THE QUARTERLY JOURNAL FOR PROGRAMMABLE LOGIC USERS

Issue 32
Second Quarter
1999

Xcell

NEW TECHNOLOGY
The Design Software Revolution

PERSPECTIVE
HDL Verification

NEWS BRIEFS
Real 64/66 PCI Solution
Virtex 8051 Core

NEW PRODUCTS
XC9500XV CPLD Family

APPLICATION
Frequency Counter
Serial Communication Link

15th ANNIVERSARY 1984 1999

COVER STORY
Three articles discuss the history of Xilinx and the programmable logic industry.

XILINX®

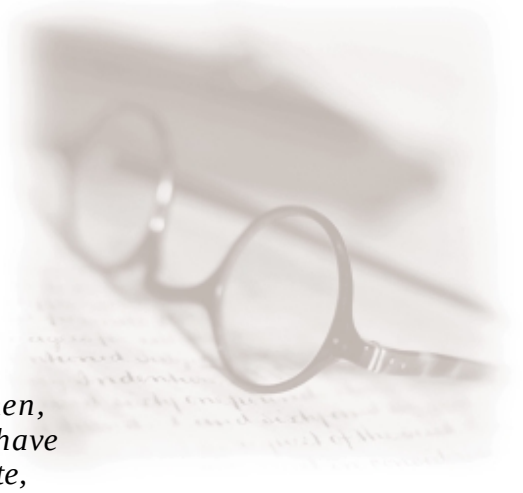
EDITOR
Carlis Collins
editor@xilinx.com
408-879-4519

SENIOR DESIGNER
Jack Farage

BOARD OF ADVISORS
Dave Stieg
Dave Galli
Mike Seither
Peter Alfke

PUBLICATION SERVICES
Ruddle Creative
111 N. Market St., Suite 715
San Jose, CA 95113
Tel: 408-297-3000 or
1-800-7RUDDLE
Web: www.ruddle.com

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124-3450
Tel: 408-559-7778
Fax: 408-879-4780



Putting Things in Perspective

Xilinx invented the FPGA just 15 years ago. Since then, programmable logic devices and development tools have changed dramatically. Today, using our million-gate, high-performance, system-level devices, you can create unique designs that were never possible before, get them to market sooner, and keep them in the market longer; your "window of innovation" is practically unlimited.

Before programmable logic, your window of innovation closed when your specification went to engineering because design changes were costly, often delaying product introduction. After FPGAs arrived, you could continue to add features right up to the time your design went to manufacturing, without causing delays or adding costs. Today, with our new Virtex™ FPGA family and our Internet Reconfigurable Logic™ capability, you can continue to innovate, adding new capabilities even after your designs are in the field (including features you haven't even thought of yet). A lot has changed in the last 15 years.

In addition to lower costs, unprecedented device density, and higher performance, there are a number of key factors that make today's programmable logic a key element in the life and profitability of new systems. For example, you can use a fast-growing assortment of intellectual property (cores) to quickly and inexpensively create the most complex and risky parts of your design. You can use a wide array of the most advanced time-saving development tools to enter, simulate, and debug your designs. Plus, along with Internet Reconfigurable Logic, there are a number of striking advances in the application of FPGAs that allow you to upgrade, test, and maintain your FPGA-based designs remotely, adding features and fixing bugs at your customers' locations, anywhere in the world, via the Internet—imagine the possibilities.

In addition to lower costs, unprecedented device density, and higher performance, there are a number of key factors that make today's programmable logic a key element in the life and profitability of new systems.

In the past, custom ASICs were used for high volume, cost sensitive designs. FPGAs were just too expensive and they did not have the raw performance or logic density to make them compelling in many applications. Plus, the FPGA development tools were often difficult to learn and they lacked the high-level features found in ASIC development systems. Now however, all that has changed.

FPGAs now compete very well, on price, performance, and ease of use against custom ASICs and even against many standard off-the-shelf devices. For example, using FPGAs, you can create DSP designs that far outperform any standard DSP device, or create fully-compliant 64-bit/66-MHz PCI designs that cost less and outperform any standard PCI device. Plus, FPGAs offer you the key advantage of profitability: you can get your product to market sooner and keep it in the market longer than with any other method—period.

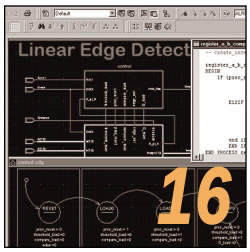
High performance, high density, cutting-edge innovation, ease-of-use, unique new applications, faster time-to-market, longer time-in-market, peace of mind–that's what you get from today's system-level FPGAs and development tools from Xilinx.

This issue of *Xcell* is intended to show you how far the programmable logic industry has progressed. I wonder what the next 15 years will bring… ✖

ARTICLES

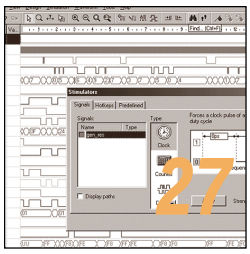
COVER STORY

A lot in the PLD industry has changed since Xilinx was founded in 1984. Three stories review the history of Xilinx and PLDs.



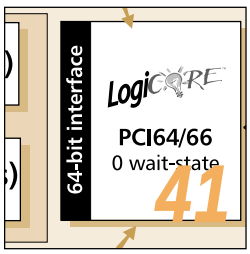
NEW TECHNOLOGY

New FPGA technologies have driven the demands on design software, yielding a new generation of synthesis, simulation, place and route and other tools for PLD designers.



PERSPECTIVE

The new generation of HDL verification tools handle larger design files, while requiring less time for the simulation process.



NEWS BRIEF

Our new PCI cores, called the Real 64/66 PCI solution, meet the demand for uncompromising compliance, flexibility and performance.

NEW PRODUCTS

The XC9500XV family of high-perforance 2.5V CPLDs provide power savings of up to 75% over today's 5V CPLDs, at a lower cost.

Inside This Issue:

15 Years of Innovation 4
The Evolution of PLDs 5
Times Have Changed 9
Silicon Xpresso: Internet Reconfigurable Logic 10
Internet Appliances 13
FPGA Synthesis 14
Design Software Revolution 16
Concept HDL 19
VeriBest's Vision 21
Prototyping with Certify 24
HDL Verification 27
Converting FPGAs to ASICs 30
Relative Location Constraints 31
Inferring Virtex Block RAM 34
Third-Party Virtex Cores 35
Success Story: SmartTouch 36
8051 Core for Virtex 37
32-Bit Ethernet Switch 38
SpartanXL and Video Apps 39
Success Story: Wildstar 40
Real 64/66 PCI 41
Success Story: 3D Graphics 43
Single DIME Module 44
Low-Cost PROM Programmer 45
XC9500XV CPLDs 46
400MHz Frequency Counter 47
Serial FPGA Com Link 50
Save Register Content During Power-Down 52
Determining Clock Skew 53
FPGA IP Center 55
Column: HDL Advisor 56
Verilog Training 58
Trade Show Programs 59
Device Selection 60
Software Selection 62

Xcell

15 Years of Innovation

by Wim Roelandts, CEO, Xilinx

Xilinx invented the FPGA just 15 years ago. Since that time, we have seen dramatic advances in device performance and density, while prices have steadily declined. These breakthroughs, combined with matching advances in our development tools, intellectual property, and support technologies, have created a revolution in logic design.

Our first FPGA, the XC2064, was shipped in 1985; it offered 800 gates, sold for $55, and was produced on a 2.0µ process. We have been shipping that device for 15 years and today it sells for just $5. Our latest generation of FPGAs, the Virtex family, is produced with a leading 0.22µ fabrication process, and offers advanced system-level features with densities over one million gates. Virtex FPGAs, along with our new Internet Reconfigurable Logic (IRL) technologies, are creating new design possibilities that no one could have dreamed of back in 1985.

Our use of the most advanced process technology is a key element of our aggressive development plan. For example, in 1997 our 0.35µ XC4000XL family went from concept to production in less that a year, the fastest product development effort in our history. With the XC4000XL family, Xilinx became the first company in the industry to ship a complete new line of ten 3.3V FPGAs. Our XC4085XL, the largest member of that family, provides up to 180,000 system gates.

By the end of 1997, we began shipping our XC4000XV family, the industry's first 0.25µ FPGAs, offering densities of up to 500,000 system gates. In 1998 we began shipping our Spartan family, the lowest cost FPGAs, ever. Today we are shipping million-gate, 0.22µ Virtex devices, and we have 0.18µ devices, operating at gigahertz speeds in our laboratories. This rapid pace of product development demonstrates our commitment to innovation, backed by the highest research and development budget of any company in our industry.

Xilinx is unquestionably the technology leader in producing the most innovative programmable logic devices. However, it takes much more than devices to capture the minds and imaginations of today's engineers. That's why we continue to push the boundaries of technology in the key areas that make programmable logic compelling and useful. We now have the most productive tools, with the shortest compile times in the industry, and we are continuously expanding our list of intellectual property (cores), so you can go from concept to full production faster and with less effort than ever before. In addition, we are creating new enabling technologies such as Internet Reconfigurable Logic that allow you to develop and test your designs remotely; you can now add features or fix bugs at your customers' locations, anywhere in the world, over the Internet. This not only extends the life of your product, it also creates exciting new possibilities for products such as Internet-based "appliances."

Our goal is to bridge the gap between your imagination and the physical world. We do this by making it as easy as possible for you to create the most advanced systems, get them to market sooner, and keep them in the market longer. Soon we will offer two-million gate devices, team-based development tools, and many more ways to use programmable logic in new and unique applications.

The future is close and arrives very quickly these days. Σ

Major Xilinx Milestones

1984 Xilinx is founded by Bernie Vonderschmitt, Jim Barnett, and Ross Freeman.

1985 Introduced the XC2000 series, our first family of field programmable gate arrays (FPGA).

1987 Introduced XC3000 series, our second family of FPGAs.

1988 Established a subsidiary in Japan.

1989 More than one million FPGA devices sold.

1990 Initial public offering.

1991 Introduced XC4000 series, third family of FPGAs.

1992 Expanded into market for complex programmable logic devices (CPLDs).

1993 Established a subsidiary in Hong Kong.

1995 Ranked world's 10th largest ASIC supplier; Xilinx Ireland facility opens.

1996 Ranked world's 8th largest ASIC supplier; Wim Roelandts joins as CEO.

1997 World's 1st 0.35 and 0.25µ FPGAs.

1998 Introduced Spartan and Virtex FPGAs, unveiled Internet Reconfigurable Logic.

1999 Xilinx celebrates 15th anniversary; introduced XC9500XV, 1st 2.5V CPLD family.

THE EVOLUTION of Programmable Logic Design Technology

A historical perspective on the evolution of Xilinx development systems and design methods.

by Craig Willert, High Volume Solutions, Software Market Manager, Xilinx, cnw@xilinx.com

With the introduction of the first programmable LCA (Logic Cell Array) in 1985, Xilinx changed the course of electronic design. This innovative new technology allowed engineers to design a single chip that performed the equivalent function of a typical printed circuit board. While this creative use of silicon has been heralded as a breakthrough technology that created a new industry, many insiders will tell you that it was the methodical delivery of innovations in the development systems tools that has enabled so many engineers to take advantage of programmable logic.

Development System Timeline:

Year	Key Features	Associated Design Flow
1984	Xilinx Design Editor	Physical Design Editor/ PIP Poking
1986	XNF2LCA1.0, APR1.0, XACTOR 2	Schematic based design, automatic logical to physical design translation, automatic place and route, in-circuit debugging
1988	XNFBA	
1990	XMAKE, Guided Design,	Automatic Design Implementation, Map-then-Merge, XDM
1992	Unified Libraries, XBLOX, Hard Macros	XACT 5 – Batch XACT 6 – Windows (DM/FE)
1994	XACT Performance, RPMs, DM/FE, Floorplanner	
1995	Foundation Series introduction, Software Localization (Japanese)	
1996	MAP, PAR, MPPR, Implementation Engine, EDIF, VHDL, and Verilog standards, LogiBlox, PCI LogiCore, ITA, JTAG, FPGA Express, Core Generator, Constraints Editor, Floorplanner (reintroduced), Foundation UPM	

—1985—
Introduction of the XC2000 Series

When Xilinx introduced its first Logic Cell Array in 1985, the Electronic Design Automation (EDA) industry was focused on delivering schematic capture and gate-level simulation tools to simplify the process of logic design. Although it was considered important to provide an easy to use design solution, the first priority of Xilinx was to enable its customers to gain access to the full complexity of its programmable logic devices. Xilinx first developed a physical design tool – XACT 1.0. This tool, which later became known as XDE (Xilinx Device Editor), gave designers the ability to view, edit, and highlight all of the logic and routing resources within a device. Furthermore, XDE featured advanced "high-level" design capabilities such as Boolean expression, Karnaugh Map, and truth table design (graphically cross-coupled, no less).

...the first priority of Xilinx was to enable its customers to gain access to the full complexity of its programmable logic devices.

The use of XDE ushered in the era of a design methodology that may be characterized as "PIP-poking" (Physical Interconnect Point). XDE's introduction was also responsible for the development of a new breed of designers – "Xilinx experts," who not only were unique in their understanding of the Xilinx design methodology, but also in their detailed understanding of Xilinx programmable logic devices.

Continued on the following page

—1986—

From PIPs to XNF

As the use of programmable logic devices became popular, and it became clear that the Programmable Logic business was viable, Xilinx turned its attention towards improving the LCA design methodology so that designers could take better advantage of the time-to-market benefits that programmable logic offers. To tackle this problem, Xilinx developed a set of design tools that would handle the translation of a design from its logical form (represented as a schematic) to the physical structures that exist within the Logic Cell Array. This spurred the creation of the "PIN2LCA, XNF2LCA, and APR programs.

PIN2LCA

The first qualified "front-to-back" design environment for Xilinx logic devices was created to work with Futurenet, a leading schematic capture tool at the time. Through the first "EDA Alliance," Xilinx and Futurenet were able to create an automatic logical netlist (logic gates and registers) to physical netlist (configured CLBs and IOBs) translation tool (PIN2LCA). By allowing an engineer to focus on designing their required circuitry using schematic capture rather than the Xilinx Design Editor, the use of programmable logic became a viable design methodology for many more designers.

Xilinx Netlist Format and XNF2LCA

In addition to working with Futurenet to develop a proprietary design flow, Xilinx recognized the need to work with other EDA companies. The Xilinx Netlist Format (XNF) was introduced in order to provide a standard for transferring a design database from any schematic package to the Xilinx design environment. Of course, to leverage this netlist standard, Xilinx had to develop the netlist translation program XNF2LCA, completing the transformation of a design from the logical database to the physical (LCA) database.

Through the publishing of this standard, and the creation of the XNF2LCA program, Xilinx ensured that any EDA company that could create XNF files would be in a position to offer advanced design methods for Xilinx devices.

—1987—

Introduction of the XC3000 Series

With the introduction of the XC3000 family, the density and complexity of the Xilinx programmable logic devices increased dramatically. Recognizing the associated growth in complexity of the programmable logic design methodology, Xilinx introduced a number of innovations to help ease the computing burden. The innovations included Automatic Place and Route, Map-then-Merge, XBLOX, Guided Design methodologies, and the XMAKE program.

Automatic Place and Route

To complete the automation of the layout of Xilinx logic devices, Xilinx developed the industry's first Automatic Place and Route (APR) program. APR 1.0 simplified the programmable logic design process by relieving the engineer of the time consuming tasks of identifying which CLBs within a LCA would contain the design's logic, and interconnecting them to other CLBs and IOBs. With the delivery of this program, the Xilinx design process went from being manual and interactive to being automated and batch oriented.

Map-then-Merge

With the introduction of the XC3000 family, the density and complexity of the Xilinx programmable logic devices increased dramatically.

Acknowledging the limitations of the performance of the typical engineer's computing platform, Xilinx patented a modular design method which is known as the Map-then-Merge approach. Taking clues from the source design's hierarchy, the Map-then-Merge approach divides the technology mapping task into smaller units, thus increasing the algorithmic throughput, and improving the processing time for designing complex programmable logic circuits. Furthermore, by separating the XNFMAP program from the XNFMERGE program, designers could discover the cause of any design errors more easily.

XBLOX

In an era where design reuse, and the inclusion of 3rd party Intellectual Property (IP or Cores) is at an all time high, it is interesting to review the XBLOX design tool that Xilinx offered its customers so long ago. In what was one of the industry's first high-level design methodologies, Xilinx patented an innovative mechanism for performing parameterized design using schematic capture. XBLOX allowed engineers to create a design with "n-width" buses, which could be specified in a single location within the schematic. The bus width parameter would then be propagated through the design, based on the designer's specifications. The propagation (or forward annoation) of the bus width parameter simplified the process of modifying a design's functionality by localizing this type of change to one design element.

XBLOX also included the ability to specify area or speed optimization of its functions on a "core-by-core" basis. The

In the era of schematic design and batch mode processing, Xilinx offered both interactive-based (GUI) processing and command line access to the algorithms that were required to layout Xilinx FPGAs.

XBLOX compiler was rerun at the time of layout, whenever design iteration called for a modification in the XBLOX specification or target device.

Guided Design

Attempting to accelerate the process of iterating on a design, Xilinx accelerated the place and route process by delivering the Guided Design capability. Guided Design allows an engineer to leverage the results from an earlier place and route run in the layout of his next version of an FPGA. While not effective in all design flows, this initial guided design technology proved to be very effective in accelerating the placement and routing of most FPGA design's when only small (iterative in nature) changes were required to a schematic.

XMAKE

In the era of schematic design and batch mode processing, Xilinx offered both interactive-based (GUI) processing and command line access to the algorithms that were required to layout Xilinx FPGAs. However, unless a designer was sitting in front of a design flow chart, the sequence of commands was not always easy to remember. To further simplify the programmable logic design process, Xilinx introduced the XMAKE program. XMAKE was designed to perform date stamp analysis, and to automatically run any out of date step in the design flow.

—1992—

Advanced Design Specifications

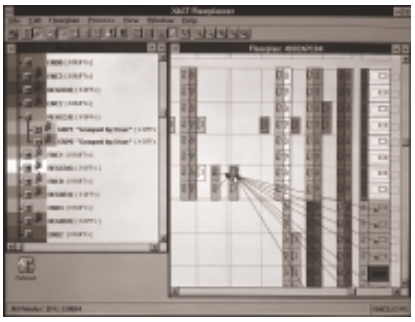
With the mainstream use of Programmable Logic Devices came a growing hunger for performance and efficiency in the use of silicon. To help its customers take advantage of the density and performance of Xilinx Logic devices, Xilinx introduced a set of tools that allowed an engineer to provide more information about a design to the layout tools. Armed with the new information provided by TimeSpecs, Relationally Placed Macros (RPM), and the Floorplanner, the layout tools were more prepared to focus on the true design requirements.

XACT Performance

Prior to the introduction of XACT Performance and TimeSpecs, skilled programmable logic designers manually placed and routed a design, or played tricks with a signal's "net criticality" in order to improve a circuit's performance. XACT Performance changed all of this by allowing design engineers to specify the circuit's performance criteria using Timespecs. Given this new information, the XACT Performance place and route tools were able to algorithmically evaluate the suitability of various layouts. While this processing required more CPU time, batch mode processing freed the designer to attend to the other requirements of his job, thus improving design time and time to market.

Floorplanner

Xilinx introduced another revolutionary capability to the programmable logic design market with the release of the XACT Floorplanner. The Floorplanner enabled a designer to impart his expert knowledge of the design as guidance to the



layout process, thus simplifying the increasingly complex problem of logic placement. Advanced programmable logic designers found that by "seeding" the placement of a small portion of a design, they could improve circuit performance by up to 25%, and cut place and route run times in half. Furthermore, the Floorplanner proved to be an excellent tool for analyzing a design's logic and routing density, providing insight into the suitability of a design for future enhancements.

—1994—

Graphical User Interfaces

With the introduction of powerful Pentium processors, and the Windows operating systems, the use of personal computers as engineering workstations became common. To foster the adoption of its operating system, Microsoft established standards for GUIs that would enable first time users of an application to feel comfortable in learning to use any new product which was designed using these standards.

Continued on the following page

With ambitions of expanding the market for programmable logic, Xilinx developed a GUI for the XACT implementation tools that adhered to MFC standards. In addition to creating a GUI which simplified the means by which the tools were run (Xilinx Flow Engine), Xilinx also added a variety of design management capabilities to this tool set, simplifying the process of managing a design's source and object files (Xilinx Design Manager).

—1995—

Xilinx Acquires NeoCAD

Xilinx acquired the advanced FPGA implementation technology and resources of NeoCAD to more rapidly deliver state of the art support for a growing number of devices. The result of merging the technology and resources of these two companies has been the creation of a powerful, fast, timing-driven set of implementation tools for both CPLDs and FPGAs.

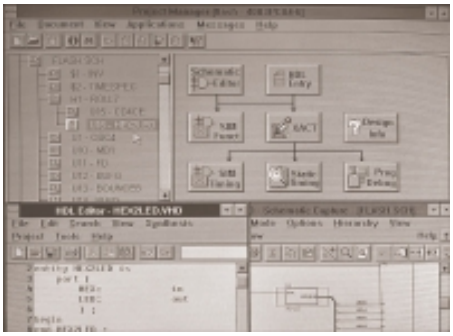
Some key software milestones that have occurred directly as a result of the acquisition:

- The first release of software support for new Xilinx architectures and devices in advance of sample silicon.
- The development of the first FPGA that has been optimized for fast and efficient layout, by customers using push-button design flows.
- The creation of the industry's best standards-based interface for working with 3rd party EDA providers, including EDIF, VHDL, Vital, Verilog, and SDF.
- The development of advanced performance driven algorithms that now run ten times faster than previous releases of XACT and NeoCAD Foundry implementation tools.
- The first localization of programmable logic tools for the Japanese market.

—1996—

Ready-to-Use PLD Design Tools

Shortly after Xilinx and NeoCAD joined forces, Xilinx introduced its new suite of tools for the design of programmable logic devices — the Foundation Series™. The Foundation Series software included all of the basic necessities required of



an easy-to-use design package, plus some capabilities that only the most advanced EDA companies were offering. The product was so well received that within the first two years, Xilinx had shipped over 10,000 seats.

To improve the Foundation Series quality of results, Xilinx signed an OEM agreement with Synopsys, Inc. for the inclusion of its FPGA Express™. FPGA Express strengthened the Foundation Series by enabling engineers to design with VHDL, Verilog, or a mixture of the two languages — a trend that has become more popular as device densities have skyrocketed, causing more people to turn to design reuse methodologies.

—1997—

The Xilinx CORE Generator™

As a result of the close working relationships with its foundries and the inherent advantages of using programmable logic as a process leader, Xilinx devices leapt to the forefront of process technology. The immediate benefit of this process leadership role to Xilinx and its customers was rapid advancements in device densities and performance. These benefits positioned Xilinx as a viable alternative for the heart of many state of the art electronic systems. As such, developers of emerging standards began turning to Xilinx as a potential provider of qualified solutions. This in turn spurred the creation of the Xilinx LogiCore and Alliance Core programs. Rather than merely being a conduit for marketing qualified VHDL and Verilog designs, Xilinx core programs focussed on developing a flexible, parameterized core delivery capability, where hard, soft, and firm cores are all made available to customers through a tightly integrated design interface – the Xilinx CORE Generator.

Conclusion

Xilinx has made key advancements in both device and software technologies, and our programmable logic solutions will continue to lead the industry for years to come.

HOW Times HAVE Changed

Reminiscing about the "good old days."

by Paul Gigliotti, Xilinx Applications Engineer, Xilinx, giglio@xilinx.com

In 1986, not long after slide rules went out of style, one of my first assignments as an engineer was to design an ASIC for an industrial controller. The design was entered into schematics using an 8086-based PC. Simulation was accomplished using a hardware accelerator that could run just over 100 test vectors a second. We only had one hardware accelerator, with two ASIC designs going on at the time. Because my project had lower priority, I had to work nights to guarantee access to the accelerator. Otherwise, I had to use the PC-based simulator, capable of about two to three vectors per second. If I had not worked nights, I might still be simulating that design.

My first ASIC design used approximately 2,000 gates, ran at 10 MHz, and had 20 TTL level I/Os. NRE for the chip was around $15,000 and the chip price was about $15.00. The design took six to eight hours to place and route on a PDP-11 mainframe computer. At that time FPGAs had been available for about a year and using two XC2018s would have offered a $130 solution to the problem. However, FPGAs were too new, and too expensive to be a viable contender for this design. Besides, who was this company Xilinx, and would they be around in 10 years, at the end of this product's life cycle?

About five years later, I designed a video acceleration card and used a Xilinx XC3090 to perform a pixel mixing algorithm. By this time FPGAs were considered a proven commodity, and Xilinx was a "household word" in the engineering community. The design used approximately 7000 gates, ran at 25 MHz, and had 95 TTL level I/Os. The chip cost was about $150. At that time, the Xilinx XACT software, running on a 33 megahertz 80386, could place the design in four to six hours. My compile times had dropped significantly, and I could do it on my desktop, instead of on a mainframe, and I was able to work daylight hours. This was real progress.

Now it's 1999 and I've been a Xilinx employee for three years. On a recent customer visit, an engineer gave me a design implemented in VHDL. As we talked, I targeted the design at the Virtex family, using the Foundation Express software. The design had nearly 150 I/Os, with 96 of them being GTL-compatible. I was able to fit the design into a 50,000 gate Virtex device, as we talked, and the design ran at over 100 megahertz. Overall, I placed and routed the design three times on my notebook computer, during our 2 hour meeting.

It's amazing how the world has changed for Xilinx over the past 15 years. The ASIC vendor that I had used in my first design has disappeared, taking the ASIC with it.

It's amazing how the world has changed for Xilinx over the past 15 years. The ASIC vendor that I had used in my first design has disappeared, taking the ASIC with it. On the other hand, the Xilinx XC2018 was in full production until about two months ago. I've gone from using a room full of computers to place and route a 2000 gate ASIC to using a 10 pound notebook computer to place and route a 50,000 gate Xilinx FPGA. Design speeds have jumped from a few megahertz to over 100 MHz. Compile times have been drastically reduced from a work day to a coffee break. In 1985, a 64-bit shift register would consume an entire XC2064, today it would fit in a single Virtex CLB.

Of course, technology keeps marching on. We already have two-million gate devices on the drawing board, and in our laboratories, we have devices running at gigahertz speeds. Our development software is continually being improved, both with our in-house development and with tools and technologies from our EDA partners. And, we continue to create new enabling technologies such as our Silicon Xpresso software that allows you to modify and test your FPGA designs, at your customers' locations, anywhere in the world, over the Internet.

There's no end in sight to the possibilities offered by programmable logic. ✖

Silicon Xpresso *and* Internet Reconfigurable Logic

Here's how Xilinx is using the Web to provide 21st century tools and applications.

by Wallace Westfeldt, Product Manager for Internet Reconfigurable Logic, Xilinx, wallace@xilinx.com

Last September, we announced our Silicon Xpresso initiative to create interactive Web-based tools and services that leverage the power of the Web for speedier design development. Then, in October, we announced our award winning Internet Reconfigurable Logic (IRL) applications, which enable you to upgrade digital products over any network. This article provides a summary and status of both Silicon Xpresso and IRL.

