ISSUE 51, WINTER 2004

Issue 51
Winter 2004

Xcell journal

THE AUTHORITATIVE JOURNAL FOR PROGRAMMABLE LOGIC USERS

Times Square Outshines the World with the Biggest, Brightest LED Display Ever Created

EMBEDDED SYSTEMS

High-Bandwidth TCP/IP

Software-Defined Radio

Embedded Nucleus PLUS

DSP

H.264/AVC on FPGAs

GSM Modems

DSP Design Kits

Virtex-4 FPGAs

Breakthrough Performance at Lowest Cost

COVER STORY Sign of the Times

XILINX®

Only FPGAs on 90nm
Shipping Since March 2003

The New SPARTAN™-3 Make It *Your* ASIC

SPARTAN™-3 FPGA

The world's lowest-cost FPGAs

Spartan-3 Platform FPGAs deliver everything you need at the price you want. Leading the way in 90nm process technology, the new Spartan-3 devices are driving down costs in a huge range of high-capability, cost-sensitive applications. With the industry's widest density range in its class — 50K to 5 Million gates — the Spartan-3 family gives you unbeatable value and flexibility.

Lots of features … without compromising on price

Check it out. You get 18x18 embedded multipliers for XtremeDSP™ processing in a low-cost FPGA. Our unique staggered pad technology delivers a ton of I/Os for total connectivity solutions. Plus our XCITE technology improves signal integrity, while eliminating hundreds of resistors to simplify board layout and reduce your bill of materials.

With the lowest cost per I/O and lowest cost per logic cell, Spartan-3 Platform FPGAs are the perfect fit for any design … and any budget.

MAKE IT YOUR ASIC

XILINX®
The Programmable Logic Company℠

For more information visit
www.xilinx.com/spartan3

Earth Friendly — **Pb-free devices available now**

FORTUNE® 2004 100 BEST COMPANIES TO WORK FOR

1984 20 YEARS 2004 LEADERSHIP

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

MANAGING EDITOR Forrest Couch
forrest.couch@xilinx.com
408-879-5270

ASSISTANT MANAGING EDITOR Charmaine Cooper Hussain

XCELL ONLINE EDITOR Tom Pyles
tom.pyles@xilinx.com
720-652-3883

ADVERTISING SALES Dan Teie
1-800-493-5551

ART DIRECTOR Scott Blair

Xcell Publications

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

And the Number Please...

What does the number 6,759,852 represent? Well, I guess it could represent a lot of different things. For example, it could be the current population of Chennai, India. It could be the phone number of Training Academy Ireland (they're nice folks, but please don't call them to verify). Or it could be the student ID number of a computer science major attending the University of Manitoba, Canada.

Had you chosen any one of these you would have been correct, but you would not have guessed the answer I was looking for.

On July 6, 2004, Xilinx® reached the 1,000th patent landmark. The patent, "VDD Detection Path in Power-Up Circuit," was U.S. Patent number 6,759,852. (I can't believe you didn't guess this.) It was issued to Maheen A. Samad in our General Products Division, Engineering department.

So what's the big deal, you might ask? Although our corporate pride may runneth over, our patent count doesn't hold a candle to some corporate giants. That may be true, but Xilinx was founded with innovation at its core, beginning with Ross Freeman's invention of the FPGA and continuing with innovative practices and ideas, many of which are commemorated in the patent hallway at our corporate headquarters.

Using our R&D dollars as a metric to measure our efficiency in converting innovation into patents, Xilinx – as a high-tech company – ranks second only to IBM™. Xilinx also ranks 131st in the number of patents held, making it one of the most innovative companies worldwide.

