric mean values can then be compared with a ratio.
- For negative numbers results or results converging to zero, such as slacks, arithmetic mean is appropriate. When comparing the arithmetic mean as a percentage, make sure that function is symmetric by ensuring that reversing the values in the function gives an equal and opposite result. One such function is *(A-B)/min(A,B)*.

If the design is not meeting performance requirements, then read the "Improving Performance" section. Additional settings for maximum performance are discussed, along with logic resource, power, and compile time changes to expect when using these settings.

Logic Resource Comparisons

While timing comparisons are complicated by different timing analyzer behavior to timing constraints, logic resource comparisons are complicated by the unit used to compare logic. Figure 9 illustrates resource usage tables from the Stratix III **.fit.rpt** and Virtex-5 **.par** report files labeled for common metrics with the same letter. This is followed by Table 2, which lists other Altera FPGA families' resources to analyze. The logic resource metrics that are different between architectures are:

Combinational logic

- A pair of Altera's adaptive look-up table (ALUT) can be configured as a 6-LUT, two 4-LUTs, or other combinations of up to two functions with seven or fewer inputs and Virtex-5's LUT is a 6-LUT.
- Typically, a design requires fewer adaptive logic modules (ALMs) than Virtex-5's LUT/flipflop pairs because more functions can be packed into an ALM. Through comparison, it is interesting to see how much more efficient one LUT structure is to another. If the design is logic heavy, it may be more useful to focus on percentage of device utilization, which is provided on the same line as combinational logic count.

Refer to Altera’s white paper, *Stratix III FPGAs vs. Xilinx Virtex-5 Devices: Architecture and Performance Comparison*.

Logic-register unit

- Altera’s ALM has two ALUTs and two registers. Xilinx’s unit is a 6-LUT and a register.
- It is difficult to compare these because not only are the units different, but this metric is affected by the packing of the registers with logic.