Silicon Xpresso

Silicon Xpresso is a suite of Internet-enabled tools that leverage the Internet and the Web to make you more productive. These tools include WebFitter, all the Internet-enabled software releases since 1.5i (which connect to support.xilinx.com), and Internet Team Design.

WebFITTER

WebFITTER is a free, Web-based CPLD fitter that runs on Xilinx in-house secure servers. It includes all of the features contained in the latest Xilinx development software, currently version 1.5i of our Alliance Series® and Foundation Series® tools. By simply submitting your CPLD designs via the Internet, Xilinx can quickly and efficiently "fit" your design into any XC9000 series CPLD device.

WebFITTER is fast, it's easy to use, it requires almost no learning curve, and it is always up to date with our latest enhancements. Therefore, it allows you to evaluate your CPLD designs in Xilinx XC9000 series devices with very little effort. You can easily make informed decisions and choose the best CPLD for your particular needs. With WebFITTER you can focus your creativity where it produces the best results — on your front-end design — leaving the back-end fitting and software management to us. It costs you nothing to use WebFITTER, the reports, or the device programming files that we create for you.

support.xilinx.com

Starting with release 1.5i in October 1998, Xilinx software releases are "Internet-enabled." Simply by clicking on the help menu of the Alliance Series Design Manager or the Foundation Series Project Manager you have immediate access to support.xilinx.com, our designer-centric website.

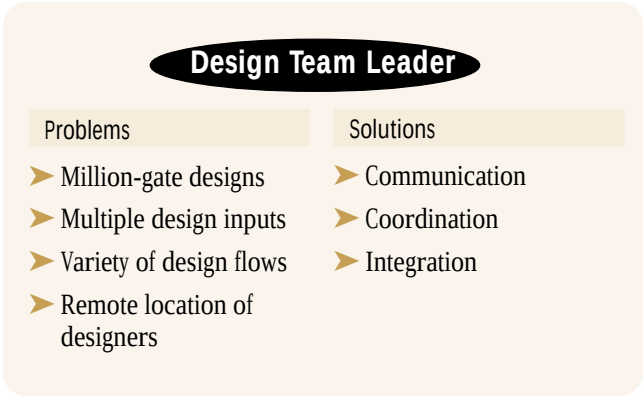
On support.xilinx.com you can quickly find the answers you need from our Answers™ database containing over 3000 records. You can easily construct online searches of this database or search all available application notes, expert journals, data sheets, on-line manuals, and articles from our Xcell Journal. To help you quickly finish your design, support.xilinx.com also provides:

- **Troubleshooting Tools** – powerful search engines updated daily.
- **Software Updates** – the latest files and service packs, for your convenience.
- **Searchable Library** – volumes of reference materials harnessed through advanced search engines.
- **Education Services** – just-in-time training, when you need it.
- **On-line documentation** – software documentation for your convenience.
- **A "what's new" section** – so that you can make sure you have the latest information.

Be sure to visit support.xilinx.com frequently and keep your design on track.

The solution to these challenges lies in efficient and asynchronous communication, an infrastructure for coordinating multiple inputs, and the ability to integrate a variety of design files.

21st Century Design Challenges

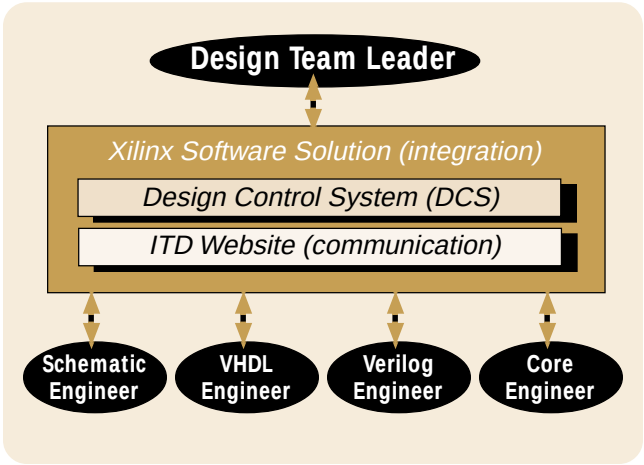


Internet Team Design (ITD)

Coming this summer, with the 2.1i release of our Alliance Series and Foundation Series software, is a development option called Internet Team Design (ITD). ITD is designed to help today's hardware design teams deal with tomorrow's application challenges that include very large and complex designs, where each team member contributes to different aspects of the design and may frequently work from different flows, from a remote location. The solution to these challenges lies in efficient and asynchronous communication, an infrastructure for coordinating multiple inputs, and the ability to integrate a variety of design files.

The Internet, and its cousin a secure *Intra*net, combined with the Xilinx ITD development option, allows your team to work closely together regardless of location or design flow. Combining Java-based technology with powerful Xilinx implementation tools, ITD forms an HTML-based communications layer that allows you to communicate asynchronously and remotely with your project leader. Then, your inputs and project reports are maintained by an automated Design Control System which allows individual team members to check-in their interactive designs simply by using their Web browser. And then finally, these designs are integrated into the application using the Xilinx Software Solution.

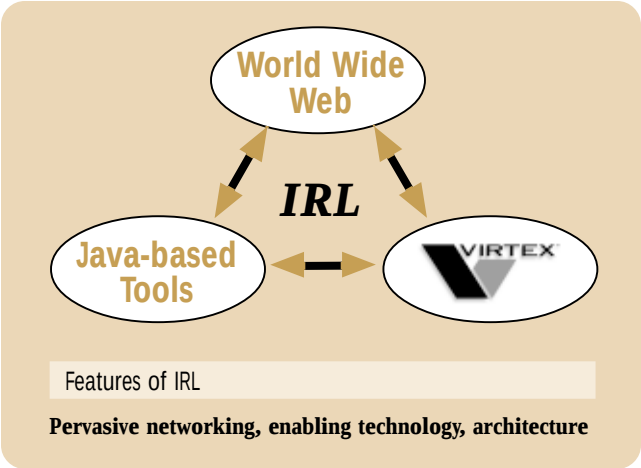
21st Century Design Solution



Internet Reconfigurable Logic

Imagine having the ability to remotely upgrade, test, and repair your logic designs anywhere on earth, or above the earth (for a satellite application perhaps). This is not tomorrow's dream; this is today's reality. To facilitate and expand this reality, Xilinx has created a systems approach called Internet Reconfigurable Logic (IRL). IRL is the combination of pervasive networking, our enabling software technology, and our advanced programmable logic architecture to provide programmable logic systems that can be easily modified, upgraded, and tested after your systems have been deployed.

Internet Reconfigurable Logic



IRL systems bring six key added values to a programmable logic application:

1. **Shorter Time to Market** – by allowing earlier releases of your systems, that can be upgraded later.
2. **Extended Time for Revenue** – by allowing modification of your system, after the release.
3. **Lower costs –** by not requiring the retooling or redeployment of devices, particularly in remote or widespread locations.
4. **Lower costs –** by eliminating the most expensive stage of manufacturing-in-system test.
5. **Expanded application opportunities** – by leveraging personalized network applications.
6. **Increased revenue** – by allowing you to charge for incremental system upgrades. The Internet usage business model is similar to the model used for cellular phones and cable TV.

Continued on the following page

Spectrum of Reconfiguration

Reconfiguration Requirements

When creating an IRL system, it's important to analyze your reconfiguration requirements. It may be that you only need to do in-system test before going online. Or, you may need to do incremental or periodic upgrades of the device either for maintenance or for selling added value. You may also want to do network reconfiguration because the system has remote or multiple deployment sites, as used in cellular phone base stations or orbiting satellites. The foremost consideration in all these examples is the frequency of reconfiguration.

The spectrum of reconfiguration spans a wide range of frequency and applications:

- **Field reconfiguration** is generally not time dependent, but it has the most pervasive application. It can be used for updates, bug fixes, and adding new functionality to your digital hardware.
- **Periodic reconfiguration** can be used for those applications where there is a regular change of supporting data, such as environmental monitoring systems, GPS, and so on.
- **Frequent reconfiguration** can be used to accelerate data processing for applications such as image processing.
- **Runtime reconfiguration** can be used where your application relies on changing variables such as detecting network protocols in a mobile application or interrupting operation with new functionality because a security or safety sensor has been triggered.

In many of these examples you need to reconfigure part of the FPGA while the rest of your logic remains the same. This is called "partial reconfigurability" which is provided by the Virtex architecture.

For field upgrades and even partial reconfigurations it is completely feasible to use standard development flows. On our IRL website (http://www.xilinx.com/products/software/sx/sxpresso.html) you can access application notes that describe several methods for field upgrades and partial reconfiguration.

For IRL systems that require either multiple device updates or rapid reconfiguration you might consider using portable Java-based tools such as the "Java API for Boundary Scan" or JBits. Check our IRL website for updates on these technologies.

In addition to the frequency of reconfiguration there are a number of other equally important items to consider such as:

- **System Level Planning** – Your system must be designed to allow in-system programming of the reconfigurable logic devices. The JTAG programming interface (available on all of the latest Xilinx FPGAs and CPLDs) is a good universal choice (see Java API for Boundary Scan), especially if you have multiple reconfigurable logic devices.

  You also need to have some way of getting the new bits into your system. The new configuration could be provided manually and loaded through some external interface on the system. If the system is on a network or has other communication capability the new configuration could be loaded remotely.

- **Reconfigurable Logic Planning** – With planning, a circuit design implemented in reconfigurable logic can be changed and still fit in the same part, operate at the same speed, and require the same I/O. However, it does require planning at design time. There are some obvious general guidelines, such as reserving gate and flip-flop capacity for future expansion, and having some timing margin. Understanding and anticipating the kinds of changes that may be required in the future is key.

Conclusion

Xilinx is dedicated to creating 21st Century design tools for 21st Century applications. We are the innovation leader, providing enormous opportunities for you, and helping you provide insanely great applications. *The revolution is brewing, have a cup on us.* Σ

Contributors: Alan Frost, Steve Guccione, Neil Jacobson, Steve Lass, Delon Levi, Scott Lewis, Frank Toth, Alica Tripp

Internet Appliances

The Possibilities are Limitless

With the advent of JTAG test circuitry, In-System Programming, and Internet Reconfigurable Logic, we are witnessing the dawn of the Internet Appliance era.

by Jesse Jenkins, CPLD Applications Manager, Xilinx, jesse@xilinx.com

As Xilinx celebrates its 15th anniversary, many new applications for programmable logic are emerging while traditional ways of designing and developing systems are quickly fading away. Many new systems are being designed to allow continuous hardware upgrade capability, over the Internet, giving them unique new capabilities and a longer life span. It's easy to envision "universal" systems that can easily change functions to become anything you can imagine.

Examples of Internet Appliances

An Internet Appliance is any hardware device that relies upon either an Internet connection or upon access to Internet information files to achieve its functionality. The "Rocket eBook," "Rio," and "PalmPilot," and are good examples:

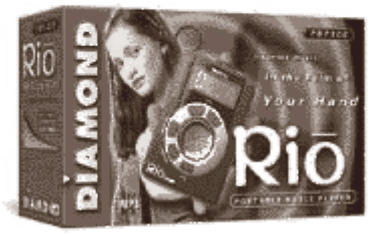
- The Rocket eBook is one of several hardware devices that display downloadable text. Packaged in a book size format, it's basically a retrieval, storage, and display device relying on dense, low power electronic memory.
- Rio is the musical version of Rocket Book. In a package the size of a telephone pager, you can pack MP3 audio files for playing CD quality sound with no moving parts, and therefore no skipping tracks. The key to its success is FLASH EPROM memory, very similar to that used in Xilinx FastFLASH XC9000 family CPLDs.
- The PalmPilot continues to be the leading Personal Digital Assistant (PDA), using access to Netscape and the Web to collect software and files and then use them when needed at some future time.

It's easy to imagine a single, FPGA-based device that can perform any of these functions, as well as functions that have not been thought of yet, easily adapting to entirely new applications, downloaded from the Web. The universal Internet Appliance is not only possible, it's inevitable.

Important PLD Features

Xilinx FPGAs and CPLDs are primed to play a key role in this market, and many are already used in a variety in Internet Appliances. The key attributes that make this possible, include:

- Low power for battery powered applications.
- Internet software support for Internet Reconfigurable Logic (IRL) applications.
- Reliable In-System Programming, with over 10K cycles guaranteed, for repeated updating.
- Physical connection download capability.
- JTAG test capability.
- Imagination – what you bring to the party.



Conclusion

With today's programmable logic devices and enabling technologies such as Internet Reconfigurable logic, the possibilities for unique new Internet appliances are limited only by your imagination. Now you can create universal products that allow you to continually implement new creative ideas, adding features and complete new functions, at your customer's location.

With today's programmable logic devices and enabling technologies such as Internet Reconfigurable logic, the possibilities for unique new Internet appliances are limited only by your imagination.

FPGA SYNTHESIS

Where We've Been, Where We're Going

Exemplar Logic and Xilinx pioneered FPGA synthesis and have been working together for over a decade to advance and improve methodologies.

by Tom Hill, Silicon Vendor Relations Manager, Exemplar Logic, tom.hill@exemplar.com

ASIC synthesis experienced rapid growth in the EDA industry during the early to mid-'90s. However, it was the programmable logic industry, that pioneered chip design using logic synthesis. Pre-dating the introduction of the first commercial ASIC synthesis tools by ten years, PALASM and ABEL were being used to synthesize PAL devices. By the mid-'80s most board designs included a programmable logic device.

Our History

Xilinx, founded in 1984, had a novel idea for a new ASIC device called a field programmable gate array (FPGA). Their architecture differed significantly from the PALs in two ways. First, it employed SRAM technology to achieve reprogrammability, which pushed gate counts far beyond PALs. Second, it offered multiple logic levels on a single device, making it more versatile and more similar to gate arrays than the two-level logic structures of PALs. This second reason spawned a relationship between Exemplar Logic and Xilinx in 1987 that would profoundly change both companies.

In 1985, Ewald Detjens, the founder of Exemplar Logic, was participating in a UC Berkley research project developing algorithms for multiple logic level, interactive synthesis (MIS). Several classmates, wishing to apply these concepts to gate arrays, went on to form the origins of one of today's largest EDA companies. Ewald, a person preferring the road less traveled, was looking for something different when he discovered Xilinx.

Ewald, a person preferring the road less traveled, was looking for something different when he discovered Xilinx.

In 1987 Xilinx introduced its first automated design environment called PPR. Prior to PPR, designers placed and routed their devices by hand using a system called XACT. PPR liberated them from this painstaking process by accepting netlists of cells and automatically performing place and route. For the first time, Xilinx devices could be designed using 3rd party design entry tools and schematic capture was quickly introduced into their design flow. While broadening their appeal to gate array designers currently using schematic capture, Xilinx was failing to effectively reach the market it really wanted: the person already using programmable logic, the PAL designer using PALASM and logic synthesis.

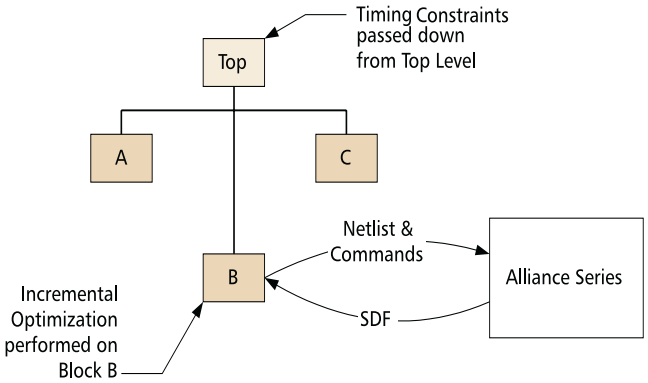
Working closely with some of the original software developers at Xilinx, Exemplar Logic introduced its first two products in 1988. Developed exclusively for Xilinx to target PAL designers, these products were called "PDS2XNF" and "XNFOPT." PDS2XNF converted PALASM files to XNF netlists of simple primitive gates and XNFOPT incorporated the MIS synthesis technology developed at UC Berkley, along with a new invention called LUT mapping, to efficiently optimize PALASM files directly into Xilinx Configurable Logic Blocks (CLBs). Exemplar Logic gave PAL designers easy access to Xilinx technology through logic synthesis and forever changed the face of the programmable logic industry.

From these humble beginnings PLD synthesis has grown to a $50 million a year industry and provides the chief method for designing Xilinx FPGAs. While the tools and devices have grown in sophistication, the design methodology has remained relatively consistent. Today, a single designer can complete most designs using a non-interactive, top-down synthesis approach and "on-board" verification. But will this meet the needs of tomorrow's FPGA designer? The answer is "no." Today's FPGA design methodology needs to evolve to avoid the productivity gap now being experienced by the gate array and standard cell communities where silicon densities have outpaced the designers' ability to utilize them.

The Virtex family represents a major advancement in programmable logic technology providing up to one million system gates of logic.

Our Future

The Virtex family represents a major advancement in programmable logic technology providing up to one million system gates of logic. With this dramatic increase in gate counts, the advantages of preserving design hierarchy are too great to ignore. Initial place and route can be performed early in the design process providing valuable device utilization and performance information while changes, made during final debug, can be limited to a local sub-block, maintaining an efficient design iteration cycle. FPGA synthesis and implementation must become interactive and iterative on a block by block basis to continue to provide an efficient design methodology for FPGAs.

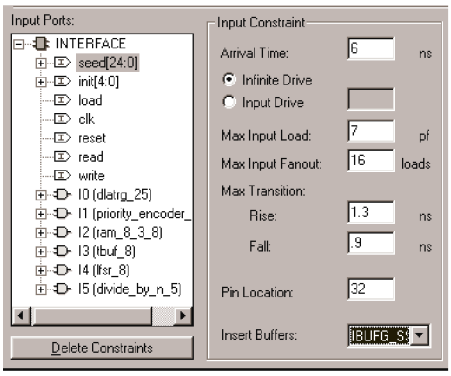


Leonardo Spectrum has been designed, not to hide design hierarchy, but rather to exploit its many advantages. A design hierarchy browser is an integral part of the user interface allowing you to easily access, manipulate, constrain, and swap hierarchical blocks. Incremental design is fully supported at the functional level allowing modules to be corrected at the RTL level, re-synthesized, and re-optimized while preserving all netlist information in the surrounding blocks. Incremental design is also supported at the synthesis level allowing constraints to be "tightened" on sub-blocks and re-optimized to correct timing or area problems discovered during place and route. Block-level design will provide the key to efficient interaction between the synthesis and place and route environments.

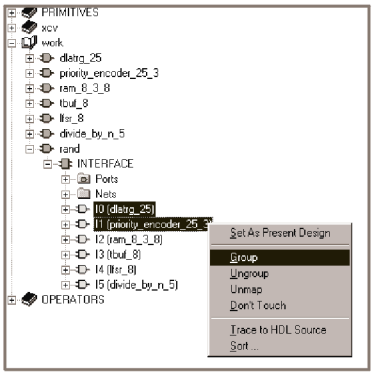
Conclusion

Exemplar Logic and Xilinx pioneered FPGA synthesis and have been working together for over a decade to advance and improve methodologies. Together we are committed to developing new technologies, including improved interactive block-based FPGA design, making programmable logic design the industry's premier design methodology. Σ

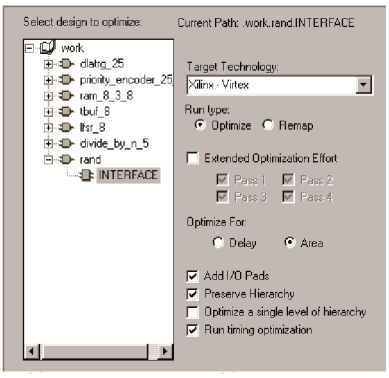
Constraint Editor



Hierarchy Browser



Optimization Interface



FPGA Technology Drives Design Software REVOLUTION

By looking at the changing use of FPGAs over time, we can understand the demands for a new generation of FPGA design software.

by Steve Bailey, HDL Solutions Manager, VeriBest Inc., sbailey@veribest.com

Over the last 15 years, Xilinx and the FPGA industry have evolved from providing FPGAs with densities of 100s of gates to devices with over one million system gates. Not only have system designs been adapted to exploit the capabilities of today's FPGAs in contexts not possible 15 years ago, designers have found the need to adopt new design methods and design software solutions that did not exist 15 years ago.

Soaring FPGA Device Complexity

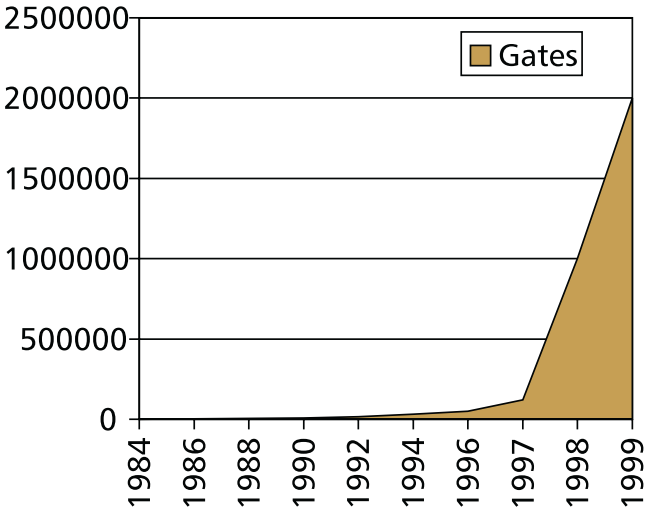


Figure 1

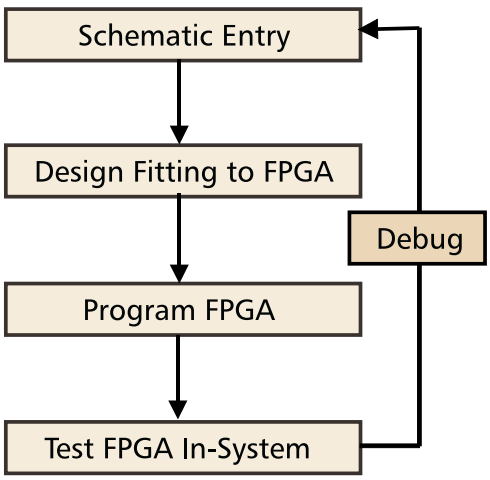
FPGA Usage Then and Now

Fifteen years ago, FPGAs were designed into systems primarily to reduce system component costs by consolidating board-level logic into fewer devices. A few hundred logic gates were replaced by a single FPGA that implemented the same functionality. Because the role of an FPGA was the consolidation of board-level logic into fewer components, FPGAs were designed by the board engineer.

The complexity of today's FPGAs, as shown in Figure 1, allows system architects to replace a broad range of ASICs with FPGAs and further consolidate and integrate the system logic into fewer and fewer components. FPGAs provide you with unprecedented flexibility at attractive costs. The advantage of no non-recurring engineering (NRE) costs, easy design modification, and in-system re-programmability make FPGAs a very attractive alternatives to ASICs. In fact, Ron Collett[1] reports that the number of boards with at least one ASIC has declined to 22% from 45% since 1994. Clearly, FPGAs are replacing more ASICs with each new technology update. As FPGA design complexity increases and more ASICs are replaced by FPGAs, more and more FPGA design is performed by the traditional chip design teams and not by the board engineer.

FPGA Design Methodology Then and Now

When the board engineer was using FPGAs primarily to integrate and consolidate 100s of gates of board-level logic, he used his time-tested board design methodology anchored by schematic capture as shown in Figure 2. It was easy and straight forward to insert a new level of hierarchy into a schematic for the FPGA and push the portion of the board schematic, that was being consolidated into the FPGA, down into the new hierarchical block. The engineer had already done



Simple Design Method for Simple Devices

Figure 2

Complexity Drives Adoption of HDL-Based Design

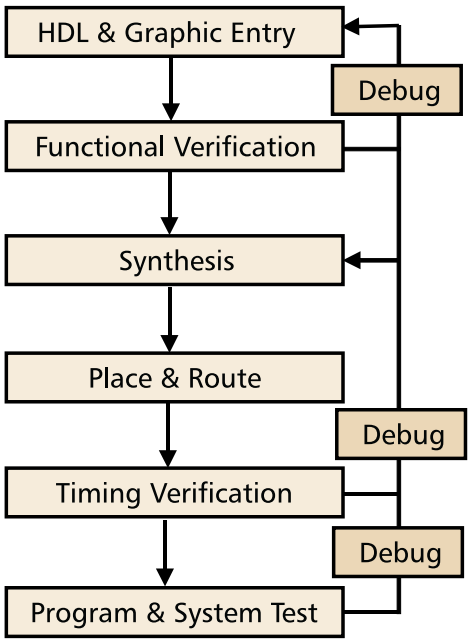


Figure 3

manual logic minimization and captured the design at the boolean logic and macrocell level. The only additional tool needed was a fitter (or FPGA place and route tool) to map the logic design into the FPGA with correct routing. The fitter then generated a chip programming file. Verification often consisted of prototyping.

Fifteen years later, the thought of designing a million-gate FPGA using a schematic design methodology defies rational thought. Today's FPGA designers are adopting HDL-based design methods at astonishing rates. HDL-based design, as shown in Figure 3, increases your productivity by allowing you to work at higher levels of abstraction — the register-transfer level instead of the boolean logic (gate) level.

Central to HDL-based design and the increased size of FPGAs are two strategically important tools: simulation for design verification and synthesis for automatic implementation of the RTL design to the gate-level (FPGA place and route level). Bread-board prototyping falls apart as a practical design verification method due to the cost of debugging

Powerful, Fast HDL Simulation Speeds Design Verification

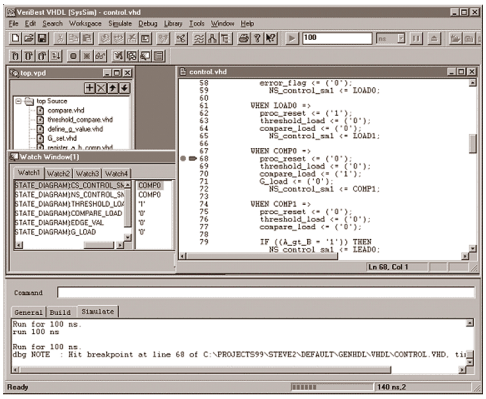


Figure 4

functionality after layout. Simulation allows design problems to be discovered earlier when it is more cost-effective to fix them.

Schematics and block diagrams still have a role in FPGA design, but that role is limited to manual implementation of tightly constrained functional blocks or to help manage complexity by graphically partitioning the design into smaller blocks.

FPGA Design Software Then and Now

Now that FPGA design is transitioning from the board engineer to the chip engineering design team, and design methodology is transitioning from schematic to HDL, FPGA design software must evolve to meet new needs.

Today's FPGA designers require the power of ASIC design tools within the traditionally, tightly integrated FPGA design environment. Fast, high-capacity HDL simulation gives you the ability to functionally verify FPGA designs at the Register Transfer Level and verify the dynamic timing and functionality after implementation.