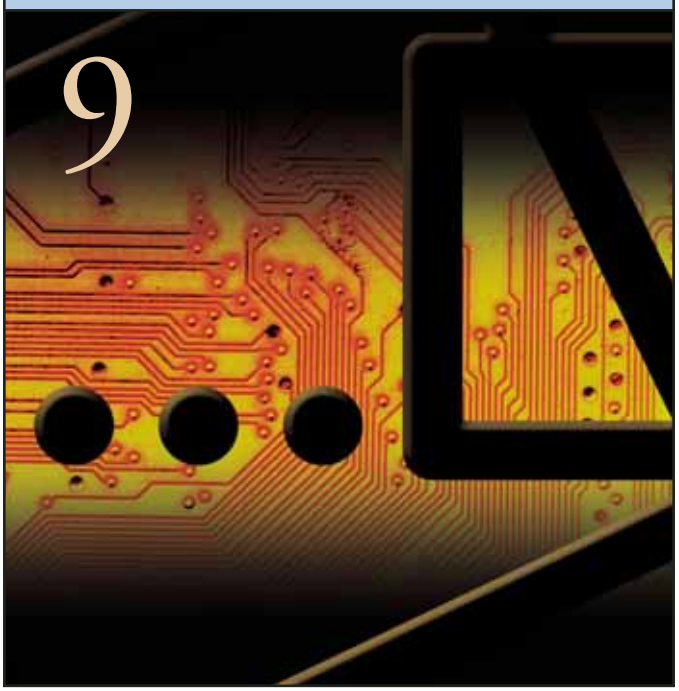
This milestone – while not hugely significant in terms of the raw number – is more about celebrating the continued innovation from Xilinx, both in the form of technology patents as well as business acumen.

This issue of the *Xcell Journal* features articles in two key technology areas: digital signal processing (DSP) and embedded processors. This issue also includes an article on the new Virtex-4™ family of FPGAs, which offers three platforms optimized for logic, DSP, and embedded processor applications. And speaking of innovation, the Virtex-4 family includes more than 120 new (and, of course, patented) features, many of which are specific to supporting high-performance signal processing and embedded processors.

With the launch of the Virtex-4 multi-platform FPGA family, the Xilinx vision expands to encompass programmable systems, which include logic, embedded processing, and very high-performance digital signal processing. As illustrated in the many articles in this issue, programmable technologies provide customers further flexibility and performance benefits to inspire innovation.

Forrest Couch
Managing Editor

EMBEDDED SYSTEMS



Embedded with Xilinx

In this series on embedded processing, the *Xcell Journal* samples a broad base of embedded processing applications.

DIGITAL SIGNAL PROCESSING

Taking Digital Signal Processing to the Extreme

In this series on digital signal processing, the *Xcell Journal* spotlights the challenges of and solutions to developing extremely high-performance DSP applications.

COVER STORY

Sign of the Times

Xilinx makes high-tech outdoor advertising in Times Square possible.

10

WINTER 2004, ISSUE 51

Xcell journal

Considerations for High-Bandwidth TCP/IP PowerPC Applications

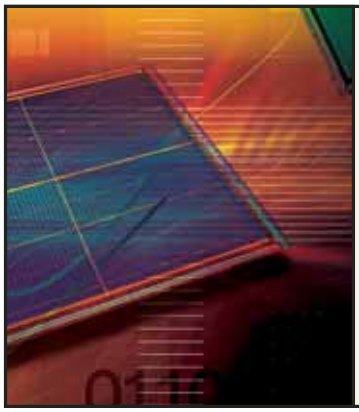
The Xilinx Gigabit System Reference Design maximizes TCP/IP performance. 14

Virtex-4: Breakthrough Performance at the Lowest Cost

Virtex-4 FPGAs deliver what you've been looking for. 33

Implementing the H.264/AVC Video Coding Standard on FPGAs

Xilinx Virtex FPGAs provide excellent co-, pre-, and post-processing hardware acceleration solutions. 40



Simulink Brings Model-Based Design to Embedded Signal Processing

The complexity of FPGA-based signal processing systems drives the need for new development approaches. 66

To subscribe to the *Xcell Journal* or to view the web-based *Xcell Online*, visit *www.xilinx.com/xcell/*.

Focusing on Programmable Technologies

Xilinx clearly leads the programmable logic business; now we are expanding into other key programmable technologies.

by Wim Roelandts
CEO, Xilinx, Inc.