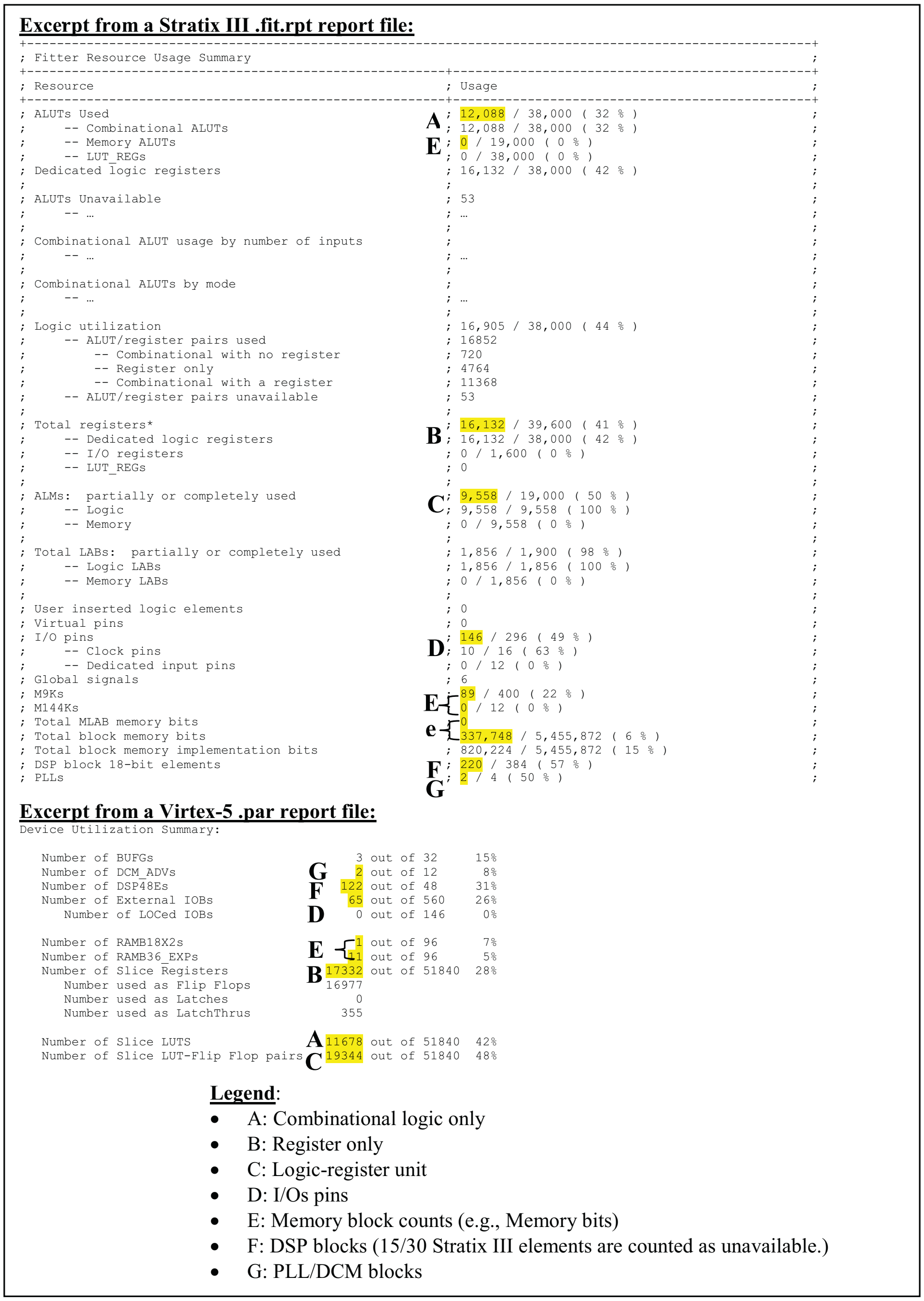
Memory blocks

- The block sizes are different: Stratix® III FPGAs offer 9-Kbit blocks, 144-Kbit blocks, and memory implemented in ALUTs (called MLABs); Virtex-5 offers pairs of 18-Kbit blocks, 36-Kbit blocks, and memory in its Slice logic detailed in Virtex-5’s Map report.
- The ideal way to compare memory utilization is by memory bits, which is labeled in the Stratix III report. Xilinx has no report of memory bits implemented by the design. (This is not to be confused with “Total Memory used (KB)” in Xilinx’s Map report, which includes the bits in the full block despite being only partially occupied.) Comparing memory usage requires a closer look at the design’s implementation of the memory blocks.

Digital signal processing (DSP) blocks

- Beware that multiplier/accumulator functionality can be implemented in logic, and this typically occurs when all DSP blocks are used and there is enough logic left to do so. Comparing DSP block usage may require a closer look at the design’s implementation.
- The sizes of DSP blocks reported are different: With Stratix III FPGAs, Altera reports the number of its occupied 18-bit DSP elements (including elements not available), while Virtex-5 can have up to 18-bit x 25-bit multipliers implemented in its 48-bit DSP element.

Figure 9. Stratix III and Virtex-5 Logic Resource Report Tables Compared



Excerpt from a Stratix III .fit.rpt report file:

```
+---------------------------------------------------------------------------------------------------------+
; Fitter Resource Usage Summary                                                                           ;
+--------------------------------------------------------+------------------------------------------------+
; Resource                                               ; Usage                                          ;
+--------------------------------------------------------+------------------------------------------------+
; ALUTs Used                                             ; 12,088 / 38,000 ( 32 % )                       ;
;     -- Combinational ALUTs                             ; 12,088 / 38,000 ( 32 % )                       ;
;     -- Memory ALUTs                                    ; 0 / 19,000 ( 0 % )                             ;
;     -- LUT_REGs                                        ; 0 / 38,000 ( 0 % )                             ;
; Dedicated logic registers                              ; 16,132 / 38,000 ( 42 % )                       ;
;                                                        ;                                                ;
; ALUTs Unavailable                                      ; 53                                             ;
;     -- …                                               ; …                                              ;
;                                                        ;                                                ;
; Combinational ALUT usage by number of inputs           ;                                                ;
;     -- …                                               ; …                                              ;
;                                                        ;                                                ;
; Combinational ALUTs by mode                            ;                                                ;
;     -- …                                               ; …                                              ;
;                                                        ;                                                ;
; Logic utilization                                      ; 16,905 / 38,000 ( 44 % )                       ;
;     -- ALUT/register pairs used                        ; 16852                                          ;
;         -- Combinational with no register              ; 720                                            ;
;         -- Register only                               ; 4764                                           ;
;         -- Combinational with a register               ; 11368                                          ;
;     -- ALUT/register pairs unavailable                 ; 53                                             ;
;                                                        ;                                                ;
; Total registers*                                       ; 16,132 / 39,600 ( 41 % )                       ;
;     -- Dedicated logic registers                       ; 16,132 / 38,000 ( 42 % )                       ;
;     -- I/O registers                                   ; 0 / 1,600 ( 0 % )                              ;
;     -- LUT_REGs                                        ; 0                                              ;
;                                                        ;                                                ;
; ALMs:  partially or completely used                    ; 9,558 / 19,000 ( 50 % )                        ;
;     -- Logic                                           ; 9,558 / 9,558 ( 100 % )                        ;
;     -- Memory                                          ; 0 / 9,558 ( 0 % )                              ;
;                                                        ;                                                ;
; Total LABs:  partially or completely used              ; 1,856 / 1,900 ( 98 % )                         ;
;     -- Logic LABs                                      ; 1,856 / 1,856 ( 100 % )                        ;
;     -- Memory LABs                                     ; 0 / 1,856 ( 0 % )                              ;
;                                                        ;                                                ;
; User inserted logic elements                           ; 0                                              ;
; Virtual pins                                           ; 0                                              ;
; I/O pins                                               ; 146 / 296 ( 49 % )                             ;
;     -- Clock pins                                      ; 10 / 16 ( 63 % )                               ;
;     -- Dedicated input pins                            ; 0 / 12 ( 0 % )                                 ;
; Global signals                                         ; 6                                              ;
; M9Ks                                                     89 / 400 ( 22 % )                              ;
; M144Ks                                                   0 / 12 ( 0 % )                                 ;
; Total MLAB memory bits                                   0                                              ;
; Total block memory bits                                  337,748 / 5,455,872 ( 6 % )                    ;
; Total block memory implementation bits                 ; 820,224 / 5,455,872 ( 15 % )                   ;
; DSP block 18-bit elements                              ; 220 / 384 ( 57 % )                             ;
; PLLs                                                   ; 2 / 4 ( 50 % )                                 ;
```