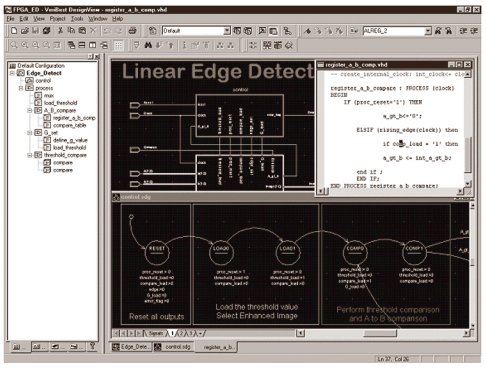
The size and functional complexity of today's FPGA designs require powerful simulators capable of handling the traditional needs of ASIC designers. VeriBest's™ HDL simulation products, shown in Figure 4, deliver technology-leading simulation speed and capacity, as well as powerful debugging capabilities, easily handling the largest of today's FPGA designs with fast verification and design debugging turnaround times.

Similarly, synthesis tools deliver productivity power by translating the design from the abstracted RTL to an architecturally optimized implementation. Synthesis tools must be able to handle the same language subset as their ASIC counterparts to facilitate re-targeting designs from ASICs to FPGAs. They must also provide ASIC-type quality of results for a variety of FPGA architectures. VeriBest's FPGA Desktop™ is available with FPGA Express™ from the industry's leading ASIC synthesis provider, Synopsys®. FPGA Desktop also fully supports integration with Synplicity's® Synplify™ for Xilinx designers who find that Synplify better meets their needs.

While FPGA design moves towards an HDL-dominated methodology, mixed schematic and HDL designs will be common for some time. Also, HDL editors are not always the best way to capture certain functional blocks. State diagrams are commonly employed to capture state machines. Similarly, many designers find flowcharts, truth tables, and state tables a more natural means to capture certain types of functionality. Finally, graphical design specification often results in design blocks that are more easily reused. Therefore, neither schematic-only nor HDL-only design capture environments are sufficient to support today's FPGA designs. That''s why VeriBest's FPGA Desktop, shown in Figure 5, provides a rich set of design capture editors including

Continued on the following page

REVOLUTION



Graphical and HDL Entry in FPGA Desktop

Figure 5

schematic/block, HDL editor, state diagram, truth tables and state tables, flowcharts, and boolean equations.

Although today's FPGAs need powerful design software, you still require the productivity of the traditional FPGA design software solutions, but you need them extended to teams of designers; FPGAs still give you time-to-market advantages.

Powerful point tools address one dimension of your productivity and your ability to meet project and market schedules. Tool integration, design data, and flow management address the remaining dimensions. All of the tools within the FPGA Desktop environment are tightly integrated including third-party tools such as FPGA Express, Synplify, and the Xilinx Foundation Series place and route software. Design kits support each Xilinx device family providing additional productivity benefits. Tight integration provides you with a single environment to learn and master.

Design data and flow management are the catalysts for the productivity benefits of the integrated FPGA Desktop environment. FPGA Desktop not only understands the tools in the integrated flow, but it also manages the design data for you, whether you are working alone or as part of a team.

Data management is both user explicit, as with the automated management of the design block hierarchy and source files, and user implicit. FPGA Desktop implicitly manages the design data and flow by preparing each step in the design flow, such as creating the synthesis tool's project, configuring synthesis, and passing the synthesized netlist to the Xilinx Foundation Series software for place and route. By automating routine data management tasks, you are free to focus on the engineering specifics of executing each step in the design flow such as constraining synthesis or specifying block functionality.

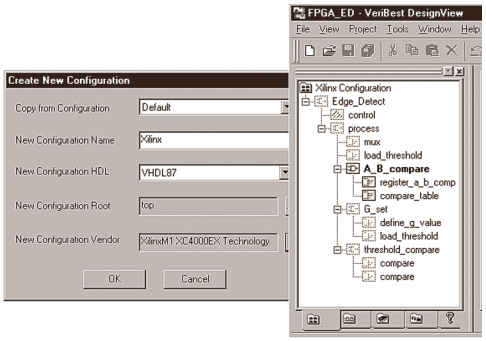
Consistent with the need for a powerful design solution, FPGA Desktop passively manages the design flow by facilitating the transitions between steps in the flow. It does not constrain you by actively managing your activity within a proscribed flow.

To explicitly enable teams of designers, FPGA Desktop allows the definition of multiple configurations for partitioning designs across multiple designers. You can also identify any block in the hierarchy as your design root for use as the focal point for each design step such as simulation and synthesis. Figure 6 shows the creation of a new design configuration with a focus on the portion of the design starting with the block A_B_Compare.

Conclusion

As Xilinx continues to lead the industry it helped create 15 years ago by driving FPGA device size and capability forward, FPGA (and ASIC) designers are evolving to fully exploit the new generation of FPGA devices by replacing more ASICs with FPGAs. As FPGAs replace ASICs, FPGA design is moving from board engineers into chip design teams.

Both new and experienced FPGA designers find that adopting HDL design methods helps them meet their tight time-to-market requirements while designing ever larger and more complex FPGAs. The combination of these FPGA technology and design trends increase the need for FPGA design solutions that provide tools powerful enough to handle ASIC designs while also delivering the productivity of an integrated FPGA design flow. VeriBest's FPGA Desktop delivers the power and productivity that today's and tomorrow's teams require, for designing with Xilinx devices. Σ



Configuring Designs and Managing Hierarchy

Figure 6

[1] "ASICs Not What It Was," Ronald Collett, Design Technology ROI, ***EE Times***, 1 February 1999, page 45.

Integrate FPGA & System Design Using Concept® HDL

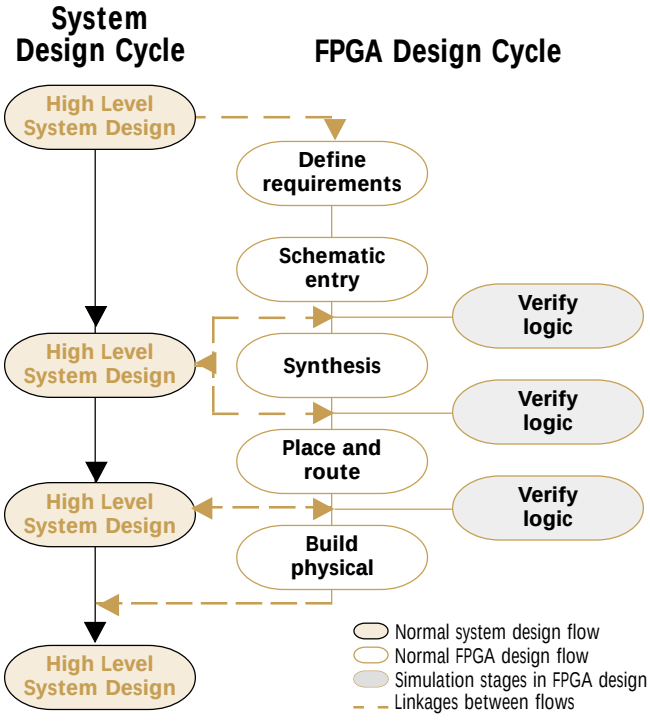
by S.Dharmarajan, Senior Member
Technical Staff, Cadence Design Systems,
rajan@cadence.com

Concept HDL from Cadence Design Systems takes a big step forward in integrating System and FPGA design cycles. The latest release of Concept HDL (PE 13.5) provides many new features for FPGA design, including the capability to concurrently design the FPGA and the system that uses it.

Concept HDL has come a long way from the times when schematic entry was limited to adding gate-level primitives from FPGA vendor libraries. Now it supports top-down, mixed-level, and bottom-up FPGA design flows. From the system design perspective, Concept HDL provides a framework that seamlessly integrates FPGA and board design cycles. FPGA designers can use all Concept HDL features already available to the system designers. They can use Global Find, Global Navigate, Hierarchy Editor, and design reuse capabilities in addition to the features of earlier releases. This article focuses on how this high level of integration between the Concept HDL-based board design flow and FPGA flow can benefit you.

The Integrated Flow diagram shows the system design phases and their integration with the FPGA design phases. This article illustrates this flow using the example of an image-processing card based on a Xilinx FPGA for use with a general-purpose microprocessor-based computer.

Integrated FPGA and Board Design Flow



High Level System Simulation

Concept HDL, based on the HDL-centric data model, provides the power of the HDL-based methodology with the convenience of schematic-based design. While designing an FPGA as a part of a system, you may initially simulate the system using its behavioral model to refine the expected behavior of the FPGA. This feature is especially useful for systems with many FPGAs. You can use different simulation models for the same component. You may simulate your designs using a combination of Verilog®, VHDL, smart models, and Hardware models. The HDL-centric architecture also provides for easy third-party tool integration and a high level of integration with simulators like Verilog-XL simulator, Affirma™ Native Compiled Verilog™ simulator, Leapfrog® VHDL simulator, and Affirma® Native Compiled VHDL simulator. You can also cross-probe the HDL code from Concept HDL — a valuable aid for debugging.

The facility to mix and match different types of simulation models provides a greater degree of design freedom. You can refine the expected behavior of the FPGA through these high level simulations and use this model to drive the FPGA design. The test vectors developed here can be used through the design cycle.

For the image processing card example, you may verify the image processing interfaces and algorithms by simulating a behavioral FPGA model with simulation models of the microprocessor, memory elements, and other system components. Thus, Concept HDL's powerful simulation capabilities allow you to refine the FPGA requirements very early in the design cycle using top-down synthesis and design of blocks

You can either directly synthesize the behavioral description of the FPGA using Synplify or design the FPGA using Concept HDL's extensive design capabilities. Concept HDL is fully integrated with Synplify from Synplicity for design synthesis so that you can specify Synplify synthesis attributes that drive and control synthesis, directly on the schematic blocks. Concept HDL automatically passes the synthesized structural netlist from Synplify to the Xilinx place and route tools.

Continued on the following page

Alternatively, you can create a complete design for the FPGA in Concept HDL. You can even choose to synthesize only parts of the design. This design can be flat or hierarchical, based on your preferences. Concept HDL helps you quickly create schematics from symbols or create symbols from schematics, and ensure consistent interfaces between different views. Concept HDL's design reuse features allow you to reuse older designs and save valuable design time. Concept HDL's simulation framework also allows you to simulate the FPGA block separately or simulate just some parts of it. You can easily simulate and compare the pre- and post-synthesis simulation results to verify the synthesis.

For the image processing example, you can reuse multipliers designed earlier, and synthesize only the control logic. Thus, the full integration with simulation helps you verify the functionality at any stage of the design. Concept HDL's extensive features and flexibility coupled to the integration with Synplify help you design and verify complex FPGAs with ease.

Smooth Flow with Xilinx Place and Route Tools

Once you complete the logic design for the FPGA block, you can pass the design to the Xilinx place and route tools by just clicking a few buttons. You can also directly invoke the Xilinx Design Manager from the Cadence Programmable IC flow.

After your FPGA design is complete, using the Xilinx Alliance Series software, you can do post-route simulations using the post-route HDL files and timing data. SDF back annotation and simulation features let you compare simulation results among the behavioral model, logic-level FPGA design, and post-route FPGA design. The Concept HDL framework allows you to use the same test vectors for different views of the design (pre-route, post-route) making the comparison easier. Thus you can verify that the FPGA meets design requirements after including all physical delays.

Concept HDL's framework allows you to verify the correctness of your designs at various levels of abstraction, at different stages of your design cycle, with minimal effort.

Easy Transfer of Data Between FPGA and PCB Tools

After your FPGA design is complete, you can automatically create a view for Allegro®, Cadence's benchmark PCB design tool. The normally tedious process, to ensure correct integration for system-level simulation and PCB design, is fully automatic, saving valuable design effort. This ensures that it is very easy to incorporate engineering changes. It is also possible to specify properties and constraints to guide the PCB tool on the FPGA design itself. The painless and reliable transfer of FPGA design information to the PCB design solution is a key feature of the Concept HDL design flow.

Managing Signal Integrity Problems with Your FPGA

After the PCB design is complete, you can generate board-level SDF data for system-level simulation to ensure timing correctness. Full integration with SPECCTRAQuest™ provides easy access to powerful high-speed system design and verification features without spending time and effort in translations. You can verify signal integrity of all signals interfacing with the FPGA before you commit to manufacturing. This is very useful for the new high-speed FPGAs from Xilinx. SPECCTRAQuest also provides you with power plane modeling and EMC analysis features critical for high-speed designs.

All this can be done without any database translations or going out of the Cadence Board Design Environment. This ensures that your design intent is captured and carried from the beginning to the end of the design cycle. This integration with Cadence's PCB design solution makes design of high speed systems more reliable.

Conclusion

Concept HDL's powerful features for concurrent system and FPGA design, along with seamless integration with Cadence's PCB design solution, provides you with advanced capabilities to reliably design and verify large and complex system designs using FPGAs. It also allows for better and smoother communication between the system design and FPGA design. Using Concept HDL gives you the flexibility and power to effectively utilize the capacity of the latest Xilinx FPGAs without worrying about the CAD tools.

In the future, Cadence's Concept HDL will also provide features for concurrent design, generic data management, and other advanced features for system design. A tight integration with Cadence's new Integrated Native Complied Architecture for logic simulation will greatly speed up mixed language simulation.

For more details about the features of Concept HDL and other Cadence products, please visit the Cadence website at www.cadence.com or contact your local Cadence AE.

Concept HDL ... provides you with advanced capabilities to reliably design and verify large and complex system designs using FPGAs.

VeriBest's Vision of the Future

VeriBest, Inc., a Xilinx Alliance partner, remembers the early beginnings of EDA tools for Xilinx design, and provides a view of the future.

Remember the XC2000? I was a programmer in Huntsville, Alabama looking at designing an EDA environment for efficiently developing XC2000 and XC3000 designs. That was about ten years ago and the terms "Graphical User Interface" and "Design on NT" were unheard of in the design community. We've come a long way from the days of DOS command lines to the new vision of DesignView™ — VeriBest's hierarchical development environment that supports the top-down, middle-out, and bottom-up design methodologies necessary for the future of IP design reuse.

From Command Lines to Tri-Pane Design Desktops

Over the years, graphical interface design has become a key issue. You want to spend time on design, not memorizing command line sequences. Today, this environment is known as DesignView. Imagine a front-end design environment using multiple block views and multiple configurations allowing you to express "what if" structures in any combination of formats while simultaneously targeting multiple Xilinx families. Couple this ultimate expression of a design with seamless stimulus generation, LogiBLOX generation, integration with HDL simulators, synthesis tools, and Xilinx design tools. What you get is the power you need to have your designs meet schedule demands with the flexibility to explore your architectural alternatives; in essence, DesignView dramatically increases your productivity.

Providing much more than the flow managers of the past, DesignView allows you to *design the way you think* and view the design from the level most comfortable for you.

DesignView Work Surface

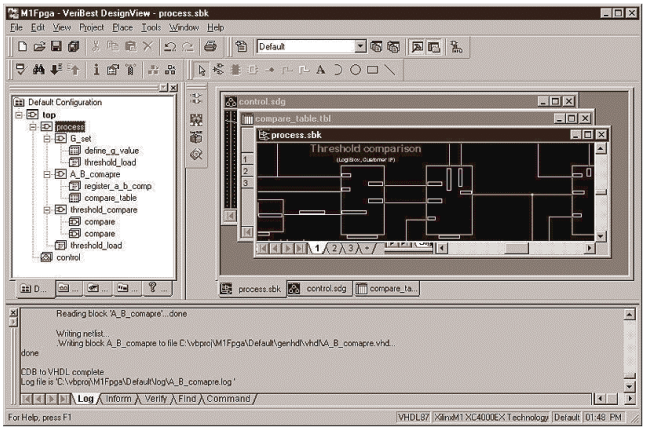


Figure 1

DesignView Work Surface

Let's look at DesignView's tri-pane work surface. The three work surface areas are:

- The Project Workspace area.
- The Design Editing area.
- The Output communication area.

The Project Workspace area contains five project tabs. These tabs allow you to access the:

1) Design hierarchy.
2) Design simulation environment (including associating and storing multiple stimuli for each hierarchical block).
3) Xilinx Place and Route files.
4) Project file manager.
5) Infoviewer (including links to the Web and Help files for VeriBest and Xilinx design tools).

Continued on the following page

The Design Editing area is the workspace area for all the design editors. Your design may consist of schematics, graphical high-level designs (incorporating state diagrams and flowcharts), and HDL (hardware description language) designs in Verilog and VHDL. A workbook mode is available, allowing you to instantly access your design files from a tabbed workbook.

The Output Communications area provides reports on tool operations, such as design verification, project searches, and HDL compilations.

The implementation of this tri-pane window environment makes DesignView unique.

From Design Flows to Design Centric

Over the Xilinx 15 year history we have witnessed dramatic changes in design processes. The traditional top-down design flow now uses bottom-up flows that permit bottom-up verification. Middle-out flows permit synthesis and verification of single elements.

The primary failure of traditional design-flow-oriented EDA tools has been the limitations imposed on you. DesignView avoids this classic mistake by allowing all design process variants. In fact, DesignView is the only EDA tool that allows you to mix multiple design processes to complete a single design.

The DesignView design environment is “design centric” which gives you direct access to any block design file, allowing you to generate a block-specific stimulus and then send the block and its children to the HDL simulator, or to your synthesis tool, with a single menu click. DesignView’s unique design-centric environment gives you the flexibility you need to design from the top down, the bottom up, or the middle out.

Decide what block-specific operations you would like to perform using the Hierarchy Browser shortcut menu, as shown in Figure 2.

Hierarchy Browser Shortcut Menu

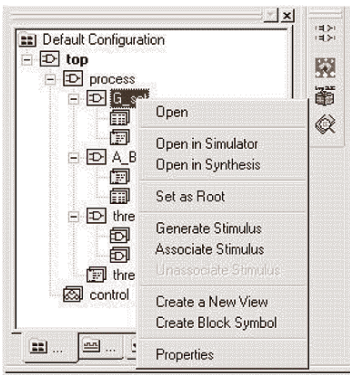


Figure 2

Over the Xilinx 15 year history we have witnessed dramatic changes in design processes. The traditional top-down design flow now uses bottom-up flows that permit bottom-up verification. Middle-out flows permit synthesis and verification of single elements.

You don’t need to worry about which HDL libraries to load into the simulators or which EDIF netlist switches to use — DesignView anticipates next tool use and automates tedious background configuration operations and file preparation for you.

Views and Configurations

Being able to view your design in a method most comfortable for you is unique with DesignView. You may create multiple views for each block in your design, by setting the “active” view from the hierarchy browser. Each block view has a corresponding design file in the Design Edit area. Display of the hierarchy is in a “default” configuration.

A configuration is the set of active block views, the selected root, the selected vendor technology, and the selected HDL language. You can create as many configurations as you want, and you can change parameters such as the vendor technology. Or, you can change the root and then simulate and route only a portion of your design, or even change the HDL language for some special simulation requirements. You can also switch between configurations and run “what-if” analysis without the hassle of recompiling or re-synthesizing; just set the configuration and you are off and running.

From X-BLOX to LogiBLOX Schematic Symbol Generation and Integration

As Xilinx moved from X-BLOX to LogiBLOX, providing even greater customization capabilities, VeriBest further enhanced LogiBLOX capabilities. The Xilinx LogiBLOX Module Generator is an integral part of DesignView, automatically generating symbols for each of your unique LogiBLOX macros. Your LogiBLOX macros appear in your Hierarchy Browser and appear to the simulator and synthesis tools as part of your HDL file list. The generated LogiBLOX symbols are identical to those displayed in the LogiBLOX selector helping you identify your unique LogiBLOX symbols on the schematic pages.

The word "dynamic" hardly describes the change that has occurred during these growing years for Xilinx, VeriBest, and the FPGA marketplace. Happy birthday Xilinx — from our view, the design future looks bright!

Design Reuse – From Graphical to HDL Based Design

The last 15 years have brought around more than just changes in design methodologies. HDL-based design and the emergence of VHDL caused us all to look at HDL generation as a key element. DesignView allows you to start your design graphically and then generate the version of HDL that you need. Taking a lesson from the programming community, reusable HDL design modules will be the future-DesignView is ready to meet that need now.

WaveBench™ – The Graphical Stimulus Editor

Part of the Design Editor offering is WaveBench. You invoke WaveBench directly from the design centric shortcut menu by selecting "Generate Stimulus" for the highlighted block. WaveBench provides you with the input signals and allows you to enter waveform data for each signal. Because you think of stimulus in terms of waveforms, we saw the need to have the ability to express stimulus in waveforms, with automatic HDL generation.

If you need more power, you can use Microsoft® Excel or Visual Basic, or Perl, to define your stimulus patterns and then read the information into WaveBench to display it, tweak it (when necessary), and then generate your new stimulus.

WaveBench Stimulus Generation Editor

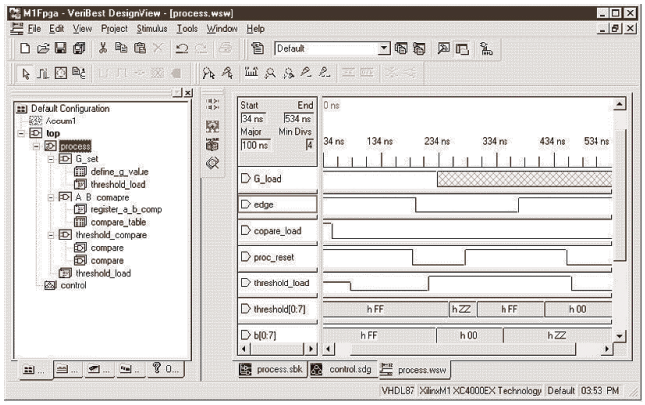


Figure 3

Emergence of HDL

With the emergence of design reuse, HDL design plays a more important role than ever before. VeriBest provides HDL simulation that's simply easy to use and ready to meet the language constructs needed for supporting the variety of synthesis processes used today. Launching simulation is seamless through the shortcut menu, allowing you to stay focused on your design.

Synthesis – Take Your Pick

DesignView supports both Synopsys FPGA Express 3.1™ and Synplicity Synplify® synthesis processing. DesignView automatically generates and manages the project files for your synthesis tool of choice. Your selected tool launches with the correct set of HDL files pre-loaded and ready for analysis.

Post-Synthesis HDL files are displayed to the Vendor Manager Tab of the Project Workspace letting you invoke the HDL simulator on the selected HDL file for a post-synthesis simulation session. Post-Synthesis vendor netlists are displayed in the Vendor Manager tab also. Once a vendor netlist is available, the Xilinx Designer icon is activated on the DesignView design kit toolbar. You are now ready to continue on to place and route.

Conclusion

The word "dynamic" hardly describes the change that has occurred during these growing years for Xilinx, VeriBest, and the FPGA marketplace. Happy birthday Xilinx — from our view, the design future looks bright!

For more information on DesignView, the core of the VeriBest FPGA Desktop product offering for Xilinx Design, visit http://www.veribest.com/sales/datasheets/3300/3300.htm. Σ

Prototyping ASICs Using Xilinx FPGAs and Certify™

by Jeff Garrison, Senior Product Marketing Manager, Synplicity, jeff@synplicity.com

The high capacity and high performance of Virtex FPGAs combined with the unique partition-driven synthesis algorithms in Certify means that even the largest multi-million gate ASIC can be prototyped with a manageable number of Virtex devices.

There are an increasing number of applications that can no longer be verified using software simulators alone. The time required to simulate a large data processing application such as video with a traditional HDL simulator is no longer practical. Furthermore, many applications need to achieve specific speed targets for the prototype to be useful. For this type of application there is a strong movement in the market toward prototyping ASICs using FPGAs. As Xilinx celebrates 15 years of success in the programmable logic market, the latest, high-density Virtex family offers an ideal solution.

ASIC designs are often prototyped on a single FPGA device, and Synplify is the ideal implementation tool for these designs. However, for very large ASIC designs, there is a need to partition the design across multiple FPGA devices. One big problem with this emerging prototype methodology is that the time and expertise required to partition a design across multiple FPGAs is very high. Designers need to make multiple time-consuming iterations between synthesis and partitioning to find a "legal" partition for the design. In addition, synthesis is typically done without understanding the partitioning, resulting in a prototype which does not run at the target speed.

Synplicity has enabled a powerful prototyping methodology through a new tool called Certify. Certify takes a fresh approach to the prototyping problem by integrating partitioning and logic synthesis. By performing synthesis during the partitioning phase, you are presented with accurate area and pin utilization information so that partitioning decisions can be made in a fast, interactive environment. A second benefit is that Certify performs partition-driven synthesis. In other words, the physical partitioning data is used to affect synthesis along with your timing constraints.

Certify is the only tool that combines multi-chip partitioning with RTL synthesis. The features and benefits of Certify are summarized in Table 1 below.

Certify's Approach is Better

The level of abstraction and automation in FPGA design tools is moving continually upward to keep pace with the new high-complexity, high-performance FPGAs being introduced. One common element that is critical in linking these new tools together, in the deep subμ world of FPGA design, is logic optimization and technology mapping-the two main steps involved in logic synthesis.

Table 1. Certify Features & Benefits

FEATURE	BENEFIT
Best-in-class FPGA synthesis	Optimal speed and efficiency of prototype
Partition-driven synthesis	Manages time budgets across FPGAs
RTL Partitioning	Partition user's HDL code, not ASIC netlist; Easier to use, higher productivity
Partitioning aids and impact analysis to guide user	Shortens time to prototype
Fast, accurate feedback on I/O and area utilization	Reduces iterations between partitioning & layout
Million+ gate capacity	No need to break up design into many small blocks
Manages Logic Replication	Optimal partitioning without changing source code
Manages Probe Point Creation	High observability for debugging without changing source
Optimized For Iterative Design	Reduces time to fix errors
Understands characteristics of prototype board	Optimal speed and efficiency of prototype
Supports popular FPGA vendors	Flexibility in implementation choice

In today's HDL-based design methodologies:

- The HDL source describes the function of the device.
- The first synthesis step (compilation) interprets the HDL in terms of registers, logic and arithmetic operations, and control circuits.
- Logic optimization attempts to reduce the complexity of the implementation using sophisticated algorithms. To this point, the design has remained independent of the implementation device architecture (FPGA device family).
- Technology mapping represents the design in terms of a specific device architecture.
- Timing optimization attempts to improve the design to meet the designer's timing constraints.
- Placement locates each technology cell within the chip.
- Routing allocates the wiring resources to provide the necessary interconnections.

The naïve assumption that each of these steps is independent leads to non-convergent iteration loops in reaching a timing goal. At each stage, the tools (or the designer, in the case of HDL source) make implementation decisions based on estimates of what will happen in the remainder of the flow. Each following stage is stuck with the decisions made at earlier stages. In the independent model, the placement stage cannot change logic to make its job easier.