Throughout the history of semiconductors, when a technology becomes programmable, it dominates. That's why, in addition to programmable logic, we have defined two key programmable technologies on which we will focus: digital signal processing and embedded processing. We have the right technology and the right business model to be a major player in all of these important and fast-growing markets.

Digital Signal Processing

Xilinx FPGAs have long been the highest performance technology for building DSP designs. With our devices and software, you can build systems that are two to three orders of magnitude faster than what a dedicated DSP device can do on its own. Putting our extremely high performance DSP functionality next to a programmable DSP allows DSP designers to develop systems with unprecedented performance and value. You also get many other advantages offered by FPGAs, including flexibility, fast time to market, and higher levels of system integration. There simply is no easier, faster, or better way to develop extreme performance DSP designs.

For example, with our new Virtex-4™ FPGA family, you can achieve 256 GigaMACs (billions of multiple accumulates per second). We have achieved this amazing performance through both advanced architecture and silicon fabrication technologies.

Applications for our high-performance DSP capabilities are growing. Broadcasting or video conferencing for high-definition television, for example, is rapidly being converted to the H.264 format. This standard requires a lot of processing power, as the target is to have the quality of MPEG-2 video at one-half the bit rate. Sophisticated motion compensation schemes are being used to achieve this goal. Standard video processors can perform this

function at smaller screen sizes up to common intermediate format (CIF) resolutions, but to go beyond this to standard definition (SD) or high definition (HD) requires the performance of a Xilinx FPGA to perform some of the more math-intensive functions (such as motion estimation) in conjunction with a programmable video processor. Our DSP capability makes Xilinx the technology of choice for these new demanding applications.

For years the only other solution for these very high performance DSP applications was custom devices – ASICs. Yet ASICs take far longer to design, cost much more to develop, cannot easily be modified to meet changing requirements, and are risky because of their complexity. Xilinx programmable devices and development tools provide a far better solution with less overall cost.

Today, the high-performance FPGA-based DSP market alone is worth more than $200 million, and we have over 80% of that market. According to market estimates, the DSP market addressable by FPGAs is expected to grow to more than $3 billion by 2007. So, as you can see, the future looks very bright for Xilinx as the demand for very high performance DSP continues to grow. We are well positioned to provide the devices, the development tools, and the support services to meet this growing demand.

Embedded Processing

We are relatively new to the embedded processing market – three years ago we introduced our Virtex-II Pro™ family, which includes an embedded hard-core IBM™ PowerPC™ processor. Although it took awhile for the idea to catch on, we now have thousands of design wins using our embedded processors. And in addition to the PowerPC processor, we now offer our 32-bit MicroBlaze™ and the 8-bit PicoBlaze™ soft-core processors. All of these embedded processors work together, using the same peripherals and IP, so you can easily create complete high-performance, multi-processor systems on a single low-cost chip.

The total embedded processor market is very fragmented because there are multiple architectures and multiple operating systems. Customers tend to stay with a known architecture because of their long-term software investment – no one wants to re-code and re-port their designs to a new architecture. That's one reason why we chose the PowerPC as our high-performance processor, because (except for cell phones and video games) it is the one most used in our industry, and it is well supported by both IBM and Motorola. Capturing even a relatively small percentage of this $15 billion market would mean significant revenue for Xilinx. Many embedded processing customers are beginning to realize the benefits of our technology – and we've only started to focus on this market segment.

Because our MicroBlaze and PicoBlaze processors are created as soft cores, they are very flexible and extensible. Plus, they are fast enough to meet the needs of many applications, very inexpensively. Combined with our high-performance PowerPC processor, they form an unbeatable alliance that can handle the most demanding applications with ease, all on a single programmable device.

Our processor strategy is to provide a range of embedded processors, all using the same peripherals and IP, all working together seamlessly on a single chip, and working seamlessly with our DSP and logic functions. Thus you can build and simulate very complex systems and produce production-ready designs faster than ever before. Then, as your requirements change or as design errors are uncovered, you can quickly modify your design and resume production without losing customers. That's the power of programmability; that's what Xilinx does best. The advantages are enormous.