Excerpt from a Virtex-5 .par report file:

```
Device Utilization Summary:

   Number of BUFGs                           3 out of 32     15%
   Number of DCM_ADVs                        2 out of 12      8%
   Number of DSP48Es                       122 out of 48     31%
   Number of External IOBs                  65 out of 560    26%
      Number of LOCed IOBs                   0 out of 146     0%

   Number of RAMB18X2s                       1 out of 96      7%
   Number of RAMB36_EXPs                    11 out of 96      5%
   Number of Slice Registers             17332 out of 51840  28%
      Number used as Flip Flops          16977
      Number used as Latches                 0
      Number used as LatchThrus            355

   Number of Slice LUTS                  11678 out of 51840  42%
   Number of Slice LUT-Flip Flop pairs   19344 out of 51840  48%
```

Legend:

- A: Combinational logic only
- B: Register only
- C: Logic-register unit
- D: I/Os pins
- E: Memory block counts (e.g., Memory bits)
- F: DSP blocks (15/30 Stratix III elements are counted as unavailable.)
- G: PLL/DCM blocks

Table 2. List of Other Altera FPGA Resources to Analyze

Family	Resource	Label in Fitter Resource Usage Summary
Stratix II FPGA	Combinational logic	ALUTs used
	Registers	Total registers*
	ALMs	ALMs: partially or completely used
	Memory blocks	M512s, M4Ks, M-RAMs, Total block memory bits
	DSP blocks	DSP block 9-bit elements
Cyclone® III FPGA	Combinational logic	Combinational with no register + Combinational with a register
	Registers	Total registers*
	Logic elements (LEs)	Total logic elements
	Memory blocks	M9Ks, Total memory bits
	Embedded multipliers	Embedded multiplier 9-bit elements
Cyclone II FPGA	Combinational logic	Combinational with no register + Combinational with a register
	Registers	Total registers*
	LEs	Total logic elements
	Memory blocks	M4Ks, Total memory bits
	Embedded multipliers	Embedded multiplier 9-bit elements
MAX® II CPLD	Combinational logic	Combinational with no register + Combinational with a register
	Registers	Total registers*
	LEs	Total logic elements
		(MAX II CPLD does not have memory or multiplier blocks)

As discussed earlier about combinational logic, the ratios of how much logic can be synthesized into the different logic resource structures of FPGA architectures can be determined, provided that the benchmark has been carefully set up for the different architectures. (See Altera's white paper, *Stratix III FPGAs vs. Xilinx Virtex-5 Devices: Architecture and Performance Comparison*.) The capacity of the Stratix III ALM is 1.8 of Virtex-5's 6-LUT/flipflop pair. Table 3 lists other ratios determined.

Table 3. Logic Capacity Ratio Between Altera and Xilinx Families

Comparison	Logic Capacity Ratio
Stratix III FPGA vs. Virtex-5	1 ALM = 1.8 6-LUT/flipflop pairs
Stratix II FPGA vs. Virtex-4	1 ALM = 1.3 Slices (2 4-LUTs, 2 flipflops)
Stratix III/Stratix II FPGAs vs. Stratix/Cyclone FPGA series	1 ALM = 2.5 LEs
Cyclone series vs. Spartan-3	1 LE = 0.5 Slice (2 4-LUTs, 2 flipflops)

As noted previously about the logic-register unit, counts of these units are dependent on how well the registers are packed with LUTs. Maximum packing is ideal for comparable results, and this only happens when the device is full or with options set to achieve minimum area. To produce the ratios presented above, compilation must use the best packing and/or minimum area settings.