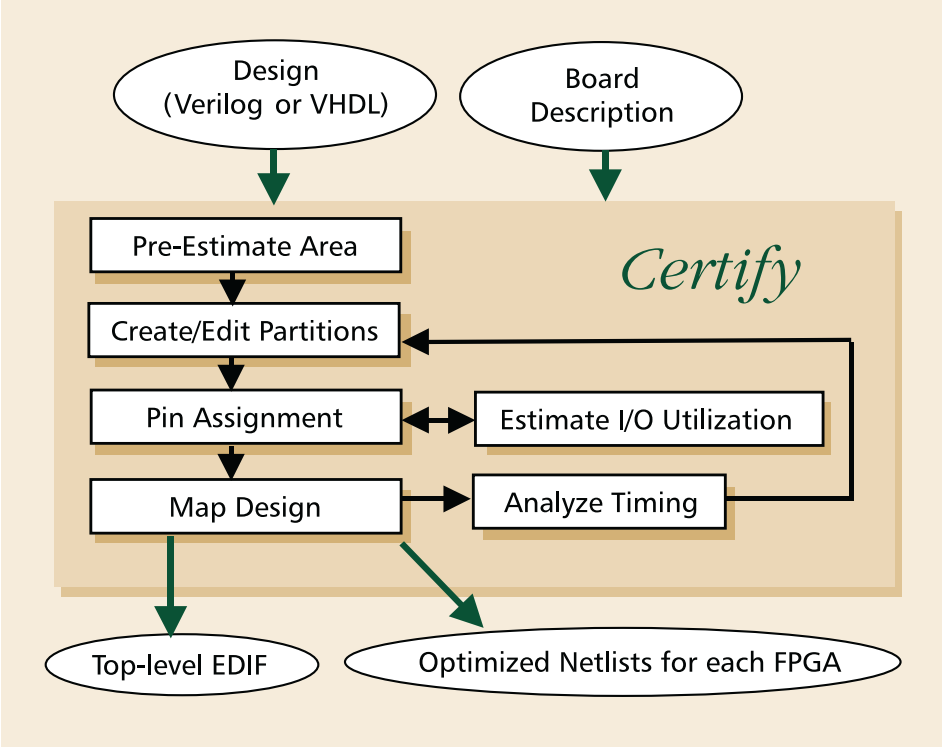
The fundamental limitation to this flow is that changes made in early stages have the most impact, but the ability to estimate is weakest there, while the later stages have the best information, but the ability to make significant changes is gone. To overcome the limitations of the flow, we must improve estimation for the early stages, and preserve high-level information for the later stages. This approach is called partition-driven synthesis. It is at the heart of the powerful new algorithms in Certify, and allows you to achieve the fastest possible prototype speed.

The Timing Estimation Problem

If improved timing estimation is necessary to break the endless iteration cycle, then we must first understand the components of timing, and what optimization tools can do to change the timing. With FPGAs of 100K gates or more, good estimation requires an understanding of the underlying architecture of the device. The speed of an FPGA design depends on many factors:

- The delay in the simplest programmable logic cell.
- The delay introduced by the programmable routing.
- The interconnection of logic to form a critical path through the circuit.
- The timing requirements of the external circuitry.

The delay in the simplest programmable cell is easy to estimate once the design has been mapped to a particular device architecture, but very difficult to estimate before mapping. The cell is intentionally very rich in its ability to represent logic functions, but the cells' capabilities vary greatly among FPGA device families. Hence, good estimation requires mapping to the chosen device architecture.



The speed of an FPGA design depends on many factors ...

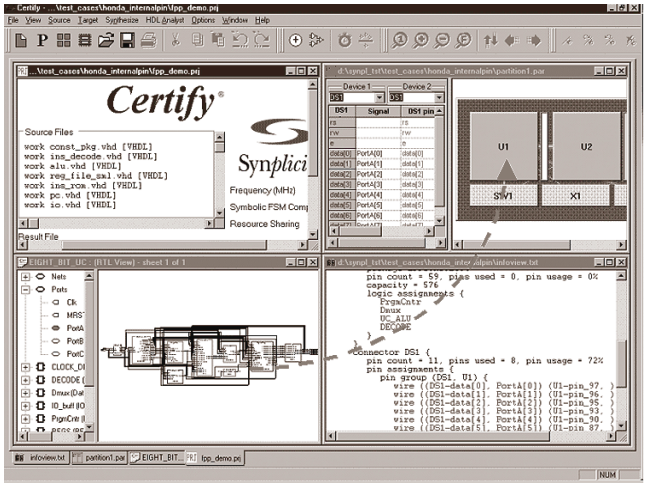
Continued on the following page

Of course, the most effective changes are major changes to the circuit topology that derive from a deep understanding of the design. This is the technique used by Certify's unique partition-driven synthesis algorithms to produce optimal results.

In a typical FPGA, the sum of the routing delays along the critical path comprises well over half of the total delay. In a mapped design, routing delay is by far the most important estimation. The delay due to a particular interconnection depends on the capacitance and resistance of the interconnection, which are in turn dependent on the physical location of the connections. It also depends on the routing architecture for the programmable device, and on the size of the particular device. Good estimation of routing delays depends on good estimates of physical placement, and on a detailed knowledge of the device architecture.

Estimating timing accurately becomes even more important when dealing with designs partitioned across multiple FPGAs. The board delays must be accurately taken into account in estimating any critical paths, and the synthesis tool must account for these delays in its optimization algorithms. The static timing analysis algorithms for finding the most critical paths, and the techniques for considering the external timing requirements are complex but well understood. Given a mapped design with good interconnect estimates, static timing analysis will reliably identify the most critical path.

Timing optimization tools have only a few options to improve the timing:



Partition by dragging and dropping HDL modules to different FPGAs

- Change the circuit topology of the critical path so that the new critical path has fewer delays.
- Change the placement so that it is easier to route with less resistance and capacitance.
- Change the routing to reduce resistance and capacitance.

Of course, the most effective changes are major changes to the circuit topology that derive from a deep understanding of the design. This is the technique used by Certify's unique partition-driven synthesis algorithms to produce optimal results.

Certify follows in its predecessor's (Synplify) footsteps for providing an intuitive and easy to use user interface.You input Verilog or VHDL code for the design along with a PCB description (also in HDL). Certify then performs a fast estimation of area and pin utilization for the HDL modules in your design. This utilization information is instantly provided to you as you drag and drop modules of HDL in the various FPGAs in the system.

After a partion has been created, Certify uses the physical partition to drive the synthesis process. Certify has the ability to optimize logic across FPGA boundaries resulting in higher performance and better overall device utilization. The output of Certify is a top-level EDIF description of the system, and highly optimized netlists for each of the FPGAs, ready for Xilinx place and route tools.

Conclusion

By utilizing Synplify's lightning-fast synthesis algorithms with Behavior Extracting Synthesis Technology™, Certify has the ability to handle huge designs. Even a prototype that requires six XCV1000s (6 Million FPGA gates) may be partitioned and fully synthesized in a few days as opposed to several weeks, not to mention the better results achieved with Certify by synthesizing across FPGA boundaries.

For more information on Certify, please contact your local Synplicity sales representative or call (408) 548-6000. Σ

The Increasing Importance of HDL Verification

OK
Not OK

How the new generation of HDL simulators can help you design the largest FPGAs with a minimum amount of time spent on the simulation process.

by Gregor Siwinski, Director of R&D, Aldec Inc., gregor@aldec.com

When Xilinx users first started creating FPGA designs based on schematics, the only verification technology available was based on a gate-level netlist simulator. The time and effort required to create tool-specific simulation test vectors and the relatively small gate count of designs made it unnecessary to use a simulator in the design flow. It was easier to program the device and test it in hardware than to verify it using the simulator.

Today, with FPGA capacities exceeding one million system gates, simulators can actually save you time by detecting problems early in the design flow. For most large designs it is practically impossible to create a reliable FPGA without using a simulator.

Behavioral RTL Simulation of HDL Code

The term "behavioral RTL simulation" is used here to describe simulation of an FPGA design prior to running any synthesis or implementation tools. Behavioral simulation verifies that your HDL code is correct and detects any functional problems. For large designs that take hours to synthesize it will save a lot of time if the FPGA can be functionally tested without running the synthesis after each change. Furthermore, behavioral simulation is typically 10 to 100 times faster than post-synthesis simulation of the same design.

Today, with FPGA capacities exceeding one million system gates, simulators can actually save you time by detecting problems early in the design flow.

Advanced HDL Debugging

Simulating HDL in a debug mode allows you to analyze your design source similar to software debuggers:

- Execute your HDL code in trace mode one line at a time.
- Breakpoints can be on any line of HDL code or signal change.
- Variables can be monitored and modified during simulation.
- Design hierarchy and signals can be viewed to inspect a value of any signal or port in the design.
- The Dataflow view presents a graphical view of a signals connectivity in the HDL design.
- Waveforms and event list windows can be used to monitor the results.

Continued on the following page

HDL simulation in debug mode using Active-VHDL™ software

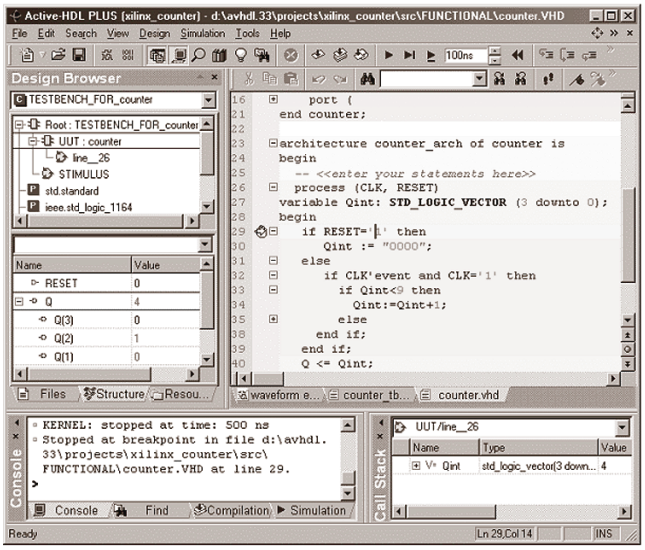


Figure 1

In most cases, the simulation of a design after synthesis should output the same results as the behavioral simulation. The purpose of post-synthesis simulation is to make sure this is the case...

HDL Test Bench

Creating simulation input can be a tedious process. That's why it is important to know all possible entry methods and use them to save design time:

- Graphical stimulators are very useful to set signals to a desired state, define clocks and formulas.
- HDL test bench files can be generated from waveform diagrams.
- HDL test bench with Smart Comparison™ using IEEE 1029.1 WAVES standard.
- Script files can automate the entire verification process (such as TCL, PERL).

Waveform-based HDL simulation with familiar stimulators

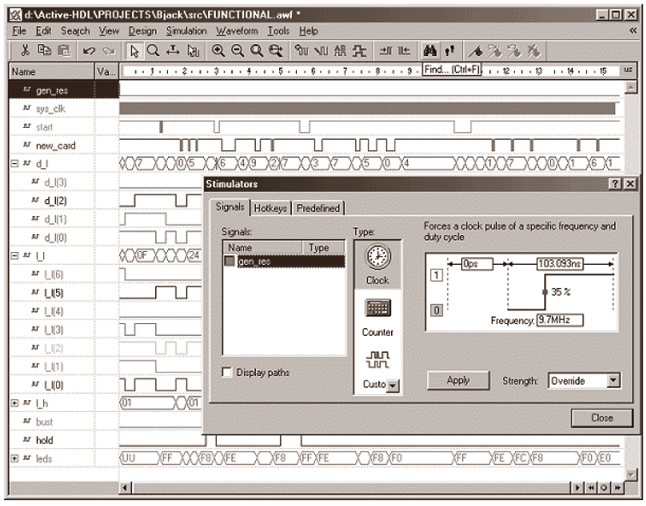


Figure 2

One very important benefit of using HDL simulation is that the test bench can be described in the same HDL language as the design itself. Aldec also generates a script file that will compile the necessary HDL files and automate the entire verification process.

Xilinx Foundation Series Designs

Many Xilinx Foundation Series software users create designs containing schematics. Schematic-based designs and mixed schematic/HDL designs can also be simulated in HDL simulators. The Foundation Series software provides a seamless interface to HDL simulation that will export all schematic portions of the design to VHDL and then invoke the VHDL simulator from the Foundation Project Manager.

Users who create graphical state machines in the Foundation Series are also able to animate the FSM diagrams during simulation. Also, the Foundation test vectors can be imported to Active-VHDL for easy transition to the HDL simulation environment.

Post-synthesis Verification

In most cases, the simulation of a design after synthesis should output the same results as the behavioral simulation. The purpose of post-synthesis simulation is to make sure this is the case and that the synthesis tool output produced the netlist which is functionally the same. Synthesis programs may implement your HDL code in a different way than you expected and the post-synthesis simulation will detect that.

Synthesis tools like Foundation Express can produce a netlist in VHDL and Verilog formats for HDL verification. The best feature of this process is the ability to use the same simulation input or test bench that was used for behavioral simulation. The post-synthesis simulation results can be compared against original outputs graphically in the waveform window.

Xilinx Foundation software
provides integration to VHDL simulation

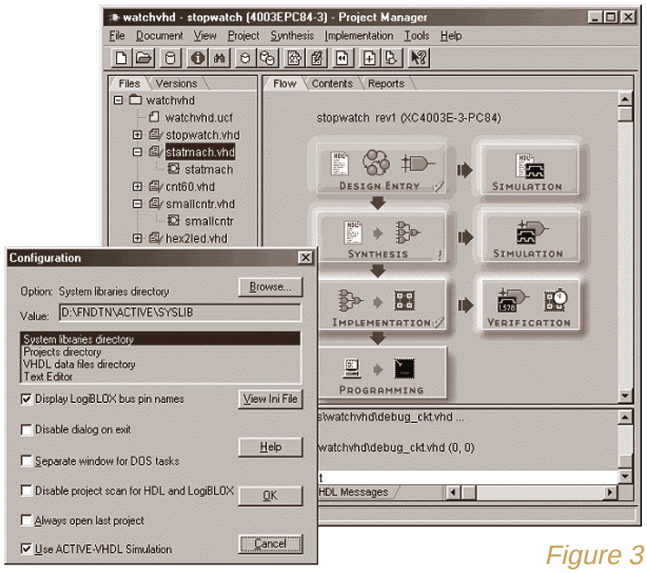


Figure 3

Timing Verification

After a design is implemented by the Xilinx tools, you have an option to export timing simulation data. This will generate an HDL simulation netlist and an SDF file containing calculated timing delays. The timing netlist uses the SIMPRIMS simulation library based on Vital primitives.

Accurate simulation of timing delays allows testing of the design functionality at the target frequency rate, to detect any setup/hold violation and timing glitches.

Similar to post-synthesis simulation, there is no need to develop new simulation test vectors. The same simulation test bench can be used for both behavioral and post-implementation simulation. All this is done in the same design and graphical environment.

Timing simulation can be very time consuming. However, with the accelerated Vital primitives used in Active-VHDL software the simulation speed can be five to ten times faster than in the Foundation Series gate-level simulator.

In the near future, you will see even more integration in the HDL design environment with features such as HDL graphical entry, code coverage analysis, and formal verification. This prepares us for the next generation of HDL verification tools.

Smart Comparison Test Bench

One obvious question is how to compare the simulation results between behavioral simulation and post-implementation timing simulation. You can no longer use the graphical comparison because of the delays that cause waveforms to shift.

The answer comes in the Smart Comparison based on the recently published IEEE WAVES standard. Among other test bench functions, WAVES provides a very convenient method of comparison of current simulation results with the golden reference previously saved in the vector file.

Active-VHDL Test Bench Wizard™ automates this process even further, saving the functional simulation results into a standard VEC file and generating the VHDL test bench program using the WAVES library. You only have to specify the so called "comparison window" which defines a period of time after each test pattern that is ignored to take account for timing delays. All discrepancies are detected during simulation giving you detailed messages about signal and time when the simulation results are different. Both expected and actual waveforms are displayed in the same screen.

Smart Comparison shows
actual and expected waveforms

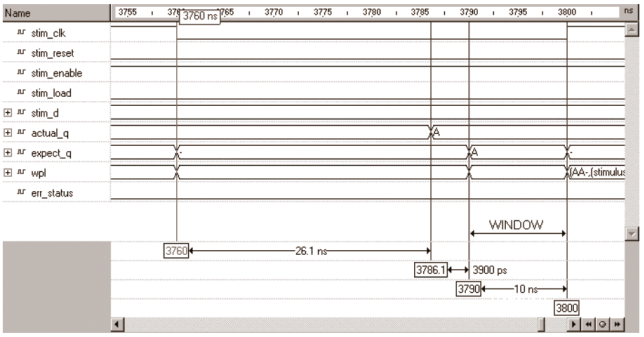


Figure 4

Summary

As you could see, the verification tools have come a long way over the last 15 years. If you would like to learn more about the benefits of HDL simulation please visit Aldec's website at http://www.aldec.com/activevhdl

In the near future, you will see even more integration in the HDL design environment with features such as HDL graphical entry, code coverage analysis, and formal verification. This prepares us for the next generation of HDL verification tools.

Converting XC4000E/EX FPGAs to ASICs

Xilinx recently released the XH3 family, a new HardWire™ conversion technology for XC4000E and XC4000EX designs.

by Rob Schreck, Product Manager, Xilinx, rschreck@xilinx.com

The XH3 architecture is a revolutionary step in FPGA-to-ASIC conversion, providing an effective silicon platform that closely emulates the XC4000E and XC4000EX architectures.

We have offered low-cost ASIC versions of our FPGAs since 1991, making it easy for you to enjoy the benefits of programmability and still reduce the cost of high volume designs. The same design files used to create your original FPGA are used throughout the HardWire conversion process; both the placement and routing of your original FPGA are preserved.

Our new XH3 family of HardWire products includes critical Xilinx FPGA features built into the die. Because of the match in feature set, we can provide form, fit, and function equivalence that closely matches our FPGAs.

Xilinx XH3 Features

The XH3 family features:

- The only turnkey, 100% compatible conversion path for Xilinx FPGAs.
- Patented DesignLock™ conversion methodology.
- Supports conversion of all Xilinx FPGA features, including Select-RAM™, JTAG, and configuration emulation.

An XC4000E or XC4000EX design is converted into a particular XH3 base array depending upon the number of gates and I/O required. Table 1 shows which XH3 base array is selected as the approximate target for each FPGA density and package size.

The Xilinx HardWire family is part of a complete solution, giving you fast and efficient prototyping and system development, as well as low cost, high volume production capability. Plus, FPGAs and HardWire products are interchangeable, offering flexibility for cost reductions, future design changes, and end-of-life management of your product.

For more information about Xilinx HardWire products, go to our website at:

http://www.xilinx.com/products/hardwire/hardwirehome.htm

Here, you will also see our "Xilinx ASIC Estimator." With this Web-based tool, you will be able to model all of your project costs (including some you may not have considered before) to see how inexpensive it is to use Xilinx FPGAs and the HardWire family for high volume applications compared to the hidden costs of using ASICs.

TABLE 1

		PACKAGE TYPE								
FPGA	**Max I/O**	**PC84**	**PQ/VQ100**	**TQ144**	**PQ160**	**PQ208**	**PQ240**	**BG225**	**BG352**	**BG432**
XC4003E	80	XH304	XH304							
XC4005E	112	XH304	XH304	XH304	XH304	XH306				
XC4006E	128	XH304		XH304	XH304	XH306				
XC4008E	144	XH306			XH306	XH306				
XC4010E	160	XH306			XH306	XH306		XH308		
XC4013E	192				XH308	XH308	XH308	XH308		
XC4020E	224					XH310	XH310			
XC4025E	256						XH312			
XC4028EX	256					XH312	XH312		XH312	
XC4036EX	288						XH312		XH312	XH312

Using Relative Location Constraints in Synplify For Improved Control of Timing and Placement

A short description of how and why to use RLOCs to control your design.

by Mala Sathyanarayan, Senior Corporate Applications Engineer, Synplicity, Inc., mala@synplicity.com

In Synplify, you can use Relative Location Constraints, RLOCs, to specifically control how logic is implemented within Xilinx structures such as FMAP, HMAP, and registers. Additionally, Synplify allows you to group these related structures together.

You will ultimately have small sections of your designs where logic placement is crucial to meeting system performances. RLOCs provide a powerful method of controlling placement in performance critical sections. These tailored RLOCs can be made as efficient as Xilinx relatively placed macros. This article summarizes how to use RLOCs with Synplify.

Synplify allows you to specify relative location constraints, which may be placed either in the HDL code or through Synplify's graphical constraint editor, SCOPE (Synthesis Constraints Optimization Environment). These location constraints are passed to the Xilinx place and route tool through the EDIF or XNF netlist. The attributes that you use are:

xc_rloc, **xc_map**, and **xc_uset**.

The example below shows how to implement a 9-input XOR gate into a single CLB. The first step is to define a 4-input XOR function (fmap or 4-input LUT), and a 3-input XOR function (hmap or 3-input LUT). The next step is to instantiate two fmaps (or 4-input LUTs, for Virtex) to XOR eight inputs, and one hmap (or 3-input LUT, for Virtex) to XOR the outputs of the two fmaps (or 4-input LUTs) and the 9th input. The final step is to instantiate the whole CLB in the top level design.

Create modules and specify them as **fmap**, **hmap** or **lut** (for Virtex) as shown below. Make sure that the module mapped to **fmap/lut** has a maximum of four inputs, and the module mapped to **hmap** has a maximum of three inputs.

Verilog

XC4000, Spartan family:

```
module fmap_xor4(z, a, b, c, d)/* synthesis
 xc_map=fmap */; // –(1)
output z;
input a, b, c, d;
assign z = a ^ b ^ c ^ d;
endmodule

module hmap_xor3(z, a, b, c)/* synthesis
 xc_map=hmap */; // –(2)
output z;
input a, b, c;
assign z = a ^ b ^ c;
endmodule
```

Virtex family: Replace line (1) and (2) with:

```
module fmap_xor4(z, a, b, c, d)/* synthesis
 xc_map=lut */; // –(1)
module hmap_xor3(z, a, b, c)/* synthesis
 xc_map=lut */; // –(2)
```

VHDL:

XC4000, Spartan family:

```
library IEEE;
use IEEE.std_logic_1164.all;
entity fmap_xor4 is
 port (a : in std_logic;
       b : in std_logic;
       c : in std_logic;
       d : in std_logic;
       z : out std_logic
   );
end fmap_xor4;

architecture rtl of fmap_xor4 is
attribute xc_map : STRING;
attribute xc_map of rtl : architecture is “fmap”; –(1)
begin
 z <= a xor b xor c xor d;
end rtl;

library IEEE;
use IEEE.std_logic_1164.all;
entity hmap_xor3 is
 port (a : in std_logic;
       b : in std_logic;
       c : in std_logic;
       z : out std_logic
   );
end hmap_xor3;
```

Continued on the following page

VHDL example for XC4000, Spartan family (continued)

```
architecture rtl of hmap_xor3 is
attribute xc_map : STRING;
attribute xc_map of rtl : architecture is “hmap”; –(2)
begin
 z <= a xor b xor c;
end rtl;
```

Virtex family:
Replace lines (1) and (2) above with:

```
attribute xc_map of rtl : architecture is “lut”;
```

Instantiate these modules at a higher level and specify the **xc_rloc** and **xc_uset** attributes on these instances. The **xc_uset** attribute is used to group the instances together, and the **xc_rloc** attribute specifies the relative locations of all the instances with the same value for **xc_uset**.

Verilog

XC4000, Spartan family:

```
module clb_xor9(z, a);
output z;
input [8:0] a;
wire z03, z47;
fmap_xor4 x03 /* synthesis xc_uset=”SET1"
 xc_rloc=”R0C0.f”*/
 (z03, a[0], a[1], a[2], a[3]);
fmap_xor4 x47 /* synthesis xc_uset=”SET1"
 xc_rloc=”R0C0.g” */
 (z47, a[4], a[5], a[6], a[7]);
hmap_xor3 zz /* synthesis xc_uset=”SET1"
 xc_rloc=”R0C0.h” */
 (z, z03, z47, a[8]);
endmodule
```

Virtex family:

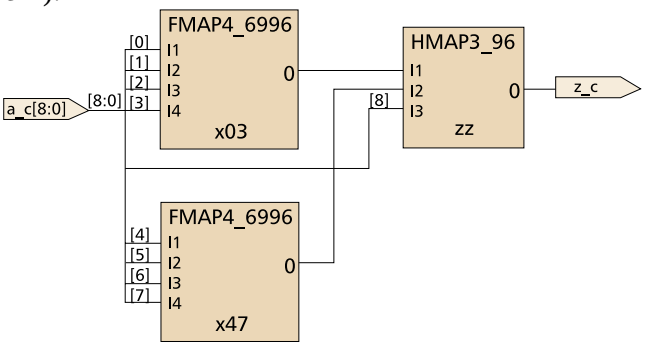
```
module clb_xor9(z, a);
output z;
input [8:0] a;
wire z03, z47;
fmap_xor4 x03 /* synthesis xc_uset=”SET1"
 xc_rloc=”R0C0.S0"*/
 (z03, a[0], a[1], a[2], a[3]);
fmap_xor4 x47 /* synthesis xc_uset=”SET1"
 xc_rloc=”R0C0.S0" */
 (z47, a[4], a[5], a[6], a[7]);
hmap_xor3 zz /* synthesis xc_uset=”SET1"
 xc_rloc=”R0C0.S1" */
 (z, z03, z47, a[8]);
endmodule
```

VHDL:

XC4000, Spartan family:

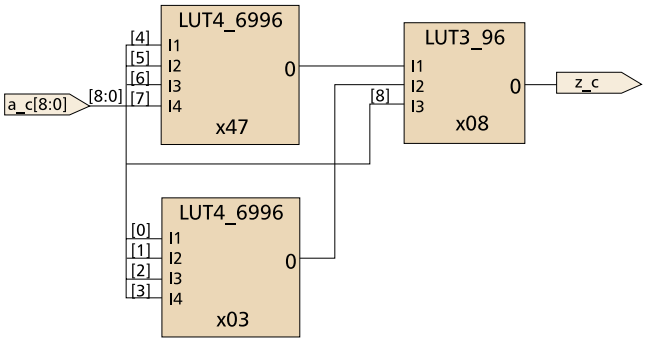
```
library IEEE;
use IEEE.std_logic_1164.all;
entity clb_xor9 is
 port (a : in std_logic_vector(8 downto 0);
       z : out std_logic
   );
end clb_xor9;

architecture rtl of clb_xor9 is
signal z03, z47 : std_logic;
component hmap_xor3
 port (a : in std_logic;
       b : in std_logic;
       c : in std_logic;
       z : out std_logic
   );
end component;

component fmap_xor4
 port (a : in std_logic;
       b : in std_logic;
       c : in std_logic;
       d : in std_logic;
       z : out std_logic
   );
end component;
attribute xc_uset : string;
attribute xc_rloc : string;
attribute xc_uset of x03 : label is “SET1”;
attribute xc_rloc of x03 : label is “R0C0.f”; –(1)
attribute xc_uset of x47 : label is “SET1”;
attribute xc_rloc of x47 : label is “R0C0.g”; –(2)
attribute xc_uset of zz : label is “SET1”;
attribute xc_rloc of zz : label is “R0C0.h”; –(3)
begin
 x03 : fmap_xor4 port map(a(0), a(1), a(2), a(3), z03);
 x47 : fmap_xor4 port map(a(4), a(5), a(6), a(7), z47);
 zz : hmap_xor3 port map(z03, z47, a(8), z);
end rtl;
```

This means all instances that have USET=SET1 should have relative locations R0C0 (meaning they should be in the same CLB).