Focusing on the Future

Our original focus was on supporting logic designers – the traditional customers for our devices. However, DSP and embedded processing designers are very different from logic designers; they use different tools, they have different needs and expectations, and they approach their designs in different ways. For example, DSP designers usually work with algorithms such as Fast Fourier Transforms and FIR filters; embedded processing designers work with high-level languages such as C or C++, while logic designers usually work in VHDL or Verilog™. Although the final implementation is in an FPGA, the design approach is very different for each of these customers, and thus we must support these customers in different ways.

Therefore, to ensure that we are addressing the needs of each market segment, I decided to create two new divisions within Xilinx; one to focus on the DSP market and one to focus on the embedded processing market. Each new division, headed by a vice president, will focus on providing the development tools, devices, IP (cores), support services, and marketing functions to fully capitalize on these growing technologies. We intend to be the leader in all of these key programmable technologies.

Conclusion

Xilinx is the only company that can bring all these programmable technologies together in a single device, giving you a tremendous advantage in performance, cost, and time to market. If you do a system-on-chip design in an ASIC, it will require tens of millions of dollars in up-front (NRE) investment, and ASIC designs are risky because you not only have to do the logic design – you must also do the physical design. This can only be justified for high-volume, low-cost applications.

With Xilinx you can do a system-on-chip design with no NRE. And because the chip itself is already designed and debugged, you don't need to worry about physical design issues such as crosstalk and power distribution. All you need to do is develop the logic design, which can be quick and easy using our growing family of IP and development tools that solve many complex design problems for you.

Basically, now we can offer a system-on-chip for the masses, because now we have the advantages of an ASIC in a flexible and programmable device. Now you can create a single chip that includes DSP and embedded processors, along with IP and custom logic, for much less cost and no risk. All these programmable technologies, available on a single device, give you a significant advantage.

FREE TECHNICAL WORKSHOPS! REGISTER NOW AT WWW.XILINX.COM/PW2004

Programmable World 2004

Experience The New Era Of Systems Design

Don't miss the industry's hottest new programmable technology, with in-depth workshops focused on today's most critical design challenges. See the agenda below and register today!

Sessions

Connectivity: Solving high-speed serial design challenges
- Designing Serial Backplanes with Xilinx Solutions
- Multi-Gigabit Serial Interconnect Requirements
- Next-Generation SONET/SDH Networks
- Successful 10 Gbps Backplane Design using Xilinx FPGAs – **Ansoft/Xilinx**
- Enabling the Jump from Megabit to Multi-Gigabit I/O Design using Mentor Graphics Signal Integrity Analysis Solutions – **Mentor Graphics**

DSP: Designing high-performance DSP systems
- The FPGA Platform Radio: The Enabler for B3G and 4G Communication Systems
- Accelerating Productivity through New DSP Design Techniques for FPGAs
- Accelerate High-Performance Real-Time Video & Imaging Applications with an FPGA and a Programmable DSP
- Harness the Power of the Virtex-4™ FPGA XtremeDSP™ Slice and Get the Highest-Performance DSP Functionality

Logic: New tools, techniques and architectural features
- Spartan-3™ Low-Cost Design Techniques
- HDL Coding Techniques to Exploit the New Capabilities of Virtex-4 FPGAs
- Physical Synthesis – **Synplicity**
- Memory Interface Solutions with Virtex-4 FPGAs
- Dramatically Accelerate In-Circuit Debug of FPGA-based Systems – **Agilent**

Processors: A comprehensive embedded solution
- Implementation and Debug of a Dual PowerPC™ System with Floating Point Co-Processor
- FSL-SW Acceleration with MicroBlaze™ or Multi-Channel Data Acquisition
- Accelerating Development of RTOS-based Embedded Systems with Integrated Embedded Tools
- Ethernet Solutions for Xilinx Processors

