Xcell journal

THE AUTHORITATIVE JOURNAL FOR PROGRAMMABLE LOGIC USERS

Issue 54
Third Quarter 2005

Design Challenges: Avoiding the Pitfalls, Winning the Game

PERFORMANCE

The PlanAhead Experience

COST

Using a CPLD to Implement a QWERTY Keypad

POWER

Performance vs. Power: Getting the Best of Both Worlds

CONNECTIVITY

Managing Signal Integrity

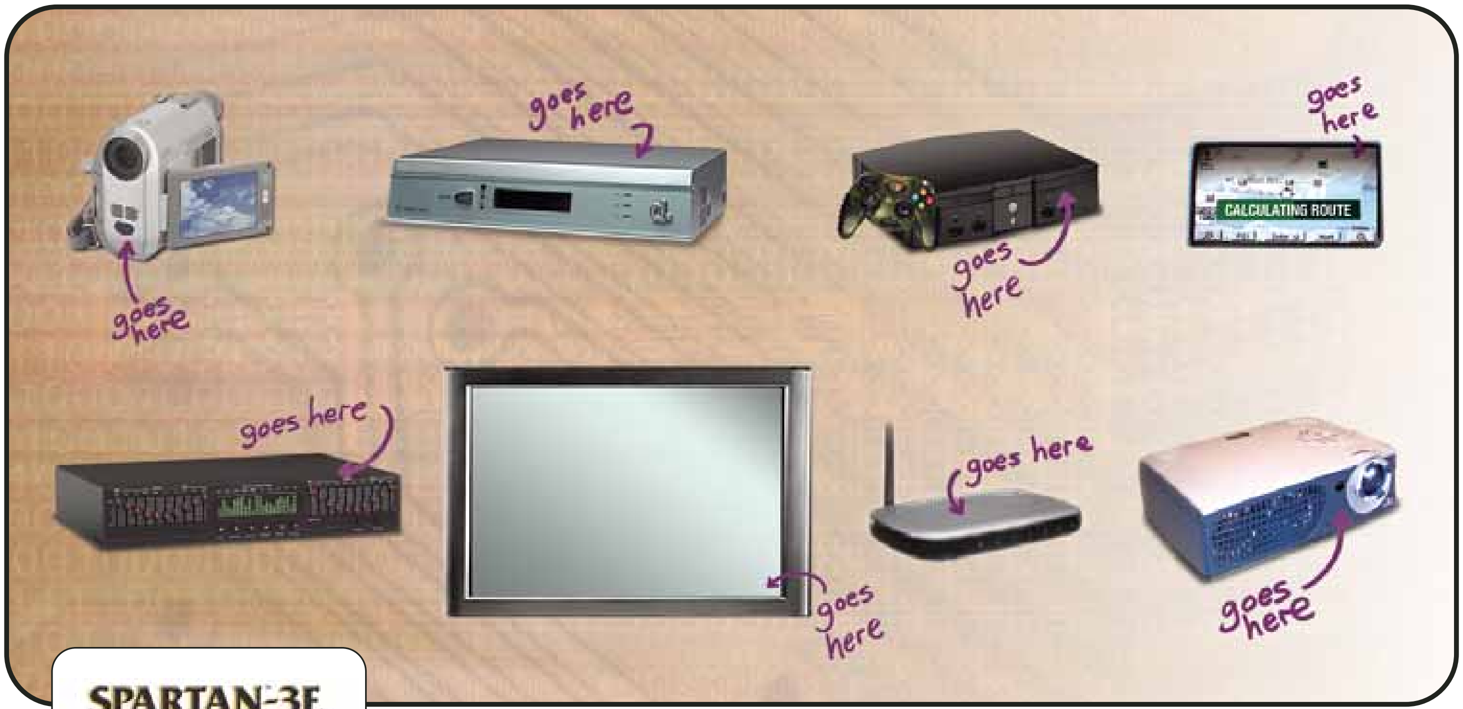
DEBUGGING

Speed FPGA Debug with Mixed-Signal Oscilloscopes

XILINX®

Introducing the new Spartan-3E family — the world's lowest-cost FPGAs

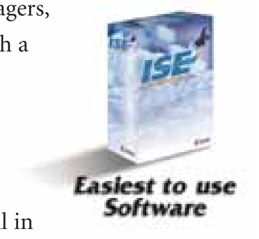
Priced to go.



SPARTAN-3E
FPGA

The industry's first 100,000-gate FPGA for only $2.00*

Spartan-3E Platform FPGAs offer an amazing feature set for just $2.00! You get 100K gates, embedded multipliers for high-performance/low-cost DSP, plenty of RAM, digital clock managers, and all the I/O support you need. All this in a production-proven 90nm FPGA with a density range up to 1.6 million gates.



Perfect for digital consumer apps and much more!

With the Spartan-3E series, we've reduced the previous unit cost benchmark by over 30%. Optimized for gate-centric designs, and offering the lowest cost per logic cell in the industry, Spartan-3E FPGAs make it easy to replace your ASIC with a more flexible, faster-to-market solution. Compare the value for yourself . . . and get going on your latest design!

MAKE IT YOUR ASIC

XILINX®
The Programmable Logic Company℠

For more information visit
www.xilinx.com/spartan3e

Earth Friendly — Pb-free devices available now

FORTUNE® 2005 100 BEST COMPANIES TO WORK FOR

* Pricing for 500K units, second half of 2006

EDITOR IN CHIEF	Carlis Collins editor@xilinx.com 408-879-4519
MANAGING EDITOR	Forrest Couch forrest.couch@xilinx.com 408-879-5270
ASSISTANT MANAGING EDITOR	Charmaine Cooper Hussain
XCELL ONLINE EDITOR	Tom Pyles tom.pyles@xilinx.com 720-652-3883
ADVERTISING SALES	Dan Teie 1-800-493-5551
ART DIRECTOR	Scott Blair

Xcell PUBLICATIONS

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124-3400
Phone: 408-559-7778
FAX: 408-879-4780
www.xilinx.com/xcell/

Life's Challenges: Opportunity or Curse?

"Life is full of challenges." You have probably heard this thought often. But how often have you really used challenges to learn more about yourself and grow from the experience?

You probably view challenges as unwelcome speed bumps on the road to success and happiness. But are they really?

My personal experience has shown that challenges are tremendous opportunities for moving to the next level. Whether personal, professional, or otherwise, challenges can reveal your blind spots, provide the motivation for making a change, and show you where you need to make necessary changes and grow.

At Xilinx®, we are very aware of the design challenges facing today's system designers. Accordingly, we are constantly looking to see what and how we need to change to keep up with market demands. We also realize that we too may have a few blind spots, so we are continually asking our market for suggestions on how to improve.

This issue of the *Xcell Journal* focuses on design challenges, comprising articles on system performance, total cost, power management, connectivity, and debug.

To highlight just a few, Adrian Cosoroaba's article, "Achieve Breakthrough Performance in Your System," discusses how Virtex™-4 FPGAs set new records in system performance while consuming minimal power and providing superior signal integrity.

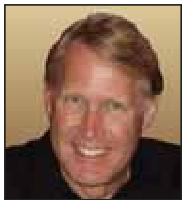
For reducing total cost, there are several articles illustrating examples of low-cost system implementations. They include using a CPLD to implement a QWERTY keypad; implementing video games in FPGAs; using Xilinx ISE™ 7 design tools to get the most out of your silicon while slashing project costs; staying connected while driving safely using advanced telematics; and a low-cost programmable PCI Express solution.

In Anil Telikepalli's power management article, "Performance vs. Power: Getting the Best of Both Worlds," he discusses the fact that excessive power is expensive in many ways. Excessive power creates the need for special design and operational considerations – everything from heat sinks to fans to sophisticated heat exchangers. Even the cost of larger power supplies must be considered.

In the connectivity article "Bridging System Packet Interfaces," co-authored by several Xilinx engineers from the IP Solutions Division, Xilinx IP and reference designs provide superior solutions for implementing custom bridging solutions between protocols. This article discusses the benefits and utility of the SPI-4.2 to Quad SPI-3 Bridge, which demonstrates how to bridge four SPI-3 cores to a single SPI-4.2 core.

And finally, Ross Nelson's debug article, "Hardware/Software Co-Verification," discusses how to gain full visibility into your software and hardware and achieve a faster design iteration loop in the process.

We hope you enjoy this issue on design challenges. And remember: life is full of challenges, but perception is everything.

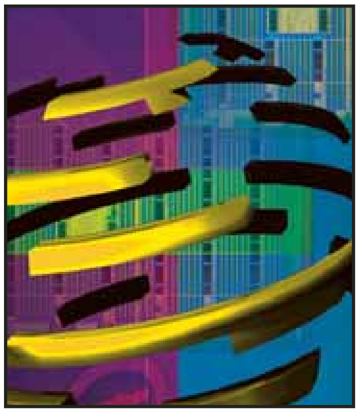


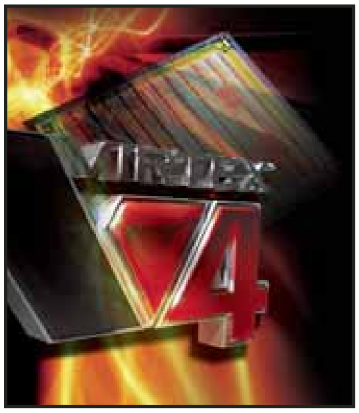
Forrest Couch

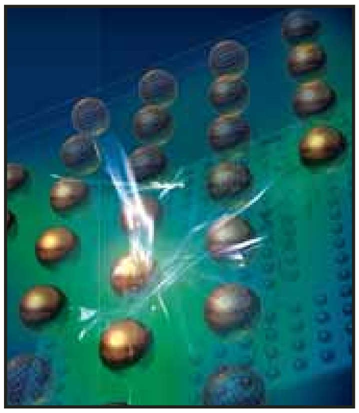
Forrest Couch
Managing Editor

THIRD QUARTER 2005, ISSUE 54

Xcell journal







When Hardware Met Software

Bridging the great divide of hardware/software integration.

by Anthony Townsend
Xilinx Design Services Engineering Manager
Xilinx, Inc.
tony.townsend@xilinx.com

You have been there before – and if not, you will be. After a seemingly infinite number of requirement changes and a schedule so short that you have given up all semblance of a life for the last two months, your design has finally passed all block and system-level simulations. You should be happy. But you are not. Because you are about to enter the most difficult, stressful, and aggravating part of any project: hardware/software integration.

Hardware/software integration is a sore spot at virtually every company. Just ask friends working for other companies; like the pointy-haired manager, hardware/software integration woes are universal. Take a stroll through the technical section of your local bookstore; you will find volumes about designing for high speed, real time, high performance, low power, and test. What you will not find is a book about designing for integration.

No one has figured out the magic formula for integration – not even Xilinx®. This article is not intended to be a cure-all for your integration woes, but we can show you a few ways to ease the pain.

Hardware Said ...

The design flow shown in Figure 1 is typical of what happens in the course of a hardware design from conception to completion. The majority of the flow is straightforward – despite the feedback paths, rarely do you encounter anything that you cannot overcome. Once you have completed coding the design and passed simulation, you either have boards waiting or are expecting them shortly.

Once the initial smoke tests are done on the board, you begin bringing the design to life. You start with basic access tests to ensure that the host can communicate with the board. You write basic software test code to test the interfaces to memory, the interfaces and operation between chips, and the interface to the host processor. You write and execute code testing the memory map, interrupts, status, and control registers, and verify the timing. Now it is time to hand the tested board over to the software team.

Now that you have delivered the board to software, guess what – the hardware does not work. It is irrelevant that the test code you wrote verified the design because nothing works. The data is corrupt, the interrupts don't clear (if they work at all), and you are lucky the system doesn't explode the moment they power it on.

You sit down with the software engineer and ask what they are doing. You explain that your test code works and that you are sure the hardware is correct. You wonder what these guys have been doing for the last six months – don't they even have basic software functioning?

Software Said ...

The design flow shown in Figure 1 is typical of what happens in the course of a software design from conception to completion. The majority of the flow is straightforward – despite the feedback paths, rarely do you encounter anything that you cannot overcome. In the absence

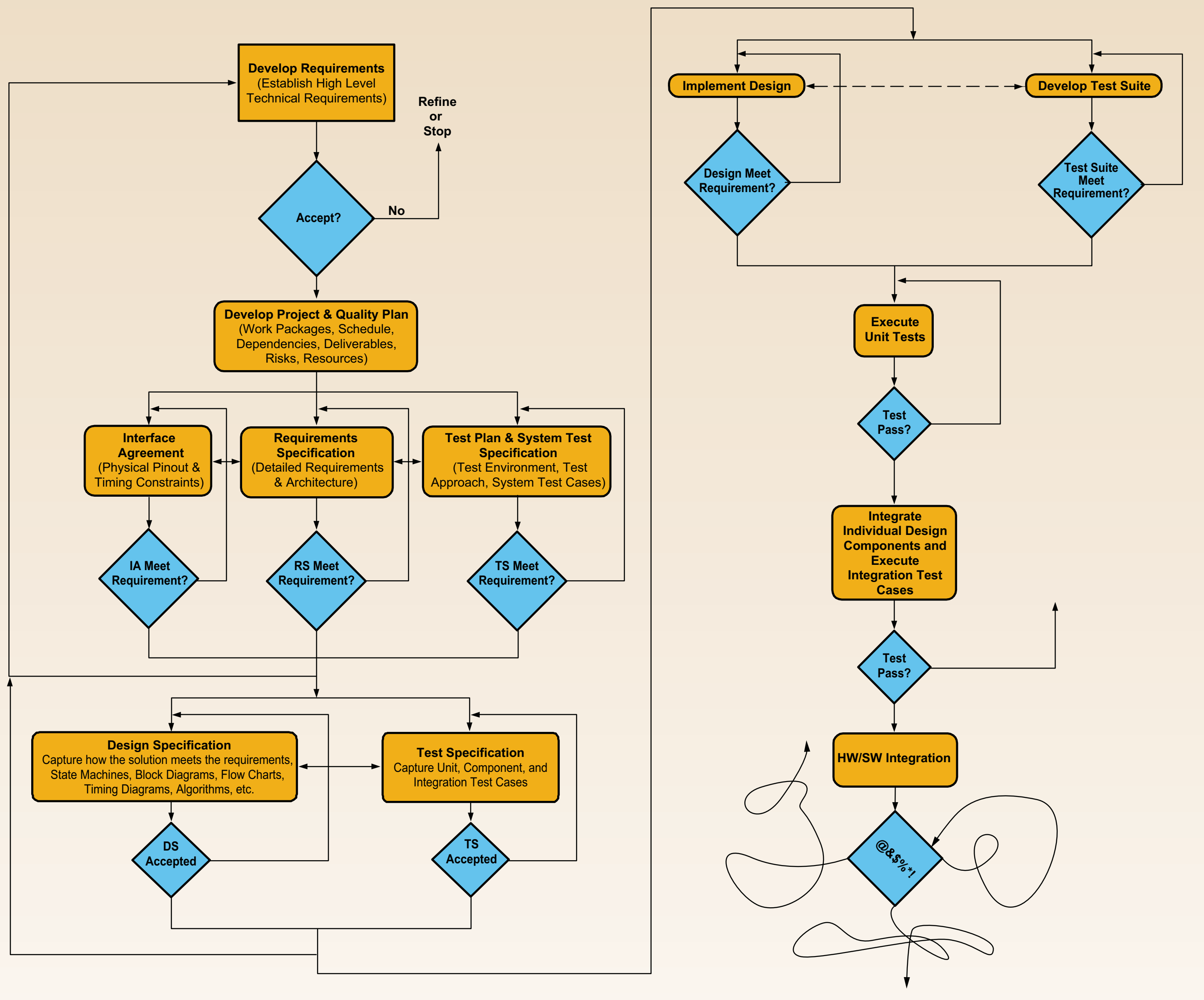

Develop Requirements (Establish High Level Technical Requirements)
Refine or Stop
Accept?
No
Develop Project & Quality Plan (Work Packages, Schedule, Dependencies, Deliverables, Risks, Resources)
Interface Agreement (Physical Pinout & Timing Constraints)
Requirements Specification (Detailed Requirements & Architecture)
Test Plan & System Test Specification (Test Environment, Test Approach, System Test Cases)
IA Meet Requirement?
RS Meet Requirement?
TS Meet Requirement?
Design Specification Capture how the solution meets the requirements, State Machines, Block Diagrams, Flow Charts, Timing Diagrams, Algorithms, etc.
Test Specification Capture Unit, Component, and Integration Test Cases
DS Accepted
TS Accepted
Implement Design
Develop Test Suite
Design Meet Requirement?
Test Suite Meet Requirement?
Execute Unit Tests
Test Pass?
Integrate Individual Design Components and Execute Integration Test Cases
Test Pass?
HW/SW Integration
@&$%*!

Figure 1 – Typical design flow

The issue here is not what is common but what is not. The design community as a whole has spent much more time and resources developing hardware simulators and emulators.

of a hardware platform at the onset of the project, you developed an emulated host environment that allows you to develop and test your code.

You start with basic tests of the OS. You test function calls, interrupts, and the GUI. You optimize the code for performance and verify the operational paths. You check the algorithms and verify the corner cases. You do everything possible to ensure that the software will be ready when the boards are available, only to be told you will receive the hardware six weeks later than expected. (The project end date does not move.) You finally receive the hardware.

You track down the power supply and cables that did not show up with the board. You power up the board – the system appears dead. Now you begin the arduous task of debugging and working through the system. You discover that the hardware is Big Endian, despite an earlier agreement that data will be transferred in Little Endian format.

You sit down with the hardware engineer and ask what they are doing. They immediately start telling you how their test code works and that they are sure the hardware is correct. You wonder what these guys have been doing for the last six months – don't they even have basic hardware functioning?

The Counselor Says ...

Although the preceding sections are somewhat exaggerated, the odds are that you recognized either a past or current project in them. But the reason why you may not see a lot written about this subject is because the issues involved don't make much sense.

If Figure 1 applies to both hardware and software, then obviously the design flows are very similar and almost interchangeable – shouldn't that help with integration? What about the software/hardware interface document (or whatever you call it at your company) that describes how the hardware and software interact – wouldn't it preclude any problems when integrating?

The issue here is not what is common but what is not. The design community as a whole has spent much more time and resources developing hardware simulators and emulators. This began in earnest with the advent of VHDL by the military and has expanded to Verilog and more recently C-to-gates technology. Simulation technology has developed to the point that it is quite reliable and accurate. Most vendors now provide HDL models that you can use to simulate entire hardware systems as opposed to individual devices. On the flip side, the engineering community has largely ignored the other parts of the system.

Software testing without a target platform relies on the designers to build a model of the system on a host. Software engineers have to develop and test the system software using this model. And although this model is, in many cases, even more complex than the software to be designed, it cannot accurately model all of the facets of a system (much less all individual components).

Unlike the hardware models available to hardware designers, few companies offer similar models to software designers. Co-simulation offers little advantage to either software or hardware designers, because it can be prohibitively time-consuming to simulate even a few simple instructions – much less an entire routine or operation.

The design flows may look basically the same, but when it comes to the way in which they are executed they are quite different. It is this difference (and the lack of understanding the difference) that accounts for the majority of the issues during integration. To some extent, the hardware team does not understand that the software team requires the actual hardware to truly validate even what the hardware team may consider a simple function.

On the other hand, the hardware team has become in many cases over-reliant on technology. Most hardware designers will tell you that the person writing a test bench and the person doing the design should be different people. However hardware engineers will write their own test code for board debug and could be lulled into a false sense of validation, making the same wrong assumptions writing the test code as when designing the logic.

To minimize the pain of integration, you must rely not on technology, but communication, understanding, and empathy. Understanding what your counterparts on the other side of the wall are doing and the processes and techniques they follow and use will allow you to better prepare for integration. Communicating with them will minimize the time and pain associated with integration. Initially this will require a little extra effort on both sides.

One of the best ideas is to have the software team write the test code that the hardware team uses to verify the board. Although this is typically not planned or scheduled, it makes sense from several angles. First, if the software team writes the test code, it minimizes the possibility that a requirement could be interpreted differently by the two teams – it will be resolved in this early phase. Second, many of the small problems that can cause major delays later (such as Big Endian versus Little Endian) will be found and their impacts minimized. This, combined with a software test plan that allows the hardware team to know what software will be tested (and in what order) can go a long way in easing integration.

The second idea is to have the hardware team document their design early and often. As the software team cannot truly test until they have target hardware, they must rely heavily on the documentation supplied by the hardware team. It is critical that the software team receives complete documentation – and that the documentation is kept up to date. The hardware team must also consult with the software team before making design changes. Although a change may seem trivial from a hardware perspective, it could result in an extensive software change, particularly in the latter stages of the project.

Finally, the software team must have a stable target platform. This does not mean a perfect, bug-free platform but rather a consistent, known platform. It is important that the hardware team relay any and all known board/logic issues to the software team. This will prevent the software team from wasting time chasing down a known problem.

Happily Ever After ...

Specifications may fluctuate. Timelines may change. Management may have pointy hair. Many technical and operational challenges exist in every project, but nothing jeopardizes a project or brings more stress to a design engineer than when hardware and software are integrated. The challenge is to plan and design for that event from the beginning.

There is no magic formula for a smooth hardware/software integration plan. However, there should be a plan and an understanding of what the other guy is doing – or is going to do. Having this understanding and actually talking with your hardware or software counterpart will make integration a lot less painful. A few sacrifices now will be rewarded in the end.

Although the software team may not have planned on writing test code for the hardware team, burning a few extra cycles to write it will allow both teams to detect and correct problems early. This – combined with a good integration/test plan, well-documented hardware (including unintended design features), and a stable hardware platform – will go a long way in making integration easier.

Complete Software Radio Solution on a single 6U card

Features

- 6 Million gate Virtex II FPGA
  - Scableable Software Defined Radio
  - Embedded Processing via MatLab
- TMS320C6416 DSP
- 105 MHz, 14 bit 2 Ch. Analog I/O
- 32 MB RAM
- 32/64-bit cPCI bus
- PMC expansion site
- STAR Fabric interface

Quixote

Get your data sheets now!
www.innovative-dsp.com/quixote

sales@innovative-dsp.com
805.520.3300 phone • 805.579.1730 fax

Innovative Integration
. . . real time solutions!

FREE Multimedia Demo

Seamless FPGA for Xilinx Virtex-II Pro

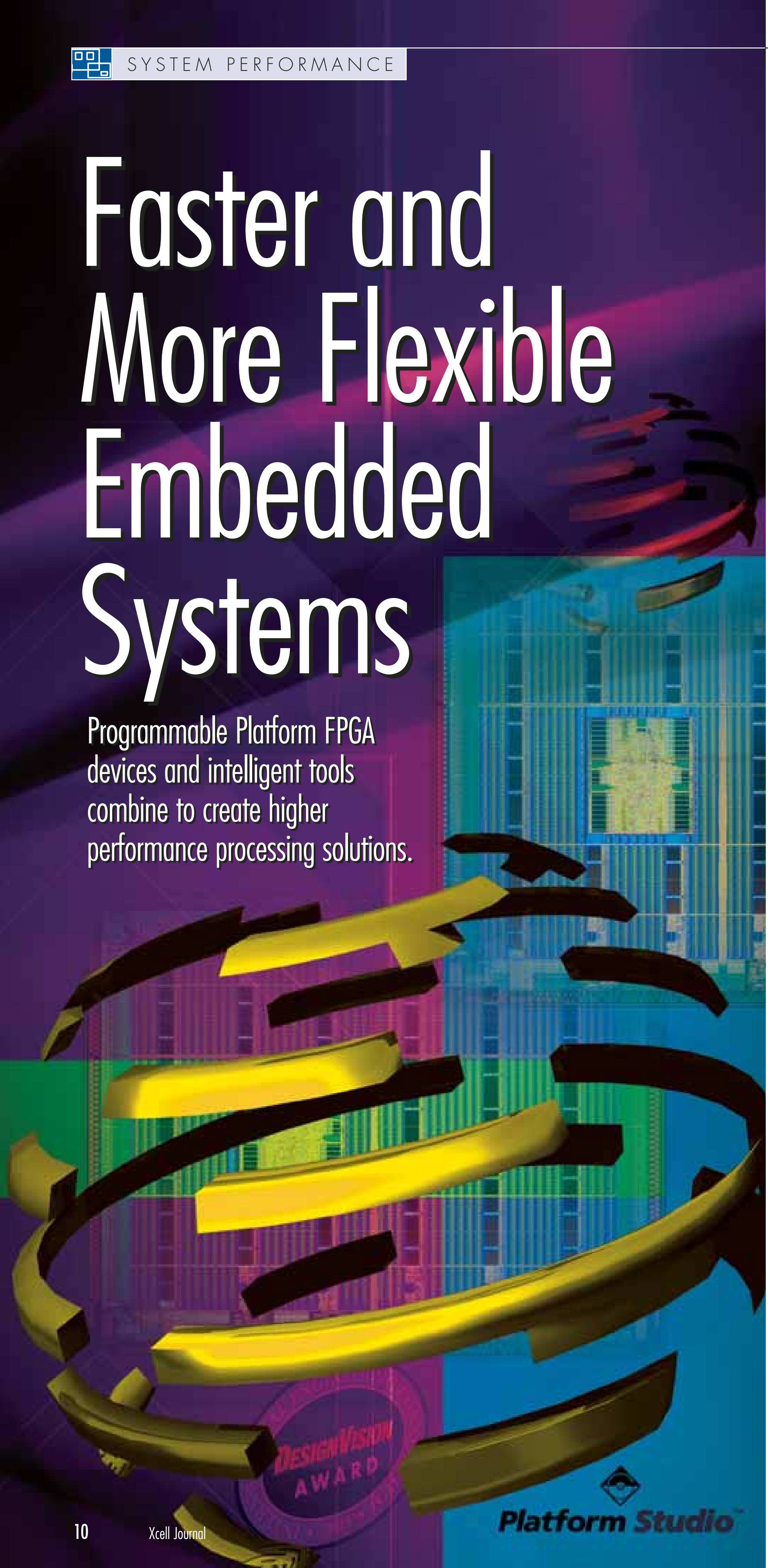
XILINX

During this demo, you will learn the following:
- Simplifying the embedded processor verification flow
- Easily importing designs from Xilinx Platform Studio EDK to Seamless FPGA
- Verification with Seamless FPGA: accelerating the simulation process
- Performance profiling and analysis

Watch the demo today:
http://www.mentor.com/products/fv/hwsw_coverification/seamless_fpga/demo_request.cfm

For additional details about Seamless FPGA, visit us at www.seamlessfpga.com

Mentor Graphics



Faster and More Flexible Embedded Systems

Programmable Platform FPGA devices and intelligent tools combine to create higher performance processing solutions.

by Jay Gould
Product Marketing Manager,
Xilinx Embedded Solutions Marketing
Xilinx, Inc.
jay.gould@xilinx.com

Are there any real-time computing requirements that don't mandate that your new systems be faster and more flexible than your previous designs? Ever-changing industry standards dictate that tomorrow's embedded system designs accommodate a new level of customization, while high performance demands challenge traditional processing designs.

You don't want to be restricted by an overly customized design, and you can't just keep toggling the system clock at ever-increasing speeds to improve performance. There has to be a better way to create faster and more flexible embedded processing systems.

Platform FPGAs are programmable SOCs that support a multitude of sophisticated designs, and include on-board memory, DSP capability, embedded processing, and hardware accelerated co-processing. The re-programmability and field upgradeability of these new devices mean that you can fix bugs, enhance features, optimize performance, and add emerging industry standard support throughout product life cycles and even after deployment in the field.

With these powerful capabilities immersed in a programmable SoC device, all you need are the appropriate tools to unleash and harness this embedded performance.

Intelligent Tools

In recent industry surveys, design engineers made it clear that they often value intelligent tools more than the actual devices and operating systems they use to build their own end products. If this trend is accurate, choosing the appropriate tool suite before beginning your next embedded system design will be crucial to your product schedule and overall success.

Today's development environments need to provide "platform-aware" tools that understand all system options and support multiple types of processor cores, as well as the creation and customization of co-processing and IP. Design wizards and automatic module generation will reduce errors and streamline the development process, while integrating hardware and software debuggers together will enable you to find and fix bugs faster. If you choose wisely, intelligent tools will accelerate development and optimize performance.

Standard Flows and Innovation

Xilinx® created the Xilinx Platform Studio (XPS) tool suite with the development of Platform FPGAs in mind, supporting existing traditional flows for both hardware (HDL/netlists for FPGAs) and embedded

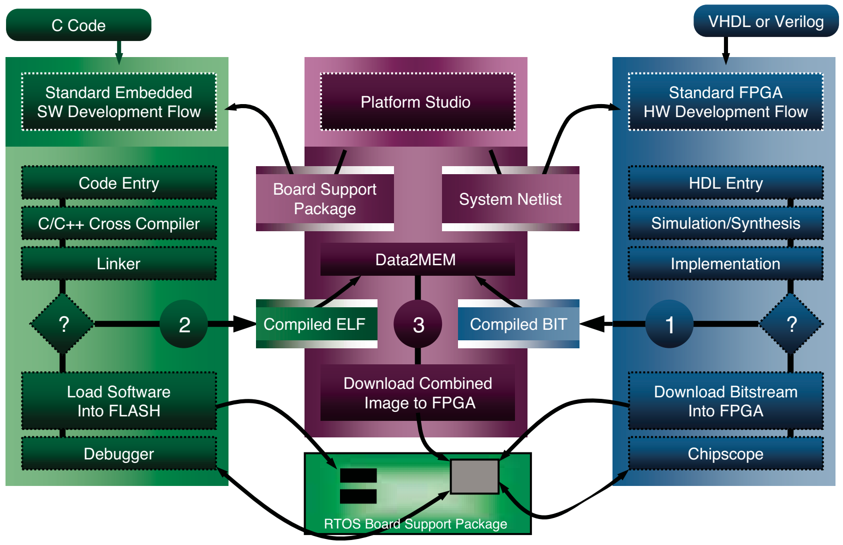


Figure 1 – XPS design flow

software design (C/ELF code for processing core engines) (Figure 1). In addition to providing a unified development tool suite for supporting the complete spectrum of programmable processing solutions, XPS won the IEC's (International Engineering Consortium) DesignVision Innovation Award for introducing new capabilities that accelerate embedded development. With platform-aware tools like XPS, you can quickly create a real-time hardware/software system through an abstract flow of design wizards.

The design wizards guide you through the process of creating a basic system and can reduce errors by masking-off design options not supported by your initial selections and assumptions. For example, although XPS supports both PowerPC™ hard- and MicroBlaze™ soft-processor core designs, the tools are smart enough to remove MicroBlaze options if you have chosen PowerPC, and vice versa.

Importing, creating, and customizing IP is streamlined through a separate design wizard, and supports an IP repository to facilitate IP reuse elsewhere on this design or in the future on a different design.

XPS additionally innovates and accelerates the development process with a variety of automatic generators that replace tedious and error-prone manual design steps. Being aware of the Platform FPGA silicon properties and options, XPS can automatically generate software drivers for selected peripherals, generate sample test code for board options, and even create BSPs (board support packages) for some of the more widely used RTOS/eOS (real-time operating systems/embedded operating systems) such as Wind River Systems's VxWorks or embedded Linux.

XPS also provides a unique utility (Data2Mem) that merges C code into the FPGA bitstream, enabling software development and debug to proceed in real time without time-consuming re-runs of FPGA place and route tools.

Xilinx even provides new efficiencies with a unified JTAG connection methodology that combines FPGA download, FPGA debug, C code download, and software debug capabilities through a single probe. Other traditional methods require multiple probes and switching hardware connections between the different steps.

In fact, XPS uniquely integrates the hardware and software debuggers together so that they can cross-trigger each other. This new visibility into the system allows embedded design teams to find and fix bugs faster, regardless of whether the flaws originate in hardware or software.

Acceleration Through Co-Processing

Let's say that you now have a flexible processor-based platform that satisfies most of your system requirements. How fast can you clock the core to meet your performance requirements?

You probably have realized that clocking your processor faster won't take care of all of your performance challenges. Besides the physical limitations of discrete processors and heat dissipation, accelerated clocking can't ensure that your core can service and complete all the real-time event responses and applications with which you have burdened it. More and more "multiprocessor" solutions are emerging to partition and offload lower priority tasks from a main control processor so that the main unit can ensure real-time responses.

Programmable platforms introduce some additional ways to approach this problem, with off-the-shelf devices that you customize yourself for your own unique applications. Supporting both hard and soft processor cores, one solution offered by Platform FPGAs is to focus high-priority tasks on an immersed hard processor while offloading lower priority tasks to a soft-processor core instantiation. You have the option to add one or more MicroBlaze soft processors to a Platform FPGA device already running an embedded PowerPC engine. Example devices supporting this are the Virtex™-II Pro FPGA or new Virtex-4 FX family devices with built-in PowerPCs. The PowerPC cores in these devices can be complemented with MicroBlaze IP cores inserted as macros and built out of FPGA hardware resources in the silicon.

Another alternate and promising approach is to implement the concept of

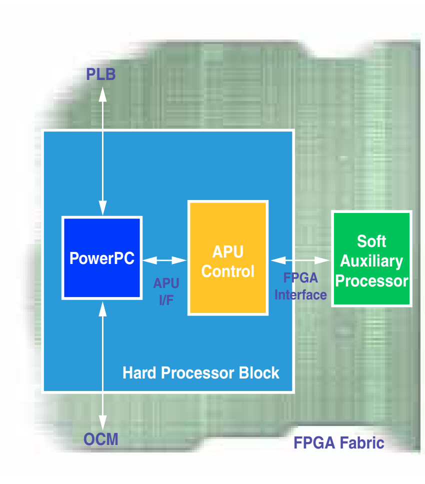


Figure 2 – Virtex-4 FX APU

"co-processing" and use the intelligent tools to build a direct connect from the embedded PowerPC cores to high-performance FPGA fabric, where hardware accelerator functions can operate as extensions to the PowerPC. As shown in Figure 2, you can improve the overall system performance by offloading computationally demanding applications from the main CPU.

By its very nature, FPGA hardware fabric is parallel in structure and can be used to accelerate system functions orders of magnitude faster than clocking methods can provide. In this example, the PowerPC core is complemented by an APU (Auxiliary Processor Unit), which interfaces to a parallel soft processor that can handle applications such as data processing, floating-point mathematics, and video processing. This direct connection provides a high-bandwidth, low-latency solution with parallel advantages over other multi-core processor and arbitrated busing solutions.

Performance Analysis

Do you need to find out where your performance is lost in your design? Embedded software debugging and analysis is always a bit of a challenge because code execution is often "invisible" to you. On paper, your design looks like it meets specifications, but when running in real-time hardware with asynchronous interrupts and real-world situations, you find that often you don't meet your own performance requirements. Now is the time when intelligent tools can provide you with a unique view inside the operating device rather than leave you guessing outside of a black box.

Version 7.1 of Xilinx Platform Studio introduces a series of performance analysis tools and views that provide great insight as to how your software is actually executing and where performance is leaking away from you (Figure 3). By knowing which software functions take up the most execution time and which functions call other functions – as well as the number of times called – you can get an illuminating view of exactly how your embedded design is running. Functions that take a long time to execute, or functions that are called a large number of times by other routines, may be excellent candidates to accelerate by moving them to parallel hardware as co-processing extensions.

Figure 3 also shows that if the tools track and display your software execution clearly, you can quickly and easily identify areas that could be more efficient. This can save a lot of what-if experiment scenarios that are time-consuming and often result in relatively small performance improvements. In-lining some C code or an entire function may provide tiny localized speed-ups, but moving time-consuming routines into high-performance FPGA hardware can often result in an order-of-magnitude improvement. With intelligent views of the code execution by specific function names, you can see exactly which software routines to adjust, providing a much higher return on improving system performance.

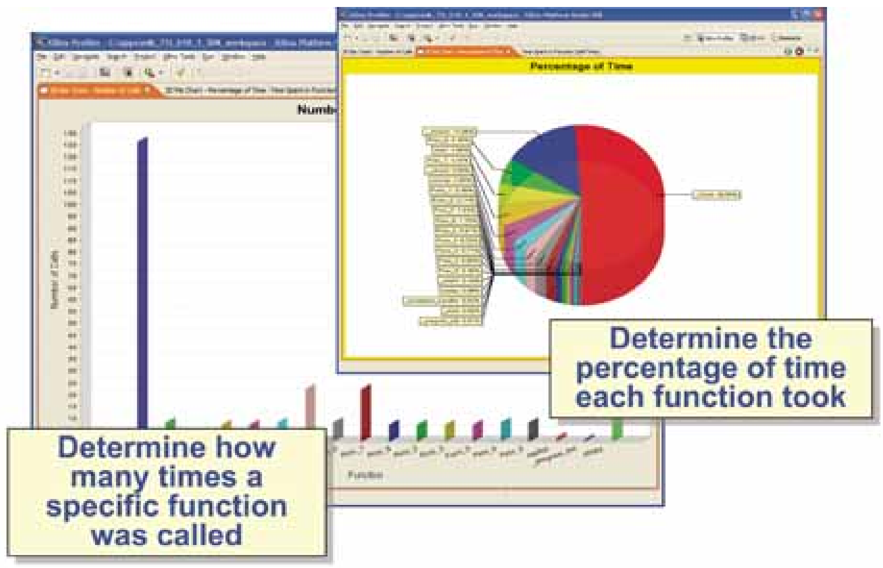

Percentage of Time
Number of Calls
Function
Determine the percentage of time each function took
Determine how many times a specific function was called

Figure 3 – XPS performance analysis views

Conclusion

Intelligent platform-aware tools can help you identify the inefficiencies in your embedded software code and allow you to optimize performance. Knowing which specific software functions you need to streamline allows you to evolve your hardware/software partitioning and accelerate more modules in programmable FPGA fabric.

The high-performance nature of parallel FPGA hardware resources and the advent of easy-to-use, programmable co-processing technologies like the Virtex-4 FX APU enable you to create faster and more flexible embedded processing systems.

Xilinx offers clear advantages for embedded processing over traditional discrete or competitive FPGA solutions. Our tools, combined with our programmable embedded Platform FPGAs, offer a significant performance improvement for real-time developers.

To learn more about the Platform Studio tool suite, please visit *www.xilinx.com/edk*. A good starting point to learn about all of our embedded processing solutions is *www.xilinx.com/processor*.

Designing Control Circuits for High-Performance DSP Systems

These simple techniques could save you days of work.

by Narinder Lall
Sr. DSP Marketing Manager
Xilinx, Inc.
narinder.lall@xilinx.com

Brad Taylor
System Generator Applications Manager
Xilinx, Inc.
brad.taylor@xilinx.com

FPGAs have made significant strides as engines for implementing high-performance signal processing functions, whether for ASIC replacement or performance acceleration in the signal processing chain with DSP processors. Although much has been written about how to use FPGAs as signal processors, not much has been published about building control circuits within such systems.

There are perhaps two key decisions to make when implementing control circuits for FPGA-based DSP systems:

- Should the control circuit be implemented in hardware or developed as a software algorithm?
- What building blocks are available to make the development of the control circuit as efficient and painless as possible?

Software or Hardware?

In this first stage, you can make tradeoffs between algorithms that are implemented in hardware, and those that are better implemented in software using a soft microprocessor (Xilinx® PicoBlaze™ and MicroBlaze™ processors) or hard embedded microprocessor (PowerPC™ 405). Table 1 shows the tradeoffs between hardware- and software-based approaches.

A number of attributes need to be considered when making tradeoffs between these approaches. These include:

- **Algorithm complexity.** You can easily implement simple algorithms (like those that do not need many lines of C-code) in both software and hardware. Although no absolute measure exists to correlate how many lines of C-code represent one slice, a good rule of thumb is that one line equals 1 to 10 slices. When algorithm complexity rises, implementing and testing the algorithm in hardware becomes more challenging. You can more easily implement complex algorithms in lines of C code on a microprocessor, which is the preferred route most designers choose.
- **Need for an RTOS.** If an RTOS is a mandatory piece of the control algorithm, this again favors a software approach that exploits the use of the hard embedded PowerPC on Virtex™-II Pro or Virtex™-4 FX FPGAs, or on external microprocessors. RTOS support for these microprocessors currently includes support from Wind River and MontaVista.
- **Communication with the host.** Communication with a host processor will often – but not always – require a bus architecture of some kind. In this instance, a microprocessor such as the

MicroBlaze processor or PowerPC processor is ideal, as both support bus architectures such as the OPB. Hardware-based host communication using state machines, although possible, can be somewhat more cumbersome.

- **Speed of decisions.** If you specify speed of decisions as clocks per decision, then for decisions that are needed quickly it is obvious that a hardware circuit will be preferred, if not required. For decisions that can be made in hundreds or thousands of clock ticks, software-based algorithms will be sufficiently capable to handle this level of performance.
- **Need for floating point.** Although floating point is largely tangential to control functions, cases do exist where systems employ floating point for control. One example is the calculation of filter coefficients. In sonar systems that require matrices to be inverted, floating-point control is often preferred, as it is often easier to develop with. Floating-point control is also preferred when control precision is high and the algorithms are not available in fixed point.

Once you've decided on hardware or software, you have access to a number of building blocks. Each one is particularly suited to different types of control tasks.

Types of Control Tasks and Possible Circuits

Many different types of control tasks exist. For this article, we have chosen to focus on the following types of problems, which are commonly found in signal processing systems.

- Hardware-Based
  - – Data-Driven Multiplexing
  - – Implementing Finite State Machines (FSMs)
  - – Sample Rate Control
  - – Sequencing – Pattern Generation
- Software-Based
  - – Implementing Low-Rate Control Algorithms
  - – High Complexity Control of Physical Layer Data Paths (MAC layer) (outside the scope of this article)

	Clocks Per Decision	Algorithm Complexity	RTOS	Floating Point	Communicate With Host
Hardware-Based Control	1-10	Simple	No	No	Difficult
Software-Based Control	100-100000	Complex	Yes	Sometimes	Easy

Table 1 – Hardware/software tradeoffs

Table 2 shows a summary of the tools and types of typical control tasks. These are not hard-and-fast recommendations – merely some suggestions for some of the better options. As Xilinx System Generator for DSP is the tool of choice for modeling and designing DSP systems onto FPGAs, in this article we'll also provide some examples using free demos contained within System Generator that demonstrate the use of the control circuit.

Task 1: Data-Driven Multiplexing

An example of a control task that does not require monitoring of the current state is data-driven multiplexing, in which data is monitored and tests are performed on that data. The results of those tests determine the output of the control circuit. Figure 1 shows an example of data-driven multiplexing. Here the function is determined by a simple MATLAB function called xlmax. Input y is selected unless x > y, in which case input x is selected.

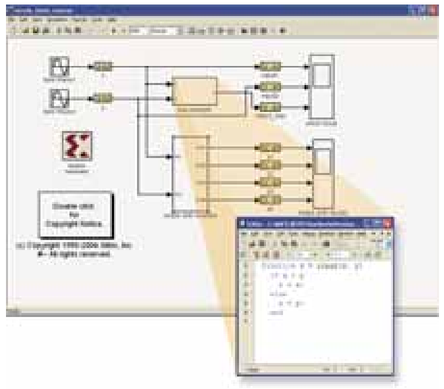


Figure 1 – Data-driven muxing using m-code

Task 2: Implementing FSMs

Finite state machines are used when decisions must be made based on the current "stored" state of the input(s). For hardware-based high-performance DSP systems, it is not uncommon to see circuits where monitoring of state(s) is performed every clock tick.

Although you can implement them in many ways, perhaps the most common ways to implement FSMs within a System Generator design are through m-code CASE statements (popular with algorithm

Toolkit	Class of Control Problem				
	Sequencing	State Machine	Data-Driven Muxing	Sample Rate Control	Low-Rate Control Algorithm
Pattern Generation Components (ROMs, Expressions, Comparators, Counters, Delays)	X				
M-code		X	X		
Comparators/Muxes			X		
FIFOs/Clock Enables/Up/Down Conversion				X	
PicoBlaze					X
MicroBlaze/PowerPC 405					X

Table 2 – Types of control problems and tool kits available in System Generator

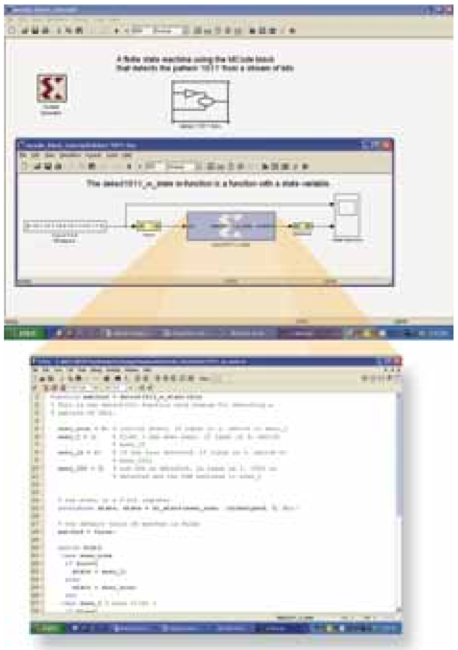

A finite state machine using the MCode block that detects the pattern 1011 from a stream of bits
The detect1011_w_state m-function is a function with a state variable.

Figure 2 – Implementing an FSM in System Generator

developers) and writing HDL (preferred by hardware engineers). HDL can be easily incorporated into System Generator designs using a black box and co-simulated using ModelSim if necessary.

Figure 2 illustrates how easily you can implement an FSM in System Generator using the m-code block that pulls in a MATLAB script contained in the file detect1011_w_state. The purpose of this script is to detect a 1011 pattern from a signal passed through from the MATLAB workspace.

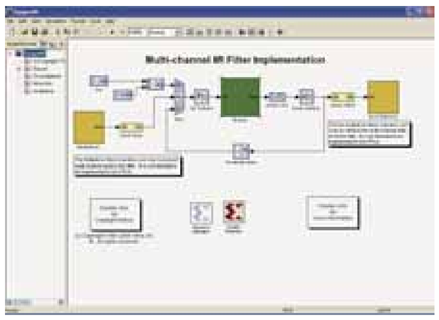


Figure 3 – Sample rate control

Task 3: Sample Rate Control

In high-performance DSP systems, samples often arrive into a system or a piece of a system at a different rate than that of the FPGA clock. We recommend that engineers therefore learn techniques for performing sample rate control. Possible solutions within System Generator that facilitate the design of sample rate control circuits include up/down sampling, clock domains, FIFOs, and clock enables.

Figure 3 demonstrates how you can implement multiple IIR filters using a single time-shared second-order section (biquad). Specifically, 15 distinct IIR filters, each consisting of 4 cascaded biquads, are realized in a "folded" architecture that uses a single hardware biquad. Hardware folding is a technique to time-multiplex many algorithm operations onto a single functional unit (adder, multiplier). For low-sample-rate applications like audio and control, the required silicon area can be significantly reduced by time-sharing the hardware resources.

This design uses a number of control circuits, including a count-limited counter feeding into a two-input mux (that selects between the serial data and the feedback path) and up-sample and down-sample blocks (that control the data rate through the biquad).

Task 4: Sequencing (Pattern Generation)

Sequencing problems usually involve the need for a periodic control pattern that is predictable and not necessarily dependent on the current "stored" state. A common solution for sequencing is to use a simple pattern generator. You can build a pattern generator using building blocks like counters, comparators, delays, ROMs, or the logic expression block within the System Generator block set. The beauty of this underutilized technique lies in its simplicity – yet many designers often opt for more complex, unnecessary state machines.

An example of a pattern generator is contained in the biquad block in Figure 3. The address generator within the biquad block (not shown) generates all of the addresses of the RAMs and ROMs, as well as the write-enable signal for the single-port RAM in the folded biquad module.

Task 5: Low-Rate Control Algorithms

Implementing low-rate algorithms using a Xilinx microprocessor is becoming increasingly common. With a choice of three mainstream processors – the PicoBlaze 8-bit processor, MicroBlaze 32-bit processor, and embedded IBM PowerPC 405 32-bit processor – you have the ability to scale depending on the task at hand. Common tasks that often necessitate the need for on-chip processors include calculating filter coefficients, scheduling tasks, detecting packets (such as in an FEC receiver), and RTOS implementation.

Figure 4 shows a simple control circuit built using the PicoBlaze microprocessor. This example forms the receive path of a 16-QAM demodulator that performs adaptive channel equalization and carrier recovery on a QAM input source. An

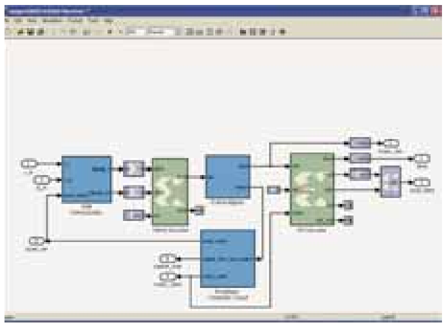


Figure 4 – QAM packet detection using the PicoBlaze microprocessor

attached synchronization marker (ASM) applied by the transmitter is stripped from the demodulated data before concatenated FEC is applied. The PicoBlaze microcontroller controls the RS decoder, maintains frame alignment of the received packets, and performs periodic adjustments of the de-mapping QAM-16 quadrant reference.

Conclusion

When implementing control circuits for high-performance FPGA-based DSP systems, you have access to a number of building blocks within System Generator to make this an easier task. Tables 1 and 2 list some of the tradeoffs to consider and summarize possible control solutions.

All of these designs – and many more – are included within the Xilinx System Generator tool, which retails for $995. You can also try the tool free for 60 days by downloading the evaluation version at *www.xilinx.com/systemgenerator_dsp*.

Achieve Breakthrough Performance in Your System

Virtex-4 FPGAs set new records in system performance while consuming minimal power and providing superior signal integrity.

by Adrian Cosoroaba
Marketing Manager
Xilinx, Inc.
adrian.cosoroaba@xilinx.com

Performance in today's systems is defined by more than FPGA clock rates. Every system has different requirements, and the maximum achievable performance is determined by various factors such as logic fabric performance, I/O bandwidth, embedded processing, and DSP performance, among others. These requirements can also be subject to power restrictions, as well as signal integrity and cost budgets.

Xilinx® developed the Virtex™-4 FPGA family after consulting hundreds of customers to address these requirements and make it easier than ever to meet system performance goals. In this article, we'll look at how Virtex-4 FPGAs provide new and unique capabilities to help you meet diverse requirements for system performance.

System Design Challenges

With each new generation of devices, semiconductor vendors are able to offer higher clock rates, due to shrinking process geometries. However, today's system performance challenges go beyond traditional glue logic and maximized clock rates. In a PC, for example, the real system performance bottleneck lies not in clock frequency but in how the other blocks of the system work together at the desired frequency.

Let's consider these challenges in the perspective of applications employing high-performance FPGAs. Seemingly diverse applications like video stream processing, packet data processing, storage systems, wireless base stations, and many others incorporate similar functions, including:

- Incoming and outgoing data streams
- Bridging multiple connectivity standards
- Arithmetic and DSP (signal conditioning and data processing)
- External memory interfacing
- State machines
- Data buffering
- Embedded processing (Figure 1)

To facilitate these applications, Virtex-4 FPGAs include common building blocks as embedded – yet parameterizable – hard IP. The integration of complex functions like DSP slices, embedded CPUs, dedicated I/O circuitry, and on-chip RAM (block RAM, FIFOs) provides you with unprecedented capabilities to build programmable systems within a single FPGA device.

Meeting system requirements takes the right combination of I/O bandwidth, programmable logic, on-chip RAM, DSP, and embedded processing. To provide the ideal combination of functions, Virtex-4 FPGAs come in three flavors (LX, SX, and FX platforms) comprising 17 devices.

Virtex-4 FPGAs offer not only enhanced logic fabric capabilities, but also customized XtremeDSP™ MACs and embedded PowerPC™ processors that give you enough performance headroom to reach your design performance goals.

I/O bandwidth is often the limiting factor in the quest for performance. To remove I/O bottlenecks, Virtex-4 FPGAs have unique built-in 1 Gbps ChipSync™ source-synchronous circuitry and 622 Mbps to 10.3125 Gbps serial transceivers that can help you achieve bandwidth targets.

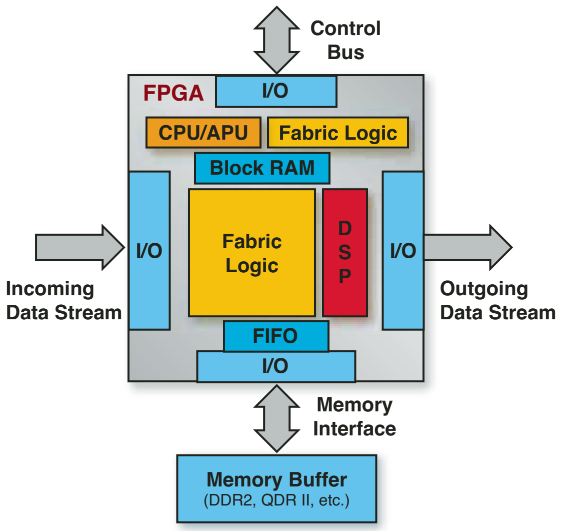


Figure 1 – FPGA-based system

System Performance Categories

Let's look at various aspects of performance and Virtex-4 FPGAs in the context of seven major performance categories: logic fabric, embedded processing, DSP, on-chip RAM, high-speed serial, I/O memory bandwidth, and I/O LVDS bandwidth. Figure 2 offers a comparison with the nearest 90 nm FPGA vendor in each of these categories.

Logic Fabric Performance

Xilinx enhanced the performance of its already fast programmable logic fabric by building Virtex-4 devices with advanced 90 nm technology. A flexible look-up table (LUT) architecture (with the ability to covert any LUT into a 16-bit RAM or 16-bit shift register), a high-speed carry chain, and arithmetic blocks provide further performance gains.

The 500 MHz global clocking structure, the key driver behind logic performance, is fully differential to reduce skew, jitter, and duty-cycle distortion. Virtex-4 FPGAs also provide a hierarchical clocking structure (global and regional clocks) and clock management circuitry. Evaluations of logic fabric performance using a suite of real-world designs demonstrate a performance advantage as much as 70% above our nearest 90 nm competitor. Averaged across this suite of designs, the Virtex-4 performance advantage is 15%. This performance boost means that Virtex-4 devices effectively provide an extra speed-grade advantage.

Embedded Processing

Virtex-4 FX platform FPGAs provide up to two enhanced PowerPC 405 cores, each delivering 702 DMIPS performance

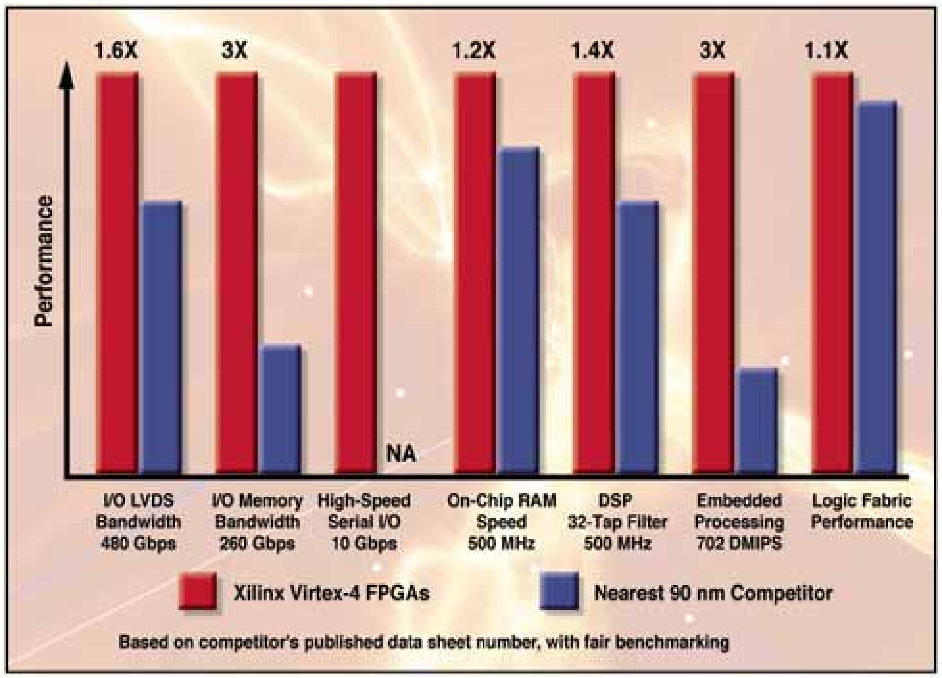


Figure 2 – Performance comparison for Virtex-4 FPGAs

at 450 MHz, while consuming only 0.45 mW/MHz. This is more than three times the performance of the best soft microprocessor cores.

Moreover, the new Auxiliary Processor Unit (APU) controller makes it easy to reach even higher levels of performance by integrating custom co-processors and hardware accelerators. The APU controller provides a low-latency path for connecting co-processor modules implemented in the FPGA to the embedded PowerPC processor. These user-defined, configurable hardware accelerator functions operate as extensions to the PowerPC 405, offloading the CPU from demanding computational tasks. For example, implementing floating-point calculations in hardware improves performance by a factor of 20 over software emulation. A 10/100/1000 Mbps tri-mode Ethernet MAC implemented alongside a PowerPC processor enables Ethernet connectivity.

DSP Performance

The XtremeDSP™ slice is a versatile, user-configurable block providing twice the DSP performance of previous implementations while drawing less than 1/7th the power. Each slice contains a dedicated two's complement, signed 18 x 18 bit multiplier, and a three-input adder/subtracter/accumulator with feedback path. With as many as 512 XtremeDSP slices running at 500 MHz, a single Virtex-4 FPGA delivers 256 GigaMAC/s (18 x 18 GMACs) performance.

You can configure the XtremeDSP slices to implement multipliers, counters, multiply-accumulators, and many more functions, all without consuming logic fabric resources. The ability to implement complex systolic functions without incurring the delay of fabric routing provides significant performance gains. For example, in a 32-tap FIR implementation, the Virtex-4 FPGA outperforms competing devices by 40%.

On-Chip Memory Performance

The Virtex-4 family carries forward the size and basic structure of on-chip memory, 18 Kb dual-port block RAM (proven in previous generations), but adds a data-output pipeline register to increase speed to 500 MHz. The two ports still have individual width control, and in write mode you can choose between automatically reading the previously stored data or the new data. Two neighboring block RAMs, when combined, form a 32K x 1 RAM without loss of speed, or a 512-deep 64-wide RAM with automatic Hamming error correction – without using any extra logic.

Each block RAM also contains its own FIFO controller, a unique Virtex-4 FPGA feature that provides 500 MHz functionality without additional logic resources. Compared to competing devices, the block RAMs provide at least 20% better performance.

But getting your FPGA internal blocks to run fast is only half the battle. Maximum system performance requires efficient interaction between the FPGA and other components in your system. Virtex-4 FPGAs offer the flexibility to achieve the highest possible bandwidth for chip-to-chip, board-to-board, and box-to-box connectivity.

High-Speed Serial I/O

As designs move to faster interface speeds, serial interconnect saves power and board space while reducing design complexity and cost. Virtex-4 RocketIO™ MGTs offer performance from 622 Mbps to 10.3125 Gbps, one of the broadest ranges offered by any device. The transceivers are fully programmable and can implement a myriad of speeds and serial standards. Link-layer IP is available for such standards as PCI Express, Serial-ATA, Fibre Channel, Gigabit Ethernet, and Aurora.

Memory I/O Bandwidth

The great majority of systems today need a data buffer external to the FPGA for temporary storage. This buffer's bandwidth can be the critical factor in determining overall performance.

Memory interfaces like DDR2 SDRAM, QDR II SRAM, or RLDRAM II are source-synchronous, with per-pin data rates of more than 533 Mbps. Memory bandwidth is determined not only by the per-pin data rate but also by the width of the bus. The ChipSync circuitry built into every I/O simplifies the physical layer interface and provides the capability to implement buses three times wider than other programmable solutions, for bandwidths as high as 260 Gbps.

To enable reliable data capture, ChipSync circuitry also includes built-in delay elements, adjustable in 75 ps increments, to ensure the proper alignment between clock and data signals. The unique capability to calibrate timing at run time, rather than at design time, substantially improves design margins. Xilinx also provides hardware-verified reference designs, development systems, and software tools to further speed up the implementation of memory interfaces.

