Keysight Takes a New Approach to FPGA I/O Optimization to Significantly Reduce PCB Design Time

Introduction

New problems require new solutions. This has never been more true in the realm of FPGA design than it is today. Higher and higher pincount devices have placed a significant strain on classic, over-the-wall methodologies. These techniques, which worked for many years, typically addressed FPGA and PCB design efforts as two distinct disciplines, where the FPGA designer defined the I/O assignments of the FPGA and passed those assignments to the PCB designer.

Ignoring, for the moment, the ramifications on the PCB, this process generally worked pretty well, especially with smaller devices and if the overall system did not have demanding performance requirements.

But as FPGAs have become larger and larger, nearing (and soon, exceeding) 2000 pins, these design philosophies simply fall apart. These big devices demand a fundamentally different approach to FPGA I/O design — an approach that considers, concurrently, the effects that the I/O assignments have on the FPGA and the PCB. This success story chronicles a successful deployment of this new process, based on a realworld design done by a world leader in telecommunications, Keysight Technologies using Xpedition FPGA I/O optimization tools.

Figure 1: Keysight’s 11-FPGA PCB.

Design Overview

Before discussing some of the reasons for Keysight’s success, it’s helpful to understand the characteristics of the design they were dealing with:

- Eleven 1148-pin Virtex 4 LX series FPGAs
- Large DRAM busses on several devices (up to 250 pins in some cases)
- Multiple high-speed (10 GBPS) inter-FPGA busses approximately 48 pins wide
- Wide physical interfaces to PHY devices and backplane
- Very few spare pins
- Complex FPGA restrictions
- 14 differently-sized banks in Virtex 4
- Clocks associated with I/O must be on specific signals
- I/O restrictions, e.g. clockcapable inputs do not support LVDS
- Mixed voltage environment (1.8V/2.5V/3V/3.3V) for different banks
- 26 layer PCB
- 8 inner tracking (signal) layers
- 9800 components
- 9500 nets

Figure 1 is a screenshot of the top of the final board. The large devices are the eleven FPGAs. By any measure, this was a very complex project.
Keysight’s previous experiences suggested that roughly four-to-eight weeks, per FPGA, would be needed for the I/O pin assignment process alone. In addition, I/O restrictions hinted that problems would be likely and indeed, subsequent V4-series designs confirmed that. Keysight also knew that it was difficult to maintain links between the FPGA and board designs as the pinouts changed. Given these challenges, Keysight felt compelled to consider new design strategies.

A View From the Top

Keysight’s goal was to bring FPGA and PCB designers together in a common environment so that each team member can see the ramifications of the FPGA pin assignments on the entire system. The mechanism that I/O optimization uses to accomplish this is conceptually very simple: a view of the PCB with a dynamically-assignable FPGA library element instantiated in place of the FPGA’s typical PCB footprint. The FPGA component, having come from a library of ‘intelligent’ FPGA devices, assists the user in making proper pin assignments. Since changes to those assignments are shown in real-time, the potential effects on the rest of the board can be seen immediately.

Figure 2 is a screenshot of Keysight’s board in Xpedition. The substantial rats nest (some of which are shown in green and orange), needs to be sorted out.

The same Keysight design is shown inside the I/O optimizer (Figure 3, next page). The area in the bottom right is the “multicomponent” window, which is a view of the PCB, and the FPGA currently undergoing active assignment. Other windows convey signal, pin, symbol (not shown) and console (transcribing) information. By displaying the inter-component connections from a board-level perspective, I/O optimization technology allowed Keysight to optimize the pin assignments for each FPGA.

Connecting to the FPGA Tools

Unstated in all of this is how Xpedition links to the FPGA tools. The I/O optimizer is not an FPGA design tool: it can’t route and it can’t synthesize. As such, it must read and write the files necessary to close the loop with the FPGA designer (making pin assignment changes in the schematic or PCB tools, or within I/O optimization, is of very little use if those changes can’t be communicated to the FPGA tools). This is critical: moving a signal from one pin to

Figure 2: ‘Rats nests’ in a section of Keysight’s PCB.
another, even an adjacent pin, may appear harmless, but can actually cause the FPGA to fail timing. Towards this end, I/O optimization can generate or update the associated P&R and synthesis constraint files via a simple pull-down menu. Figure 4 illustrates how I/O optimization fits into the flow, bridging the FPGA and schematic/PCB design process.

**Evaluate the Tools — And Don’t Fear Change**

The need to carefully evaluate new EDA tools before deploying them on a project cannot be overstated and I/O optimization is no exception.

Keysight spent several months assessing I/O optimization — its functionality, performance, stability, compatibility with existing tools and processes, quality and reliability. While this effort uncovered some issues, none were considered significant enough to detract from the overall viability of the technology.

One of the primary findings was that it worked best when used in conjunction with hierarchy.