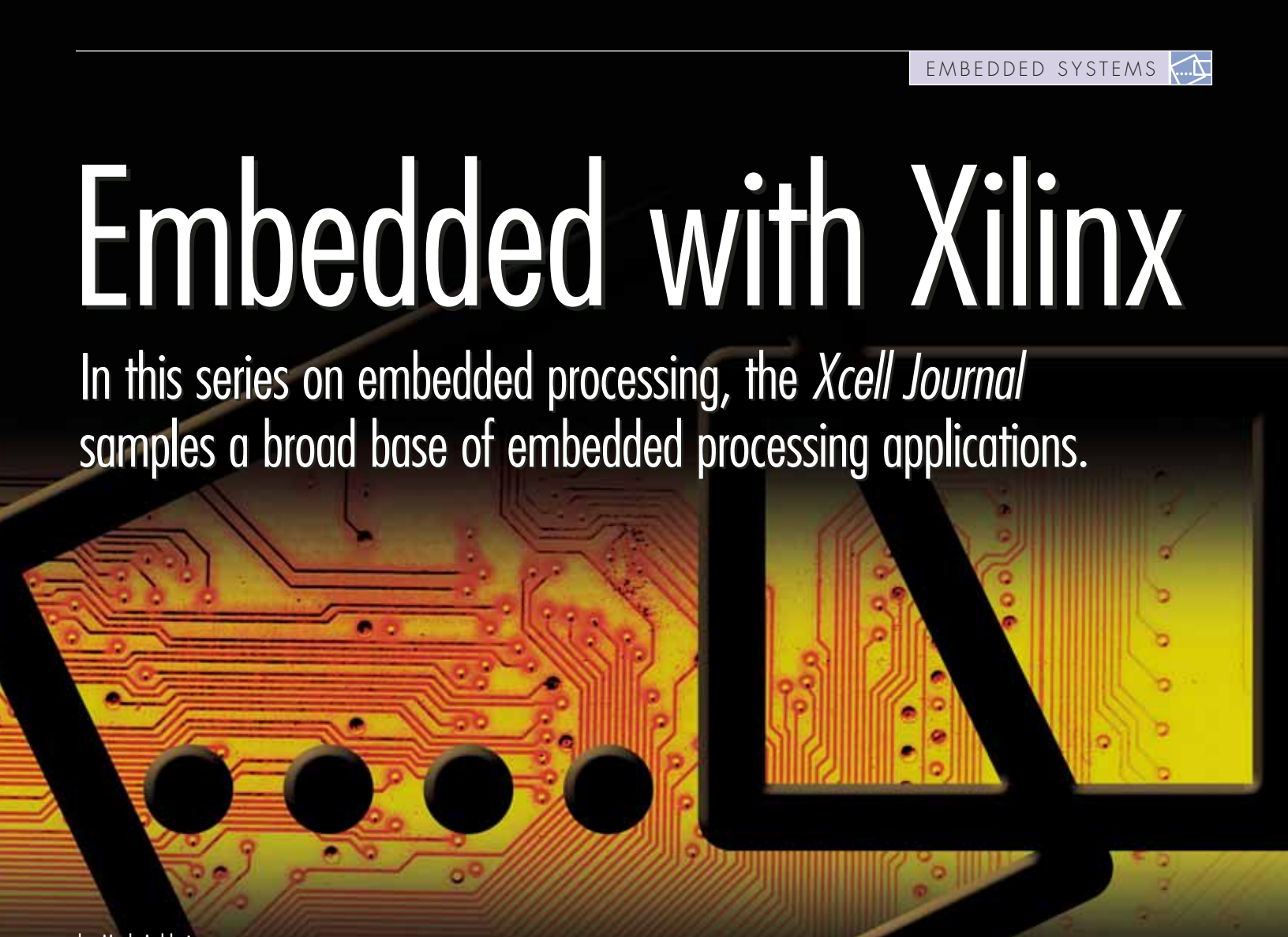
PRESENTED BY:

Mentor Graphics · Synplicity · ANSOFT · XILINX® The Programmable Logic Company™ · Agilent Technologies · Texas Instruments · Reed Electronics Group

Alpha Data · The MathWorks · Memec Insight · HU Horizons · Avnet · Linear Technology · Nallatech · Annapolis Micro Systems, Inc. · CD Technologies · EDN

Embedded with Xilinx

In this series on embedded processing, the *Xcell Journal* samples a broad base of embedded processing applications.



by Mark Aaldering
Vice President, Embedded Processing Division
Xilinx, Inc.
mark.aaldering@xilinx.com

In today's world, just about every system incorporates some form of embedded processing in an amazing array of markets and applications. During the last few years, Xilinx, our partners, and our customers have developed and shared a vision to build and assemble all of the elements required for a complete and robust range of embedded processing solutions adapted for FPGA technologies.

In this edition of the *Xcell Journal*, we have assembled articles representing a wide range of embedded processing applications. These include articles on state-of-the-art commercial applications, real-time operating systems, multi-processor debugging environments, testing of complex hardware modules, and high-speed Internet communication protocols.

With our accelerated success in the embedded processing arena, it is appropriate that this series of articles coincides with the recently announced formation of the Embedded Processing Division. This division brings talent and technology together in an organization to accelerate development of an even wider range of embedded system solutions, optimizing the full capabilities of our silicon architectures at multiple performance and price points.

This new division reinforces our commitment to the increasingly diverse and changing embedded systems market and represents the evolution of three years of embedded processing experience.

Table of Contents

Sign of the Times

Xilinx makes high-tech outdoor advertising in Times Square possible.

by Jason Daughenbaugh
Sr. Design Engineer
Advanced Electronic Designs, Inc.
jason.daughenbaugh@aedmt.com

New York City's Times Square is known as the "Crossroads of the World." Approximately 1.5 million people pass through the intersection of Broadway and 42nd Street every day, and millions more see the area daily on television broadcasts. No better place for outdoor advertising exists. As a result, dazzling signs have become a Times Square trademark.

Every advertiser wants to have the best advertising medium possible, so new signs must use the latest technology. Times Square tenants rely on MultiMedia, which manufactures the majority of the spectacular signs in Times Square. When MultiMedia asked our company, Advanced Electronic Designs, Inc. (AED), to design an LED sign for JPMorgan Chase™ in Times Square, we needed a huge amount of signal processing, data distribution, and interfacing. We also needed to design the sign very quickly. We met this challenge by utilizing the advantages of Xilinx® components.

We used Virtex-II™ XC2V1000 FPGAs for video processing, and for control and distribution we chose low-cost Spartan-3™ XC3S200 FPGAs. To configure the FPGAs, we chose the Platform Flash XCF00 configuration PROM family. And for final distribution of the data on the 3800 LED blocks, we used XC9572XL PLDs.

The Design

An LED sign is like a large computer monitor; video data goes in and is displayed on the sign. The sign comprises red, green, and blue LEDs that turn on and off (pulse-width modulation) to generate more than four trillion colors.

What made this particular design a challenge was the scale, both in terms of physical size as well as the amount of data and the transfer rates involved. The sign is 135 feet long and 26 feet tall. With nearly two million pixels, it is the highest definition LED display in the world. This is ten times the resolution of the average television screen and twice the resolution of top-of-the-line HDTV sets.

After considering our options, designing with Xilinx programmable logic was the obvious choice. The high-performance, low-cost FPGAs are well suited for all three main components of this design: video processing, data distribution, and sign control.

Figure 1 – The world's highest resolution LED display is based on Xilinx devices.

Video Processing

The video processor accepts a variety of video inputs. It captures these video streams as 36 bit RGB (12 bits per color). It then crops and places these inputs onto a master sign image for display. Color-space conversion adjusts image characteristics such as color temperature and balance. Additional processing corrects for individual LED differences. We also use proprietary image processing algorithms to operate the LEDs efficiently while maintaining optimal image quality.