LVDS I/O Bandwidth

ChipSync technology simplifies the design of differential parallel bus interfaces, with embedded SERDES blocks that serialize and de-serialize parallel interfaces to match the data rate to the speed of the internal FPGA circuits. Additionally, this technology provides per-bit and per-channel de-skew for increased design margins, simplifying the design of interfaces such as SPI-4.2, XSBI, and SFI-4, as well as RapidIO.

Virtex-4 FPGAs incorporate ChipSync technology into every I/O, providing the most flexible I/O solution available. This enables wider 1 Gbps LVDS buses for up to 480 Gbps bandwidth, 60% higher than the competition.

Other Performance Challenges

Achieving the desired system performance with your FPGA is often impeded by signal integrity, cost, and power budget restrictions.

The innovative Application Specific Modular Block (ASMBL) architecture enables I/O, clock, power, and ground pins to be located anywhere on the silicon chip, not just along the periphery. This architecture alleviates the problems associated with I/O and array dependency, power and ground distribution, and hard-IP scaling.

Furthermore, the Virtex-4 FPGA packaging technology, SparseChevron, enables distribution of power and ground pins evenly across the package. The benefit to you is improved signal integrity. As demonstrated by Dr. Howard Johnson, Virtex-4 FPGA devices have seven times less simultaneously switching output (SSO) noise and crosstalk when compared to competing devices.

The ASMBL architecture, with its column-based implementation of programmable logic, DSP slices, block RAM, I/O columns, MGTs, clocking, and PowerPC embedded cores, provides another significant benefit in that it allows a more flexible allocation of resources. This enables Xilinx to offer three Virtex-4 FPGA platforms: the LX platform, optimized for logic resources; the SX platform, optimized for DSP; and the FX platform, optimized for embedded processing and high-speed serial applications.

Device power budgets impose an additional impediment to meeting performance goals. Because power consumption increases with clock rate, you may exceed your power budget at frequencies below your performance target, even if your chosen device has more performance on tap. Selecting a device with low power consumption will help you achieve performance goals while staying within your power budget, and can deliver the additional benefits of lower system cost and higher reliability through reduced power supply and cooling requirements.

Virtex-4 FPGAs incorporate unique triple-oxide 90 nm technology that significantly reduces static power. Additionally, by implementing commonly used functions such as embedded IP, Virtex-4 FPGAs further reduce dynamic power when compared to previous generations or competing devices. Measurements and analysis of Xilinx against competing tools and silicon show that Virtex-4 FPGAs consume 1 to 5W less than the competition's 90 nm FPGAs.

Conclusion

Virtex-4 FPGAs incorporate innovative built-in silicon features, extensive embedded IP, triple-oxide 90 nm technology, and unique packaging to provide designers with capabilities that enable breakthrough performance at the lowest cost.

For more information about getting started with your Virtex-4 FPGA design, visit *www.xilinx.com/virtex4*.

Nucleus

Embedded FPGA made easy.

Increase your embedded software performance by using Nucleus PLUS on your processor-based Xilinx designs. By integrating with Xilinx's MLD technology, Nucleus PLUS is configured to your system design automatically, making software integration with your FPGA design easy. The compact design of Nucleus PLUS can run from on-chip memory to help minimize power dissipation and deliver increased performance. This, combined with a wealth of middleware, including the Nucleus NET TCP/IP solution supporting Xilinx Ethernet IP, makes it ideal for products targeted at the networking, telecommunications, datacoms and consumer markets.

Nucleus PLUS is readily available for:

- **MicroBlaze**
- **PowerPC 405**
- **Spartan-3 and Virtex-4**

Discover how Nucleus software can make your embedded development for Xilinx FPGAs easy. Go to our dedicated Web page for more articles and free downloads at **www.acceleratedtechnology.com/xilinx.**

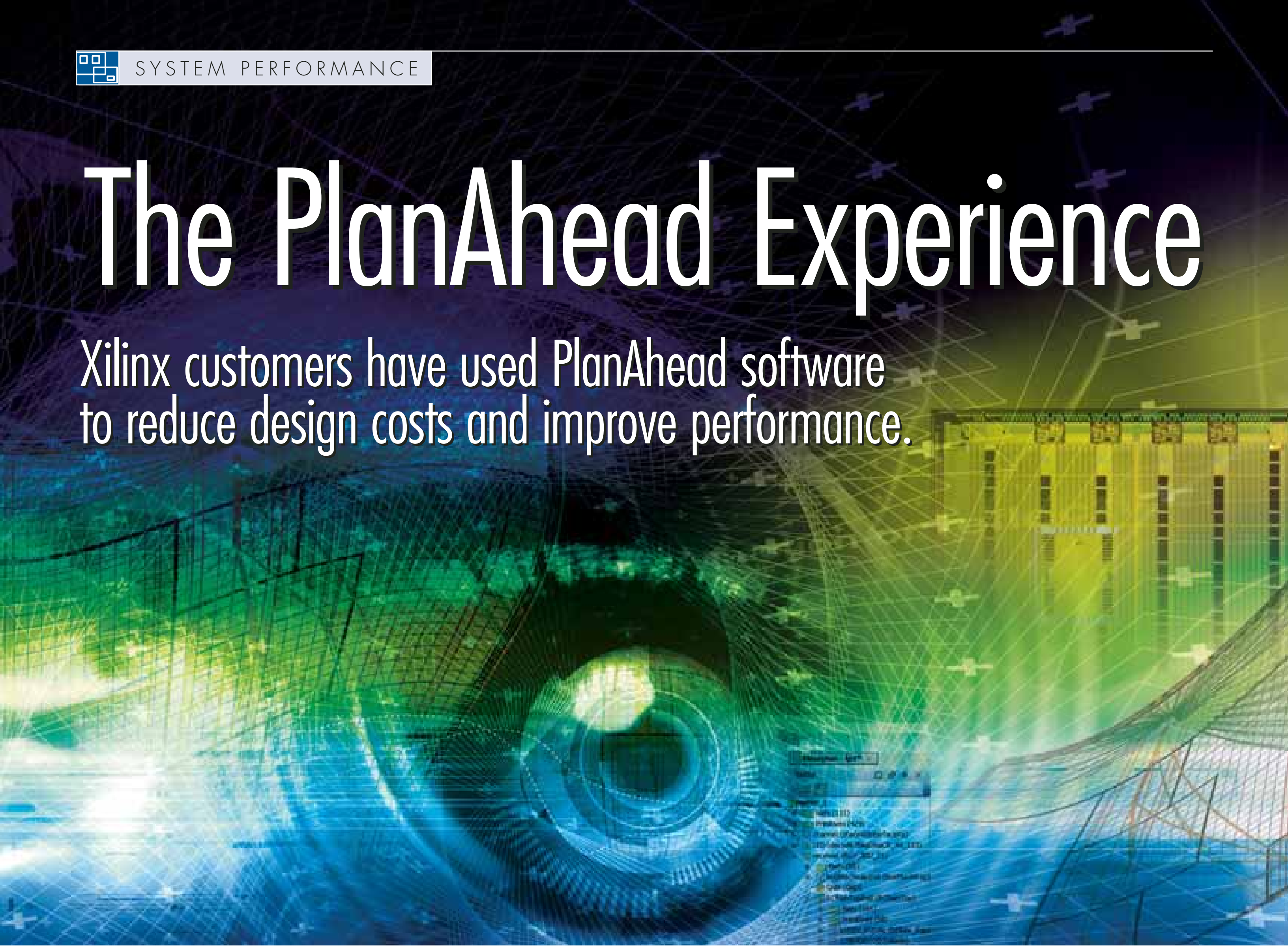
Nucleus EDGE, our new embedded development environment based on Eclipse, offers a complete tools solution for developing designs with Xilinx PowerPC cores. Nucleus EDGE provides you with a Builder, Editor and Project Manager, utilizing optimizing compilers, comprehensive embedded debuggers and RTOS profiling tools.

For more information on the Nucleus complete development solution from UML through to embedded RTOS, go to **www.acceleratedtechnology.com.**

Nucleus. Embedded made easy.

Accelerated Technology®
A Mentor Graphics Division

Accelerated Technology, A Mentor Graphics Division
info@AcceleratedTechnology.com • www.AcceleratedTechnology.com



The PlanAhead Experience

Xilinx customers have used PlanAhead software to reduce design costs and improve performance.

by Chris Zeh
Product Applications Engineer
Xilinx, Inc.
chris.zeh@xilinx.com

A growing number of Xilinx® customers are enjoying improved performance, reduced engineering time and design costs, and the ease of use offered by the PlanAhead™ design and analysis tool. These customers are expressing the same realization – a complete paradigm shift in their FPGA design flow.

Over the past several weeks, I have talked with several application engineers at Xilinx about their experiences with PlanAhead software. I found three main hurdles where the PlanAhead design tool offers a significant advantage: when the design fails to meet timing; when the design doesn't fit into the target device; and when the place and route run time is too long. I also found that many of the engineers use PlanAhead software to view the results from place and route.

In this article, I'll discuss the processes that these engineers and I use, and provide statistics on what customers are seeing from using PlanAhead software. When doing your own floorplanning, remember that poor floorplanning can give you worse timing, larger device utilization, and longer place and route run times. My goal is to outline the concepts to help you accomplish your performance goals.

Failure to Meet Timing

When we as application engineers receive a design that has difficulty meeting timing, we run the design without any floorplanning constraints. This entails removing existing area groups and larger component physical constraints but keeping pin placement. If the place and route time is very long, then we run TimeAhead, the static timing tool within PlanAhead software, to get an early estimation of the critical paths. The TimeAhead analysis also helps identify areas of the design that need RTL revisions to add registers to pipeline critical paths.

If possible, we run place and route on the design and view the placement and timing results within the PlanAhead environment. The timing results from the placed and routed design show us the critical paths based on actual delays (Figure 1). The device view displays the placed design, along with the timing paths from the TRCE report.

We create Pblocks or area groups based on critical paths of the design from the timing report. These Pblocks can be floating or defined as a specific rectangle or shape. Pblocks can contain logic from anywhere in the design and are not limited to the RTL logic hierarchy. Analyzing the imported placed and routed design, we use the current placement of the elements that make up the Pblock to determine a good starting point for the Pblock rectangle.

Pblocks can direct the flow of the design based on the connectivity of different modules. The connectivity of the I/Os to and between modules is shown for the placed and routed design through net bundles. We create Pblocks to cover the critical paths, which are usually associated with getting on or off the device.

The schematic view also displays the connectivity of the design (Figure 2). Pblocks are created either in the device, schematic, or netlist views. The critical paths and other timing paths relative to the existing floorplanning can be highlighted in the device and schematic views.

Normally, the schematic view is used to examine the critical modules and paths without worrying about the actual placement of those modules. The congestion around existing Pblocks and the resource utilization of a Pblock are seen in the device view. If the timing is critical within the Pblock (indicated by a high resource utilization count) then we resize the Pblock to give the congested logic more room to meet timing by condensing the logic to shorten the interconnect lengths and delays. We can also move the Pblocks to alleviate congestion or merge congested logic into a single Pblock.

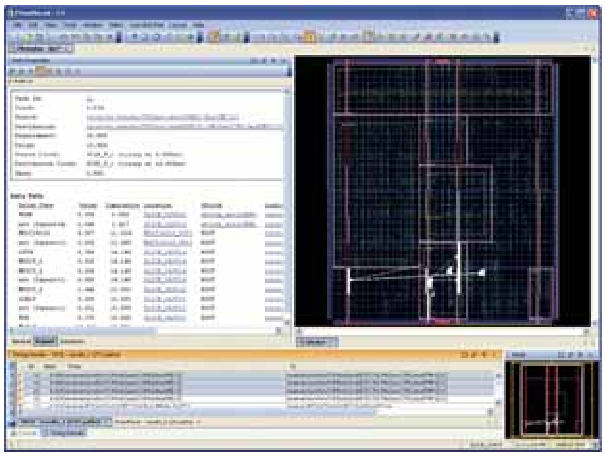


Figure 1 – Timing analysis and placement view of a sample design

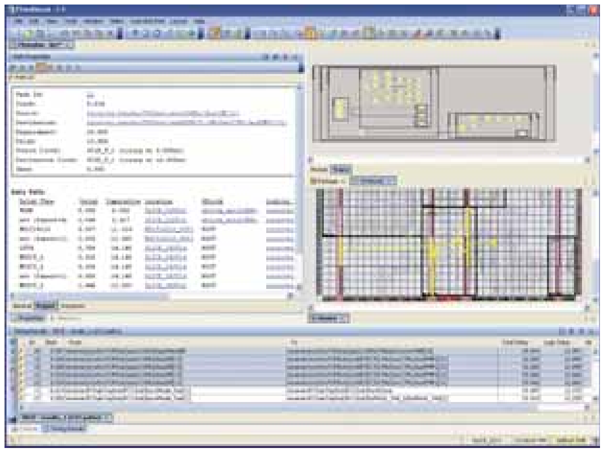


Figure 2 – Design connectivity in the schematic view and timing analysis with the larger components placed

We create and place the Pblocks, then run place and route with our floorplanning constraints. We also lock the locations of the larger components and critical logic, which includes block RAMs, DSP48s, and DCMs. Then we run place and route on the design and repeat the process by reviewing the timing report and placement of the design and modifying the constraints (Figure 2). We can also create smaller "child" Pblocks inside other Pblocks to help the grouping of a few components for timing-critical paths.

Place and route is run on individual Pblocks in addition to the entire design. This helps determine if timing can be met inside the module or Pblock without affecting the rest of the design. If we are unable to meet timing on a specific Pblock, we go back to the source code and re-write it.

PlanAhead software can import the updated netlist for any module and keep the Pblocks originally created. After meeting timing requirements, we place location constraints on all of the elements of that Pblock. This helps ensure that timing is retained when the rest of the design is placed and routed.

The majority of timing failures that we see are related to setup violations. We also see hold violations, which are tricky to fix. Most of the time, a hold violation occurs because of the placement of critical clocking components. We get a large number of clock skew and hold violations when the DCM and corresponding global buffer are placed on opposite sides of the device.

PlanAhead software also has a robust set of DRC checks to highlight these types of issues early in the flow. We use the schematic and device views to view the placement of these components. We then move the placement of either the DCM or the global buffer so that these components are near each other.

The Design Doesn't Fit

When a design has difficulty fitting into the target device, we run it without any floorplanning constraints. Usually, overlapping Pblocks exist, which we remove. We place the large components manually, as we described in the previous section, to help the place and route tools.

If timing is not a critical issue for this design, we then create Pblocks of the major hierarchical blocks of the design. The Pblocks are used to direct the flow of the design based on connectivity of the module with net bundles. The bundles show the amount of connectivity based on the size of the bundle. We place the Pblocks close to the components that drive them based on the net bundles and Pblock statistics. We also place the entire design into a Pblock and use the compression attribute on the Pblock(s) to tell MAP to pack this portion of the design as tight as possible. This attribute opens up more room on the rest of the device for the remaining logic, but has a negative effect on timing.

We also run place and route on the individual Pblocks of the design. This helps ensure that the logic placed in the Pblocks can be placed and routed within the defined Pblock size and eventually in the device specified. We can manually compress the size of the Pblock until MAP fails. Once MAP fails, we revert back to the last known good size of the Pblock.

The statistics of the Pblock may report that we used more than 100%, but MAP will determine if it passes. We repeat this process if other non-timing critical Pblocks exist and use the compression attribute to pack those Pblocks as tight as possible.

At this point, we run place and route on every Pblock as well as the entire design. If it still does not fit, we pick a larger device, or go back to the source code and rewrite it. The majority of the time, we run place and route on a Pblock-by-Pblock basis and import the updated module into the PlanAhead design tool. This process retains the existing Pblock size and physical location constraints. Once the design fits into the desired device, we lock the physical locations for the entire design.

Place and Route Time is Too Long

For the majority of designs received by the application engineers, the previous two issues are the most critical. Because the place and route run time decreases once these issues are resolved, the tools do not require the same long run times necessary to meet performance goals. If none of the previous issues are of concern, we then view the timing report and placed and routed design to create Pblocks of the major modules.

The key to decreased place and route run times is the physical location constraints for the entire design. To find the correct placement constraints for the design, we run place and route on each Pblock and size it according to the timing constraints for each Pblock. This is also possible when utilizing an incremental update approach based on the logical hierarchy in the design. PlanAhead software can selectively update any module in the design with an iterated module netlist. This allows us to take advantage of an incremental synthesis approach. We can determine the placement of Pblocks by viewing the connectivity between modules and I/O pads, and also between the modules themselves. The non-timing-critical Pblocks can be compressed manually and the larger components placed.

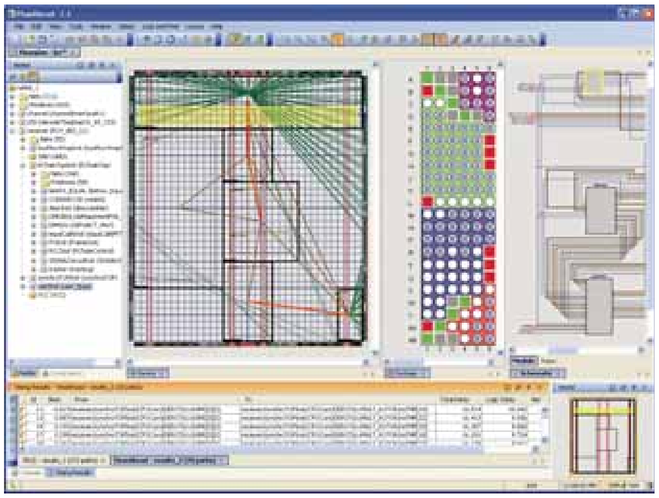


Figure 3 – The device, package, and schematic views of the design with timing analysis from TimeAhead

	Before PlanAhead Software	After PlanAhead Software
Customer A	~90 % LUT Utilization Fails Timing	~82 % LUT Utilization Meets Timing with Two New Netlists
Customer B	Fails Timing Request: 250 MHz Actual: 235 MHz	Meets Timing with 0 Timing Errors and 0 Timing Score
Customer C	Fails Timing Long Place and Route Run Time	Timing Errors and 0 Timing Score Reduce Place and Route Run Time by ~85%
Customer D	Fails Timing Device is Too Small for Design	Timing Errors and 0 Timing Score Reduced Utilization by 11% and Now Fits
Customer E	Fails Timing	Reduce Timing Score by ~98% Reduced Timing Errors by ~97%
Customer F	Fails Timing Long Place and Route Run Time	Reduced Timing Errors by ~90% Reduced Place and Route Run Time by ~30%

Table 1 – Results from customers using the PlanAhead design tool

Place and route is then run on the design, repeating the process by reviewing the timing report and placement of the design and modifying the constraints accordingly. You may have to combine the approaches and try these techniques over several interactions if you have a combination of several of these issues.

View the Design

Many of the application engineers use the PlanAhead design tool as a visual representation of the design before and after running place and route. We use the tool to graphically see the timing paths and placement of the design. The schematic, device, and package views give us different angles on the design (Figure 3). The device view gives us the ability to see the correlation between the SLICE/CLBs and the I/O banks. Timing paths are also clearly displayed and corrective action can be taken.

Through interacting with the application engineers, the vast majority of customers have reported good experiences and results. Table 1 illustrates some customer issues before and after using PlanAhead software.

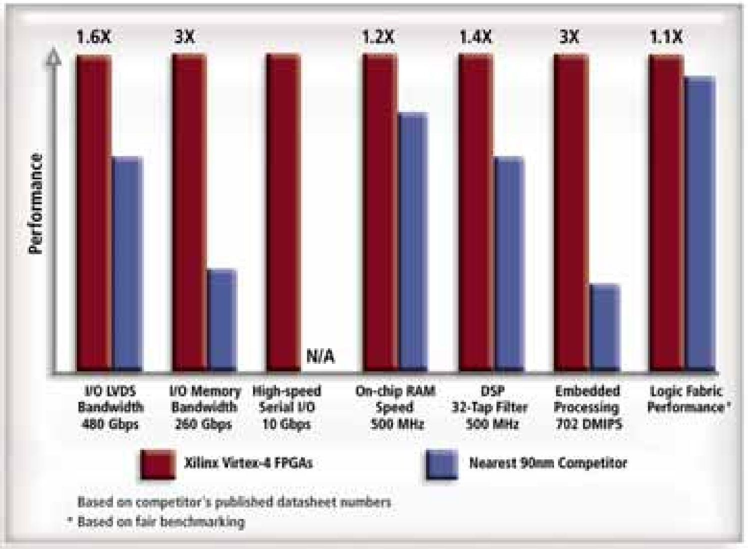
Conclusion

In this article, I have discussed the ways that Xilinx customers and application engineers are using the PlanAhead design tool to overcome three main areas of concern. These concepts should help you accomplish your performance goals. I hope you will agree with one recent customer, who stated, "PlanAhead software allows us to achieve quicker verification and prototyping, with fewer iterations."

For more information about the PlanAhead design tool, visit *www.xilinx.com/planahead.*

Real Leadership. Real Value. Real Performance.

LX SX FX



Virtex-4 beats competing FPGAs in every performance category

Virtex-4 FPGAs beat the competition in EVERY performance category.

Only Virtex™-4 Platform FPGAs offer you superior performance in every aspect of a system. No other FPGA comes close to Virtex-4, which delivers a significant advantage in I/O bandwidth, on-chip RAM speed, DSP & processing compute bandwidth, and logic fabric performance. Having all the right features is the only way to achieve breakthrough performance at the lowest cost.

The World's Fastest AND Lowest Power FPGAs!

Virtex-4 FPGAs also give you high performance without breaking your power budget, with significantly lower power consumption than competing 90nm FPGAs, 73% lower static power, up to 86% lower dynamic power, and 94% lower inrush current.

Visit our website today, and find out more about the *real* leader in FPGA performance.

XILINX
The Programmable Logic Company℠

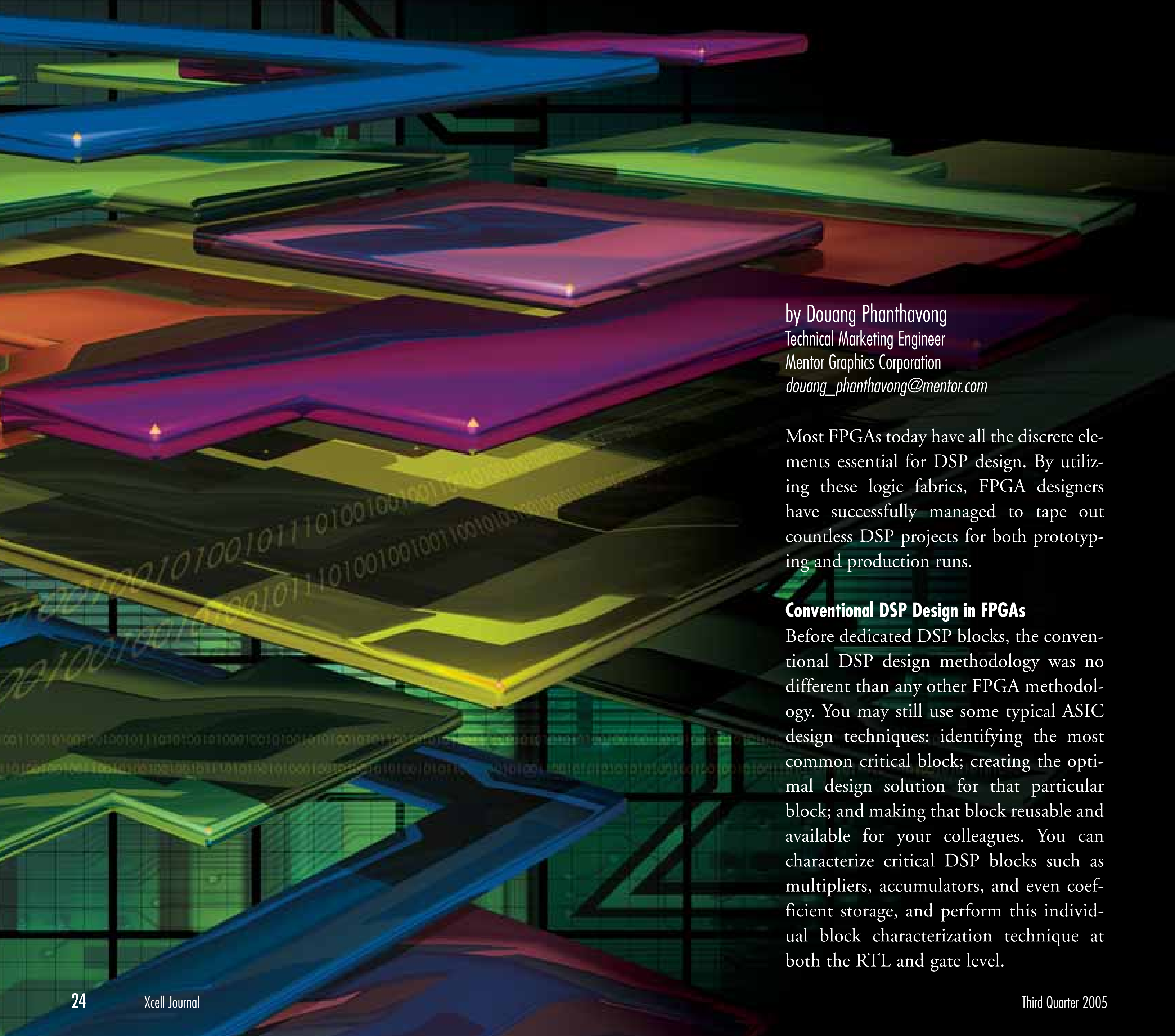
www.xilinx.com/virtex4/performance

www.techonline.com
View The TechOnLine Seminar Today

Breakthrough Performance At The Lowest Cost

Designing with DSP48 Blocks Using Precision Synthesis

Achieve close to custom silicon performance in FPGA DSP designs.



by Douang Phanthavong
Technical Marketing Engineer
Mentor Graphics Corporation
douang_phanthavong@mentor.com

Most FPGAs today have all the discrete elements essential for DSP design. By utilizing these logic fabrics, FPGA designers have successfully managed to tape out countless DSP projects for both prototyping and production runs.

Conventional DSP Design in FPGAs

Before dedicated DSP blocks, the conventional DSP design methodology was no different than any other FPGA methodology. You may still use some typical ASIC design techniques: identifying the most common critical block; creating the optimal design solution for that particular block; and making that block reusable and available for your colleagues. You can characterize critical DSP blocks such as multipliers, accumulators, and even coefficient storage, and perform this individual block characterization technique at both the RTL and gate level.

In addition, you may also use techniques such as special floorplanning, block-based design (bottom-up methodologies), and special constraints files to meet design requirements. After going through these design processes, you still have to surmount a few big hurdles, such as minimizing place and route iterations and interconnect delays. Unfortunately, unlike logic delays, you have little control over these obstacles.

Typically, the FPGA design timing budget has already factored in these routing/interconnect delays. For many DSP design applications, however, exceeding certain routing delay limits is simply unacceptable. That is why dedicated DSP blocks in high-end FPGAs – such as the Xilinx® XtremeDSP™ slice (also referred to as DSP48) in Virtex™-4 devices – are playing a critical role in designing high-performance DSP systems.

How to Compete with ASICs

How important is it to achieve a maximum operating bandwidth (sometimes referred to as "custom silicon performance") when designing a complex DSP system using current FPGAs? The obvious answer, of course, is "very important." Until recently, the only available dedicated arithmetic block was the multiplier, a key element for DSP functions.

To be more competitive with ASICs in this space, however, all major DSP elements – including multipliers, adders, subtractors, pipeline registers, and other arithmetic operations – must perform close to custom silicon levels. Access to a synthesis tool that allows you to take full advantage of this advanced DSP silicon functionality is also important. In this article, we'll use the DSP48 dedicated block available in Virtex-4 devices and the Mentor Graphics Precision Synthesis tool as examples to illustrate relevant challenges and solutions.

Dedicated DSP48 blocks, combined with the advanced compiler in Precision Synthesis, enable you to seamlessly implement various high-performance DSP functions. Applications benefiting from these features include digital media and broadcasting and VoIP, among others.

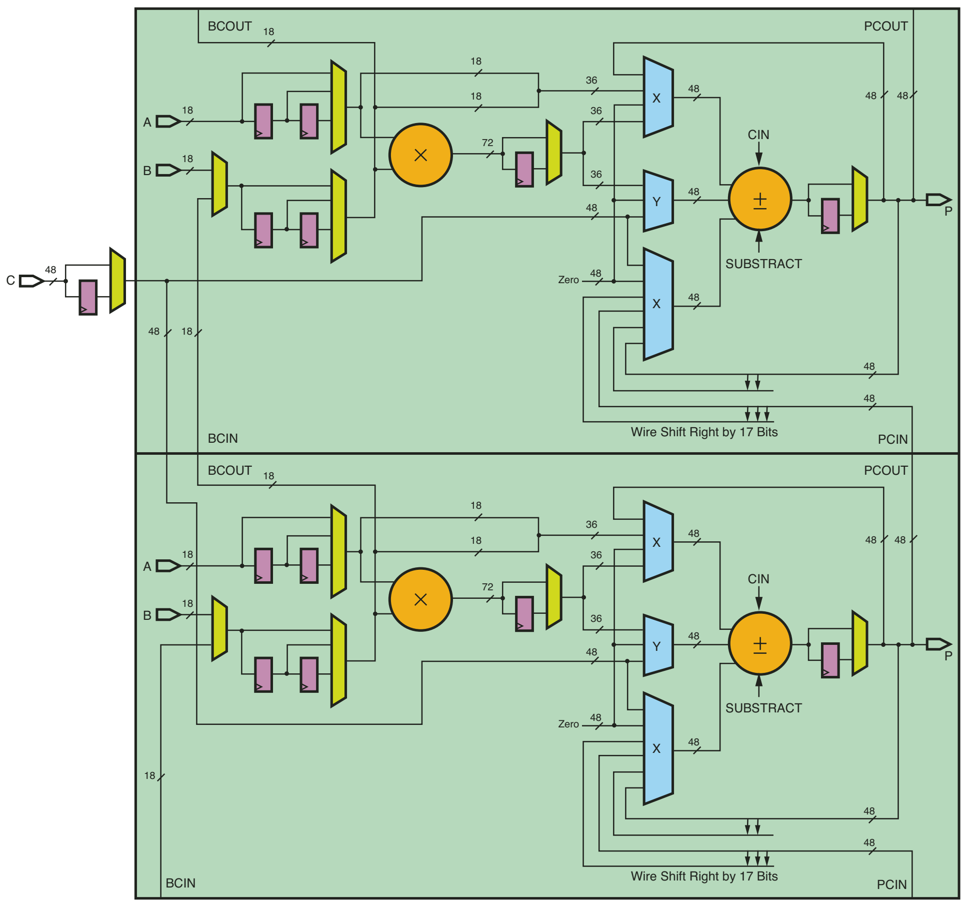


Figure 1 – DSP tile comprising two DSP48 slices

Not long ago, data paths were an obstacle when implementing DSP functions in FPGAs. They were mostly contributed by arithmetic operations such as multiplication, addition, and subtraction. A simple math equation of $Y = (C \pm (A * B) + CIN)$ can generate a plethora of LUTs with multiple levels of logic. On the other hand, a well-integrated DSP48 slice coupled with the powerful new features of an advanced tool like Precision Synthesis can enable you to extract very close to custom silicon performance on your next DSP design.

DSP48 Slice: Features and Functions

A DSP48 tile comprises two DSP48 slices, a shared 48-bit C bus, and internal dedicated interconnect. The DSP48 slice itself comprises the all-important elements for DSP functions (Figure 1). The math portion of the DSP48 slice comprises an 18 x 18-bit two's complement multiplier followed by three 48-bit datapath multiplexers, followed by a three-input 48-bit adder and subtractor. The data and control inputs feed directly to the arithmetic portions, or are optionally registered once or twice (by AREG and BREG) to accommodate the construction of various highly pipelined DSP applications.

Multiplier

The multiplier accepts two 18-bit two's complement operands producing a 36-bit two's complement result. The result is sign-extended to 48 bits and can optionally be fed into an adder/subtractor to form a MULT-ADD arithmetic function. The adder/subtractor accepts three 48-bit two's complement operands, and produces a 48-bit two's complement result.

When the result is sign extended from the 36 bit to 48 bit, the most significant bit (MSB) is simply copied 12 times to make a

48-bit result. For example, a “36’b 0101 1111 1111 10101 1101 1110 1111 1000 1110” result would become “48’b 0000 0000 0000 0101 1111 1111 10101 1101 1110 1111 1000 1110”.

Precision Synthesis supports this automatic sign extension when any of the DSP48 operators are inferred. This is a simple concept, but very powerful, because it allows you to perform wider arithmetic operations without having to manually set the correct bus width for the output result.

Accumulator (Adder/Subtractor)

The adder/subtractor stages are functions of the inputs, which are driven by the upstream multiplexers, carry-select logic, and multiplier arrays. The CIN, X multiplexer output, and Y multiplexer output are always added together. You can control this combined result to be selectively added to or subtracted from the Z multiplexer output: Adder Out = (Z ± (X + Y + CIN)).

Pipeline Registers

This is a unique advantage of the DSP48 block compared to other DSP FPGA architectures. Each DSP48 slice contains the following pipeline registers at each stage:

- One or two pipeline registers for A and B inputs (AREG and BREG)
- One pipeline register at the output multiplier stage (MREG)
- One pipeline register at the output stage (PREG)
- One pipeline register at the C input
- One pipeline register for opmode and other control signals

DSP48 Ports

The DSP48 slice input and output ports support many common DSP and math algorithms. Two direct 18-bit input data ports are labeled A and B. As shown in Figure 1, two DSP48 slices within a DSP48 tile share a 48-bit input data port labeled C. Each of them has one direct 48-bit output port labeled P, a cascaded input datapath (B cascade), and a cascaded output datapath (P cascade), providing a cascaded input and output stream between adjacent DSP48 slices.

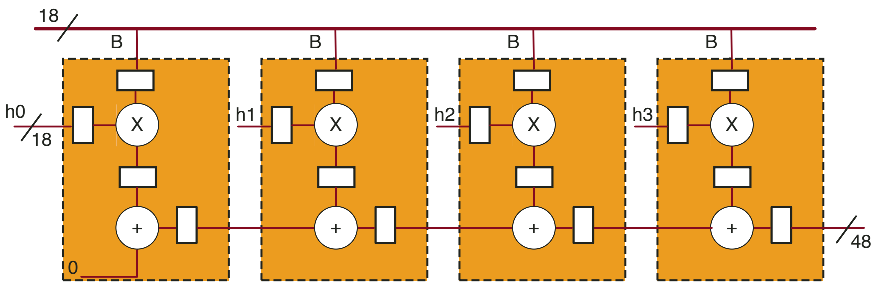


Figure 2 – A transposed FIR filter block diagram

Operating Mode

The 7-bit operating mode (opmode) inputs provide a way for the design to change its functionality from clock cycle to clock cycle if desired. There are more than 40 dynamically controlled opmodes, although you cannot set all possible combinations (as described in the Virtex-4 datasheet, *www.xilinx.com/bvdocs/userguides/ug073.pdf*). The opmode bits can be optionally registered under the control of the configuration memory cells. Precision Synthesis automatically assigns a correct opmode for each DSP48 operator being inferred. The synthesis tool uses a simple control signal – along with the arithmetic operations in the HDL source code – to accurately determine each DSP48 function.

```
module top (h, B, P, clk);

     parameter tap = 4;
     parameter sub_in_width = 18;
     parameter sub_out_width = 48;
       `define  sub_out_width 48

     input  clk;
     input  signed[sub_in_width-1:0]  B;
     input  signed[sub_in_width*tap-1:0]  h;
     output signed[sub_out_width-1:0] P;

     // Wiring array declarations being used for internal signals
     wire   signed[sub_out_width-1:0] PCOUT_INT  [0: tap-1];
     wire   signed[sub_out_width-1:0] PCIN_INT   [0: tap-1];


     mult_add U0 (.A(h[sub_in_width-1:0]),
                  .B(B), .clk(clk),
                  .PCIN(`sub_out_width'b0),
                  .PCOUT(PCOUT_INT[0]) );
     generate
     genvar i;
     for (i=1; i < tap; i=i+1) begin : dsp48
          mult_add  UI (.A(h[i*sub_in_width+(sub_in_width-1) :  i*sub_in_width]),
         .B(B),
         .clk(clk),
         .PCIN(PCOUT_INT[i-1]),
         .PCOUT(PCOUT_INT[i]) );
     end
     endgenerate

     assign P = PCOUT_INT[tap-1];

     endmodule
 //========================================

  module mult_add(B, A, PCOUT, PCIN, clk);
  parameter sub_in_width = 18;
  parameter sub_out_width = 48;

  input  clk;
  input  signed[sub_in_width-1:0] A,B;
  input  signed[sub_out_width-1:0] PCIN;

  output reg signed[sub_out_width-1:0] PCOUT;

  //a,b,m,and p pipelined registers
  reg    signed[sub_in_width-1:0]  breg;
  reg    signed[sub_in_width-1:0]  areg;
  reg    signed[sub_out_width-1:0] mreg;

  always @ (posedge clk)
   begin
     areg   <= A;
   breg   <= B;
   mreg   <= areg * breg;
     PCOUT  <= mreg +PCIN;
    end

  endmodule
```

Figure 3 – Verilog coding examples for FIR filter adder tree design using DSP48 in Virtex-4 FPGAs

Precision Synthesis with Virtex-4 FPGAs

In advanced FPGA architectures, the basic DSP building blocks (for delay, data storage, multiplication, addition, subtraction, summation, and accumulation) are no longer built using discrete components. To tightly integrate these essential DSP components in high-end FPGAs, a well-planned DSP block is designed as part of the FPGA chip’s dedicated resources. A Virtex-4 DSP48 block supports many independent functions, including multiplier, multiplier-accumulator (MAC), multiplier followed by adder, three-input adder, barrel shifter, and pipeline.

To take full advantage of these DSP blocks without having to learn about the implementation details in depth, any good FPGA synthesis tool must provide intelligent and accurate inference and mapping capabilities for DSP functions. Using the Precision Synthesis tool, you can focus your design time more effectively on more important tasks and critical deliverables and meet increasingly tight project schedules.

Using Precision Sythesis, ISE 7.1_sp1	Target Device	LUTs	Reg	Block Mult & DSP	Post PnR Timing
Virtex-4 (1.5v, 90-nm copper process)	4VSX35ff668: 12	0	0	16	2.151ns
Virtex-II Pro (1.5v, 0.13 µm, 9-layer copper process)	2VP30ff1152: 7	720	1332	16	4.998ns
Virtex-II (1.5v, 0.15/0.12 µm, 8-layer metal process)	2V4000ff1152: 6	720	1332	16	6.905ns
Virtex-E (1.8v, 0.18 µm metal, 6-layer process)	v3200efg1156: 8	8722	2165	0	16.005ns
Virtex (2.5v, 0.22 µm, 5-layer metal process)	v800fg680: 6	8722	2165	0	20.704ns

Table 1 – Precision Synthesis post-place and route area and timing results

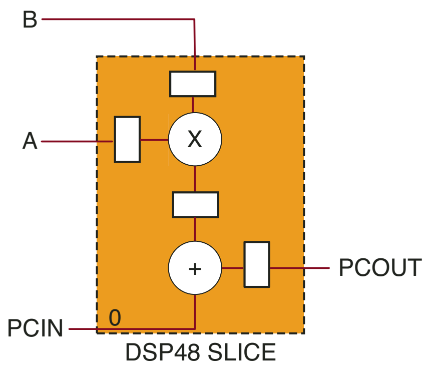


Figure 4 – A transposed MULT-ADD block diagram

Transposed FIR Filter

Figure 2 shows a block diagram, whereas an example of the coding style of a transposed FIR filter structure is illustrated in Figure 3. The post-place and route area and timing results are in shown in Table 1, in which Precision Synthesis will infer the MULT_ADD operator (block diagram shown in Figure 4); map MULTIPLIER with input-pipeline registers; and adder logic using DSP blocks.

As you can see from Table 1, Precision Synthesis uses the same RTL design to target different Virtex families, starting from the first Virtex device (2.5v, 0.22 µm, five layers metal process) to the latest Virtex-4 device (1.5v, 90 nm copper process). You may notice that the exponential QoR improvement begins with the Virtex-II family; most of the improvement was contributed by the integration of a dedicated multiplier in Virtex-II FPGAs.

Since then, these dedicated resources have been continuously enhanced and eventually transformed into the advanced DSP48 slice in the Virtex-4 family. Based on the results in Table 1, it would be difficult for anyone to doubt the tremendous QoR improvements that the Virtex-4 device has brought to today's FPGA DSP design community. The advantages of dedicated DSP48 slices over discrete DSP elements are also quite clear.

The transposed FIR filter structure in Figure 2 is optimal for use with the DSP48 slice. Precision Synthesis can absorb all FPGA fabric into DSP48 slices, including pipeline registers, adders, and multipliers.

You may choose to use one of many different approaches to code this DSP design block. Let's describe one approach where you only have to implement a simple MULT-ADD operator and use Verilog 2001 to generate the rest of the blocks. The design specifications are:

- Signed 18-bit input sampled (B(n))
- Signed 18-bit input coefficients (h(n))
  - Use registers to store coefficients
- Signed 48-bit output stream (P(n))
- 16 taps

Precision Synthesis supports the Verilog 2001 generate statement, which you can use to generate this MULT-ADD operator as shown by the coding example (Figure 3) and technology schematic (Figure 5). The pros to using the transposed FIR filter far outweigh the cons.

Advantages:

- Low latency – the maximum latency never exceeds the pipelining time through the slice containing the first coefficient. Typically, this is three clock cycles from the time data is input to the displayed result.
- Efficient mapping to the DSP48 slice – mapping is enabled by the adder chain structure of the transposed FIR filter.
- No external logic – no external FPGA fabric is required, enabling you to achieve the highest possible performance.

Disadvantages:

- Performance may be limited by a high fan-out input signal if a large number of taps exist.

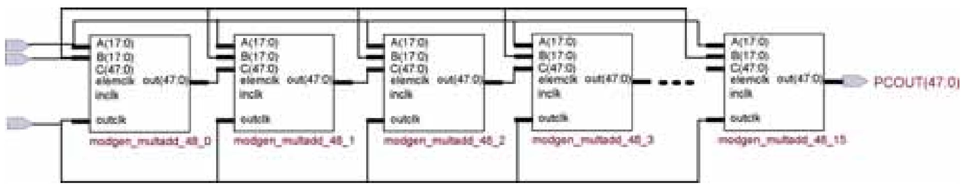


Figure 5 – Technology schematic for the 16-tap transposed FIR filter design

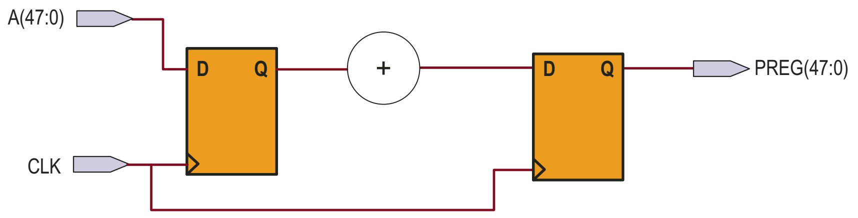


Figure 6 – RTL schematic for a simple INCDEC design

Mapping Beyond Multipliers

Precision Synthesis automatically infers and maps all multiplier and arithmetic operators into DSP48 where possible. In DSP designs, however, critical data paths are not necessarily always lying on multipliers. Adders, subtractors, counters, and other operators could also be the source of the critical timing path. To help deal with these situations, Precision Synthesis lets you control each arithmetic operator by individually manipulating the mapping. These features deliver two main advantages:

- An extra boost in solving timing problems when necessary.
- DSP mapping controllability when desired because of resource availability and timing requirements.

Precision Synthesis gives you the option to map the following operators into a DSP48 slice. Figure 6 shows the RTL schematic of a DEC operator:

- Adder/subtractor/addsub – default to CARRY CHAIN and/or LOGIC
- INC/DEC/INCDEC – default to CARRY CHAIN and/or LOGIC
- EQ/NEQ/LT/LTE/GT/GTE – default to CARRY CHAIN and/or LOGIC
- Counter – default to CARRY CHAIN and/or LOGIC
- Mult/mult-add/mult-acc – default to DSP48

You can map the DEC operator into a DSP48 slice by using the Precision Synthesis GUI (Figure 7) or interactive command line as follows:

- From the RTL hierarchical browser, right-click on the DEC operator. Select Set Attributes > New.
- Enter attribute name and value.
- You can use the command line to accomplish the same task: set_attribute -design rtl -name use_resource -value DSP48 -instance rtlc_1_dec_0.

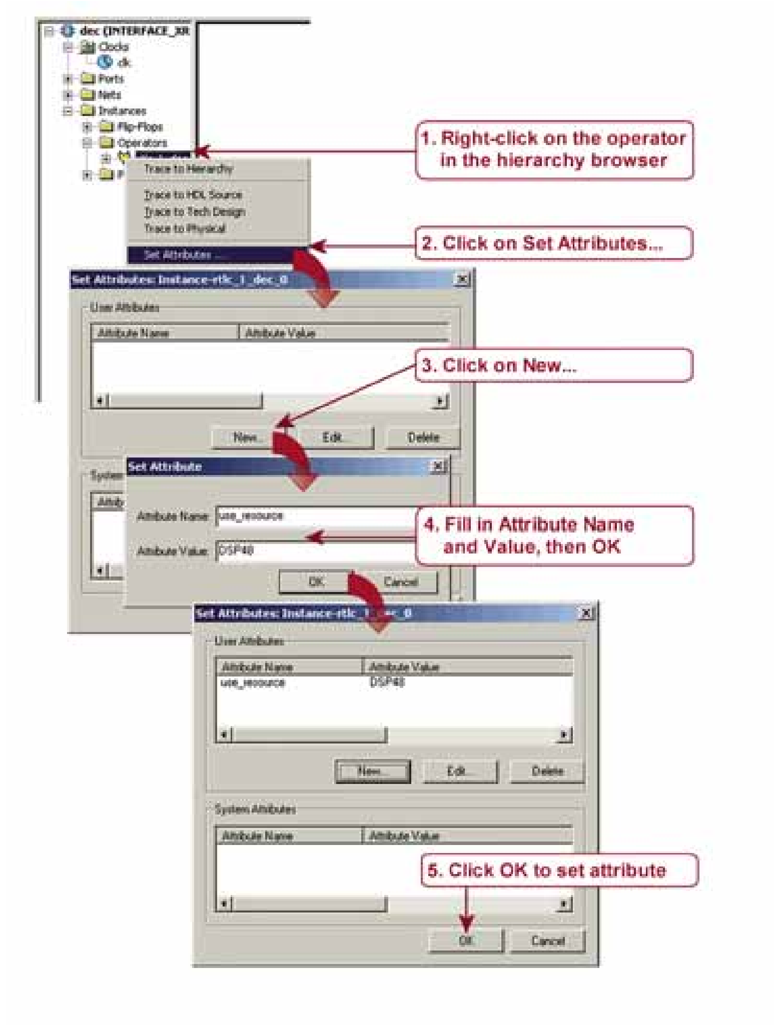


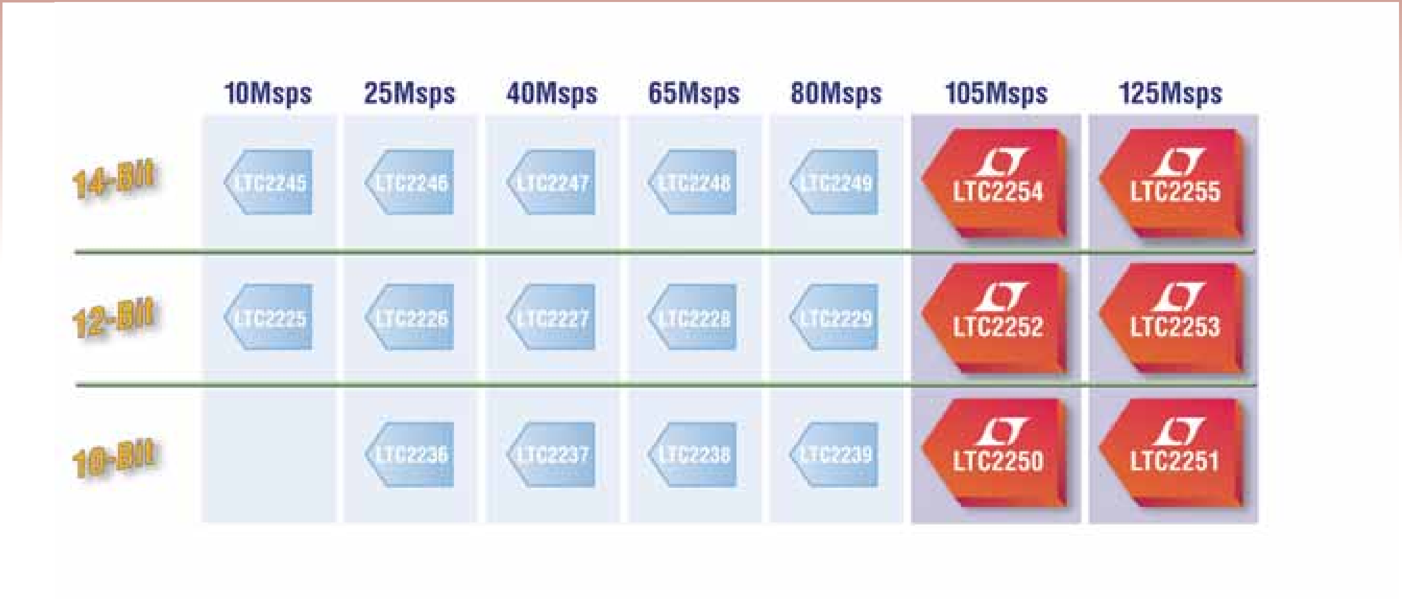
Figure 7 – Setting the use_resource attribute in Precision Synthesis

Conclusion

For MULT-ACC with adder/subtractor and multiple stages of internal input pipeline registers, the Xilinx Virtex-4 architecture has specific advantages over other vendors. For wide multipliers without accumulators, you may find that DSP48 in Virtex-4 devices and other DSP vendors compete more closely in terms of performance. But regardless of your design application, it would be beneficial to test-drive the DSP48 functions in tandem with the Mentor Graphics Precision Synthesis tool for your next design project.

By supporting advanced DSP48 inferencing capabilities, Precision Synthesis makes it possible to easily model DSP RTL behavior at a very high level. It also provides added flexibility with RTL coding styles. The DSP48 inference capability in Precision Synthesis, combined with the advanced DSP48 slice in Virtex-4 FPGAs, provides a powerful system solution if you are considering designing in DSP with today's advanced FPGA architectures.

14-Bit, 125Msps ADC Only 395mW



	10Msps	25Msps	40Msps	65Msps	80Msps	105Msps	125Msps
14-Bit	LTC2245	LTC2246	LTC2247	LTC2248	LTC2249	LTC2254	LTC2255
12-Bit	LTC2225	LTC2226	LTC2227	LTC2228	LTC2229	LTC2252	LTC2253
10-Bit		LTC2236	LTC2237	LTC2238	LTC2239	LTC2250	LTC2251

Lowest Noise, Lowest Power, Smallest Solution Size, Pin-Compatible Family

The LTC®2255 family features excellent AC performance with 72.4dB SNR, 88dB SFDR and extremely low power. At just 395mW, the LTC2255 consumes nearly half the power of the competition, benefiting wireless base stations, imaging systems and portable instrumentation where efficiency and cooling are critical. Part of a pin-compatible family in a small 5mm x 5mm QFN package requiring only a few tiny external components, they offer the smallest solution size available.

Features

- Sample Rate: 125Msps/105Msps
- Single 3V Supply (2.85V to 3.4V)
- Low Power: 395mW/320mW
- 72.4dB SNR
- 88dB SFDR
- Clock Duty Cycle Stabilizer
- Pin-Compatible Family in 5mm x 5mm QFN Package

High Speed ADC Evaluation Board

Info

Nu Horizons Electronics Corp.

Tel: 1-888-747-NUHO

FREE High Speed ADC Evaluation Board, for qualified individuals:

www.nuhorizons.com/Linear

NU HORIZONS ELECTRONICS CORP.

LINEAR TECHNOLOGY

Never Design Another FIFO

The FIFO Generator IP core delivers fully optimized FIFO solutions for any configuration, freeing you to focus on your own design challenges.

by Tom Fischaber
Staff Design Engineer, IP Solutions Division
Xilinx, Inc.
tom.fischaber@xilinx.com

James Ogden
Design Engineer, IP Solutions Division
Xilinx, Inc.
james.ogden@xilinx.com

In digital designs, first-in first-out memory queues (FIFOs) are ubiquitous constructs required for data manipulation and buffering – tasks that are often very challenging. Are all clock domain crossings properly timed and synchronized? How do I convert my 16-bit data path to 64 bits? These elements of a FIFO design are difficult and time-consuming to implement, and are often error-prone. The Xilinx® FIFO Generator core solves these challenges and provides an assortment of complex FIFO designs through a convenient, configurable graphical user interface (GUI), enabling you to focus on your system requirements.

From application notes and reference designs to IP cores, Xilinx has a long history of developing FIFOs. With the introduction of the FIFO Generator, almost any imaginable FIFO configuration is provided as a fully optimized, pre-engineered solution delivered through Xilinx CORE Generator™ software. The FIFO Generator supports a suite of memory types, including block RAM, distributed RAM, shift registers, and the Virtex™-4 built-in FIFO. The core also supports write and read interfaces with either a single common clock or dual independent clocks. These and other options are easily customizable through the GUI. In this article, we'll highlight the benefits of the FIFO Generator solution and how it can help you quickly develop a FIFO that exactly meets your needs.

Common and Independent Clock Domains

The FIFO Generator supports FIFOs both with a single common clock and dual independent clocks for write and read operations. The common clock configuration provides small, fast, low-latency FIFOs supporting a variety of status flags, and is ideally suited for single-clock data-buffering applications.

The independent clock configuration provides even greater utility, solving notoriously difficult and error-prone FIFO designs at the touch of a button. The FIFO Generator handles the synchronization between clock domains, placing no requirements on phase and frequency. Not only does the FIFO Generator core solve the complexities of independent clock designs, but it also provides a variety of additional capabilities (including a full suite of status flags), enabling you to customize the FIFO Generator for your application.

The first page of the FIFO Generator GUI is shown in Figure 1, and highlights how you can configure the FIFO with a single common clock or dual independent clocks using various memory types. Figure 1 also includes the key features supported in each configuration for the FIFO Generator v2.1 release, available for free in ISE™ 7.1i software with IP Update 1.

Virtex-4 Built-In FIFO Support

Included in the Virtex-4 architecture is a built-in FIFO controller with every on-chip block RAM (see Peter Alfke's article, "FIFOs Made Easy," in the First Quarter 2005 issue of the *Xcell Journal*). By utilizing this new built-in FIFO, the FIFO Generator provides easy access to these high-performance independent clock FIFOs, saving valuable FPGA fabric resources while providing substantially lower power consumption.

The FIFO Generator also expands on the capabilities of built-in FIFOs by providing FIFOs of arbitrary width and depth, as well as providing additional status flags. The concatenating of multiple embedded FIFOs and additional logic for status flags are handled automatically, enabling very high-performance designs to be supported with only minimal FPGA resources. A functional diagram of the built-in FIFO design is illustrated in Figure 2.

Non-Symmetric Aspect Ratios/FWFT

Applications requiring data width conversion almost always require two clocks to operate at different frequencies. When the frequencies are related, this task can be easy to implement. However, if the two clock rates have no relationship, this task can be daunting. The FIFO Generator provides automatic width conversion of the stored data, while allowing any relationship between the two clock domains. Figure 3 illustrates how a FIFO with an 8-bit write interface and 2-bit read interface operates, providing a 4:1 write-to-read aspect ratio.

Although FIFOs with different bus widths and independent clocks are complex to design, the FIFO Generator makes this design feature just as simple to create and use as any other FIFO configuration. The solution further combines this functionality with other rich features, including a full suite of status flags and first-word fall-through (FWFT).

FWFT provides the ability to look ahead to the next word available from the FIFO without issuing a read operation. When data is available in the FIFO, the first word falls through the FIFO and appears automatically on the output bus. FWFT is useful in applications that require low latency access to data and in applications that require throttling based on the contents of the data that is read. FWFT support is new in the FIFO Generator v2.1 release.

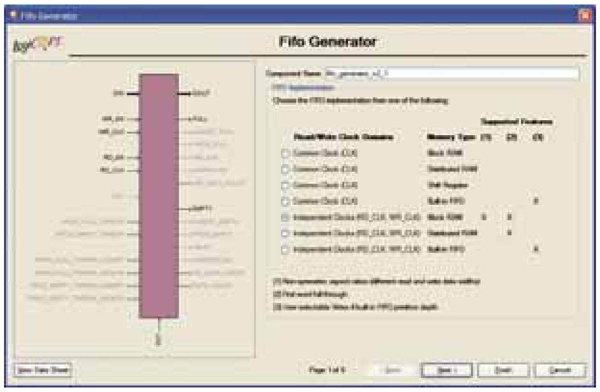


Figure 1 – FIFO Generator GUI

Memory Types

In addition to the built-in FIFO in Virtex-4 devices, the FIFO Generator supports a suite of memory types including block RAM, distributed RAM, and shift registers. The FIFO Generator combines the selected memory type in width and depth, always generating an optimized solution. Table 1 highlights some of the benefits of each memory type.

Conclusion

The Xilinx FIFO Generator core is an all-in-one FIFO solution, providing complex capabilities at the touch of a button. Most traditional FIFO capabilities are already included, as are special features such as built-in FIFO support for Virtex-4 FPGAs, non-symmetric aspect ratios, and FWFT. This core provides peace of mind for system designers – we focus on the FIFO design while you solve your own design challenges. Xilinx will continue to enhance the FIFO Generator for Xilinx FPGA families, as well as adding high-value features that meet your next-generation design requirements.

For more information about the FIFO Generator, visit *www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=FIFO_Generator*. For more information about the Virtex-4 built-in FIFO and its benefits, visit *www.xilinx.com/publications/xcellonline/xcell_52/xc_pdf/xc_v4fifo52.pdf*.

We would love to hear about your experience with the FIFO Generator, or if you have suggestions for new features. To provide feedback about your experience or request new capabilities, e-mail *fifo_generator@xilinx.com*.

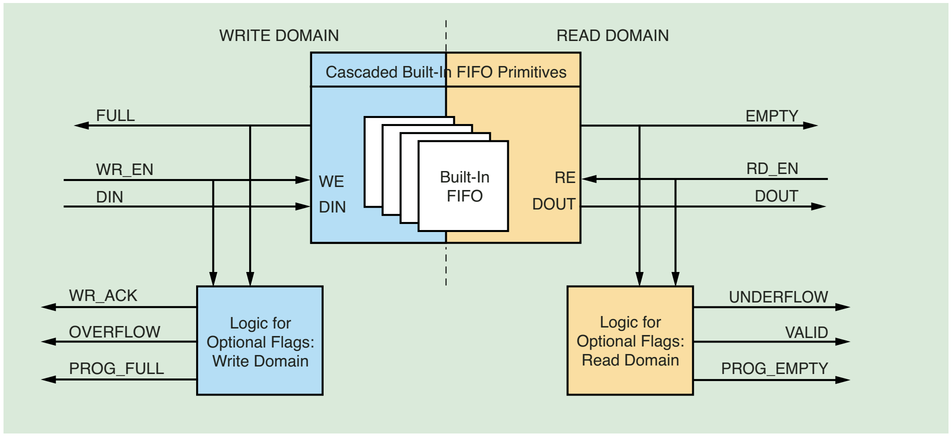


Figure 2 – Built-in FIFO capabilities expanded by the FIFO Generator

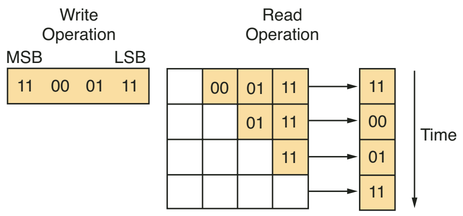


Figure 3 – Non-symmetric aspect ratios (4-to-1 ratio)

	Independent Clocks	Common Clocks	Small Buffering Applications	Medium to Large Buffering Applications	High Performance	Minimum Fabric Resources
Built-In FIFO	●	●		●	●	●
Block RAM	●	●		●	●	●
Shift Register		●	●		●	
Distributed RAM	●		●		●	

Table 1 – Memory type support and benefits

Using a CPLD to Implement a QWERTY Keypad

You can use a Xilinx CPLD to expand a typical handset DTMF keypad into a QWERTY keypad.

by Mike Gulotta
Xilinx FAE
Xilinx, Inc.
mike.gulotta@xilinx.com

As cell phones and other portable handheld devices continue to add features, design trade-offs are constantly evolving. Popular features such as text messaging and web browsing demand more data entry, but that can be cumbersome with traditional dual tone multiple frequency (DTMF) (0-9, #, *) keypads. Using this type of keypad requires multi-tap data entry, which is inefficient and error-prone.

