WP229 (v1.0) July 6, 2005

# Synthesis and Implementation Strategies to Accelerate Design Performance

By: Hitesh Patel

The 2005 EDA Branding Study shows that 71% of FPGA projects have issues meeting their timing budgets. Designers can employ several strategies to meet their timing goals, such as: HDL code changes, and synthesis and implementation tools settings.

This paper describes the synthesis and implementation tools strategies, such as Xplorer™, that can be employed to maximize design performance in actual designs with a detailed user constraints file (UCF) or benchmark designs where the user is evaluating the best achievable performance for a specified clock domain.

To highlight the impact of these strategies, the paper shows performance improvements achieved on eight OpenCores designs, comparing the Virtex™-4 and Spartan™-3 FPGAs performance with competitive offerings.

## Synthesis Tools Strategies

Synthesis optimizations can have a dramatic impact on design performance because synthesis tools are operating at higher levels of abstraction. After parsing through the HDL behavioral source code and extracting known functions (e.g., arithmetic functions, multiplexers, memories, and so on), synthesis tools then map these functions on the target architecture features. The tools trade area and performance based on design constraints and tools settings, and these influence the usage of optimizations such as replication, merging, retiming, and pipelining. As a result, the right tools settings in synthesis can have a positive impact on design performance. Here are the recommended synthesis settings to maximize design performance:

- Timing Constraints: Apply timing constraints or use automatic constraining if available
- Black Boxes: Read in EDIF cores from CORE Generator™ (FIFOs, memories, etc.)
- Finite State Machines (FSM): Turn on FSM optimization
- Target Device: Specify the target device so synthesis can allocate resources based on availability
- Pack IO Registers into IOBs: Disable to push fabric performance.
- Resource Sharing: At times, turning resource sharing off can create a faster circuit
- Retiming: Redistributing logic between registers to balance delays while preserving latency
- Other: Pipelining, Register Duplication and Fanout settings

After synthesis, the next stage in the design process is implementation.

## Implementation with Xplorer

Xplorer is a perl script that seeks the best design performance using ISE. After synthesis generates an EDIF (*.edf file), the design is ready for implementation. During this phase, the user has the option to use the Graphical User Interface (GUI) Project Navigator in ISE to apply design constraints and explore different tools settings for best performance. An alternative approach may be to use Xplorer. Xplorer has been designed to help achieve optimal results for designs by employing smart constraining techniques and a variety of physical optimization strategies. Because no unique set of ISE options or timing constraints work best on all designs, Xplorer works to explore the right set of tools options to either meet design constraints or to find the best performance for the design. Hence, Xplorer has two modes of operation: Performance Evaluation Mode and Timing Closure Mode.

### Performance Evaluation Mode

In this mode of operation, Xplorer optimizes the design performance for the user-specified clock domain. The performance evaluation mode is also especially useful for designers who want to easily evaluate the performance of a reference design. The user specifies the design name, a single clock to optimize, and a starting frequency. Beginning with an initial frequency, Xplorer runs multiple Place and Route (PAR) iterations with different physical architecture specific optimization strategies to seek the best performance. If a starting frequency is not specified, Xplorer estimates the starting frequency based on pre-PAR timing estimates.

Starting with an initial frequency goal, Xplorer implements the design using timing-driven PAR and tightens or relaxes the timing constraints depending on whether the frequency goal is achieved, as shown in Figure 1. Adjusting timing constraints such that PAR is neither under nor over-constrained enables Xplorer to deliver the optimal design performance. In addition to timing constraints, Xplorer also utilizes physical optimization strategies such as Global Optimization and

timing-driven Packing and Placement. Global Optimization performs pre-placement netlist optimizations on the critical region and timing-driven Packing and Placement provides a closed-loop packing and placement such that the placer recommends logic packing techniques that deliver optimal placement.

