



# Maximizing Design Performance for Virtex-5 FPGAs

ISE software gives you the tools to achieve the timing goals of a Virtex-5 design.

by Michelle Fernandez  
Software Technical Marketing Engineer  
Xilinx, Inc.  
[michelle.fernandez@xilinx.com](mailto:michelle.fernandez@xilinx.com)

As FPGAs push the performance envelope, maximizing design performance requires knowledge of the device architecture and design software. The 65-nm Xilinx® Virtex™-5 FPGA family delivers the industry's highest performance, with new ExpressFabric™ technology, diagonally symmetric routing, enhanced on-chip memory, DSP slices, and high-speed I/O. To maximize system performance, you should use proper design techniques such as defining timing constraints and selecting options in synthesis and implementation that work best for your design. In this article, I'll describe how to achieve faster timing in the fewest design iterations.

## Understanding the Architecture

When evaluating a new FPGA architecture like the Virtex-5 family, it is important to study the user guide and data sheet to understand the hardware features.

The Virtex-5 FPGA family is based on a new ExpressFabric architecture that delivers higher speeds, a new 6-input LUT structure that reduces logic levels, and diagonally symmetric routing that minimizes delays. Each CLB contains two slices that have four 6-input LUTs and four registers configurable in many ways. For maximum slice packing, it is imperative that you understand the slice interconnectivity and any shared resources.

Virtex-5 FPGAs contain hard IP such as embedded memory (block RAM) and math functions (DSP48E slices) tuned to 550 MHz. Here are some design consid-

erations if any of these hard-IP blocks show up as part of your critical paths:

- Check to see if your design is making the most of the block's features and that the synthesis tool is inferring the features as expected from your RTL.
- When using the embedded block RAM memory or the DSP48E slices, it is important to use their dedicated pipeline registers when possible to reduce setup and clock-to-out timing.
- Another consideration is the mix of block RAMs or DSP48E slices in the design, and the trade-off between using dedicated blocks or implementing the same function in slices to allow for placement flexibility.

The choice of clocking resources can also affect a design's performance. Virtex-5