“We were aware of the emergence of I/O optimization,” said Ross MacIsaac a senior design engineer at Keysight’s South Queensferry site in Scotland. “We opted to introduce it into our process at a point in the design cycle that was later than we were comfortable with as there is a strong preference to use proven tools.”

“However,” continued MacIsaac, “the size of the task merited taking that risk. The support issues were offset by the knowledge that we had very strong support from Mentor.

We were using the tool while it was still frequently being revised but we had good high level and local support from Mentor. We also had faith that Xilinx and Mentor would exchange information efficiently; this was more efficient and lower risk than Keysight also being involved in the component library process.

The effort involved in other schemes was just so much that it was expected to exceed any tool teething problems. Our management were prepared to take the risk of introducing this new technology. This board was the right problem at the right time for the tool and the FPGAs. Both
“We realized early on that hierarchy would enable us to more fully exploit the features in Xpedition’s I/O optimizer,” explained MacIsaac. “As such, although hierarchy was already used extensively within Keysight, we were careful to architect this system to ensure that I/O optimization, as well as our established EDA tools, would deliver us the most productivity. There is no doubt that the use of hierarchy reduced our design cycle time.”

Figure 5 illustrates a core element — a feature which, when used in tight combination with a hierarchical design process, allows it to provide significant productivity improvements.

The top-level schematic contains two functional blocks (virtual representations of the FPGA) and some connecting nets and ports. Below the functional blocks reside the actual symbols or ‘fractures’ that represent the FPGA. In this example, the functional blocks, as well as the FPGA fractures and the schematics on which those fractures reside, have all been created by the toolset.

As the design progresses, it can automatically update the functional blocks, fractures, and lower level schematics. The burden of keeping the schematic view of the FPGA synchronized with the rest of the database, while eliminating manual, tedious, error-prone tasks, allows I/O optimization to deliver radical productivity increases.

Caveats and Lessons Learned

Much of what Keysight undertook with this effort had never been attempted before, at least on this scale. Along the way, they learned a few lessons, as did Mentor:
Keep the tool chain simple and use proven methods. Don't inject too many variables at one time into a new process.

Use as much of the tool's automatic symbol and schematic creation and maintenance features as the process will allow.

Finely partition the FPGA. Use an individual symbol for each signal bank, power block, configuration block, JTAG block, etc. Doing so can greatly reduce the potential for errors.

Let automation create and maintain the schematic sheets that contain the logic banks. Don't mix power, configuration, and JTAG blocks with logic blocks and don't place user-defined logic (including capacitors, resistors, etc.) on the same sheets as the I/O optimization sheets. Partition the design such that the I/O optimizer is able to take control of the schematics containing the logic banks.

Any team-based design approach, especially one with multiple FPGAs, requires that the project be forked and merged at several strategic points in the design cycle. Getting the most from the tools in these types of situations requires careful planning and forethought. In other words, don't expect the tool to magically solve the project management problems. Take the time to learn how it can help, then deploy it accordingly.

Cleverly partitioning the design into sections that can be attacked as homogenous blocks, and an application of hierarchy that enables the design to be quickly reconstructed from those blocks, can significantly improve design team efficiency.

Summary
This success story makes the assertion that I/O optimization, deployed into a process that takes full advantage of its capabilities, can significantly reduce design cycle times while simultaneously producing a superior design. The following statistics, taken from an Keysight presentation and wrap-up, illustrate this:

- Schematic start to PCB layout complete in less than ten months five months earlier than expected
- Overall system design time reduced by roughly 30-40%
- Pin assignment effort reduced from 4-8 weeks per FPGA to 1-2 weeks per FPGA including time to compile the FPGA to check design rules
- FPGA pinout restrictions could be checked directly against board schematics using the same data, not parallel updates of different data

Automatic generation and maintenance of the FPGA functional and physical schematic symbols helped to significantly reduce design entry errors.

By cleverly architecting their design, and molding their processes to take advantage of their EDA tools, Keysight was able to realize significant productivity gains.

Specifically, they were able to reduce their pin assignment effort from 4-8 weeks per FPGA to 1-2 weeks per FPGA (including the time needed to compile the FPGA to check design rules), going from schematic start to PCB layout complete in less than ten months while reducing overall system design effort by roughly 50% and overall design cycle by 30-40%.

What does this experience mean for Keysight? Explained MacIsaac: “Our success with I/O optimization on this project has convinced us to deploy the tool as a world-wide Keysight standard for FPGA schematic entry. Initiatives to develop best practices for this are underway.”

For the latest product information, call us or visit: w w w. m e n t o r. c o m

©2015 Mentor Graphics Corporation, all rights reserved. This document contains information that is proprietary to Mentor Graphics Corporation and may be duplicated in whole or in part by the original recipient for internal business purposes only, provided that this entire notice appears in all copies. In accepting this document, the recipient agrees to make every reasonable effort to prevent unauthorized use of this information. All trademarks mentioned in this document are the trademarks of their respective owners.