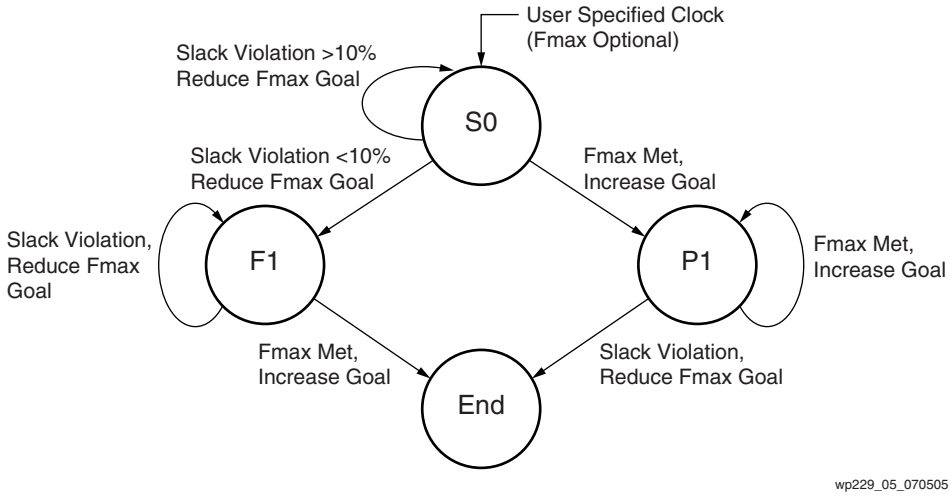


*Figure 1:* **Xplorer Performance Evaluation Mode Flow**

If the design has a User Constraint File (UCF), Xplorer optimizes for the user constraints in addition to the specified clock domain. Because Xplorer runs multiple PAR iterations, specifying a frequency goal in Xplorer minimizes the time to achieve the best possible performance.

## Timing Closure Mode

The Timing Closure Mode has been designed for users who have a design with timing constraints in their UCF, and their intent is for the tools to meet the constraints specified in the UCF. In this case, the user should not specify a clock using the -clk <clock name> switch. Xplorer looks for the UCF to examine the timing constraints goals. Using these constraints together with optimization strategies, such as Global Optimization, Timing Driven Packing and Placement, Register Duplication, and Cost Tables, Xplorer implements the design in multiple ways to deliver optimal design performance.

## Architecture Support

All Xilinx FPGA architectures are supported by Xplorer, and optimizations are performed based on architecture features.

## Usage

Xplorer is run from the command prompt by typing:

```
xplorer <design name> [-clk <clkname>] [-p <partname>]
```

`<design name>:`

Name of the top level edif/ngc file.

`-clk <clkname>:`

Name of the clock to be optimized. If the -clk option is omitted, the script uses the timespecs defined in the UCF file.

`-p <partname>:`

The device name (e.g. XC4VLX100-11FF1152). The default value is the part specified in the input design.

`-freq <value in MHz>:`

The initial frequency goal (in MHz). Without this option the script automatically calculates a reasonable starting value.

`-uc <ucf file>:`

The UCF file name. The default value is <design name>.ucf.

Example command for Performance Evaluation Mode:

```
xplorer cordic -clk clk -freq 400 -p XC4VLX15-12FF668
```

Example command for Timing Closure Mode:

```
xplorer <design name> -uc <ucf file> -p <partname>
```

Results, with the tools settings used, are summarized in xplorer.rpt. The best run is identified at the end of this report file. To view advanced options and to download Xplorer, go to: http://www.xilinx.com/xplorer.

## Performance Improvement Results

To highlight the performance impact of these optimization strategies, this section shows performance improvements for the Xilinx customer design suite. In addition, performance improvements for eight OpenCores designs for Xilinx FPGAs and competitive FPGAs are also demonstrated. The OpenCores designs are written in synthesizable RTL and synthesized to the target technology without any code modifications. These designs can be downloaded from: http://www.opencores.org.

### Benchmark Methodology

The Benchmarking Methodology can have a significant impact on design performance, and a poor benchmarking methodology can provide inaccurate data. Xilinx has worked with FPGA tool industry leaders to develop a sound methodology for benchmarking, providing an accurate way to measure FPGA performance. During the synthesis phase, timing-driven synthesis and the synthesis strategies described in the Synthesis Tools Strategies section have been deployed. During the implementation phase, Xplorer is used in the Performance Evaluation mode. The tools used are Synplify Pro 8.1 and ISE 7.1service pack 2. For additional details on the benchmark methodology, refer to the Xilinx white paper: *Achieving Breakthrough Performance in Virtex-4 FPGAs, WP218*.

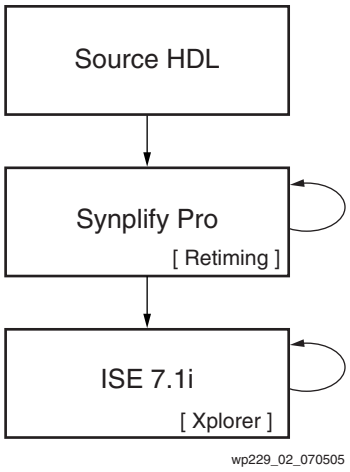


*Figure 2:* **Benchmarking Design Flow**

The Stratix II and Cyclone II data is captured from Altera's *FPGAs for High Performance DSP Applications* white paper.

## Xilinx Xplorer Advantage

Xilinx has a high-density, high-performance customer design suite of over 50 designs that compile and implement successfully for both Xilinx and competitive FPGA offerings. These designs range in density from LX15 to LX200, covering but not limited to market segments such as: Consumer, Video, Storage, Telecom/Datacom, DSP, and Glue Logic. For this extensive design suite, Xplorer - RTL Retiming provides an average performance improvement of 26% for Virtex-4, shown in Figure 3.