There may be room for interpretation in combinational logic, memory block, and DSP/multiplier counts due to synthesis and LUT structures. However, register and I/O pin counts should be similar (within +/- 5 percent). Discrepancies in register and I/O pin counts indicate HDL set up problems such as optimizing away unused logic. In such cases, revisit the "Writing Equivalent HDL for Different FPGA Architectures" section. If logic usage is critical for the design, then read the "Improving Area (Resources)" section. Additional settings for minimizing area are discussed, along with performance and compile time changes to expect when using these settings.

Power Usage Comparisons

When examining power estimation, it is important to appreciate the factors that affect the accuracy of estimation. Because of the nature of characterizing timing delays in the CAD models, timing results can give a black or white answer as to whether timing can be met. Apply special attention to understand what factors affect core static power and what factors affect core dynamic power. Static power is the power consumed independent of design activity. Dynamic power is the additional power consumed due to toggling signals. Power estimation accuracy can depend heavily on the quality of the designer's input, which includes the following:

Static power factors

- *Device information* - family, size of part, device package, temperature grade, and supply voltage
- *Design information* - resource counts and their configurations, placement, routing, assembler bit settings, and operating conditions

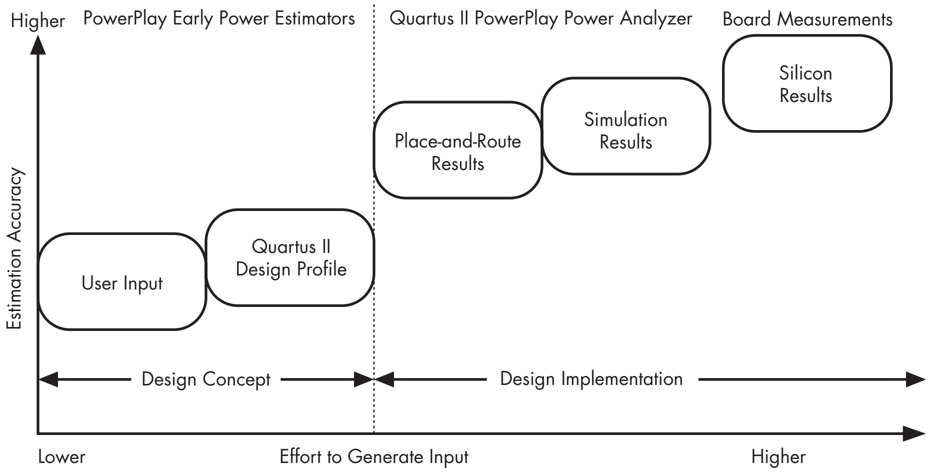
Dynamic power factors

- *Signal activities* - toggle rate and simulation vectors
- *Dynamic power scales linearly with toggle rate*, as capacitive load is charged more frequently for logic and routing.
- *If simulation vectors are provided*, the toggle rate and static probability for each signal is calculated from its waveform.
- *Completeness of timing constraints* - clocks and their relationships, I/O timing, set up, and hold.

☞ Note that Altera's push-button power estimates rely heavily on clock f_{MAX} constraints. If there are no simulation vectors or clock f_{MAX} constraints, the reported dynamic power estimate will be zero in the push-button power flow.

There are two categories of tools used to predict power consumption: power estimators and power analyzers. Power estimators, such as Altera's PowerPlay Early Power Estimator (EPE) and Xilinx's Xilinx Power Estimator, are used early in the design cycle to provide preliminary power estimates. Power analyzers, such as Altera's PowerPlay Power Analyzer (PPPA) and Xilinx's XPower, are used later in the design cycle and provide more accurate estimates. Figure 10 charts the accuracy of Altera's PowerPlay tools with their respective designer's inputs, with the horizontal axis representing increasing effort to generate results for inputting into the tools and the vertical axis representing increasing estimation accuracy.

Figure 10. Accuracy of Power Estimation vs. Effort to Generate Inputs for Analysis



As illustrated in Figure 10, there are several approaches to examining a design's power usage for different scenarios:

Early power estimation (spreadsheet-based power estimators)

- *Appropriate for planning a design concept early in the design stage*. This could be any stage prior to placement and routing. Each PowerPlay EPE spreadsheet for Altera devices (available from the Altera website) is accompanied by a user guide, which outlines the inputs for the spreadsheet. Inputs include device details, resource counts, fanouts, and toggle rates. Example toggle rates are 12.5 percent for logic and I/O, 50 percent for DSP and RAM (toggle and read/write) blocks, and 100 percent for RAM blocks enable rate. As shown in Figure 10, such power estimates may not accurately reflect hardware power requirements, and this is due to two reasons. One reason is that simulation vectors are not provided here and dynamic power is proportional to dynamic behavior specified by the vectors. The second reason is that place-and-route information is missing and, therefore, the design's total capacitive load cannot be determined. The following settings must be made carefully:
  - Temperature of 85°C is important for comparison purposes.
  - The process setting of "typical" is common, and checking worst-case models is recommended for static power.
  - Make sure I/O standards are comparable.
  - When comparing architectures, set the frequency to the slowest frequency achieved by either architecture and use this for all architectures' power analyses. Disregard this if the purpose of the comparison is to maximize performance on each architecture respectively.