One option to make text entry easier is to use a QWERTY keypad (Figure 1). This type of keypad employs 40 or more keys versus the normal 12 in a DTMF handset, although the additional keys make the handset larger and involve more electronic components.

Text message users may be willing to trade size for a QWERTY keypad; text entry is much easier and you can use two thumbs to enter text messages or data. Recently, some cell phone manufacturers have released handsets with QWERTY keypads that cater to text users.

There are many ways to design data entry keypads, but no standard exists. In this article, we'll examine one possible solution to the design challenge of adding additional keys to a traditional DTMF-type keypad.

QWERTY Building Blocks

Our solution uses a Xilinx® CoolRunner™-II CPLD; its low power, small package options, and low cost make it an ideal candidate for this application.

Going from a DTMF to a QWERTY keypad requires more keys and thus more general-purpose I/O (GPIO). For instance, a DTMF keypad may have only four rows and three columns, whereas a QWERTY keypad may have as many as eight rows and eight columns. But keypad size can vary depending on the requirements of the end system.

Figure 1 – QWERTY keypad (Motorola model A630)

Typically, a processor or DSP is used as an interface to the keypad's rows and columns (see Figure 2). The processor scans the rows and monitors the columns for a logic change. When a change occurs, it indicates that the user pressed one of the buttons. By knowing which row was being scanned and which column changed state, the processor can deduce which specific button was pushed.

Expanding I/O

When designing keypads that require more I/O (as is the case with a QWERTY keypad), you may find that your existing processor does not have enough GPIO. A possible solution is to use a CPLD as an I/O expander to reduce the number of I/Os required from the processor.

Figure 3 introduces a CPLD in between the processor and keypad where one side of the CPLD interfaces to the keypad rows/columns and the other side interfaces to the processor's available GPIO. In this example, an 8 x 8 keypad requires the same number of processor GPIO ports as the 4 x 4 keypad (actually one less) when using a CPLD. Without the CPLD, the processor would require 16 GPIO ports instead of 7.

Scanning and Encoding

In addition to reducing the processor's GPIO requirements, the CPLD can offload some of the processor functions, such as scanning the rows and monitoring the columns for a change in state. When the user presses a key, the CPLD stops scanning and immediately produces an encoded word, which is sent to the processor. The encoded word lets the processor know which key was pressed. This will reduce processor I/O requirements because an encoded word is used to convey which button is pressed to the processor.

In the example shown in Figure 3, six bits are used to represent the encoded word. Six bits provide 2^6 or 64 different values, each representing a different key. However, one value must represent the state when no keys are pressed. So really, only 63 keys can be represented in this example (without adding an additional GPIO).

The processor does not need to scan the keypad because this is being performed by the CPLD; however the processor is still required to monitor for changes on its GPIO – only it would not need to deduce which key was pressed because the information is encoded in a six-bit word.

Switch debounce is also needed. It can be designed in the CPLD or the processor, depending on which device has available resources. Performing this in the processor will keep the size and cost of the CPLD to a minimum.

Summarizing this design example, the CPLD scans the keypad for a pressed key and provides an encoded word to the processor to read and interpret. This function not only offloads the processor from scanning but expands the GPIO.

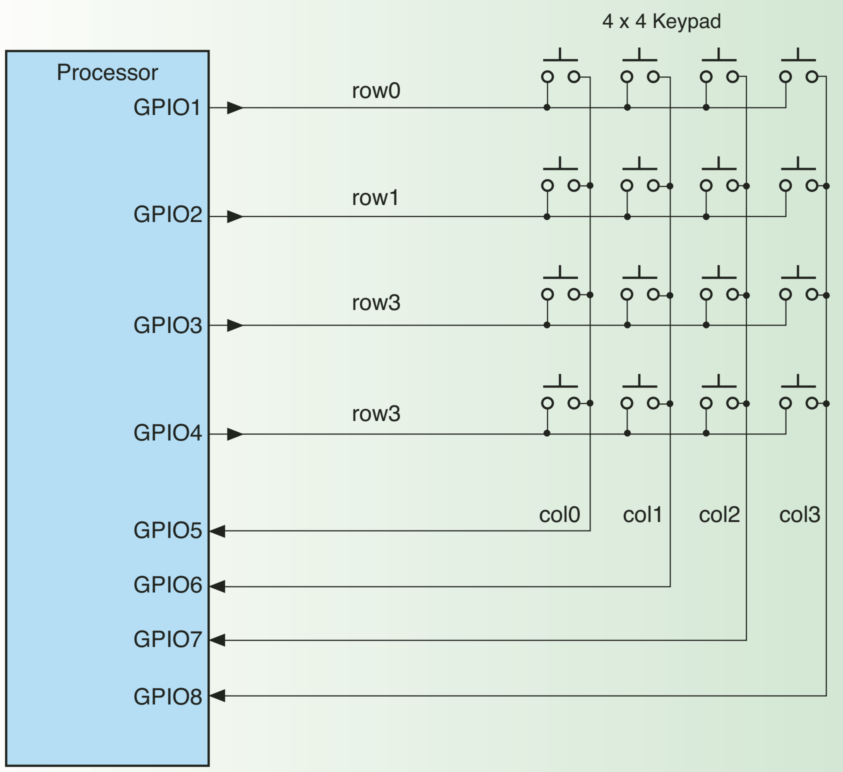


Figure 2 – Simple 4 x 4 keypad connected to a processor requiring eight GPIO

The design fits nicely (~75% utilization) in a CoolRunner-II 32-macrocell device, leaving ~25% headroom for other possible functions. Plus, there are additional design ideas that can reduce power and make use of CoolRunner-II power-saving features.

CPLD Design Details

To scan the keypad rows, a barrel shift register is initialized with all ones, except for a single bit preset to zero. Each bit of the shift register drives an output pin of the CPLD that is connected to a row of the keypad. As the shift register is clocked, the zero bit shifts through the barrel shifter and scans the rows by driving them low one at a time. The keypad columns are inputs to the CPLD and each input is pulled up through an internal pull-up resistor.

When no keys are being pushed, all column inputs to the CPLD are passively pulled up to logic high. All the column inputs are AND'd together and a logic one at the output indicates that no keys are being pressed.

The output of the AND is used as an enable to the shift register. When a key is pressed, the connection between the row and column is made and the column with the key being pressed is driven low by the row associated with that key. The output of the AND will go low and disable the shift register when the key is pressed.

At this point the shift register is driving the row of the key being pressed to a low, and the column of that key is also at a low. To correlate this information, two encoders are used: one for the row bits (outputs of the shift register) and another for the column inputs. The outputs of the two encoders are grouped together to form the encoded word that is sent to the processor. Figure 4 shows a block diagram of the operation.

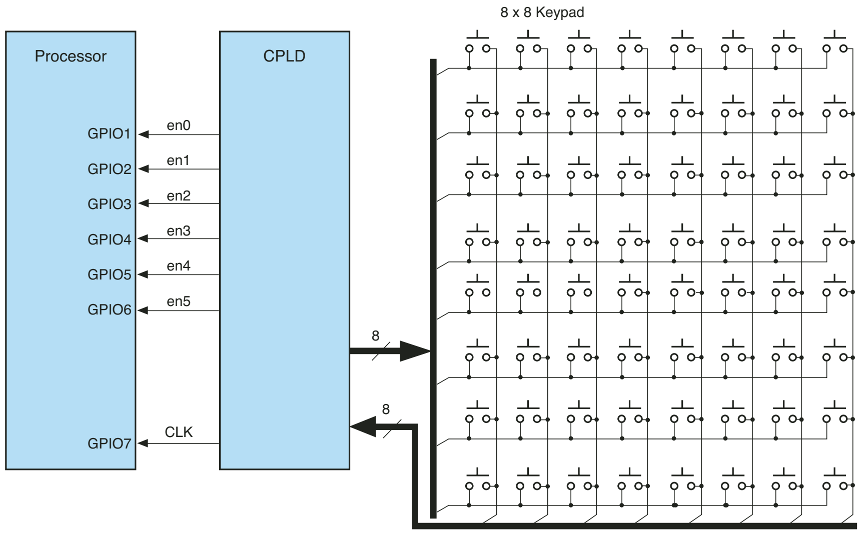


Figure 3 – GPIO expansion with a CoolRunner-II CPLD

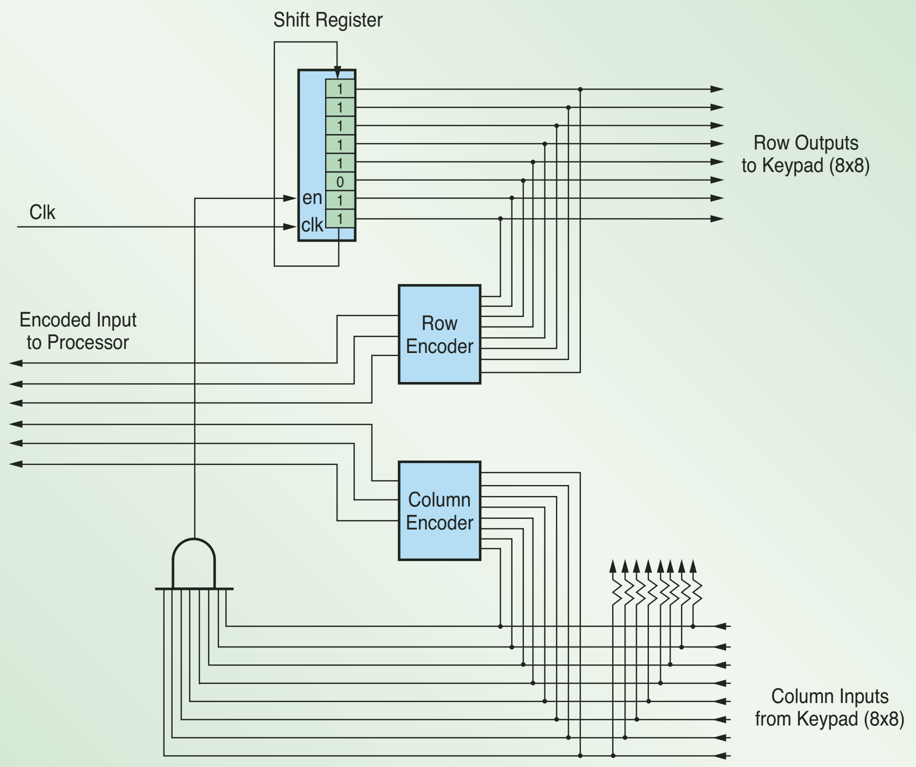

Shift Register
1
1
1
1
1
0
1
1
en
clk
Clk
Row Outputs to Keypad (8x8)
Encoded Input to Processor
Row Encoder
Column Encoder
Column Inputs from Keypad (8x8)

Figure 4 – Block diagram

Conclusion

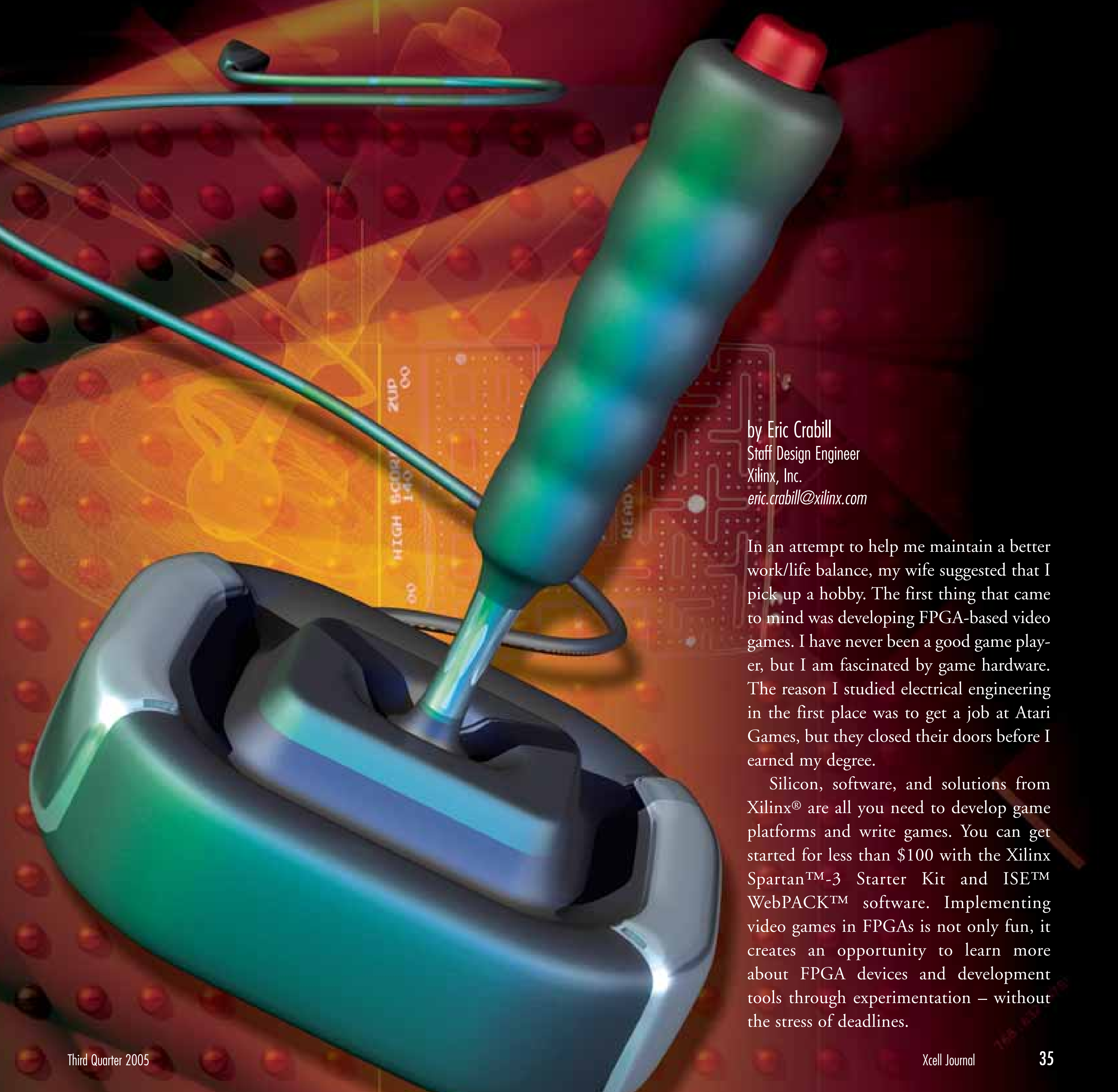
By using Xilinx CoolRunner-II CPLDs, you get design flexibility and low power. Besides I/O expansion, other "glue" functions can be absorbed into the CPLD, such as voltage translation, I/O standards translation, and input hysteresis.

Because CPLDs are programmable, you can use the same device with different keypads and in different products, leading to higher volume quantities and lower cost. The ability to reprogram them (along with simple-to-use design tools) allows for late design changes and lower risk.

To learn more about this application, see the Xilinx application note, "Implementing Keypad Scanners with CoolRunner-II," at *www.xilinx.com/bvdocs/appnotes/xapp512.pdf.* For more information about Xilinx CPLDs, visit *www.xilinx.com/cpld/.*

FPGA-Based Video Games

Implementing video games in FPGAs is fun and educational.



by Eric Crabill
Staff Design Engineer
Xilinx, Inc.
eric.crabill@xilinx.com

In an attempt to help me maintain a better work/life balance, my wife suggested that I pick up a hobby. The first thing that came to mind was developing FPGA-based video games. I have never been a good game player, but I am fascinated by game hardware. The reason I studied electrical engineering in the first place was to get a job at Atari Games, but they closed their doors before I earned my degree.

Silicon, software, and solutions from Xilinx® are all you need to develop game platforms and write games. You can get started for less than $100 with the Xilinx Spartan™-3 Starter Kit and ISE™ WebPACK™ software. Implementing video games in FPGAs is not only fun, it creates an opportunity to learn more about FPGA devices and development tools through experimentation – without the stress of deadlines.

FPGA Gaming Platforms

Simple gaming platforms only require three components: a human interface device for input, another for output, and a mechanism for implementing the game logic or game program. Many commercial development boards with Xilinx FPGAs satisfy these requirements.

However, to create something that I could truly call my own, I chose to design a platform that was:

- Entirely Xilinx-based
- Inexpensive to build
- Suitable for cooperative development
- Capable of respectable performance

After a number of iterations, I created a block diagram (Figure 1) for a simple and yet highly flexible reconfigurable video game platform that met my requirements.

The platform is designed to support the largest Spartan-3 device available in the free ISE WebPACK tools – the XC3S1000; this device is used to implement the game logic or game program. The platform also uses the Xilinx System ACE™ CF configuration controller to great advantage.

At power on, the SystemACE CF controller configures the XC3S1000 from one of eight user-selectable configurations on the Compact Flash card. After the initial power-on configuration loads, the XC3S1000 can instruct the controller to reconfigure using any of the other configurations. Furthermore, the design in the XC3S1000 may access the Compact Flash card through the System ACE CF controller as a disk. You can use the extra space on the Compact Flash card for program and data storage.

In terms of human interface devices, there are two joystick ports, a stereo audio output jack, and a VGA port capable of 4,096 colors. The platform also provides a standard serial port to use for linking systems, textual interaction through a terminal, or debugging. Schematics and photoplots for this design are available at the FPGA Games website (see the resources list at the end of the article).

A completed prototype of the platform (Figure 2) is quite small at 4 x 6 inches and could easily be smaller, possibly fashioned into a handheld unit with an integrated LCD screen. The Compact Flash card becomes, in effect, the game cartridge. The fun doesn't start, though, until you have some games to play!

Games Via Emulation

Emulation is the sincerest form of flattery. With a suitable platform, you can emulate a wide variety of games and game systems, including those found exclusively in arcades. There are two general methods: software emulation and hardware emulation.

In either case, emulation may involve the use of ROM images, which contain copyrighted software and audiovisual content. Make sure that you obtain legal ROM images for emulation – either by purchasing original ROMs or by obtaining the legal rights to use downloaded ROM images from services such as StarROMS. Don't let flattery turn into thievery.

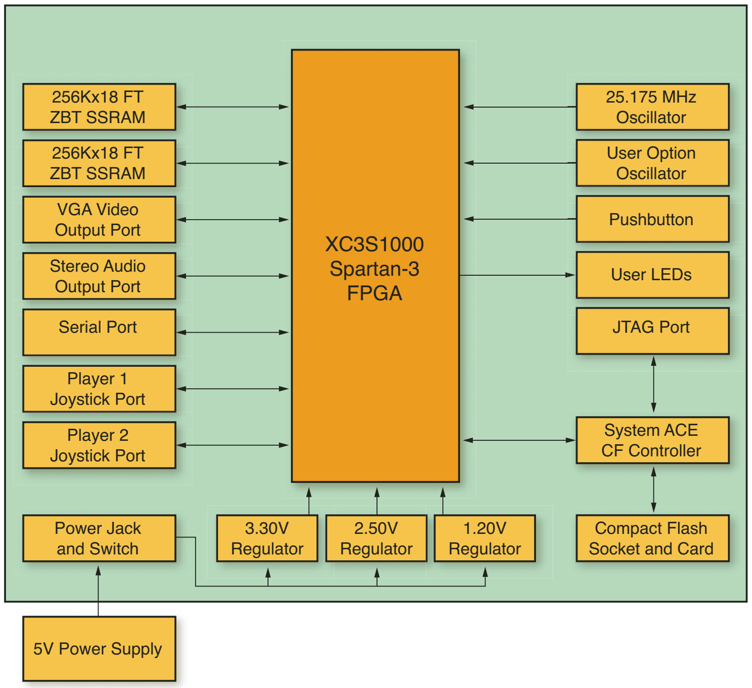


Figure 1 – Block diagram of a reconfigurable video game platform

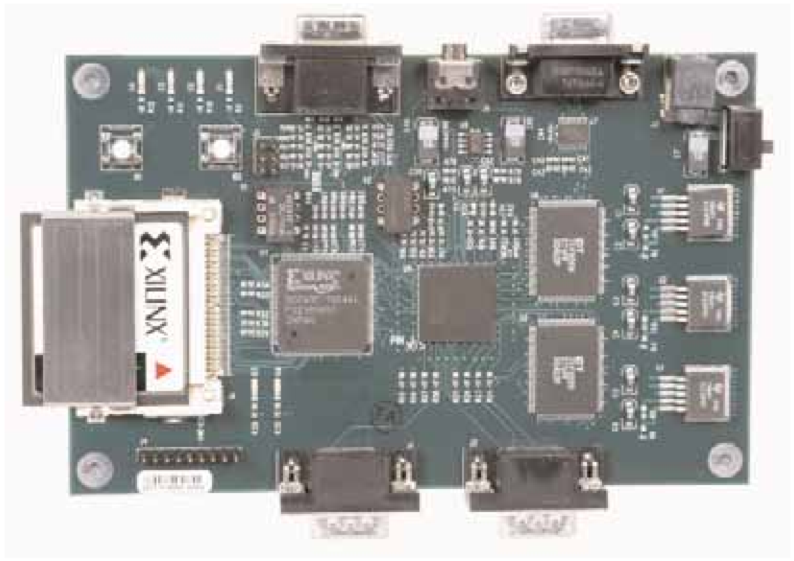


Figure 2 – Prototype of a reconfigurable video game platform

The Atari 2600 Video Computer System, originally released in 1977, makes a wonderful case study of emulation. Although the system is nearly 30 years old, only recently have people been able to emulate it well.

Software Emulation

Software emulation involves the creation of a program that runs on one system (the host) to simulate the behavior of another (the target). For game systems, the emulation of the target must include human interface devices as well as the game logic, which is typically a microprocessor running a game program.

In most cases, software emulation requires a host with performance at least an order of magnitude greater than the target. This is because it may take a fair number of instructions (or cycles) on the host to emulate what happens in one instruction (or cycle) of the target.

Software emulation of game systems is entirely feasible if you use Xilinx Platform FPGA devices such as the Virtex™-4 FX family with integrated PowerPC™ processors. For example, you can emulate the Atari 2600 Video Computer System on a PowerPC host created with the Xilinx Embedded Development Kit. If you are coding-challenged like me, porting an existing emulator is quicker than writing one from scratch.

For each emulated cycle of the target system, the emulator must exactly match the target behavior. However, most emulators do not run with cycle-for-cycle accuracy in real time. One technique is to periodically emulate behaviors faster than real time, and then halt until the next period begins.

Hardware Emulation

Hardware emulation involves re-creating the original hardware – or something functionally equivalent – in programmable logic. This approach may yield the most authentic emulation, but requires detailed knowledge of the emulated system, sometimes down to the transistor level.

Again, consider the Atari 2600 Video Computer System. It is based on a 6507 processor (a derivative of the venerable 6502), the 6532 peripheral (RAM, I/O, and a timer), and a ROM (4 KB). The processor is available from the Open Cores website, and the rest are easy to create.

An additional custom peripheral called the Television Interface Adapter (TIA) implements audio and video output and user inputs. The TIA was designed in NMOS, and while its schematics are available, the TIA is surprisingly complex for a few thousand gates. The TIA design uses dynamic logic, latches, all manner of asynchronous resets and presets, gated clocks, and circuits cascaded using ripple techniques. To further complicate the situation, game programmers learned how to exploit undocumented "features" in TIA hardware. As a result, any hardware implementation must truly reproduce the original behavior in real time, cycle for cycle.

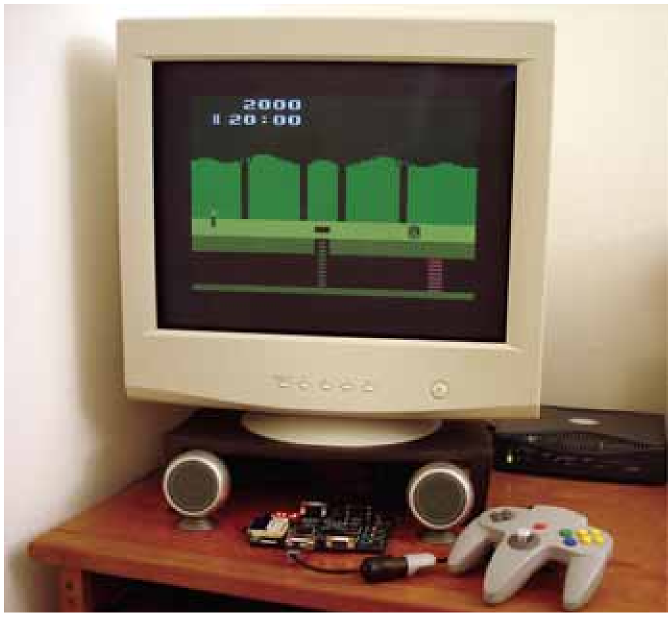


Figure 3 – Hardware emulation of the Atari 2600 VCS running Pitfall by Activision (courtesy of Ed Henciak)

Ed Henciak, an expert Xilinx user from Pennsylvania, had the understanding, skills, and patience to implement a hardware emulation of the Atari 2600 Video Computer System in an FPGA. I was so amazed with his results that I sent him a prototype unit of my game platform. He sent me back some exciting pictures of his design running on it (Figure 3). In the near future, Ed plans to release his design with a low-cost FPGA-based game platform of his own.

Unleash Your Imagination

If you aren't nostalgic for the video game systems of yesteryear, unleash your imagination and implement a game you enjoy from the ground up. I find wonderful ideas by scouring the websites of hobbyists working with discrete embedded processors.

Some of the best games are simple. You can start small with an addictive puzzle or action game (for example, Tetris, Minesweeper, Pong, or Breakout), implementing it directly in logic or with a Xilinx PicoBlaze™ processor.

Once you are comfortable with games like this, you can graduate to more complex ones running on Xilinx MicroBlaze™ or PowerPC processors. The Xilinx Embedded Development Kit is a great tool for this activity, as it provides a wealth of peripherals to choose from and complete development environments for both processors.

Conclusion

You can develop games on a wide variety of Xilinx FPGA development boards. It is a fun and rewarding activity. If you are interested in reading more, the following websites provide great information, motivation, and example projects:

- *www.fpgaarcade.com*
- *http://office-dsan.hp.infoseek.co.jp (Japanese)*
- *http://members.iinet.net.au/~msmcdoug/pace*
- *www.fpga-games.com*
- *www.retrogames.com*
- *www.starroms.com*
- *www.opencores.org.*

High Performance Doesn't Have to Mean High Costs

ISE 7 design tools deliver the technology to get the most out of your silicon while slashing project costs.

by Lee Hansen
Sr. Product Marketing Manager
Xilinx, Inc.
lee.hansen@xilinx.com

Every customer I know is feeling the pressure to cut design costs. Project budgets continue to shrink, while the pressure to be first to market forces shorter design cycles and fewer engineers per project. But did you know that there's a wealth of technology meant to help you do just that – shorten your design cycle, solve your design bottlenecks, and lower your overall design costs? These features are already built into Xilinx® Integrated Software Environment (ISE™) software.

Higher Performance – Faster Project Completion

The measured 70% performance advantage of ISE design tools (versus competing PLD tools) applies beyond bleeding-edge, high-performance digital projects. Lower speed projects also benefit from this performance advantage by allowing you to hit your performance targets early in the design cycle. You spend less time tweaking and iterating through the implementation phase.

ISE place and route tools also help you ensure efficient implementation. The place and route tools and reports can offer interactive suggestions about how you can change your HDL code. These suggestions help make more efficient use of FPGA resources and can save overall design space.

The architecture wizards inside ISE design tools are a collection of graphical-based menus and dialog boxes with which you can quickly and easily set the parameters of advanced silicon features.

More high-performance technology is packed directly into ISE design tools than any other PLD design offering, including core capabilities like timing-driven mapping, global optimization, design re-timing, and FPGA physical synthesis. Together, this ProActive Timing Closure technology leads to higher overall design performance, and more technology focused on helping you achieve timing closure.

The timing-driven map option helps deliver better utilization to your target Xilinx FPGA device, particularly if the device is already more than 90% utilized (the point at which most PLD users have to consider moving up to the next higher density and more costly device). Timing-driven map combines placement with logic slice packing to improve placement quality for "unrelated logic."

Using timing-driven map offers you the potential to stay in your chosen device – even if utilization is pushing 90% or higher – when competing tools would have forced the design into a larger and more expensive device.

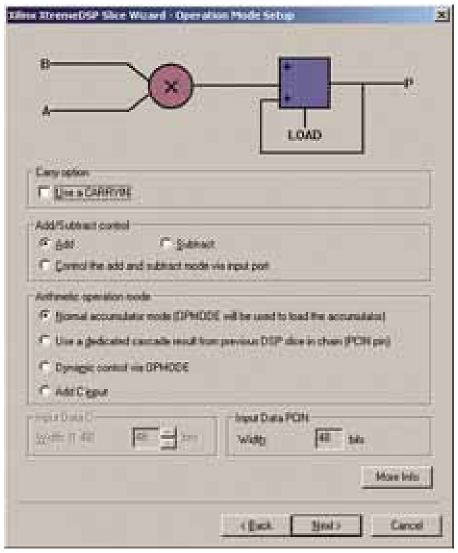
Cutting Through Design-Flow Bottlenecks

ISE design tools focus on solving the problems that plague traditional PLD designs and deliver new tools and technology to speed you through the design flow faster, saving time and money. The ISE 7.1i version includes a host of new tools – including Technology Viewer, message filtering, design summary, ISE Simulator, and ModelSim Xilinx Edition-III – all focused on "ease-of-design" to get you through to project completion faster.

New ISE 7.1i Tools and Technologies

The Technology Viewer lets you view your post-synthesis HDL-based design at the block level in a schematic-like display. It is built on the same graphic interface as RTL Viewer, so there are no new commands or menus to learn. Full hierarchy is represented, so you can easily push down or pop up through your design, highlighting and identifying critical elements by pin, net, or instance name. The Technology Viewer – combined with the RTL Viewer, ISE Floorplanner, and FPGA Editor – bring you more ways to visualize your design and to see and control exactly how the implementation phase is completing your design, helping avoid time-consuming problems and re-implementation steps.

ISE 7.1i software includes a new design summary view that takes the most commonly sought-after design information



Carry option
- [ ] Use a CARRYIN

Add/Subtract control
- [✓ Add]
- Subtract
- Control the add and subtract mode via input port

Arithmetic operation mode
- [✓ Normal accumulator mode (OPMODE will be used to load the accumulator)]
- Use a dedicated cascade result from previous DSP slice in chain (PCIN pin)
- Dynamic control via OPMODE
- Add C input

Input Data C: Width (1-48) 48 bits

Input Data PCIN: Width 48 bits

Figure 1 – XtremeDSP architecture wizard

and places it in one easy-to-use automated display, eliminating the need to search through multiple tool reports and outputs to find exactly what you need. The design summary also contains a list of hyperlinked detailed implementation reports. You can easily jump to more detailed information. The design summary saves design time by delivering the core up-to-date design information.

The new message filtering capability lets you select the report information that you deem non-critical to your design and suppress it from future reports. Message filtering delivers more streamlined and pertinent report information, and allows you to quickly see the data necessary to your project, making debug and verification quicker and easier.

ISE and ChipScope™ Pro 7.1i tools offer new remote programming and remote debug capabilities. ChipScope Pro and iMPACT programming tools can now run in server/client mode over a TCP/IP connection. You can sit in your office while debugging or programming a board next door in the lab or on the other side of the world. You can share a single board or debug system in the lab with other engineers on your team, or allow help desk personnel to debug a problem remotely at a customer site. Remote programming and debug save you the cost of additional software, and make more efficient use your existing project workstation setup.

Existing Features that Drive Design

The architecture wizards inside ISE design tools are a collection of graphical-based menus and dialog boxes with which you can quickly and easily set the parameters of advanced silicon features. For example, the XtremeDSP™ slice wizard, shown in Figure 1, provides control over the Virtex™-4 XtremeDSP silicon slice technology. This new silicon capability lets you build high-performance DSP filters and custom pre- or post-co-processing DSP algorithms.

The XtremeDSP slice wizard lets you specify accumulator, adder/subtractor, multiplier, or multiplier and adder/accumulator DSP modes. You can graphically set input and output bus data widths, pipelining options, clock enable, and reset pin setups, and then review parameters and output the results as HDL-ready code. Once configured, the architecture wizard writes editable VHDL or Verilog source code that is instantiated directly into your target project. The architecture wizards help reduce the learning curve associated with new silicon releases, and allow beginning FPGA designers to quickly get up to speed programming even the most advanced silicon.

The pin and area constraints editor (PACE) delivered within ISE design tools includes graphical pin and area management that is both powerful and easy to use, as shown in Figure 2. You can drag-and-drop pin assignments onto a graphical map of the device, either by footprint or architectural area. You can group pins logically and by color for easy recognition, specify I/O standards and banks, prohibit I/O locations, and verify legal pin assignments on the fly.

PACE can also interface through CSV files, letting the FPGA engineer send pin definitions directly to PCB layout, or read PCB layout information and back-annotate that information to the FPGA design. PACE can be the starting point of your project, and PACE writes out HDL language template files based on the hierarchy and logic area groups you've defined. PACE includes several design rule checks including simultaneous switched output (SSO), which help predict ground bounce problems and account for exact pin delay across your entire design. PACE delivers a wealth of productivity that can help reduce design headaches and lower project costs.

Incremental Design – Shorter Re-Implementation Times

Incremental design, first introduced in ISE 5.1i software nearly three years ago, can slash re-implementation time by as much as 75%. Your design is first floorplanned in PACE or our optional PlanAhead™ software. The design is then completed through the normal implementation cycle. If subsequent modifications are required, incremental design re-implements only the area(s) affected by the design change, leaving the other completed design areas intact and dramatically shortening the design cycle. Incremental design can deliver more design cycles when you need them and shorten your design time.

An Array of Verification Options

Verification is one of the most time-consuming and time-critical phases of the design flow. Incomplete or time-consuming verification strategies can take up more than half of the overall design flow and leave critical logic areas unpredictable. ISE delivers enhancements to the verification flow that help cut the time and cost of verification.

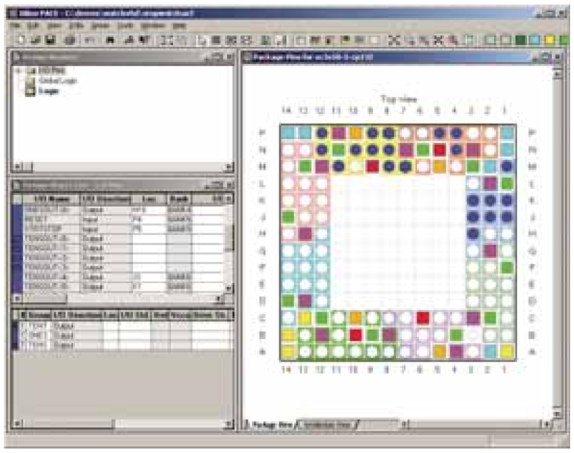


Figure 2 – PACE – pin and area constraints editor

Two HDL Simulation Choices

The ISE Simulator is a new, full-featured HDL simulator delivered with and integrated directly into ISE design tools. It offers the ability to simulate directly from the ISE Project Navigator process window, where test benches, stimulus, and output graphics are generated. ISE Simulator supports VHDL and Verilog, functional and timing simulation, and is licensed through the fast and painless Xilinx software registration ID process – no licensing dongles or Ethernet keys.

ISE 7 software is also the release of the new ModelSim Xilinx Edition-III HDL Simulator. Free to all ISE customers, MXE-III Starter offers 50% faster HDL simulation and 20 times more design capacity than the previous version. For large-density FPGA designs, you can purchase the MXE-III full version, which provides five times the design capacity and 30% faster performance than the MXE-III starter.

ISE design tools preserve design hierarchy throughout the entire design flow, while other solutions are forced to flatten design hierarchy during implementation and verification and then reconstruct that hierarchy for debug. By preserving hierarchy, Xilinx and partner verification tools are able to compile and run your design faster. And because signal names, components, and design instances are preserved throughout the flow, debug is more accurate and cross-probing is easier.

Unrivaled Real-Time Verification

ISE design tools also link directly to our optional, separately purchased ChipScope Pro real-time debug environment. The ChipScope Pro tools insert low-profile logic analyzer, bus analyzer, and virtual I/O software cores during design capture. These cores are then synthesized and implemented into your silicon, allowing you to view:

- Any internal signal within the FPGA
- Embedded processor signals, including the IBM CoreConnect processor local bus or on-chip peripheral bus supporting the IBM PowerPC™ 405 inside Virtex-4 FX devices
- Embedded processor signals for the MicroBlaze™ soft-processor core

Signals are captured at or near operating system speed and brought out through the programming interface, freeing up pins for your design, not debug. You can then analyze captured signals through the ChipScope Pro software logic analyzer. ChipScope Pro software can literally slash as much as 50% off traditional ASIC and structured-ASIC verification flows.

Conclusion

The advanced technology built into ISE 7 software can cut your design and verification times, slash project costs, and offer potentially lower device savings in the long run. All ISE configurations, ChipScope Pro analyzer, PlanAhead software, MXE-III, and ISE Simulator are available for purchase from the Xilinx online store, Xilinx distributors, or by calling (800) 888-FPGA (3742).

Smart Telematics Systems from Xilinx and Microsoft Corp.'s Automotive Business Unit

Spartan-3 FPGAs are chosen for their design flexibility and performance.

by Rodney Stewart
System Architect, Automotive
Xilinx, Inc.
rodney.stewart@xilinx.com

David Vornholt
Strategic Relationships Manager
Xilinx, Inc.
david.vornholt@xilinx.com

According to a U.S. Department of Transportation study, people worldwide spend more than 500 million commuter hours per week in automobiles. With so much time behind the wheel, people are looking for ways to stay entertained, talk to loved ones, and perhaps even complete some tasks that they would normally complete in the workplace.

Staying connected while in the automobile is paramount on the list – just look at cell phone usage. Also, encountering heavy traffic along the way, not taking the right route, or something as mundane as running out of fuel can affect punctuality.

How can drivers stay connected while driving safely and make it to their destination on time? The smart way is to have communication and control activated through voice command in combination with a connection to the Internet. This is delivered in the Microsoft Telematics Platform, a hub for the seamless integration of various mobile devices and the delivery of information through the Internet and wireless networks.

The Microsoft Telematics Platform offers:

- Advanced high-quality speech recognition and synthesis technology
- On-demand web services such as traffic jam avoidance, accessing current headlines, or finding the closest gas station with the lowest prices through MSN Autos (currently only in the U.S.)
- Customized navigation – points of interest or turn-by-turn directions with the help of GPS
- PDA/cell phone integration with Bluetooth technology, which wirelessly connects cell phones and PDAs to the vehicle's electronics system, allowing drivers to use their voice to make and receive calls, get meeting reminders, and access important data through the car's audio system
- Remote diagnostics to check on the "health" of the vehicle, including problem and maintenance alerts, potentially improving engine performance over the life of the car

Microsoft Corp.'s Automotive Business Unit and Xilinx® have worked together to create a reference platform that delivers these benefits with a low cost point to catalyze the development of simpler, more reliable, and affordable solutions to drivers around the world.

A Flexible and Scalable Platform

The traditional automotive electronic design approach has been to develop very specific, tailored, and rigid solutions based on the needs of automotive manufacturers. Telematics and infotainment are forcing the automotive industry to rethink the products and systems designed into a typical "connected car."

The convergence of the consumer world into the vehicle – in applications such has telematics – has forced "consumer development" thinking into an industry that is traditionally slow, conservative, and cost driven. New requirements carried across from the consumer industry demand rapid change, as consumers always expect to have the next big thing.

This demand is forcing the need for flexible architectures and changes to design methodology that can cope with not only current applications but future and possibly unknown features. This conflicts with the multi-year development and validation cycles that typical automotive electronic designs generally require. It is now essential that a platform developed today (for a vehicle to be released in two to three years) has sufficient system resources to cope with unexpected changes both throughout the product development cycle and after introduction.

As with any platform, flexibility and scalability are key to the successful adoption of the architecture, from basic systems through to high-performance, high-end telematics systems. With this in mind, Microsoft has developed a true automotive standard telematics platform that is customizable and scalable.

The platform incorporates an ARM 9-based microcontroller, supports memory from 32 MB flash/32 MB DRAM upwards, and includes integrated GPS Bluetooth and a GSM phone module. External vehicle connections include a CAN network interface as well as protected analog and digital I/O for functions such as LED drivers and button inputs. The basic architecture of the platform is shown in Figure 1.

Microsoft took advantage of the flexibility and high integration possibilities of FPGA technology. A Spartan™-3 XC3S400 FPGA was used in this platform for multiple independent purposes such as a GSM phone interface, vehicle interfaces (CAN controller and K-line), and sophisticated audio signal conditioning and routing (shown in Figure 2).

The high levels of integration that FPGAs offer also have the advantage of containing multiple buses, interfaces, and clocks within one device, making design with EMI more manageable. In addition, reducing component count and board space leads to lower production costs and a higher quality of manufacture – important factors in any automotive design.

Understanding the nature of vehicle development and the multitude of vehicle

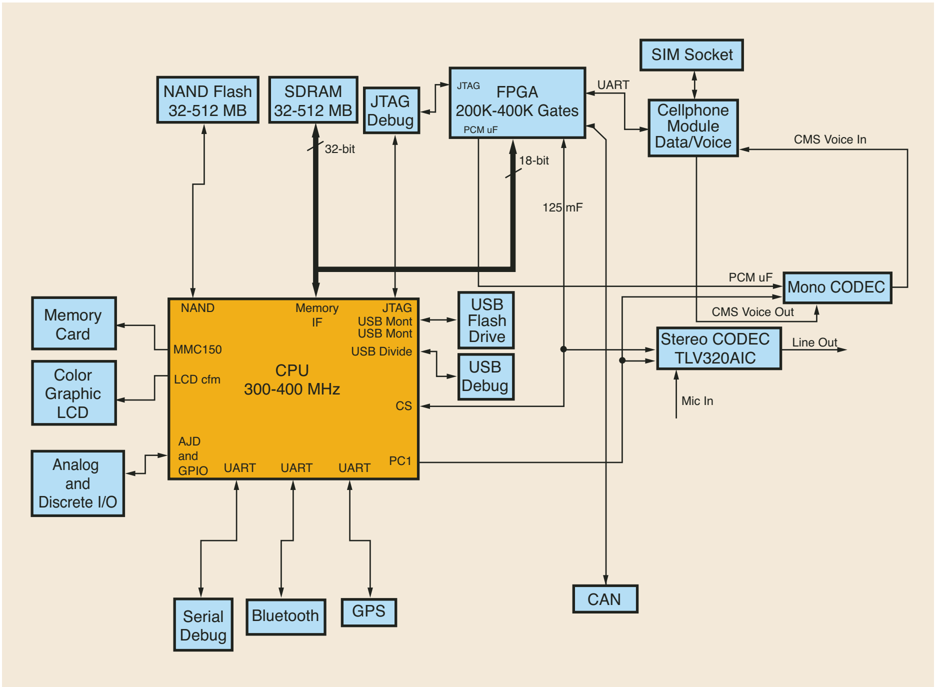


Figure 1 – Microsoft telematics platform hardware architecture

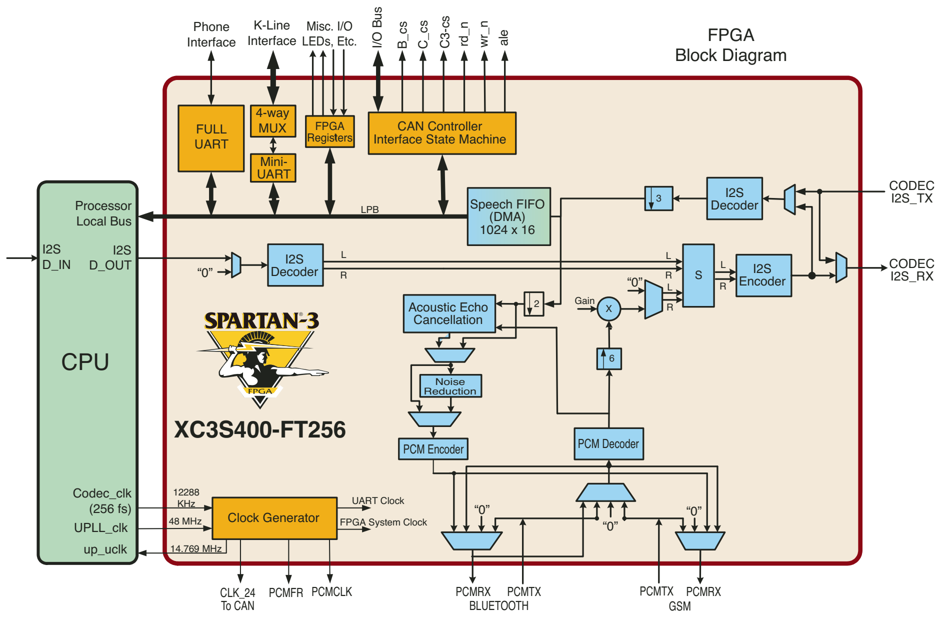


Figure 2 – Xilinx Spartan-3 FPGA design

interfaces available, Microsoft intentionally designed a flexible solution that allows rapid changes to the back-end vehicle interface without affecting the underlying architecture and performance of the system. For example, in the future it would be possible to adapt the FPGA solution to suit the needs of the end application with automotive buses such as MOST, IDB-1394, or another digital vehicle network.

Voice Recognition System

Central to the Microsoft Telematics Platform is the voice recognition (VR) system. The audio signal path within any VR system is analog biasing/filtering, digitization, and digital filtering before the signal is finally presented to the VR engine for speech processing.

Within this path, multiple opportunities exist for unwanted noise to be introduced into the system (both onboard the electrical platform and within the vehicle environment even before the electronics). Both the product developer and the vehicle manufacturer must ensure that the microphone position and type are correctly suited to the application and environment.

In a perfect world, the VR engine will receive clean, consistent speech signals – but given the dynamic nature of the vehicle environment, acceptable voice recognition implementation is not a straightforward exercise. Factors such as vehicle speed, window position (open/closed), road noise, and weather conditions (rain/wind) only add to already difficult VR problems such as languages, accents, and gender. These added factors have increased the importance of preconditioning using highly adaptive digital filtering algorithms before the signal is presented to the VR engine.

Microsoft chose to implement this signal preconditioning in hardware and take advantage of Xilinx parallel DSP processing. Spartan-3 FPGAs, with as many as 104 embedded 18-bit multipliers, are ideal for implementing compact DSP structures such as MAC engines, distributed arithmetic FIR filters, and fully parallel FIR filters in a low-cost device.

Microsoft also offloaded processor-intensive software filtering into hardware. Of course, this pre-processing is possible in ASSPs such as dedicated DSP chips. But the benefits gained through high levels of integration in other parts of the platform would be lost.

The combination of telematics and VR allows implementations of adaptable and upgradeable VR engines and DSP filters tailored to suit certain types of users and environments (Language: English, Accent: Scottish, Gender: Female).

The importance of designing automotive products (especially in the infotainment section of the vehicle) with sufficient spare bandwidth to cope with new and unexpected future upgrades also applies to the FPGA. It is now becoming clear to automotive OEMs that architectures that allow for flexible and scalable firmware are a necessity in future platforms.

Although not currently implemented in the Microsoft platform, it would be possible to easily add soft processors to act as system co-processors. Just as the DSP processing was offloaded from the main processor in Microsoft's design, it would also be possible to use embedded processors (such as the Xilinx 32-bit MicroBlaze™ soft processor or 8-bit PicoBlaze™ microcontrollers) to take some of the processing load from the main system processor.

FPGAs for Automotive Applications

In-car electronics have seen tremendous growth in recent years, not only in traditional body control and engine management but in the new areas of driver assistance systems and telematics applications. Figures recently published by the IEEE indicated an annual increase in car electronics of 16%, with a prediction that by 2005 electronics will account for 25% of the cost of a mid-size car.

Telematics systems exhibit characteristics more like those of consumer products – short time to market, short time in market, and changing standards and protocols. These issues impact the way engineers approach designs and select the hardware needed to quickly create, iterate, and support future upgrading.

FPGA technology can now solve these requirements. Xilinx is committed to serving telematics and car infotainment applications through its Xilinx Automotive (XA) family, which delivers:

- Extended temperature ranges – up to 125°C
- Full production part approval process (PPAP) support
- Industry-recognized AEC-Q100 device-qualification flow
- Compliance with the worldwide automotive quality standard ISO TS 16949, as well as Pb-free packaging to meet the RoHS directive

These devices, based on our Spartan family of FPGAs, are ideal for digital designs requiring low cost per logic cell (system gate), low cost per I/O, and advanced features such as multiple I/O standards on a singe device and embedded multipliers for high-speed DSP.

Conclusion

Backed by a commitment from supporters such as the Microsoft Automotive Business Unit and Xilinx Automotive, the vision of Microsoft's Telematics Platform is now becoming a reality. The convergence of key technologies is being adopted today by first-tier automobile manufacturers in a platform that enables:

- A valuable and affordable telematics solution
- Reliable connectivity through wireless networks
- High-quality voice recognition
- A broadly supported operating system for application developers
- Low-cost hardware

This is giving rise to a "virtuous cycle" of continuous investment by developers, who will use these platforms to create even more value for end users.

For more information, visit *www.microsoft.com/automotive/windowsautomotive/about.mspx/* and *www.xilinx.com/automotive/*.

A Low-Cost Programmable PCI Express Solution

Our solution comprises a Philips PX1011A-EL1 device and a Spartan-3/E FPGA containing the PCI Express Endpoint core.

by David "Andrew" Brierley-Green
Senior Principal Engineer
Philips Semiconductors
david.brierley-green@philips.com

Ho Wai Wong-Lam
Program Manager
Philips Semiconductors
ho.wai.wong-lam@philips.com

PCI Express is a new interconnect standard that provides a serial replacement for the PCI, AGP, and PCI-X buses, which are commonly used in computer and embedded systems. The market has embraced the adoption of PCI Express in computers, and a wide variety of PCI Express plug-in cards and ExpressCards are now available.

Many plug-in cards and embedded systems that use FPGAs to implement PCI and PCI-X are expected to migrate to PCI Express in the coming years. Higher throughput, lower printed circuit board complexity, and the unification of various interconnectivity standards into one PCI Express standard are all factors that entice more developers to adopt PCI Express technology.

A programmable solution offers flexibility, short time to market, and low upfront costs, and is ideal for emerging and low- to mid-volume applications. Philips Semiconductors offers the PX1011A-EL1 x1 PCI Express PHY to form a low-cost programmable PCI Express solution with Xilinx® Spartan-™3/E FPGAs containing the Xilinx PCI Express Endpoint core. This combination achieves a price point that is a fraction of previously available programmable solutions for PCI Express. It enables designers of high-volume applications to take advantage of a programmable and compliant PCI Express solution.

PCI Express and Its Physical Layer

Like all modern networking and connectivity standards, PCI Express uses a layered protocol model. Data is transferred at 2.5 Gbps over a single PCI Express lane; this configuration is referred to as an x1 link. PCI Express is scalable in that you can bundle multiple lanes together. For example,

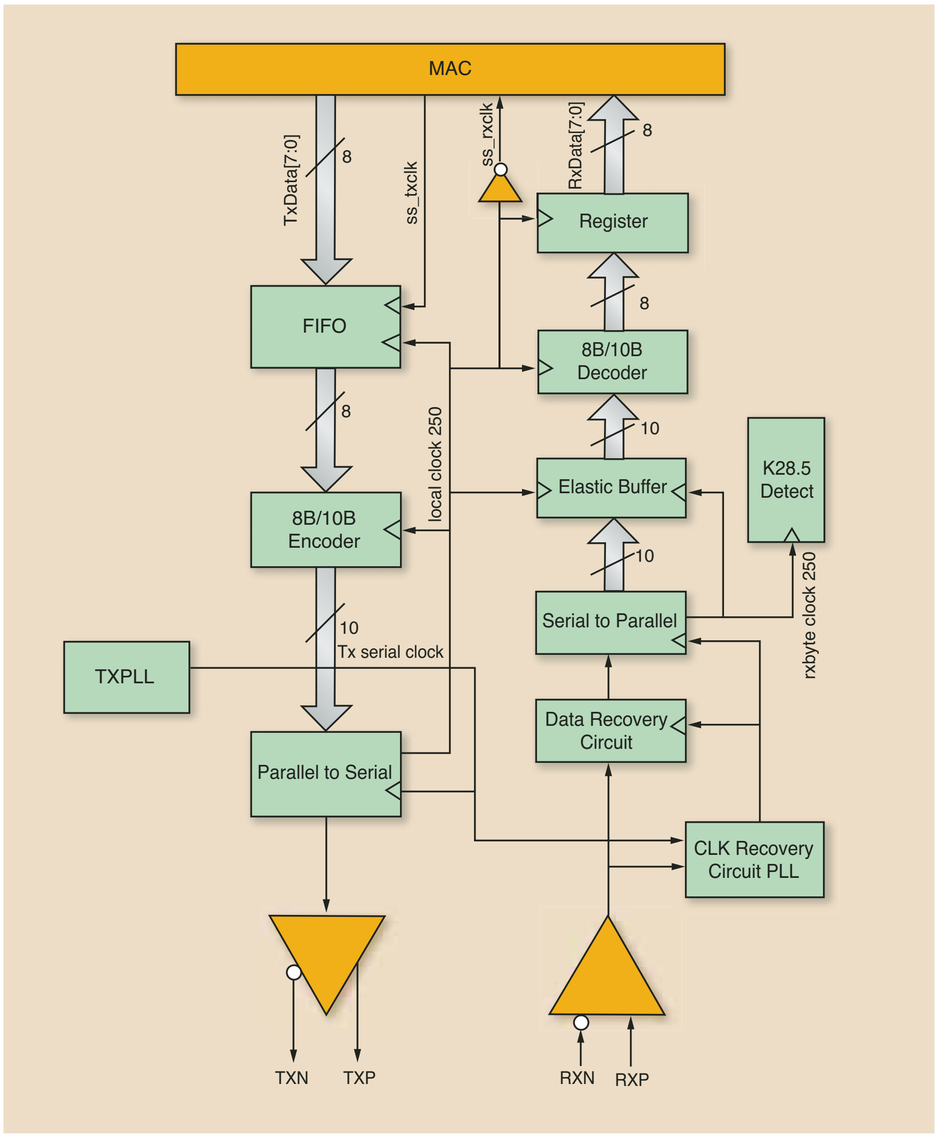


Figure 1 – Block diagram of PX1011A-EL1

an x4 link consists of four x1 lanes. A maximum of 32 lanes is allowed in a link, resulting in an aggregate bandwidth of 80 Gbps in each direction.

The physical layer's main function is to get packets of data across the link with a 10-12 or lower bit error rate. Factors such as signal voltage levels, equalization, receiver performance, transmit bit order, coding, and link initialization and training are dealt with in the physical layer. The main purpose of link initialization and training is to configure the link width (number of lanes), lane ordering, and correct polarity reversal within a differential conductor pair. The logical physical sublayer handles link training.

Intel defines a specification for the PIPE – Physical Interface for PCI Express – between the media access layer (MAC) and PHY. Part of the logical sub-layer of the physical layer resides in the MAC portion of the PIPE interface. The physical coding sublayer (PCS) of the physical layer and the electrical physical layer reside in the PHY portion of the PIPE interface. Therefore, PCI Express PHY devices that are based on the PIPE interface do not contain all of the functions described in the physical layer of the PCI Express specification.

The data link layer provides packets to and receives packets from the physical layer. The data link layer makes the unreliable link appear perfect to higher layers by retransmitting packets with errors. The transaction layer translates PCI transaction requests from the software layer into packets called transaction layer packets (TLPs) and relies on the underlying data link and physical layers to deliver the TLPs to the corresponding transaction layer at the destination node. An important function of the transaction layer is flow control.

PX1011A-EL1

A block diagram of the PX1011A-EL1 is shown in Figure 1. The interface between the PX1011A-EL1 and the higher layer logic is a variant of the PIPE interface.

On the transmit side, the PX1011A-EL1 receives 8-bit words of data from the MAC, along with a control bit that indicates whether the 8-bit word is data or a control character. The data is transferred at a rate of one word per cycle of a 250 MHz clock. The data is first buffered in a FIFO, which allows for phase differences between the PIPE clock and the internal 250 MHz transmit clock generated by the transmit phase-locked loop (TXPLL). The data is then 8B/10B encoded, with the 10-bit data serialized and differentially transmitted onto the transmission line.

On the receive side, the PX1011A-EL1 receives serial differential data from the transmission line. A clock is recovered from the data; this clock is used to sample the serial data in the center of the data eye. The retimed serial data is then passed through a serial-parallel converter.

The next step is to find the 10-bit symbol boundaries with a special 10-bit character called a "comma," or K28.5 character. The K28.5 character cannot occur by chance because of the concatenation of any other legal 10-bit characters. Once symbol synchronization has been achieved, the realigned 10-bit characters are passed through an elastic buffer that compensates for the frequency difference (if any) between the recovered and locally generated transmit clocks. This frequency compensation happens when you add or remove special characters (called "skip" characters) that are provided in the PCI Express protocol for this purpose. The retimed 10-bit characters are then decoded and the resulting 8-bit data words are registered and output from the PX1011A-EL1 to the MAC device.

The PX1011A-EL1 (shown in Figure 2) uses an 81-pin LFBGA package with 0.8 mm pitch. The package is approximately 9 x 9 mm and has a height of 1.05 mm. The power dissipation is less than 300 mW total during active operation. A leaded version is available, with a lead-free version to follow.

PXPIPE Interface

The PIPE specification was defined for an on-chip interface, and it does not address very well the difficulties that arise with a chip-to-chip connection. The PIPE interface defines a single 250 MHz clock, called PCLK, to which both the transmit and receive sides of the interface are synchronized. PCLK is an output of the PHY and an input to the MAC. This means that the MAC responds to the rising edge of PCLK by shifting out a word for transmission. Taking into account of the time of flight for signals across the channel, which includes PCB traces and possibly a connector, we see that the two instances of one-way propagation delay must be included in the timing budget. The total of the round-trip propagation delay plus the clock-to-out delay of the MAC plus the setup time of the PHY must be less than one PCLK period (4 ns).

The classic solution to this problem is to use source-synchronous clocking. With source-synchronous clocking, the clocks propagate in the same direction as the data rather than in the opposite direction. Because both clock and data incur the same propagation delay, they cancel out the timing budget. You could, in theory, have a propagation delay longer than the clock period.

By comparison, the Intel PIPE specification solves this timing budget problem by introducing the option of a 16-bit data interface, thus doubling the timing budget from 4 ns to 8 ns. This solution comes at a cost of higher pin count (thus a larger package and higher cost) and introduces an extra layer of logic to convert from 16 to 8 bits, thus increasing latency.

Philips Semiconductors has defined an alternative version of the PIPE interface named PXPIPE. Rather than a single byte-rate clock, we provide two clocks: a transmit byte clock and a receive byte clock. The PXPIPE interface also specifies the use of SSTL2 signaling. Being an on-chip interface, the original PIPE specification does not specify any signaling levels.

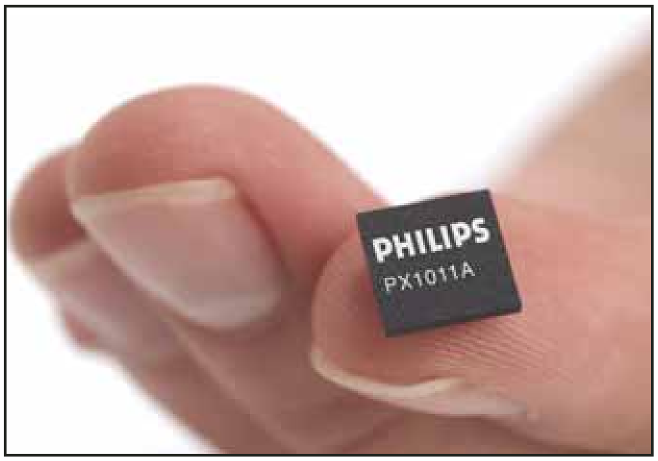


Figure 2 – PX1011A-EL1 device

Applications

The small size and low power consumption make the PX1011A-EL1 ideally suited for ExpressCard applications. ExpressCards are the new generation of PCMCIA PC cards used to add functionality to portable computers. ExpressCards use either a USB2.0 or PCI Express x1 host interface. They have restrictive component height requirements and stringent power consumption limits.