Virtex family: Replace the **xc_rloc** value to include the slice also:

```
attribute xc_rloc of x03 : label is “R0C0.S0”; –(1)
attribute xc_rloc of x47 : label is “R0C0.S0”; –(2)
attribute xc_rloc of zz : label is “R0C0.S1”; –(3)
```



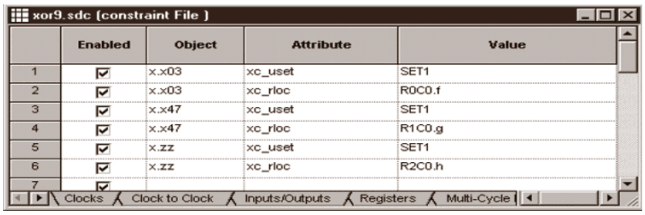
Make a top level design, instantiating the CLB:

Verilog:

```
module xor9top(z, a);
output z;
input [8:0] a;

clb_xor9 x(z, a);
endmodule
```

VHDL:

```
library IEEE;
use IEEE.std_logic_1164.all;

entity xor9top is
 port (z : out std_logic;
       a : in std_logic_vector(8 downto 0)
    );
end xor9top;

architecture rtl of xor9top is
component clb_xor9
 port (z : out std_logic;
       a : in std_logic_vector(8 downto 0)
    );
end component;
begin

x: clb_xor9 port map ( z, a );
end rtl;
```



The RLOC attributes can also be applied through SCOPE: *The constraints are then passed to the Xilinx place and route tool through the EDIF or XNF netlist.*

Conclusion

This article shows how RLOCs can be embedded in RTL code to control the implementation and placement of logic elements in CLBs. RLOCs are a powerful tool for Synplify designers who need additional control to achieve timing requirements.

For more information, please refer to the Synplicity website at http://www.synplicity.com for the full application note. Σ

This article shows how RLOCs can be embedded in RTL code to control the implementation and placement of logic elements in CLBs. RLOCs are a powerful tool for Synplify designers who need additional control to achieve timing requirements.

Inferring Virtex Block RAM with Leonardo Spectrum

Leonardo Spectrum, from Exemplar Logic Inc. helps you implement RAM in Virtex FPGAs.

by Tom Hill, Silicon Vendor Relations Manager, Exemplar Logic, Inc., tom.hill@exemplar.com

Two methods for implementing Virtex RAM are supported by Leonardo Spectrum:

- **Asynchronous RAM** is implemented from the memory elements contained within the lookup tables (LUTs). Leonardo Spectrum automatically detects asynchronous single-port, dual-port, and dual-bus RAM, from RTL behavioral models, and optimizes your implementation using LUT RAM.
- **Synchronous RAM** is implemented using Virtex Block RAM resources. All synchronous single-port, dual-port, and dual-bus RAM is now implemented using the Block RAM resource. Support for synchronous RAM is new, now available in Leonardo Spectrum 99.1, released in March, 1999.

Coding Examples

Synchronous, Single-port RAM Example

VHDL

```
architecture rtl of ram is

type mem_type is array (2**address_width downto 0)
 ofUNSIGNED(data_width – 1 downto 0) ;
signal mem : mem_type ;
 begin
 I0 : process (we,clk,address,mem,data_in)
 begin
 if (clk'event and clk = '1') then
 if (we = '1') then
 mem(conv_integer(address)) <= data_in ;
 end if ;
 end if;
 data_out <= mem(conv_integer(address)) ;
 end process ;
end RTL ;
```

VERILOG

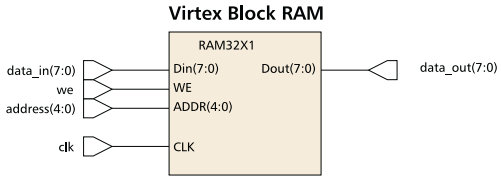
```
 module ram (data_in, address, we, data_out);
 parameter data_width=8, address_width=8,
 mem_elements=256;

 input [data_width-1:0] data_in;
 input [address_width-1:0] address;
 input we;
 output [data_width-1:0] data_out;

 reg [data_width-1:0] mem [mem_elements:0];

 always @(posedge clk) begin
 if (we) mem[address] = data_in;
 end
 assign data_out = mem[address];
endmodule
```

Table 1 – RAM Implementation Resource

	Single Port *Shared Data Port*	Dual Port	Single Port, *Separate Data Ports*	ROM
Synchronous	**Block RAM**	**Block RAM**	**Block RAM**	-
Asynchronous	**LUT RAM**	**LUT RAM**	**LUT RAM**	**LUT RAM**

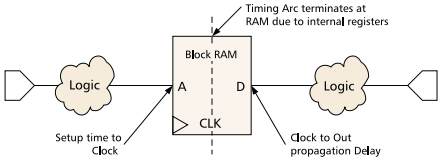
Resulting Circuit

There is no limit to the size of the RAM that can be inferred. Leonardo Spectrum will build up a RAM array out of available elements for a particular technology. In the above example two 32x4 RAMs are required.



Timing Analysis and Optimization

The Virtex Block RAM is a synchronous RAM; there are no asynchronous paths through the RAM. Leonardo Spectrum provides timing models for Block RAM which will generate input setup time and clock-to-out propagation delays similar to the way registers are modeled.



Device Resources

The current version of Leonardo Spectrum does not keep track of Block RAM resources and may implement more Block RAM than is available in the targeted device. If this happens you will need to disable block RAM inference on some blocks by setting the following variable:

```
> set extract_ram FALSE
```

Conclusion

Block RAM can dramatically improve the utilization of a Virtex device. By having dedicated resources, memory can be implemented in a Virtex device without sacrificing lookup tables that are used for random logic. Leonardo Spectrum's inference capability makes it easy to design in Block and LUT RAMs.

For more information about Exemplar Logic, Leonardo Spectrum and Block RAM inference, visit our website at www.exemplar.com or contact Exemplar at 510-789-3333. Σ

Third-Party Developers Deliver First Cores for Virtex FPGAs

by Mike Seither, Director of Public Relations, Xilinx, mike.seither@xilinx.com

New Virtex system-level architecture yields an immediate boost in performance.

Xilinx recently announced the availability of the first wave of third-party software cores that support the new Virtex FPGA family. The cores are available today from partners in the Xilinx AllianceCORE® program and include predefined networking, communications, encryption, and microprocessor peripheral functions.

"The Virtex family is enjoying tremendous market acceptance," said Dennis Segers, vice president of FPGA development and general manager of the High End FPGA Business Unit at Xilinx. "For the first time, customers have access to FPGAs with system-level features and up to one million system gates. This is fueling the demand for cores, which play a critical part in the success of large designs. More of our AllianceCORE partners are recognizing this opportunity, and they are choosing the Virtex family as the primary platform for developing new intellectual property."

AllianceCORE partners report that cores previously designed for Xilinx XC4000 FPGA devices operate 25 to 30 percent faster when converted to run on Virtex devices. All of the new Virtex FPGA cores have been certified to ensure that they work within the Xilinx software design flow.

"The results we are seeing on Virtex FPGAs are nothing short of extraordinary," said James Doherty, managing director of Integrated Silicon Systems (ISS). "Our first run of the ISS-designed DVB Reed-Solomon cores in Virtex devices resulted in an immediate performance improvement of about 28 percent. We expect to see even greater gains in performance by better leveraging some of the system-level features available in Virtex FPGAs."

"The Virtex family features a revolutionary FPGA architecture that is unmatched in the industry," said Timothy Smith, managing director of Memec Design Services of Mesa, Arizona (www.memecdesign.com). "It allows us to consider the development of FPGA cores that were not possible before."

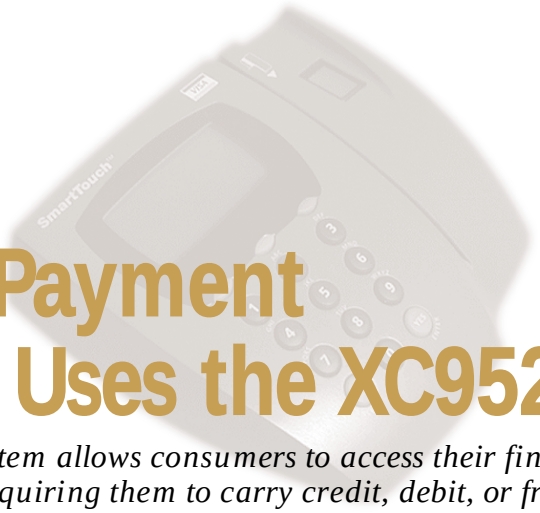
The new cores for Virtex FPGAs available now from AllianceCORE partners include:

- **CAST, Inc**. of Pomona, New York, (www.cast-inc.com) is supplying the C2910A micro-program controller, the C2901 bit-slice unit, the C_UART compact universal asynchronous receiver-transmitter (UART), and the C8259A programmable interrupt controller. New from CAST is the X-DES cryptoprocessor core.
- **CoreEl MicroSystems** of Fremont, California, (www.coreel.com) is supplying a new UTOPIA slave interface and a 10/100 Ethernet media access controller (MAC) core.
- **Integrated Silicon Systems** of Belfast, Northern Ireland, (www.iss-dsp.com) is supplying Reed Solomon encoder and decoder cores.
- **Memec Design Services** is supplying their Reed Solomon encoder core, a new IIC master and slave interface, and Reed Solomon decoder cores.
- **Virtual IP Group** of Santa Clara, California, (www.virtualipgroup.com) is supplying the M8254 timer/counter and M8255 programmable peripheral interface cores.

Virtex Architecture Improves Core Performance

The Virtex family was developed to address system-level design on an FPGA. As a result, the combination of the device architecture and development software greatly simplify the process of designing and using cores. Third-party core providers experience improved performance with less need to optimize for the FPGA architecture. "The Virtex architecture is a natural for implementing IP," said Newton Abdalla, president of CAST, Inc. "The capabilities it provides to support system-on-a-FPGA design are beyond anything that is available in the market today."

All cores are available for purchase today directly from the AllianceCORE partners in either Xilinx optimized netlist or source code formats. Datasheets for each core can be found on the Xilinx website at www.xilinx.com. Σ



Cardless Biometric Payment System Uses the XC95216

The SmartTouch system allows consumers to access their financial accounts without requiring them to carry credit, debit, or frequent shopper cards to prove who they are.

by Phil Lapsley, Vice President of Engineering, SmartTouch, plapsley@smarttouch.com

Imagine being able to buy goods and services at a retail point of sale without needing to carry your wallet, your cash, your credit cards, or your debit cards; in fact, without needing to carry anything at all. Berkeley, California-based start-up SmartTouch, Inc. is making that vision a reality with its patented Cardless Biometric Payment System.

The SmartTouch processing network uses a finger image and a numeric code to identify the consumer, and then translates this identity into financial account information. "It's the ultimate in convenience, security, and freedom," says Phil Lapsley, VP of Engineering at SmartTouch.

A consumer interacts with the SmartTouch system through the SmartPad point-of-sale terminal. The SmartPad is responsible for collecting a consumer's finger image and numeric code, for transmitting this information into the SmartTouch network, and for displaying results back to the consumer.

The SmartPad consists of a 133-MHz 486 CPU with DRAM and flash ROM memory, a finger-image scanner, a keypad, an LCD display, a magnetic-stripe card reader, and a number of communication ports. Connecting it all together is a Xilinx XC95216 CPLD that interfaces the CPU to the finger-imaging chip, the LCD, the keypad, the magnetic stripe reader, and the communication ports.

According to Lapsley, "Using a Xilinx 9500-series CPLD was one of the best decisions we made. The in-system programmability allowed us to quickly iterate towards the exact design that our customers needed, without wasting time burning serial EEPROMS. And now that we're in the field, the SmartPad can reprogram its CPLD under software control to deal with field upgrades."

"Moreover," says Lapsley, "the 95216 has allowed us to change things on the fly to support new requirements. For example, we can interface with several different fingerprint imaging chips. And thanks to the XC9500 pin locking capabilities, we've been able to do all this without having to re-layout our PCBs."

Lapsley also cited technical support as one of the reasons why he has been so pleased with Xilinx. "Our FAE, Bruce Shem, has been wonderful. He gets back to us promptly when we have questions and he makes sure we have the most up-to-date software. He actually came out and installed the software on our workstations when we first started working with Xilinx parts. "

So, when can you expect to see SmartTouch in a store near you? "Soon," says Lapsley. "We're currently in two field trials–one at a chain of high-quality fast-food restaurants and another at one of the major credit card companies. People are excited about the technology. We're lining up our next pilot sites right now, and our first major rollouts should follow soon after." Σ

"... the 95216 has allowed us to change things on the fly to support new requirements. For example, we can interface with several different fingerprint imaging chips. And thanks to the XC9500 pin locking capabilities, we've been able to do all this without having to re-layout our PCBs."

High Performance 8051 Core for Virtex FPGAs

The efficient Virtex architecture allows over 230 MHz equivalent operation.

by Mike Seither, Director of Public Relations, Xilinx, mike.seither@xilinx.com

Xilinx and Dolphin Integration SA of Meylan, France, a Xilinx AllianceCORE partner, recently announced, the immediate availability of the Flip-8051 core. The Flip-8051 is the industry's fastest implementation of the popular 8051 microcontroller optimized for the Virtex FPGA family. The core is aimed directly at embedded markets including automotive, industrial, medical equipment, and consumer products.

"The 8051 microprocessor has been a standard in the embedded industry for years," said Jean-François Pollet, manager of the Flip line of virtual components at Dolphin. "Since most legacy standard 8051 chips are obsolete, a programmable logic version allows companies to quickly reduce the cost or enhance the performance of their systems without rewriting legacy microcontroller code. Our solution, combined with the system-level capabilities of the Virtex FPGAs, paves the way for a rich family of microcontroller configurations for programmable logic-based system-on-chip solutions."

Customers such as Plessis Electronics Ltd., Leighton Buzzard, U.K., a manufacturer of electronic musical instruments, are already taking advantage of this system-on-chip capability.

"We found the transition to the FPGA-based Flip-8051 core to be straightforward," said Serge Plessis of Plessis Electronics. "We estimate that we saved months using this core in the FPGA. The efficient architecture of the core provided us with much more performance than we needed, making design of the overall system very easy."

High Performance Core Architecture

The Flip-8051 core was optimized to execute multiple instructions in a single machine cycle. As a result, the average instruction is executed eight times faster than the legacy 8051 architecture. The current Virtex version of the core operates at a clock frequency of 29.8 MHz, yielding an instruction execution performance equivalent to a legacy 8051 processor operating at 238 MHz. This capability makes the core an ideal solution for low-power applications such as portable systems, sensors and smart-cards because legacy 8051 performance can be achieved at one-eighth the clock frequency.

Pricing and Availability

The Flip-8051 core forms the heart of a family of processors that include lower performance options as well as microcontroller configurations that include peripherals such as timers and serial interfaces. Pricing starts as low as $10,000 for an EDIF format for Virtex FPGAs. Other design file formats are available. A VHDL source-code version is available with a test bench at extra cost.

In addition, Dolphin offers an FPGA-based board that allows you to evaluate a hardware version of the core immediately. Dolphin also offers design services, annual maintenance agreements, and technical support via fax and email. Contact Dolphin Integration directly for more information.

About Dolphin Integration

Dolphin Integration is an IP provider that offers an array of design services and support that range from turnkey design to consulting for customers who want to efficiently integrate Flip cores into their system-on-chip designs. Dolphin's corporate charter is to help customers meet the challenges of time-to-market with a system-on-board or a system-on-chip design. Additional information is available on the Web at www.dolphin.fr.

Virtex Family Provides Gigabit Capabilities for 32-Bit Fault Tolerant Ethernet Switch

by Mike Seither, Director or Public Relations, Xilinx, mike.seither@xilinx.com

The million-gate Virtex XCV1000 device is used to build a two-port Gigabit switch.

The new Virtex FPGAs are enabling Performance Technologies, a leading supplier of high availability networking switching solutions, to add Gigabit functionality to their Nebula® family of high performance, fault tolerant Ethernet switching products. PTI's Ethernet switch family includes the industry's first truly fault tolerant backbone switch, the 32-port Nebula 8000 switch, which uses unique mirroring techniques and a sophisticated failover architecture. Other models in the family include the Nebula 4000 16-port Workgroup Switch, and the Nebula 6000 64-port High Density Departmental Switch. The Nebula family interfaces to Pulsar®, PTI's proven ASIC-based switching fabric.

PTI's Nebula 8000 Fault Tolerant Switch is particularly unique in its class and is ideally suited for mission critical "around-the-clock" application environments such as banking, brokerage, medical, government, transaction processing, or other business and mission-critical applications.

All switch models include features such as hot swappable load-sharing power supplies, port trunking, embedded HTTP server for network management, sophisticated VLANs, Quality of Service support, advanced security/filtering, high-speed WAN access, industry standard WAN and LAN interfaces, and the standard full-duplex Gigabit uplink ports using the Virtex XCV1000 FPGA.

The Virtex family comprises the industry's first FPGAs that address system-level design issues. The Virtex devices offer a robust feature set with densities ranging from 50,000 to one million system gates. The million-gate Virtex FPGAs — an industry first at this density — and other Virtex family members are available now.

"We did an exhaustive search and found that only the Xilinx Virtex FPGAs could provide us with the performance and density necessary to add Gigabit capability to the Nebula switch family," said John Peters, vice president of development at Performance Technologies. "We were very impressed with the system-level capabilities of the Virtex FPGAs, particularly the digital delay locked loops and support for multiple I/O standards."

About Performance Technologies

Performance Technologies is a world class supplier of leading-edge, high-availability network switching and data communication solutions. The company's products include a wide range of fault tolerant Fast Ethernet switching products for business and mission-critical network environments, intelligent LAN/WAN communications controllers for high performance workstations and servers, and network interface cards for mass storage devices.

Performance Technologies targets the financial services, telecommunications, information processing, and defense industries. PTI is a publicly traded company (NASDAQ:PTIX). Visit the PTI website at www.pt.com or switching website at www.gigabit.com.

Nebula and Pulsar are trademarks of Performance Technologies, Inc.

Using SpartanXL for Low-cost Video Applications

A low-cost SpartanXL solution for driving a TFT LCD Panel with a graphical overlay.

by Edgard Garcia, Multi Video Designs, Consultant, edgard.garcia@mvd-fpga.com

Here at Multi Video Designs, we recently completed a project for a prestigious French military customer. The design included a TFT LCD Panel using a video signal multiplexed with graphical data. The data was received from an external source through an RS232 interface, and then stored on-board.

The three key requirements of this application were:

- The interface must run at video data rates without any break in the output stream.
- Multiplexing between video and graphical data in one clock period.
- The solution must be as inexpensive as possible.

We decided to use a Xilinx FPGA, because we needed the ability to work with a pixel clock of 30 MHz. It also allowed us to use the pre-defined Dual-Port RAM and Multiplexer functions available in the Xilinx Logiblox generator. To keep costs as low as possible, we chose the XCS20XL-PQ208-4 from the Spartan family. The advantage of this solution is that we have a lot of possibilities, with very little hardware on board.

In this application, the FPGA is not configured by a PROM or an EEPROM, but by a microcontroller. The configuration file is stored in an external EEPROM, which is written to by the host via the RS232 line. This configuration gave us the ability to perform a lot of iterations of the configuration file without any hardware manipulation.

The EEPROM is also used to store the font characters. The user can ask for a character to be transferred to any position in the graphical RAM for display on screen. The user can also modify the font or generate graphical information via the RS232 line.

The internal resources of the FPGA allowed us to easily create multiplexers and FIFOs (32x16 bits and 32x8bits) with no implementation or timing problems. In addition, with a little hardware modification, we could easily drive more than one TFT screen.

Conclusion

The internal resources of Xilinx FPGAs are perfect for implementing low cost video applications. The on-chip FIFOs were critical components for developing this real-time video application, helping us to easily achieve our 30MHz performance requirements.

For more information about Multi Video Designs contact: Edgard Garcia
Tel : (33) 5 62 13 52 32
Fax : (33) 5 61 06 72 60
E-mail : edgard.garcia@mvd-fpga.com
or info@mvd-fpga.com

To keep costs as low as possible, we chose the XCS20XL from the Spartan family. The advantage of this solution is that we have a lot of possibilities, with very little hardware on board.

WILDSTAR™ High-Speed DSP Board Performing 11.8 Gflops/Board

Annapolis Micro Systems recently announced the release of WILDSTAR, a new family of Commercial Off the Shelf (COTS) High Speed Digital Signal Processing boards, available in VME64X, Compact PCI, and PCI bus, capable of performing 11.8 GFLOPS/board (20-bit floating-point FFT) and 23.7 GOPS/board (16-bit fixed-point MACs).

by Jane S. Donaldson, Annapolis Micro Systems, jdonald@annapmicro.com

WILDSTAR uses three Virtex™ FPGAs as processors, combining the power and speed of a dedicated DSP processor, with the reprogrammability of FPGAs. When an application is downloaded into the FPGAs, the board becomes a customized parallel processing system with processors crafted for that particular application. In addition, because the processing elements are FPGAs, the system as a whole can be modified while the application is running. This allows the board to perform many different customized algorithms at hardware speeds.

Furthermore the WILDSTAR architecture is completely compatible with the Xilinx Internet Reconfigurable Logic (IRL) tools so that it can be easily updated over any network after it has been deployed in the field.

Specifications

The WILDSTAR family was designed to solve the most demanding DSP problems. WILDSTAR is delivered with a Multi-Radix FFT IP core binary file, which can compute a single 1K, 32-bit floating point FFT in approximately 25 microseconds, using only one of the three Virtex XCV1000 FPGAs on the board. Continuous processing can be streamed together, to achieve a time of approximately 20.5 microseconds to compute the same 1K 32-bit floating point FFT at up to 49 megasamples/second, still in one FPGA. If the continuous processing is streamed through two FPGAs, the average time to compute the 1K 32-bit floating point FFT becomes 10.3 microseconds, at up to 99 megasamples/second.

WILDSTAR has an I/O bandwidth of up to 1.6 gigabytes/second into the board, through optional I/O daughter cards. The 6U-sized VME64X and CompactPCI boards can take up to two I/O cards, including support for Myrinet™, Raceway™, and high speed matched impedance (MICTOR™) connectors.

The PCI WILDSTAR board will accept new PCI WILDSTAR I/O daughter cards, as well as all the I/O cards from the WILDFIRE™ Family. Annapolis Micro Systems supplies a variety of I/O cards for PCI, including analog and digital camera, RS422, E1, E3, T1, and a new card called the Virtex™ I/O card, which has a single Virtex™ XV300 part and adds 172 lines of high speed digital I/O.

Price and Availability

STARFIRE™ is a smaller version of the PCI WILDSTAR, with only one Virtex XCV1000 chip. WILDCARD™ is a Cardbus™ card, initially available with a Virtex XCV300 device. Annapolis Micro Systems, Inc. is currently taking orders for the WILDSTAR™, STARFIRE™, Virtex™ I/O, and WILDCARD™ boards. Initial pricing for the Virtex-based boards starts at $4,500 for the Virtex I/O card and approaches $50K for the larger WILDSTAR boards. Pricing for the XC4000XLA-based boards starts at $1,750.

For more information and pricing call Jane S. Donaldson or Bill Hulbert at (410)841-2514. You can reach us through email at wfinfo@annapmicro.com or visit our website at http://www.annapmicro.com. Σ

Xilinx Ships The

Real 64/66 PCI™

Industry's First General Purpose 64-bit, 66 MHz PCI Solution

by Per Holmberg, LogiCORE Product Manager, Xilinx, per.holmberg@xilinx.com

The Virtex FPGA family, along with our new PCI cores, meets the demand for uncompromising PCI compliance, flexibility, and performance.

In a move that brings programmable logic to the forefront of high-performance system level integration, Xilinx recently announced the immediate availability of The Real 64/66 PCI™ solution. The Real 64/66 PCI core is the first complete solution that enables you to design fully compliant yet flexible single-chip 64-bit, 66 MHz PCI v.2.2 bus interface systems.

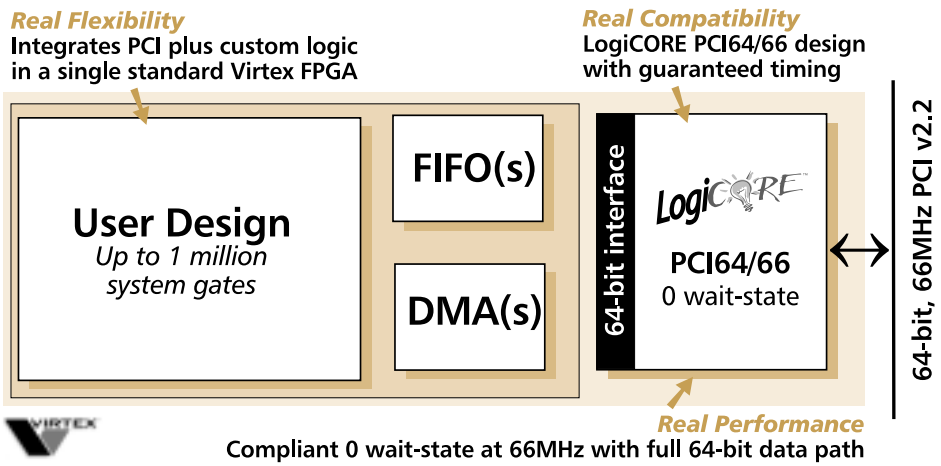
The Real 64/66 PCI solution from Xilinx:

- Is available as a commercial product today.
- Offers full compliance with the v2.2 PCI bus interface specification.
- Provides 64-bit, 66MHz performance throughout the complete design.
- Gives you the flexibility to build a single-chip design using standard FPGAs.

"With the Real-PCI 64/66 products from Xilinx, we were able to implement a fully compliant PCI interface in our new Mx2/PCI product family plus other functions such as direct memory access (DMA), four dual-port FIFOs, and 200,000 gates of our own unique design in a single device," said John Beck, principal engineer at DOME imaging systems, Inc., Waltham, Mass. The Dome Mx2/PCI is the first in a new family of high resolution display controllers for the medical imaging market that can handle transfers of over 500 MBytes/second from the host.

"After evaluating different solutions in the market, we found that only Xilinx could meet the demanding requirements for full 66 MHz PCI compliance," Beck said.

"The Real 64/66 PCI represents the first time that an FPGA supplier has delivered a general purpose solution before manufacturers of standard chip-sets," said Wim Roelandts, Xilinx president and CEO. "This is a significant milestone that underscores the inherent benefits of standard FPGAs produced with the most advanced silicon processes. More important, The Real 64/66 PCI solution allows designers to integrate very high-performance, high-density 66MHz PCI systems tailored to their specific needs."