  Recognize that comparing dynamic power between vendors can be misleading because vendors make different assumptions on routing. Altera built EPE models to lie in the center of observed distribution (average) as the assumed amount of routing used in a design. Other spreadsheet-based power estimators make far more optimistic assumptions, resulting in underestimated dynamic power.

Integrated power analysis tools (CAD flow-based power estimators)

- *The design has completed the place-and-route step, and has full post-fit netlist (timing) simulation vectors to input into the power analyzer.* This flow provides higher accuracy because all node activities reflect actual design behavior, provided that supplied input vectors are representative of typical design operation. Results are better if the simulation has filtered out glitches. The disadvantage of this method is that simulation times can be long.

  When designing the vectors for power estimation, ensure that they reflect full typical activity of the design and they are not just a string of corner cases or just error tests, because reported dynamic power is proportional to the dynamic behavior of the design. Keep in mind which simulation model is used in the simulator tool, because different models result in different power estimation. Altera recommends using the transport delay model along with using glitch filtering in both the simulator and PowerPlay Power Analyzer.

  Details on simulation models and how to import simulation data are in the *PowerPlay Power Analysis* chapter of the *Quartus II Handbook*.

- *The design has completed the place-and-route step, but post-fit netlist simulation vectors are not available for input into the power analyzer.* Instead, this scenario uses RTL simulation supplemented by vectorless estimation. RTL (functional) simulation provides toggle rates and static probabilities for all pins and registers in the design. Vectorless estimation fills in the values for all the combinational nodes between pins and registers. This method yields good results, since vectorless estimation is reasonably accurate, given that the proper pin and register data is provided. This flow usually provides a simulation time benefit to the user in the third-party RTL simulator.

  Altera's PowerPlay Power Analyzer offers vectorless estimation; Xilinx's XPower 9.2 does not offer estimation for nodes between pins and registers while using RTL simulation. Thus, Xpower is not able to do a full power analysis for this and the following two scenarios.

  Details on how to use PowerPlay Power Analyzer with vectorless estimation are in the *PowerPlay Power Analysis* chapter of the *Quartus II Handbook*.

- *The design has completed the place-and-route step, but no simulation data is available for input to the PowerPlay Power Analyzer.* Instead, this scenario uses signal activities from vectorless estimation and user-supplied input pin activities. This option provides a lower level of accuracy, because vectorless estimation for registers is not entirely accurate.
- *The design has completed the place-and-route step, but no signal activity is provided.* Instead, PowerPlay Power Analyzer considers the user defaults only. This option provides the lowest degree of accuracy, but gives a quick and dirty result.

Board measurements

- *Prototype of the design is on the FPGA.* Taking board-level measurements is the most accurate method of determining the design's power usage. Different measurements must be made for each of the voltage rails (e.g., various V_{CCIO} voltage and I/O banks, and V_{CCINT}.) At this stage of the design process, the power estimation of the earlier scenarios is being verified in silicon.

While looking at Altera's PPPA, notice the quartus_pow --output_epe command option to have PPPA export a file for EPE. This command is currently supported after placement and routing. At this stage of the design, it is more accurate to use PPPA, as illustrated in Figure 10, than to use EPE with the exported file, because PPPA takes into account place-and-route information, whereas EPE has no knowledge of that information. The output_epe option is only provided as a convenience for those who want to quickly get power estimates for different device settings without having to redo placement and routing.