The PX1011A-EL1/Spartan-3/E PCI Express solution has many potential applications. This list shows some of the potential areas, and by no means represents an exhaustive enumeration. The Philips/Xilinx solution opens up a new horizon of potential applications that can make use of a low-cost and programmable PCI Express solution. The scope of applications is limited only by your imagination.

- Storage/RAID
- PC controlled and embedded test equipment
- Digital TV tuners
- Home printers
- Disk recorders
- Professional graphics boards
- Professional cameras
- Image capture and processing
- Digital media creation
- Digital music mixers
- Network security systems
- Voice over IP
- DSL modems
- Medical imaging (ultrasound, X-rays)

“PX Surfboard” Demo Design

A demo design called the “PX Surfboard” showcases the Philips/Xilinx PCI Express solution. The demo can be used for interoperability and compliance testing. In addition, it shows how you can design a system using the PX1011A-EL1/Spartan-3 solution. Schematics and Gerber files are also available.

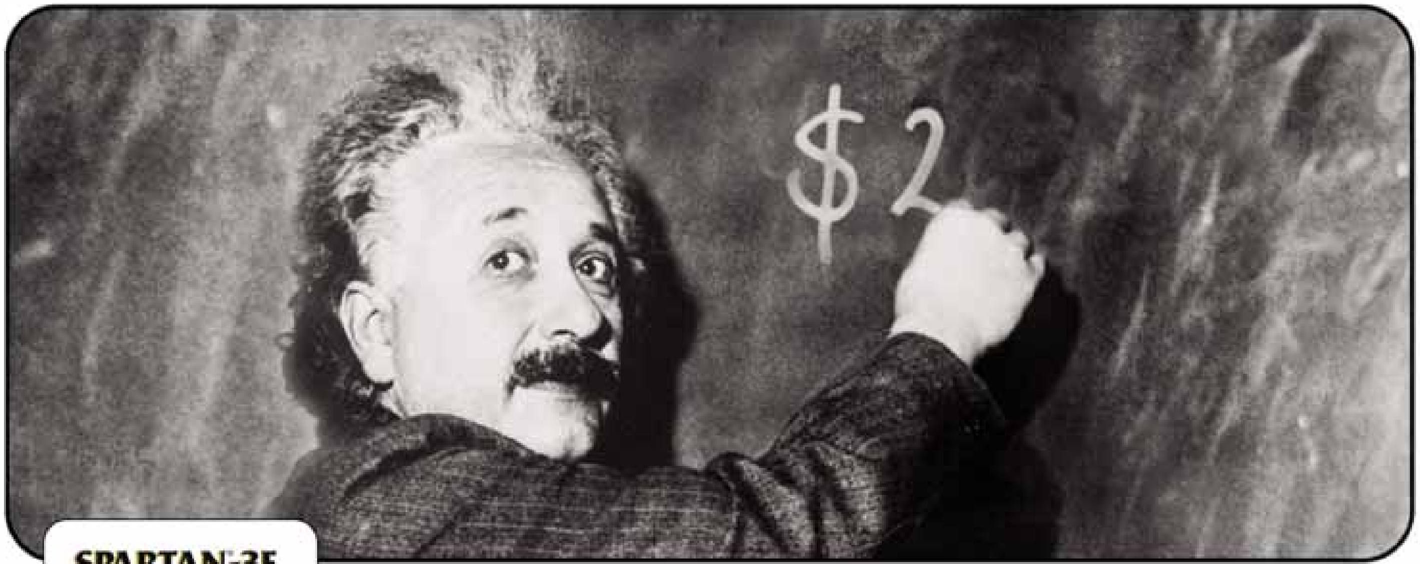
Conclusion

The Philips PX1011A-EL1 device, used in conjunction with a Xilinx Spartan-3/E FPGA containing the PCI Express Endpoint LogiCORE™ solution, provides a low-cost, low-power, fully standards-compliant PCI Express solution with proven interoperability with other vendors’ PCI Express solutions. For more information, please visit the following websites:

- *www.semiconductors.philips.com/markets/connectivity/wired/pciexpress/solution/index.html*
- *www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=DO-DI-PCIE-PIPE*

Introducing the new Spartan-3E family — the world's lowest-cost FPGAs

$$E = \frac{\$2}{100\text{K gates}}$$



SPARTAN-3E
FPGA

Amazing product…Incredible price…An industry first!

Wow! A 90nm platform FPGA loaded with features for just $2.00!* The new Spartan-3E devices, optimized for gate-centric designs, are perfect for consumer digital apps and so much more. Reducing the previous unit cost benchmark by over 30%, Spartan-3E FPGAs make it easier than ever to replace your ASIC with a more flexible, faster-to-market solution.

Easiest to use Software

Get the most for your money

With Spartan-3E FPGAs, two dollars buys a lot — 100K gates (the lowest cost per logic cell in the industry), PLUS embedded multipliers for high-performance/low-cost DSP, tons of RAM, digital clock managers, and all the I/O support you need. With a density range up to 1.6 million gates, and our production-proven 90nm process, now is the time to make your money work smarter!

MAKE IT YOUR ASIC

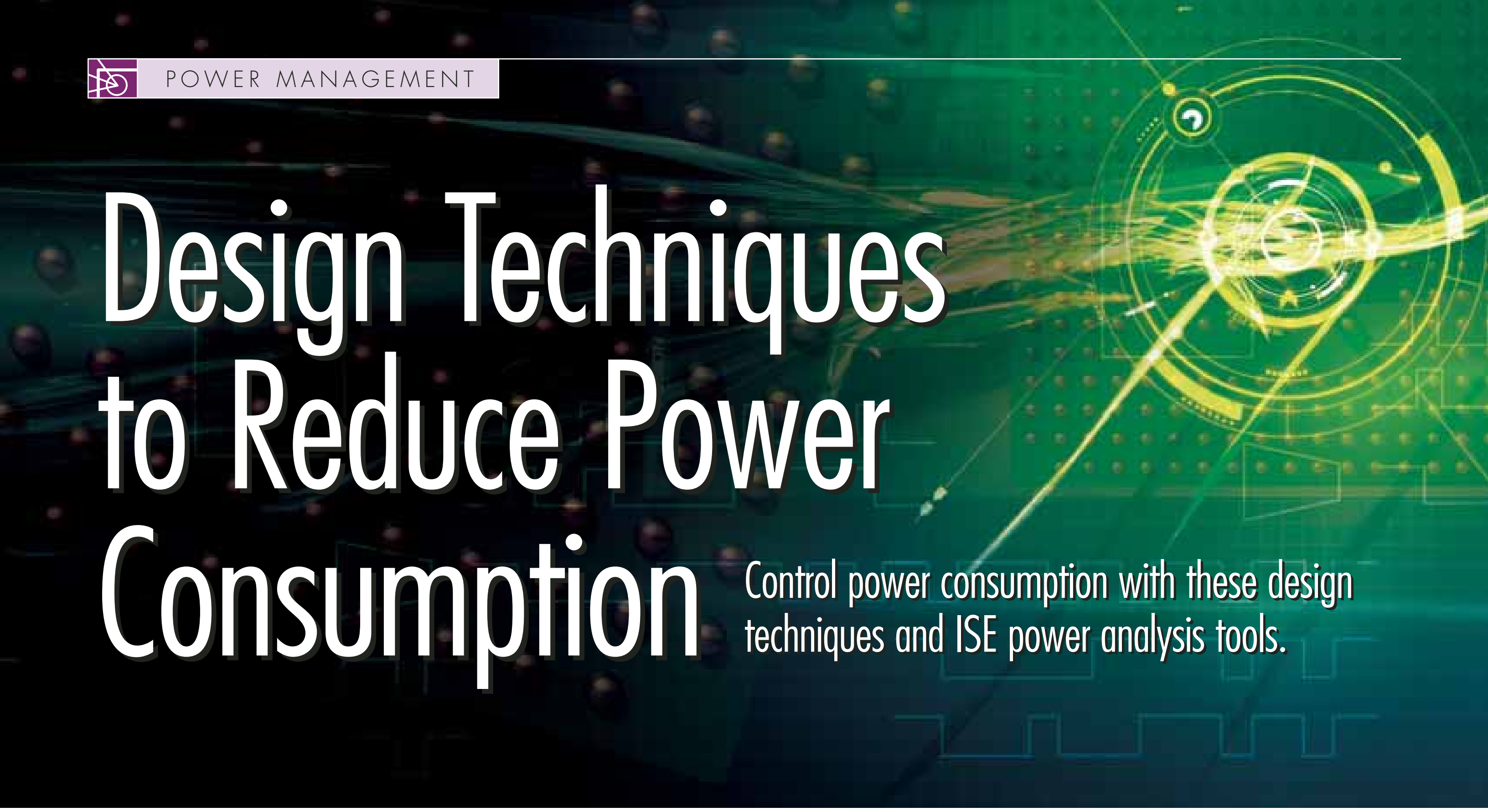
XILINX®
The Programmable Logic Company℠

For more information visit
www.xilinx.com/spartan3e

Earth Friendly
Pb-free devices available now

FORTUNE® 2005
100 BEST COMPANIES TO WORK FOR

* Pricing for 500K units, second half of 2006



Design Techniques to Reduce Power Consumption

Control power consumption with these design techniques and ISE power analysis tools.

by Arthur Yang
Sr. Product Applications Engineer
Xilinx, Inc.
arthur.yang@xilinx.com

Each generation of FPGAs gets increasingly faster, denser, and larger. What can you do to ensure that power doesn't increase in conjunction? A number of design decisions can impact the power consumption of your system, ranging from the obvious choice of device selection to the more minute details of choosing state machine values based on frequency of use.

To understand why the design techniques we'll discuss in this article conserve power, let's give a brief primer on power consumption.

Power comprises two factors: dynamic and static power. Dynamic power is the power required to charge and discharge the capacitive loads within the device. It is highly dependent on frequency, voltage, and loading. Each of these three variables is under your control in one form or another.

$$\text{Dynamic Power} = \text{Capacitance} \times \text{Voltage}^2 \times \text{Frequency}$$

Static power is the sum of power caused by leakage (source-to-drain and gate leakage, often lumped as quiescent current) for all of the transistors in the device, as well as any other constant power requirements. Leakage current is highly dependent on junction temperature and transistor size. For more details, see Xilinx® White Paper 221, "Static Power and the Importance of Realistic Junction Temperature Analysis," at *www.xilinx.com/bvdocs/whitepapers/wp221.pdf*.

Constant power requirements would include current leakage due to termination, such as a pull-up resistor. Not much can be done to affect leakage, but constant power may be controlled.

Think About Power Early

The decisions you make about power have the greatest impact in the early stages of your design. Deciding on a part can have huge implications on power, whereas inserting a BUFGMUX on a clock will have much less impact. It is never too early to start thinking about power for your next design.

The Right Part for the Job

Not all parts have the same quiescent power. As a general rule, the smaller the device process technology, the higher the leakage power. But not all process technology is created equal. For example, the dramatic differences in quiescent power for 90 nm technology between Virtex™-4 devices and other 90 nm FPGA technology can be seen in Xilinx White Paper 223, "Power vs. Performance: The 90 nm Inflection Point," at *www.xilinx.com/bvdocs/whitepapers/wp223.pdf*.

However, as quiescent power rises as process technology shrinks, dynamic power decreases because smaller processes come with lower voltage and capacitance. Consider what will be more relevant to your design – standby (quiescent) power or dynamic power.

All Xilinx devices have dedicated logic in addition to the general-purpose slice logic cells. These take the form of block RAM, 18 x 18 multipliers, DSP48 blocks, and SRL16s, among others. You should always use dedicated logic rather than its slice-based equivalent. Not only does dedicated logic have higher performance, but it requires less density and therefore consumes less power for the same given operation. Consider the types and quantity of dedicated logic when evaluating your device options.

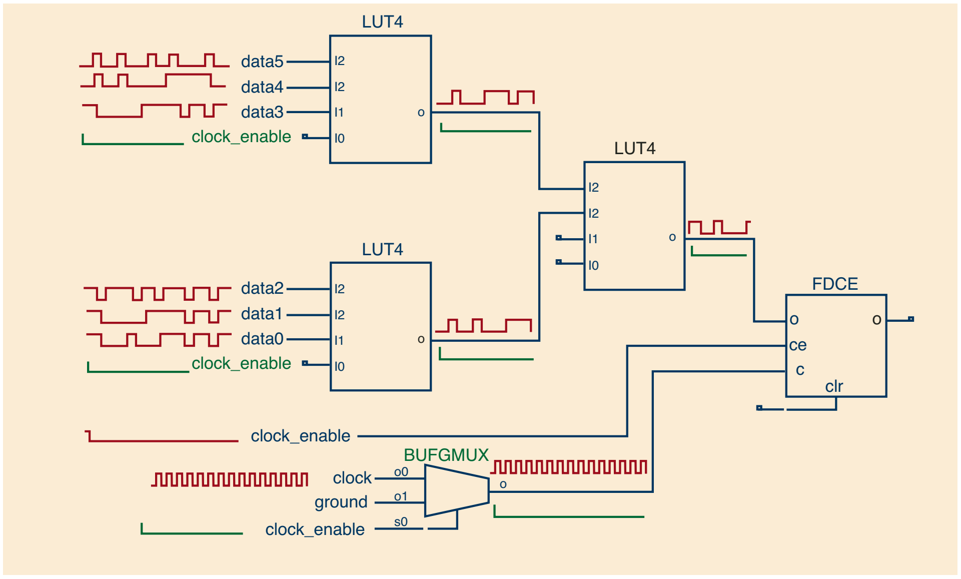


Figure 1 – Reducing power with enable signals

Selecting an appropriate I/O standard can save power as well. These are simple decisions, such as choosing the lowest drive strength or lower voltage standards. When a high-power I/O standard is required for system speeds, plan on a default state to lower power. Some I/O standards (such as GTL/+) require a pull-up to function properly. So if the default state of the I/O were high instead of low, the DC power through the termination resistor would be saved. For GTL+, setting the proper default state for the 50 ohm termination to 1.5V can result in 30 mA power savings per I/O.

Data Enable

Chip select or clock-enable logic is often used to enable registers when the data on the bus is relevant to them. Take this a step further and "data enable" the logic as early as possible to prevent unnecessary transitions between the data bus and combinatorial logic to the clock-enabled registers, as shown in Figure 1. The waveforms in red indicate the original design; the ones in green indicate the modified design.

Another option is to perform this "data enable" on the board instead of on the chip. Xilinx Application Note 347, "Decrease Power Consumption of a Processor using a CoolRunner™ CPLD," at *www.xilinx.com/bvdocs/appnotes/xapp347.pdf*, discusses this concept to minimize processor clock cycles. The concept here is to use a CPLD to offload simple tasks from the processor, allowing it to stay in a standby mode longer.

Applying this same idea to FPGAs is certainly feasible as well. Although FPGAs do not necessarily have a standby mode, using a CPLD to intercept bus data and selectively feed data to the FPGA can save unnecessary input transitions. CoolRunner-II CPLDs contain a feature called "data gate," which disables logic transitions on the pin from reaching the internal logic of the CPLD. The data gate enable may be controlled either by logic on-chip or by a pin.

State Machine Design

Enumerate state machines based on the anticipated next state condition and choose state values that have few switching bits between common states. By doing so, you can minimize the amount of transitions (frequency) for state machine nets. Identifying common state transitions and selecting values appropriately is a simple way to reduce power with little impact on the design. Simpler encoding styles (one-hot or grey-code) also utilize less decode logic.

Consider a state machine where the frequent state transitions are between states 7 and 8. If you select binary encoding for this state machine, this means that for every state transition between 7 and 8, four bits would need to change state, as shown in Table 1.

If the state machine were designed using a grey-code instead of binary, this would limit the amount of logic transitions required to move between these two states to only one bit. Alternatively, if states 7 and 8 were encoded as 0010 and 0011, respectively, this would also serve the same purpose.

Original State Machine Encoding				Current	Modified State Machine Encoding			
S3	S2	S1	S0	State	S3	S2	S1	S0
↓ 0	↓ 1	↓ 1	↓ 1	State 7	0	↓ 0	1	0
↓ 1	↓ 0	↓ 0	↓ 0	State 8	0	↓ 1	1	0
0	1	1	1	State 7	0	0	1	0

Table 1 – Reducing signal transitions with state encoding

Clock Management

Of all the signals in a design that can draw power, clocks are the largest offenders. Although a clock may run at 100 MHz, the signals derived from this clock often run at a small fraction of the main clock frequency (commonly 12% to 15%). In addition, the fanout for clocks is naturally high – so these two factors show that clocks should be studied for purposes of power reduction.

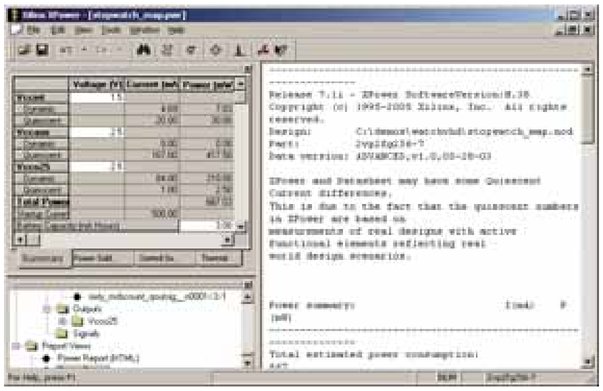
If a section of a design can be in an inactive state, consider using a BUFGMUX to disable the clock tree from toggling, instead of using clock enables. Clock enables will prevent registers from toggling unnecessarily; however the clock tree will still toggle, consuming power. But clock enables are better than nothing.

Isolate clocks to use the fewest amount of quadrants possible. Unused clock tree quadrants will not toggle, thereby lowering the load on the clock net. Careful floorplanning may achieve this goal without affecting the actual design.

Power Estimation Tools

Xilinx provides power estimation tools in two forms: a pre-implementation tool called Web Power Tools and a post-implementation tool called XPower. The Web Power Tools at *www.xilinx.com/power* provide power estimation based on ballpark estimates of logic usage. With this, you can get a power assessment with only a design utilization estimate – no actual design files.

XPower is a post-implementation tool that analyzes the actual device usage and, in conjunction with actual post-fit simulation data (in the form of a VCD file), delivers accurate power data. With XPower, you can analyze design changes for impact on overall power without touching a piece of silicon.



```
Release 7.1i - XPower SoftwareVersion:H.38
Copyright (c) 1995-2005 Xilinx, Inc.  All rights
reserved.
Design:       C:\demos\watchvhdl\stopwatch_map.ncd
Part:         2vp2fg256-7
Data version: ADVANCED,v1.0,05-28-03

XPower and Datasheet may have some Quiescent
Current differences.
This is due to the fact that the quiescent numbers
in XPower are based on
measurements of real designs with active
functional elements reflecting real
world design scenarios.


Power summary:                          I(mA)    P
(mW)
----------------------------------------------------
Total estimated power consumption:
```

Figure 2 – XPower

Web-Based Power Tools

Web-based power estimation is the quickest and easiest way to get an idea of device power consumption early in the design flow. A new version of these tools is released every quarter, so information is current, and no installation or downloading is required – just an Internet connection and a web browser. You can specify design parameters and save and load design settings, eliminating the need to re-enter design parameters with iterative use. Just an estimate of design behavior and a target device will get you started.

XPower – Integrated, Design-Specific Power Analysis

XPower, a free part of all Xilinx ISE™ design tool configurations, allows you to get a much more detailed estimate of your design-based power requirements. XPower estimates device power based on a mapped or placed and routed design. XPower calculates an estimate of power with an average design suite error of less than 10% for mature in-production FPGA and CPLDs. It considers device data along with your design files and reports estimated device power consumption at a high level of accuracy, customized to your specific design information.

XPower is integrated directly into ISE software and gives hierarchical and detailed net power displays, detailed summary reports, and a power wizard that makes it easy to run for new users. XPower can accept simulated design activity data and runs in both GUI and batch mode (Figure 2).

XPower considers each net and logic element in the design. The ISE design files provide exact resource use; XPower cross-references routing information with characterized capacitance data. Physical resources are then characterized for capacitance. Design characterization is continuous and ongoing for newer devices to provide the most accurate results. XPower uses net toggle rates as well as output loading. XPower then computes power and junction temperature, and can display individual net power data as well.

Conclusion

Increasing demands for cheaper and simpler thermal management – as well as power supplies coupled with the increasing power requirements of cutting-edge FPGAs – have elevated the concept of designing for low power to greater heights. The latest device offering from Xilinx, Virtex-4 FPGAs, offers the high performance of 90 nm without the assumed dramatic increase in static power. When used with Xilinx power estimation tools and considerations for low-power design, meeting your power goals is easier than ever.

Xilinx Events and Tradeshows

Xilinx participates in numerous trade shows and events throughout the year. This is a perfect opportunity to meet our silicon and software experts, ask questions, see demonstrations of new products, and hear other customer success stories.

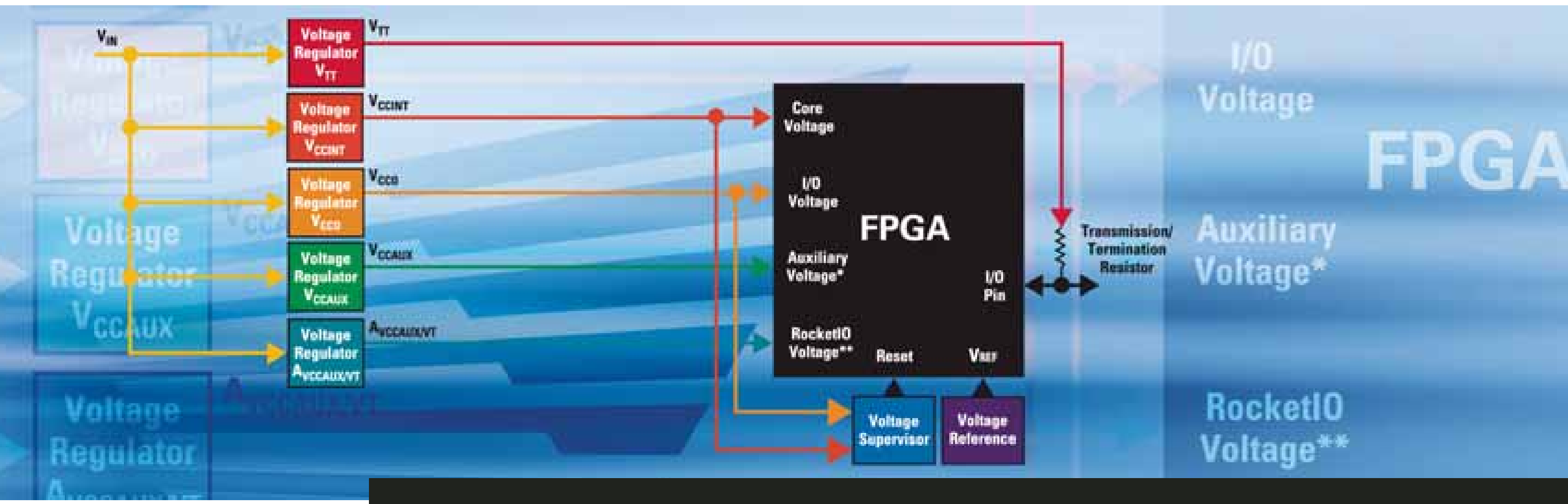
For more information and the current schedule, visit *www.xilinx.com/events/*.

Worldwide Events Schedule

North America

July 12-13	NSREC Seattle, WA
Aug. 16-18	National Instruments Week Austin, TX
Aug. 23-25	Intel Developer Forum Fall San Francisco, CA
Sept. 7-9	MAPLD Washington, D.C.
Sept. 13-14	Embedded Systems Conference Boston, MA
Oct. 18-20	MILCOM Atlantic City, NJ

Support Across The Board.™



Power Management Solutions for FPGAs

National Devices supported:

- Voltage Regulators
- Voltage Supervisors
- Voltage References

Xilinx Devices supported:

- Virtex™
- Virtex-E
- Virtex-II
- Virtex-II Pro
- Virtex-4FX, 4LX, 4SX
- Spartan™-II
- Spartan™-IIE
- Spartan-3, 3E, 3L

National Semiconductor
The Sight & Sound of Information

XILINX®

Avnet Electronics Marketing has collaborated with National Semiconductor® and Xilinx® to create a design guide that matches National Semiconductor's broad portfolio of power solutions to the latest releases of FPGAs from Xilinx.

Featuring parametric tables, sample designs and step-by-step directions, this guide is your fast, accurate source for choosing the best National Semiconductor Power Supply Solution for your design. It also provides an overview of the available design tools, including application notes, development software and evaluation kits.

Go to em.avnet.com/powermgtguide to request your copy today.

AVNET® electronics marketing

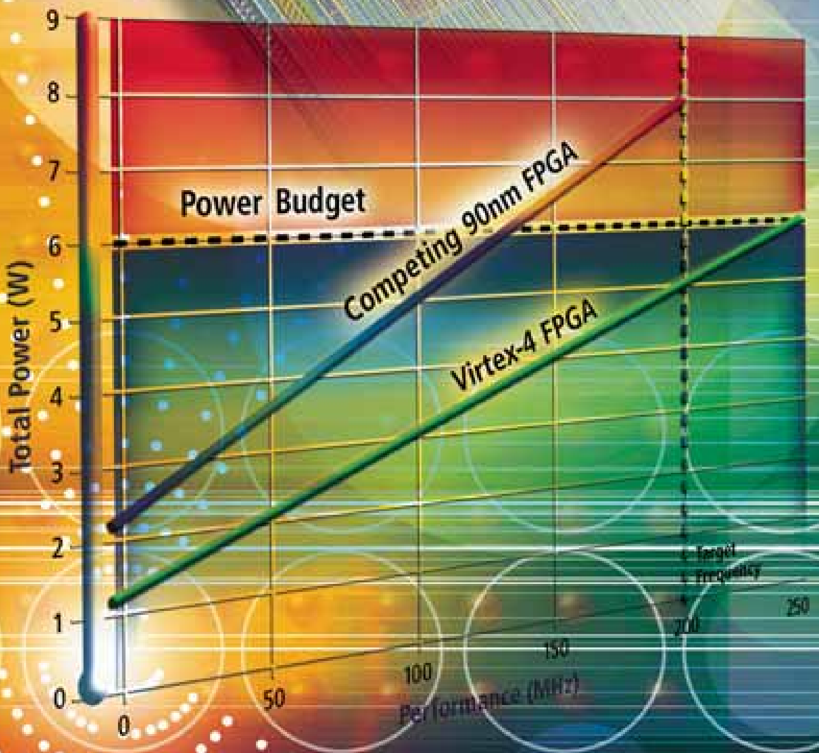
Enabling success from the center of technology™

1 800 332 8638
www.em.avnet.com

MEDA Supplier Authorized Distributor

Performance vs. Power: Getting the Best of Both Worlds

Xilinx conquers the 90 nm inflection point.



by Anil Telikepalli
Marketing Manager, Virtex Solutions
Xilinx, Inc.
anil.telikepalli@xilinx.com

The debate over which high-performance 90 nm FPGA has the lowest power is heating up. The industry has crossed a critical inflection point at the 90 nm process, where performance competes with power and thermal budgets. Customers want as much performance as possible, but increasingly the decision about which FPGA to use is based on which device consumes the least amount of power.

Excessive power is expensive in many ways. It creates the need for special design and operational considerations – everything from heat sinks to fans to sophisticated heat exchangers. Even the cost of larger power supplies must be considered.

Perhaps the most critical issue is the effect excessive power can have on reliability. As junction temperatures rise, transistors consume more power, further increasing the device temperature. Left unchecked, this phenomenon leads to thermal runaway. Continuously operating systems with junction temperatures from 85°C to over 100°C threaten the reliability of the device.

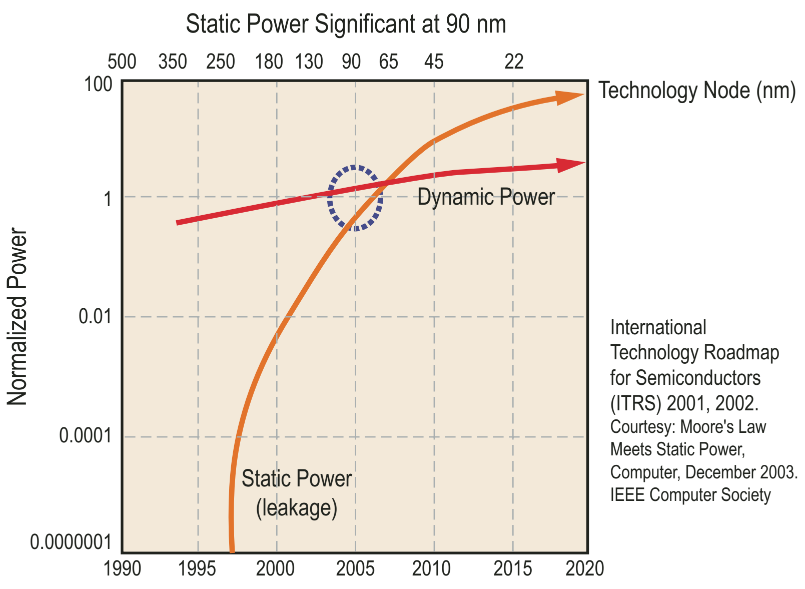


Figure 1 – The 90 nm inflection point

Fortunately, Xilinx® encountered the first evidence of this 90 nm inflection point more than three years ago, in the early development stages of Spartan™-3 FPGAs (the first Xilinx FPGA family with the 90 nm process). Xilinx began immediately developing new ways to cope with the inherent power issues posed by the 90 nm process. Consequently, when the higher performance Virtex™-4 family was introduced in September 2004, Xilinx was confident that the new family would simultaneously deliver the highest performance and lowest power consumption in a 90 nm FPGA.

Reducing Power in FPGAs

There are two major components to power consumption: static power and dynamic power. Each component poses a unique challenge. For the 90 nm FPGA, the most challenging component is static power.

Static Power

Static power is the standby power that is wasted even if the design is not performing any function. It occurs as a result of leakage current in the transistors within the FPGA. Leakage current increases as transistors get smaller with each new process. This principle is one of the major reasons the 90 nm process crosses a major inflection point (Figure 1).

For the first time, static power is threatening to eclipse dynamic power as the component responsible for the greatest amount of total power consumption in an FPGA. As processes get smaller, core voltage decreases and parasitic capacitance decreases; consequently, the rate of increase in dynamic power drops, despite the increase in frequency that accompanies a new process. In contrast, below 0.25 µms static power has grown exponentially with each new process.

This is where the inflection point really becomes a critical factor for the FPGAs and where Xilinx has established a substantial lead. Smaller transistors are faster, but they leak more. Thicker gate oxides reduce leakage, but they also reduce performance. However, unlike ASICs, ASSPs, and microprocessors, Xilinx FPGAs do not need all of their transistors to switch at maximum speed. A substantial number of transistors make up the configuration memory cells used for programmable logic, while pass transistors are used to implement the programmable interconnect routing. Configuration memory cells do not need to be fast, and programmable interconnect transistors only need to be fast from source to drain and not under gate control. These factors have allowed Xilinx to selectively increase gate-oxide thickness to reduce leakage current without compromising performance.

Virtex-4 FPGAs incorporate a new process approach called triple-oxide technology to solve the static power problem. Although this third gate-oxide layer is still very thin, these transistors exhibit substantially lower leakage than the standard thin-oxide transistors used in Virtex-II Pro FPGAs and in various other parts of Virtex-4 FPGAs.

In addition, Xilinx optimized a number of other transistor parameters (including V_T) to balance performance and leakage across I/O, configuration memory, interconnect pass transistors, and logic and interconnect buffers. Figure 2 shows that Virtex-4 FPGAs consume 50% less static power than their predecessor, 130 nm Virtex-II Pro FPGAs. We believe that this

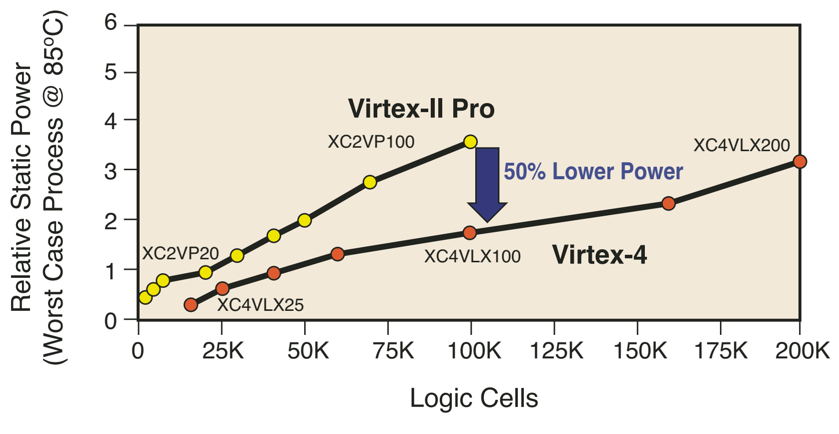


Figure 2 – The use of "triple-oxide" technology reverses the trend: the Virtex-4 device actually consumes less static power than its 130 nm predecessor.

is the first time in FPGA history that static power decreased when moving to a new, smaller process node.

Dynamic Power

The three contributing elements to dynamic power in an FPGA are core voltage (V), frequency (f), and parasitic capacitance (C). In addition, dynamic power is proportional to the data toggle rate (k). Fortunately, core voltage and capacitance decrease with each new process node, which lowers dynamic power. Conversely, increasing the operating frequency of a design increases dynamic power. The well-known formula for dynamic power that applies here is:

$$P = k * CV^2f$$

One major area of opportunity to reduce dynamic power consumption in FPGAs involves the way in which a design uses embedded functions. Embedded functions consume less static and dynamic power when implemented as hard-wired functions instead of configurable logic blocks and programmable interconnects. Hard fixed logic uses far fewer transistors than programmable logic. Additionally, the lack of programmable interconnect transistors in hard-wired embedded functions further reduces dynamic power consumption.

These hard IP cores occupy far less real estate, deliver much higher performance, and consume 80-95% less power than soft IP versions of the same functions. And by making these hard IP cores programmable and parameterizable, you can maintain the flexibility inherent to FPGAs.

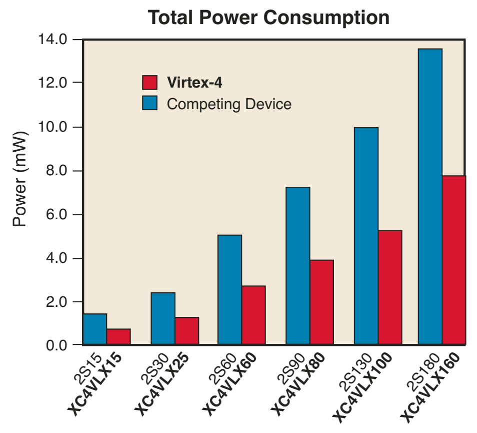
Functions that Xilinx provides as hard IP cores in Virtex-4 FPGAs include:

- 450 MHz PowerPC™ processors for all microcontroller and embedded processing applications with an APU (auxiliary processing unit) interface for hardware acceleration
- 500 MHz XtremeDSP™ slice capable of simple math and filter functions to complex high-performance DSP functions
- 500 MHz digital clock managers (DCM) and phase-matched clock dividers (PMCD) that support clock synthesis, clock management, and phase matching
- A ChipSync™ block in every I/O with built-in SERDES and a data-alignment function to simplify source-synchronous interfaces in memory, networking, and telecom applications
- RocketIO™ transceivers (622 Mbps-10.3125 Gbps) with built-in physical coding sublayer (PCS) and physical media attachment (PMA)
- Tri-mode Ethernet MACs (10/100/1000 Mbps) that can interface directly with RocketIO transceivers
- Smart RAM memory with distributed RAM and 18 Kb block RAM – each block RAM has built-in FIFO logic to convert RAM into FIFO and comes with built-in error correction code (ECC) circuits

Besides the obvious advantages associated with moving these commonly used blocks into hard IP, you must not overlook the inherent contribution that Xilinx advanced silicon modular block (ASMBL) architecture makes to the Virtex-4 power advantage. Because each of the three Virtex-4 platforms – the LX, FX, and SX – satisfies distinct requirements for a particular application domain (logic, embedded processing, and signal processing), their standard ratio of logic cells, memory, I/O, DSP, and processors has been optimized for that domain. Consequently, the Virtex-4 device is the first FPGA to offer domain-optimized power consumption.

End-Market Power Requirements

Having achieved substantial power savings both in static power (as a result of triple-oxide technology) and dynamic power (using embedded hard IP), you might wonder what it all means for your designs. The simplest example often provides the best perspective. Using an equivalent amount of generic logic and memory in Virtex-4 devices and the nearest competitor's devices of equivalent density, with no consideration of other embedded IP, the Virtex-4 FPGA saved 1-5W in power (see Figure 3). But how does this translate to measurable benefits in real-world applications?



Design Details -- Logic & Memory

Static Power at T_j = 85°C

Dynamic Power at 200 MHz
- 50% of LUTs and FFs in Virtex-4 device; equivalent ALUTs and FFs in corresponding competing device, 12.5% toggle rate
- M4K blocks used in competing device, equivalent 18 Kb block RAM in corresponding Virtex-4 device

Figure 3 – Virtex-4 designs consume 1 to 5 watts lower power per FPGA

Power Budgets

Every product has a power budget driven by standards, cost goals, and reliability requirements. As power consumption and temperature are interrelated, it is important to meet operating temperature goals as well. System architects have specific power budgets at the system level – for each board as well as for devices used

Product acceptability, reliability, and profitability depend as much or more on power efficiency as they do on performance.

on the board. Markets where high-performance FPGAs are used, such as wired and wireless networks, storage/servers, automotive, and aerospace/defense, also have aggressive power budgets. Let's discuss a few applications where tight power budgets are critical.

Wired Networks: Metro Aggregation

Metro aggregation refers to the aggregation of access connections at central offices (COs) within a metropolitan area network (MAN). The equipment within each CO must operate continuously, placing a heavy burden on operational costs and the effective capacity of power supplies and air conditioning systems. Any means by which equipment vendors can help reduce total system power consumption equates to real benefits for service providers.

The power budgets for the cards in a rack of metro aggregation equipment usually average 20-30W. The FPGAs used in these boards consume 4-5W each, and many designs use multiple FPGAs.

For example, power budgets for a multi-service provisioning platform line cards and FPGAs include:

- 12-port DS3 card: 30W; FPGA = 4-5W
- 4-port OC-12 card: 28W; FPGA = 4-5W
- 12-port 10/100 Base-T card: 50W; FPGA = 4-5W
- 32-port T1/E1 card: 9 W; FPGA = 2-3W

Using Virtex-4 FPGAs in these applications would dramatically benefit service providers' operational costs. Each Virtex-4 FPGA can save 1-5W when compared to competitive 90 nm FPGAs.

Wired Networks: Metro Access

Unlike the metro aggregation equipment deployed in COs, metro access equipment exists at the edge of the network. It is deployed outdoors, where air flow is limited and air conditioning is virtually non-existent. Example systems include passive optical networks (PONs), digital loop carriers (DLCs), and cable modem termination systems (CMTS). These systems operate continuously at temperatures often well above 85°C, taking junction temperatures as high as 100°C. Transistor leakage current – and hence static power – increase with temperature. As a result, equipment vendors in this space are constrained by stringent power budgets (10 to 12W per card, 4 to 8W per FPGA) to ensure reliability.

As power-sensitive as these applications are, saving as little as 0.5W can make a design workable. Virtex-4 devices eliminate as much as 1-5W per FPGA, dramatically benefiting both equipment vendors and service providers.

Wireless Base Stations

Because of its quick deployment and low establishment costs, the growth of the cell phone market has overtaken the growth of fixed-telephony networks. Once again, service providers can measure the value of reduced power consumption in Virtex-4 FPGAs both in terms of the mitigation of reliability issues (arising from the outdoor environment in which the base stations are deployed) as well as the reduction of operational expenditures.

Service providers running a typical wireless base station network of 35,000 units can save more than $1M per year just in electricity charges. Consider the following power budgets:

- 16 line cards/base station;
  1 FPGA/line card
- Power budget/line card = 20W
- FPGA power budget = 6W

Based on an extremely conservative estimation of a 2W power reduction using Virtex-4 FPGAs, service providers would see a 32W power savings per base station, amounting to a savings of 1.12 MW for the entire network. Using 10¢/KWh, this saves about $1M per year for 35,000 base stations in the network.

Cutting 32 watts per base station also impacts service providers' bottom lines in the form of capital expenditure reductions for cooling equipment costs, battery backup costs, and power supply and power management costs.

Conclusion

The battle to deliver maximum performance at the lowest cost has taken center stage in the evolution of FPGAs. Today, customers are demanding minimum power expenditure as well. Power conservation impacts every budget, whether technological or financial. Product acceptability, reliability, and profitability depend as much or more on power efficiency as they do on performance. Besides offering a robust feature set, Virtex-4 FPGAs exhibit a real power consumption advantage.

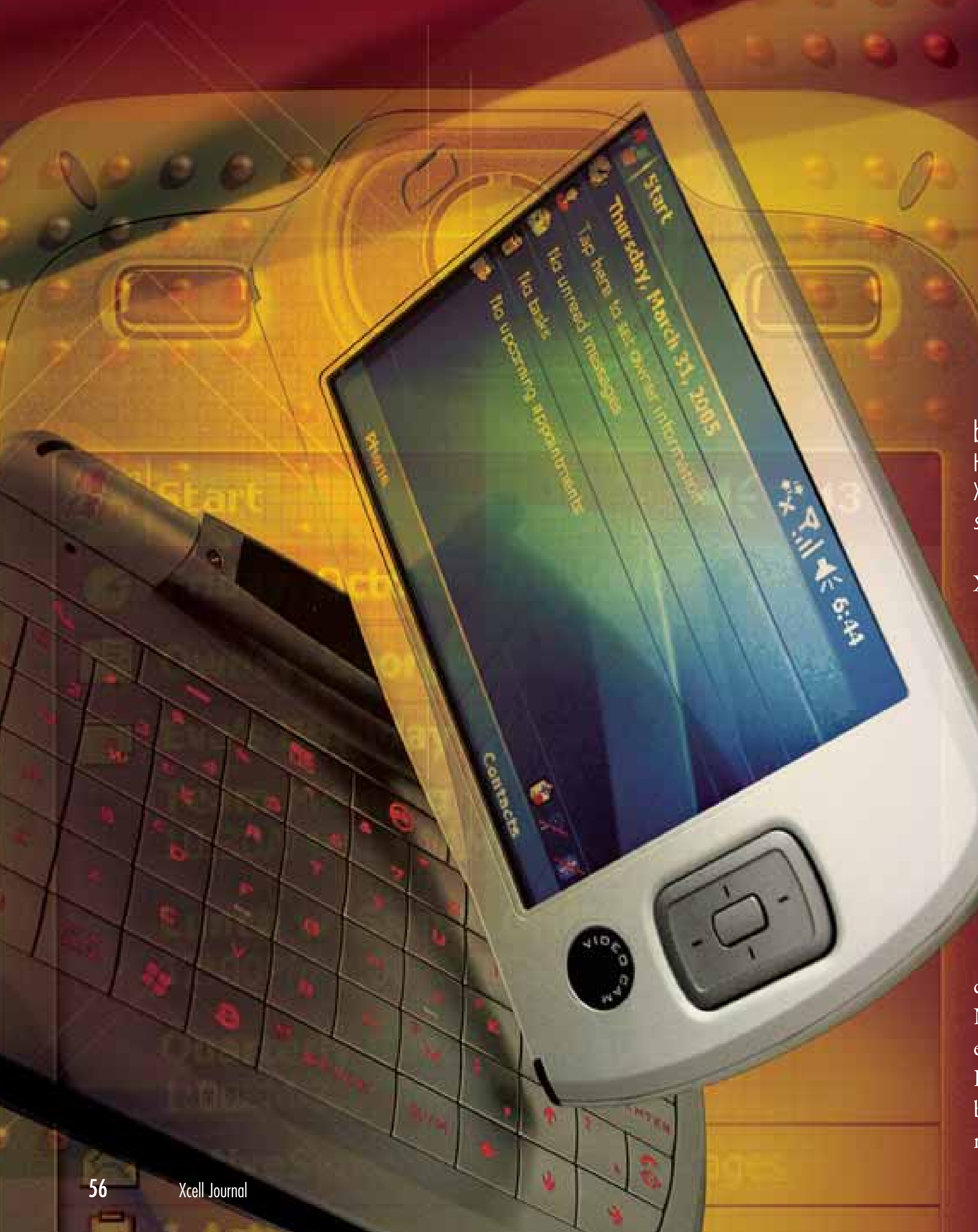
Nonetheless, competition in the FPGA market does not end with 90 nm devices. Interesting new dynamics arise when moving into the 65 nm node and below. Fortunately for Xilinx, one inherent value of using triple-oxide technology is that it scales nicely with each new process.

As for the value of embedding hard IP where appropriate, it is practically an industry axiom. Xilinx has incorporated the right amount of programmable embedded IP with programmable logic to make the whole solution more flexible, with higher performance and lower power. In the long term, customers will only use platform FPGAs that provide the best performance and power.

For more information about power budgets, seminars/tutorials, white papers, and power analysis/optimization tools, visit *www.xilinx.com/virtex4/lowpower*.

Merging CPLD Features into Handheld Applications

The integration of reprogrammable logic into high-volume, low-power consumer applications.



by Steve Prokosch
High Volume Marketing Manager
Xilinx, Inc.
steve.prokosch@xilinx.com

Xilinx® CPLDs have been a great solution for control logic, state machines, and simple system integration for years, but were not used in consumer portable equipment because of the stigma of high power consumption. In the late 1990s, CPLDs ventured into the low-power domain with the first CoolRunner™ product family. But prices remained out of reach for many high-volume applications. Today, through a steady progression of Moore's law, pricing is low enough to be competitive with discrete logic devices. Now you can easily implement a wide variety of logic functions in a single package. Plus, you can save board space and get the benefits of reprogrammable logic to maximize time to market.

CPLDs are also offering more integration of common features, such as voltage translation, I/O standards translation, HSTL and SSTL memory interfaces, clocking features, and higher performance flip-flops.

Some designers may still believe that CPLDs are just logic and flip-flops. But with today's new breed of CPLDs, you get a lot more from a single device. For instance, new discrete logic devices are introduced every year, driven by demands from system integration that arise from product legacy mismatch. CPLDs fill this niche nicely by offering multiple I/O banks at low cost points.

CPLDs are also offering more integration of common features, such as voltage translation, I/O standards translation, HSTL and SSTL memory interfaces, clocking features, and higher performance flip-flops. But integration isn't the only addition to current CPLD product offerings; CoolRunner-II CPLDs include an elaborate scheme to prevent read back and copycat designs. Potential thieves would have to go to extreme measures for the encapsulated design file, such that the money and effort required becomes truly cost prohibitive.

CoolRunner-II CPLDs also offer low-power features to assist you in lowering overall dynamic power consumption. These features include gating inputs, clock frequency scaling (division and doubling), and input hysteresis. With these features, you can differentiate your product and still consume low power, maintain low cost, and get your product to market quickly.

Success Story

HTC Corporation specializes in designing and manufacturing mobile computing and communication solutions for OEM and ODM customers. Today, with design expertise in consumer products, HTC has expanded into the wireless handset market. Figure 1 shows the HTC GSM/GPRS Magician handset platform with camera, touch screen, SD/MMC memory expansion, microphone, audio jack, and mini USB connector for synchronization.

The company's products include smart phones, smart music phones, PDA phones, and compact PDAs. With this strong focus, Microsoft chose HTC as a platform development partner for Windows CE designs.

According to THT Business Research, HTC is the world's largest producer of pocket PC-based PDAs, accounting for 48% of total OEM supply. They began manufacturing processes in the second quarter of 2001, foreseeing a huge market in G3 handsets. Now HTC produces more than 90% of the PDA phones based on Windows CE and continues to ship more than 3.5 million units per year.

Figure1 – HTC Magician handset platform

By being cost competitive and continually adding features that other handset manufacturers wait to see a demand for, HTC has grabbed market share from some of the most well-known manufacturers. Studies show that low-end handsets are decreasing in volume, while feature-rich handsets are continuing to grow (see Table 1). By 2006, approximately 40% of new handset sales will be in the entry-level category, a figure forecasted to decline 10% each year thereafter. With this dramatic shift, handset manufacturers will have to change their products rapidly to meet consumer trends.

In addition, Windows CE operating system handsets have seemed to catch on with users. According to a Gartner Research report from April 2005, Windows CE-based PDAs have taken the lead in shipments (Table 2).

Three years ago, HTC began to study alternative logic devices that would give their products a competitive advantage in the market. They looked at price, features, ease of use, and power consumption. The study also included integration challenges and how to keep re-inventing new products with market dynamics.

To keep pace with changing technology in displays, touch screens, memory, and wireless communications, HTC needed a flexible and feature-rich solution. With changes to these technology products occurring every six to eight months, redesigns would occur often to incorporate the latest and lowest cost components. HTC examined multiple vendors from both a support perspective and best features/lowest cost competitiveness. The company also looked at partnerships, paying particular attention to those who stand behind promises of pricing, roadmaps, and technical assistance.

A Clear Choice

With a list of both supplier and component criteria, HTC chose Xilinx as a preferred vendor for low-power CPLD products. Weighing competitors not only by price, but by reputation of product, delivery, quality, and attention to special needs, Xilinx came out on top. According to HTC, the device features were what tipped the scale. Not only did the base power consumption meet or exceed power budgets, but key low-power enhancements played a large role in part selection. "HTC has received a substantial amount of performance capabilities in Xilinx products," said Peter Chou, HTC's president. "Their combination of leading-edge technologies, complete programmable system design, and full technical service support are essential to the success of HTC."

When compared to other suppliers, power consumption was basically equal; the deciding factor was integrating differences from other competing products. Integrated low-power features gave HTC designers a creative method to both increase battery life and phone features. By using clock features, the designers eliminated external oscillators and reduced costs. These clocking features also helped reduce overall power consumption and increase battery life. With this one feature, they reduced board space without using additional power.

HTC has embraced low-power CPLDs with a passion, taking advantage of the features offered by CoolRunner-II products ...

Another low-power feature that helped overall battery life was input gating. This technique has been used on other products, but Xilinx was one of the first companies to incorporate it in a CPLD. This feature helped reduce overall dynamic power consumption by powering down circuits to their standby or quiescent state. By turning off circuits that are only used part of the time, battery life can be extended well beyond a competitor's product with the same features.

Device integration also added a key advantage to HTC's designs. By using a single device to shift voltage levels for certain circuits, discrete devices were eliminated. Plus, the availability of small chip-scale packages saved board space when compared to discrete devices. By consolidating various functions into a single reprogrammable device, HTC saved layout problems when faced with high integration goals. Also, by condensing signal paths on a PCB layer, layout became much easier.

	2001	2002	2003	2004	2005	2006	2007	2008
Entry Level	97.7	95.4	84.2	68.6	52.5	40.1	29.3	20.8
Midrange Feature	1.9	3.1	10.6	21.1	28.0	31.4	32.2	34.3
High End	0.1	0.6	3.4	7.3	12.6	19.0	26.4	30.3
Converged Mobile Device	0.4	0.8	1.8	3.0	7.0	9.5	12.0	14.6

Table 1 – Worldwide mobile phone shipment share by feature tier, 2001-2008 (%) (Data supplied by IDC – worldwide mobile phone 2005-2008 forecast by feature tier)

Operating System	Units Shipped	Market Share
Windows CE	1,759,497	45.5%
Palm OS	1,301,740	33.7%
RIM	698,000	18.1%
Linux	24,300	0.6%
Others	82,000	2.1%

Table 2 – Shipment of PDAs by operating system worldwide

Faster Designs with WebPACK

To make use of special features as HTC has, you need to be able to implement them without extra effort. Low-power features such as clock frequency scaling, signal gating, and hysteresis are easy to use and have example code listed in ISE™ WebPACK™ software. This enabled HTC to turn designs around faster with the confidence that all critical timing and power consumption goals would be met.

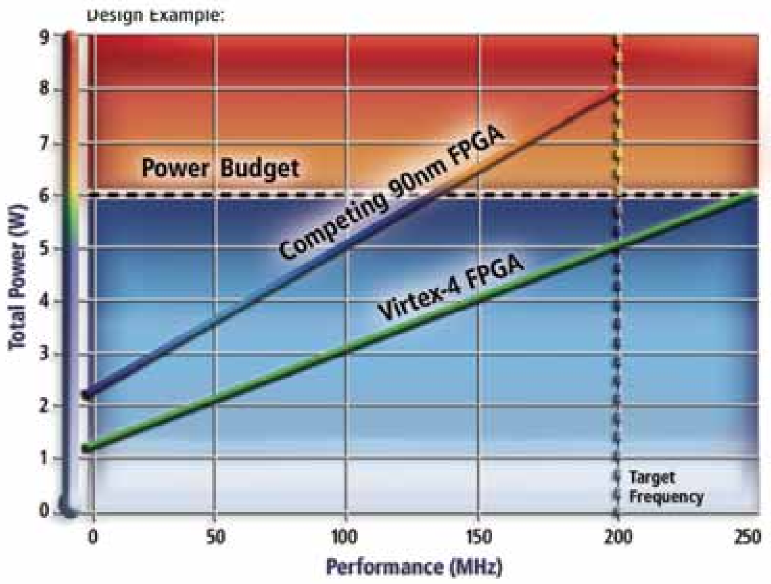
The best part of Xilinx WebPACK software is that the features are free. Not only can you design with high-level languages like VHDL and Verilog, but functional and timing verification tools work flawlessly. Plus, the XPower power analysis tool gives you a reliable estimation of how much power will be consumed at each point of the operation. This easy-to-use software tool gave the HTC designers confidence that the results they saw through simulation were the results they measured in actual designs. For HTC, precisely knowing what to expect let them pack the most into their designs.

Conclusion

HTC has embraced low-power CPLDs with a passion, taking advantage of the features offered by CoolRunner-II products and obtaining a larger share of mid- to high-end handset sales than their competitors. With more users embracing more features, HTC is well positioned to continue their leadership and maintain growth in emerging markets. Through the use of Xilinx silicon products and software tools, HTC will continue to design innovative solutions and deliver these products on time.

To further explore the Xilinx family of CPLDs, visit *www.xilinx.com/cpld/index.htm*. To see HTC's line of handsets, visit *www.htc.com.tw*.

WITH COMPETING FPGAs, POWER HAS BECOME A BURNING ISSUE.



Design Example: LX60 vs. 2S60. Target Frequency = 200 MHz. Worst-case process.
20K LUTs, 20K Flip-Flops, 1Mbit On-Chip RAM, 64 DSP Blocks, 128 2.5V I/Os
Based on Xilinx tool v4.0 and competitor tool v2.1
For higher density devices, achieve up to 5W lower power

VIRTEX 4

Get 1-5W lower power per FPGA, only with the Virtex-4 family

Check the specs for yourself at realistic operating temperatures ($T_j = 85^{\circ}C$). Different logic architecture or dielectric just won't do it. No competing FPGA comes close to Virtex-4 for total power savings—take it to the lab and see.

- **73% lower static power**
- **Up to 86% lower dynamic power**

UNIQUE TRIPLE-OXIDE TECHNOLOGY & EMBEDDED IP

At the 90nm technology node, power is the next big challenge for system level designers. An inferior device can suffer leakage, dramatic surges in static power, and thermal runaway. That's why we designed our Virtex-4 FPGAs with Triple-Oxide Technology™, embedded IP, and power-saving configuration circuitry. Now you can meet your performance goals, while staying within the power budget.

Visit ***www.xilinx.com/virtex4/lowpower*** today, and get the right solution on board before your power issues start heating up.

XILINX®
The Programmable Logic Company℠

www.xilinx.com/virtex4/lowpower

www.techonline.com

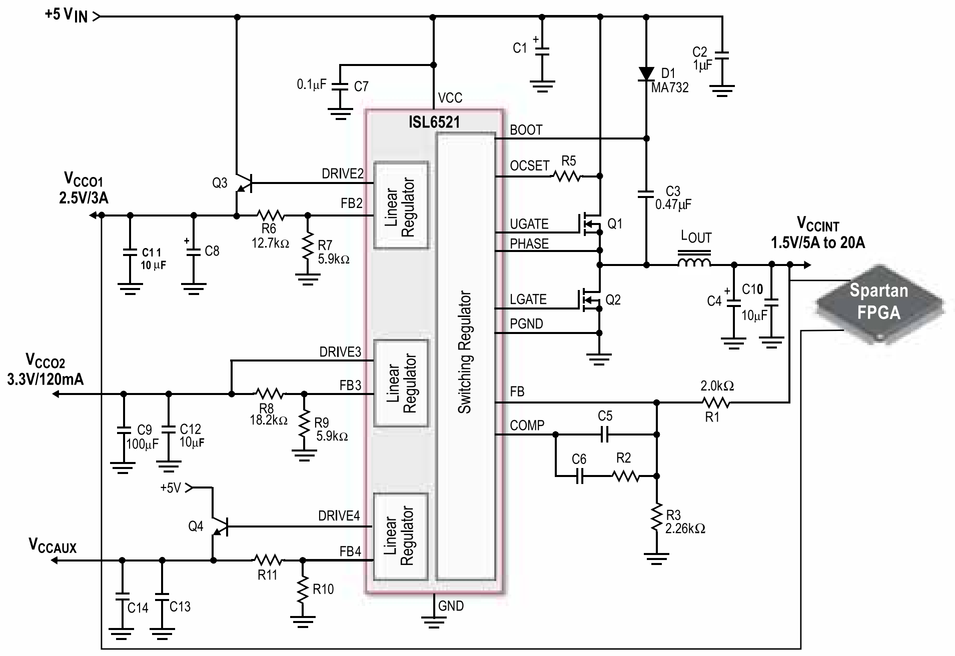
View The TechOnLine Seminar Today

BREAKTHROUGH PERFORMANCE AT THE LOWEST COST

Power Solutions for FPGA Design Has Never Been So Easy

Intersil offers a wide range of Power Supply solutions for the latest generations of Xilinx FPGA-based systems.

Reduce your Spartan-3 board size and increase efficiency with Intersil's multiple output DC/DC Switching Regulators or Switching Regulators with Integrated MOSFETs.



Intersil's Xilinx Spartan-3 Power Solutions

	Peak I_{CCINT} Current (A) Requirement*		Recommended Intersil Power Solutions	
Device	@ 325MHz	@150 MHz	V_{IN}=+5V	V_{IN}=+3.3V
XC3S50	0.6	0.3	ISL6410, ISL6410A, or EL7536	
XC3S200	1.5	0.7	ISL6528, EL75XX Family or ISL6521	EL75XX Family or ISL6526
XC3S400	2.3	1.1		
XC3S1000	4.2	2.0		
XC3S1500	6.6	3.2		
XC3S2000	9.6	4.6	ISL6521 or ISL6539	ISL6526
XC3S4000	15.3	7.2		
XC3S5000	17.6	8.4		

Selecting the Right Switching Regulator for Spartan-3 FPGA-Based System

How many power supplies do you need?

Three or Four...

For up to 20A - use Intersil ISL6521

Two...

For smallest size - use Intersil ISL6528

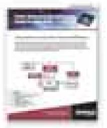
For highest efficiency - use Intersil ISL6539

One...

For ease of use and Integration up to 0.5A - use Intersil ISL6410 and ISL6410A Switching Regulators with Integrated FETs

For ease of use and Integration from 1A to 8A - use Intersil EL75XX Switching Regulators with Integrated FETs

For most flexibility up 20A - use Intersil ISL6526



For detailed application note on using Intersil Power Management Solutions in Xilinx® FPGAs go to **www.intersil.com/xfpgas**

Document includes Intersil's recommeded power supply solutions, IC schematics, functional block diagrams, BOMs, selection tables, Xilinx® FPGA power tables, DDR memory power solutions, sequencers and supervisory circuits, and layout considerations.