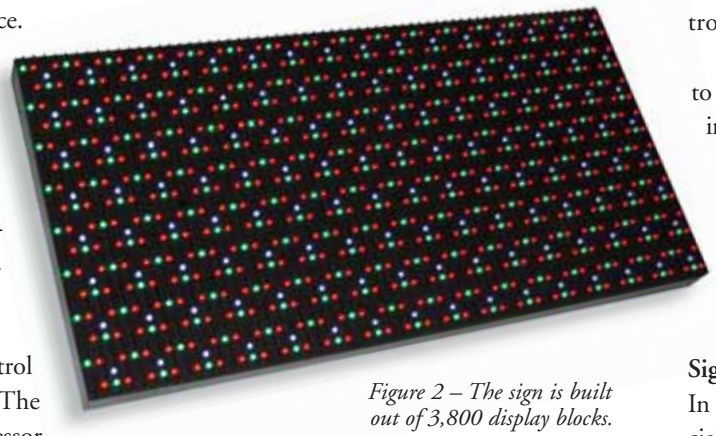


Figure 2 – The sign is built out of 3,800 display blocks.

Data Distribution

Video data starts in a control room and ends at the LEDs. The first step is the video processor, which is located in the control room. The video processor breaks the images into manageable chunks to send to the many modules of the sign so that each LED displays the data for the corresponding pixel. More than 3 Gbps of video data alone is required to operate the LEDs. In addition to video data, we also transfer a variety of control and status functions.

Not wanting to re-invent the wheel, we chose Ethernet as our data distribution medium. Our video processor has multiple Gigabit Ethernet ports that interface to the sign. Gigabit Ethernet can be transferred over fiber-optic cable, allowing great distances between the controller and the sign itself.

We were able to use off-the-shelf switches to distribute the data within the sign and put inexpensive 10/100 Ethernet ports on the individual distribution boards. The availability of Ethernet protocol analyzers, such as the open-source project Ethereal, allowed us to easily analyze and debug the system.

Sign Control

In advertising, time is money; thus it is crucial to monitor the sign at all times. The control system monitors temperatures throughout the sign to ensure that adequate cooling is present. Voltages are monitored to detect malfunctioning power supplies. The control system maintains error and resend counts to detect faulty data links. It also provides an interface to upgrade the FPGAs remotely for enhancements and bug fixes.

The reconfigurable nature of Xilinx FPGAs allows us to provide feature upgrades and bug fixes to the customer via e-mail…

The Benefits of Xilinx Devices

Xilinx devices include a large number of features that are ideal for our sign project:

- The reconfigurable nature of Xilinx devices is necessary for a project like this. Without FPGAs, the only alternative would have been an ASIC. But an ASIC was not feasible for this project for several reasons.

  First, this project had a very tight schedule. An ASIC could not have been completed in the time allotted. Second, the volumes of the components in this sign are not of sufficient volume to hide the NREs of an ASIC. Third, an ASIC lacks the development opportunities of an FPGA. To me, as an engineer, this reason is the most important. No matter how much simulation you perform, there can always be unexpected bugs. In an ASIC, these bugs are expensive; in an FPGA, they can be fixed easily.

  Another FPGA advantage is that it can meet future needs through feature upgrades; an ASIC cannot. The reconfigurable nature of Xilinx FPGAs allows us to provide feature upgrades and bug fixes to the customer via e-mail, making it easy for them to apply to the sign. Through an Ethernet interface, the FPGA reprograms the Platform Flash configuration PROM and automatically reboots.

- Video processing requires a large number of multiply operations. The video processor must perform color-space conversion and apply calibration coefficients in real time. It would require a large portion of FPGA logic resources to build multipliers. Instead, this can be done very efficiently by utilizing the embedded multipliers. Building pipelined processing structures with the embedded multipliers allowed us to easily meet the processing requirements.

- This design required a large variety of signaling standards. The flexible Xilinx I/O blocks allowed us to connect directly to a large number of different interfaces. Voltages ranged from standard 3.3V CMOS down to 1.5V HSTL. We required single-ended and differential interfaces. In some cases we could have used external driver and receiver parts, but that would have added complexity and cost to the product.