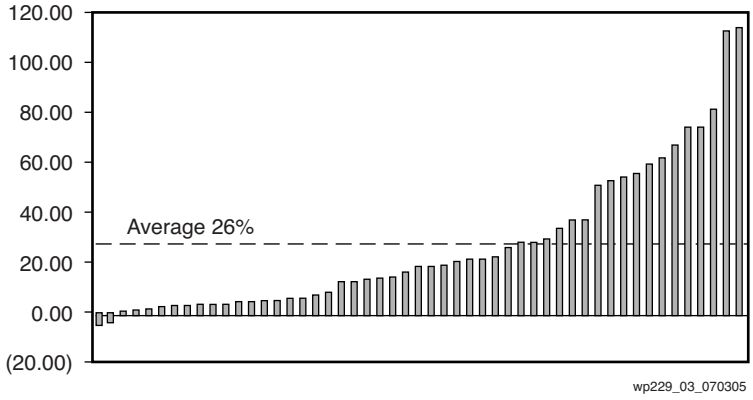


*Figure 3:* **Percentage Performance Improvement with the Virtex-4 FPGA for a Customer Design Suite Using Xplorer - RTL Retiming**

For the eight OpenCores designs, Figure 4 shows the relative performance improvement ratio for Virtex-4 and Spartan™-3 FPGAs using Xplorer-RTL Retiming. This graph shows that the Xplorer-RTL retiming strategy improves design performance for all eight OpenCores designs, with the Simple FM Receiver (FM) design seeing an 83% performance improvement for Virtex-4 devices and 64% for Spartan-3 devices.

*Table 1:* **Xplorer-RTL Retiming Performance Advantage for 8 OpenCores Designs**

| Comparison Category | Xplorer-RTL Retiming Performance Advantage |
|---|---|
| Virtex-4 FPGA | Up to 83%<br>(on average, 19% higher performance for all eight designs) |
| Spartan-3 FPGA | Up to 64%<br>(on average, 23% higher performance for all eight designs) |

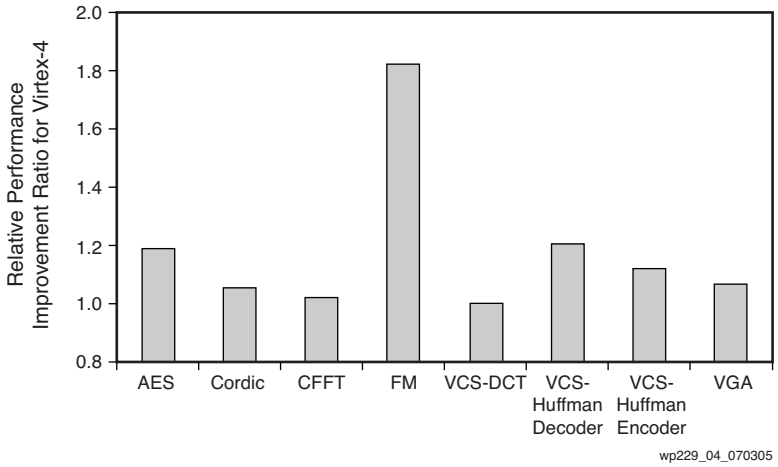


*Figure 4:* **Relative Performance Improvement Ratio with the Virtex-4 FPGA Using Xplorer-RTL Retiming**

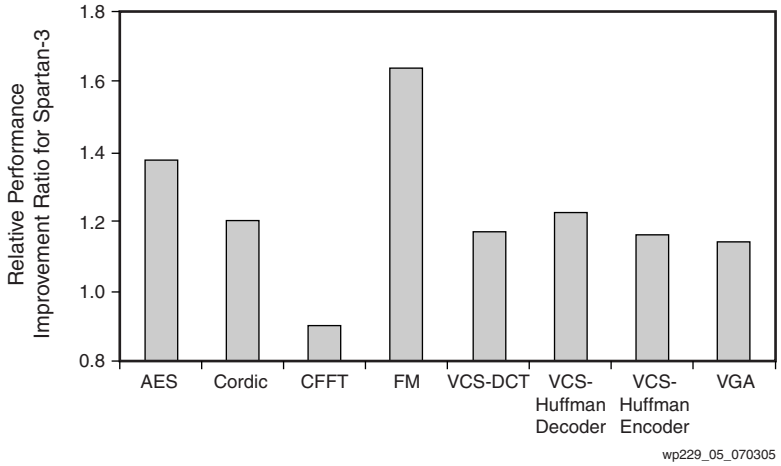


*Figure 5:* **Relative Performance Improvement Ratio with the Spartan-3 FPGA Using Xplorer-RTL Retiming**

## Xplorer Competitive Advantage

For the eight OpenCores designs, the high performance FPGA comparison shows that the Virtex-4 device is on average 7% faster than Stratix II. For the extensive high-

performance, high-density customer design suite, the Virtex-4 device is on average 14% faster than Stratix II.