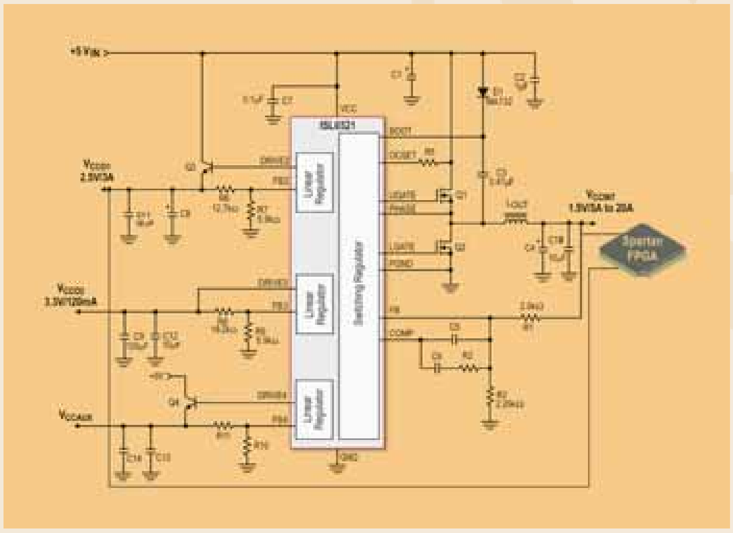
Download ISim:PE, Intersil's personal edition offline design simulation tool at **www.intersil.com/iSim**

Find the power management device that matches your design requirements at **www.intersil.com/pmps**

Intersil – Switching Regulators for precise power delivery.

intersil®
HIGH PERFORMANCE ANALOG

The Intersil ISL6521 Simplifies Power Solutions for Xilinx FPGAs



Features

- Provides one regulated voltage-switching regulator capable of 20A and three linear regulators capable of 120 mA or up to 3A with an external transistor
- Externally resistor-adjustable outputs
- Simple single-loop control design / voltage-mode PWM control
- Fast PWM converter transient response / high-bandwidth error amplifier / full 0% to 100% duty ratio
- Excellent output voltage regulation / all outputs: ±2% over temperature
- Overcurrent fault monitors / switching regulator does not require extra current sensing element, using instead the MOSFET's Rds(on)
- Small converter size / 300 kHz constant frequency operation / small external component count
- Commercial and industrial temperature range support
- Pb-free available (RoHS compliant)

Applications

- FPGA and PowerPC™-based boards
- General purpose, low-voltage power supplies

High integration with full features enables space and cost savings.

The Intersil ISL6521 quad regulator IC solution includes one switching regulator for loads as high as 20A and three linear regulators that each drive 120 mA or 3A with external transistors. High integration brings these four regulators together with full protection, power-on reset, and softstart features in one space- and cost-saving IC.

Applications for this new IC range broadly, providing power solutions for FPGAs, ASICs, POL, embedded systems, and I/O across medical, industrial, computing, and telecom.

The ISL6521 combines multiple switchers and/or linears in a single 16-lead SOIC package, delivering the integration and flexibility FPGA-based designers need. This single IC solution increases available board space while reducing costs and the number of required external components.

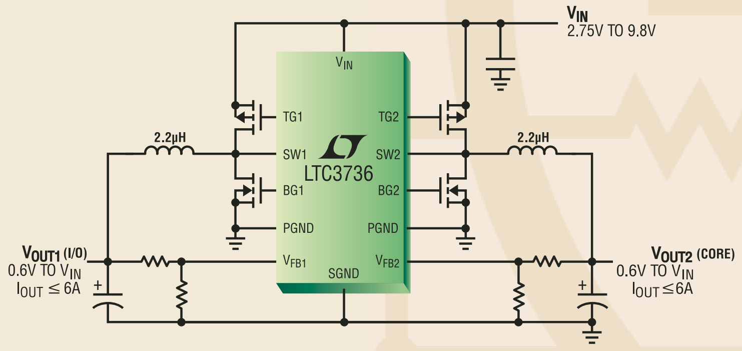
The ISL6521 PWM controller is intended to regulate the low-voltage supply that requires the greatest amount of current (usually the core voltage for the FPGA, ASIC, or processor) with a synchronous rectified buck converter. The linears are intended to regulate other system voltages, such as I/O and memory circuits. Both the switching regulator and linear voltage reference provide ±2% of static regulation over line, load, and temperature ranges. All outputs are user-adjustable by means of an external resistor divider. All linear controllers can supply up to 120 mA with no external pass devices.

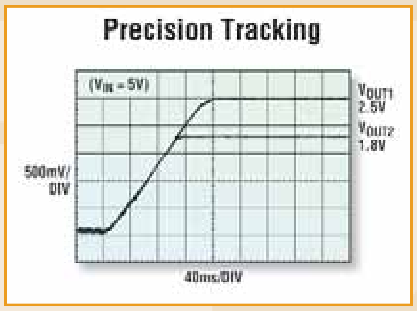
Employing bipolar NPNs for the pass transistors, the linear regulators can achieve output currents of 3A or higher with proper device selection. The ISL6521 monitors all output voltages. The PWM controller's adjustable overcurrent function monitors the output current by using the voltage drop across the upper MOSFET's rDS(ON). The linear regulator outputs are monitored through the FB pins for undervoltage events.

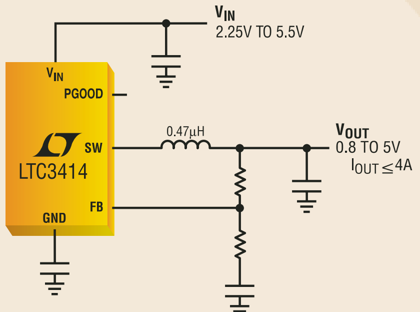
For a copy of "Power Management Application Guide for Xilinx® FPGAs," visit *www.intersil.com/Xilinx/index.asp.*

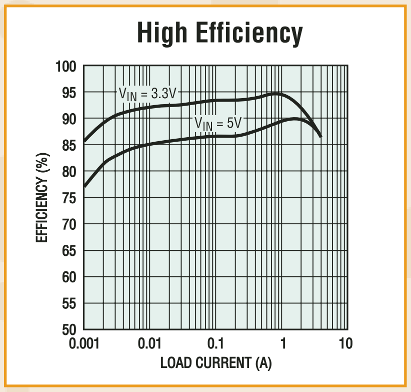
intersil
HIGH PERFORMANCE ANALOG

Spartan-3 FPGA Power Management Solutions









Linear Technology DC/DC converters power Spartan-3 devices.

Linear Technology offers a broad range of regulators to simplify the design and selection of DC/DC converters to power the Xilinx® Spartan™-3 family of FPGAs. Our converters range from low dropout linear regulators to sophisticated dual-output switching controllers that offer high efficiency and on-chip power supply tracking. Linear's free SwitcherCAD™ software allows quick and simple simulation of power supply designs. To download SwitcherCAD software, please visit *www.nuhorizons.com/linear*.

Single Output Regulators for Spartan-3 FPGAs — Core Voltage: 1.2V

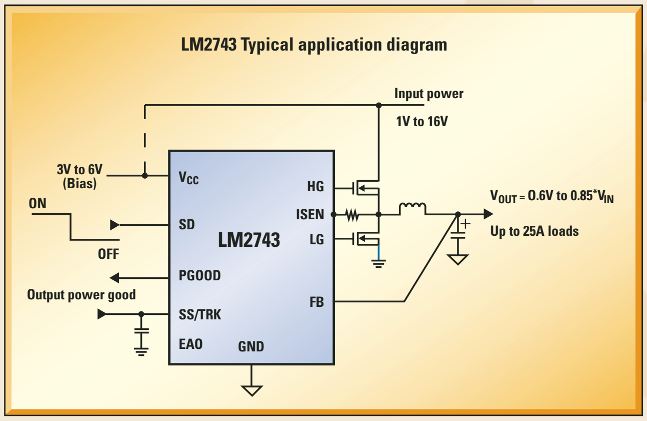
Input Supply	≤200 mA	≤500 mA	≤1A-1.5A	≤2A-2.5A	≤3A-5A	≤25A
1.8V	LTC3025 Linear	LT3021 Linear	LTC3026 Linear	LT3150 Linear	LTC3713 Controller	LTC3713 Controller
2.5V to 5V	LTC3405A Switcher	LTC3406 Switcher	LTC3412 Switcher	LTC3414 Switcher LTC3801 Controller	LTC3801 Controller LTC1773 Controller	LTC3713 Controller LTC3832 Controller LTC1778 Controller
≤12V	LT1616 Switcher	LT1976 Switcher LT1767 Switcher	LT1976 Switcher LT3431 Switcher LTC1778 Controller	LTC1771 Controller LTC1778 Controller	LTC1778 Controller	LTC1778 Controller
≤24V	LT1934 Switcher	LT1976 Switcher LT1767 Switcher	LT1976 Switcher LT3431 Switcher LTC1778 Controller	LT3431 Switcher LTC1778 Controller	LTC1778 Controller	LTC1778 Controller

NU HORIZONS ELECTRONICS CORP.
www.nuhorizons.com/linear

LINEAR TECHNOLOGY
www.linear.com

Versatile FPGA Power Solutions from National Semiconductor

LM2743/44 Low-Voltage Synchronous Buck Controllers



- Highly efficient 2A to 25A solution
- Input voltage from 1V to 16V
- Adjustable output voltage as low as 0.6V
- Power-good flag and output enable
- 1.5% reference accuracy over temperature
- Current limit without sense resistor
- Programmable softstart
- Switching frequency from 50 kHz to 1 MHz
- Available in small TSSOP-14 packaging

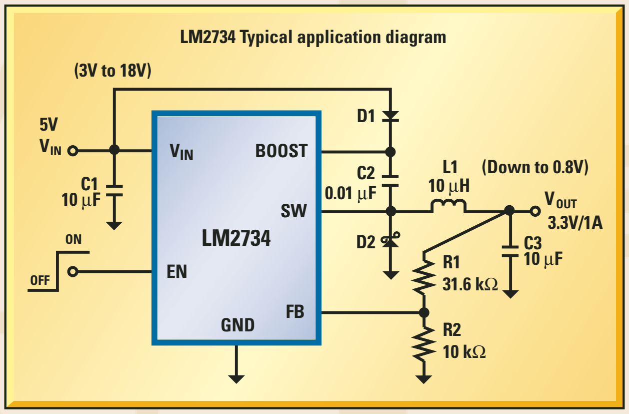
The LM2743 and LM2744 are high-speed synchronous buck regulator controllers that drive external MOSFETs to supply as much as 25A of current. They can provide simple down conversion to output voltages as low as 0.6V. Although the control sections of the ICs are rated for 3 to 6V, the driver sections are designed to accept input supply rails as high as 16V. The use of adaptive non-overlapping MOSFET gate drivers helps avoid potential shoot-through problems while maintaining high efficiency.

A wide range of switching frequencies from 50 kHz to 1 MHz gives Xilinx® FPGA and system power supply designers the flexibility to make better trade-offs between component size, cost, and efficiency. A versatile softstart and tracking pin allows ratiometric, coincidental, or offset tracking, which are critical for FPGA systems. An evaluation board is available.

For free samples, evaluation boards, or more information, visit *power.national.com* or call 1-800-272-9959.

National Semiconductor
The Sight & Sound of Information

LM2734/36 1A SOT-23 Buck Regulators



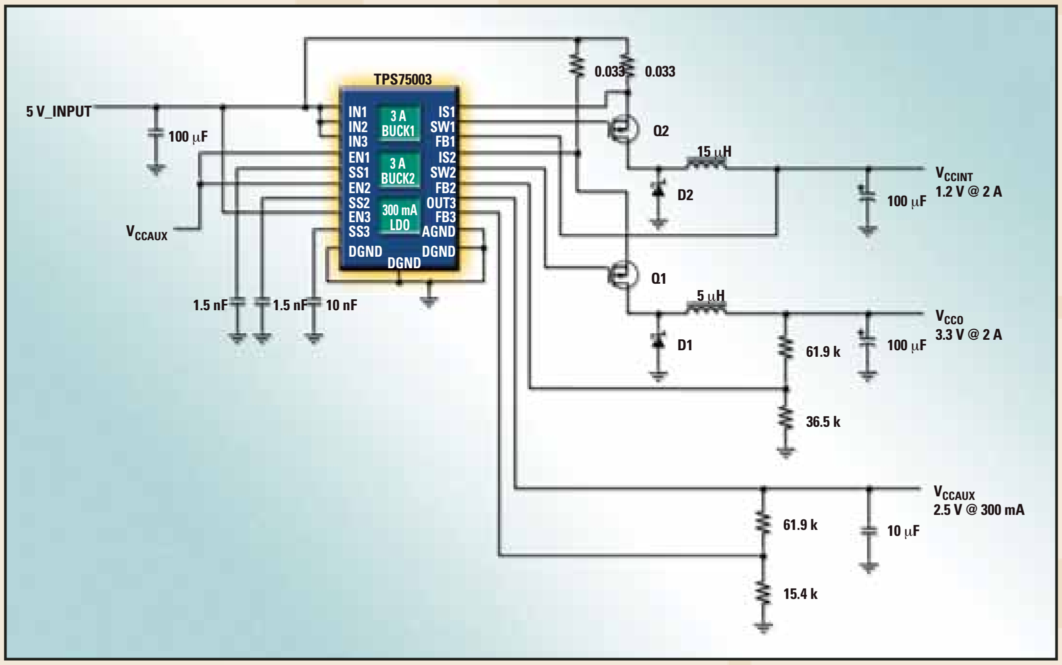
- Complete, easy-to-use switcher solution has the smallest footprint and highest power density in the industry
- Choice of switching frequencies allows designers to trade-off efficiency against solution size and EMI
- Current mode control improves phase margin, line regulation, and rejection of transients
- Internal softstart circuitry, cycle-by-cycle, thermal shutdown, and over-voltage protection

The LM2734 and LM2736 are monolithic, high-frequency (550 kHz and 1.6 MHz) PWM step-down DC/DC converters in tiny six-pin thin SOT-23 packaging. They provide local DC/DC conversion for Xilinx FPGAs with currents up to 1A.

Both regulators need no external compensation and are supported by WEBENCH®, National's online design tool. The ability to drive up to 1A loads with an internal 300 mΩ NMOS switch using state-of-the-art 0.5 µm BiCMOS technology results in the best power density available. The world-class control circuitry supports exceptionally high frequency conversion over the entire 3V to 20V input operating range, down to the minimum output voltage of 0.8V. Even though the operating frequencies are very high, efficiencies as high as 90% are easy to achieve. Additionally, a current-mode control loop provides fast transient response and accurate regulation in the smallest possible PCB area. An evaluation board is available.

Feature	LM2734	LM2736
Input Range	3.0V to 20V	3.0V to 18V
Output Load	1A	750 mA
Output Range	0.8V to 18V	1.25V to 16V
Internal References	0.8V, 2%	1.25V, 2%
Operating Frequency	550 kHz/1.6 MHz/3 MHz	

TI Power Solutions



Highly integrated triple supply powers Spartan-3 core, I/O, and VCCAUX rails.

Features

- Two 95% efficient, 3A buck controllers and one 300 mA LDO
- Adjustable output voltages:
  - From 1.2V for bucks
  - From 1.0V for LDO
- Input voltage range of 2.2V to 6.5V
- Independent softstart for all three power supplies
- LDO stable with small ceramic output capacitor
- Independent enable for each supply for flexible sequencing
- 4.5 mm x 3.5 mm x 0.9 mm 20-pin QFN package
- 1 ku price: $1.90

Applications

- DSL modems
- Set-top boxes
- Plasma TV display panels
- DVD players

The TPS75003 power management IC for Xilinx® Spartan™-II, Spartan-IIE, and Spartan-3 FPGAs integrates multiple functions to significantly reduce the number of external components required – and simply your designs. Combining increased design flexibility with cost-effective voltage conversion, the device includes programmable softstart for in-rush current control and independent enables for sequencing the three channels. The TPS75003 meets all Xilinx startup profile requirements, including monotonic ramp and minimum ramp times.

For more information about the complete line of TI power management solutions for Xilinx FPGAs, including a library of reference designs, schematics, and BOMs, visit *www.ti.com/xilinxfpga*. For questions, samples, or an evaluation module, e-mail *fpgasupport@list.ti.com*.

TEXAS INSTRUMENTS

Fast and Flexible

Embedded Processing

Xilinx and Avnet are pleased to bring you hands-on, FREE in-booth workshops at ESC Boston, so you can gain valuable embedded design experience first hand. Only Xilinx Platform FPGAs offer programmable flexibility with the optimal choice of integrated hard and/or soft processors. It's a fast and flexible solution for your processing needs.

Hands-On Workshops

- Explore the latest Virtex-4 FX PowerPC system, including the power of the Auxiliary Processor Unit (APU) Controller, and the award-winning capabilities of Xilinx Platform Studio *(9/13 – Noon, 2:30pm & 5pm, 9/14 – 11:15am & 1:45pm)*
- Configure the latest MicroBlaze soft processor system with an integrated Floating Point Unit (FPU) in minutes *(9/13 – 1:15pm & 3:45pm, 9/14 – 10am, 12:30pm & 3pm)*

Live Demonstrations

- Evaluate the new development kits for PowerPC and MicroBlaze
- Realize the possibilities of single and dual-core PowerPC systems with the breakthrough Virtex-4 FX FPGA
- Test drive the latest configurable MicroBlaze 32-bit processor with low-cost Spartan-3 FPGA

Join Xilinx and Avnet at Embedded Systems Conference Boston — Booth #501

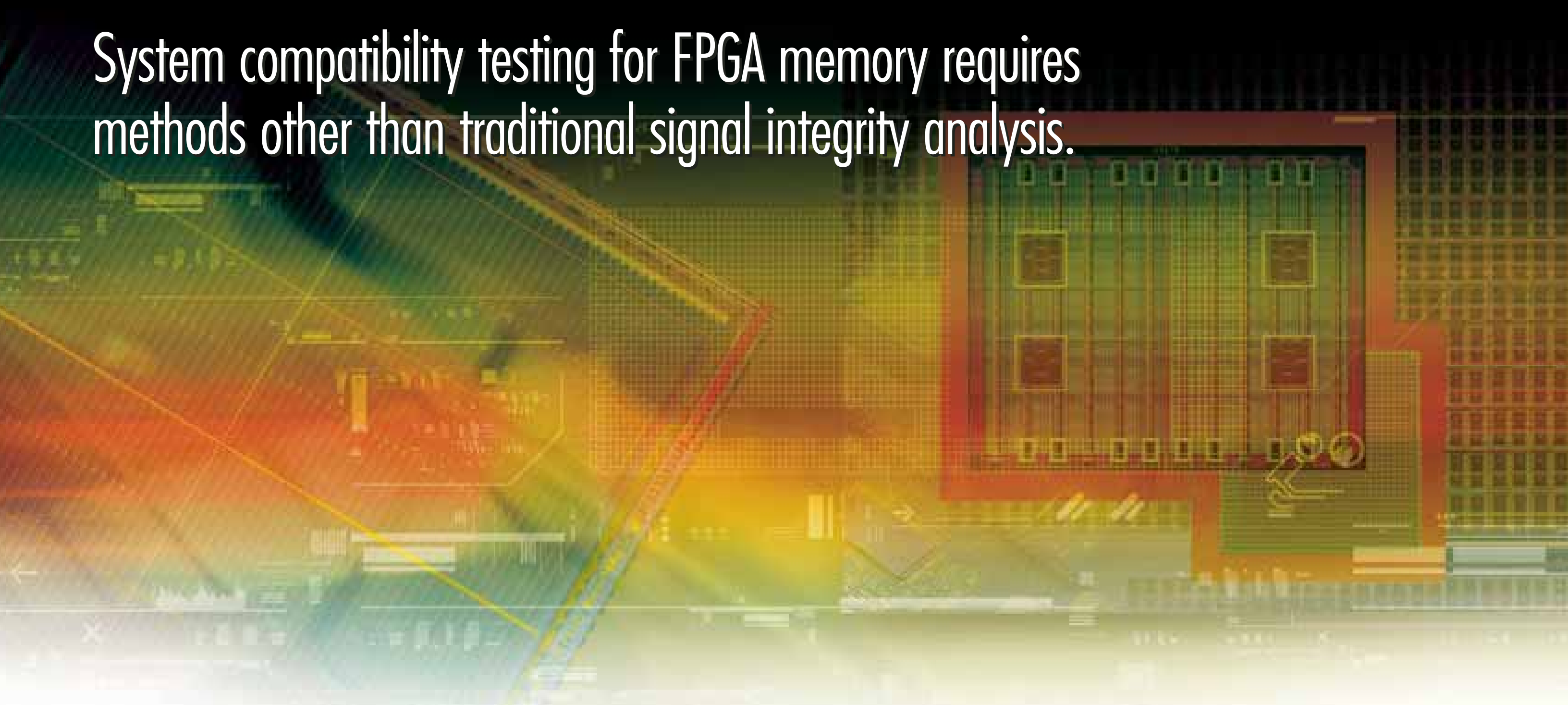
IN-DEPTH PRODUCT DEMONSTRATIONS | FREE HANDS-ON WORKSHOPS IN THE BOOTH | RECEPTION 5PM 9/13 TUESDAY

XILINX®

AVNET®

How to Detect Potential Memory Problems Early in FPGA Designs

System compatibility testing for FPGA memory requires methods other than traditional signal integrity analysis.



by Larry French
FAE Manager
Micron Semiconductor Products, Inc.
lfrench@micron.com

As a designer, you probably spend a significant amount of time simulating boards and building and testing prototypes. It is critical that the kinds of tests performed on these prototypes are effective in detecting problems that can occur in production or in the field.

DRAM or other memory combined in an FPGA system may require different test methodologies than an FPGA alone. Proper selection of memory design, test, and verification tools reduces engineering time and increases the probability of detecting potential problems. In this article, we'll discuss the best practices for thoroughly debugging a Xilinx® FPGA design that uses memory.

Memory Design, Testing, and Verification Tools

You can use many tools to simulate or debug a design. Table 1 lists the five essential tools for memory design. Note that this is not a complete list as it does not include thermal simulation tools; instead, it focuses only on those tools that you can use to validate the functionality and robustness of a design. Table 2 shows when these tools can be used most effectively.

This article focuses on the five phases of product development, as shown in Table 2:

- **Phase 1** – Design (no hardware, only simulation)
- **Phase 2** – Alpha (or Early) Prototype (design and hardware changes likely to occur before production)
- **Phase 3** – Beta Prototype (nearly "production-ready" system)
- **Phase 4** – Production
- **Phase 5** – Post-Production (in the form of memory upgrades or field replacements)

The Value of SI Testing

SI is not a panacea and should be used judiciously. SI should not be overused, although it frequently is. For very early or alpha prototypes, SI is a key tool for ensuring that your system is free of a number of memory problems, including:

- Ringing and overshoot/undershoot
- Timing violations, such as:
  - – Setup and hold time
  - – Slew rate (weakly driven or strongly driven signals)
  - – Setup/hold time (data, clock, and controls)

Tool	Example
Electrical Simulations	SPICE or IBIS
Behavioral Simulations	Verilog or VHDL
Signal Integrity	Oscilloscope and probes; possibly mixed-mode to allow for more accurate signal capture
Margin Testing	Guardband testing and four-corner testing by variation of voltage and temperature
Compatibility Testing	Functional software testing or system reboot test

Table 1 – Memory design, test, and verification tools

Tool	Design	Alpha Proto	Beta Proto	Production	Post-Prod
Simulation – Electrical	Essential	Very Valuable	Limited Value	Rarely Used	No Value
Simulation – Behavioral	Essential	Very Valuable	Limited Value	Rarely Used	No Value
Signal Integrity	Unavailable	Critical	Limited Value	Rarely Used	No Value
Margin Testing	Unavailable	Essential	Essential	Essential	Essential
Compatibility	Unavailable	Valuable	Essential	Essential	Essential

Table 2 – Tools for verifying memory functionality versus design phase

– Clock duty cycle and differential clock crossing (CK/CK#)

– Bus contention

By contrast, SI is not useful in the beta prototype phase unless there are changes to the board signals. (After all, each signal net is validated in the alpha prototype.) However, if a signal does change, you can use SI to ensure that no SI problems exist with the changed net(s). Rarely – if ever – is there a need for SI testing in production.

SI is commonly overused for testing because electrical engineers are comfortable looking at an oscilloscope and using the captures or photographs as documentation to show that a system was tested (Figure 1). Yet extensive experience at Micron Technology shows that much more effective tools exist for catching failures. In fact, our experience shows that SI cannot detect all types of system failures.

Limitations of SI Testing

SI testing has a number of fundamental limitations. First and foremost is the memory industry migration to fine-pitch ball-grid array (FBGA) packages. Without taking up valuable board real estate for probe pins, SI is difficult or impossible because there is no way to probe under the package.

Micron has taken several hundred thousand scope shots in our SI lab during memory qualification testing. Based on this extensive data, we concluded that system problems are most easily found with margin and compatibility testing. Although SI is useful in the alpha prototype phase, it should be replaced by these other tests during beta prototype and production.

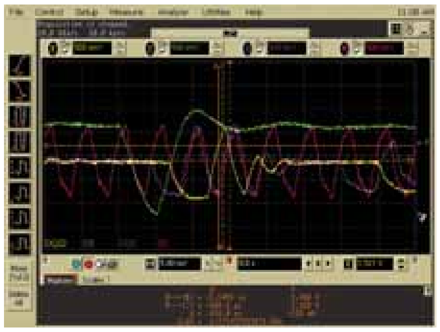


Figure 1 – Typical signal integrity shot from an oscilloscope

Here are some other results of our SI testing:

- SI did not find a single issue that was not identified by memory or system-level diagnostics. In other words, SI found the same failures as the other tests, thus duplicating the capabilities of margin testing and software testing.
- SI is time-consuming. Probing 64-bit or 72-bit data buses and taking scope shots requires a great deal of time.
- SI uses costly equipment. To gather accurate scope shots, you need high-cost oscilloscopes and probes.
- SI takes up valuable engineering resources. High-level engineering analysis is required to evaluate scope shots.
- SI does not find all errors. Margin and compatibility testing find errors that are not detectable by SI.

The best tests for finding FPGA/memory issues are margin and compatibility testing.

Margin Testing

Margin testing is used to evaluate how systems work under extreme temperatures and voltages. Many system parameters change with temperature/voltage, including slew rate, drive strength, and access time. Validation of a system at room temperature is not enough. Micron found that another benefit of margin testing is that it detects system problems that SI will not.

Four-corner testing is a best industry practice for margin testing. If a failure is

How Does the Logic Analyzer (or Mixed-Mode Analysis) Fit In?

You may have noticed that Table 1 does not include logic analyzers. Although it is rare to find a debug lab that does not include this tool as an integral part of its design and debug process, we will not discuss logic analyzers in this article. Because of the cost and time involved, they are rarely the first tool used to detect a failure or problem in a system. Logic analyzers are, however, invaluable in linking a problem, after it has been identified, to its root cause. Like signal integrity (SI), logic analyzers should be used after a problem has been detected.

…margin and compatibility testing will identify more marginalities or problems within a system than traditional methods such as SI.

going to occur during margin testing, it will likely occur at one of these points:

- Corner #1: high voltage, high temperature
- Corner #2: high voltage, low temperature
- Corner #3: low voltage, high temperature
- Corner #4: low voltage, low temperature

There is one caveat to this rule. During the alpha prototype, margin testing may not be of value because the design is still changing and the margin will be improved in the beta prototype. Once the system is nearly production-ready, you should perform extensive margin testing.

Compatibility Testing

Compatibility testing refers simply to the software tests that are run on a system. These can include BIOS, system operating software, end-user software, embedded software, and test programs. PCs are extremely programmable; therefore, you should run many different types of software tests.

In embedded systems where the FPGA acts like a processor, compatibility testing can also comprise a large number of tests. In other embedded applications where the DRAM has a dedicated purpose such as a FIFO or buffer, software testing by definition is limited to the final application. Thorough compatibility testing (along with margin testing) is one of the best ways to detect system-level issues or failures in all of these types of systems.

Given the programmable nature of Xilinx FPGAs, you might even consider a special FPGA memory test program. This program would only be used to run numerous test vectors (checkerboard, inversions) to and from the memory to validate the DRAM interface. It could easily be written to identify a bit error, address, or row – in contrast to the standard embedded program that might not identify any memory failures. This program could be run during margin testing. It would be especially interesting for embedded applications where the memory interface runs a very limited set of operations. Likely, this type of test would have more value than extensive SI testing of the final product.

Tests Not To Ignore

The following tests, if ignored, can lead to production and field problems that are subtle, hard to detect, and intermittent.

Power-Up Cycling

A good memory test plan should include several tests that are sometimes skipped and can lead to production or field problems. The first of these is power-up cycling. During power-up, a number of unique events occur, including the ramp-up of voltages and the JEDEC-standard DRAM initialization sequence. Best industry practices for testing PCs include power-up cycling tests to ensure that you catch intermittent power-up issues.

Two types of power-up cycling exist: cold- and warm-boot cycling. A cold boot occurs when a system has not been running and is at room temperature. A warm boot occurs after a system has been running for awhile and the internal temperature is stabilized. You should consider both tests to identify temperature-dependent problems.

Self-Refresh Testing

DRAM cells leak charge and must be refreshed often to ensure proper operation. Self-refresh is a key way to save system power when the memory is not used for long periods of time. It is critical that the memory controller provide the proper in-spec commands when entering and exiting self-refresh; otherwise, you could lose data.

Like power-up cycling, self-refresh cycling is a useful compatibility test. If an intermittent self-refresh enter or exit problem is present, repeated cycling can help detect it. Applications that do not use self-refresh should completely skip this test.

Sustaining Qualifications

One last area to consider is the test methodology for sustaining qualifications. That is, what tests should you perform to qualify a memory device once a system is in production? This type of testing is frequently performed to ensure that an adequate supply of components will be available for uninterrupted production.

During production a system is stable and unchanging. Our experience has shown that margin and compatibility testing are the key tests for sustaining qualifications. Because a system is stable, SI has little or no value.

Conclusion

In this article, our intent has been to encourage designers to rethink the way they test and validate FPGA and memory interfaces. Using smart test practices can result in an immediate reduction in engineering hours during memory qualifications. In addition, proper use of margin and compatibility testing will identify more marginalities or problems within a system than traditional methods such as SI. No "one-size-fits-all" test methodology exists, so you should identify the test methodology that is most effective for your designs.

For more detailed information on testing memory, see Micron's latest DesignLine article, "Understanding the Value of Signal Integrity," on our website, *www.micron.com.*

Losing Less from Lossy Lines

Managing signal loss when designing with RocketIO transceivers.

by Bill Hargin
Product Manager, HyperLynx
Mentor Graphics
bill_hargin@mentor.com

A few months ago, I needed a few new hard drives: two desktop drives for home and a bigger 120 GB drive for my laptop, which was out of space. Browsing through a Seattle-area computer store, I compared the prices of Ultra ATA drives (older, parallel architectures) and Serial ATA (SATA) disk drives, discovering that the higher throughput of the SATA drives had resulted in price points that were twice those of the Ultra ATA drives. To me, this presented a crisp picture of the economic drivers behind the SERDES technology wave: higher throughput commanding a higher price, yet lower manufacturing costs.

You would think hardware designers would be making a mad dash toward serial design. However, I find that the "dash" toward SERDES seems to hover around 20 percent. Pondering this, I have come to believe that there are three primary reasons for the reticence among the remaining 80 percent:

1. Speed. Many applications are not pushing the speed envelope.
2. Resistance to change. Even innovative engineers are creatures of habit.
3. Real or perceived technical hurdles. SERDES design requires a different approach than wide, parallel bus design.

In this article, I'll address the third reason in detail. Although my focus is on Xilinx® Virtex™-II Pro RocketIO™ technology, the information would apply to any serial interface, including RocketIO transceivers in Virtex-4 devices.

Line Loss

An ideal lossless transmission line assumes that a signal propagates down the line with no energy loss. In other words, if a 1.0V signal with a 1.0 ns rise time enters one end of the line, the same 1.0V and 1.0 ns signal will come out the far end.

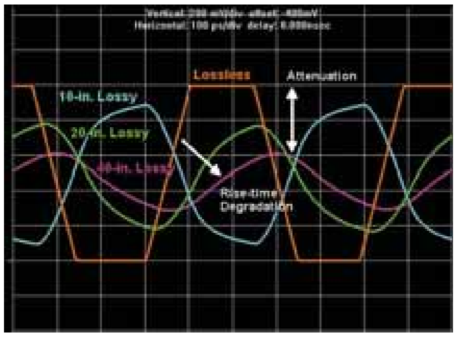


Figure 1 – Reciever waveforms for a lossless line, as well as 10, 20, and 40 in lines, showing both attenuation and rise-time degradation for a 2 GHz signal. (Simulated with Mentor Graphics HyperLynx.)

This is a good approximation when signal rise times are on the order of 1.0 ns (or slower), and with trace lengths of 10 in (or shorter). However, as rise times trend toward 100 ps and lengths get significantly longer (as in a backplane), the lossy effects of transmission lines begin to influence signal quality dramatically. As Figure 1 shows, these effects – both attenuation and rise-time degradation – vary directly with length at higher frequencies.

To describe and accurately predict the behavior of real interconnects, two important mechanisms that absorb energy from the signal must be modeled:

- Resistive loss. From DC through frequencies up to a few megahertz, the current in a trace moves through the entire cross-sectional area of the trace. At higher frequencies, however, current flows along the perimeter of a line rather than uniformly across the entire cross-section. As a result, the series resistance of the signal and return path conductors increases with the square root of frequency as the effective cross-section of the interconnect path is reduced. Resistive loss is also referred to as "skin effect." But no matter what you call it, it is the same phenomenon, and something you should be concerned about at high frequencies.

- Dielectric loss. The second important loss mechanism is dielectric loss, which is simply the conversion of electrical energy from the alternating electric field into heat. Dielectric loss, often specified in decibels per meter, increases with frequency and varies inversely with a material's "loss tangent" – a function of the material's resin type and molecular structure. Depending on resin content, "vanilla" FR-4 has a loss tangent ranging from 0.02-0.03. Lower loss tangent equates to more of the output signal getting to its destination, as well as higher material costs compared to FR-4. GETEK, for example, has a loss tangent of 0.012. Nelco 4000-13 is 0.01. And the loss tangent for Rogers 4003 is as low as 0.0027. For an actual design, you will want to discuss the tradeoffs with your board vendor.

Loss, Jitter, and ISI

I've often heard engineers use the terms inter-symbol interference (ISI), jitter, and loss to refer to the same thing: the unknown cause of a less-than-optimal signal or bit-stream. In fact, these are different phenomena.

Random jitter is used to describe random events that result in a delay between the expected and actual signal transition. The distribution of random effects follows a classic Gaussian distribution, where the results vary wider (in time) as more data is observed. This data is typically provided by the driver manufacturer.

Deterministic jitter encompasses the list of systematic interconnect effects that will reoccur if you repeat the same stimulus. Dielectric and resistive loss, as well as crosstalk, reflections, via parasitics, return-path discontinuities, and any systematic aspect of an interconnect design contribute to deterministic jitter.

If the combined effects of random and deterministic jitter are significant enough, ISI will result, indicating that bit distinctions have become "blurry" at the receiver. This becomes particularly serious when rise-time degradation becomes comparable to the bit period of the signal. As a result, the shape of the received waveform will depend on the prior bit pattern (ISI).

These effects are best modeled using an oscilloscope that supports eye diagram analysis. Eye diagrams provide a visual display of the signal quality over many bit transitions with both deterministic and random jitter serving to close the "eye."

At a glance, an eye diagram will show if an interconnect is acceptable. Pass/fail criteria are often specified by an eye mask, shown as a blue hexagon on the left-hand side of Figure 2. Eye masks, which conform to the various SERDES specifications, define minimum and maximum keep-out regions where proper bit transitions should not appear for proper receiver interpretation of the driver's intent.

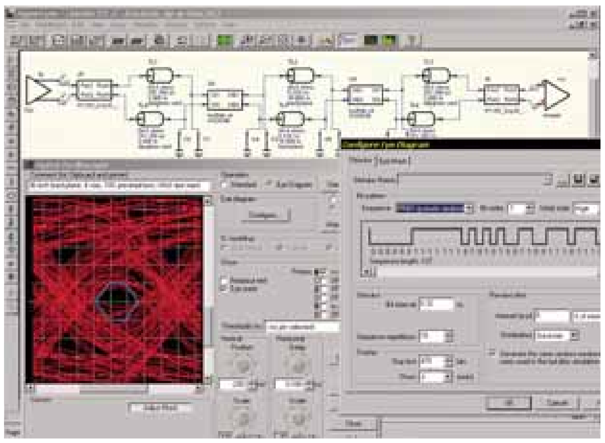


Figure 2 – The eye shows encroachment on the XAUI eye mask when using 10% pre-emphasis for a 36 in FR-4 transmission path with four vias and two Teradyne VHDM-HSD5 connectors (simulated with HyperLynx LineSim GHz).

Typically used bit patterns, or "pulse trains," include 8b/10b encoding and PRBS (pseudo-random bit stimulus). The 8b/10b data transmission scheme is considered ideal for high-speed local area networks and computer links. Realistic, long character sequences eventually hit some kind of worst-case history, but this can take a long time. A PRBS stimulus, as shown on the right-hand side of Figure 2, provides a means to force as much "action" on a serial data path as possible, in the smallest number of cycles. It is "pseudo-random" because it actually repeats after a pre-determined "n" number of bits.

Losing Less from Lossy Lines

Assuming that you have followed good routing rules (for example, achieving a consistent differential impedance of 100 ohms and avoiding excessive routing skew), there are five major ways to mitigate loss:

1. Reduce resistive loss by widening traces. Because resistive loss or "skin effects" are the result of a reduced cross-sectional area in a trace, wider traces result in a larger cross-sectional area, so the percent reduction due to resistive loss becomes smaller.

2. Reduce dielectric loss by shortening lines. Dielectric loss is a function of the material used and the length over which a signal is transmitted. Shortening the overall interconnect length (which may not be possible in many systems) can be an effective means of eliminating loss.

3. Reduce dielectric loss by employing lower loss tangent dielectrics. To reduce dielectric loss, more expensive materials – with lower loss tangents – can be considered, perhaps after exhausting less-expensive approaches.

4. Increase driver pre-emphasis. Boost the initial voltage level of each edge to compensate for high-frequency loss. (The trade-off here is additional power consumption.)

5. Equalization at the receiver. The incoming signal's lower frequency components are intentionally attenuated to artificially balance between high- and low-frequency components. The result is then amplified, with equalization of the low- and high-frequency signal components.

Although it is possible to estimate rise-time degradation and loss based on rules of thumb, the only way to get a realistic prediction of the impact from losses is to use a software simulator with the capability of simulating lossy lines. The HyperLynx Virtex-II Pro RocketIO Design Kit, along with HyperLynx simulation software, will enable you to simulate the preceding effects with Xilinx RocketIO technology.

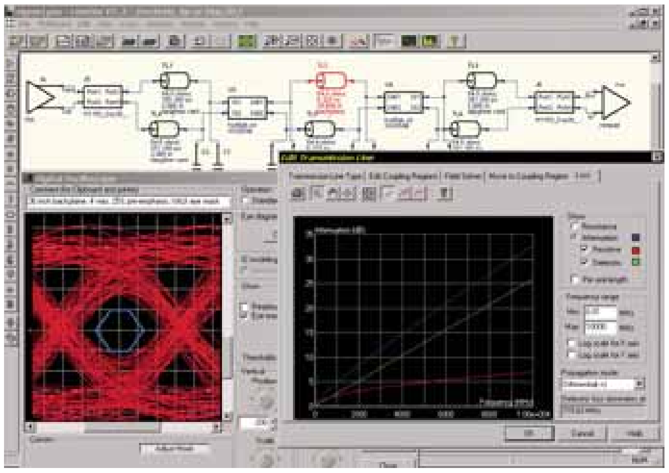


Figure 3 – Results with pre-emphasis bumped to 25% show significant improvement. Also shown is a graph of resistive and dielectric loss (dB) as a function of frequency (MHz) for the backplane (simulated with HyperLynx LineSim GHz).

The HyperLynx RocketIO Design Kit

In an effort to make multi-gigabit interconnect implementation as painless as possible, Xilinx and Mentor Graphics have teamed up to provide the RocketIO Design Kit for HyperLynx. Editable, pre-configured circuits in the kit are ready to simulate for both chip-to-chip applications and PCB backplanes – including connectors and pre-configured differential striplines.

The FR-4 traces used are a differential pair of centered striplines, 12 mils wide and 20 mils apart, with a 50 ohm characteristic impedance.

Figure 2 shows simulation results for the Xilinx backplane example with a 36 in transmission path and four vias. The eye in the figure shows encroachment on the XAUI (10 Gb Extended Attachment Unit Interface) eye mask when using 10 percent pre-emphasis. Here, we can clearly see the effects of ISI at the receiver.

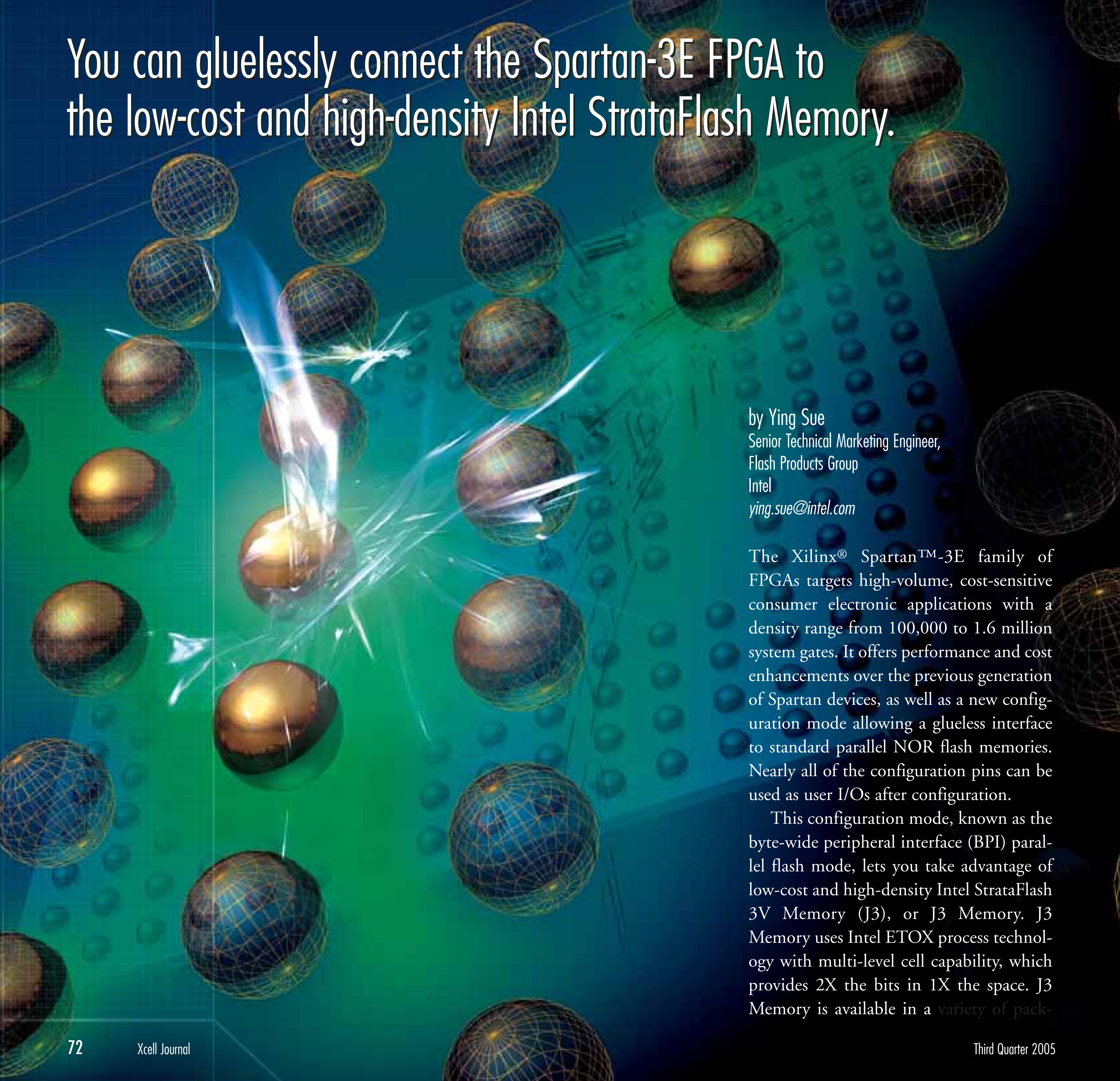
Although you could engage any of the five primary loss-reduction mechanisms to open up the eye, let's assume that the 36 in backplane is a design requirement and that the four vias are inevitable. One of the remaining options is pre-emphasis, which can be bumped up to 20, 25, or 33 percent. Figure 3 shows the results at 25 percent pre-emphasis: the eye is wide open, which is great. With the wide margin around the XAUI mask's internal keep-out region, I would want to simulate this again at 20 percent pre-emphasis in an attempt to conserve power.

Conclusion

Although I simulated only one solution here, any permutation of the five ways to deal with loss is possible. Moreover, you could also examine non-ideal routing effects, including simulation of the impacts of differential length skew. With the availability of helpful simulation tools like HyperLynx, you don't need to fabricate prototypes – wondering whether you've over-designed or under-designed, spent too much or spent too little.

In a future extension of this theme, we'll revisit the Xilinx SIS Kit backplane design and examine the impact of the new equalization technology in the Virtex-4 implementation of RocketIO transceivers.

The HyperLynx RocketIO Design Kit is available from *www.mentor.com/hyperlynx*.



Connecting Intel StrataFlash Memory to Spartan-3E FPGAs

You can gluelessly connect the Spartan-3E FPGA to the low-cost and high-density Intel StrataFlash Memory.

by Ying Sue
Senior Technical Marketing Engineer,
Flash Products Group
Intel
ying.sue@intel.com

The Xilinx® Spartan™-3E family of FPGAs targets high-volume, cost-sensitive consumer electronic applications with a density range from 100,000 to 1.6 million system gates. It offers performance and cost enhancements over the previous generation of Spartan devices, as well as a new configuration mode allowing a glueless interface to standard parallel NOR flash memories. Nearly all of the configuration pins can be used as user I/Os after configuration.

This configuration mode, known as the byte-wide peripheral interface (BPI) parallel flash mode, lets you take advantage of low-cost and high-density Intel StrataFlash 3V Memory (J3), or J3 Memory. J3 Memory uses Intel ETOX process technology with multi-level cell capability, which provides 2X the bits in 1X the space. J3 Memory is available in a variety of packages and densities for increased flexibility and can be gluelessly connected to the Spartan-3E FPGA to store any of the following:

- Bitstreams for one or more FPGAs
- Boot code, parameters, and data for soft CPU cores in the FPGA
- Multiple bitstreams for the same FPGA utilizing the MultiBoot feature of the Spartan-3E device

Figure 1 shows how dual bitstreams (for MultiBoot) and code/data storage can coexist in a J3 Memory device.

Design Notes

Figure 2 illustrates the connection between two Spartan-3E FPGAs and a J3 Memory flash device in a 3.3V environment. In this section, we'll examine key design considerations, such as power sequence, reset, hot-swap, flash content protection, and x8/x16 mode toggling.

Power-Up Sequence

Three supply voltages are required to support the Spartan-3E device and J3 Memory in a 3.3V application:

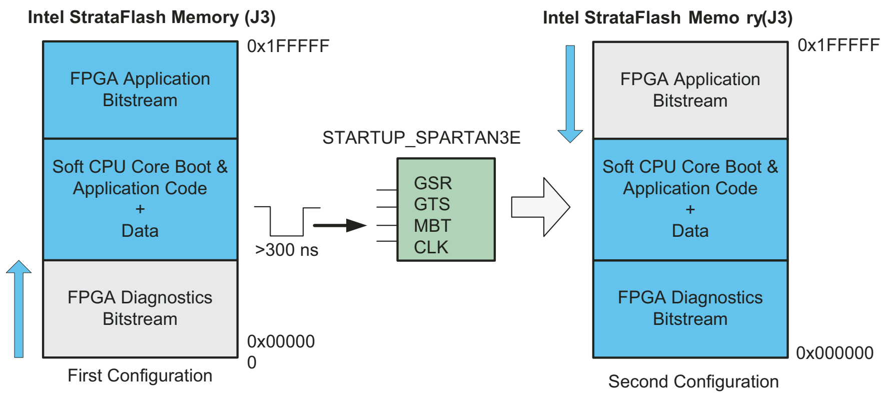
- 3.3V: connected to the VCC and VCCQ supplies of J3 Memory, as well as to the VCCO_1 and VCCO_2 supplies of the FPGA
- 2.5V: connected to the VCCAUX supply of the FPGA
- 1.2V: connected to the VCCINT supply of the FPGA

VCCO_0 and VCCO_3 supplies of the FPGA can be 3.3V, 2.5V, 1.8V, 1.5V, or 1.2V, as required by the application.

The Spartan-3E FPGA datasheet describes the power-on precaution in the serial flash mode (SPI mode), where the FPGA might be reading from the flash device before the flash memory is ready. The same consideration applies to parallel flash devices, because typical flash devices require some time (60 µs for J3 Memory) to complete internal initialization after the voltage reaches a nominal level (2.7V for J3 Memory).

If the 2.5V and 1.2V are valid and the 3.3V reaches 0.4V to 1.0V (a voltage level below the minimum operating voltage of a flash device such as J3 Memory), the FPGA starts its programming sequence before the flash device is ready. Either of two scenarios might occur:

- 3.3V is valid (reaching 2.7V) before 2.5V and 1.2V reach their minimum required voltage. This scenario typically does not cause issues because the flash device (such as J3 Memory) is ready for reading when the FPGA starts its programming sequence.
- 3.3V is valid after 2.5V and 1.2V. To work around this scenario, you can typically use a 3.3V voltage monitor to hold the PROG_B or INIT_B pin


Intel StrataFlash Memory (J3)
0x1FFFFF
FPGA Application Bitstream
Soft CPU Core Boot & Application Code + Data
FPGA Diagnostics Bitstream
0x00000 0
First Configuration
>300 ns
STARTUP_SPARTAN3E
GSR
GTS
MBT
CLK
Intel StrataFlash Memo ry(J3)
0x1FFFFF
FPGA Application Bitstream
Soft CPU Core Boot & Application Code + Data
FPGA Diagnostics Bitstream
0x000000
Second Configuration

For more information on using the Xilinx Spartan-3E FPGA in this setup, see Xilinx datasheet DS312-2

Figure 1 – Multiple bitstream and code/data storage in flash devices (B4980)

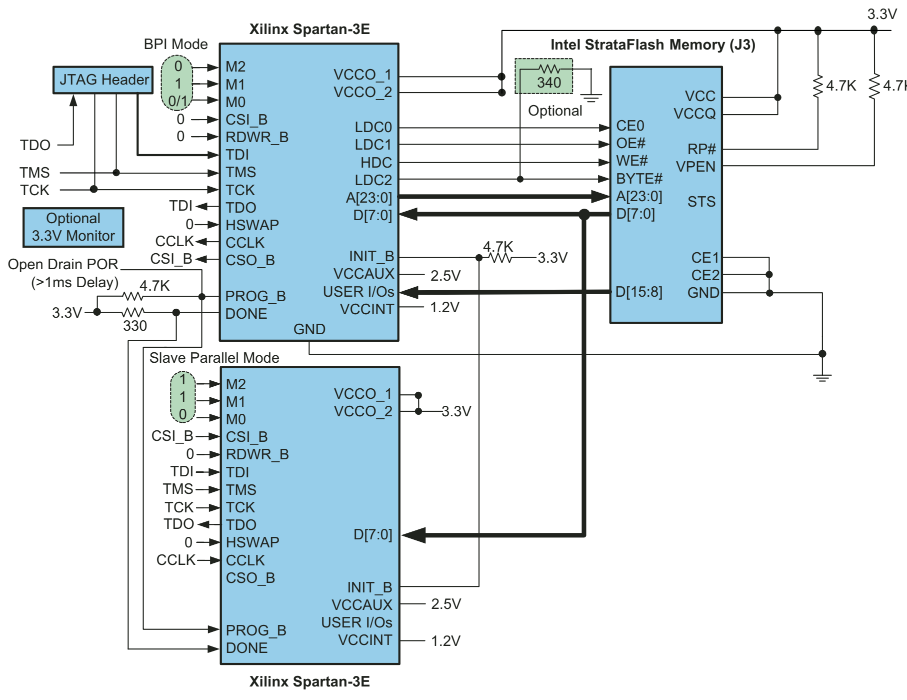


Figure 2 – A 2x Xilinx Spartan-3E FPGA to Intel StrataFlash memory (J3) connection (B4981)

– low for at least 1 ms after the 3.3V power supply reaches the minimum operating threshold.

Reset

On J3 Memory, the RP# pin is the reset input. In reset, the internal flash circuitry is disabled and outputs are placed in a high-impedance state. The RP# pin places a flash device (such as J3 Memory) into asynchronous page mode (read-array) when a minimum low pulse (35 µs) is applied.

The following connection options are available for the RP# pin:

- Connecting RP# to 3.3V. This connection means that the flash device (such as J3 Memory) does not reset until power is cycled. This connection is suitable for applications where the FPGA is programmed only on power up and is not reprogrammed without power cycling. This connection is also applicable if you are certain that the risk of putting the flash device into non-read-array mode while the PROG_B pin is toggled is minimal.
- Connecting RP# to System Reset. When this connection is made, you must ensure that the PROG_B pin is not driven from low to high before RP# goes high, so that the FPGA does not start reading from the flash device before it comes out of reset. For J3 Memory, this duration is 150 ns to 210 ns after RP# goes high, under typical operating voltages.
- Connecting RP# and PROG_B together. When an FPGA reprogram is issued, this connection automatically resets the flash device to read-array mode. The output of a voltage monitor can be used to drive both inputs, but only under the following conditions:
  - The minimum reset pulse width for the flash device (such as J3 Memory) is met. The PROG_B pin on the FPGA requires only 0.3 µs low pulse, but J3 Memory requires 35 µs.
  - The TPL delay, from the time the PROG_B pin on the FPGA transitions high until INIT_B transitions high, exceeds the typical range of 150 ns to 210 ns (the R5 parameter in the Intel StrataFlash Memory [J3] datasheet). According to the Xilinx datasheet, the TPL minimum is approximately 2 ms.

HSWAP

To prevent inadvertent access to the flash memory during power up of the FPGA – for example, the flash memory is accidentally put into any non-read-array mode – the HSWAP pin can be set to 0. This setting enables internal pull-up resistors that pull LDC0 (CE#), LDC1 (OE#), and HDC (WE#) to high. Special consideration must be given to the LDC2 pin.

If the application requires the HSWAP pin to be pulled high, then external pull-up resistors are required on the LDC0, LDC1, and HDC outputs. A pull-down resistor of 4.7K is required on the LDC2 (BYTE#) pin.

x8 Model-Only Operation

In BPI mode (M[2:0] = 0b010 or 0b011), the FPGA powers up in x8 mode, and drives the LDC2 pin (connected to the BYTE# pin of the flash device) low throughout the configuration period. If the application does not require x16 mode, tie J3 Memory's BYTE# pin low and do not connect the LDC2 output to J3 Memory.

Toggling Between x8 and x16 Modes

If the LDC2 pin is connected to the BYTE# pin of the flash device, the FPGA can drive the LDC2 pin high to switch the flash device from x8 mode to x16 mode after configuration.

When toggling between the byte (x8) and word (x16) modes, the least significant address location and mode switching delay must be considered. On a J3 Memory device the A0 address line selects the byte location.

The switching latency between x8/x16 modes is 1000 ns (the R12 tFLQV/FHQV parameter in the Intel datasheet) from the time the LDC2 pin changes logic state until valid data can be output from the flash device. This latency must be taken into consideration, together with the HSWAP setting, because HSWAP controls the ability to tri-state LDC2 upon power up, in the following two scenarios:

- HSWAP is tied high (internal pull-ups are disabled). Connect a 4.7K pull-down resistor to LDC2. This pin is pulled low upon power up and remains low during configuration. J3 Memory is in x8 mode when the FPGA starts to load its configuration bitstream. LDC1 and LDC0 must have pull-up resistors so that the flash device is not accidentally selected or put into non-read-array mode during power up.
- HSWAP is tied low. In this scenario, the LDC2 pin is pulled high through an internal pull-up resistor upon power up and then driven low at the same time as LDC0 (CE#), LDC1 (OE#). The FPGA starts reading configuration data in less than 1000 ns, a period required for the flash device to switch modes. Three workarounds exist:
  - Workaround #1, as shown in Figure 1, requires a 340 ohm pull-down resistor on the LDC2 output to overcome the internal pull-up resistor and ensure that the flash device is in x8 mode when the FPGA starts to read its configuration data. The 340 ohm value is based on Spartan-3 data and can be increased as more characterization data on the internal pull-up of the FPGA becomes available. The downside to this workaround is that a large output buffer must be used to overcome the strong pull-down resistor when an application requires switching the LDC2 pin high to use the x16 mode.
  - Workaround #2 takes advantage of the fact that the FPGA requires an initialization sequence before it starts configuration. When the flash is switching from x8 to x16 mode, invalid data is present on the bus, which prevents the FPGA from seeing this sequence until the flash device completes its mode switching. If the FPGA device does not

see the start sequence, it continues to increment the address, step through the remaining valid addresses, and then wrap around to address 0 until it finds the right sequence. This can result in a prolonged bitstream load time (the bitstream is eventually loaded).

– Workaround #3 requires prefixing the FPGA bitstream with as many as 16 bytes of 0xFF dummy data. This prefix helps the FPGA device find its start sequence when it reaches the address for byte 17, at which time the flash would have completed its mode switching. This workaround involves Xilinx modifying its bitstream generation code and slightly increases the bitstream size. Contact Xilinx for the availability of this workaround.

Other Design Considerations

- Addressing. The Spartan-3E FPGA can address as much as 256 Mb of flash memory. When populated, a lower density flash device can optionally be stuffed with a higher density flash device. In this case, the unused address pins on the lower density flash device can be safely connected to their corresponding address pins on the FPGA. These unused pins are no-connects on the flash device.
- ConfigRate setting. The initial access time for J3 Memory ranges from 110 ns to 150 ns, depending on density. Set the FPGA's maximum CCLK configuation rate (ConfigRate) setting appropriately.
- Flash content protection. On J3 Memory, the VPEN input can be used to protect the flash memory content. When the VPEN input is driven below Vpenlk (2.2V), the flash content cannot be altered. If unused, this input can be tied to 3.3V. Alternatively, it can be connected to a pin on the FPGA, to allow/disallow flash content alteration.
- Power supply decoupling. When the flash device (such as J3 Memory) is enabled, many internal conditions change. Circuits are energized, charge pumps are switched on, and internal voltage nodes are ramped. Such internal activities produce transient signals. To minimize these effects, a 0.1 uF ceramic capacitor is required across each VCC/VSS and VCCQ signal. Place capacitors as close as possible to the device connections.
- FPGA configuration pin re-use. Most of the pins driving the flash device (such as J3 Memory) can be used as general-purpose I/Os. However, do not reuse the following pins:

  – LDC0: Flash chipset enable

  – LDC2: Flash byte/word mode control
- Execute-In-Place (XIP). Intel offers a suite of software that supports XIP, where code is executed directly out of the flash device (such as J3 Memory) to reduce the external RAM requirement and power consumption. Software designs for a soft CPU core can use such collateral. If the XIP usage model is deployed for the CPU code and data, then the VPEN input must be pulled high. Also, memory blocks containing flash configuration code and boot code can be individually locked using a software command to prevent accidental programming or erasure.