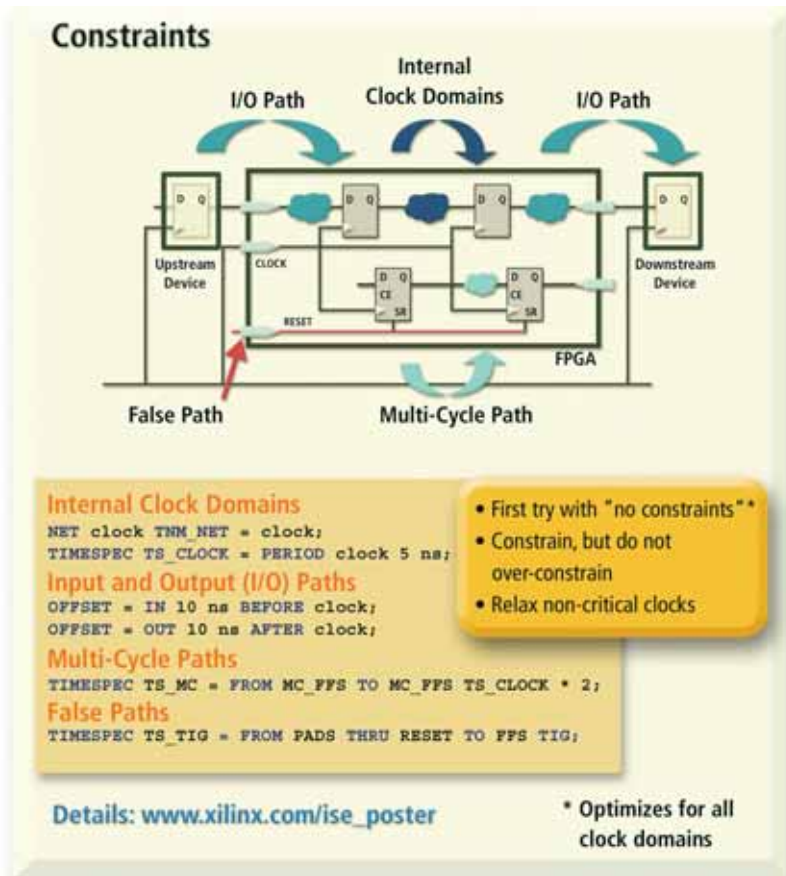


Figure 1 – Proper timing constraints

FPGAs have I/O, regional, and global clocking resources. These devices are divided into multiple clock regions, which at most can contain 4 regional clocks and 10 global clocks. During design planning, it is important to analyze how many clock regions you plan to use as well as specific clocks within a clock region. Placing your I/Os so that their interface logic does not require all of the clock resources in a clock region gives ISE™ software greater placement flexibility.

### Define Timing Requirements

Synthesis and ISE implementation tools are driven by the performance goals that you specify with timing constraints for internal clock domains, I/O paths, multi-cycle paths, and false paths (see Figure 1). Defining realistic timing constraints will prevent excessive replication and longer run times.

In your synthesis report, check for any replicated registers and confirm that the timing constraints that apply to the orig-

inal register also cover the replicated registers for implementation. When writing timing constraints, group the maximum number of paths with the same timing requirement before generating a specific constraint to minimize implementation run times and memory usage.

### Driving Synthesis

Here are some design considerations for getting optimal results from synthesis tools:

- Use proper coding techniques to ensure that the inference of your RTL by synthesis takes advantage of the architectural features.
- Add any lower level netlists to your synthesis project to better optimize HDL that interfaces to those netlists.
- If critical paths in your implementation are not seen as critical in synthesis, try Synplify Pro's "-route" constraint to force synthesis to focus on that path.

### Synplify Pro® Settings

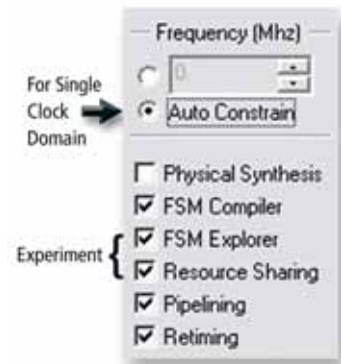


Figure 2 – Recommended Synplify Pro settings

### ISE Synthesis "Process Properties" Dialog

	Property Name	Value
Synthesis Options	Optimization Effort	High
	Synthesis Constraint File Create a XCF File (UCF Syntax) NET clock PERIOD = 5 ns	<input checked="" type="checkbox"/>
	Read Cores	<input checked="" type="checkbox"/>
HDL Options	Resource Sharing	Experiment
Xilinx-Specific Options	Register Balancing	YES
	Pack I/O Registers into IOBs	Experiment

Figure 3 – Recommended ISE synthesis (XST) settings

- Explore the synthesis tool settings. (See Figure 2 for Synplify Pro and Figure 3 for Xilinx Synthesis Technology [XST] suggested tool settings.) There are also a variety of attributes that can affect synthesis optimizations. These attributes are an easy way to affect synthesis without having to re-code (see Table 1).

Certain tool settings, such as retiming in Synplify Pro and register balancing in XST, can impact area. If your design is affected by high fan-out nets and you want the synthesis tool to reduce that fanout, use fan-out attributes specifically on that net versus globally reducing the fan-out limit. Avoid maintaining hierarchy if critical paths cross over the hierarchical boundaries. Before implementation, review the warnings in your synthesis report.



## Choosing Implementation Options

Having obtained an acceptable timing estimate from the synthesis tool, you can use the implementation tools to determine the true performance of the design. The ISE default mode is the performance evaluation mode, which enables you to get high-performance results out of your implementation tools without having to specify timing goals.