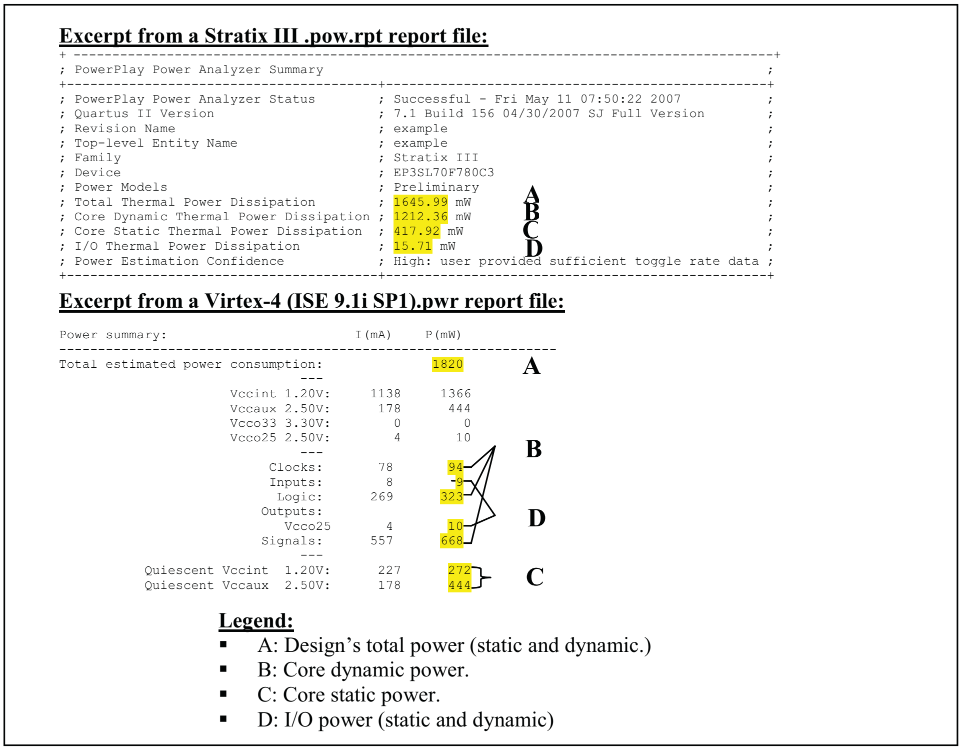
When comparing the power consumption of different architectures, a relative power comparison could be carefully done with undetermined settings consistently set for all compared architectures. The measured power value would be inaccurate, but would serve as a relative, and useful, comparison of how an FPGA architecture's power analysis result compares to another.

For more accurate power comparisons, power estimation can be used to compare architectures for a given design based on resource usage by the targeted device family. Since the estimators are completely reliant on the user's input, it is especially important to verify that the settings made are fair and equivalent between tools. Like timing analysis, beware of differences in assumptions and default setting of the different tools. For competitive comparisons be sure to:

- Verify that the junction temperature is the same across architectures. This can be adjusted by adjusting the thermal information.
- Normalize I/O standards (especially for small designs.)

This section emphasizes that using the power analyzers gives the most accurate estimate of a design's power usage short of measuring it in silicon. Thus, a side-by-side comparison of Altera's and Xilinx's power estimator spreadsheets is not shown here. Figure 11 illustrates the power analyzer report summary comparisons of a Stratix III FPGA and Virtex-4 (at the time of writing, Virtex-5's XPower report was erroneous in ISE 9.2i SP2), with common metrics labeled with the same letter.

Figure 11. Stratix III and Virtex-4 Power Analysis Report Summaries Compared



Excerpt from a Stratix III .pow.rpt report file:

```
+ -----------------------------------------------------------------------------------------+
; PowerPlay Power Analyzer Summary                                                         ;
+----------------------------------------+-------------------------------------------------+
; PowerPlay Power Analyzer Status        ; Successful - Fri May 11 07:50:22 2007           ;
; Quartus II Version                     ; 7.1 Build 156 04/30/2007 SJ Full Version        ;
; Revision Name                          ; example                                         ;
; Top-level Entity Name                  ; example                                         ;
; Family                                 ; Stratix III                                     ;
; Device                                 ; EP3SL70F780C3                                   ;
; Power Models                           ; Preliminary                                     ;
; Total Thermal Power Dissipation        ; 1645.99 mW                                      ;
; Core Dynamic Thermal Power Dissipation ; 1212.36 mW                                      ;
; Core Static Thermal Power Dissipation  ; 417.92 mW                                       ;
; I/O Thermal Power Dissipation          ; 15.71 mW                                        ;
; Power Estimation Confidence            ; High: user provided sufficient toggle rate data ;
+----------------------------------------+-------------------------------------------------+
```

Excerpt from a Virtex-4 (ISE 9.1i SP1).pwr report file:

```
Power summary:                              I(mA)     P(mW)
-----------------------------------------------------------------
Total estimated power consumption:                     1820
                                 ---
                      Vccint 1.20V:          1138      1366
                      Vccaux 2.50V:           178       444
                      Vcco33 3.30V:             0         0
                      Vcco25 2.50V:             4        10
                                 ---
                           Clocks:             78        94
                           Inputs:              8         9
                            Logic:            269       323
                          Outputs:
                             Vcco25             4        10
                          Signals:            557       668
                                 ---
           Quiescent Vccint  1.20V:           227       272
           Quiescent Vccaux  2.50V:           178       444
```