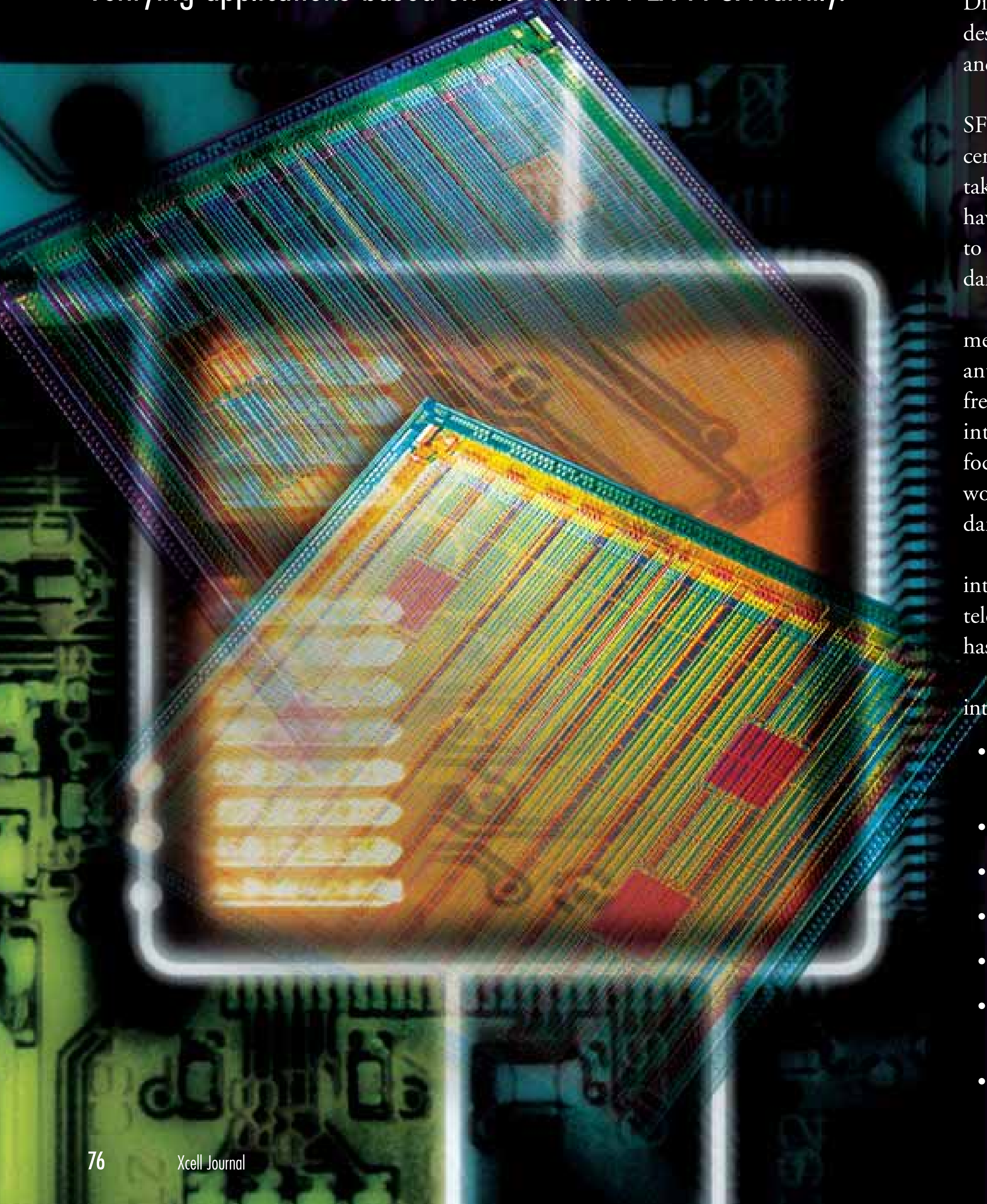
Conclusion

The addition of a BPI configuration mode to the Spartan-3E device enables consolidation of FPGA bitstreams and boot/application code into standard NOR flash memory. As a result, you can take advantage of the low-cost nature and wide density ranges of both the Spartan-3E FPGA and the Intel StrataFlash J3 NOR memory to cost-effectively target a wider range of high-volume applications not achievable with previous generation Spartan FPGAs.

For more information about J3 Memory, visit *www.intel.com/design/flcomp/prodbref/298044.htm*. The Xilinx Spartan-3E FPGA Family Complete Datasheet is available at *http://direct.xilinx.com/bvdocs/publications/ds312.pdf*.

Virtex-4 Source-Synchronous Interfaces Tool Kit

The Virtex-4 ML450 Source-Synchronous Interfaces Tool Kit provides a complete development platform for designing and verifying applications based on the Virtex-4 LX FPGA family.



by David Naylor
Sr. Product Applications Engineer
Xilinx, Inc.
david.naylor@xilinx.com

Many of today's telecom and networking systems use high-bandwidth interfaces based on low voltage differential signaling (LVDS) or other differential I/O standards. Differential I/O standards simplify system design by improving system performance and signal integrity.

Protocols based on I/Os such as SPI-4.2, SFI, RapidIO, and HyperTransport are central to leading-edge system design. To take advantage of these technologies, you have to work through multiple challenges to ensure device interoperability and standards compliance.

Xilinx® provides Virtex™-4 development boards as well as standards-compliant intellectual property (IP) cores and free reference designs for major system interface protocols. This allows you to focus on user application design and not worry about interoperability and standards compliance.

With the Virtex-4 source-synchronous interfaces tool kit, designing networking, telecom, servers, and computing systems has never been faster or easier.

The Virtex-4 ML450 source-synchronous interfaces tool kit includes the following:

- Virtex-4 ML450 development board (XC4VLX25FF668 FPGA)
- 5V/6.5 AC/DC power supply
- Country-specific power supply line cord
- RS232 serial cable, DB9-F to DB9-F
- Four clock module daughter boards
- Two sets of “blue ribbon” loopback cables for LVDS testing
- Documentation and reference design CD-ROM

The Virtex-4 ML450 Development Board

The main component in the Virtex-4 ML450 source-synchronous interfaces tool kit is the ML450 development board (Figure 1). Featuring a Virtex-4 XC4VLX25FF668 FPGA coupled to high-speed connectors, this board supports the development of high-speed interface designs using several popular protocols.

The ML450 demonstration board is a simple board providing several connector interfaces to the FPGA. The board comprises basic support circuitry, including power regulators, a serial RS232 connector, a small graphics LCD, a few user push buttons and LEDs, and a DDR-1 SDRAM. These simple peripherals allow a PC to communicate with the FPGA and also provide basic input and output indicators. Configuration is enabled via a JTAG connector, or you can use a System ACE™ CompactFlash card for bitstream storage and loading.

We designed the board to demonstrate the high-speed I/O capability of the Virtex-4 FPGA. Eighty differential channels are pinned out to four Samtec connectors. Forty pairs each are routed to two connectors on either side of the FPGA, allowing you to designate "transmit" and "receive" interfaces. Two "mini coax" flat cables are provided with the kit, enabling loopback of high-speed data between transmit and receive connectors.

In addition to these differential signals, another 32 pairs are routed to a HyperTransport Consortium DUT (device under test)-compliant connector, allowing for the development of an interface to other HyperTransport-based boards.

The ML450 evaluation platform supports a wide range of communications standards, including SFI-4. Figure 2 shows the user interface of an SFI-4 demo running on an ML450 board. The user interface includes a bit error rate tester (BERT) that measures the integrity of the data received from 16 LVDS transmitters. You can select from several pseudo-random bit sequences to simulate data, and error counters on the user interface maintain a running count of all bit errors that occur during transmission. The BERT also keeps track of which channels are receiving errors to provide further troubleshooting visibility in a multi-channel design.

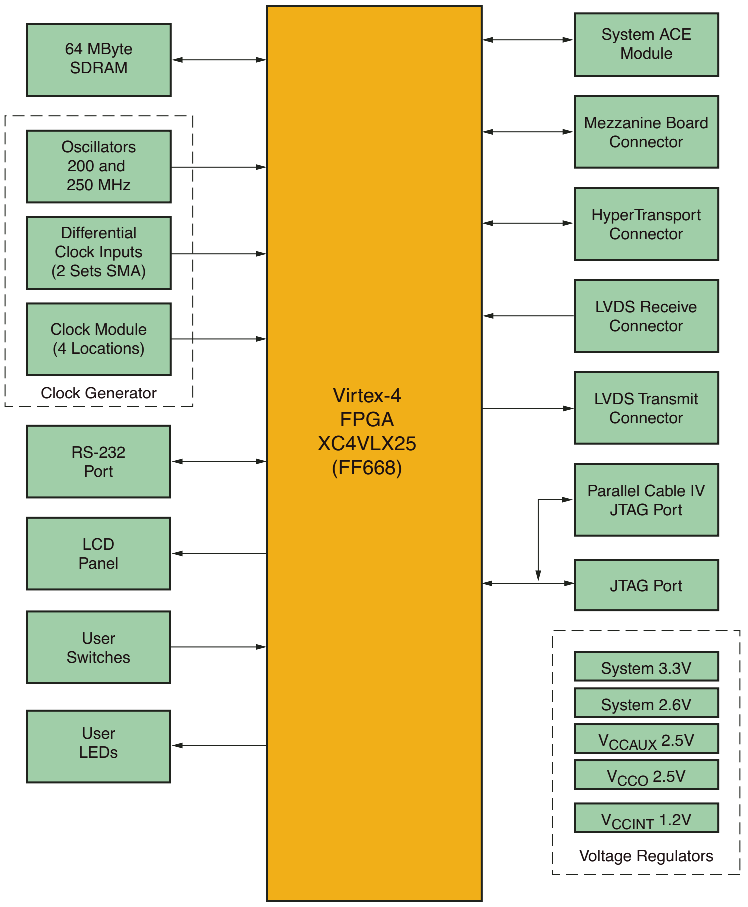


Figure 1 – Virtex-4 ML450 networking interfaces development board

In terms of performance, the ML450 platform supports a 16-channel single data rate design (SDR) running up to 700 MHz (Figure 3) and a 16-channel double data rate (DDR) design running up to 500 MHz (Figure 4). The transmission medium of the 16 channels consists of two Samtec connectors, a 12-inch ribbon cable, and 10 inches of FR4.

The ML450 also provides the ability to command the supply rails of the FPGA to assume ±5% of their nominal values. This feature provides a convenient way for you to stress your design and identify marginal behavior. The voltages are controlled through the user interface by simply moving a slider button to the left or right. In Figure 2, VccAux is raised to +5%, while the other supplies remain nominal.

A small daughtercard that plugs onto the board provides the high-speed clock used by the SFI-4 design. This daughtercard generates a programmable LVDS clock between 200 MHz and 700 MHz, eliminating the need for a cumbersome bench-top pulse generator.

Board Features

The ML450 development board includes the following:

- XC4VLX25FF668 FPGA
- Eight clock sources:
  - 200 MHz and 250 MHz on-board oscillators
  - Two sets of SMA differential clock input connectors
  - Four Samtec clock module connectors
- One 64 x 128 pixel LCD
- One DB9-M RS232 port
- A System ACE CompactFlash configuration controller that allows storing and downloading of as many as eight FPGA configuration image files

- Four LVDS Samtec connectors (a total of 40 input channels and 40 output channels)
- One HyperTransport connector (HyperTransport Consortium DUT connector-compliant)
- On-board power regulators with ± 5% output margin test capabilities

Clock Generation

The clock generation section of the ML450 development board provides all necessary clocks for the Virtex-4 FPGA. Eight clock sources are provided as follows:

- Epson EG2121CA 2.5V 250 MHz differential low-voltage positive emitter-coupled logic (LVPECL) oscillator
- Epson EG2121CA 2.5V 200 MHz differential LVPECL oscillator
- Two differential SMA clock inputs
- Four Samtec user clock sockets

The differential SMA clock inputs are connected to the global clock inputs of the FPGA, accessing the upper and lower halves. An on-board 200 MHz oscillator calibrates the I/O delay, and an on-board 250 MHz oscillator is provided for use with the HyperTransport IP.

The four clock modules included in the kit are:

- Type A: direct balanced differential SMA input
- Type B: Epson EG2121CA 2.5V 400 MHz differential LVPECL
- Type C: ICS programmable, 200 MHz to 700 MHz
- Type D: unbalanced, single-ended transformer coupled into LVDS

Note that all clock module daughter board outputs are converted to LVDS on the daughter boards.

SDRAM Memory

The ML450 development board provides 64 MB of DDR-1 SDRAM memory (Micron Semiconductor MT46V32M16N-5B).

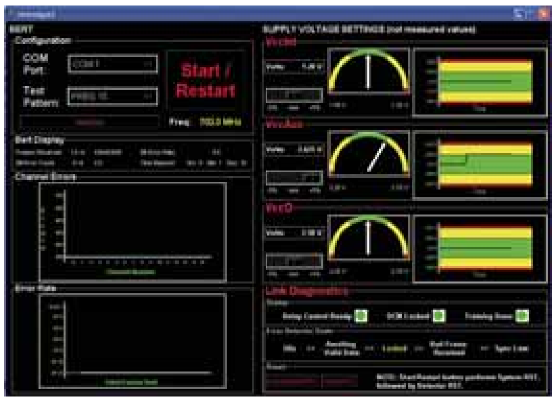


Figure 2 – BERT user interface

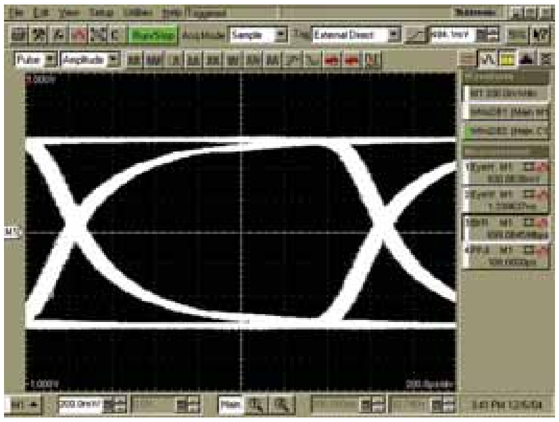


Figure 3 – 700 MHz SDR LVDS eye diagram

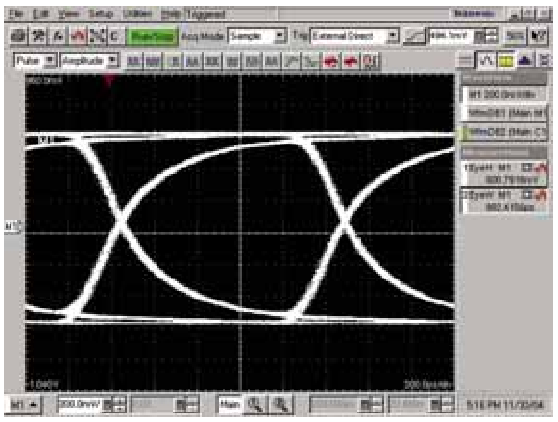


Figure 4 – 500 MHz DDR LVDS eye diagram

Liquid Crystal Display

The ML450 development board provides an 8-bit interface to a 64 x 128 LCD panel (DisplayTech Q64128E-FC-BC-3LP, 64 x 128).

RS232 Port

The ML450 development board provides a DB9-M connection for a simple RS232 port. The board uses the Maxim MAX3316 device to drive the RD, TD, RTS, and CTS signals. You must provide a UART core internal to the FPGA to enable serial communication.

System ACE Interface

In addition to a JTAG configuration connector, the ML450 development board provides a System ACE interface to configure the Virtex-4 FPGA. The interface also gives software designers the ability to run code (for soft-processor IP within the FPGA) from removable CompactFlash cards.

LVDS Connectors

The ML450 development board provides 40 channels of transmit signals and 40 channels of receive LVDS signals. These signals are distributed across two Samtec QSE-DP connectors for transmitting and another two connectors for receiving.

HyperTransport Connector

The ML450 development board provides 16 channels of transmit and receive data, along with miscellaneous control signals on the Samtec QSE HyperTransport connector.

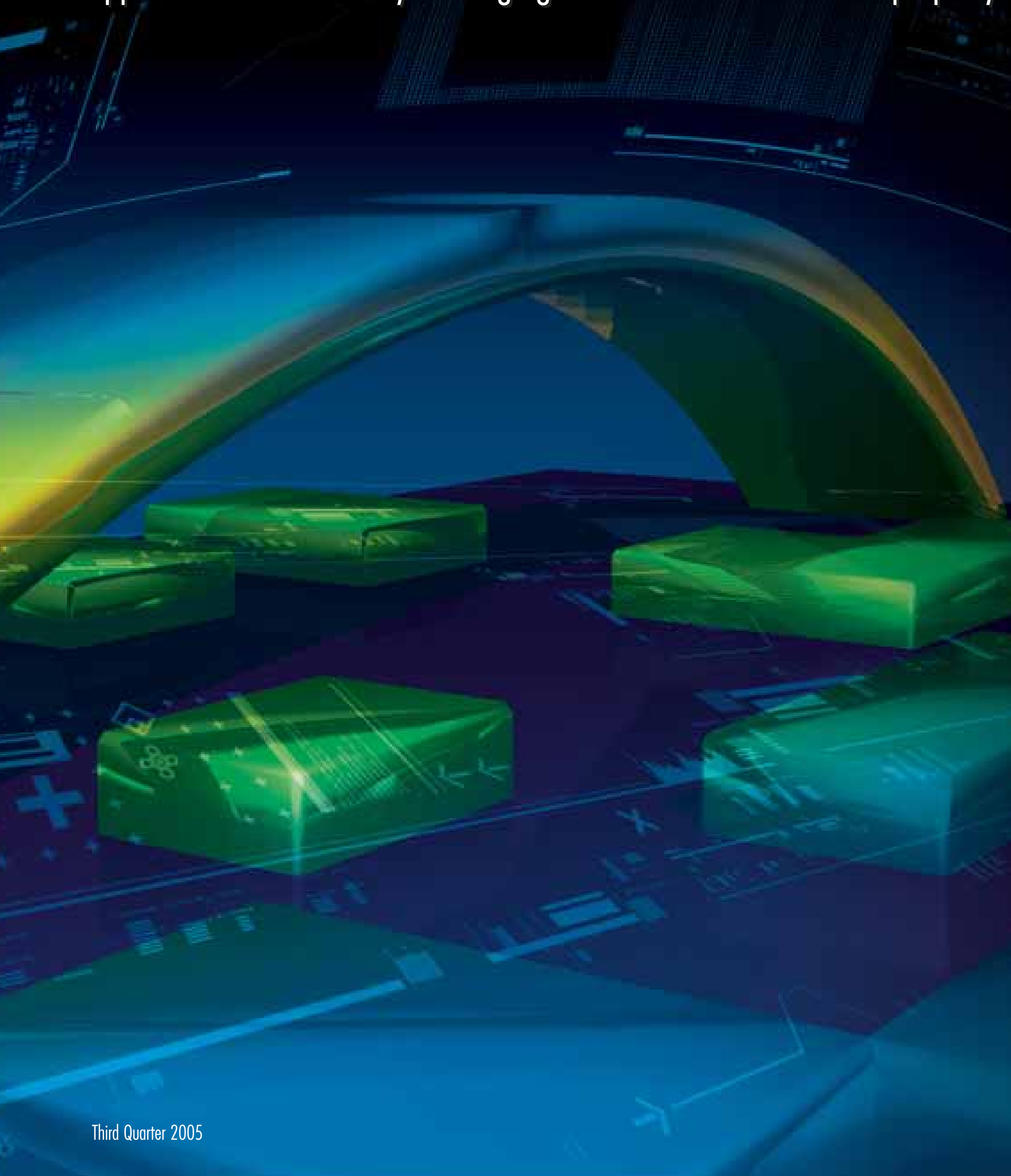
Conclusion

With the Virtex-4 source-synchronous interfaces tool kit, designing networking, telecom, servers, and computing systems has never been faster or easier.

For more information on the demonstration board and the kit, visit *www.xilinx.com/ml450/*.

Bridging System Packet Interfaces

The London Bridge was not designed in a day, but your system packet interface bridging applications can be – by leveraging a full suite of intellectual property and hardware platforms.



by Mark McLaughlin
Design Engineer, IP Solutions Division
Xilinx, Inc.
mark.mclaughlin@xilinx.com

Tom Fischaber
Staff Design Engineer, IP Solutions Division
Xilinx, Inc.
tom.fischaber@xilinx.com

Jeremy Goolsby
Staff Design Engineer, IP Solutions Division
Xilinx, Inc.
jeremy.goolsby@xilinx.com

In the last few years, the Optical Internetworking Forum's (OIF) system packet interfaces (SPI) for 2.5 Gb/OC-48 (SPI-3) and 10 Gb/OC-192 (SPI-4.2) have become the de-facto standards on all leading framer ASSPs. The SPI interfaces have also permeated into the network processor's space, including next-generation processors such as Intel's IXP2800 and IXP2400. Although these interface standards have been widely adopted, they pose significant design challenges to the system architect under pressure to deliver a fully compliant solution where time to market is paramount. Xilinx® Virtex™-4 architectures provide an ideal platform for implementing these multi-gigabit system packet interface applications.

Virtex-4 devices, combined with the portfolio of Xilinx pre-engineered IP solutions, are enabling system designers to build next-generation products faster than ever. Often these products involve bridging between multiple protocols, an application perfectly suited for FPGAs.

Figure 1 illustrates two examples of common bridging applications. The first FPGA bridges four SPI-3 (PL3) interfaces into a single SPI-4.2 interface, leveraging existing framers while also enabling support for the popular Intel IXP2800 network processor. The second FPGA bridges the SPI-4.2 interface to a backplane using the Virtex-4 embedded multi-gigabit transceivers. The Virtex-4 FX family supports a wide range of backplane applications, including PCI Express, XAUI, and Aurora.

To aid you in the development of your bridging applications, Xilinx provides a suite of application notes and reference designs. Leveraging the demonstration boards provided by Xilinx and its partners, you can complete these designs in hardware in a matter of hours or days, not weeks or months. Available under "Application Notes" at *www.xilinx.com/support/library.htm*, they include:

- Gigabit System Reference Design (XAPP 536)
- Gigabit Ethernet Aggregation to SPI-4.2 with Optional GFP-F Adaptation (XAPP 695)
- Mesh Fabric Reference Design (XAPP 698)
- Gigabit Ethernet to Aurora Bridge (XAPP 777)
- SPI-4.2 to Quad SPI-3 Bridge (XAPP 525)

In this article, we'll discuss the benefits and utility of the SPI-4.2 to Quad SPI-3 Bridge, which demonstrates how to bridge four SPI-3 cores to a single SPI-4.2 core. This solution implements channelized buffering, arbitration, and flow control using system packet interface protocols. Any system designer requiring this logic can successfully utilize the examples in this design.

SPI-4.2 to Quad SPI-3 Bridge

The outline of the SPI-4.2 to Quad SPI-3 bridge design is illustrated in Figure 2. This design not only converts electrical interfaces between SPI-3 and SPI-4.2, but also includes data buffering and width conversion, arbitration, and flow-control management.

For the dataflow from the SPI-3 to SPI-4.2 interface, the bridge accumulates multiple streams of data (as many as four different SPI-3 interfaces) and generates a single out-

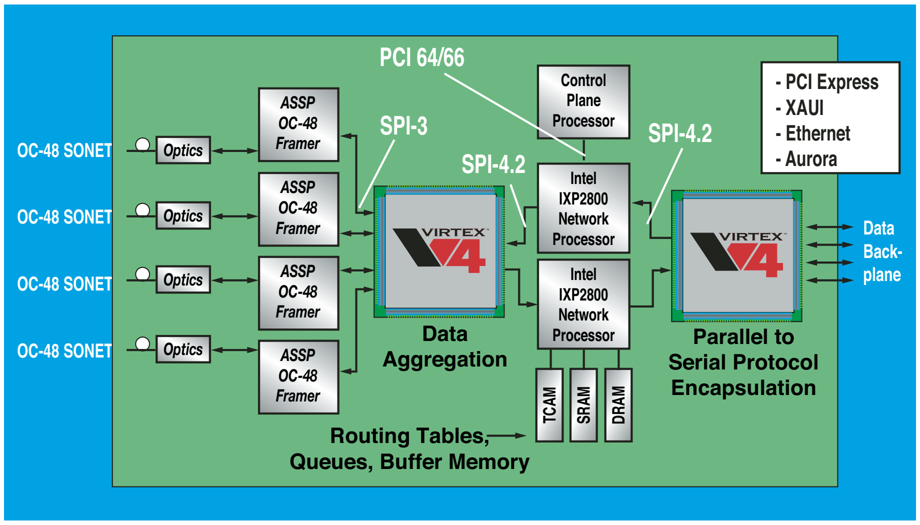


Figure 1 – Xilinx line-card example utilizing Virtex-4 FPGAs

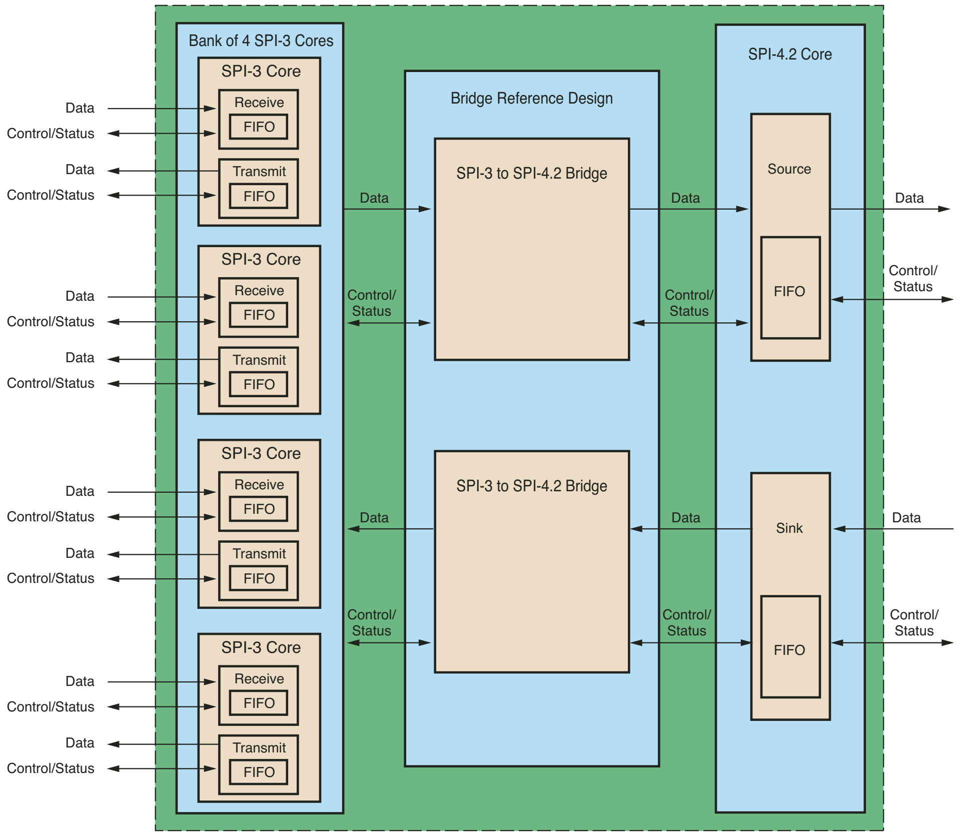


Figure 2 – Quad SPI-3 to SPI-4.2 bridge block diagram

put stream of SPI-4.2 data. A programmable amount of data is stored in each channel, and flow control from the SPI-4.2 interface determines the amount of data to transfer on each channel. Simple round-robin arbitration is implemented, but can easily be enhanced to provide more complex algorithms based on your system requirements.

In the SPI-4.2 to SPI-3 direction, the bridge de-multiplexes a single SPI-4.2 interface into individual channels, generating four separate SPI-3 output streams. A programmable amount of data is stored in each channel, and flow control from the SPI-3 interface determines the amount of data to transfer on each channel.

The reference design provides all of the required design files (in both VHDL and Verilog) to implement the bridge logic between the four SPI-3 cores and the SPI-4.2 core. The design supports a suite of parameters to customize the design based on user requirements. These include:

- FIFO thresholds to determine flow-control information
- Additional device/package information
- Static and dynamic alignment configurations of the SPI-4.2 core
- Configurable FIFO depths

The suite of design files and documentation provide a complete solution for your SPI-4.2 to quad SPI-3 bridging needs. You can find a detailed description of this bridge, as well as supporting design files, in application note XAPP 525, referenced previously.

System designers creating bridging applications outside of this SPI-3 to SPI-4.2 application will also find this reference design highly valuable. Per-channel data buffering, arbitration, and flow control are some examples of the features implemented by this bridge. The channelized FIFOs are created by the Xilinx FIFO Generator, which supports a suite of features including non-symmetric aspect ratios and first-word fall-through (FWFT). Non-symmetric aspect ratios enable easy width conversion between different data widths, and FWFT provides the ability to look ahead to the next word available from the FIFO without issuing a read operation. For more details on the FIFO Generator, see “Never Design Another FIFO,” also in this issue of the *Xcell Journal*.

IP Core Solutions

The SPI-3 and SPI-4.2 cores require high-speed I/O operation and tight timing to meet the OIF specifications. These cores easily meet these requirements through Virtex-4 ChipSync™ technology, enabling the use of the embedded I/O SERDES and dynamic phase alignment. This greatly alleviates the design complexities for SPI users, enabling them to focus on their system requirements instead of the SPI interfaces.

The SPI solutions are delivered through the CORE Generator in standard IP releases, and provide immediate simulation capability at no cost. You can also obtain a hardware evaluation license, which enables the SPI cores to be downloaded into hardware for full system evaluation. The cores will timeout after a couple of hours, enabling full hardware evaluation in your application.

The suite of SPI IP includes the SPI-4.2 for OC-192 applications and the SPI-3 Link, SPI-3 PHY, and SPI-4.2 Lite for OC-48 applications. The SPI-4.2 and SPI-3 IP provide simple out-of-the-box solutions for these complicated interface protocols.

SPI-4.2 and SPI-4.2 Lite Cores

The Xilinx SPI-4.2 IP is a fully verified, plug-in SPI-4.2 interface solution. It provides you with a wealth of configuration options to tailor the core for your specific design requirements, including the ability to operate at data rates exceeding 1 Gbps using dynamic phase alignment.

In addition to the SPI-4.2 full-rate core, Xilinx also provides the SPI-4.2 Lite core. The SPI-4.2 Lite IP leverages the efficiency of the SPI-4.2 interface for slower OC-48 applications. It is fully compliant to the SPI-4.2 OIF specification, except that it operates at a maximum frequency of OC-96 (5 Gbps), but requires less than 50% of the resources of the full-rate SPI-4.2 core.

SPI-3 PHY and Link Cores

The Xilinx SPI-3 Link and PHY cores provide complete solutions for your OC-48 applications, and support a suite of options to customize the cores to meet your application needs. Both the SPI-3 Link and PHY cores support not only a 32-bit interface, but also 8- and 16-bit interfaces for interfacing to a suite of framers and network processors. Previously available as fixed-point solutions, the SPI-3 PHY and Link cores will be released in ISE™ 7.1i IP Update 3, available in Q3 2005.

Conclusion

Xilinx IP and reference designs provide superior solutions for implementing custom bridging solutions between protocols. You can use the XAPP525 bridge design out of the box for SPI-4.2 to quad SPI-3 bridging applications or for any system that requires channelized buffering, arbitration, and flow control using system packet interface protocols. In addition, the SPI cores provide fully compliant, drop-in solutions for these complicated interfaces. The SPI solutions alleviate design challenges and result in faster time to market for Xilinx customers. The low cost and full feature set of the Virtex-4 family, combined with the suite of SPI IP and multi-gigabit bridge designs, provide an unbeatable combination.

Although we highlighted only the SPI IP in this article, Xilinx has a wealth of IP solutions for all of your connectivity and bridging needs. For more information, please visit Connectivity Central in the IP Lounge at *www.xilinx.com/products/design_resources/conn_central/index.htm*.

For more information regarding the SPI-4.2 core and its benefits in the Virtex-4 architecture, please refer to the following article from a previous edition of the *Xcell Journal*: *www.xilinx.com/publications/xcellonline/xcell_52/xc_v4spi52.htm*. For more information about the SPI-4.2 and SPI-3 IP solutions, visit *www.xilinx.com/ipcenter/posphyl4/spi42_core.htm*.

Managing Signal Integrity

Being heard above the noise.

by Steve Sharp
Sr. Marketing Manager
Xilinx, Inc.
steve.sharp@xilinx.com

Panch Chandrasekaran
Connectivity Marketing Manager
Xilinx, Inc.
panch.chandrasekaran@xilinx.com

The problem of signal integrity is a lot like trying to carry on a conversation at a crowded trade show. If you and the person you're talking to are in a quiet corner of the hall with some nice padded walls around you and not too many other people nearby, it isn't a problem. Try the same conversation in the middle of the exhibit floor with hundreds of people all around, noise from neighboring exhibit booths, and no walls to break up or absorb the sound, and you've got a problem.

Back in the good old days of logic design, we didn't give much thought to signal integrity. We had 5V power supplies, DIP packages with leads that actually went through the board, and high-speed microprocessors running at a heady 5 MHz.

If you paid a little attention to board layout and put a decent ceramic bypass capacitor next to each chip, you probably didn't have to worry about your signals. Ones stayed ones and zeroes stayed zeroes. Even 100 mV of noise on a signal wouldn't be enough to change its logic level.

Today, designers are caught between design requirements for ever-increasing bit rates, faster edge rates, higher clock speeds, and technology advances that keep lowering operating voltages, reducing package sizes and ball pitch, and forcing more components into a smaller amount of board area.

Signal Integrity Today

Take a look at present-day source-synchronous interfaces. DDR and QDR memory interface speeds are rapidly increasing, with DDR2 speeds at more than 500 Mbps. The bit rates are getting faster and the buses getting wider. With faster bit rates comes faster edge rates, which now can be just a few hundred picoseconds.

Faster is better, but there are a few hurdles to deal with. Parasitic inductance and capacitance – which didn't matter a whole lot at lower speeds – are suddenly very important. The resulting noise because of the parasitics is a big concern. Today it is common to have FPGAs with hundreds of I/Os switching, causing high levels of simultaneous switching output noise (SSN). This affects your system in many ways, especially causing jitter, which can reduce your timing margin or even cause system failure.

You cannot allow yourself to ignore signal integrity and gamble that your system will work as designed. You might be forced to reduce the clock rate just to get the system to work, or be forced into a complete board re-design to correct signal integrity issues.

What Kind of Integrity Do You Have?

Having good signal integrity usually means controlling unwanted noise on logic signals. Noise usually falls into one of two main domains:

- Level-related noise affects the logic level of the signal. If the noise is large enough, the signal may cross the threshold from a desired logic state to an undesired state and propagate into other logic.
- Time-related noise, or jitter, affects the position of a signal transition and causes setup/hold windows for data sampling to be violated, thereby allowing incorrect data to be sampled and propagated through the system.

The combination of level noise and jitter combine to reduce signal margins in both the voltage and time domains, effectively reducing the "eye" or window in which good data is available.

Controlling Noise

A well-designed package is critical to signal integrity. Noise can emanate from many sources in a system. If the noise source is on the board, there are some potential solutions once you find out where the problem is (probably after a long and laborious debug process). If the problem is in the package, you have little or no choice but to change the design, vendor, or parts. This is a time-consuming process that can affect product revenue significantly. For this reason, it is imperative to have a well-designed low-inductance package.

When speeds were still fairly low, short signal paths did not alter signal characteristics. Today, with rise times in the hundreds of picoseconds (even if bit periods are a few nanoseconds), the frequency components of signals run into gigahertz, causing even very short signal paths like package traces to impact signals.

For every signal line, there is an associated return path for the return currents. For single-ended signals, these return paths are usually GND or VCC reference planes. To maintain a 50 ohm line, the returns should be in close proximity to the signal.

Although PCB traces are less of a concern, you must pay close attention to vias. For large FPGAs the breakout region – the area between the package balls to the PCB – is extremely critical, as it comprises a dense concentration of signal vias.

SSN is generally observed as "ground bounce" and can be caused by two different phenomena.

First, noise because of via-field crosstalk is a function of loop inductance, which is a function of the proximity of ground/power reference pin locations to the signal pin. Signal pins farther away from a reference pin are more susceptible to noise.

This problem is exacerbated when a number of I/Os in the region switch simultaneously. Proper distribution of ground/power and signal pins in a package is extremely critical – in other words, a good pinout architecture.

Second, maintaining a clean power supply to the FPGA is also critical to maintain acceptable signal integrity. Noise margins are reduced as VCC values drop down to 1.2V.

Furthermore, any noise in the power rail translates to jitter at the output, shrinking available timing margins. As noise depends on package inductance and the number of simultaneously switching I/Os, optimal signaling requires a good low-inductance package.

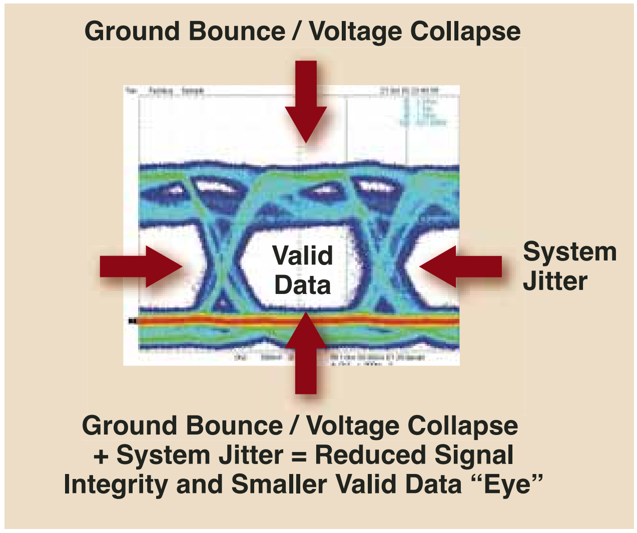

Ground Bounce / Voltage Collapse
Valid Data
System Jitter
Ground Bounce / Voltage Collapse + System Jitter = Reduced Signal Integrity and Smaller Valid Data "Eye"

Figure 1 – How noise and jitter can affect an eye diagram.

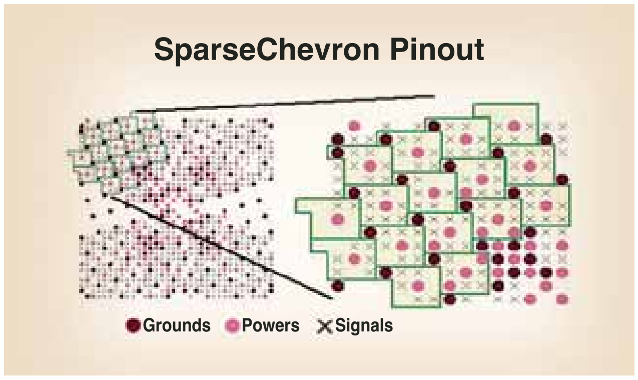


Figure 2 – Xilinx SparseChevron pinout places return pins adjacent to each signal pin.

Tackling the SSN Challenge

One package that tackles the SSN challenge is the Xilinx® Virtex™-4 FPGA package. Most notably, the package enables better noise performance on higher speed single-ended interfaces, which are more susceptible to noise than differential interfaces such as LVDS.

The pinout architecture of the package is responsible for roughly 80% of the total noise. The Virtex-4 FPGA package achieves optimal pin distribution through a tiled pattern – a regular array of signal, ground, and power pins called SparseChevron pinout (Figure 2).

The signal-to-ground-to-power ratio of the package is 8:1:1. Because both power and ground are equally effective as return current paths, the package effectively has a signal-to-return ratio of 4:1. Also, the pins are distributed so that every signal pin is adjacent to a return pin, ensuring that the return current loop is kept to a minimum.

Additionally, the abundance of return paths in any given area of the package provides a low impedance path for the return currents. The pinout also confines noise from an aggressor to a smaller area so that the influence of the aggressor drops rapidly with distance. Because crosstalk noise is cumulative, this results in a lower total SSN.

Simplifying Signal Termination

On-chip termination (active termination) removes external components and places termination closest to where it matters (driver or receiver).

To maintain the ideal 50 ohm line impedance, it is normal design practice to have termination resistors on each signal. For hundreds of signal I/Os, this can translate to many hundreds of external termination resistors. The physical challenges of placing the resistors on the board and their connections to the power and ground planes are not trivial.

The Xilinx Controlled Impedance Technology (XCITE) on-chip active I/O termination used in Virtex FPGAs solves many of the problems associated with signal termination. XCITE provides both parallel and serial equivalent options for single and differential termination. Impedance is controlled using an internal reference voltage and is available on all I/O pins. This active termination provides automatic temperature and volt-

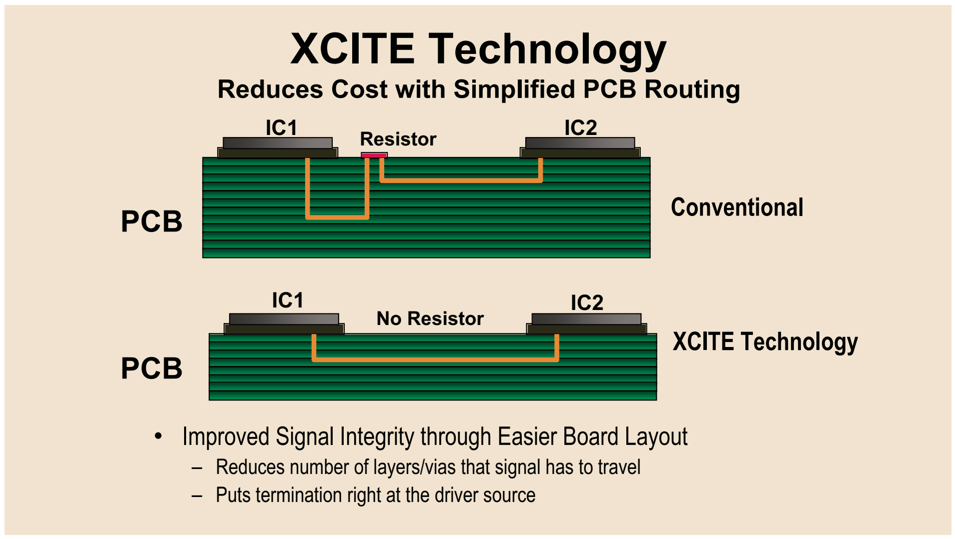

XCITE Technology
Reduces Cost with Simplified PCB Routing
IC1
Resistor
IC2
PCB
Conventional
IC1
No Resistor
IC2
PCB
XCITE Technology
• Improved Signal Integrity through Easier Board Layout
– Reduces number of layers/vias that signal has to travel
– Puts termination right at the driver source

Figure 3 – Xilinx XCITE technology reduces cost through simplified PCB routing.

age compensation; puts the termination inside the buffer circuitry where it belongs; and saves board space and cost by eliminating hundreds of discrete resistors. Figure 3 shows the simplified board layout and signal trace paths using both conventional and Xilinx XCITE DCI termination technology.

Power Plane Integrity

Power and ground planes are important to maintaining signal integrity in FPGA designs. To maintain the characteristic impedance (Zo) across the frequency range of interest, reference planes for single-ended signals should be very low impedance.

Otherwise, the result is impedance discontinuities, causing jitter due to reflections. In addition, noisy power and ground planes affect circuit performance on the die, causing additional jitter. It is important to design packages with continuous power and ground planes to minimize impedance.

Typically, PCB designers use decoupling capacitors to filter out noise and maintain a clean power supply. For reducing high-frequency noise, decoupling capacitors are placed close to the noise source. Leading-edge ASICs and FPGAs are equipped with very low-inductance decoupling capacitors within the package to aid cleaning the power-supply noise.

Compensating for Signal Integrity Issues

Improving the signal integrity of your system will enhance the data valid window (the eye) of the high-frequency signals reaching your FPGA I/O pins. However, this is only half the battle. Even superior designs exhibit shrinking data valid windows, as shown in the 533 Mbps DDR2 SDRAM example shown in Figure 4. The input circuitry needs the capability to capture the data by centering the clock to the middle of the shrinking data valid window.

Virtex-4 FPGAs have unique ChipSync™ technology built into every I/O block that makes data capturing easier and more reliable. It includes a precision delay called IDELAY that generates the tap delays necessary to center data to the FPGA clock. Memory strobe edge detection logic, included in the I/O block, uses this precision delay to detect the edges of the memory strobe from which the pulse center can be calculated. Delaying the data by the number of delay taps counted between the first and second edges aligns the center of the data window with the edge of the FPGA clock output. The tap delays generated by this precision delay block allow alignment of the data and clock to within 75 ps resolution.

ChipSync technology also simplifies the design of differential parallel bus interfaces, with embedded SERDES blocks that serialize and de-serialize parallel interfaces to match the data rate to the speed of the internal FPGA circuits. Additionally, this technology provides per-bit and per-channel de-skew for increased design margins, simplifying the design of interfaces such as SPI-4.2, XSBI, and SFI-4, as well as RapidIO.

Conclusion

Signal integrity is a key issue in today's high-speed designs and will continue to be important as more high-speed signals are squeezed into smaller amounts of board space, packages get denser, and ball spacing shrinks.

Signal integrity issues can affect voltage and time domains, combining to reduce the window of available valid data in a system. If the issues become large enough, systems may not work at all or be extremely unreliable, forcing long and costly system redesigns.

It might never be possible to completely eliminate signal noise in a high-speed system. But paying attention to several key areas can minimize noise or adjust timing so that system performance is not compromised. These include using well-engineered, low-inductance packages; using devices with built-in power supply decoupling; using active signal termination where necessary; and choosing devices with the ability to adjust the relationship between the data valid window and the clock.

For more information, visit *www.xilinx.com/signalintegrity.*

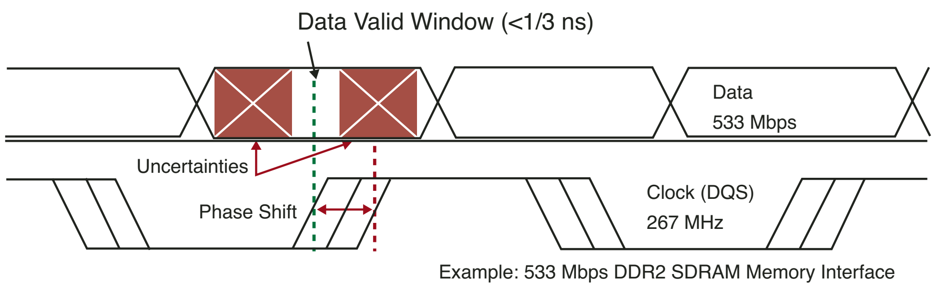
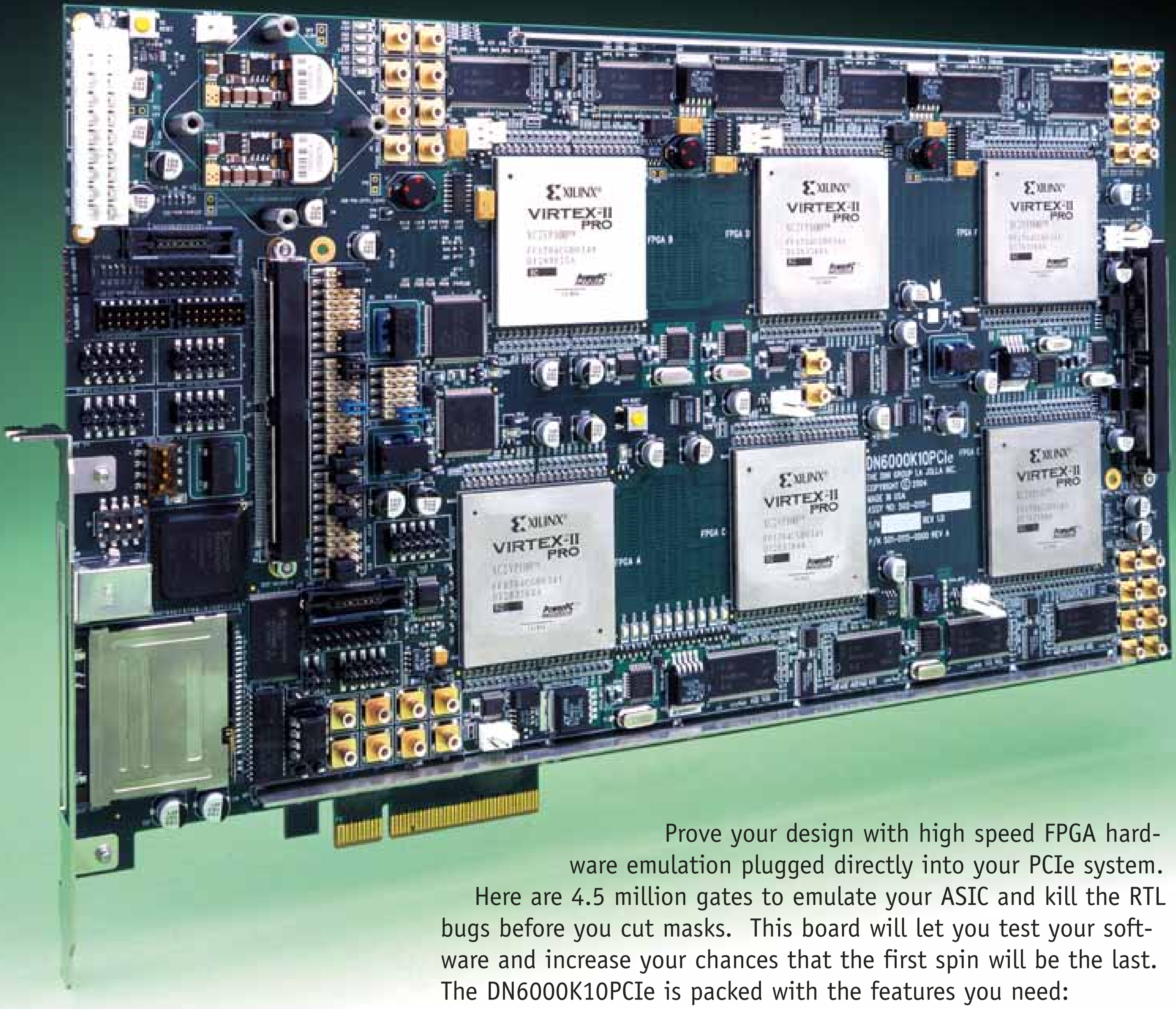

Data Valid Window (<1/3 ns)
Data
533 Mbps
Uncertainties
Phase Shift
Clock (DQS)
267 MHz
Example: 533 Mbps DDR2 SDRAM Memory Interface

Figure 4 – Data valid window for a 533 Mbps DDR2 SDRAM interface

Prototype Your PCIe ASIC *HERE*



Prove your design with high speed FPGA hardware emulation plugged directly into your PCIe system. Here are 4.5 million gates to emulate your ASIC and kill the RTL bugs before you cut masks. This board will let you test your software and increase your chances that the first spin will be the last. The DN6000K10PCIe is packed with the features you need:

- 1,4 and 8-lane versions
- Six VirtexII-Pro FPGAs (-2vp100s, the big ones)
- 10 DDR (64Mx16) and 4 SSRAMs (2Mx36) external to the FPGAs
- Expansion capability to customize your application
- Synplicity Certify® models for quick and easy partitioning

Like all our products, this new PCI Express bus board will help you get your ASIC to market on time and in budget. Call The Dini Group today-- PCIe is already here.

See us at DAC Booth #1879

The DiNI Group

1010 Pearl Street, Suite 6 • La Jolla, CA 92037 • (858) 454-3419 • Email: sales@dinigroup.com
www.dinigroup.com

Hardware/Software Co-Verification

Gain full visibility into your software and hardware – and achieve a faster design iteration loop in the process.

by Ross Nelson
Seamless FPGA Product Manager
Mentor Graphics Corporation
ross_nelson@mentor.com

You've probably been there: clever detective work leads you to a small change in the HDL for your embedded processor-based design. Now you just have to run synthesis, place and route, and darn ... you suddenly realize it will be another day before you can see the result.

Large devices allow you to stuff a whole system into the FPGA, but debugging these complex systems with limited visibility – and a one-day turnaround – can consume weeks of your precious time.

Hardware/software co-verification has been successfully applied to complex ASIC designs for years. Now available to FPGA designers, Seamless FPGA from Mentor Graphics brings together the debug productivity of both a logic simulator and a software debugger. Seamless FPGA co-verification enables you to remove synthesis and place and route from the design iteration loop, while yielding performance gains 1,000 times faster than logic simulation.

Shortening the Design Iteration Loop

Because development boards are readily available, many FPGA designers incorporate them into the highly iterative design loop. Unfortunately, the development board brings major overhead to every design iteration. This overhead comes in the form of logic synthesis, followed by place and route. Although necessary to produce a final design, you can remove these time-consuming steps from the highly iterative design debug loop by targeting simulation as the verification platform.

With simulation as the verification engine, the only overhead between editing the HDL and verification becomes a relatively quick compile of your HDL. The time you can save on your next embedded FPGA is easy to calculate: How many times did you run place and route on your last FPGA design? And how long did place and route consume your PC for each run?

It's true that simulation runs slower than the real-time speed of a development board. Seamless FPGA provides some innovative ways to dramatically increase the rate at which your embedded software simulates. The increase in a typical system is several orders of magnitude.

Improving Hardware and Software Visibility

To debug your FPGA design, you need full and clear visibility. You need to know what is happening in the hardware and what the software is doing. You need to be able to change a register, or force a signal to a different state. Sometimes you need to be able to stop time and take a closer look. The more visibility you have, the more quickly you can see the problem or prove you have resolved the bug.

Hardware Visibility

Probing inside or even on the pins of your FPGA is a challenge. The ChipScope™ Pro analyzer from Xilinx® helps with this, but in a logic simulator (in addition to viewing every signal) you can also change their values. Working from your source HDL, you can step through the code, view variables, or stop time. For detailed, immediate, and hassle-free visibility, it is hard to beat logic simulation.

Software Visibility

Software visibility in logic simulation is another item with which to contend. Running the fully functional processor model allows you to execute software, but knowing what is in R3 of the processor is almost impossible if you are given only waveforms.

Co-verification provides an enhanced processor model connected to a software debugger. In the Mentor Graphics XRAY debugger, you can view and change everything from registers to memory, stack, and variables. XRAY also provides a source code view with symbolic debug. You can step through code at the source or assembly level and use breakpoints to halt execution or run powerful macros.

If you are using the Accelerated Technology Nucleus real-time operating system (RTOS), you can view the status of tasks, mailboxes, queues, pipes, signals, events, semaphores, and the memory pool.

Much Faster Than Logic Simulation Alone

Running substantial amounts of software on a standard processor model in logic simulation is not practical; the run times are just too long. However, running this software actually turns out to be one of the most effective verification strategies available. The payoff for running diagnostics, device drivers, board support package (BSP) code, booting the RTOS, and running low-level application code is huge. It is not surprising that verifying hardware – by putting it through its paces the way the software will actually use it – is effective. Similarly, the software is tested against the actual design (including any external board-level components that are included in the simulation) before the board is actually built.

The challenge has always been to run enough software to really boot the system and do something interesting. Co-verification is able to speed up the run time by taking advantage of one simple observation: most of the simulation time is spent re-validating the same processor-to-memory path. Although you need to test your memory subsystem and try several dozen corner cases, you don't need to repeat those same tests over again every time you fetch an instruction from memory. Similarly, you need to verify that the processor can push a value on the stack and pop it off again with the correct result, but repeating this test every time a software function is called would be overkill.

Accesses to hardware peripherals always generate bus cycles in the logic simulation, but instruction fetches and stack operations can typically be offloaded for faster execution. By allowing you to specify which bus cycles are run in the logic simulator and which are not, Seamless FPGA allows you to make the performance tradeoff. And you can change this specification at any time during your simulation session. You can run through reset with full cycle-accurate behavior, and then switch off instruction fetches and stack accesses to boot the RTOS.

Accessing memory through the logic simulator requires several hardware clock cycles. Each clock cycle requires significant work in the logic simulator as it drags along the heavy weight of all the other logic in your FPGA. Using a "back door" to directly access the memory contents, instead of running the bus cycle in the logic simulator, allows accesses to occur many orders of magnitude faster.

The speedup is very significant. For example, the following data is from a typical design configuration with a PowerPC™ running Nucleus on the Xilinx Virtex™-II Pro FPGA. Booting the Nucleus RTOS in logic simulation alone requires 12 hours and 13 minutes. The same task with these techniques employed accomplishes the task in only six seconds – 7,330 times faster.

Using this technique, Seamless FPGA maintains one coherent view of memory contents through a back door into Xilinx block RAM memory models or any other memory device. So if your DMA controller drops something into memory that the processor later executes, it will still all work together correctly. And if the processor generates a large data packet and instructs hardware to transmit it using DMA, there are no data inconsistencies.

Identifying Processor Bus Bottlenecks

The performance of your FPGA platform can be seriously impacted by the memory structure of the design. What should be located in cache versus block RAM or external memory? Where are the bottlenecks? Do other bus masters starve the processor? Questions like these are important, but getting the answers can be difficult without real data from your hardware/software application.

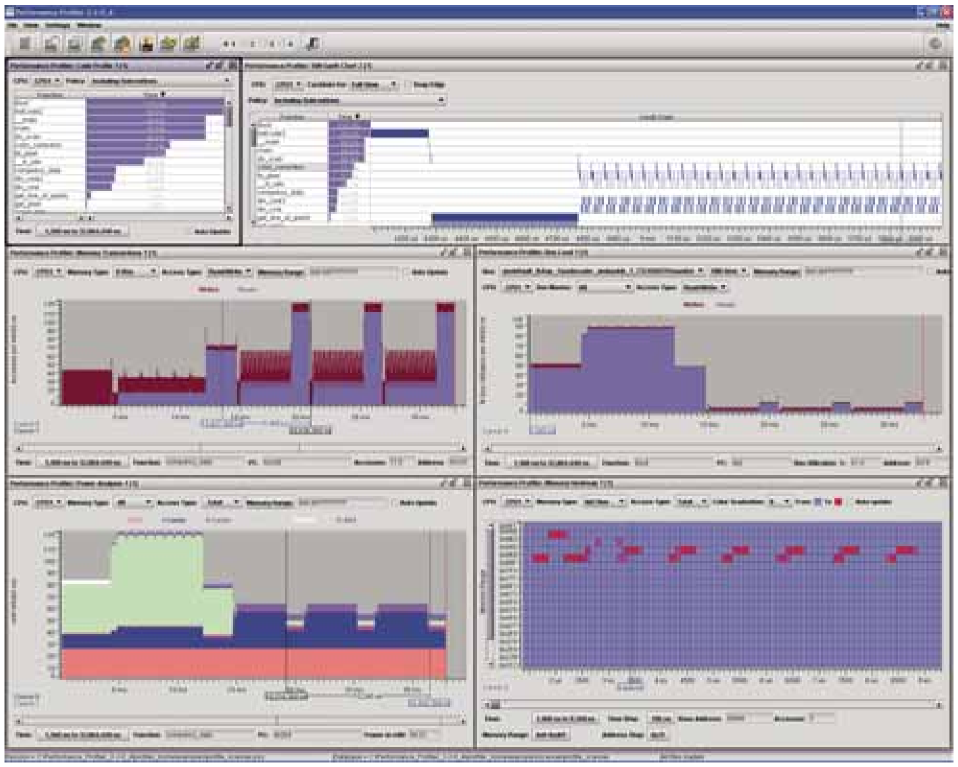


Figure 1 – Seamless FPGA's system profiler helps you tune performance by providing detailed data on bus transactions and utilization, software function execution time, bus arbitration delay, memory hot spots, and software code profiling.

…software doesn't execute alone – it interfaces with hardware, and the hardware/software interface often stretches across disciplines and design teams.

Seamless FPGA gathers performance data from the simulation and displays it graphically in the system profiler (Figure 1), enabling you to identify:

- Which functions are consuming most of the CPU time
- Unexpected lulls or bursts of activity
- Cache efficiency and memory hot spots
- Code execution and duration at the function level
- Bus utilization and bus master contention

Ease of Use and Integration

Seamless FPGA is easy to use and set up. Using the knowledge you have already entered in Xilinx Platform Studio (XPS), Seamless FPGA automatically configures itself to co-verify your design. You may already know how to use ModelSim, and Seamless FPGA leaves the full functionality and user interface unchanged. The XRAY software debugger uses many of the same menu icons for operations like step, step over, and run.

To set up Seamless FPGA, simply choose File > Import from Xilinx Platform Studio and specify your XPS project file. The import process does all of the setup steps and in about one minute proceeds to invoke ModelSim and the XRAY debugger.

If you have two or more Xilinx processors in your design, you will have additional software debugger windows, one for each processor.