The next step is to run timing-driven mapping (MAP) and place and route (PAR). Timing-driven MAP performs closed-loop packing and timing-driven placement, while PAR performs the routing of the design. Both MAP and PAR should run with their effort levels set to high to achieve optimal results.

Physical synthesis options in implementation can re-optimize and pack logic based on knowledge of the critical paths of a design, leading to better placement and routing. The physical synthesis options are implemented during the MAP process and include global netlist optimization, localized logic optimization, retiming, register duplication, and equivalent register removal. Details on each of these options can be found in the Xilinx White Paper, "Physical Synthesis and Optimization with ISE 8.1i," available at [www.xilinx.com/bvdocs/whitepapers/wp230.pdf](http://www.xilinx.com/bvdocs/whitepapers/wp230.pdf).

## Xplorer Utility

Xplorer is a tool that helps to determine the set of implementation options that result in the best performance for a design. Xplorer has two modes: timing closure and best performance. The timing closure mode evaluates your timing constraints and tries different sets of implementation options to achieve those goals. In best performance mode, you can give the tool a clock domain to focus on; the tool will try to achieve the best frequency for the clock. This is helpful when benchmarking a design's maximum performance.

## Evaluating Your Critical Paths

By understanding the characteristics of your critical path, you can make better decisions about what to do for your next design itera-

tion. A datapath comprises both logic and interconnect delay. Individual component delays that make up logic delay are fixed. You can reduce logic delay by reducing the number of logic levels or by redefining the structure of the logic.

In comparison, interconnect delay is much more variable and is dependent on the placement of the logic. Before running your design through PAR, a timing analysis after MAP is recommended. Although this timing report will only have estimates for your routing delays, it can give you an idea of the critical paths the implementation tools are working on. If the critical paths have a high number of logic levels, you may want to work on improving the logic levels versus running it through PAR.

If your design has an excessive amount of logic levels:


1. Try the physical synthesis options in MAP.
2. Go back to synthesis and verify that critical paths reported in implementation match what is reported in synthesis.
3. Review the synthesis inference of your HDL code.

If there are few logic levels but certain datapaths are not meeting timing:

1. Evaluate fan-out on routes with long delay.

2. If the critical path contains hard-IP blocks such as block RAMs or DSP48E slices, verify that the design takes full advantage of the embedded registers. Also understand when to make the trade-off between using these hard blocks or using slice logic.
3. Analyze clock skew.
4. If the logic appears to be placed far apart, floorplanning of critical blocks may be required. Only floorplan where necessary.
5. If area groups were created for a design with a previous version of software or before many design changes, consider removing those area groups.
6. Consider placing hard-IP blocks such as block RAMs for DSP48E slices.

## Conclusion

Virtex-5 FPGAs are optimized for high-performance designs, while ISE software has the capabilities you need to quickly achieve design closure, improve productivity, and efficiently verify your designs. Xilinx provides a comprehensive suite of software tools (powered by ISE Fmax technology) that improves design performance. However, the more that you can do up-front with good coding styles, defining timing constraints, and resource planning, the easier it will be for downstream tools to achieve your timing requirements. 

	XST	Synplify Pro
Fan-out Control	max_fanout	syn_maxfan
Directs Inference of RAMs to Block RAMs or SelectRAM	ram_style	syn_ramstyle
Directs Usage of DSP48 Slice	use_dsp48	syn_multstyle/syn_dspstyle
Directs Usage of SRL16	shreg_extract	syn_srlstyle
Controls % of Block RAMs Utilized	n/a	syn_allowed_resources
Preservation of Register Instances During Optimizations	Keep	syn_preserve
Preservation of Wires	Keep	syn_keep
Preservation of Black Boxes with Unused Outputs	Keep	syn_noprune

\* You can find XST documentation at <http://toolbox.xilinx.com/docsant/xilinx82/books/docs/xst/sst.pdf>. Synplify Pro documentation is located in the tool help documentation.

Table 1 – Helpful synthesis attributes\*