Real Availability

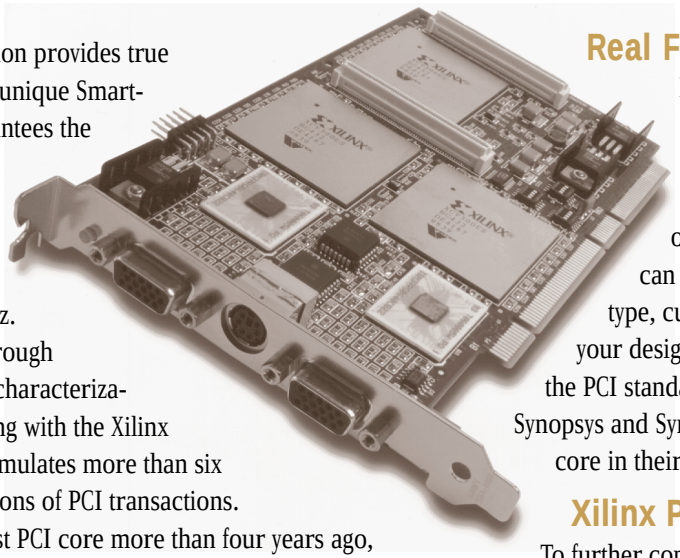
More than a dozen Xilinx customers have been actively engaged in a beta program since September using The Real 64/66 PCI core to design high-performance applications such as Gigabit Ethernet, ATM and Fiber Channel adapters, DSP and imaging boards, disk drive arrays, and high-end printer interfaces. Beta customers include Cisco Systems, Ascend Communications, Dome imaging systems, and Kodak.

Continued on the following page

MEMBER
PCI LOCAL BUS
SPECIAL INTEREST GROUP

This rich PCI heritage has given Xilinx experience with PCI that is unmatched in the industry. It has allowed us to develop the design and verification processes necessary to build and support high-quality PCI products.

The Dome Mx2/PCI from Dome imaging systems.



Real Compliance

The Real 64/66 PCI solution provides true compliance by using our unique SmartIP® technology that guarantees the critical minimum, maximum, and hold timing required for a true zero wait-state burst operation at 66 MHz. Compliance is verified through hardware testing, device characterization, and regression testing with the Xilinx internal test bench that simulates more than six million unique combinations of PCI transactions.

"Xilinx released its first PCI core more than four years ago, and our PCI products to date have been used in more than 1,000 customer designs," said Rich Sevcik, senior vice president of software, cores and support at Xilinx. "This rich PCI heritage has given Xilinx experience with PCI that is unmatched in the industry. It has allowed us to develop the design and verification processes necessary to build and support high-quality PCI products. The Real 64/66 PCI core, which can be downloaded from the Xilinx website, reinforces our on-going Silicon Xpresso initiative to use the Internet to increase designers' productivity."

Real Performance

The Virtex FPGAs are manufactured on a state-of-the-art 0.22µ process that meets all timing requirements for 64-bit, 66-MHz performance, up to the theoretical maximum throughput of 528 Mbytes per second.

The Real 64/66 PCI solution is fully verified by Xilinx for the Virtex XCV300-6 BG432 and XCV1000-6 FG680 devices, which offer densities of 300,000 and one million system gates, respectively.

Real Flexibility

Implemented in Xilinx standard Virtex FPGAs, The Real 64/66 PCI solution allows you to benefit from the real flexibility provided only by standard, off-the-shelf FPGAs. For example, you can choose Virtex device size and package type, customize the PCI feature set, and adapt your design, as needed, to meet future changes in the PCI standard or new PCI requirements. Both Synopsys and Synplicity support The Real 64/66 PCI core in their design flows.

Xilinx PCI Training

To further complete the Xilinx PCI solution, Xilinx will offer a two-day PCI course, beginning in May, for customers who are planning PCI systems. The course will give an introduction to the PCI standard, cover configuration and integration of Xilinx PCI cores, system integration, verification and debugging. In addition to Xilinx classes, PCI design services are available from a number of partners in the worldwide Xilinx XPERTS design consulting program.

Real-PCI Design Kits

Xilinx is developing a complete PCI64 Design Kit (just like the one currently available for our 32-bit PCI products). The kit will include Real-PCI designs, reference designs, a prototyping board, and software driver development tools. The PCI64 Design Kit will be available by the Summer of 1999.

Pricing

The Real 64/66 PCI core for the Virtex XCV300 BG432 device is available now from Xilinx. The Real 64/66 PCI product is priced at $14,995. To order The Real 64/66 PCI see www.xilinx.com/pci. Σ

Xilinx-based Virtual Reality

One Billion Operations Per Second!

What do virtual reality, your kid's most recent PC game, and professional flight simulators have in common? The answer is real time 3D-graphics acceleration, a function provided by specialized circuits called rasterizers.

by Luis del Pino, General Manager, Memondo Graphics S.L., info@memondo.es

There are several specialized rasterizing ASICs in the market. Custom made circuits like these however, have an important limitation: you must use them "as is," making them unsuitable for certain low- to mid-volume applications.

We wanted to develop a high-performance, state of the art rasterizer which could also be easily (and cost-effectively) adapted to different application scenarios. The requirement for adaptability lead us to FPGA technology, but the big question was: could an FPGA deliver the required performance? The answer, as it turned out, was that Xilinx FPGAs can.

The architecture of the Xilinx FPGAs is particularly well adapted to 3D graphics applications, because of three main reasons:

- The distributed memory concept, which allows you to overcome the most common bottlenecks in 3-D graphics systems. You can tightly couple the processing and storing elements, instead of having the data passing through a common bus.
- 3D-graphics algorithms can be decomposed into a complex sequence of linear operations, which can be easily implemented using the Xilinx carry chains.
- The large number of available logic cells and routing resources allows you to use pipelining and parallelization techniques to increase the speed of the resulting algorithm.

Our Derissa D66 is a 3D-graphics rasterizer using an XC4085XL device. It provides Gouraud shading and perspective-correct texture mapping functionality, 66 MPixel/sec maximum speed, 16-bit z-buffer, and double frame buffering. Applications include VR systems, arcade games, LBE systems, and high-end simulators.

Derissa D66 provides a single chip solution including a PCI-compatible host interface, four built-in memory controllers, eight interpolation units, and four perspective correction modules. The design was realized using the Foundation 1.4 tools. In order to achieve the performance goals, LogiBLOX and Coregen modules were extensively used in the design, as well as custom made RPMs for the critical sections of the circuit. The result is a device capable of performing one billion operations/sec, with 600 MBytes/sec data transfer rate to/from the supporting memories.

An incremental design approach was selected in which each module was simulated, added to the design, and software tested, before adding a new module. To facilitate this approach, the architecture of the rasterizer has been optimized both for the underlying FPGA structure and for the characteristics of the implementation tools. As a result, compile time is less than three hours in a Pentium 300, allowing us to avoid using guided design, and making it possible to change the design in record time.

Conclusion

Xilinx FPGAs deliver the performance required by real time 3D graphics applications, while preserving the flexibility of a programmable device. You don't have to adapt your 3D-graphics needs to a custom circuit — using the Xilinx-based Derissa rasterizer; you can adapt the circuit to your custom needs instead.

For more information, contact Memondo Graphics S.L. (www.memondo.es) Σ

Buh-Buh-Buh-Billion!

Single DIME Module

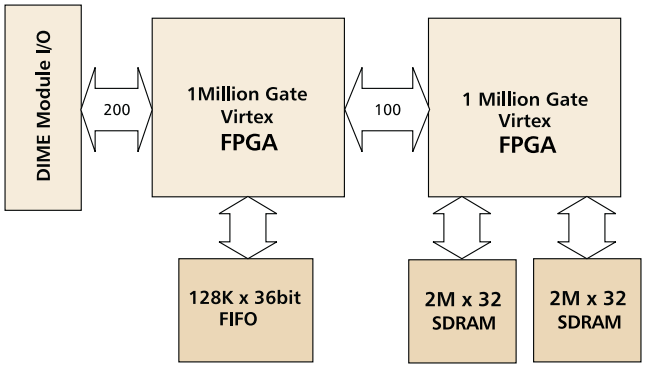
Delivers 2.2 Million Programmable Gates and Growing

Ten years ago, who would of thought that at the turn of the century we could squeeze over 2.2 Million programmable logic gates, 16-Mbytes of memory, and a 4-Mbit FIFO into a few square inches?

by Allan Cantle, Managing & Technical Director, Nallatech Ltd., a.cantle@nallatech.com

It is now possible to implement significant DSP algorithms into a single "DIME" module and have them running faster than you ever imagined. For example, at Nallatech Ltd. we are currently implementing a 13 x 13 Convolution function with 16-bit wide coefficients and data paths on this module, so you don't need to concern yourself with the problems of multiple power supplies and complex BGA packaging. You can instantly utilize two of the largest FPGAs in the world on this one module and all you need are your current FPGA development tools.

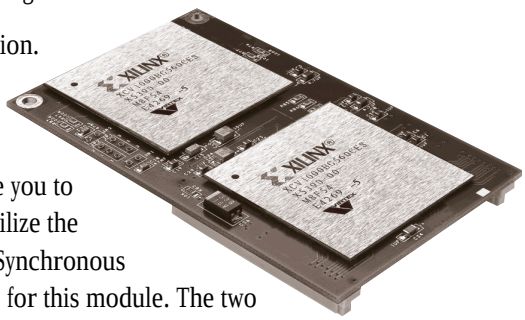
Module Architecture



Nallatech's range of DIME motherboards and modules enable you to rapidly develop custom applications without the need to design and manufacture custom PCB's.

This DIME module has been designed to conquer those complex two dimensional image processing problems that have been the mainstay of the DSP microprocessor technologies and dedicated ASICs to date. Examples of image processing functions that this module can easily perform in real time include:

- 2D Convolution/Correlation.
- Image Pattern Recognition.
- Graphics Generation.
- 2D Morphology.



Example VHDL functions that enable you to quickly and easily utilize the onboard FIFOs and Synchronous DRAMs are available for this module. The two independent SDRAMs allow you to implement a versatile arrangement of data analysis and manipulation techniques.

When used in conjunction with Nallatech's Ballynuey PCI Carrier Card (described in the Q199 issue of Xcell) you can be up and running with this DIME module in minutes. Your FPGA designs are downloaded directly over the PCI bus to the FPGAs through the integrated JTAG boundary scan chain from the supplied configuration software. No PROMs or download cables are required.

Conclusion

Nallatech's range of DIME motherboards and modules enable you to rapidly develop custom applications without the need to design and manufacture custom PCB's. This allows a much quicker time to market as well as a straight forward rapid prototyping platform.

For additional Xilinx information see: www.nallatech.com. Σ

Low-Cost Programmer FOR Xilinx Serial PROMS

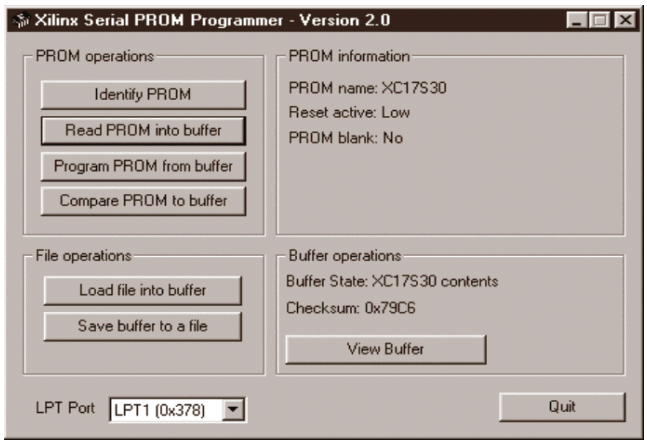
A low-cost programmer for the entire family of Xilinx XC1700 serial PROMS.

by Lance Roman, President, Roman-Jones, Inc., rj@roman-jones.com

The SPROM-MPS programmer from Roman-Jones, Inc. targets the XC1700E/EL(X), XC1701/L, XC17512L, and XC17xx/XL(L) SPROM families. The basic model, shown in Figure 1, accepts 8-pin Dip packages, using a standard machine pin IC socket. A zero insertion force (ZIF) socket is available at a nominal charge. With our optional set of five socket adapters, all package types are accommodated. The compact unit plugs into the parallel printer port of any PC; there are no other cards to install. It operates from a 9V battery, eliminating the need for an AC adapter.

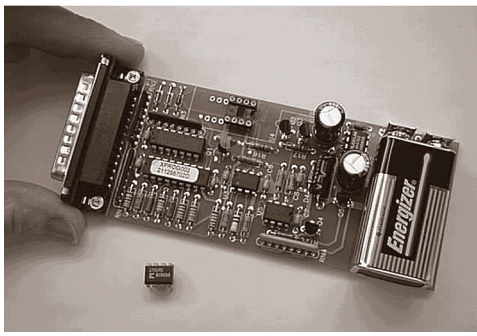
Software

The simple to use application software installs itself from the Web and operates with a graphical user interface under Windows '95, '98, and 'NT. It is downloaded from the Roman-Jones website as an executable (.exe) file. You just double click on *sprom32.exe* from Windows Explorer to run it.



The corporate website is located at www.roman-jones.com.

This single-window application gives you the following capabilities:



- Parallel port selection.
- Automatic identification of inserted PROM.
- Program device.
- Verify program.
- Load PROM contents into buffer.
- Load Hex (*.mcs) file into buffer.
- Save buffer to file.
- View buffer contents.

The software accepts the Intel HEX format files generated by the Xilinx PROM File Formatter Utility.

For the Windows NT environment, an additional device driver must be installed before running the application executable. This self-extracting ZIP file with instructions is available from our website.

For DOS users, there is a non-graphics version of the application, *sprom16.exe*, that operates on anything from an old 8088 machine with DOS 2.1 and 265K RAM, to the current Pentium® III machines.

The SPROM-MPS was developed by Roman-Jones, Inc., a Xilinx-recognized consulting company involved in contract engineering for the electronic and computer industry. Roman-Jones, Inc. has several degreed electrical and computer engineers located in Michigan, doing business nation wide. The SPROM-MPS lists for $99.95 and can be ordered on-line or by phone, 616.326.5194. ⧓

XC9500XV

High-Performance 2.5V ISP CPLD

Manufactured on the latest generation FastFLASH process, the XC9500XV family is the industry's first 2.5V CPLDs, providing power savings of up to 75% over current 5V CPLDs, at a lower cost.

by John Ahn, Xilinx, ahn@xilinx.com

The XC9500XV family provides the same advanced architectural features and densities as our popular 3.3V XC9500XL family introduced last year.

The advanced architecture of both families allows for easy design implementation, so you can concentrate on system design rather than on chip-level details. The unique features of the architecture include a 54-input block fan-in which contributes to superior pin-locking capability; built-in input hysteresis for improved noise margin; bus-hold circuitry for better I/O control; hot-plugging to eliminate the need for power sequencing; and local/global clock control to provide maximum flexibility to each individual macrocell.

High Performance Through Advanced Process Technology

The advanced FastFLASH process used in the XC9500XV allows for high performance versions, with pin-to-pin delays as low as 3.5 nanoseconds and system frequencies as fast 225 MHz by the end of 1999.

The XC9500XV devices are also optimized for in-system programming (ISP) and features the industry's most extensive IEEE 1149.1 JTAG support. Manufacturing, testing, programming, and field-upgrading of CPLD-based electronic products can be easily streamlined through the use of ISP and the Java API for Boundary Scan programming standard.

With the introduction of the XC9500XV family, Xilinx now offers a complete 2.5V programmable logic solution, from the 36-macrocell XC9536XV CPLD to the one-million gate Virtex XCV1000 FPGA.

The XC9500XV Family at a Glance

	XC9536XV	XC9572XV	XC95144XV	XC95288XV
Macrocells	36	72	144	288
Fastest tpd (ns)	3.5/4/5/7	4/5/7/10	4/5/7/10	5/7/10/15
Highest Fsys (MHz)	225	200	200	178
Package Options (User I/O)	PC44 (34) VQ64 (34) CS48 (36)	PC44 (34) VQ64 (52) TQ100 (72) CS48 (36)	 TQ100 (81) TQ144 (117) CS144 (117)	 TQ144 (117) PQ208 (168) BG256 (168) CS280 (168)

With the introduction of the XC9500XV family, Xilinx now offers a complete 2.5V programmable logic solution ...

A 400MHz Frequency Counter

This article describes a full-featured, single-chip frequency counter that operates at 400 MHz, consumes only 130 mW at the maximum input frequency, and occupies less than 90% of an XC4002XL, the smallest XC4000 family member.

by Bernie New, Peter Alfke, Applications Engineering, Xilinx, bernien@xilinx.com, peter@xilinx.com

The conventional block diagram of the frequency counter is shown in Figure 1. The heart of the design is a six-digit decade counter that is driven by a programmable pre-scaler. This pre-scaler is gated by a half-second pulse, and the frequency is determined from the number of input cycles counted in this period.

The time base for the counter is created from a standard 32,768-Hz crystal oscillator. Its output is divided to provide the half-second gating pulse. In a short interval between the gating pulses, the contents of the decade counter are decoded for the 7-segment displays, and the segment states are captured.

The frequency counter has a three-decade auto-ranging capability. At the end of each half-second period, the count value is examined to determine if it is in range. If it is not, the amount of pre-scaling is adjusted for the next half-second period. Hysteresis is built into the auto-ranging circuits to stop any display hunting when the input frequency is at a range boundary.

When the input frequency falls below the auto-range capacity, the display of leading zeros is suppressed. The outputs to the liquid-crystal display are modulated at 128 Hz to provide AC drive directly to the LCD.

Semi-synchronous Design

The design uses a cascade of synchronous 2-bit state machines, with each stage clocking the next asynchronously. This design style was influenced by the XC4000XL architecture . Each CLB provides two flip-flops that share the clock. A fully asynchronous design would waste half of the flip-flops since there is no individual clock access.

Typically, the 2-bit state machine is a modified Johnson counter. The 4-input function generator that precedes each of the flip-flops has three uncommitted inputs that can be used to modify the state sequence.

Detailed Design Description

The first stage of the counter is the most critical. At 400 MHz, it is operating at the maximum possible toggle frequency, and the design must, therefore, be kept as simple as possible.

Frequency Counter Block Diagram

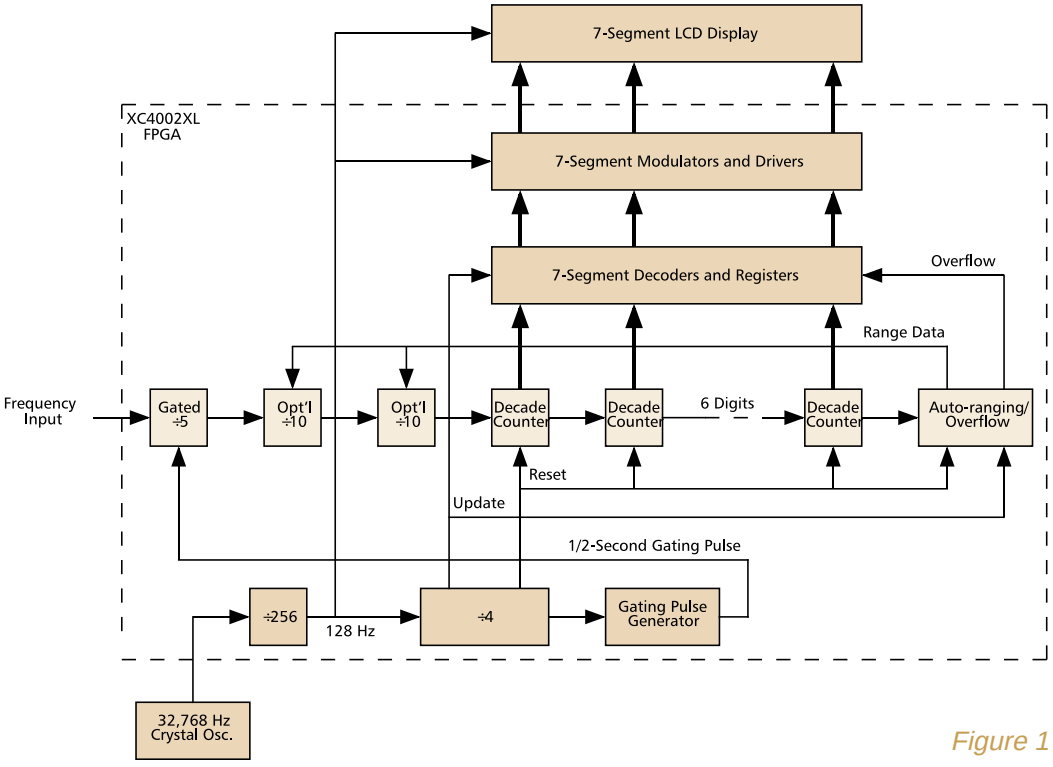


Figure 1

Continued on the following page

The first stage of the counter is the most critical. At 400 MHz, it is operating at the maximum possible toggle frequency, and the design must, therefore, be kept as simple as possible.

400MHz Divide-by-2 Stage

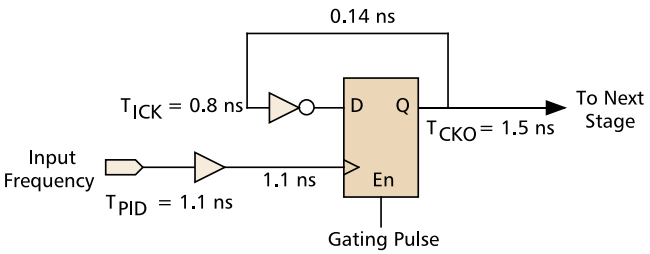


Figure 2

Consequently, the first stage is an unconditional divide-by-2, as shown in Figure 2. The clock-to-setup delay of 2.44 ns permits 400-MHz operation even under worst-case conditions. The flip-flop is located in the leftmost column of CLBs. This location gave the shortest route from the IOB, just 1.1ns.

While speed requirements necessitate simplicity around the first flip-flop, its clock-enable input can be used to effectively gate the input signal. If the first flip-flop is disabled, no subsequent clocks are generated and the whole counter chain is disabled. Thus, the effect is the same as removing the input from the counter.

Fixed Divide-by-5 Stage

The residual pre-scaler in the lowest frequency range is divide-by-5. This is in conflict with having the first stage be an unconditional divide-by-2, since five is an odd number.

The solution to this problem is shown in Figure 3. A divide-by-2/divide-by-3 counter divides with a toggle flip-flop. This flip-flop is then fed back to control the modulus of the counter, alternating it between divide-by-2 and divide-by-3. The result is that the flip-flop toggles at one-fifth of the input clock with a 2:3 mark-space ratio.

In this case, however, the output is taken directly from the counter. When combined with the first stage, this gives a division ratio that alternates between four and six. This averages to divide-by-5, but with a variable mark-space ratio. Over two periods, the mark-space ratio is 2:2:2:4. Two clock edges are produced every ten input clocks, and the division ratio is correct.

The count sequence of the divide-by-2/divide-by-2/3 counter was selected to allow the feedback signal more time to set-up. The control input is “don’t care” except at the second clock edge after the toggle-flip-flop is clocked. Thus there are two clock cycles for the feedback path to settle. With a 400-MHz input, 10 ns is available which is more than adequate.

Divide-by-5 Stage

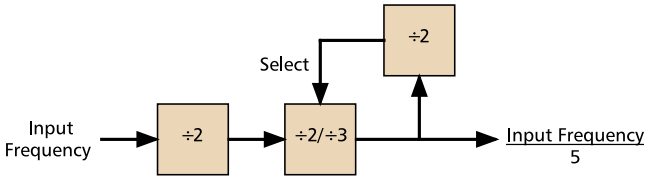


Figure 3

Optional Divide-by-10 Stages

The optional divide-by-10 pre scaler stages are both the same.

The circuit is a simple modification of the divide-by-5 described above. The only difference is that the toggle-flip-flop is replaced by a 2-bit Johnson counter. The division ratio is achieved by controlling the divide-by-2/divide-by- counter to divide successively by 2, 3, 3, and 2 within one cycle of the Johnson counter.

When a stage is not required, a multiplexer selects the input clock instead of the divided output. A simple clock multiplexer is adequate since the select control only changes when the counter is disabled.

Decade Counter

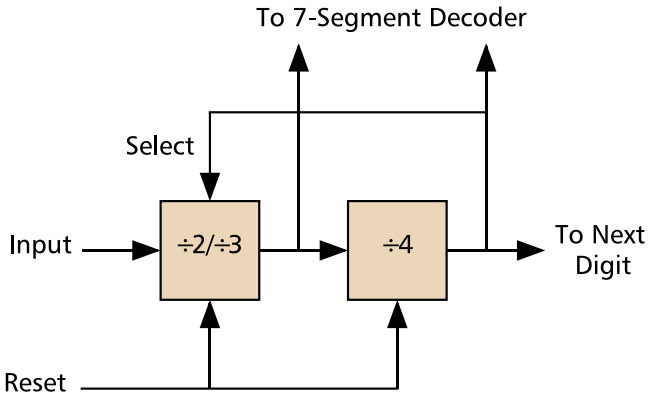


Figure 4

Decade Counter

The decade-counter design, shown in Figure 4, is based on the divide-by-10 pre-scaler. The non-binary sequence is not a problem, because LUTs are used as decoders for the 7-segment displays, and any mapping is possible. The design of the counter also provides leading-zero suppression. As shown in the state diagram in Figure 5, there are two zero states that

have different state assignments. One is a displayed-zero state, while the other is a suppressed-zero state.

When the counter is reset to the suppressed-zero state, all zeros are leading zeros. As counting progresses, however, the counter rolls over to the displayed-zero state. This cannot happen until at least one carry has been generated, and the zero can no longer be a leading zero. Whether a leading zero is actually suppressed or not can be determined in the 7-segment decoder. The suppressed-zero location in the LUT is programmed either for blank or a duplicate zero.

In this frequency counter design, only the three most significant leading zeros are suppressed. The others are displayed along with a kilohertz decimal point to indicate low input frequencies.

Decade Counter State Diagram

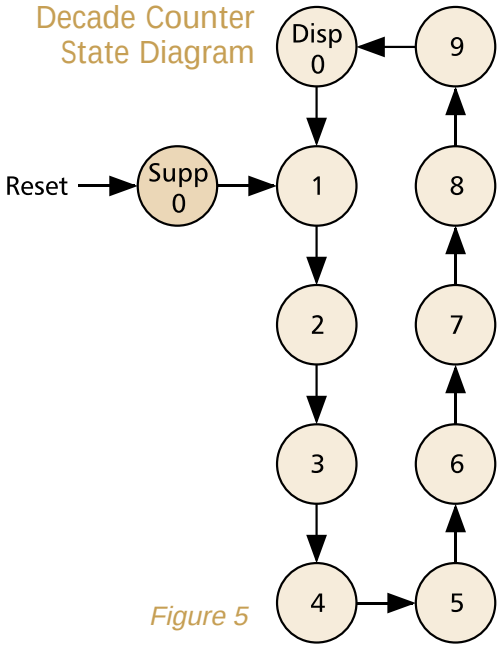


Figure 5

7-segment Decoders and Drivers

The 7-segment decoders are simple LUTs that are connected to the decade counters. During the interval between gating pulses, the LUT outputs are clocked into a register. This register, in turn, drives the display, holding it stable for one counting period.

The segment bits are modulated as they pass through their respective IOBs, thus providing AC drive to the LCD. This modulation uses a 50% duty-cycle128-Hz signal from the time-base and an XOR function that is built into the IOBs. The 128-Hz signal also drives the LCD substrate.