Once ModelSim and XRAY have been invoked (Figure 2), you are ready to verify your design. In ModelSim, enter any stimulus commands needed – typically this is reset and clock, plus any design-specific stimulus – and then click "run." In XRAY, click "go" or "step" to start stepping through your embedded code. By default, all bus cycles are routed to the hardware simulation.

To increase software execution speed, three icon selections are provided. These icons are labeled "optimizations" because they increase the rate of software execution by directing Seamless FPGA to access memory contents through a back door without requiring the logic simulator to run every bus cycle. The first button directs all instruction fetch cycles to use the back door. A second button allows you to specify any number of address ranges, which use the back door. When accesses use the back door, you can either choose to keep advancing the logic simulation in lock step with the software or remove that requirement.

The optimization settings can be changed at any time on the fly during a simulation session. This allows you to quickly run to a certain point in your software, and then enable all bus cycles for detailed cycle-accurate verification.

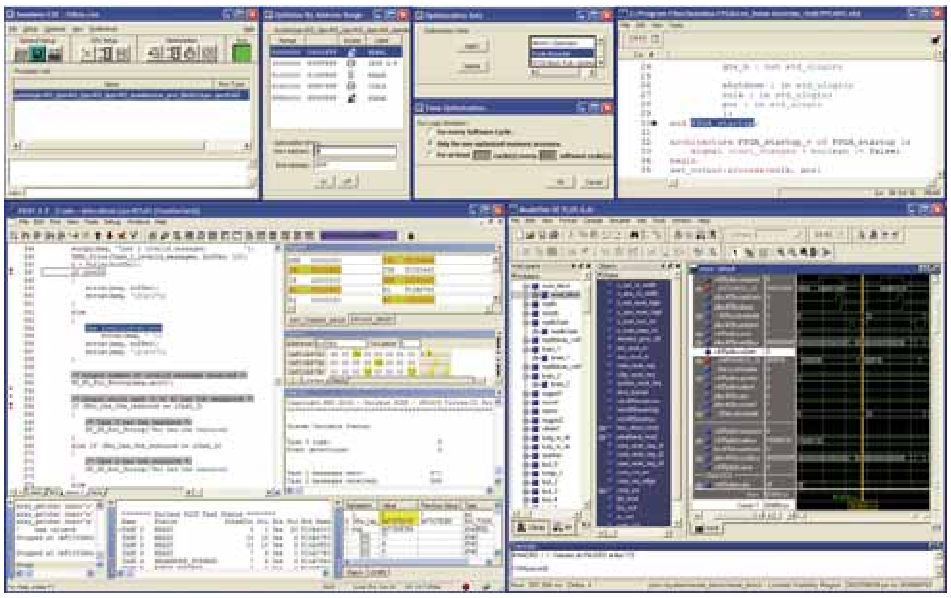


Figure 2 – Seamless FPGA running the Nucleus RTOS on a Xilinx Virtex-II Pro PowerPC processor. XRAY (lower left) provides symbolic software debug and RTOS task status. ModelSim (lower right) enables full hardware debug and control. Seamless FPGA control panels (upper left) allow dynamic selection of bus cycles routed to ModelSim. The ModelSim HDL source window is also displayed (upper right).

Conclusion

With large FPGA designs employing embedded processors, it's not possible to complete a design in a few weeks. These designs are very sophisticated; unfortunately, so are the bugs that you must track down and resolve to produce an effective system on schedule.

Software content in your FPGA can bring lower system costs, higher configurability, and increased functionality. But software doesn't execute alone – it interfaces with hardware, and the hardware/software interface often stretches across disciplines and design teams.

Seamless FPGA bridges the hardware/software gap with a productive software and hardware debug environment that provides the visibility to find bugs and performance bottlenecks efficiently. And once you have fixed them, you can quickly turn the fix and verify it, without having to wait for your PC to rumble through place and route for hours on end.

Try Seamless FPGA on your design today. For your free 30-day evaluation copy, visit *www.seamlessfpga.com*. The included example design and Quick Start Guide will get you up and running in no time. For more information, e-mail *seamless_fpga@mentor.com*.

X-ray vision for your designs

Agilent 16900 Series logic analysis system with FPGA dynamic probe

- *Increased visibility with FPGA dynamic probe*
- *Intuitive Windows® XP Pro user interface*
- *Accurate and reliable probing with soft touch connectorless probes*
- *16900 Series logic analysis system prices starting at $21,000*

Agilent Direct

Get a quick quote and/or FREE CD-ROM with video demos showing how you can reduce your development time.

U.S. 1-800-829-4444, Ad# 7909
Canada 1-877-894-4414, Ad# 7910
www.agilent.com/find/new16900
www.agilent.com/find/new16903quickquote

Now you can see inside your FPGA designs in a way that will save days of development time.

The FPGA dynamic probe, when combined with an Agilent 16900 Series logic analysis system, allows you to access different groups of signals to debug inside your FPGA—without requiring design changes. You'll increase visibility into internal FPGA activity by gaining access up to 64 internal signals with each debug pin.

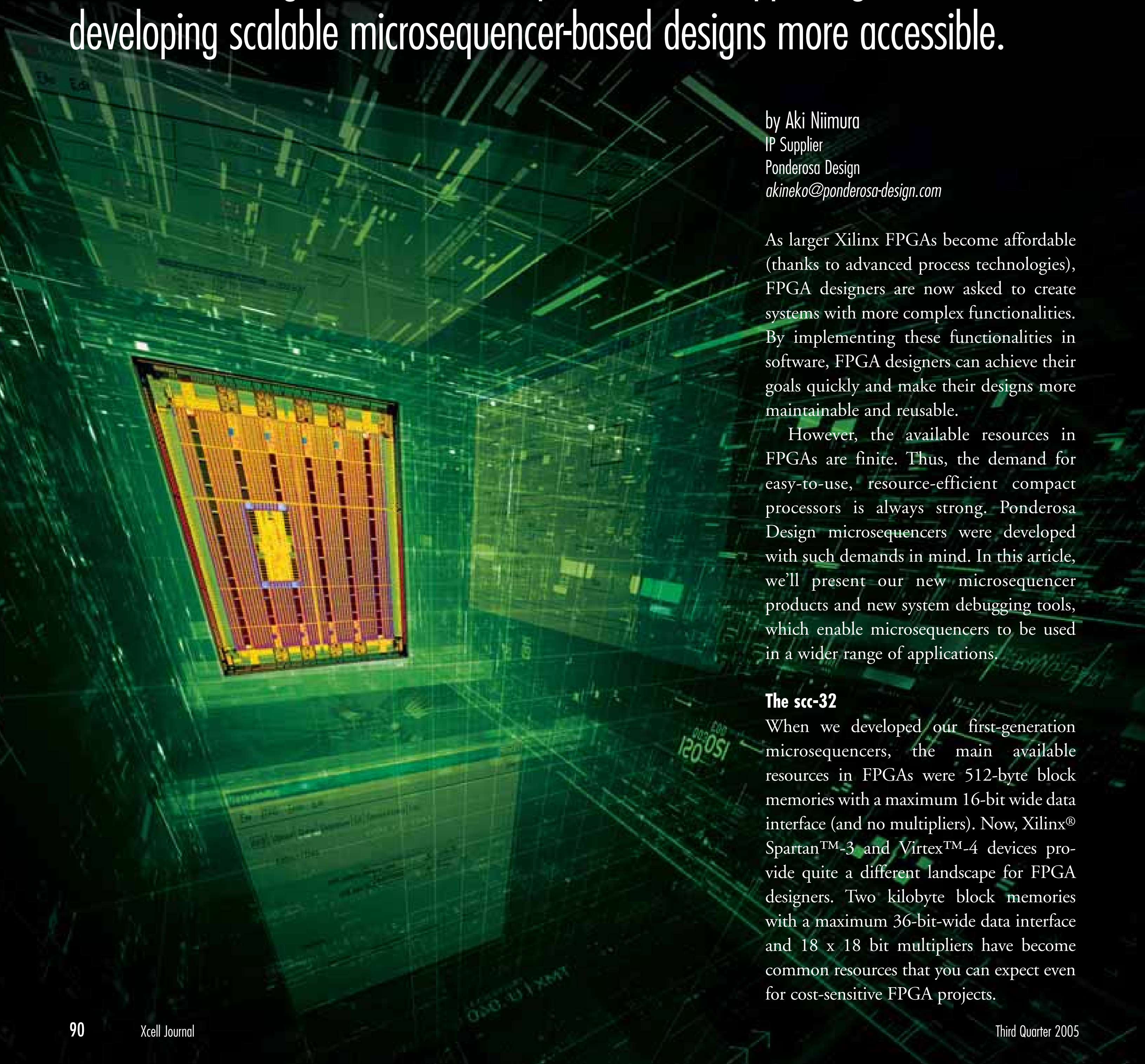
You'll also be able to speed up system analysis with the 16900's hosted power mode—which enables you and your team to remotely access and operate the 16900 over the network from your fastest PCs.

The intuitive user interface makes the 16900 easy to get up and running. The touch-screen or mouse makes it simple to use, with prices to fit your budget. Optional soft touch connectorless probing solutions provide unprecedented reliability, convenience and the smallest probing footprint available. Contact Agilent Direct today to learn more.

Agilent Technologies
dreams made real

The scc-32/scc-16 Microsequencers and AHBDBG System Debugger

Ponderosa Design's new microsequencers and supporting tools make developing scalable microsequencer-based designs more accessible.



by Aki Niimura
IP Supplier
Ponderosa Design
akineko@ponderosa-design.com

As larger Xilinx FPGAs become affordable (thanks to advanced process technologies), FPGA designers are now asked to create systems with more complex functionalities. By implementing these functionalities in software, FPGA designers can achieve their goals quickly and make their designs more maintainable and reusable.

However, the available resources in FPGAs are finite. Thus, the demand for easy-to-use, resource-efficient compact processors is always strong. Ponderosa Design microsequencers were developed with such demands in mind. In this article, we'll present our new microsequencer products and new system debugging tools, which enable microsequencers to be used in a wider range of applications.

The scc-32

When we developed our first-generation microsequencers, the main available resources in FPGAs were 512-byte block memories with a maximum 16-bit wide data interface (and no multipliers). Now, Xilinx® Spartan™-3 and Virtex™-4 devices provide quite a different landscape for FPGA designers. Two kilobyte block memories with a maximum 36-bit-wide data interface and 18 x 18 bit multipliers have become common resources that you can expect even for cost-sensitive FPGA projects.

The program size is smaller because the instruction is one byte long. Stack architecture doesn't use deep pipelining, resulting in a predictable interrupt latency.

We developed our newest microsequencer, the scc-32, to fully utilize such resources, as well as to provide 32-bit data handling capability. As the name implies, the scc-32 is a 32-bit controller – but how does it compare to the MicroBlaze™ soft-core processor? The scc-32 is not designed as a generic microcontroller like MicroBlaze or Power PC™ processors. Our microsequencers are designed to take different roles and work with generic microcontrollers instead.

Our microsequencers employ a stack architecture, while today's generic microcontrollers employ a register-based architecture. Stack architecture is suitable for custom processors for FPGAs because the core is compact and resource-efficient (an effective use of block RAM). The program size is smaller because the instruction is one byte long. Stack architecture doesn't use deep pipelining, resulting in a predictable interrupt latency.

Contrary to common perceptions, supporting 32-bit data types is not difficult, nor does it consume a lot of resources in the FPGA. Our microsequencers use stack architecture, where the data size is irrelevant to each stack operation. When it comes to FPGA resources, all 32-bit data is stored in block RAM rather than registers. However, the arithmetic logic unit (ALU) must be a 32-bit ALU.

The scc-32 uses unified memory architecture (UMA). It needs three logically independent memories (data stack, program stack, and register file) in addition to a program memory. With UMA, three logically independent memories are unified into a single memory. One block RAM (38-bit wide, 512-word deep) can hold a 32-level-deep data stack, a 16-level-deep program stack, 144 global registers, and 24 auto registers per function call.

The scc-32 has a 16-bit program space, 64 KB, while our previous generation has an 11- or 13-bit program space. A larger program space means that a large amount of data (such as coefficient tables or message strings) can be included in a program. As our microsequencers have instructions to read from/write to program memory, program memory space can be used as an extra storage or data space to share with another process.

One important architectural change we observed from older Xilinx FPGAs to newer Xilinx FPGA families is the absence of internal tri-state buffer (TBUF) resources. Our older microsequencers utilize TBUFs to construct multiplexers with a large number of inputs. As they are no longer available in newer Xilinx FPGA families, the scc-32 is designed without internal TBUFs and optimized to minimize the complexity of such data multiplexers.

The scc-16

Our first-generation microsequencers were 16-bit controllers. They had a much tighter programming model; for example, the scc-IIs, one of our first microsequencers, had a 2 KB program size limit, five-level function call, eight global registers, and eight auto registers per function call. Not all applications require 32-bit data types, but you may want to use a newer programming model like the scc-32.

	f_{op}	LU	BRAM	TBUF	FPGA
scc-32	50MHz	1236	2		SP3
scc-16	50MHz	845	2		SP3
scc-IIs	25MHz	801	1		SP3
scc-IIs	25MHz	596	1	273	SP2

Table1 – "Hello" project (core + UART)

The scc-16 microsequencer uses an identical architecture and instruction set to the scc-32 (some instructions are dropped because there is no need to support 32-bit data types). The scc-16 provides a smaller footprint, which is close to the one that the first-generation microsequencer delivered (see Table 1). For example, even with the smallest Spartan-3 FPGA (the XC3S50), the scc-16 "hello" project only consumes 53% of the device, leaving ample space for other logic to be implemented.

Supporting a 32-Bit System Bus

A common practice is to use a system bus to create a larger system using bus-compliant modules. We originally designed our microsequencers for stand-alone use, but in some situations, you may want to connect a microsequencer to a system bus. For example, a microcontroller can download a program to a microsequencer on the fly so that the block can be used as a reconfigurable functional block. A microsequencer can also share a resource with other controllers.

The Advanced Microcontroller Bus Architecture (AMBA) high-performance bus (AHB bus) from ARM Ltd. is a system bus similar to the CoreConnect used with MicroBlaze and PowerPC processors. We use 32-bit AHB bus, which results in a 32-bit address space (4 GB). To interface to the AHB bus, we developed two wrapper modules, ahb32wrap (AHB master) and scc32ahb/scc16ahb (AHB slave). For the AHB master, macros are provided to access the 32-bit address space, as the scc-32 native instructions cannot access 32-bit address space directly.

The following code excerpts demonstrate how accessing the AHB bus is coded in the "SC" program. The SC language is a proprietary high-level language specifically for the SCC-II microsequencer family.

```
ahbwrite(DMAC_CTRL, 0x00000101);
ahb_status = ahbread(DMAC_STAT);
```

With the wrappers, the microsequencer's internal signals as well as the entire program memory are exposed to the AHB bus. To facilitate the development of a system with the AHB bus, we developed a new debugging tool in addition to our original stand-alone JTAG debugger.

The AHBDBG

The AHBDBG is a debugging tool for systems with the AHB bus (Figure 1). It provides a wide range of features to debug an AHB bus-based system, while our original

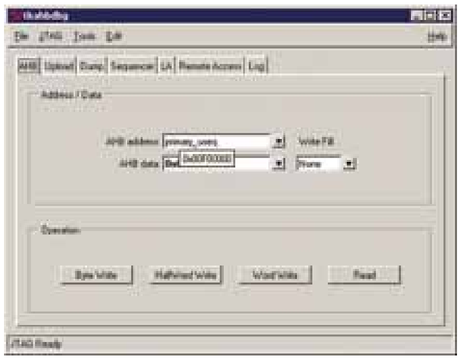


Figure 1 – AHBDBG screen shot

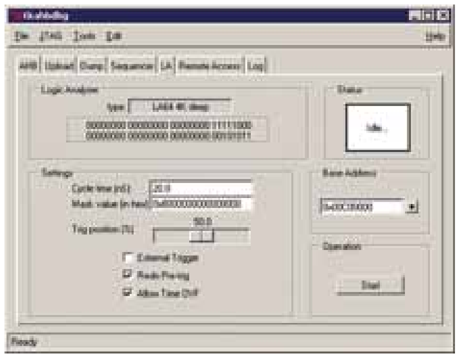


Figure 2 – AHBDBG with logic analyzer page

JTAG debugger only provides those features necessary to try a program with an FPGA on the board.

The AHBDBG communicates with a small AHB master – the jtag2ahb module – through a JTAG interface to generate AHB bus accesses. In addition to obvious features such as bus read/write/dump, two important features are worth mentioning. The first feature is the AHB-based logic analyzer (Figure 2). You may think this is yet another ChipScope™ analyzer, but the logic analyzer available with the AHBDBG is quite different. First, it must be explicitly instantiated. Because of this, you can provide your trigger signal if you need a complex trigger condition. Second, it only saves signals when they change (recording events). This is really necessary to capture bus activities, which may last for many cycles. The tool also allows you to compress much further by sacrificing timing relationship accuracy.

With the optional compression mode turned on, if the timing period between event A and event B exceeds the 16-bit cycle counter, it is truncated to the maximum of the cycle counter. Otherwise, a null event (an event containing no value change information but a time stamp), is generated to guarantee the timing accuracy.

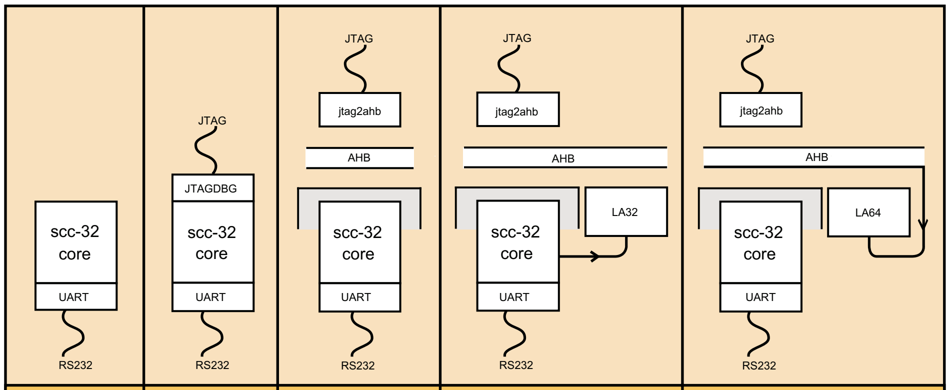
There are four types of AHB-based logic analyzers: 16 bits wide, 32 bits wide, 64 bits wide, and 128 bits wide. You can configure the depth of the logic analyzer trace memory. A 32-bit-wide logic analyzer is sufficient for microsequencer debug, whereas a 64-bit-wide logic analyzer is probably sufficient for AHB bus monitoring (Table 2). The captured events are uploaded to the host side; the AHBDBG produces a VCD dump file so that you can use a simulation waveform viewer to see the waveform. The signals can be bundled to a set of wires and buses with meaningful signal names using a helper tool, vcdwizard.

The second unique feature is the remote access feature. The AHBDBG can be a gateway to your hardware system. When the remote access feature is turned on, it will listen to the network (specified TCP port) and translate a network message to an AHB bus access. This feature allows you to exercise the FPGA hardware using programs such as Python, Perl, or C++.

We provided this feature because our users asked us to include specific features for their projects, and the remote access feature provides a generic way to provide project-specific features. We later found that the remote access feature could be used in other situations beyond its original intent. For example, you could do system prototyping using a remote program before developing a real embedded program.

The AHBDBG has a layered software design; the bottom layer is a layer to talk to a physical device. Currently, it supports Parallel Cable III through a printer port or Ethernet Pod (proprietary to Ponderosa Design). However, it can also support any device that can mimic a printer port interface.

In addition to the devices mentioned previously, a virtual device "sim" is also supported. As the name implies, the "sim" device is a virtual device for Verilog simulation, which the AHBDBG can "control." Any commands given to the AHBDBG are ultimately converted to AHB accesses in a simulation. We found this scheme very helpful, as it provides an intuitive way to run Verilog simulation. The current scheme uses Unix IPC (Inter-Process



Standalone	JTAG Debugger	AHBDBG	AHBDBG + 32-bit LA	AHBDBG + 64-bit LA
1236 LUTs	1367 LUTs	1729 LUTs	2085 LUTs	2437 LUTs
2 BRAMs	2 BRAMs	2 BRAMs	4 BRAMs	6 BRAMs
17%	19%	24%	29%	33% of XC3S400

Table 2 – The scc-32 core size with various debugging options

Communications) and Verilog PLI, so it is not a universal solution for everyone. However, many simulators provide a special hook so that controlling Verilog simulation from the AHBDBG is possible even where Unix IPC is not available.

The AHBWIZARD

When constructing a larger system with the AHB bus, we realized that creating a top-level file that holds all AHB submodules and arranging bus multiplexers is tedious, time-consuming, and error-prone. Moreover, you need to constantly maintain the file as new AHB modules are added or subtracted from the system.

We developed the AHBWIZARD to automate the process such that you can rearrange your AHB system without rewriting the top-level file. For example, you may want to have a logic analyzer module to debug the system, but you don't want to have such modules in the production FPGA design.

The AHBWIZARD separates the AHB library (definition files) and the tool itself (GUI) such that you can add a new AHB module definition file or modify how codes are generated (such as signal naming conventions). At start-up, the AHBWIZARD scans the library directory, enumerates the modules available, and displays them in the window. You can just drag-and-drop the modules you want to add (Figure 3).

A custom property sheet pops up to

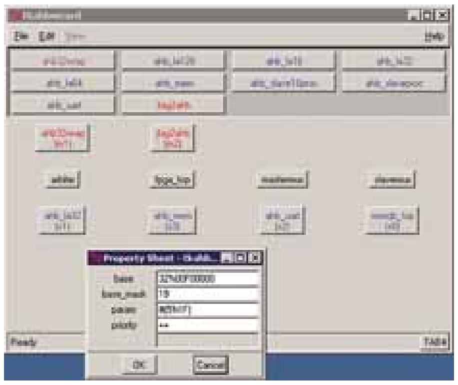


Figure 3 – AHBWIZARD screen shot

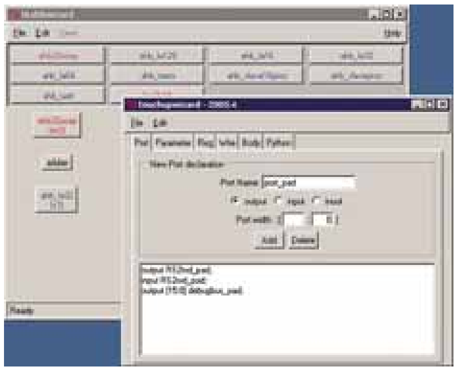


Figure 4 – AHBWIZARD with touch-up wizard screen shot

specify property information about the module, such as address decoding or AHB master priority, which is in turn used in the module's code generation. Glue modules such as AHB slave to master multiplexer are automatically generated accordingly.

If the generated top-level file is not complete, you can use an optional "touch-up" module to add or modify the generated top-level file. A helper tool (touchupwizard) generates the touch-up module (Figure 4).

We provide several commonly used AHB modules for use with the AHBWIZARD, in addition to our AHB-ready microsequencer modules.

Multiprocessors in FPGAs

As the footprints of our microsequencers are small, you could use more than one microsequencer in a FPGA. Although you can assign a different program to each microsequencer, you can also assign the same program to multiple microsequencers. We think that such a configuration – which we call SIMD (single-instruction-stream, multiple-data-stream) configuration – could be very beneficial for certain types of applications.

When controlling a robot, for example, the left and right sides probably require the same control flow. With two microsequencers sharing one program memory, the program only needs to deal with one side, resulting in a simpler program. Additionally, you can use a portion of the program memory for processor communication, as our microsequencers can write to a program memory.

Of course, the required number of block RAMs is half the required number of block RAMs for two separate microsequencers, such that a larger program can be crammed into an FPGA. You can achieve such a scheme with minor core modifications because the block RAM in an FPGA is dual-port memory, allowing two microsequencers to access the program memory without disturbing another (Figure 5).

Multiprocessors in FPGAs is an interesting developing field. As the AHBDBG accesses the microsequencer through the AHB bus, any number of microsequencers can be supported.

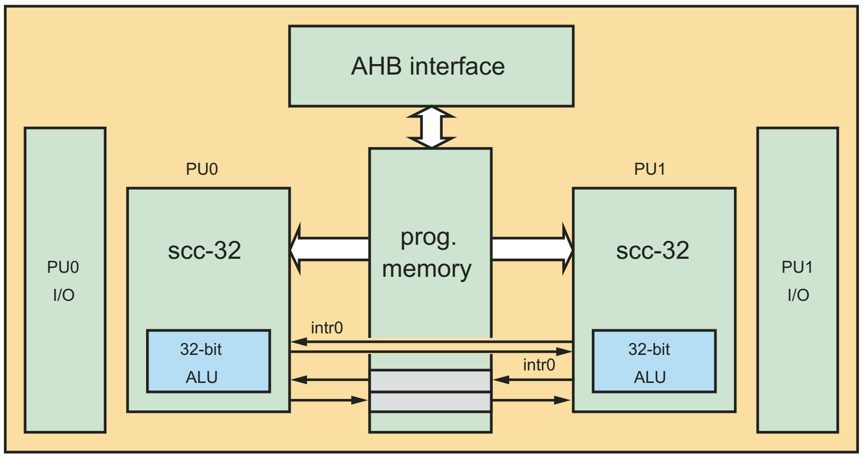
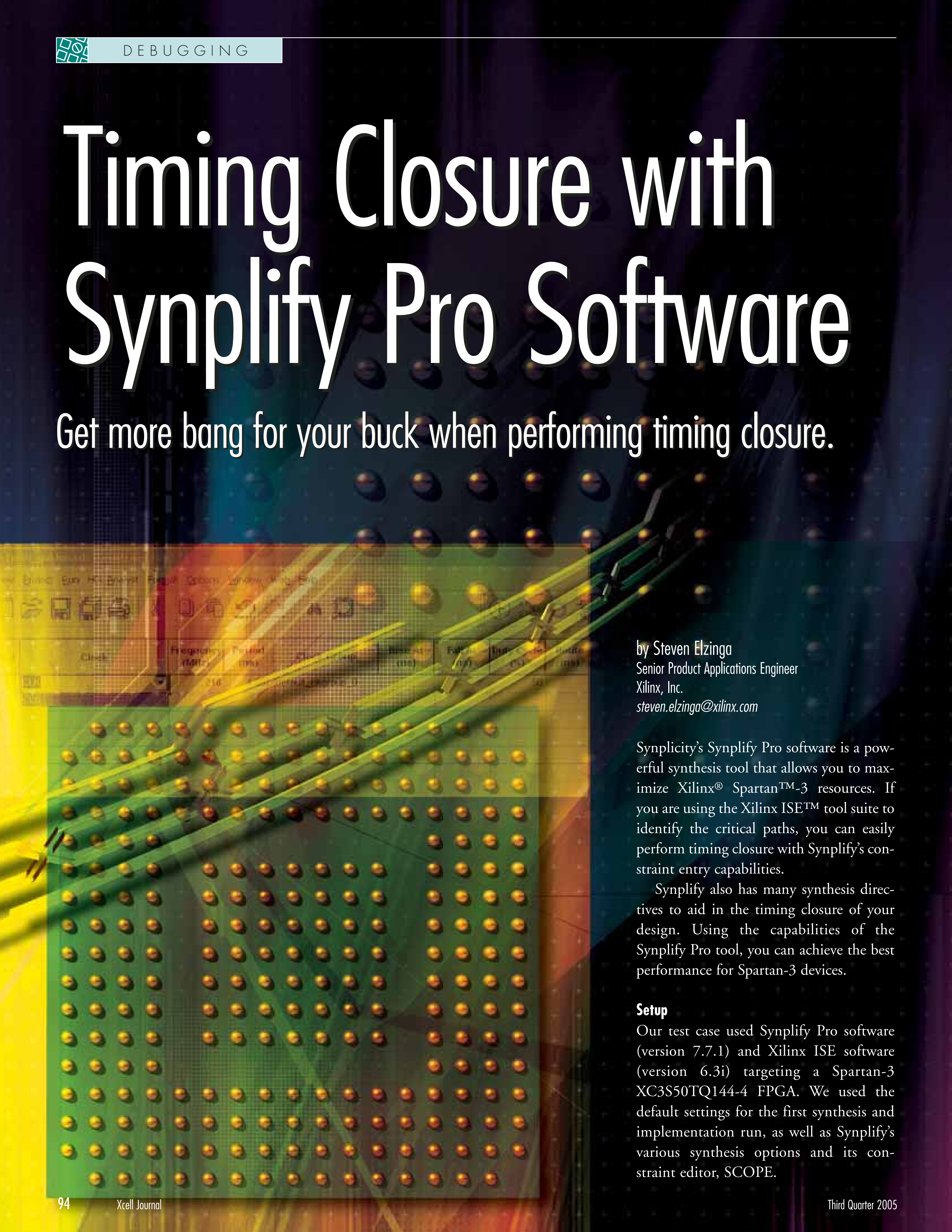


Figure 5 – Multiprocessor scheme with the scc-32

Conclusion

The ideas presented here are used in ASIC design processes, so we are glad to bring these advanced design methodologies to FPGAs. However, this cannot be done without the newest Xilinx FPGA families.

For more information, visit *www.ponderosa-design.com*, or e-mail *info@ponderosa-design.com.*



Timing Closure with Synplify Pro Software

Get more bang for your buck when performing timing closure.

by Steven Elzinga
Senior Product Applications Engineer
Xilinx, Inc.
steven.elzinga@xilinx.com

Synplicity's Synplify Pro software is a powerful synthesis tool that allows you to maximize Xilinx® Spartan™-3 resources. If you are using the Xilinx ISE™ tool suite to identify the critical paths, you can easily perform timing closure with Synplify's constraint entry capabilities.

Synplify also has many synthesis directives to aid in the timing closure of your design. Using the capabilities of the Synplify Pro tool, you can achieve the best performance for Spartan-3 devices.

Setup

Our test case used Synplify Pro software (version 7.7.1) and Xilinx ISE software (version 6.3i) targeting a Spartan-3 XC3S50TQ144-4 FPGA. We used the default settings for the first synthesis and implementation run, as well as Synplify's various synthesis options and its constraint editor, SCOPE.

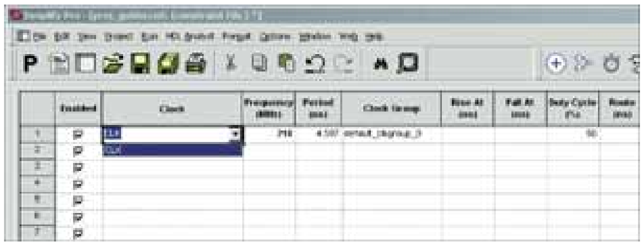


Figure 1 – Period constraint in Synplify Pro software

Because we cannot access SCOPE within ISE software, we created separate Synplify and ISE projects. By using the same project directories for both the Synplify Pro project and the ISE project, the newly created EDIF and NCF files can be immediately available to the ISE tool. The NCF, which has the same syntax as a UCF, is made automatically from the constraints entered through the Synplify tool.

To make sure that the NCF gets used, we selected the Synplify Pro implementation option "Write Vendor Constraint File" on the implementation results tab. The NCF file has to have the same name as the EDIF file so that ISE software will automatically use it.

Modifying Default Settings

Running the design through the Synplify tool with the default settings gave a report with an estimated frequency of 163 MHz and an actual frequency after implementation of 115 MHz. The design was auto-constrained by Synplify Pro software to 191 MHz, which resulted in a 191 MHz period constraint in the NCF. We arbitrarily set a design goal of 190 MHz – so now we need to improve by 75 MHz.

This design has case statements that we can interpret as finite state machines. The FSM Explorer option is turned on, which explores different encoding styles for the state machines and decides on the best implementation. The retiming option allows the synthesis tool to move registers through asynchronous logic so that a more evenly distributed delay is achieved between registers. Selecting both of these synthesis options increased the synthesis-estimated speed of the design to 217.7 MHz.

In the majority of designs, over-constraining causes detrimental results. Without having specified a synthesis period constraint, the Synplify Pro tool auto-constrained the design to 267 MHz after the FSM Explorer and retiming options were selected. As this constraint is much greater than the estimated results, we used a constraint of 218 MHz in SCOPE (Figure 1).

Once the SDC file was created, we added it to the project. With the constraint now added, Synplify Pro software reports a speed of 219.2 MHz. A constraint of 220 MHz gives the same estimated result – 219.2 MHz.

Analysis

With the default settings in ISE software, we achieved a result of 164 MHz. This result is based off the period constraint that Synplify software passed into the NCF. Setting the PAR effort level to high gives a result of 166.7 MHz. Inspecting the report from Timing Analyzer, we see a common critical path through an instance called "s_":

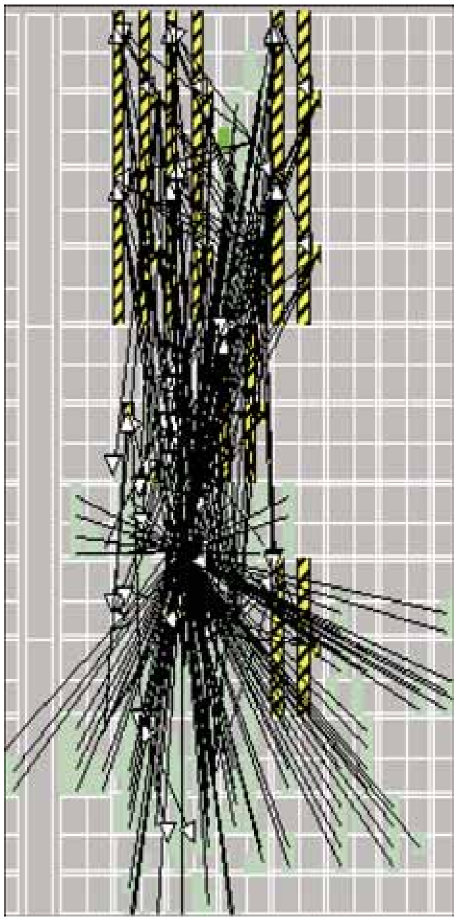


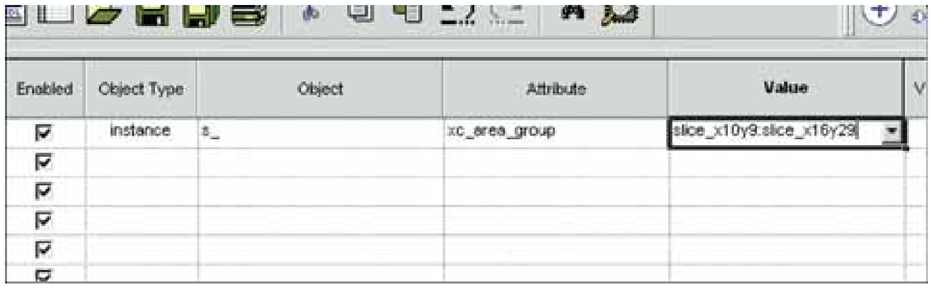
Figure 2 – Connectivity of "s_"

```
================================================================================
Timing constraint: TS_CLK = PERIOD TIMEGRP "CLK"  4.545 nS   HIGH 50.000000 % ;

 1669 items analyzed, 43 timing errors detected. (43 setup errors, 0 hold errors)
 Minimum period is   6.000ns.
--------------------------------------------------------------------------------
Slack:                  -1.455ns  (requirement - (data path - clock path skew +
uncertainty))

:
:

     Location               Delay type         Delay(ns)    Physical Resource
                                                            Logical Resource(s)
-------------------------------------------------  -------------------
SLICE_X21Y27.YQ             Tcko               0.720        s_.SYND2_inv[1]
                                                            s_/inv_/y_259_0_dreg[1]
SLICE_X15Y15.G3             net (fanout=5)     1.731        s_.SYND2_inv[1]
SLICE_X15Y15.Y              Tilo               0.550        N_423
                                                            s_/mult_/z_1_0[1]
SLICE_X15Y15.F4             net (fanout=1)     0.015        s_/mult_/z_1_0[1]/O
SLICE_X15Y15.X              Tilo               0.550        N_423
                                                            g0_i_x2_1_0
SLICE_X12Y16.F2             net (fanout=1)     0.552        N_423
SLICE_X12Y16.X              Tilo               0.608        s_.DATA_OUT_ret_72
                                                            g0_0_x2_0_2
SLICE_X12Y13.BY             net (fanout=1)     1.014        s_.mult_out[1]
SLICE_X12Y13.CLK            Tdick              0.260        s_/LOC2__ret
                                                            s_/LOC2__ret
-------------------------------------------------  -------------------
Total                                          6.000ns  (2.688ns logic, 3.312ns route)
```

Using the actual constraint of 190 MHz decreased design performance after implementation from 189 MHz to 180 MHz.

Floorplanner

With the help of Xilinx Floorplanner, we can determine the connectivity of "s_". In Figure 2, "s_" is selected (yellow). The black lines represent its connectivity in relation to the other parts of the design.



Enabled	Object Type	Object	Attribute	Value
✓	instance	s_	xc_area_group	slice_x10y9:slice_x16y29
✓				
✓				
✓				
✓				
✓				

Figure 3 – Area group constraint

This instance is spread out, so an area group constraint is necessary. Instead of entering the area group constraint into a UCF, we entered the constraint SCOPE, which allows the Synplify tool to make timing decisions based off the physical placement of the logic. In fact, we achieved worse results when we entered the constraints directly into a UCF file (bypassing the synthesis tool).

Area Groups

You can enter the area group constraint through SCOPE by selecting the attributes tab. Each cell in SCOPE has a pull-down menu with only the correct values available. As shown in Figure 3, the "s_" instance is found and the xc_area_group constraint is selected.

After several iterations, we found a good area group constraint that worked well with ISE software, producing a minimum period of 187 MHz. We continued this iterative process, trying area groups on different instances as well as trying different PAR cost tables (a PAR cost table gives a different starting point for the place and route process).

Ultimately, we placed area group constraints on two instances inside of "s_" and used a PAR cost table setting of 8 to get the final minimum period of 189 MHz, close to the arbitrary goal of 190 MHz and a significant improvement over the unconstrained design speed of 115 MHz.

As a final confession, we used the NCF period constraint of 220 MHz. Using the actual constraint of 190 MHz decreased design performance after implementation from 189 MHz to 180 MHz. This is not normal behavior from PAR. Over-constraining in PAR can have the same detrimental results as over-constraining in synthesis, so a design with a positive effect of an over-constrained period in PAR represents a corner case.

Conclusion

Our design was small and easily fit in the smallest Spartan-3 device. However, using the same methodologies outlined in this article, a significantly larger design can meet timing using Synplify Pro's constraints and switches and passing those constraints to ISE software.

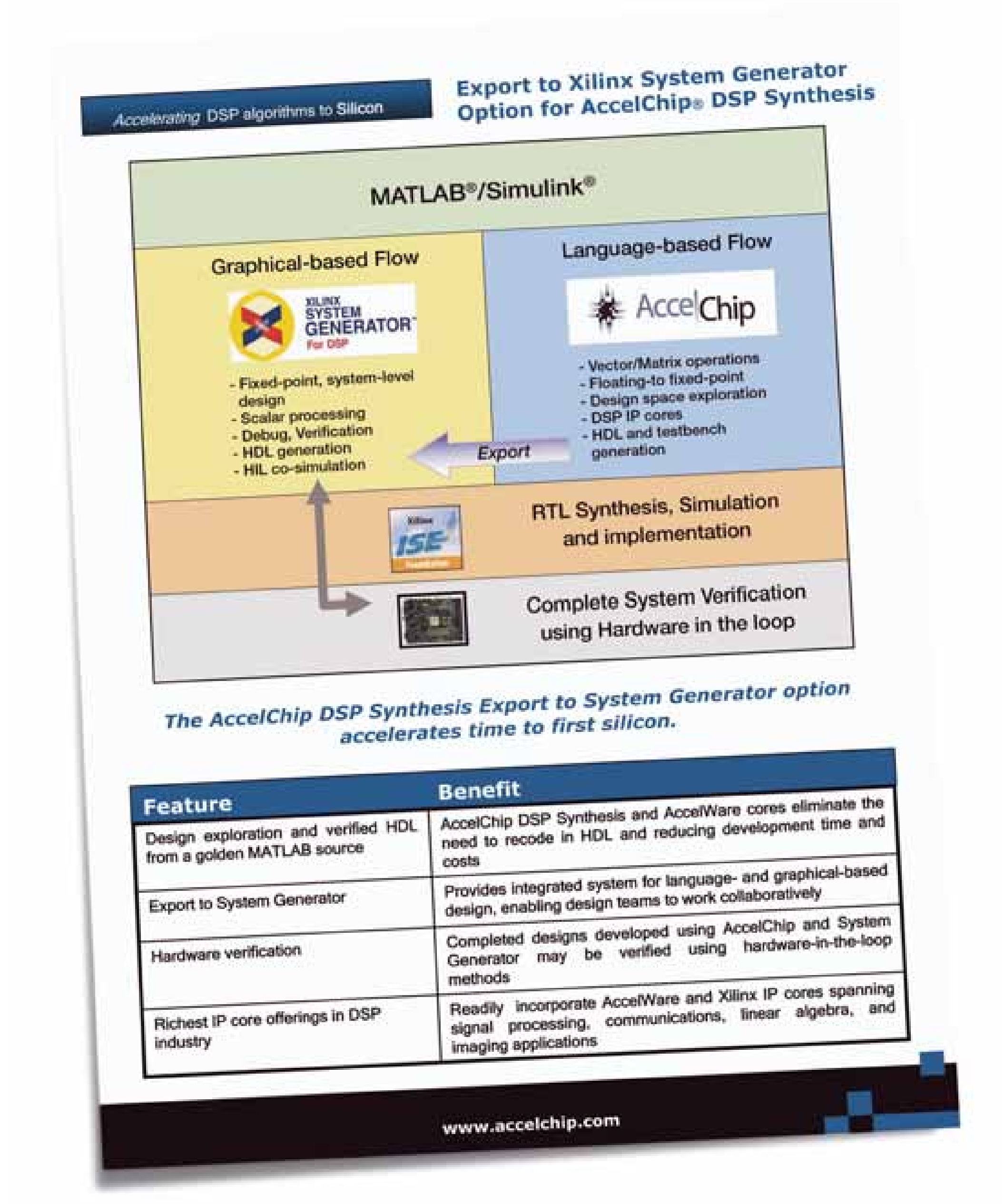
Using the switches in Synplify Pro software, SCOPE, and Xilinx Floorplanner, our design met the arbitrary goal and a significant timing closure on a Spartan-3 design.

For more information, visit *www.synplicity.com/products/synplifypro/index.html* and *www.xilinx.com/spartan3.*

Get the latest Virtex news delivered to your desktop
XILINX April, 2005 Virtex Newsletter
Subscribe Now!
www.xilinx.com/virtex4
XILINX

STOP OVER-THE-WALL ENGINEERING.

(FINALLY, ALGORITHM DEVELOPERS, DSP SYSTEM ARCHITECTS AND HARDWARE DESIGNERS ARE ALL ON THE SAME PAGE)



Accelerating DSP algorithms to Silicon

Export to Xilinx System Generator Option for AccelChip® DSP Synthesis

MATLAB®/Simulink®

Graphical-based Flow

XILINX SYSTEM GENERATOR™ For DSP

- Fixed-point, system-level design
- Scalar processing
- Debug, Verification
- HDL generation
- HIL co-simulation

Language-based Flow

AccelChip

- Vector/Matrix operations
- Floating-to fixed-point
- Design space exploration
- DSP IP cores
- HDL and testbench generation

Export

RTL Synthesis, Simulation and implementation

Complete System Verification using Hardware in the loop

The AccelChip DSP Synthesis Export to System Generator option accelerates time to first silicon.

Feature	Benefit
Design exploration and verified HDL from a golden MATLAB source	AccelChip DSP Synthesis and AccelWare cores eliminate the need to recode in HDL and reducing development time and costs
Export to System Generator	Provides integrated system for language- and graphical-based design, enabling design teams to work collaboratively
Hardware verification	Completed designs developed using AccelChip and System Generator may be verified using hardware-in-the-loop methods
Richest IP core offerings in DSP industry	Readily incorporate AccelWare and Xilinx IP cores spanning signal processing, communications, linear algebra, and imaging applications

www.accelchip.com

Company partnerships don't usually generate a lot of interest, but a partnership that yields team-building technology does. For example, the industry's first DSP design flow from a mixed MATLAB/Simulink environment to a verified FPGA. It's based on a tight integration between AccelChip DSP Synthesis and Xilinx System Generator for DSP that accelerates development of signal processing systems. Work more collaboratively and get designs to market faster by eliminating the need to recapture in HDL. Readily incorporate AccelWare and Xilinx IP cores and verify the entire system using hardware-in-the-loop simulation, further speeding development. How's that for teamwork? **Stop your Over-the-Wall engineering. Download free whitepapers at accelchip.com/papers or call (408) 943-0700.**

XILINX® AccelChip

Speed FPGA Debug with Mixed-Signal Oscilloscopes

You can make internal measurements using Agilent scopes and the Xilinx ChipScope Pro analyzer.

by Joel Woodward
Senior Product Manager
Agilent Technologies
joel_woodward@agilent.com

Digital designers have long reached for an oscilloscope for debug. As FPGAs have become the centerpiece of digital design, the need to quickly debug systems that include programmable logic is stronger than ever. However, traditional oscilloscope technology has not kept up with the functional debug of FPGAs. A new breed of oscilloscopes known as mixed-signal oscilloscopes (or MSOs for short) delivers vital capabilities if you are developing systems with FPGAs.

Like traditional oscilloscopes, MSOs offer the same rich feature set for taking parametric measurements to measure signal integrity, jitter, and signal characterization. You can choose between versions that have either two or four analog input channels. MSOs come in a variety of bandwidths ranging from 300 MHz to 1 GHz bandwidth; these capabilities are important for checking signal parametrics. For example, you can readily change I/O standards and drive strengths using Xilinx® FPGA Editor and measure the real-world I/O characteristics using an MSO's scope channels.

The major difference between MSOs and traditional DSOs (digital storage oscilloscopes) is the addition of 16 digital asynchronous sampling channels on the MSO. Plus, you can choose how fast these digital channels sample. The digital channels offer deep memory storage independent of the analog channel memory storage. You can employ the capabilities of the digital channels in a number of different ways that are particularly valuable for developing systems that incorporate FPGAs.

Bus Triggering and Display

Traditional oscilloscopes provide digital triggering capabilities that allow them to trigger on patterns across analog channels. With a four-channel oscilloscope, you can trigger on a single pattern that is as wide as four signals.

Debug often requires looking at buses using a specific event as the trigger condition. Using an MSO's digital channels, you can trigger on a digital pattern as wide as 16 signals. This can be a powerful capability if you need to look at a state machine, an embedded microcontroller, or a data bus. In addition, you can also trigger and capture measurements across all four analog channels, extending the trigger width up to 20 signals, as shown in Figure 1.

Correlating Analog and Digital Measurements

Although you can use the digital channels to make strictly digital measurements, their capabilities are best employed for looking at problems that are both functional and parametric in nature – for example, triggering on a digital bus and having this trigger condition arm the scope measurement.

At Agilent, one of our design teams experienced an infrequent software glitch on an embedded product under development. This anomaly manifested itself very infrequently – about once a week. The software team developed diagnostics that caused the problem to happen on a more frequent basis. With this diagnostic software, they found that the problem occurred during read cycles on a PCI bus embedded in an FPGA.

The team routed PCI status signal out to pins and connected the MSO's digital channels to these signals. Engineers quickly set up the MSO digital trigger on a PCI read cycle and then set the MSO scope channels to acquire when the MSO digital channels recorded a PCI bus read cycle.

With the ability to trigger on a specific bus cycle, the team was quickly able to resolve the problem. They found a clock with a too-slow rise time that impacted primarily read cycles. The team modified the design and downloaded a new configuration file into the FPGA. The combination of reprogrammable FPGAs and MSO measurements allowed the design team to fix the problem and ship the product on schedule.

Extensive Internal Visibility

To access internal signals, you might typically use the route-out approach to bring signals to pins that can be probed using an oscilloscope. Using traditional oscilloscopes, you would have access to either two or four signals at a time. This narrow signal visibility can complicate debug, as a number of problems require simultaneous visibility across a higher number of signals. To access new signals, you must change the design, re-synthesize, and run a new place and route to make your signals accessible to the oscilloscope. This process can take hours.

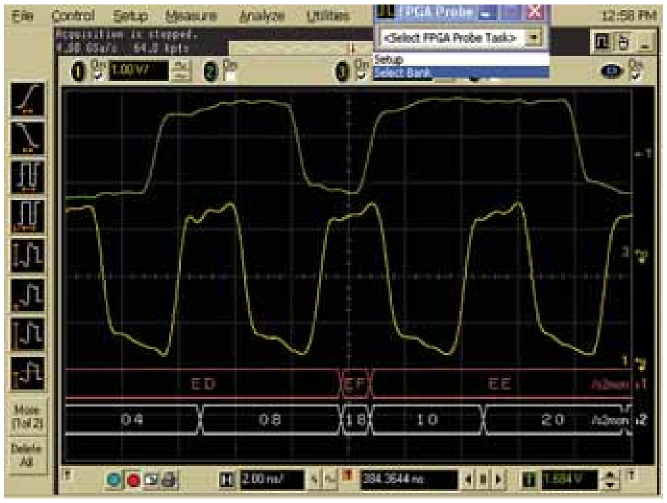


Figure 1 – In addition to scope channels, mixed-signal oscilloscopes provide 16 digital channels. This gives you the ability to trigger and display as many as 20 signals simultaneously. The width of the digital channels is particularly well suited for measuring signals internal to Xilinx FPGAs.

With the digital channels of an MSO, you have visibility of as many as 16 internal FPGA signals at a time. The power of the MSO's digital channels can be further extended when combined with on-chip technologies such as the Xilinx ChipScope™ Pro tool and Agilent FPGA Dynamic Probe.

The ChipScope Pro analyzer allows design teams to incorporate an Agilent debug core in their FPGA designs. The debug core, known as ATC2, provides an easy way to route signals to pins, enables a faster setup of the MSO, and allows you to quickly measure new groups of internal signals. This capability extends the reach of the 16 digital channels into the FPGA design.

Let's look at a simple communication system to illustrate the value of an MSO's digital channels for FPGA debug. A state machine drives the process of sending out packets of 16-bit data along with a transaction ID. Parallel data is serialized, sent on a serial channel, de-serialized, and brought into a monitor. A second state machine at the monitor drives the process of receiving the packets and strips off the data and transaction IDs so that the packets can be pulled off to external memory. This state machine also generates acknowledge IDs that are fed back to the transmit side to communicate that data was received. The design originally dedicated 16 pins for a debug port.

Using the ChipScope Pro core inserter, you can parameterize an ATC2 core. A benefit of the core inserter is that you need not modify the original HDL design, as core insertion occurs post-synthesis and before place and route. You would simply specify, using the core inserter, which internal signals to group together as an active signal bank. Place and route uses the original user constraint file, so no additional work is required.

The core inserter also produces a small file that contains the specified signal names and groups. This file, known as a .cdc, is read by the FPGA Dynamic Probe application running on the MSO. When changing which signal group is presented to the MSO, the instrument automatically uses this signal naming file to correctly update signal names on the display. As opposed to the route-out approach, which can take hours to bring new signals to pins, the FPGA Dynamic Probe allows you to access a new group of internal signals in about one second.

For fast debug, the design team needed visibility into four sections of the design. The designer thus created a core with four signal banks to give access to 64 signals, 16 at a time, over the 16-pin debug port (as shown in Figure 2). ATC2 cores can be parameterized to have as many as 64 signal banks, providing access to 1,024 internal signals with the MSO's 16 digital channels.

Timing Cores

You can configure ATC2 cores as either timing (asynchronous) or state (synchronous) cores; both types of cores are supported with the MSO. The core inserter injects a core into a design post-synthesis and before place and route. If you specified a timing core, the place and route tools do not put any flops between the signal being probed and the output pin; the routing of the signal to pin for measurement is treated as a false path. This allows the place and route tools to ignore any speed constraints associated with routing a specific signal to a pin.

The timing core does include a JTAG controller, but the controller typically runs very slowly (<5 MHz), as it is only used for small information exchanges such as selecting a new signal bank. Timing cores can be effective, as they allow you to look at signals across multiple clock domains or at anomalies that have a duration of less than one clock cycle. The primary trade-off associated with timing cores is that skew will exist between signal paths.

State Cores

Traditional oscilloscopes as well as MSOs provide asynchronous acquisition. Samples are stored using an adjustable clock reference internal to the scope. This can make it difficult to accurately capture and decipher synchronous events as the instrument captures invalid transitions between clock cycles.

A more effective way to capture synchronous information on a single clock domain is to parameterize the ATC2 core as a state core. A state core will have minimal impact on design timing because of its pipelined architecture. A total of four flops are placed between the signal being probed and an output pad (Figure 3). The design tools place the first flop as close as possible to the signal being probed. The additional three stages of pipelining allow the signal three clock cycles before reaching the output pad. The pipelined architecture of the ATC2 core allows the place and route tools to have a much greater probability of meeting the original timing goals of the design.

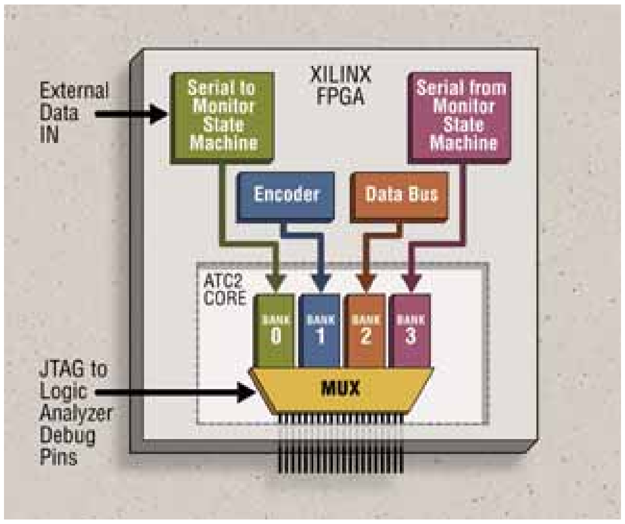

External Data IN
XILINX FPGA
Serial to Monitor State Machine
Serial from Monitor State Machine
Encoder
Data Bus
ATC2 CORE
BANK 0
BANK 1
BANK 2
BANK 3
MUX
JTAG to Logic Analyzer Debug Pins

Figure 2 – ATC2 cores, inserted using the ChipScope Pro tool, allow you to quickly switch which signals are connected to pins for measurement. Each ATC2 core can be parameterized with as little as one signal bank or as many as 64 signal banks.

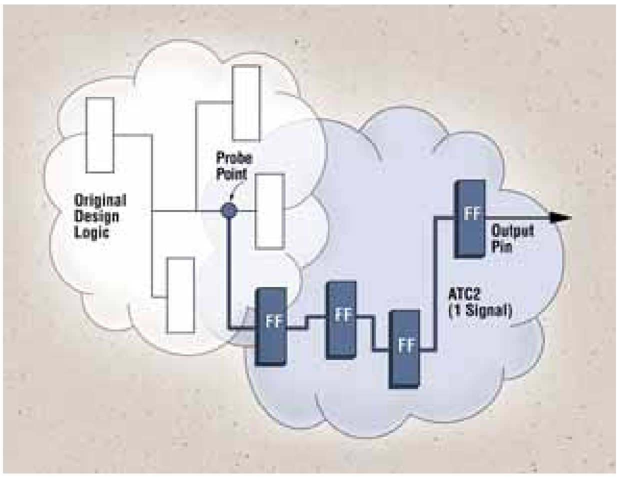


Figure 3 – A pipelined architecture uses four stages to route a signal to the debug port using the FPGA Dynamic Probe. This automated approach gives the place and route tool flexibility to meet timing requirements because the router can use timing solely within the ATC2 core to move across the chips.

As the core is synchronous, the place and route tools eliminate skew between signal paths. The primary trade-off with a state core is that it works with a single time domain. Using the state core approach, you can measure across clock domains by inserting multiple state cores. The MSO can access multiple ATC2 cores, one at a time, in a single FPGA or distributed across multiple FPGAs on a single scan chain.

The MSO's digital channels provide exclusively asynchronous acquisition. For FPGA debug, a method exists for allowing the MSO to display synchronous measurements, even though the initial acquisition occurs asynchronously. The ATC2 state core outputs a clock signal and signal states synchronous with the clock. The MSO's digital channels acquire this pre-formatted state information. Then the MSO post-processes this measurement using a state display feature that allows you to specify one signal as the clock. The MSO filters out to all transitions between valid states. This makes it possible to make synchronous measurements internal to the FPGA.

Conclusion

The reprogrammable nature of FPGA technology makes rapid iterative real-world debug a great companion to simulation. As FPGAs become even more sophisticated, the need for efficient internal visibility increases. Mixed-signal oscilloscopes provide unique measurement capabilities that align with the needs of those designing systems that incorporate FPGAs. Applications that help you exploit the digital measurement capabilities of MSOs are a catalyst for shorter development cycles and higher quality designs.

For more information about the FPGA Dynamic Probe for Agilent MSOs, visit *www.agilent.com/find/msoFPGA.*

Designer Challenges for Pb-Free and Green Products

Ease the transition to green solutions with Xilinx Pb-free products.

by Paula Ungs
Sr. Marketing Manager
Xilinx, Inc.
paula.ungs@xilinx.com

Many challenges are associated with designing products that are not only safer for the environment but meet the legislative and OEM requirements that are being adopted around the world. To ensure success, you must understand the global requirements and deadlines for the removal of hazardous substances from electronic products. At a more practical level, you must understand how Pb-free devices impact board design. Xilinx® understands these challenges and offers a strategy to ease the transition to Pb-free and environmentally friendly products.

Global Environmental Requirements

To develop products that can be shipped globally, you must comply with global requirements and deadlines for the removal of Pb and other hazardous substances from electronic products. As shown in Figure 1, the most eminent requirement is the European Union (EU) directive for the restriction of the use of certain hazardous substances (RoHS). This directive bans shipments of electronic products into the EU after July 1, 2006 that do not comply with the directive's restriction on Pb, mercury, cadmium, hexavalent chromium, PBB, and PBDE flame retardants.

Japan recognizes RoHS and also enforces their own green initiatives, driven by recycling/reuse laws in their country. China recently announced plans to adopt legislation that will be more strict than the RoHS legislation – banning, rather than restricting, the named substances by the same July 1, 2006 timeframe. Individual states in the U.S. are also considering bans on the same list of materials – and potentially others.

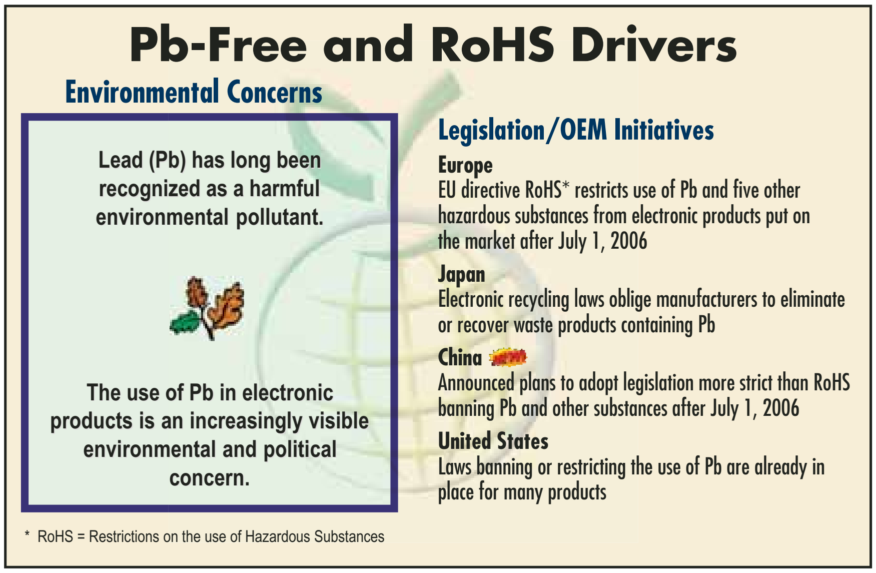
Pb-Free in Board Designs

How do Pb-free solutions impact board designs? As a rule of thumb, Pb is distributed in electronics in the solder (75%), board (20%), and components (5%), so you must understand the implications to the overall solution. "Green" components in the industry feature new material sets that require changes to solder materials and processes, which in turn affect board selection. The good news is that most Pb-free packages have the same form, fit, and function as standard packages; no special board layout considerations are required from the component side.