Legend:

- A: Design’s total power (static and dynamic.)
- B: Core dynamic power.
- C: Core static power.
- D: I/O power (static and dynamic)

When examining static power, note that the result is dependent on the size of the device selected. Therefore, the difference in offered sizes of different FPGA architectures can skew static power results. This is especially important to realize for small designs, since only a fraction of a device is being used (the device’s intrinsic static power dominates the total static power). Stratix III FPGAs are less susceptible with their high-speed and low-power logic array blocks (LABs), since static power is lower for LABs not in use. However, a toy design in a large Stratix III FPGA will still have skewed static power results.

Compare core dynamic power, core static power, and I/O power carefully. Each design has a different mix of these results, and it is important to determine which are important to evaluate. Typically, the core dynamic power is the largest player, and it is also the one which can be significantly optimized.

If minimizing power usage is critical for the design, then read the “Improving Power” section. Additional settings to optimize for power are discussed, along with what performance and compile time changes to expect when using such settings.

Improving Results of a Single Metric

FPGA CAD tools are developed to balance trade-offs, such that the initial results are a good start overall, and obtained with relatively reasonable computing resources. The following are the trade-offs:

- Performance versus {area, power, compute time}
- Area versus {performance, power, compute time}
- Power versus {performance, area, compute time}

If better results are desired, user settings are available to guide the CAD tools to optimize one criterion while sacrificing the others. As well, when evaluating whether an architecture offers the best of either performance, area, or power, fair comparisons can only be made by turning on the extra effort features to optimize for that criteria. This means multiple compilations would be attempted when evaluating more than one criterion. For example, studying how well different FPGA architectures do with both performance and power requires separate studies for performance and power, each with separate settings to optimize for that metric.

To determine the best settings, Quartus II software's Design Space Explorer (DSE) automatically produces results for a variety of the settings listed above for the required optimization and picks the best settings. In addition, it also gives the option of seed sweeping. The Quartus II fitter uses a seed to specify the starting value that randomly determines the initial placement for the current design. Changing the starting value may or may not produce better fitting. By sweeping a variety of seeds, it ensures consistency in results, thus building confidence in results.

For details on how to use DSE, refer to the *Design Space Explorer* chapter of the *Quartus II Handbook*. Also refer to the "Meet Your FPGA Design Requirements with Maximum Productivity" net seminar to reduce time to run multiple variations of a design's compilation.

Below are lists of Quartus II settings for each type of optimization. Further details on these settings are in the *Area and Timing Optimization* and the *Power Optimization* chapters of the *Quartus II Handbook*.

Improving Performance

The following are settings recommended to maximize performance and are included in DSE's "Physical Synthesis with Retiming Space:"

- Physical Synthesis for Performance has three options that are useful for maximizing performance. They are "Perform Physical Synthesis for Combinational Logic," "Perform Register Duplication," and "Perform Register Retiming". By default, the options are off, and turning all of them on is useful; leaving the effort level at "Normal" is highly recommended.
- Optimization Technique specifies the overall optimization goal for analysis and synthesis. The default setting is "Balanced," and when set to "Speed," analysis and synthesis maximizes performance. If the design has third-party netlists, also turn on WYSIWYG primitive resynthesis such that the netlisted parts of the design can take advantage of the setting specified in the Optimization Technique logic option.
- Logic Cell Insertion - Logic Duplication allows the fitter to insert buffer logic cells between two nodes and duplicate a logic cell within a LAB when there are unused logic cells available in a LAB. The default is "Auto," and when set to "On," the performance of the design can be improved by this optimization with a compilation time increase.
- Router Timing Optimization Level controls the effort level of the algorithms used by the router to improve circuit timing (reduce delay). The default level is "Normal;" when set to "Maximum," it achieves the highest possible performance, but may increase the compilation time.

More extensive ways to maximize performance are available in Tools > Advisors > Timing Optimization Advisor.

Improving Area (Resources)

The following are settings recommended to minimize area and are included in DSE's "Area Optimization Space":

- *Optimization Technique* specifies the overall optimization goal for analysis and synthesis. The default setting is "Balanced," and when set to "Area," analysis and synthesis minimizes logic usage. If the design has third-party netlists, also turn on WYSIWYG primitive resynthesis such that the netlisted parts of the design can take advantage of the setting specified in the Optimization Technique logic option.
- *Restructure Multiplexer* repacks multiplexers more efficiently for area, allowing the design to implement multiplexers with a reduced number of LEs. The default setting is "Auto"; when set to "On", it decreases LE usage but may negatively affect design performance.
- *Auto-Packed Registers* setting controls how aggressively the fitter combines registers with other function blocks in order to reduce the area of the design. The default setting is "Auto," and when set to "Minimize Area," the fitter aggressively combines unrelated functions in order to reduce the area required for placing the design, at the

expense of design performance. When this option is set to “Minimize Area with Chains,” the fitter even more aggressively combines logic functions that are part of arithmetic or register cascade chains or that can be converted to register cascade chains.