Time-base

The time base is derived from a crystal oscillator module. 32,768 Hz, a standard watch frequency, was chosen because the half-second gate pulse could be obtained by simple binary division.

The first four division stages are a cascade of 2-bit Johnson counters, and the resulting 128 Hz signal modulates the LCD outputs. In addition, the 128 Hz is divided in another 2-bit Johnson counter that is decoded to provide the control signals that update the display register and reset the counter.

The 32-Hz signal from the final Johnson Counter clocks a synchronous counter that outputs a 16:1 mark/space ratio. The mark is the half-second gating pulse, and the space enables the Johnson-counter decoders to create one set of update and reset pulses. Synchronous operation of this final counter was chosen to ensure a precise half-second pulse. At 32 Hz, the clock-distribution power is negligible.

Implementation

The design was entered using schematic capture. To guarantee the performance objectives, six CLBs and one IOB were hand-placed using LOC constraints in the schematic. These hand-placed resources represent less than 10% of the total FPGA. The remaining logic was mapped and placed automatically.

The ripple nature of the design did not permit clock frequencies to be specified in the design. Instead, net-delay constraints were place on a total of 9 nets. These constraints covered the input and pre-scaler stages.

80-MHz operation was required up to the input of first decade counter. This requirement was to guarantee that the counters would still operate correctly if the maximum input frequency is applied when no optional pre-scaling is selected. Otherwise, the auto-ranging circuits would receive invalid information and might fail to operate.

Results

Using a 3.6-V NiCad battery, the counter operates reliably at 420 MHz. As the input frequency varies, the supply current changes from 2 mA with no input to 40 mA at the maximum input frequency. At idle, the current draw is dominated by the time-base crystal oscillator.

The design used 56 of 64 CLBs, less than 90% of the device. Including four test-point signals, all of the available IOBs were used. Using the default settings of the implementation software, the design compiles in just four minutes.

Conclusions

The design is somewhat unconventional in the rate at which its frequency requirements reduce as one moves away from the input. However, it is not that unusual to find small regions of high frequency operation in an otherwise moderate frequency design. The frequency counter demonstrates that, with only a minor amount of manual effort devoted to the high-speed regions, the whole design can easily be implemented in an FPGA.

The semi-synchronous design style proved very effective. One non-obvious advantage is that the data and the clock are combined into a single signal. This greatly simplifies the placement and routing since there are no timing relationships to be maintained between stages. Σ

A Serial Communication Link Between FPGAs

A synchronous point-to-point serial communication link between two FPGAs.

by Paul Gigliotti, Field Applications Engineer, Xilinx, paul.gigliotti@xilinx.com

This article describes a method for transmitting serial data between FPGAs, as shown in Figure 1. The data is transmitted in packets to allow for error detection and re-synchronization of the serial data link. This example design assumes a transfer of 32 bits, but the size can vary as needed.

A Serial Link Between FPGAs

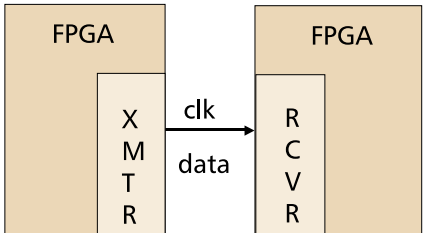


Figure 1

The Start Sequence and Zero-Insertion

The basic trick is to begin the transaction with a start sequence, and ensure that the sequence does not occur within the remainder of the packet. This is easier than it sounds; it's just a matter of selecting the right start sequence. In this example, I transmit a start sequence of 111110 and then insert a zero into the data whenever four ones have occurred, to ensure that the start sequence is not repeated within the data, as shown in Figure 2. The receiver circuit, on the other hand, waits until it sees the start sequence before receiving data, and while receiving data, it throws away the zero that follows four ones. Furthermore, while receiving data after the start sequence has been detected, if five ones in a row are received then an error has occurred.

Zero Insertion

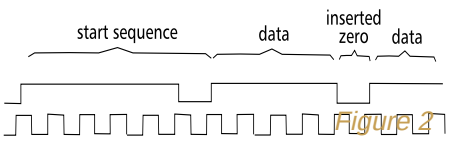


Figure 2

Transmitter State Machine

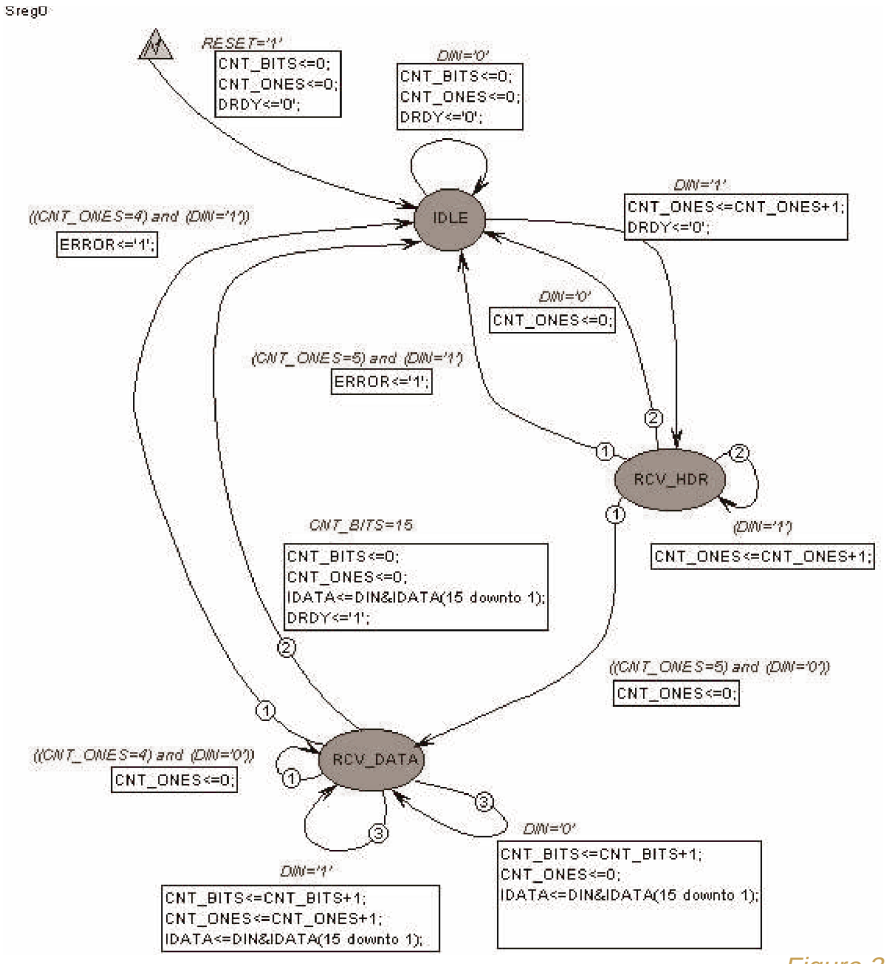


Figure 3

This application offers a simple yet effective method for achieving synchronization between the transmitter and receiver circuits. You can easily extend the design to append parity and CRC bits to the transmitted data.

Transmitter State Machine

Data is transmitted whenever the state machine detects a "Send" request, as shown in Figure 3. The state machine begins by transmitting the header or start sequence and then follows with the data. Notice, that while transmitting the data, it counts the number of "ONES-IN-A-ROW' that it has transmitted, and if the limit has been reached, a zero is inserted and transmitted as part of the data.

Receiver State Machine

The receiver state machine is constantly looking for a start sequence, as shown in Figure 4. Once a start sequence has been received, it then begins receiving data, looking for inserted zeros and flagging errors as they occur.

Simulation Results

Figure 5 shows the results of the transmitter and receiver working in tandem with an FFFF and then a 0000 being sent/received

Receiver State Machine

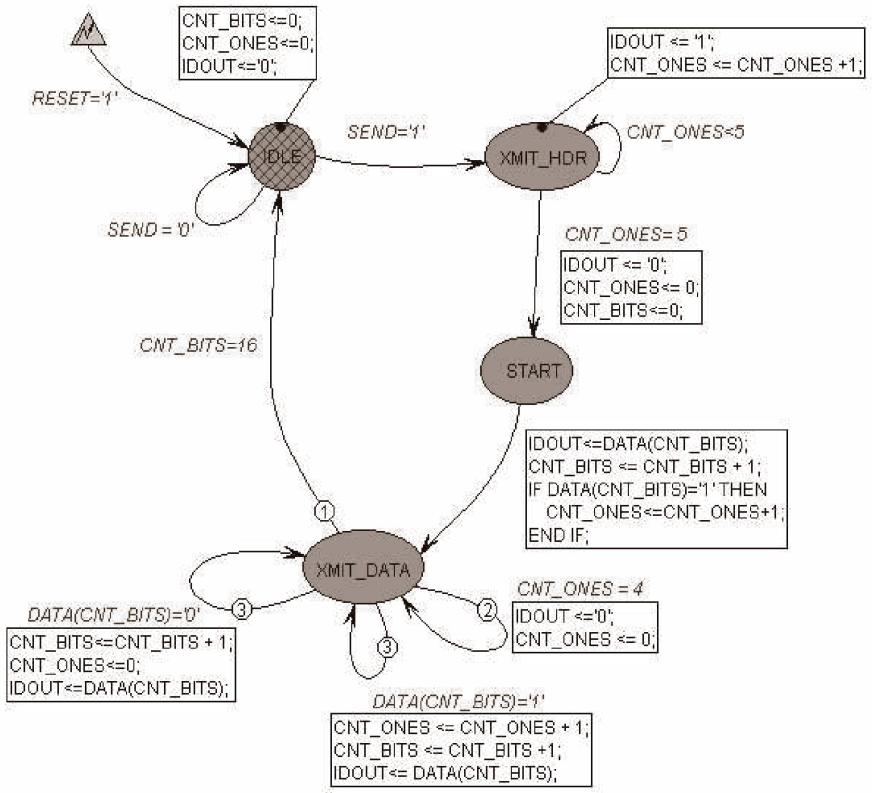


Figure 4

Sample Transmission

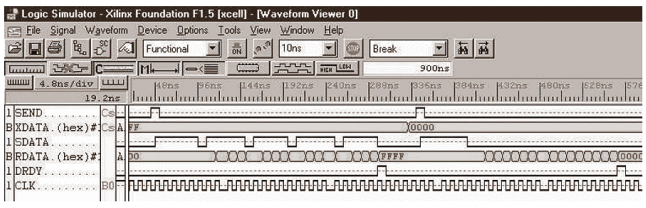


Figure 5

Conclusion

This application offers a simple yet effective method for achieving synchronization between the transmitter and receiver circuits. You can easily extend the design to append parity and CRC bits to the transmitted data.

How to Save Register Content During Power-down

by Peter Alfke, Applications Engineering, Xilinx, peter@xilinx.com

Activating power-down in a SpartanXL device reduces the supply current to less than 100 µa, de-activates all inputs and outputs, and resets all internal flip-flops. The configuration data is retained, but all user data is lost. Here is a way to save your data.

The SpartanXL look-up tables can be used as RAM, and they are not affected by power-down. Therefore, you can save all essential data by loading it into the look-up table space before entering power-down. This method will cost you just two device pins (input A and output B), one external 10K resistor, and an internal state-machine.

Here's How it Works:

1. Connect input pin A to the external power-down signal.
2. Connect output pin B to the dedicated "Powerdown" pin of the device.
3. Connect the resistor between input pin A and the Powerdown pin.
4. Connect output pin B directly to the Powerdown pin.
5. Program the device to drive output pin B High during normal operation.

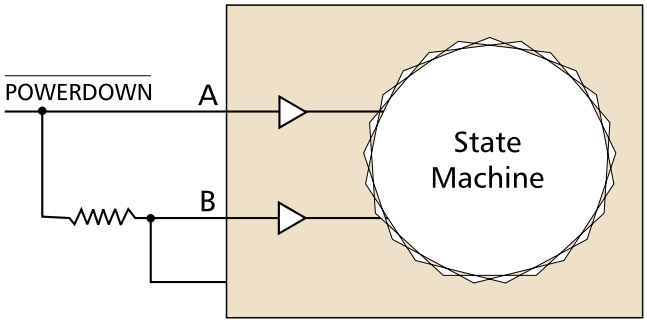


Figure 1

During normal device operation, hold input pin A High (with an external control signal). To initiate power-down, pull pin A Low with an external signal. Program the internal state machine to samples the A input whenever it might be appropriate to enter the power-down state. Sampling A=Low initiates a state-machine sequence that:

1. Stops normal operation.
2. Transfers all vital data into RAM.
3. Pulls output pin B Low.
4. Because pin B is connected to the Powerdown pin, the device immediately goes into power-down mode.

During power-down, the device ignores all inputs by considering them Low, turns off all internal pull-ups, and places all outputs into a 3-state condition. The dedicated Powerdown pin, however, is now held Low by the resistor connecting it to the A input (which is held low by the external power-down signal).

To revert to normal operation, pull pin A (and thus Powerdown) High, causing the chip inputs and outputs to become active again. When the internal state machine samples A=High, it restores data from the look-up tables back to the original flip-flop locations, and then lets your logic restart.

Eliminating SPROM Stand-By Current

The SpartanXL serial-configuration PROM is active only during the milliseconds of configuration time. However, it has a continuous 50 µa of stand-by or idle current that can be completely eliminated by doing the following:

1. Connect the SPROM Ground pin to LDC
2. Program LDC and Din to be in a 3-state condition during user operation.

Some designers have expressed concern that this reduces the SPROM supply voltage by the amount of Vol on the LDC pin. But all CMOS outputs are really resistive, and their voltage drop is proportional to the sink or source current. At 5 mA, Vol is less than 125 mV, as shown in the IBIS files. The Vcc characteristics of the SPROM have sufficient margin to make this operation reliable, as long as Vcc never drops below 3.0 V.

Determining Clock Skew

When the Virtex DLL Drives Multiple Copies of a Clock Off Chip

by Craig Abramson, Field Applications Engineer, craig.abramson@xilinx.com

High-speed logic boards often require multiple low-skew clock buffers to distribute high-speed clocks. With a little attention to detail, Virtex FPGAs can distribute the high-speed clocks for you.

With Virtex FPGAs, you can synchronize an external clock (driven out to your board) with an on-chip clock. Multiple on-chip DLL's make this possible. However, you may need several copies of the clock driven off-board, and using external clock buffer/drivers may negate any benefit of on-chip synchronization. You need a way to determine skew for multiple copies of a clock signal driven off-board by several IOBs.

Single Clock Driven Off-chip

Below is a block diagram of the architecture that would synchronize a clock driven off-chip to an on-board clock.

Single clock, CLK_P, being driven off chip.

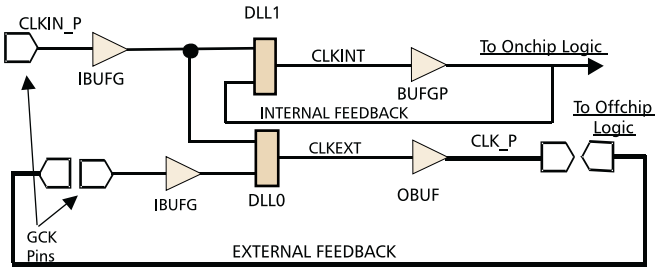


Figure 1

In the past, FPGA's were used primarily to consolidate all of a boards digital logic functions into a single device. With the new architectural features of the Virtex family, even more functionality can be pulled into the FPGA.

Multiple Clocks Driven Offchip

Let's look at an example where we want to replicate the output of DLL0, as shown in Figure 2.

Multiple copies of CLKEXT being driven offchip

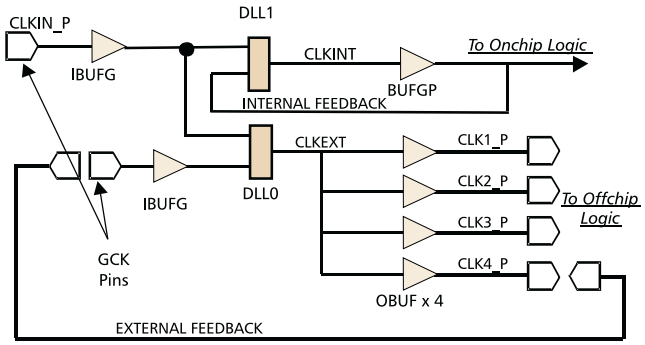


Figure 2

For the purpose of circuit analysis, you may want to minimize the skew on CLKEXT by specifying a constraint. Using the following constraint tells the place and route tools the amount of skew we can tolerate in the signal CLKEXT, which is the input to four different OBUFs.

```
NET CLKEXT MAXSKEW = 100ps ;
```

Note that the skew will be due to the differences in the routing distance from the source DLL's output to the inputs of the OBUFs. The delay through the OBUFs will contribute little to the total skew.

You also need to specify an intelligent choice of pin locations for CLK1_P, CLK2_P, CLK3_P, and CLK4_P. Selecting the four pins closest to DLL0's CLKIN pin gives the best results. (Viewing the chip in Epic can quickly tell you which bonded out IOBs are closest to the DLL). For a V300-BG352 device the UCF entries would look like this:

Continued on the following page

```
NET CLK1_P LOC = B15;
NET CLK2_P LOC = A15;
NET CLK3_P LOC = C13;
NET CLK4_P LOC = B13;
```

Running the design through the tools with the architecture shown in Figure 2, and using the listed location and skew constraints, yields the following results:

```
================================================================
Timing constraint: NET "CLKEXT" MAXSKEW = 100 pS ;
 1 item analyzed, 0 timing errors detected.
 Maximum net skew is 0.037ns.
--------------------------------------------------------------
Slack: 0.063ns CLKEXT
Report: 0.037ns skew meets 0.100ns timing con-
 straint by 0.063ns
From To Delay(ns) Skew(ns)
DLL3.CLK0 B15.O 1.153 0.000
DLL3.CLK0 C13.O 1.190 0.037
DLL3.CLK0 B13.O 1.153 0.000
DLL3.CLK0 A15.O 1.190 0.037
```

Be sure to select "Report Paths in Timing Constraints" in the Timing Reports Tab to direct the Xilinx tools to report everything that's covered by timing constraints (such as the skew of CLKEXT in this example).

Minimizing skew internal to the FPGA is only part of the entire system level skew problem. During board layout you must carefully match the external clock net delays as well. Also, be sure to bring the DLL's clock and external feedback signal in through GCK pins and IBUFG pins as illustrated.

Examine the Design in Epic

By examining the design in Epic, you can see that the careful selection of IOBs made it very easy for the place and route tools to meet the skew specification. If you select the net CLKEXT in the "Epic List" window, you can see that the routes connecting the DLL's output to the input of each of the four IOBs appear to be fairly equal in length. This will translate into a very low skew. To confirm the numbers reported above, while still in Epic and with the CLKEXT net still selected, press the "DELAY" button. The delays associated with the CLKEXT will scroll by in the Epic window.

EPIC view of DLL and associated IOBs.

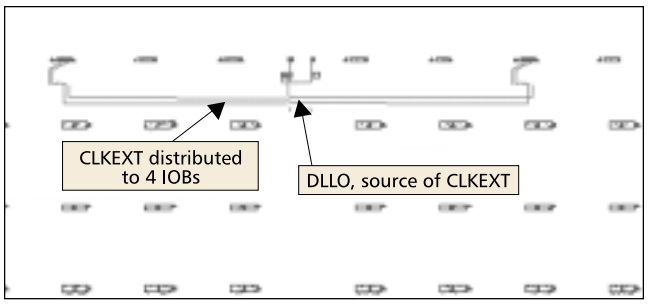


Figure 3

If more copies of the clock need to be driven off-chip, you should first examine the chip in EPIC to determine which I/O locations should be selected. Then, by simply adding the location constraints to your User Constraints File, and a few lines of code to your VHDL or Verilog file, you can perform the same skew analysis.

Code Example

The following code demonstrates the instantiations necessary to drive four IOBs with CLKEXT.

Verilog code example of DLL driving for IOBs.

```
// ----------------------------
// Simple desgn that allows analysis of skew of
// DLL output being distributed to several IOBs.
// ----------------------------
module skewtest (CLKIN_P,CKL1_P,CLK2_P,CLK4_P,cklfbin_p
input CLKIN_P; //Drive the DLLs
input clkfbin_p; //Pin in for external feedback
output CLK1_P;
output CLK2_P;
output CLK3_P;
output CLK4_P;
wire CLKIN,CLKINT,CLKEXT,CLK,clkfbin;

// ----------------------------
// The ibufg is needed to drive the clock input to both DLLs
// as well as the feedback coming from clocks driven off chip.
// ----------------------------
IBUFG bugfio0 (.I(CLKIN_P),.O(CLKIN));
IBUFG buggio1 (.I(clkfbin_p),.O(clkfbin));

// ----------------------------
// DLL0 drives external clock
// ----------------------------
CLKDLL dll0(
  .CLKIN(CLKIN),.CLKFB(clkfbin),.RST(logic0),
  .CLK0(CLKEXT),.slowclk(),.CLK180(),.CLK270(),
  .CLK2X(),.CLKDV(),.LOCKED()
  );

// ----------------------------
// DLL1 drives internal logic
// ----------------------------
CLKDLL dll1(
  .CLKIN(CLKIN),.CLKFB(clkfbin),.RST(logic0),
  .CLK0(CLKINT),.slowclk(),.CLK180(),.CLK270(),
  .CLK2X(),.CLKDV(),.LOCKED()
  );

// ----------------------------
// OBUF (output buffer) is needed to drive a synchronized clock off chip
// ----------------------------
OBUF obuf0 (.I(CLKEXT),.O(CLK1_P));
OBUF obuf1 (.I(CLKEXT),.O(CLK2_P));
OBUF obuf2 (.I(CLKEXT),.O(CLK3_P));
OBUF obuf3 (.I(CLKEXT),.O(CLK4_P));
endmodule
```

Figure 4

Conclusion

In the past, FPGA's were used primarily to consolidate all of a boards digital logic functions into a single device. With the new architectural features of the Virtex family, even more functionality can be pulled into the FPGA. In this example, distributing the DLL output among four carefully chosen IOBs yields 37 ps of skew. This is comparable to the low skew clock drivers currently on the market. In addition the clock signals being driven off chip can be 1 of 13 different I/O voltage standards. So in this scenario, two functions that would ordinarily be handled by devices external to the FPGA are now performed by the Virtex device.

For more information on creating low-skew clocks using the Virtex DLL's, see www.xilinx.com/xapp/xapp132.pdf. For more information on the many I/O standards supported by Virtex FPGAs, see www.xilinx.com/xapp/xapp133.pdf Σ

New IP Center

For FPGA Intellectual Property

A new website that includes 20 new and free reference designs, with details on 12 new XPERTS partners.

by Mike Seither, Director of Public Relations, Xilinx, mike.seither@xilinx.com

Xilinx recently announced the IP Center, an easy to use, comprehensive Xilinx website that offers complete IP solutions. These solutions range from free reference designs, cores, DSP and PCI solutions, to IP implementation tools, third-party cores, specialized system-level services, and vertical application solutions. With the opening of the site, Xilinx also announced 20 new and free reference designs available for download and 12 more partners who have joined the XPERTS consulting program. The IP Center is on the Xilinx website at www.xilinx.com/ipcenter.

"We have a whole new way of helping customers who want to implement system applications on our Virtex FPGAs, be it cores, tools, or consulting. It is all in the Xilinx IP Center," said Babak Hedayati, program director of Core Solutions Marketing at Xilinx. "This is a first for a programmable logic vendor. Using our CORE Generator system, we were first to offer configurable cores over the Internet, and since then we have rapidly expanded our Web-based IP capabilities through the Silicon Xpresso initiative."

"As we introduced new cores or updates over the Web, the benefits to the customer were immediate," Hedayati said. "For example, we were the first company to actually deliver a real compliant 64-bit, 66-MHz PCI solution, and customers were able to immediately download the design and start their implementation as soon as we announced the availability of the product."

System-Level Applications

The system-level solutions area of the Xilinx IP Center offers access to comprehensive information on IP and consulting services for applications such as wireless, networking, XDSL technologies, image and video processing, and computers. Technology solutions such as Xilinx DSP and Xilinx PCI are also covered in detail.

New DSP Reference Designs

The reference designs available in the IP Center include:

- A parameterizable variable-by-variable multiplier for Virtex FPGAs (XCVWARE).
- A pipelined HDL multiplier module that supports signed and unsigned data.
- Fourteen different fixed-point, variable-by-variable multipliers for Virtex FPGAs.
- A Constant Coefficient Multiplier generator (KCM_VGEN) for Virtex FPGAs. It supports 8-, 12-, 16-, and 20-bit data, signed/unsigned, and combinatorial/pipelined versions.
- A parameterizable RAM module using Virtex distributed on-chip RAM and SelectRAM—provided as HDL source code.
- A parameterizable Delay Element module for Virtex FPGAs, utilizing the new Virtex shift-register feature—provided as VHDL/Verilog source code.
- Two fixed-point, complex FFT modules for the XC4000X and Spartan families. It includes a 1024-point, 16-bit complex FFT and a 16- point, 16-bit complex FFT.

More XPERTS

Twelve new third-party consulting companies have joined the Xilinx XPERTS program, bringing the number to more than 40. These experts have experience designing with Xilinx solutions (architecture, software, and IP) and can implement system-level designs or perform IP integration and customization. Some members are focused on the popular Xilinx PCI solution and can integrate the Xilinx PCI cores with customers' unique back-end logic designs. Σ

Questions and Comments from Our Readers

by Roberta Fulton, Technical Marketing Engineer, Xilinx, roberta.fulton@xilinx.com

Your feedback concerning the previous columns of "HDL Advisor" has been most welcome. I am glad that so many of you have found this column helpful. While I can't answer each question personally I will address the most useful ones in this column.

Question 1: How can I assign an "integer" value (128) to an 8 bit "STD_LOGIC_VECTOR" defined signal?

Use the NUMERIC_STD package from the "IEEE Standard VHDL Synthesis Packages." The necessary functions, TO_SIGNED and TO_UNSIGNED, can be found on page 35 of the "IEEE Std 1076.3-1997 IEEE Standard VHDL Synthesis Packages."

If you don't have the IEEE standard manual itself, you may search in the NUMERIC_STD package that comes with your synthesis or simulation package for the function names.

```
— Id: D.3
 function TO_UNSIGNED (ARG: INTEGER; SIZE:
 NATURAL) return SIGNED is

— Id: D.4
 function TO_SIGNED (ARG: INTEGER; SIZE: NATURAL)
 return SIGNED ;
```

Make sure your signal (CNT in the example below) is declared either as a SIGNED or UNSIGNED type then use the appropriate function below:

```
CNT <= TO_SIGNED(128,8); or
CNT <= TO_UNSIGNED(128,8);
```

The opposite translation from SIGNED or UNSIGNED to INTEGER can be performed with the TO_INTEGER functions shown on the same page. First, declare CNT as an integer. Be sure to limit its range if you don't need the whole synthesis tool or simulation tool's default integer range. For most tools the range is 32 bits, with one bit for the sign.