*Table 2:* **Competing FPGAs Comparison for Eight OpenCores Designs**

| Comparison Category | Devices | Xilinx Performance Advantage |
|---|---|---|
| High Performance FPGA Comparison | Xilinx Virtex-4 FPGA vs. Altera Stratix II, fastest speedgrade | Up to 21%<br>(on average, 7% higher performance for all eight designs) |
| Low Cost FPGA Comparison | Xilinx Spartan-3 FPGA vs. Altera Cyclone II, cheapest speedgrade | On average, equivalent |

Table 3 and Table 4 show the performance benchmark data for:

- High performance, high density FPGAs comparing the Virtex-4 device and the Stratix II
- Low cost FPGAs comparing Spartan-3 and Cyclone II

The Virtex-4 family operates at over 400 MHz for Cordic and VCS-Huffman Encoder and has faster performance than Stratix II for seven of the eight designs. The low cost Spartan-3 family has equivalent performance to Cyclone II and is up to 10% faster than Cyclone II for the VCS-DCT design.

*Table 3:* **Virtex-4 FPGA vs. Stratix II OpenCores Benchmark Data**

| Design Name | Virtex-4 Device (MHz) | Stratix II (MHz) | Virtex-4 Device over Stratix II |
|---|---|---|---|
| AES | 234 | 231 | **1.01** |
| Cordic | **406** | 374 | **1.09** |
| CFFT | 368 | 340 | **1.08** |
| FM | 214 | 177 | **1.21** |
| VCS-DCT | 258 | 231 | **1.12** |
| VCS-Huffman Decoder | 270 | 276 | 0.98 |
| VCS-Huffman Encoder | **404** | 392 | **1.03** |
| VGA | 271 | 269 | **1.01** |
| **Xilinx Advantage** | | | **1.07** |

*Table 4:* **Spartan-3 FPGA vs. Cyclone II OpenCores Benchmark Data**

| Design Name | Spartan-3 Device (MHz) | Cyclone II (MHz) | Spartan-3 Device over Cyclone II |
|---|---|---|---|
| AES | 102 | 105 | 0.97 |
| Cordic | 178 | 176 | 1.01 |

*Table 4:* **Spartan-3 FPGA vs. Cyclone II OpenCores Benchmark Data *(Continued)***

| Design Name | Spartan-3 Device (MHz) | Cyclone II (MHz) | Spartan-3 Device over Cyclone II |
|---|---|---|---|
| CFFT | 144 | 147 | 0.98 |
| FM | 77 | 77 | 1.00 |
| VCS-DCT | 130 | 119 | 1.10 |
| VCS-Huffman Decoder | 130 | 131 | 0.99 |
| VCS-Huffman Encoder | 178 | 190 | 0.94 |
| VGA | 120 | 124 | 0.97 |
| Average | | | **1.00** |

### How to Realize Additional Performance Gains

At times, the above mentioned synthesis and implementation strategies might still not be adequate to meet the target timing goals. In these cases, the RTL source code should be examined and a coding strategy employed that drives the synthesis tools to infer architecture primitives, such as the DSP48, that are geared to operate at up to 500 MHz. To learn about HDL coding guidelines that yield the best implementation, refer to the Xilinx white paper: *Achieving Breakthrough Performance in Virtex-4 FPGAs, WP218.*

## Summary

The Xplorer - RTL Retiming strategy helps users to optimize logic performance for a specific clock domain and meet their existing timing constraint design goals by employing Xplorer to identify the right set of tools options for optimal design performance.

For an extensive design suite of over 50 customer designs, Xplorer - RTL Retiming provides an average performance improvement of 26% for Virtex-4 FPGAs compared to competing devices.

Based on the benchmarking results from eight OpenCores RTL designs:

- Xplorer - Retiming provide an average 19% performance improvement for Virtex-4 and 23% for Spartan-3.
- Virtex-4 high density and high performance FPGAs offer up to 21% and an average 7% better performance than Stratix II. Virtex-4 performance can far exceed any other FPGAs in the market.
- Spartan-3 provides the industry's lowest cost per I/O FPGA while maintaining comparable performance to Cyclone II. In addition, with extended features such as LUT  RAM and SRL16, Spartan-3 offers better device utilization than competing FPGAs.

## Revision History

The following table shows the revision history for this document.

| Date | Version | Revision |
|---|---|---|
| 07/06/05 | 1.0 | Initial Xilinx release. |