More extensive ways to minimize area are available in Tools > Advisors > Resource Optimization Advisor, which includes two “Physical Synthesis for Fitting” options that work to reduce area only if the design fails to fit. “Perform physical synthesis for combinational logic” removes duplicate logic and “Perform logic to memory mapping” allows mapping of logic and registers into unused memory blocks during fitting to achieve a fit. By default the options are off, and turning both of them on is useful if the design fills the device.

Improving Power

The following are settings recommended to minimize area and are included in DSE’s “Power Optimization Space”:

- *Area-Driven Synthesis* settings minimize area, thus minimizing power usage. From the settings listed in the “Improving Area (Resources)” section, review the ones that are made in analysis and synthesis.
- *Analysis and Synthesis PowerPlay Power Optimization* setting determines how aggressively analysis and synthesis optimizes the design for power. The default setting is “Normal compilation;” when set to “Extra effort,” analysis and synthesis perform additional memory block power optimizations that may reduce design performance.
- *Fitter PowerPlay Power Optimization* setting determines how aggressively the fitter optimizes the design for power. The default setting is “Normal compilation;” when this option is set to “Extra effort,” the fitter performs additional power optimizations by effectively moving the logic closer during placement to localize high-toggling nets, and using routes with low capacitance. “Extra effort” may affect design performance and/or increase compile time. For the best results with Extra Effort power optimization during fitting, specify a Signal Activity File (SAF file) that lists the toggle rate of each signal in the design. Without a Signal Activity File (from simulation or other source), the Quartus II software uses assignments, clock assignments, and vectorless estimation values (PowerPlay Power Analyzer Tool settings) to estimate the signal activities. This information is used to optimize the design for power during fitting.
- *Programmable Power Technology Option* (in Stratix III FPGAs only) controls how the fitter configures tiles to operate in high-speed mode or low-power mode. The default setting is “Automatic,” and setting “Minimize Power Only” specifies that the fitter should set the maximum number of tiles to operate in low-power mode.

More extensive ways to optimize for power are available in Tools > Advisors > Power Optimization Advisor, which includes a recommendation to minimize memory block power by making simple RTL changes. Taking advantage of having the clock enable signal off in conditions when the memory is not being accessed reduces the memory block’s switching activity, thus reducing power usage.

Conclusions

Benchmarking methodology can greatly influence results in an FPGA comparison. This paper provides guidance on a methodology to accurately compare a design implemented in different FPGA architectures. To make a valid comparison, the design being evaluated must have its HDL written equivalently for the different FPGA architectures, and software constraints and settings must be set carefully such that metrics that depend on those settings are fairly evaluated.

Further Information

- Contact an Altera FAE:
www.altera.com/corporate/contact/con-index.html
- PowerPlay Early Power Estimator (EPE) spreadsheet:
www.altera.com/support/devices/estimator/pow-powerplay.jsp

White Papers

- *FPGA Performance Benchmarking Methodology*:
www.altera.com/literature/wp/wpfpgapbm.pdf
- *Stratix III FPGAs vs. Xilinx Virtex-5 Devices: Architecture and Performance Comparison*:
www.altera.com/literature/wp/wp-01007.pdf
- *Performing Equivalent Timing Analysis Between Altera TimeQuest and Xilinx Trace*:
www.altera.com/literature/wp/wp-01047-performing-equivalent-timing-analysis-between-timequest-and-trace.pdf
- *Power Optimization in Stratix III FPGAs*:
www.altera.com/literature/an/an437.pdf

Quartus II Handbook Chapters

- *PowerPlay Power Analysis*:
www.altera.com/literature/hb/qts/qts_qii53013.pdf
- *Design Space Explorer*:
www.altera.com/literature/hb/qts/qts_qii52008.pdf
- *Area and Timing Optimization*:
www.altera.com/literature/hb/qts/qts_qii52005.pdf
- *Power Optimization*:
www.altera.com/literature/hb/qts/qts_qii52016.pdf

Net Seminar

- “Meet Your FPGA Design Requirements With Maximum Productivity”:
www.altera.com/education/webcasts/all/wc-2007-stratix3-quartus2.html

Acknowledgements

- Carolyn Lam-Leventis, Sr. Software Engineer, Software and System Engineering, Altera Corporation

ALTERA®

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