Question 2: I assigned the range of an integer signal (CNT) from 0 to 15 and added 1 to this value, how can I make sure the compiler knows the value (CNT) is an unsigned value? e.g. CNT <= CNT + 1;

Declare the signal (CNT, in your example) as the UNSIGNED type as found in the NUMERIC_STD package from the "IEEE Std 1076.3-1997 IEEE Standard VHDL Synthesis Packages."

```
————————————————————————————————————————
— Numeric array type definitions
—
  ————————————————————————————————————————

 type UNSIGNED is array (NATURAL range <>) of
 STD_LOGIC;
 type SIGNED is array (NATURAL range <>) of
 STD_LOGIC;
```

This package will also allow you to add the integer 1 to the unsigned value of CNT. Normally two different types cannot be added together in VHDL, but the NUMERIC_STD package allows you to add SIGNED or UNSIGNED types to integers because it has overloaded the operator. This means it does the type conversion for you.

```
— Id: A.8
 function "+" (L: SIGNED; R: INTEGER) return
 SIGNED;
— Result subtype: SIGNED(L'LENGTH-1 downto 0).
Result: Adds a SIGNED vector, L, to an INTEGER, R.

— Id: A.5
 function "+" (L: UNSIGNED; R: NATURAL) return
 UNSIGNED is
 begin
 return L + TO_UNSIGNED(R, L'LENGTH);
 end "+";
```

Please note that for an UNSIGNED number the integer must be natural.

Question 3: How can I compare a "STD_LOGIC_VECTOR" defined signal to a "integer" value? e.g. if (CNT = 38) then ...

If you used SIGNED or UNSIGNED types instead of STD_LOGIC_VECTOR, you could use the NUMERIC_STD package as referred to above for its overloaded relational operators for SIGNED, UNSIGNED and integer types. For example:

```
-- Id: C.29
function "=" (L:UNSIGNED; R: NATURAL) return
BOOLEAN is ...
```

You could also us the type conversion in the NUMERIC_STD pacakge. I don't recommend using more than one numerica package, even if you found another one with overloaded operators for STD_LOGIC_VECTOR and integer types. Some numeric packages in the public doman use the same types and overloaded operators as the IEEE standard. The reader can become confused which type and operator definitions are actually being used.

For clarity, I prefer that type conversions be stated separately rather than as part of the comparison statement itself as given below:

```
NEW_B <= TO_INTEGER(B);
```

Then you can use the new signal in the comparison as in:

```
if (NEW_B > 5) then ……..
```

NOTE: Not all synthesis companies may be fully compliant with the IEEE Std 1076.3-1997. You may need to use another comparable numeric or arithmetic package instead.

The package STD_LOGIC_ARITH, created by Synopsys before the IEEE standard was available, is an example of a package that also contains the SIGNED and UNSIGNED types and the overloaded "+" operator. The STD_LOGIC_SIGNED, also from Synopsys, has the boolean comparison operators such as >=. If STD_LOGIC_SIGNED is used however, you must also use STD_LOGIC_ARITH which it calls. In that case NUMERIC_STD should not be used or you could get confused about which package is providing the operator definitions.

Theoretically, the packages should be portable no matter what their source. This is generally true of simulators. In practice, synthesizers may take awhile to adapt to another new package; when they do, they may have to adapt the source code of the package to the synthesis attributes. These attributes have no affect on the simulation.

Reader Comment 1: "... when it comes to VHDL coding approaches and synthesis results, it's very important to us to record the optimal approach(es) for both minimal area and minimal delay."

For the comparators I discuss in my *HDL Advisor Column* for *Xcell 31* (1Q99), the results of "best coding style" differ depending on whether you compile for area or delay.

The "fastest" coding style under one optimization such as "area" may not be the fastest when compiling for "delay." Note that some other variables are the software tool itself and the version of that tool. I am restricted by licensing agreements to not making direct comparisons between synthesizers. Therefore, I take a result from one tool, in the latest version available at the time, to get examples that fairly represent the results for all of the available tools.

For clarity, the article's "best results" were optimized for minimum area, and word_wise compares were faster. I generally try minimum area first as I find the results usually give close to the best delays without exploding the number of CLBs.

If minimum delay is selected, then bit-wise is faster with a sacrifice of using more CLBS.

"Best" really depends on how close you are to meeting area or performance constraints. For one design you may be willing to sacrifice a little area to tweak the performance, or give up a little performance in order to fit in a smaller size device.

As mentioned earlier, for synthesis tools especially, your results may vary. The algorithms change between vendors and even between versions from the same vendor.

I would advise you to try it yourself, with your tool set. And, don't forget that a different word-size causes different algorithms to kick in, so what is true for 4-bits may not be true for 32-bits.

Reader COMMENT 2: "Maybe you could have explained that this shouldn't work if you're following the VHDL-standard , but in practice it's ok for some synthesis vendors."

The reader notes that for the use of "-" as a "don't care" in VHDL from *HDL Advisor Column*, *Xcell 30*, I imply that the tools are broken, if they don't allow this.

As the reader states, the vendor that follows my example is a bit outside of the VHDL standard. They've taken the same approach as in Verilog. So, in practice, it works for some synthesis tools but it is wrong according to the VHDL-standard.

My tendency too is to be a standard-follower, but you must decide whether "standard" or "general practice" is best for you.

And as experienced HDL users have long noted, interpretations of the HDL standards may also be different between different synthesis and simulation vendors, especially IEEE Std 1087-1987 on which some vendors still rely, though the standard was updated in 1993. So, once again, try the examples in your choice of tools.

Verilog CBT New Computer Based Training

Verilog CBT is the first computer-based training course from Xilinx, allowing you to learn Verilog at your own pace, without ever leaving your office.

by Alicia Tripp, Product Marketing Manager for Services and Support, alicia@xilinx.com

From the convenience of your desktop, you can learn a comprehensive Verilog language by using the Verilog CBT program. Verilog CBT is a training technique that is based on our popular three-day Verilog class. This professional learning tool delivers all the essentials for writing, simulating, and synthesizing Verilog HDL for Xilinx devices. Thus, converting the conventional course, traditionally delivered in day segments, into a computerized self-study program. As a result, you are provided with a comprehensive Verilog learning application right at your desktop.

Key Features

- High quality HDL instruction developed by Verilog experts.
- Easy navigation through a complete set of Verilog training modules.
- Fully indexed for easy reference.
- Extensive design exercises and Verilog examples.
- Coding guidelines, formatting conventions, and practical tips.
- Glossary of key Verilog terms.
- Instruction and advice to help you avoid common errors.
- Serves as an excellent reference tool for the life of a project.
- On-screen instruction with matching audio narration.
- Powerful Macromedia ™ CBT application.

Get up to Speed Quickly

The main lesson window presents code examples, a diagram, graphical tools, and tips. A secondary window contains the instructor's narrative. By working through each of the ten modules of this comprehensive course, you will develop good Verilog coding techniques and gain HDL design experience in a hands-on training environment.

How to Order

Verilog CBT is available now for only $295.00. To order a copy of Verilog CBT, contact your local Xilinx Distributor and use the following part number to place the order: TC-VERILOG-CD.

For additional information about Verilog CBT and other training courses available from Xilinx, visit http://support.xilinx.com. Under the Education heading, select Courses. Or, call 877-XLX-CLASS. Σ

Verilog CBT is a training technique that is based on our popular three-day Verilog class. This professional learning tool delivers all the essentials for writing, simulating, and synthesizing Verilog HDL for Xilinx devices.

Xilinx Trade Show Programs

See our 1999 International Trade Show schedule and plan to attend where possible.

by Darby Mason-Merchant, Trade Show Manager, Xilinx, darby@xilinx.com

1999 started off with a flurry of activity around the globe for Xilinx. With our new trade show strategy, Xilinx has ventured into more vertical events such as the Portable Design 99 in San Diego and the Wireless Symposium 99 in San Jose. These two shows were the perfect opportunity to show Xilinx SpartanXL and XC9500XL CPLD families at work in applications requiring low power capabilities at a lower cost.

Through our worldwide tradeshow programs, you can get hands-on experience with industry-leading PLD technologies such as our million-gate Virtex family, Silicon Xpresso, Foundation Series and Alliance Series software, WebFITTER, DSP cores, and Real PCI 64/66 cores. Trade show information is listed on our WebLINX page at: www.xilinx.com/company/tradeshows.htm

Wim Roelandts Gives Keynote Address at ICASSP 99

Xilinx was proud to have our CEO, Wim Roelandts, give the keynote address at the International Conference on Acoustics, Speech, and Signal Processing (ICASSP 99) in mid-March in Phoenix. His presentation titled "IC Technology Enables New Programmable DSP Solutions" gave the audience an eye-opening look at what our new technology offers for creating high performance DSP applications. The audience was noticeably impressed by how far FPGA technology has progressed in providing solutions for communications and image processing applications, and by our new Internet Reconfigureable Logic (IRL) technology that makes it easy to remotely update equipment, in the field. A copy of this presentation can be found at: www.xilinx.com/company/tradeshows/posticassp99.htm

36th Design Automation Conference

Show Dates: June 21-23, Ernst N. Morial Convention Center, New Orleans, LA

We invite you to attend DAC 99, the foremost trade show event for design automation in the US. Stop by our booth (#2532) to learn and witness how engineering teams around the world will collectively design for mega-gate devices as well as how users will remotely perform field upgrades via the Internet. With Xilinx, the tools and IP are available today to make this all possible. Σ

1999 N.A. Trade Show Schedule

Feb.1-4	ICEPD99 (Portable Design 99)	San Diego, Calif.
February 1-4	ICEPD99 (Portable Design 99)	San Diego, Calif.
Feb. 21-23	FPGA99 Conference	Monterey, Calif.
Feb. 22-24	Portable by Design 99	San Jose, Calif.
March 15-19	ICASSP99	Phoenix, Ariz.
March 26-27	Palm Beach Investors Expo	Palm Beach, Fla.
March 30	SNUG 99	San Jose, Calif.
April 26-28	DSP Spring Conference 99	Santa Clara, Calif.
April 99	FCCM Conference 99	Napa, Calif.
May 24-27	PC Developers Expo 99	Santa Clara, Calif.
June 9-11	WITI Technology Summit 99	Santa Clara, Calif.
June 21-25	36th Design Automation Conf.	New Orleans
Nov 1-4	DSP World / ICSPAT Conf. 99	Orlando, Fla.

European Trade Show Schedule

April 13-16	Intertronic	Paris
April 20	Wireless Comms 99	Kista, Stockholm
April 23	Wireless Comms 99	Bracknell, UK
May 11-12	Advanced FPGA and PLD Day	Sandown, UK
May 18	Advanced FPGA and PLD Day	Stockholm
May 26-27	Embedded Systems Show	Olympia, UK
June 2-4	Embtech World 99	Paris
Nov 1-2	IP Europe 99	Edinburgh

Japanese Schedule

June 30–July 1	7th Japan FGPA/PLD Conf.	Sunshine City

South East Asian Schedule

April 8-9	IIC '99	Guanzhou, China
April 12-13	IIC '99	Shanghai, China
April 15-16	IIC '99	Beijing, China
Oct. 21-22	EDA&T	Hsinchu,Taiwan
Oct. 25-26	EDA&T	Beijing, China
December	Multi-Tech	Taipei,Taiwan

For more information about Xilinx worldwide Trade Show Programs, please contact one of the following Xilinx team members or see our website at: www.xilinx.com/company/tradeshows.htm

US Shows: Darby Mason-Merchant at: darby@xilinx.com.

European Shows: Andrea Fionda at: andrea.fionda@xilinx.com.

Japanese Shows: Tetsuo Souyama at: tetsuo.souyama@xilinx.com

SouthEast Asian Shows: Mary Leung at: mary.leung@xilinx.com

	Usable Gates (K)	Logic Gates	System Gates (see note 1)	Logic Cells (see note 2)	Total CLBs	Total Flip-Flop	Max. I/O	Max. RAM Bits	Config. Memory (K Bits)	XC1700 Serial PROM Req.
XC4000EX Family - 5 Volt										**.5µm Triple Layer Metal Process**
XC/XQ4028EX	18-50	28K	50K	2,432	1,024	2,560	256	32.8K	668	XC1701
XC4036EX	22-65	36K	65K	3,078	1,296	3,168	288	41.5K	833	XC1701
XC4000XLA Family - 3.3 Volt										**Advanced .35µm Five Layer Metal Process**
XC/XQ4013XLA	10-30	13K	30K	1,368	576	1,536	192	18.4K	393	XC17512L
XC4020XLA	13-40	20K	40K	1,862	784	2,016	224	25.1K	522	XC17512L
XC4028XLA	18-50	28K	50K	2,432	1,024	2,560	256	32.8K	668	XC1701L
XC/XQ4036XLA	22-65	36K	65K	3,078	1,296	3,168	288	41.5K	833	XC1701L
XC4044XLA	27-80	44K	80K	3,800	1,600	3,840	320	51.2K	1,015	XC1701L
XC4052XLA	33-100	52K	100K	4,598	1,936	4,576	352	62.0K	1,215	XC1702L
XC/XQ4062XLA	40-130	62K	130K	5,472	2,304	5,376	384	73.8K	1,434	XC1702L
XC4085XLA	55-180	58K	180K	7,448	3,136	7,168	448	100.4K	1,925	XC1702L
XC4000XV Family - 2.5 Volt										**.25µm Five Layer Metal Process**
XC40110XV	75-200	110K	220K	9,728	4,096	8,704	448	131.1K	2,488	XC1704L
XC40150XV	100-300	150K	300K	12,312	5,184	11,520	448	165.9K	3,373	XC1704L
XC40200XV	130-400	200K	400K	16,758	7,056	15,456	448	225.8K	4,551	XC1704L & XC17512L
XC40250XV	160-500	250K	500K	20,102	8,464	18,400	448	270.9K	5,434	XC1704L & XC1702L
Spartan Family - 5 Volt and 3.3 Volt (XL)										**Advanced .5µm & .35µm (XL) Process**
XCS05/XL	2-5	3K	5K	238	100	360	80	3.2K	54	XC17S05 (XL)
XCS10/XL	3-10	5K	10K	466	196	616	112	6.3K	95	XC17S10 (XL)
XCS20/XL	7-20	10K	20K	950	400	1,120	160	12.8K	179	XC17S20 (XL)
XCS30/XL	10-30	13K	30K	1,368	576	1,536	192	18.4K	249	XC17S30 (XL)
XCS40/XL	13-40	20K	40K	1,862	784	2,016	224	25.1K	330	XC17S40 (XL)
Virtex Family - 2.5 Volt										
XCV50	20-50	20K	50K	1,728	384	1,536	180	57K	559	XC1701L
XCV100	30-100	30K	100K	2,700	600	2,400	180	79K	781	XC1701L
XCV150	45-150	45K	150K	3,888	864	3,456	260	104K	1,041	XC1701L
XCV200	60-200	60K	200K	5,292	1,176	4,704	284	133K	1,336	XC1702L
XCV300	80-300	80K	300K	6,912	1,536	6,144	316	164K	1,752	XC1702L
XCV400	130-400	130K	400K	10,800	2,400	9,600	404	236K	2,546	XC1704L
XCV600	185-600	185K	600K	15,552	3,456	13,824	500	319K	3,608	XC1704L
XCV800	250-800	250K	800K	21,168	4,704	18,816	512	416K	4,715	XC1704L & XC1701L
XCV1000	330-1M	330K	1M	27,648	6,144	24,576	612	524K	6,128	XC1704L & XC1702L

QPRO™ High-Reliability QML Products	Usable Gates (K)	Logic Gates (k)	System Gates (k) (see note 1)	Logic Cells (see note 2)	Total CLBs	Total Flip-Flop	Max. I/O	Max. RAM Bits (k)	Config. Memory (K Bits)	XC/XQ SPROM Req.
XQ4000EX Family - 5 Volt										**.5µm Triple Layer Metal Process**
XQ4005E	3-9	5	9	466	196	616	112	6.3	9.5	XC17256D
XQ4010E	7-20	10	20	950	400	1,120	160	12.8	178	XC17256D
XQ4013E	10-30	13	30	1,368	576	1,536	192	18.4	248	XC17256D
XQ4025E	15-45	25	45	2,432	1,024	2,560	256	32.7	422	XC17256Dx2
XQ4028EX	18-50	28	50	2,432	1,024	2,560	256	32,7	422	XC17256Dx2
XC4000XL Family - 3.3 Volt										**Advanced .35µm Five Layer Metal Process**
XQ4013XL	10-30	13	30	1,368	576	1,536	192	18.4	393	XQ1701L
XQ4036XL	22-65	36	65	3,078	1,296	3,168	288	41.5	833	XQ1701L
XQ4062XL	40-130	62	130	5,472	2,304	5,376	384	73.8	1,434	XQ1701Lx2
XQ4085XL	55-180	85	180	7,448	3,136	7,168	448	100.4	1,925	XQ1701Lx2
Virtex Family - 2.5 Volt										**.25µm Five Layer Metal Process**
XQV100	30-100	30	100	2,700	600	2,400	180	79	781	XQ1701L
XQV300	80-300	80	300	6,912	1,536	6,144	316	164	1,751	XQ1701Lx2 or XQ1704L
XQV600	185-600	185	600	15,552	3,456	13,824	500	319	3,608	
XQV1000	330-1M	330	1M	27,648	6,144	24,576	512	524	6,128	

	Macrocells	Max. I/O	Pin-to-Pin Delay (ns)
XC9536	36	34	5
XC9572	72	72	7.5
XC95108	108	108	7.5
XC95144	144	133	7.5
XC95216	216	166	10
XC95288	288	192	10
XC9500XL Family - 3.3 Volt CPLDs			
XC9536XL	36	36	4
XC9572XL	72	72	5
XC95144XL	144	117	5
XC95288XL	288	192	6
XC9500XV Family - 3.3 Volt CPLDs			
XC9536XL	36	36	3.5
XC9572XL	72	72	4
XC95144XL	144	117	4
XC95288XL	288	192	5

Speed Grade Options	Slowest	Fastest
XC4000EX	-4	-2
XC4000XLA	-09	-07
XC4000XV	-09	-07
Virtex	-4	-6
Spartan	-3	-4
SpartanXL	-4	-5
XQ4005X	-4	-4
XQ4000XL	-3	-3

Serial PROM Package Options	Density	PD8	SO8	VO8	PC20	SO20	PC44	VQ44	3 Volt	5 Volt
XC1736E	36K	Y	Y	Y	Y					Y
XC1765E (EL)	65K	Y	Y	Y	Y				Y (EL)	Y
XC17128E (EL)	128K	Y		Y	Y				Y (EL)	Y
XC17256E (EL)	256K	Y		Y	Y				Y (EL)	Y
XC17512L	512K	Y			Y	Y			Y	
XC1701	1M	Y			Y	Y				Y
XC/XQ1701L	1M	Y			Y	Y			Y	
XC1702L	2M						Y	Y	Y	
XC1704L	4M						Y	Y	Y	
XC17S05	54K	Y		Y						Y
XC17S05XL	55K	Y		Y					Y	
XC17S10	95K	Y		Y						Y
XC17S10XL	96K	Y		Y					Y	
XC17S20	178K	Y		Y						Y
XC17S20XL	179K	Y		Y					Y	
XC17S30	248K	Y		Y						Y
XC17S30XL	249K	Y		Y					Y	
XC17S40	329K	Y				Y				Y
XC17S40XL	331K	Y				Y			Y	

QPRO Serial PROM Package Options	Density	DD8	CC44	S020
XC17256D	256K	Y	-	-
XQ1701L	1M	-	Y	Y
XQ1704L	4M	-	Y	Y

Notes:
1. System Gates include 20-30% of CLBs used as RAM
2. A Logic Cell is defined as a 4 input LUT and a Register

Ordering Information for FPGAs / CPLDs

Example:

XC4036XLA - 08 PQ 160 C

- XC4036XLA: Device Type / Speed Grade
- 08: Speed Grade
- PQ: Package Type
- 160: Number of Pins
- C: Temp. Range

FPGA / CPLD Package Options and User I/O

	XC4000EX (5 Volt) FPGAs		XC4000XLA (3.3 Volt) FPGAs								XC4000XV (2.5 Volt) FPGAs				Spartan™ (5V, XL 3.3V) FPGAs					XC9500 (5 Volt) CPLDs						XC9500XL/XV (3.3V/2.5V) CPLDs				Virtex™ (2.5 Volt)								
	XC4028EX	XC4036EX	XC4013XLA	XC4020XLA	XC4028XLA	XC4036XLA	XC4044XLA	XC4052XLA	XC4062XLA	XC4085XLA	XC40110XV	XC40150XV	XC40200XV	XC40250XV	XCS05/XL	XCS10/XL	XCS20/XL	XCS30/XL	XCS40/XL	XC9536	XC9572	XC95108	XC95144	XC95216	XC95288	XC9536XL/XV	XC9572XL/XV	XC95144XL/XV	XC95288XL/XV	XCV50	XCV100	XCV150	XCV200	XCV300	XCV400	XCV600	XCV800	XCV1000
IOBs	256	288	192	224	256	288	320	352	384	448	448	448	448	448	80	112	160	192	224	34	72	108	133	166	192	36	72	117	192	192	240	288	336	384	480	576	672	768
PLCC Packages																																						
44																				34	34					34	34											
84															61	61					69	69																
CSP Packages																																						
48																				34						36	38											
144																												117		94	94							
PQFP / HQFP Packages																																						
100																					72	81	81														
160			129	129	129	129	129	129	129	129												108	133	133													
208	160		160	160	160	160	160	160	160	160							160	169	169					166	168				168								
240	193	193	192	193	193	193	193	193	193	193	193	193						192	193										166	166	166	166	166	166	166	166	
304	256	256			256	256	256	256	256	256																											
VQFP Packages																																						
44																				34																	
64																										36	52											
100															77	77	77	77																				
TQFP / HTFP Packages																																						
100																					72	81	81				72	81									
144																112	113	113										117	117	98	98							
CBFP Packages																																						
228	192																																					
BGA Packages																																						
256			192	205	205													192	205										180	180	180	180					
352	256	288			256	288	289	289	289	289	289	289												166	192			192			260	260	260				
432		288				288	320	352	352	352	352	352	352	352																			316	316	316	316	
560								352	384	448	432	432	432	432																				404	404	404	404
PGA Packages																																						
299	256																																					
411		288																																				
559												448		448																								
FinePitch BGA																																						
256																														176	176	176	176					
456																																260	284	312				
600																																			404	404	404	
680																																				500	512	512

Available Now

Available Q199

XC4000XLA Available Sept. (XC4013, XC4062 (HQ2400)
Remainder available Q4

SpartanXL available Q4

XC9500XV Available 3Q99 *(XC9536XV, XC9572XV, XC95288XV)*

Xilinx Alliance Series and Foundation Series Features

	Alliance Series		Foundation Series			
Features Included			**Design Environment**			
	Schematic & Synthesis		Schematic & ABEL		Schematic & Synthesis	
	ALI-BAS	ALI-STD	FND-BAS	FND-STD	FND-BSX	FND-EXP
EDA Libraries and Interfaces for Cadence, Mentor, Synopys, and ViewLogic	✓	✓				
Turns Engine (Workstation Only)	✓	✓				
Synthesis Constraint Editor and Timing Analyzer						✓
Esperan MasterClass Lite VHDL Tutorial					✓	✓
HDL Synthesis Tools (ABEL, VHDL, and Verilog)					✓	✓
HDL Design Tools: HDL Wizard, Context Sensitive Editor, Graphical State Editor, and Language Assistant			✓	✓	✓	✓
Schematic Editor			✓	✓	✓	✓
Simulator (Functional and Timing)			✓	✓	✓	✓
HDL Synthesis Libraries (UniSim and Simprim)	✓	✓	✓	✓	✓	✓
Implementation Tools: Design Manager, Flow Engine, Timing Analyzer, Hardware Debugger, LogiBLOX, JTAGProgrammer, PROM File Formatter, Graphical Constraints Editor, Graphical Floorplanner	✓	✓	✓	✓	✓	✓
EDIF, VHDL (VITAL), and Verilog Back Annotation	✓	✓	✓	✓	✓	✓
LogiBLOX™ Module Generator	✓	✓	✓	✓	✓	✓
Xilinx CORE Generator	✓	✓	✓	✓	✓	✓
CPLD Devices (XC9500 and XC9500XL)	✓	✓	✓	✓	✓	✓
FPGA (Low Density/High Volume Devices): XC4000E/XL (Up to XC4010E/XL) Spartan and SpartanXL (All) XC3000A, XC3000L, XC3100A, XC3100L XC5200 (Up to XC5210)	✓		✓		✓	
FPGA (Unlimited Device Support): Virtex XC4000E/X (All) Spartan and SpartanXL (All) XC3x00A/L (All) XC5200 (All)		✓		✓		✓
Xchecker Cable (Workstation Only)	✓	✓				
JTAG Cable (PC Only)	✓	✓	✓	✓	✓	✓

VIRTEX The Industry's First One-Million Gate FPGA, in an all NU Design Kit!

Now there's a simple way to experience the unprecedented density and performance available with VIRTEX FPGAs from Xilinx. Ranging in densities from 50,000 to 1,000,000 system gates, Virtex is destined to change the way you design, both today, and well into then next millinium.

NU HORIZONS has combined the VIRTEX XV300 device with Xilinx' Alliance Series™ Software1.5i, to offer a logic solution that's designed to meet the challenges of your most complex next generation designs.

XILINX®
The Programmable Logic Company℠

VIRTEX™ *It's all about design flexibility*

Available in a variety of package options

Nu Horizons has a technically strong sales and engineering team ready to service your needs. Let us show you that we understand how important flexibility, quality and speed are to you. Both our automated New York and San Jose warehouses are ISO9002 Registered. We are also "Year 2000 Compliant." Our value-added services include state-or-the-art programming, tape and reel, custom bar coding and vacuum heat sealing. We also offer customizable Electronic Commerce called TEAM, along with auto-replenishment, bonded inventory and Material Management Programs. Call our toll free number to speak to any of our Xilinx product experts, or visit our web site at: ***www.nuhorizons.com.*** *Remember if you need a programmable logic solution CALL Nu Horizons today! We can ship it to you immediately!*

YEAR2000 COMPLIANT

ISO 9002 REGISTERED

World Wide Web:
www.nuhorizons.com

E-Mail:
sales@nuhorizons.com

Toll Free 1-800-747-NUHO

— **Corporate Headquarters** 70 Maxess Road, Melville, New York — **ALABAMA** Huntsville — **ARIZONA** Tempe — **CALIFORNIA** San Jose, Thousand Oaks, Irvine, Sacramento, San Diego — **FLORIDA** Ft. Lauderdale, Altamonte Springs, Tampa — **GEORGIA** Norcross — **ILLINOIS** Schaumburg — **MARYLAND** Columbia-Baltimore, Washington Metro — **MASSACHUSETTS** Wakefield — **MINNESOTA** Eden Prairie — **NEW JERSEY PENNSYLVANIA** Pine Brook, Mt Laurel, Eastern Pennsylvania — **NEW YORK** Melville, Rochester — **NORTH CAROLINA** Raleigh — **OHIO** Cleveland — **TEXAS** Carrolton, Austin — **INTERNATIONAL**

NU HORIZONS ELECTRONICS CORP.