Pb-free lead-frame packages (for example, PQG and TQG) that use matte tin plating on the leads are of little concern, since they are considered "backward compatible" to traditional SnPb manufacturing processes. However, many Pb-free BGA package solutions in the industry use a SnAgCu solder ball material that is not backward compatible with traditional manufacturing processes. The SnAgCu material requires higher peak reflow temperatures than those specified for standard packages. This situation makes it difficult for designers to mix Pb-free and standard components on the same board. So the decision must be made up-front as to whether the board will be Pb-free or not.

The BOM for Green Products

Crafting the bill of material (BOM) can be difficult when designing Pb-free solutions. Pb-free devices cannot always be mixed with standard devices on the same board because of differences in the manufacturing process. Therefore, you need to know if a given device is Pb-free or standard when including it in a board design. What makes


Pb-Free and RoHS Drivers

Environmental Concerns

Lead (Pb) has long been recognized as a harmful environmental pollutant.

The use of Pb in electronic products is an increasingly visible environmental and political concern.

Legislation/OEM Initiatives

Europe
EU directive RoHS* restricts use of Pb and five other hazardous substances from electronic products put on the market after July 1, 2006

Japan
Electronic recycling laws oblige manufacturers to eliminate or recover waste products containing Pb

China NEW
Announced plans to adopt legislation more strict than RoHS banning Pb and other substances after July 1, 2006

United States
Laws banning or restricting the use of Pb are already in place for many products

* RoHS = Restrictions on the use of Hazardous Substances

Figure 1 – Environmental concerns drive global legislation.

this complex is that some companies in the industry are simply "converting" standard devices to Pb-free. And because the part number does not change in this scenario, the part type cannot be easily identified.

And what about the BOM cost? Many designers are challenged to create cost-effective solutions that can compete in a tough market. Many component companies in the industry are applying cost adders to Pb-free versions of their products to offset the higher costs they may be experiencing. What was already a difficult constraint is now exacerbated by unexpected higher costs.

Availability and Reliability

As a designer, you surely want to know that the parts designed onto a board are available and reliable. You should feel comfortable that the parts you have selected will be available when the board is prototyped or built in production. Component manufacturers in the industry are getting up to speed with green products at different rates, so the simultaneous availability of components is hard to predict.

You will also want to make sure that Pb-free parts are high-quality, reliable devices. Because material sets have changed, it is imperative that the new devices are fully qualified and that the packages can withstand the higher reflow temperatures now required.

The Transition to Pb-Free

To ensure that you can easily develop solutions that support green requirements, Xilinx is committed to providing RoHS-compliant products to the market ahead of all directives. Xilinx also stays active in industry consortia to ensure advanced knowledge of any new environmental legislation that may affect electronic products. Xilinx green components will comply with all global requirements.

Pb-free packages from Xilinx have the same form, fit, and function as standard packages. Xilinx and its suppliers have carefully selected material sets for Pb-free components; these material sets are among the most commonly used in the industry. Xilinx Pb-free packages use matte tin plating for lead-frame packages and SnAgCu solder balls for BGA and flip-chip packages. All Pb-free packages from Xilinx are fully RoHS compliant, and in most cases are available at no additional cost over standard products (see Table 1).

To simplify Pb-free part identification, Xilinx instituted a special part number (Figure 2). You can easily designate and identify Pb-free products, thus saving time and effort. Furthermore, Xilinx adopted a "dual-line" strategy, which makes both standard and Pb-free products available in parallel so that companies can transition at their own pace.

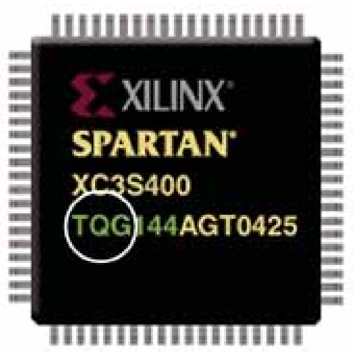


Figure 2 – Xilinx Pb-free part numbers include an additional "G" character.

Conclusion

When designing with Xilinx Pb-free products, you can be assured that all products are qualified and as highly reliable as standard products. Package materials were enhanced to withstand the thermal stress from higher reflow temperatures associated with new Pb-free materials.

For more information regarding quality and reliability, read the Device Reliability Report on the Xilinx website, *www.xilinx.com/quality*. For more information about Xilinx Pb-free products, visit *www.xilinx.com/pbfree*.

Package Type	Pb-Free Materials	RoHS Compliant?	Enhanced for Higher Temps?	Change to Form, Fit, Function?	Compatible w/SnPb Solder?
Lead Frame	100% Matte Tin Plating on Leads	Yes	Yes	No	Yes
BGA and Flip-Chip	Tin-Silver-Copper (Sn4Ag 0.5 Cu) Solder Balls	Yes	Yes	No	Not Recommended

Table 1 – Xilinx Pb-free package information

A Shortcut to Effective Hierarchical Designs

The PlanAhead QuickStart! service enables you to rapidly obtain optimal FPGA performance with minimal device utilization.

by Jonathan Trotter
Titanium Business Development Manager
Xilinx, Inc.
jonathan.trotter@xilinx.com

FPGAs were once used in smaller, simpler applications. A small team could create designs that reached desired performance levels with some HDL knowledge and minimal FPGA fabric experience. With larger densities, ever-increasing features, and threatening competition, new design techniques are necessary to predictably achieve maximum device performance and shorten development time.

You can apply some of these new design techniques by using constraints within Xilinx® ISE™ software, entering constraints using several tools (or manually with a text file). These older methods work, but the chance of making a mistake is high. Each time you make a modification using these traditional tools, you are forced to wait for another place and route (PAR) run. In the end you might find out that the changes were ineffective, and now you are stuck in an endless loop trying to close on timing.

ASIC designers have been taking full advantage of hierarchical design methods for years. Now it's time for FPGA designers to learn how to use the PlanAhead environment and follow new design techniques and requirements.

With enough effort and iterations, you will eventually have an idea of what it takes to achieve timing closure. But how much time will you need – and how many iterations will be required – before you get the design and constraints to converge on timing? Is it a matter of refining the constraints, placement, or changing the RTL structure?

When using a flat methodology to make even minor changes to a given logic block, you must redo PAR for the entire design. This adds up to a significant amount of wasted time, as 50 or more PAR iterations – at 8 or more hours apiece – are common with today's larger FPGA netlists.

The PlanAhead QuickStart! service delivers individualized service that includes a QuickStart! application engineer at your site for a week. This Xilinx expert will train design teams on the PlanAhead™ hierarchical design environment, which allows designers to create block-based incremental designs, run timing analysis, reduce PAR times, create reusable modules, and group fabric for optimal routing and consistent results. After a two-day training course on PlanAhead design tools, the QuickStart! engineer will then provide support and consultation customized specifically to address your design requirements.

Benefits of Floorplanning

Two important but opposing features of a great design are obtaining maximum speed and minimal device utilization. A design with a highly optimized HDL netlist can still fail to meet timing with non-optimal logic placement. Each time a design undergoes PAR, the internal logic can move or shift because placement is not predictable. As a result, performance can change dramatically with each PAR run. Synchronous elements placed in a scattered fashion can slow timing because of routing delays, but these can be dramatically reduced – and device utilization minimized – by closely grouping the related logic. Timing-driven floorplanning helps you reach the highest speeds while using a minimal amount of FPGA fabric, saving more fabric for extra features and options.

Benefits of Hierarchical Design

The PlanAhead QuickStart! service rapidly enables design teams to utilize a hierarchical design methodology and increase design performance. This methodology enables the design to be broken up into separate hierarchical blocks or modules. Once PAR is locked in for each block, placement and routing between the blocks is performed. The less routing delay between the internal sub-modules of a block and between blocks, the more predictable timing closure is for the overall design.

Each separate block can undergo PAR by different team members. They will also have the control to make changes to their blocks independently of their teammates. Instead of making changes to the design as a whole, individual team members can make small incremental changes to portions of the design. This approach increases productivity because it reduces the total number of time-consuming PAR runs. The modular design flow also reduces the time needed for each PAR run. Isolating the problem to specific block(s) eliminates the extra PAR iterations usually required when working with a flat methodology.

PlanAhead Design Tools

Historically, it was difficult to follow hierarchical design methodologies with older FPGA design tools. Several software applications or design tools were needed at various stages in the design implementation. The hierarchical design planning capability of PlanAhead design tools includes an advanced user interface, making it easy to use. Multiple views highlight the resources, connectivity, and logical and physical hierarchy, enabling design teams to quickly inspect and rectify problem areas. They can also create and manipulate physical hierarchy independently from logical hierarchy. The tool is powerful and allows designers to simultaneously plan and analyze multiple physical implementations.

Benefits of PlanAhead QuickStart!

ASIC designers have been taking full advantage of hierarchical design methods for years. Now it's time for FPGA designers to learn how to use the PlanAhead environment and follow new design techniques and requirements. The QuickStart! engineer will provide a two-day Designing with PlanAhead course, followed by three days of on-site customized support and consultation. The engineer will be familiar with PlanAhead fundamentals, as well as other design aspects that may need consideration. After a week of dedicated support, your team will be familiar with fundamentals of modular and block-based design, working at an expert level with the PlanAhead hierarchical design environment.

Conclusion

Many designers are comfortable with a push-button flow, which is defined by simply writing and synthesizing HDL and using Xilinx implementation tools without special options or design constraints. Most design teams can obtain desired results with this flow. But for designers creating powerful applications, floorplanning and hierarchical block-based design techniques are essential. The PlanAhead hierarchical design environment allows you to design with a new powerful methodology. The PlanAhead QuickStart! service enables you to reap the benefits in only a week. To find out more about PlanAhead QuickStart!, contact your Xilinx representative or visit *www.xilinx.com/paq*.

Xilinx Education Services: Knowledge Creates Performance

FPGA design courses provide the necessary strategies and techniques to increase performance.

by Rhett Whatcott
Senior Engineer/Course Developer
Xilinx, Inc.
rhett.whatcott@xilinx.com

Every designer ultimately needs to finish their design on time and on budget while meeting performance requirements. Xilinx® courses provide you with the necessary knowledge to deliver your projects predictably and within budget.

Xilinx Education Services provide six curriculum paths that focus on a specific area of design specialization: Languages, FPGA Design, PCI Design, DSP Design, High-Speed Design, and Embedded Design. In this article, we'll explore the FPGA Design path and show how each course builds on itself, from fundamentals to achieving high-performance design objectives.

For example, our FPGA series of courses offers you a successful step-by-step timing closure strategy. In addition to increasing performance, this has the added benefit of reducing power because of less routing. All of our logic courses offer you best practices, tips, and techniques for utilizing Xilinx ISE™ software, cores, and architectural features. The following course descriptions describe some specific design challenges that you will tackle.

Fundamentals of FPGA Design Course

Fundamentals of FPGA Design covers basic hardware and software features, coding tips, design performance, and reliability.

Does your in-circuit functionality change from one implementation to the next? Are you struggling with reliability issues? Through concise asynchronous and equivalent synchronous design examples, this course will show you how to avoid circuit reliability pitfalls and replace them with reliable synchronous circuits. Synchronous design techniques provide reliability, structure, and lay the foundation for increasing performance.

Do you struggle with creating early pin constraints for your FPGA? Using PACE (the Pinout Area and Constraint Editor) for creating pin constraints, you can take advantage of the FPGA fabric rather than creating a bottleneck in performance.

Wouldn't it be nice to have complex cores already created and optimized for Xilinx FPGAs? You will use the Xilinx Architecture Wizard to optimize cores for instantiation.

You will learn to:

- Use Xilinx ISE Project Navigator to implement an FPGA design
- Read suitable reports to determine whether design goals were met
- Assign pin locations with PACE to enhance performance
- Configure a clocking scheme with the Architecture Wizard software
- Use the Constraints Editor to create global timing constraints, which drive performance
- Specify software options that can boost performance
- Use synchronous design techniques to create reliable designs and improve performance

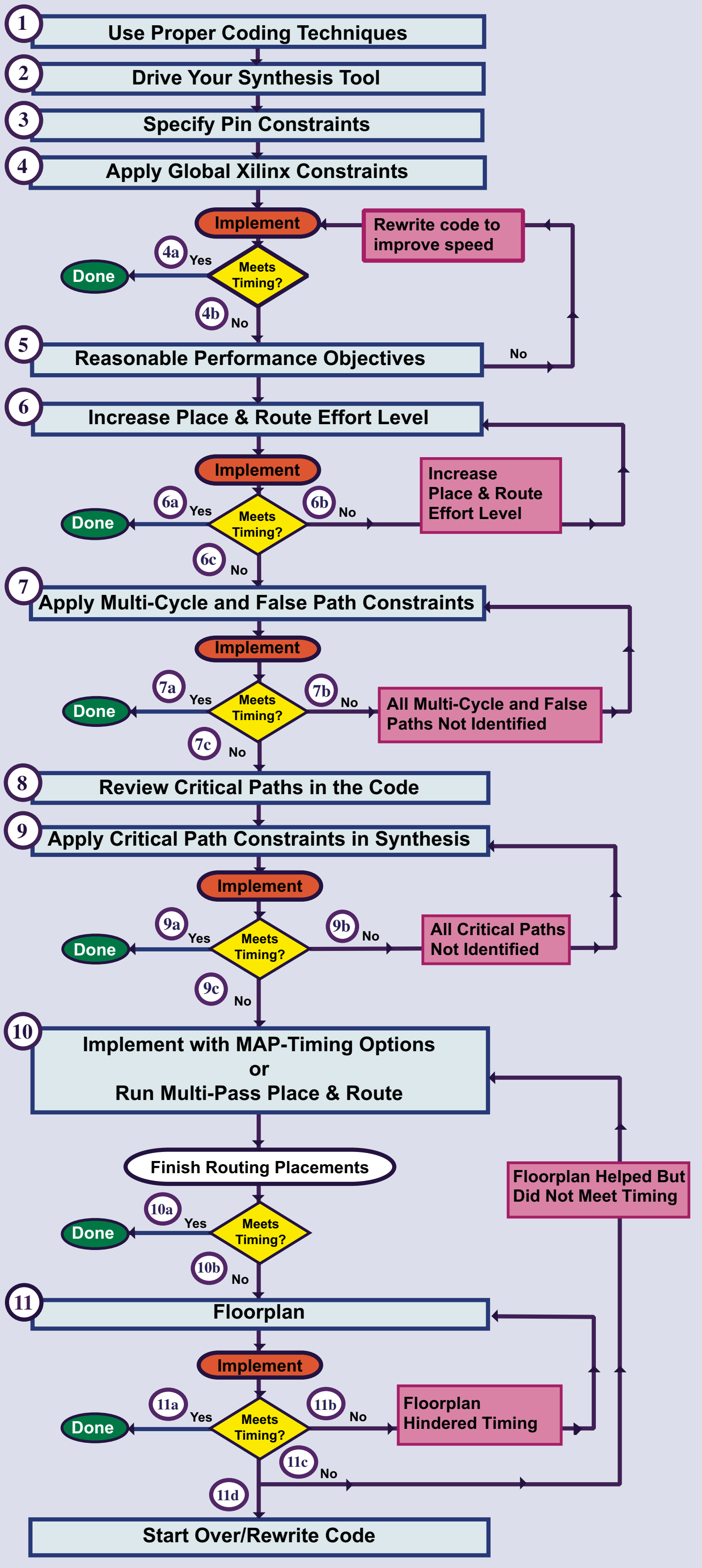
The Fundamentals of FPGA Design course provides the necessary foundation to begin using Xilinx FPGAs, create reliable designs, improve area, and increase performance. The prerequisites for this course are basic logic/digital design knowledge and basic VHDL or Verilog knowledge.

Designing for Performance Course

Are you having trouble meeting your performance goals? Do you find that changing software options doesn't help – or makes matters worse? In the Designing for Performance course, we will teach you a simple timing closure flow (see Figure 1), as well as which options to use and what to expect.

For example, say that you need to increase your performance by 25%. What should you do? Using the techniques taught in this course, you can take a design that can barely achieve 125 MHz performance and by the end of the course increase performance to 200 MHz – an increase of 38%.

Are you having trouble synchronizing an incoming signal? Do you need to cross data from one clock domain to another? In



1 Use Proper Coding Techniques
2 Drive Your Synthesis Tool
3 Specify Pin Constraints
4 Apply Global Xilinx Constraints
Implement
Rewrite code to improve speed
4a Yes
Done
Meets Timing?
4b No
5 Reasonable Performance Objectives
No
6 Increase Place & Route Effort Level
Implement
Increase Place & Route Effort Level
6a Yes
6b No
Done
Meets Timing?
6c No
7 Apply Multi-Cycle and False Path Constraints
Implement
7a Yes
7b No
Done
Meets Timing?
All Multi-Cycle and False Paths Not Identified
7c No
8 Review Critical Paths in the Code
9 Apply Critical Path Constraints in Synthesis
Implement
9a Yes
9b No
Done
Meets Timing?
All Critical Paths Not Identified
9c No
10 Implement with MAP-Timing Options or Run Multi-Pass Place & Route
Finish Routing Placements
Floorplan Helped But Did Not Meet Timing
10a Yes
Done
Meets Timing?
10b No
11 Floorplan
Implement
11a Yes
11b No
Done
Meets Timing?
Floorplan Hindered Timing
11c No
11d
Start Over/Rewrite Code

Figure 1 – Timing closure flow chart

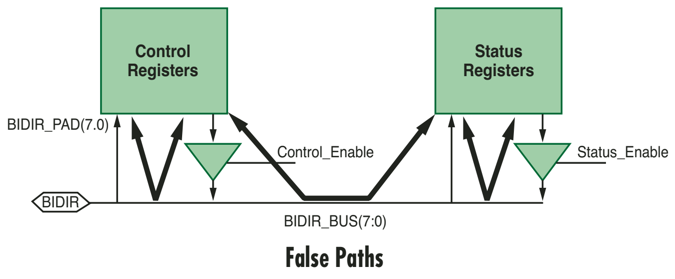


Figure 2 – False paths through bi-directional pad

this course, we will further explore synchronous design techniques, teaching you how to pipeline, duplicate high-load signals, and use synchronization circuits.

Do you find that your synthesis software settings do not always accomplish what you would like? Are you not certain what each option does? Utilizing synthesis settings, you can increase the lab design's performance by as much as 20%.

Does your design have multi-cycle and false paths? By correctly constraining the lab design with multi-cycle and false path constraints (see Figure 2), you will find that the design runs 20% faster than originally noted. Correctly constrained, your design may benefit by even greater amounts.

How do you find failing timing constraints – and how do you fix them? Understanding how to use the information provided by Xilinx Timing Analyzer is the central catalyst to increase the course's lab design performance by 38%.

How do you use the myriad of implementation settings to improve results? Do you need to increase your performance by 2%, 10%, or more? You will identify which advanced implementation options provide the most benefit for each situation.

You will learn to:

- Write HDL code to efficiently target Virtex™-II-based device resources
- Create customized and optimized cores in CORE Generator™ software
- Use Timing Analyzer to pinpoint timing errors and identify a strategy to improve performance
- Use design/coding techniques and software options to achieve timing closure
- Correctly and completely constrain your design with global and path-specific timing constraints in the Xilinx Constraints Editor
- Improve design performance and manage software runtime by using the optimal software settings

The knowledge obtained from the first two courses in this track will provide you with the necessary knowledge to tackle

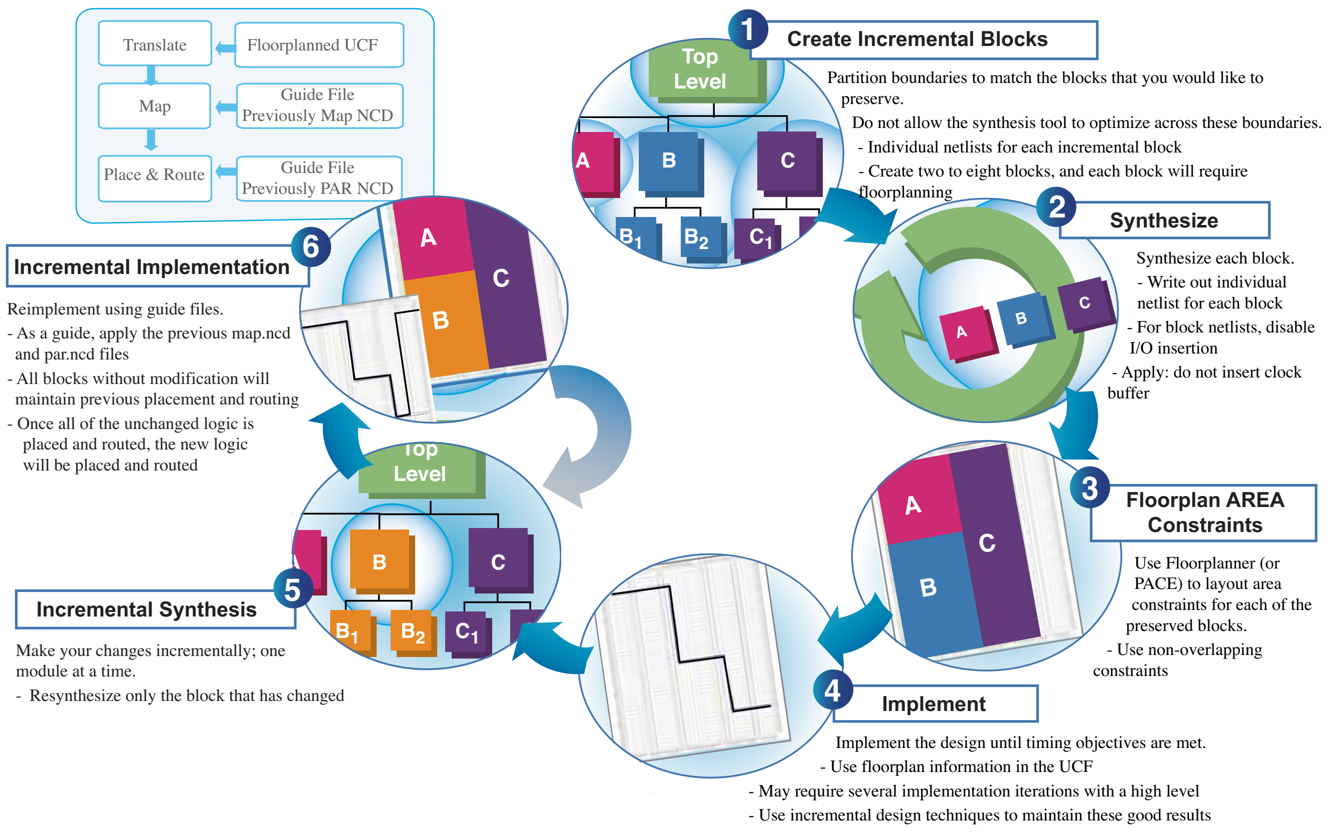

Translate
Floorplanned UCF
Map
Guide File Previously Map NCD
Place & Route
Guide File Previously PAR NCD
1
Create Incremental Blocks
Top Level
A
B
C
B1
B2
C1
Partition boundaries to match the blocks that you would like to preserve.
Do not allow the synthesis tool to optimize across these boundaries.
- Individual netlists for each incremental block
- Create two to eight blocks, and each block will require floorplanning
2
Synthesize
Synthesize each block.
- Write out individual netlist for each block
- For block netlists, disable I/O insertion
- Apply: do not insert clock buffer
3
Floorplan AREA Constraints
Use Floorplanner (or PACE) to layout area constraints for each of the preserved blocks.
- Use non-overlapping constraints
4
Implement
Implement the design until timing objectives are met.
- Use floorplan information in the UCF
- May require several implementation iterations with a high level
- Use incremental design techniques to maintain these good results
5
Incremental Synthesis
Make your changes incrementally; one module at a time.
- Resynthesize only the block that has changed
6
Incremental Implementation
Reimplement using guide files.
- As a guide, apply the previous map.ncd and par.ncd files
- All blocks without modification will maintain previous placement and routing
- Once all of the unchanged logic is placed and routed, the new logic will be placed and routed

Figure 3 – Incremental design techniques/design flow

your most pressing design needs. You will learn techniques that may allow you to use a slower speed grade device or fit your design into a smaller device – ultimately saving you money. As you master the tools and design methodologies taught in Designing for Performance, you will be able to create your design faster, with increased performance, shortening your development time and therefore costs.

Prerequisites for this course include Fundamentals of FPGA Design and basic VHDL or Verilog knowledge.

Designing with the Virtex-4 Family Course

At this point, you have taken Fundamentals of FPGA Design and Designing for Performance, but you want to design into the latest and greatest device, the Virtex-4 FPGA. The next course, Designing with the Virtex-4 Family, has an emphasis on teaching you to obtain the highest possible performance (up to 500 MHz internally) and reduce power consumption. This course covers a myriad of new and enhanced capabilities of the Virtex-4 FPGA fabric, with a major emphasis on lab exercises.

How do you utilize the new DSP48 resource for arithmetic operations? How do you utilize it for DSP applications? In lab exercises, you will use the DSP48 for DSP and arithmetic applications requiring 500 MHz operation. Did you know that you can also use this resource to implement logic resources such as a 6:1 multiplexer? We will give you the necessary information to take advantage of this dynamic, high-performance, and low-power resource.

How would you design a clocking scheme with ten global clocks and six regional clocks? In lab exercises, you will use the new Xesium clocking resources (see Figure 5) to design a complex clocking scheme in Virtex-4 devices. This includes utilizing the enhanced DCM, new phase-matched clock divider (PMCD), enhanced global clock buffers (BUFGCTRL), and new regional clocking resources (BUFIO and BUFR).

Have you heard about the block RAM performance enhancements and dedicated FIFO resources? You will create a 500 MHz block RAM core employing the new optional output register. You will also create a core utilizing the new dedicated FIFO16 resources.

Do you need to design a source-synchronous interface? The new Virtex-4 IOB tile includes ISERDES and OSERDES resources. You will examine the use of the new Xesium clocking resources and the ISERDES/OSERDES resources for creating your own source-synchronous interface. You will also learn about the available automated tools, ChipSync wizard, and memory interface generator (MIG) for creating source-synchronous interfaces.

You will learn to:

- Utilize the Xesium global (32) and regional (2 per region) clock networks
- Dynamically reconfigure the DCM's frequency synthesized output (CLKFX) and fine phase shift (DPS)
- Create phase-matched divided clocks using the new PMCD
- Create customized source-synchronous interface cores with the ChipSync wizard or memory interface generator (MIG)
- Increase the performance of your memory resources using the new Virtex-4 block RAM and FIFO16
- Increase performance and reduce power of your arithmetic and DSP circuits utilizing the DSP48 resource

We are positive that this course will provide the knowledge you need to take full advantage of the performance and power savings offered by the new Virtex-4 FPGA family. The prerequisites for this course include Fundamentals of FPGA Design and Designing for Performance, in addition to intermediate VHDL or Verilog knowledge.

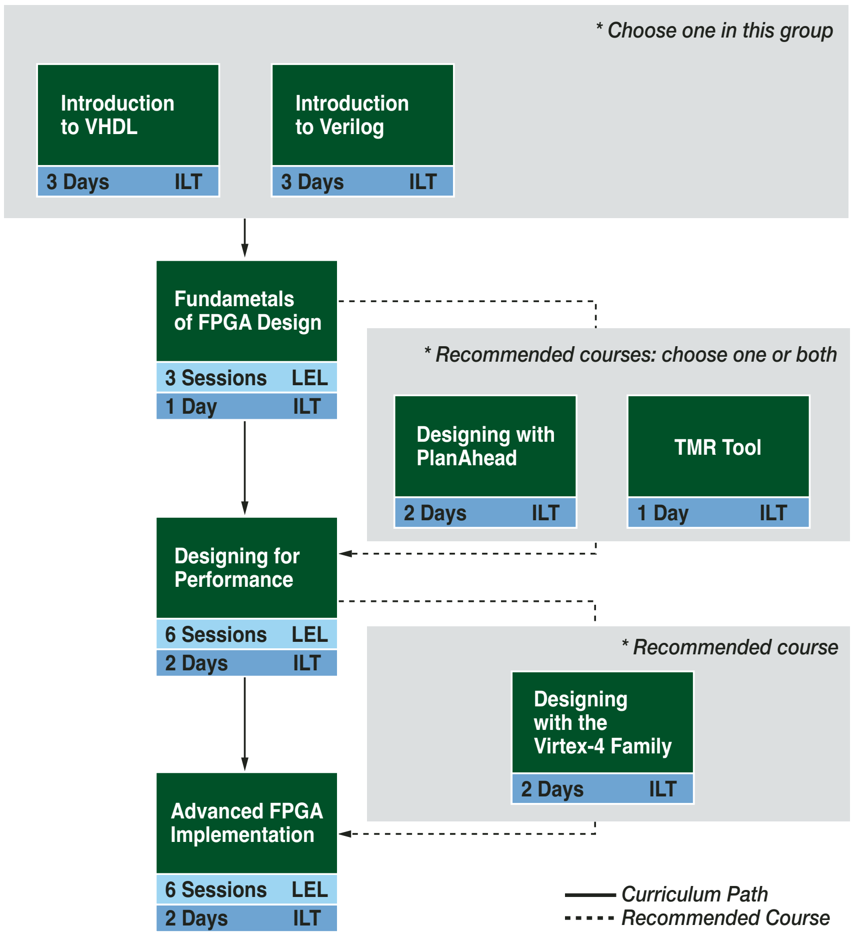


Figure 4 – Curriculum path for FPGA design courses

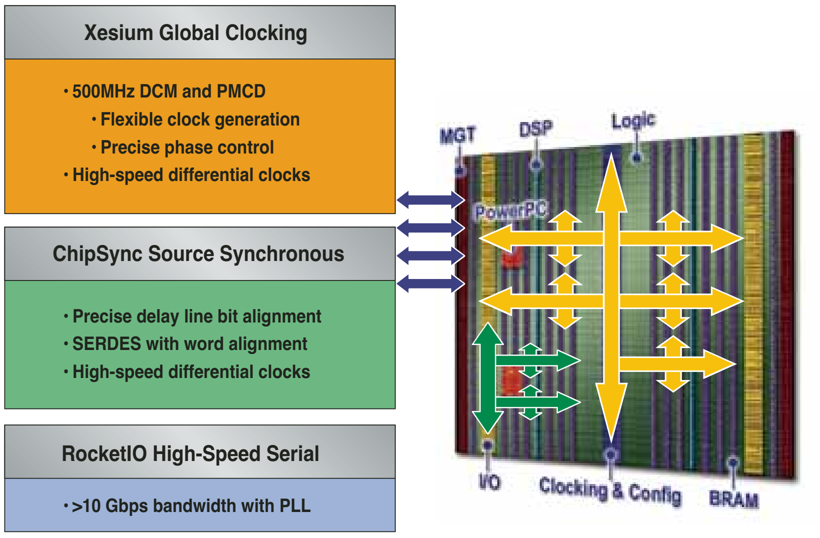

Xesium Global Clocking
• 500MHz DCM and PMCD
• Flexible clock generation
• Precise phase control
• High-speed differential clocks
ChipSync Source Synchronous
• Precise delay line bit alignment
• SERDES with word alignment
• High-speed differential clocks
RocketIO High-Speed Serial
• >10 Gbps bandwidth with PLL
MGT
DSP
Logic
PowerPC
I/O
Clocking & Config
BRAM

Figure 5 – Xesium clocking resources

Advanced FPGA Implementation Course

Building on the knowledge of the three previous courses, Advanced FPGA Implementation tackles the most sophisticated aspects of the ISE tool suite and Xilinx hardware. This course covers varying topics to help you achieve better results, make changes faster, and maintain your results. Lab exercises are 50% of class time, increasing your knowledge and skills through experimentation.

After each small design change, do you get frustrated when you have to completely re-implement a design? Do synthesis and implementation take too long? Do your timing results change? In lab exercises, you will use incremental design techniques (see Figure 3) for an incremental design change, preserving your previous timing results. This can reduce your iterative synthesis and implementation run time by as much as 50%.

Are you ready to create an implementation layout? You will use Floorplanner or PACE to create a design layout to increase the performance of your design. Additionally, these tools are used for designs employing incremental design techniques.

Do you have a design that uses more than eight clocks? In Virtex-II-based devices, many clocking features are available, but you must also consider their effects. In lab exercises, you will use a simple step-by-step strategy to design a complex multiple-clock clocking scheme in a Virtex-II device, taking advantage of all of the features available. The Virtex-4 clocking resources are also introduced.

Are you having trouble meeting timing on a particular path in your design? You will identify when and how to create a relationally placed macro (RPM) for problematic timing paths in your design. An RPM ensures predictable performance results for each included element. In the lab exercise, you will use an RPM to meet performance objectives for a critical path.

Do you prefer to use scripts rather than GUIs? With the techniques, strategies, and options covered, you will learn the most effective switches for improving performance. You will also use the settings and options in the scripting lab to greatly improve the performance of the design.

You will learn to:

- Create and edit timing and placement constraints to increase performance
- Build RPMs to improve performance on critical paths and achieve predictable timing results on complex functions
- Implement efficient clocking schemes for Virtex-II and Spartan™-3 FPGAs
- Use incremental design techniques to shorten the design cycle and maintain performance results
- Quickly modify implemented designs in FPGA Editor for more efficient in-circuit testing
- Use scripts and software options to increase performance and reduce area
- Use a systematic timing closure strategy to achieve optimum performance

With this course, our intent is to arm experienced designers with the necessary techniques, options, and strategies for obtaining breakthrough performance. Once you have achieved your performance objectives, we'll also teach you how to retain performance from one iteration to the next.

Your design goals equate to our teaching objectives. Only the most qualified and experienced instructors and design engineers qualify to teach this class. Instructors tailor the instruction and discussion within the framework of the class to help you achieve your personal learning objectives. This course requires Fundamentals of FPGA Design and Designing for Performance as prerequisites. An intermediate knowledge of Verilog or VHDL is strongly recommended, as is at least six months of design experience with Xilinx FPGAs.

Conclusion

Our goal is to help you achieve your design goals in the shortest possible time. To do that, we believe we have created the most comprehensive set of courses in the industry to accommodate your complex designs and goals. You can spend days and even weeks trying to accomplish your goals without really learning to quickly, correctly, and strategically use the software and hardware. Rather than waste your time and money, come and learn how the pros do it.

For a comprehensive list of training courses available worldwide or to take a skills assessment to find out what courses you need, go to *http://xilinx.com/education*, or click on the "Education" link from the support website (the curriculum path for FPGA Design is shown in Figure 4). We truly believe in what we do and what we can help you to do with Xilinx devices.

Earth Friendly

Xilinx delivers a broad range of Pb-Free and RoHS-compliant devices in the industry's most advanced, high-performance packaging. In fact, Xilinx has already shipped over 5.8 million earth-friendly devices–more than any other supplier.

Pb-Free
RoHS Compliant

With global programs for recycling, waste reduction, and energy management, we are proud to support our customers, and our environment. For more information visit *www.xilinx.com/pbfree.*

FORTUNE® 2005
100 BEST COMPANIES TO WORK FOR

XILINX®
The Programmable Logic Company℠

www.xilinx.com/pbfree

XILINX®
The Programmable Logic Company℠

Xilinx Virtex-4™ FPGAs

www.xilinx.com/devices/

Product Selection Matrix

		Virtex-4 LX (Logic)								Virtex-4 SX (Signal Processing)			Virtex-4 FX (Embedded Processing & Serial Connectivity)					
		XC4VLX15	XC4VLX25	XC4VLX40	XC4VLX60	XC4VLX80	XC4VLX100	XC4VLX160	XC4VLX200	XC4VSX25	XC4VSX35	XC4VSX55	XC4VFX12	XC4VFX20	XC4VFX40	XC4VFX60	XC4VFX100	XC4VFX140
CLB Resources	CLB Array (Row x Column)	64 x 24	96 x 28	128 x 36	128 x 52	160 x 56	192 x 64	192 x 88	192 x 116	64 x 40	96 x 40	128 x 48	64 x 24	64 x 36	96 x 52	128 x 52	160 x 68	192 x 84
	Slices	6,144	10,752	18,432	26,624	35,840	49,152	67,584	89,088	10,240	15,360	24,576	5,472	8,544	18,624	25,280	42,176	63,168
	Logic Cells	13,824	24,192	41,472	59,904	80,640	110,592	152,064	200,448	23,040	34,560	55,296	12,312	19,224	41,904	56,880	94,896	142,128
	CLB Flip Flops	12,288	21,504	36,864	53,248	71,680	98,304	135,168	178,176	20,480	30,720	49,152	10,944	17,088	37,248	50,560	84,352	126,336
Memory Resources	Max. Distributed RAM Bits	98,304	172,032	294,912	425,984	573,440	786,432	1,081,344	1,425,408	163,840	245,760	393,216	87,552	136,704	297,984	404,480	674,816	1,010,688
	Block RAM/FIFO w/ECC (18 kbits each)	48	72	96	160	200	240	288	336	128	192	320	36	68	144	232	376	552
	Total Block RAM (kbits)	864	1,296	1,728	2,880	3,600	4,320	5,184	6,048	2,304	3,456	5,760	648	1,224	2,592	4,176	6,768	9,936
Clock Resources	Digital Clock Managers (DCM)	4	8	8	8	12	12	12	12	4	8	8	4	4	8	12	12	20
	Phase-matched Clock Dividers (PMCD)	0	4	4	4	8	8	8	8	0	4	4	0	0	4	8	8	8
I/O Resources	Max Select I/O™	320	448	640	640	768	960	960	960	320	448	640	320	320	448	576	768	896
	Total I/O Banks	9	11	13	13	15	17	17	17	9	11	13	9	9	11	13	15	17
	Digitally Controlled Impedence	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
	Max Differential I/O Pairs	160	224	320	320	384	480	480	480	160	224	320	160	160	224	288	384	448
	I/O Standards	LDT-25, LVDS-25, LVDSEXT-25, BLVDS-25, ULVDS-25, LVPECL-25, LVCMOS25, LVCMOS18, LVCMOS15, PCI33, LVTTL, LVCMOS33, PCI-X, PCI66, GTL, GTL+, HSTL I (1.5V,1.8V), HSTL II (1.5V,1.8V), HSTL III (1.5V,1.8V), HSTL IV (1.5V,1.8V), SSTL2I, SSTL2II, SSTL18 I, SSTL18 II																
DSP Resources	XtremeDSP™ Slices	32	48	64	64	80	96	96	96	128	192	512	32	32	48	128	160	192
Embedded Hard IP Resources	PowerPC™ Processor Blocks	—	—	—	—	—	—	—	—	—	—	—	1	1	2	2	2	2
	10/100/1000 Ethernet MAC Blocks	—	—	—	—	—	—	—	—	—	—	—	2	2	4	4	4	4
	RocketIO™ Serial Transceivers	—	—	—	—	—	—	—	—	—	—	—	0	8	12	16	20	24
Speed Grades	Commercial (slowest to fastest)	-10, -11, -12	-10, -11, -12	-10, -11, -12	-10, -11, -12	-10, -11, -12	-10, -11, -12	-10, -11, -12	-10, -11	-10, -11, -12	-10, -11, -12	-10, -11, -12	-10, -11, -12	-10, -11, -12	-10, -11, -12	-10, -11, -12	-10, -11, -12	-10, -11
	Industrial (slowest to fastest)	-10, -11	-10, -11	-10, -11	-10, -11	-10, -11	-10, -11	-10, -11	-10	-10, -11	-10, -11	-10, -11	-10, -11	-10, -11	-10, -11	-10, -11	-10, -11	-10
	Configuration Memory Bits	4,765,568	7,819,904	12,259,712	17,717,632	23,291,008	30,711,680	40,347,008	51,367,808	9,147,648	13,700,288	22,745,216	4,765,568	7,242,624	13,550,720	21,002,880	33,065,408	47,856,896
	EasyPath™ Cost Reduction Solutions[4]	—	XCE4VLX25	XCE4VLX40	XCE4VLX60	XCE4VLX80	XCE4VLX100	XCE4VLX160	XCE4VLX200	XCE4VSX25	XCE4VSX35	XCE4VSX55	—	XCE4VFX20	XCE4VFX40	XCE4VFX60	XCE4VFX100	XCE4VFX140

Package[1]	Area	MGT[2]	Pins	4VLX15	4VLX25	4VLX40	4VLX60	4VLX80	4VLX100	4VLX160	4VLX200	4VSX25	4VSX35	4VSX55	4VFX12	4VFX20	4VFX40	4VFX60	4VFX100	4VFX140
SF363	17 x 17 mm	—	240	240	240										240					
FF668	27 x 27 mm	—	448	320	448	448	448					320	448		320					
FF1148	35 x 35 mm	—	768			640	640	768	768	768				640						
FF1513	40 x 40 mm	—	960						960	960	960									
FF672	27 x 27 mm	12	352													320 (8)[3]	352 (12)[3]	352 (12)[3]		
FF1152	35 x 35 mm	20	576														448 (12)[3]	576 (16)[3]	576 (20)[3]	
FF1517	40 x 40 mm	24	768																768 (20)[3]	768 (24)[3]
FF1760	42.5 x 42.5 mm	24	896																	896 (24)[3]

Notes: 1. SFA Packages (SF): flip-chip fine-pitch BGA (0.80 mm ball spacing).
FFA Packages (FF): flip-chip fine-pitch BGA (1.00 mm ball spacing).
All Virtex-4 LX and Virtex-4 SX devices available in the same package are footprint-compatible.
2. MGT: RocketIO Multi-Gigabit Transceivers.
3. Number of available RocketIO Muti-Gigabit Transceivers.
4. EasyPath solutions provide conversion-free path for volume production.

Pb-free solutions are available. For more information about Pb-free solutions, visit www.xilinx.com/pbfree.

Important: Verify all data in this document with the device data sheets found at http://www.xilinx.com/partinfo/databook.htm

Product Selection Matrix

		CLB Resources					Memory Resources			DSP	CLK Resources		I/O Features				Speed		PROM	
		System Gates (see note 1)	CLB Array (Row x Col)	Number of Slices	Equivalent Logic Cells	CLB Flip-Flops	Max. Distributed RAM Bits	# Block RAM	Block RAM (bits)	Dedicated Multipliers	DCM Frequency (min/max)	# DCMs	Digitally Controlled Impedance	Number of Differential I/O Pairs	Maximum I/O	I/O Standards	Commercial Speed Grades (slowest to fastest)	Industrial Speed Grades (slowest to fastest)	Configuration Memory (Bits)	EasyPath
Spartan-3E Family – 1.2 Volt																				
XC3S100E		100K	16 x 22	960	2,160	1920	15K	4	72K	4	5/326	2	NO	40	108	Single-ended LVTTL, LVCMOS3.3/2.5/1.8/ 1.5/1.2, PCI 3.3V – 32/64-bit 33/66MHz, PCI-X 100MHz, SSTL I 1.8/2.5, HSTL I 1.8, HSTL III 1.8 Differential LVDS2.5, Bus LVDS2.5, mini-LVDS, RSDS, LVPECL	-4 -5	-4	0.6M	
XC3S250E		250K	26 x 34	2448	5,508	4896	38K	12	216K	12	5/326	4	NO	68	172		-4 -5	-4	1.4M	
XC3S500E		500K	34 x 46	4656	10,476	9312	73K	20	360K	20	5/326	4	NO	92	232		-4 -5	-4	2.3M	
XC3S1200E		1200K	46 x 60	8672	19,512	17344	136K	28	504K	28	5/326	8	NO	124	304		-4 -5	-4	3.8M	
XC3S1600E		1600K	58 x 76	14752	33,192	29504	231K	36	648K	36	5/326	8	NO	156	376		-4 -5	-4	5.9M	
Spartan-3 and Spartan-3L Families – 1.2 Volt (see note 2)																				
XC3S50		50K	16 x 12	768	1,728	1,536	12K	4	72K	4	24/280	2	YES	56	124	Single-ended LVTTL, LVCMOS3.3/2.5/1.8/ 1.5/1.2, PCI 3.3V – 32/64-bit 33MHz, SSTL2 Class I & II, SSTL18 Class I, HSTL Class I, III, HSTL1.8 Class I, II & III, GTL, GTL+ Differential LVDS2.5, Bus LVDS2.5, Ultra LVDS2.5, LVDS_ext2.5, RSDS, LDT2.5, LVPECL	-4 -5	-4	.4M	
XC3S200		200K	24 x 20	1,920	4,320	3,840	30K	12	216K	12	24/280	4	YES	76	173		-4 -5	-4	1.0M	
XC3S400		400K	32 x 28	3,584	8,064	7,168	56K	16	288K	16	24/280	4	YES	116	264		-4 -5	-4	1.7M	
XC3S1000	XC3S1000L	1000K	48 x 40	7,680	17,280	15,360	120K	24	432K	24	24/280	4	YES	175	391		-4 -5	-4	3.2M	
XC3S1500	XC3S1500L	1500K	64 x 52	13,312	29,952	26,624	208K	32	576K	32	24/280	4	YES	221	487		-4 -5	-4	5.2M	✓
XC3S2000		2000K	80 x 64	20,480	46,080	40,960	320K	40	720K	40	24/280	4	YES	270	565		-4 -5	-4	7.7M	✓
XC3S4000	XC3S4000L	4000K	96 x 72	27,648	62,208	55,296	432K	96	1,728K	96	24/280	4	YES	312	712		-4 -5	-4	11.3M	✓
XC3S5000		5000K	104 x 80	33,280	74,880	66,560	520K	104	1,872K	104	24/280	4	YES	344	784		-4 -5	-4	13.3M	✓

Note: 1. System Gates include 20-30% of CLBs used as RAMs.
2. Spartan-3L devices offer reduced quiescent power consumption. Package offerings may vary slightly from those offered in the Spartan-3 family. See Package Selection Matrix for details.

Package Options and UserI/O[1]

		Spartan-3E (1.2V)						Spartan-3 (1.2V)								
			XC3S100E	XC3S250E	XC3S500E	XC3S1200E	XC3S1600E		XC3S50	XC3S200	XC3S400	XC3S1000	XC3S1500	XC3S2000	XC3S4000	XC3S5000
Pins	Area[2]	I/O's	108	172	232	304	376	I/O's	124	173	264	391	487	565	712	784
PQFP Packages (PQ) – wire-bond plastic QFP (0.5 mm lead spacing)																
208	30.6 x 30.6 mm			158	158				124	141	141					
VQFP Packages (VQ) – very thin QFP (0.5 mm lead spacing)																
100	16.0 x 16.0 mm		66	66					63	63						
TQFP Packages (TQ) – thin QFP (0.5 mm lead spacing)																
144	22.0 x 22.0 mm		108	108					97	97	97					
Chip Scale Packages (CP) – wire-bond chip-scale BGA (0.5 mm ball spacing)																
132	8 x 8 mm			92	92				89							
FGA Packages (FT) – wire-bond fine-pitch thin BGA (1.0 mm ball spacing)																
256	17 x 17 mm			172	190	190				173	173	173				
FGA Packages (FG) – wire-bond fine-pitch BGA (1.0 mm ball spacing)																
320	19 x 19 mm				232	250	250				221	221	221			
400	21 x 21 mm					304	304									
456	23 x 23 mm										264	333	333	333		
484	23 x 23 mm						376									
676	27 x 27 mm											391	487	489	489	
900	31 x 31 mm													565	633	633
1156	35 x 35 mm														712	784

Notes: 1. Numbers in table indicate maximum number of user I/Os.
2. Area dimensions for lead-frame products are inclusive of the leads.

Pb-free solutions are available. For more information about Pb-free solutions visit *www.xilinx.com/pbfree*.

Important: Verify all data in this document with the device data sheets found at http://www.xilinx.com/partinfo/databook.htm

For the latest information and product specifications on all Xilinx products, please visit the following links:

FPGA and CPLD Devices *www.xilinx.com/devices/*	Packaging *www.xilinx.com/packaging/*	Development Reference Boards *www.xilinx.com/board_search/*	Global Services *www.xilinx.com/support/gsd/*
Configuration and Storage Systems *www.xilinx.com/configsolns/*	Software *www.xilinx.com/ise/*	IP Reference *www.xilinx.com/ipcenter/*	

XILINX® The Programmable Logic Company℠

Xilinx Spartan™-3 FPGAs

www.xilinx.com/devices/

XILINX® The Programmable Logic Company℠

Xilinx CPLD

www.xilinx.com/devices/

Product Selection Matrix – CoolRunner™ Series

					I/O Features		Speed				Clocking	
	System Gates	Macrocells	Product Terms per Macrocell	Input Voltage Compatible	Output Voltage Compatible	Maximum I/O	I/O Banking	Min. Pin-to-pin Logic Delay (ns)	Commercial Speed Grades (fastest to slowest)	Industrial Speed Grades (fastest to slowest)	IQ Speed Grade	Global Clocks	Product Term Clocks per Function Block
CoolRunner-II Family – 1.8 Volt													
XC2C32A	750	32	40	1.5/1.8/2.5/3.3	1.5/1.8/2.5/3.3	33	2	3.8	-4 -6	-6	-6	3	17
XC2C64A	1,500	64	40	1.5/1.8/2.5/3.3	1.5/1.8/2.5/3.3	64	2	4.6	-5 -7	-7	-7	3	17
XC2C128	3,000	128	40	1.5/1.8/2.5/3.3	1.5/1.8/2.5/3.3	100	2	5.7	-6 -7	-7	-7	3	17
XC2C256	6,000	256	40	1.5/1.8/2.5/3.3	1.5/1.8/2.5/3.3	184	2	5.7	-6 -7	-7	-7	3	17
XC2C384	9,000	384	40	1.5/1.8/2.5/3.3	1.5/1.8/2.5/3.3	240	4	7.1	-7 -10	-10	-10	3	17
XC2C512	12,000	512	40	1.5/1.8/2.5/3.3	1.5/1.8/2.5/3.3	270	4	7.1	-7 -10	-10	-10	3	17
CoolRunner XPLA3 Family – 3.3 Volt													
XCR3032XL	750	32	48	3.3/5	3.3	36		5	-5 -7 -10	-7 -10	-10	4	16
XCR3064XL	1,500	64	48	3.3/5	3.3	68		6	-6 -7 -10	-7 -10	-10	4	16
XCR3128XL	3,000	128	48	3.3/5	3.3	108		6	-6 -7 -10	-7 -10	-10	4	16
XCR3256XL	6,000	256	48	3.3/5	3.3	164		7.5	-7 -10 -12	-10 -12	-12	4	16
XCR3384XL	9,000	384	48	3.3/5	3.3	220		7.5	-7 -10 -12	-10 -12	-12	4	16
XCR3512XL	12,000	512	48	3.3/5	3.3	260		7.5	-7 -10 -12	-10 -12	-12	4	16

Package Options and User I/O

		CoolRunner-II						CoolRunner XPLA3					
Pins	Area[1]	XC2C32A	XC2C64A	XC2C128	XC2C256	XC2C384	XC2C512	XCR3032XL	XCR3064XL	XCR3128XL	XCR3256XL	XCR3384XL	XCR3512XL
QFN Packages (QFG) – quad flat no-lead (0.5 mm lead spacing)													
32	5 x 5 mm	21											
48	7 x 7 mm		37										
PLCC Packages (PC) – wire-bond plastic chip carrier (1.27 mm lead spacing)													
44	17.5 x 17.5 mm	33	33					36	36				
PQFP Packages (PQ) – wire-bond plastic QFP (0.5 mm lead spacing)													
208	30.6 x 30.6 mm				173	173	173				164	172	180
VQFP Packages (VQ) – very thin QFP (0.5 mm lead spacing)													
44	12.0 x 12.0 mm	33	33					36	36				
100	16.0 x 16.0 mm		64	80	80				68	84			
TQFP Packages (TQ) – thin QFP (0.5 mm lead spacing)													
144	22.0 x 22.0 mm			100	118	118				108	120	118*	
Chip Scale Packages (CP) – wire-bond chip-scale BGA (0.5 mm ball spacing)													
56	6 x 6 mm	33	45						48				
132	8 x 8 mm			100	106								
Chip Scale Packages (CS) – wire-bond chip-scale BGA (0.8 mm ball spacing)													
48	7 x 7 mm							36	40				
144	12 x 12 mm									108			
280	16 x 16 mm										164		
FGA Packages (FT) – wire-bond fine-pitch thin BGA (1.0 mm ball spacing)													
256	17 x 17 mm				184	212	212				164	212	212
FBGA Packages (FG) – wire-bond fine-line BGA (1.0 mm ball spacing)													
324	23 x 23 mm					240	270					220	260

* JTAG pins and port enable are not pin compatible in this package for this member of the family.

Note 1: Area dimensions for lead-frame products are inclusive of the leads.

Product Selection Matrix – 9500 Series

					I/O Features		Speed				Clocking	
	System Gates	Macrocells	Product Terms per Macrocell	Input Voltage Compatible	Output Voltage Compatible	Maximum I/O	I/O Banking	Min. Pin-to-pin Logic Delay (ns)	Commercial Speed Grades (fastest to slowest)	Industrial Speed Grades (fastest to slowest)	IQ Speed Grade	Global Clocks	Product Term Clocks per Function Block
XC9500XV Family – 2.5 Volt													
XC9536XV	800	36	90	2.5/3.3	1.8/2.5/3.3	36	1	5	-5 -7	-7	NA	3	18
XC9572XV	1,600	72	90	2.5/3.3	1.8/2.5/3.3	72	1	5	-5 -7	-7	NA	3	18
XC95144XV	3,200	144	90	2.5/3.3	1.8/2.5/3.3	117	2	5	-5 -7	-7	NA	3	18
XC95288XV	6,400	288	90	2.5/3.3	1.8/2.5/3.3	192	4	6	-6 -7 -10	-7 -10	NA	3	18
XC9500XL Family – 3.3 Volt													
XC9536XL	800	36	90	2.5/3.3/5	2.5/3.3	36		5	-5 -7 -10	-7 -10	-10	3	18
XC9572XL	1,600	72	90	2.5/3.3/5	2.5/3.3	72		5	-5 -7 -10	-7 -10	-10	3	18
XC95144XL	3,200	144	90	2.5/3.3/5	2.5/3.3	117		5	-5 -7 -10	-7 -10	NA	3	18
XC95288XL	6,400	288	90	2.5/3.3/5	2.5/3.3	192		6	-6 -7 -10	-7 -10	NA	3	18
XC9500 Family – 5 Volt													
XC9536	800	36	90	5	5	36		10	-5 -6 -10 -15	-7 -10 -15	-15	3	18
XC9572	1,600	72	90	5	5	72		10	-7 -10 -15	-10 -15	-15	3	18
XC95108	2,400	108	90	5	5	108		10	-7 -10 -15 -20	-7 -10 -15 -20	NA	3	18
XC95144	3,200	144	90	5	5	133		10	-7 -10 -15	-10 -15	NA	3	18
XC95216	4,800	216	90	5	5	166		10	-10 -15 -20	-10 -15 -20	NA	3	18
XC95288	6,400	288	90	5	5	192		10	-10 -15 -20	-15 -20	NA	3	18

Pb-free solutions are available. For more information about Pb-free solutions visit *www.xilinx.com/pbfree*

Package Options and User I/O

		XC9500XV				XC9500XL				XC9500					
Pins	Area[1]	XC9536XV	XC9572XV	XC95144XV	XC95288XV	XC9536XL	XC9572XL	XC95144XL	XC95288XL	XC9536	XC9572	XC95108	XC95144	XC95216	XC95288
PLCC Packages (PC) – wire-bond plastic chip carrier (1.27 mm lead spacing)															
44	17.5 x 17.5 mm	34	34			34	34			34	34				
84	30.2 x 30.2 mm										69	69			
PQFP Packages (PQ) – wire-bond plastic QFP (0.5 mm lead spacing)															
100	23.3 x 17.2 mm										72	81	81		
160	31.2 x 31.2 mm											108	133	133	
208	30.6 x 30.6 mm				168				168					166	168
VQFP Packages (VQ) – very thin TQFP (0.5 mm lead spacing)															
44	12.0 x 12.0 mm	34	34			34	34			34					
64	12.0 x 12.0 mm					36	52								
TQFP Packages (TQ) – thin QFP (0.5 mm lead spacing)															
100	16.0 x 16.0 mm		72	81			72	81			72	81	81		
144	22.0 x 22.0 mm			117	117			117	117						
Chip Scale Packages (CS) – wire-bond chip-scale BGA (0.8 mm ball spacing)															
48	7 x 7 mm	36	38			36	38			34					
144	12 x 12 mm			117				117							
280	16 x 16 mm				192				192						
BGA Packages (BG) – wire-bond standard BGA (1.27 mm ball spacing)															
256	27 x 27 mm								192						
352	35.0 x 35.0 mm													166	192
FBGA Packages (FG) – wire-bond Fine-line BGA (1.0 mm ball spacing)															
256	17 x 17 mm				192				192						

Note 1: Area dimensions for lead-frame products are inclusive of the leads.

XILINX® The Programmable Logic Company℠

Xilinx CPLD

www.xilinx.com/devices/

IS YOUR CURRENT FPGA DESIGN SOLUTION HOLDING YOU BACK?

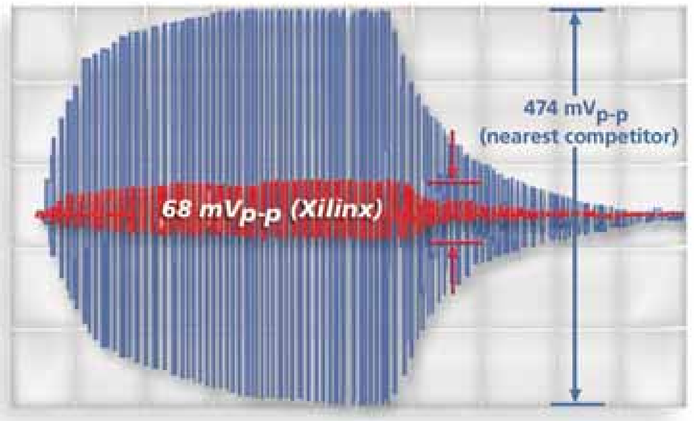
FPGA Design | Ever feel tied down because your tools didn't support the FPGAs you needed? Ever spend your weekend learning yet another design tool? Maybe it's time you switch to a truly vendor independent FPGA design flow. One that enables you to create the best designs in any FPGA. Mentor's full-featured solution combines design creation, verification, and synthesis into a vendor-neutral, front-to-back FPGA design environment. Only Mentor can offer a comprehensive flow that improves productivity, reduces cost and allows for complete flexibility, enabling you to always choose the right technology for your design. To learn more go to **mentor.com/techpapers** or call us at **800.547.3000.**

DESIGN FOR MANUFACTURING + INTEGRATED SYSTEM DESIGN
ELECTRONIC SYSTEM LEVEL DESIGN + FUNCTIONAL VERIFICATION

Mentor Graphics®

THE EDA TECHNOLOGY LEADER

"High Signal Integrity Demands A Low-Noise Chip".



Design Example: 1.5 volt LVCMOS 4mA, I/O, 100 aggressors shown.

Best Signal Integrity: 7x Less SSO Noise

Virtex-4 FPGAs deliver the industry's best signal integrity, allowing you to pre-empt board issues at the chip level, for high-speed designs such as memory interfaces. Featuring a unique SparseChevron™ pin out pattern, the Virtex-4 family provides the highest ratio of VCCO/GND pin pairs to user I/O pins available in any FPGA. By strategically positioning one hard power pin and one hard ground pin adjacent to every user I/O on the device, we've reduced signal path inductance and SSO noise to levels far below what you can attain with a virtual ground or soft ground architecture.

The Industry's Highest Signal Integrity, Proven By Industry Experts

Incorporating continuous power and ground planes, plus integrated bypass capacitors, we're eliminating power-supply noise at its source. In addition, we provide on-chip termination resistors to control signal ringing. The lab tests speak for themselves. As measured by signal integrity expert Dr. Howard Johnson, no competing FPGA comes close to achieving the low-noise benchmarks of Virtex-4 devices.

Visit ***www.xilinx.com/virtex4/sipi*** today, and choose the right high-performance FPGA before things get noisy.

Dr. Howard Johnson, author of *High-Speed Digital Design,* frequently conducts technical workshops for digital engineers at Oxford University and other sites worldwide. Visit www.sigcon.com to register.

XILINX®
The Programmable Logic Company℠

www.xilinx.com/virtex4/sipi

Breakthrough Performance At The Lowest Cost

PN 0010866