  Other high-speed I/O interfaces, such as to the DDR 333 memory, would not have been possible without direct FPGA support. The digitally controlled impedance (DCI) modes were necessary on the high-speed single-ended traces.

- With the high data rates involved and the many data interfaces, we had a large number of clock domains. The quantity of global clock nets available and the ability of the digital clock managers (DCMs) to synthesize clock frequencies made this easy. We also used the phase-shift ability of the DCM to adjust sample times on various interfaces.

- Block RAM is my favorite resource in an FPGA. Without block RAM, there are two memory options. The first option is the logic slices, using flip-flops or distributed RAM, but this is expensive and slow for anything more than 16- to 32-bit addresses. The second option is external memory, such as SDRAM. SDRAM storage is generally in the range of tens to hundreds of megabytes, leaving a huge size gap between these two memory options.

  Block RAM bridges this size gap. It can be used for a limitless number of things, from FIFOs for processing engines to loadable tables for data conversions. The flexible port-widths of block RAM allow you to use them individually or in efficient combinations. The dual-port capability makes them easy to use for transferring data between clock domains or sharing data.

- While very powerful and convenient, the PLDs and Spartan-3 FPGAs are also very inexpensive. When combined with the development advantages, the low device price makes Xilinx devices unbeatable when developing high-performance embedded systems.

PicoBlaze Processors

Device hardware capabilities are essential for any design, but development tools and tricks are also very important. The favorite toy in our Xilinx bag-of-tricks is the PicoBlaze™ processor. We could not have completed the project in the time allowed without extensive use of the PicoBlaze processor. The sign contains an impressive count of more than 1,000 of these embedded processors, with nine different designs.

PicoBlaze processors provide efficient logic resource utilization by time-multiplexing logic circuits. Many functions, especially control functions, do not need to be

10	XC2V1000 Virtex-II FPGA
323	XC3S200 Spartan-3 FPGA
333	XCF00 Platform Flash PROM
3,800	XC9572XL 72 macrocell PLD

Table 1 – This sign includes nearly 4,500 Xilinx devices.

8	Gbps video processing
18	Billion 16-bit multiply operations per second
16	DDR333 SDRAM banks
6	Gigabit Ethernet MACs
333	Fast Ethernet MACs
>1000	PicoBlaze Processors

Table 2 – Xilinx devices achieve impressive specifications.

especially fast, or don't happen very often.

One example of this would be a serial transfer to read a temperature sensor. For this application, the sensor only needs to be read every ten seconds. It would be a waste to have a state machine for temperature sensor reading that ran once every ten seconds but only took a few milliseconds to complete. The logic would be unused 99.9% of the time.

These types of functions can be efficiently combined into a single PicoBlaze processor, which in the previous example can not only read the temperature every ten seconds but perform other similar tasks in the meantime.

The PicoBlaze processor also provides a quick and easy way to develop control functions. The alternative would be to build a custom state machine for each function. The PicoBlaze processor is a programmable state machine, meaning that the state machine is already built; one just has to program it. It has an intuitive and powerful instruction set and a large codespace of 1,024 instructions. Programming the PicoBlaze processor is a quick and easy way to define many functions.

The PicoBlaze processor is also a great tool for accelerating the testing and debugging process. The PicoBlaze program code is stored in block RAM. To make a change to the program we only need to change the block RAM contents. It is possible to do this without re-implementing the FPGA, saving a lot of time.

Our favorite method of PicoBlaze processor development, which is slightly unique, is to use a PC serial port and a simple PC application to download the program code into the block RAM of a configured FPGA. We have developed an interface board that connects to the FPGA and has the serial port, as well as several seven-segment displays to which the PicoBlaze processor can write for debugging. We also allow the selection of different processors so that we can work on multiple processors through the same interface.

This interface is not only useful for debugging PicoBlaze programs, but also for debugging the logic connected to the processor. Because it is so quick and easy to write programs for PicoBlaze processors, it is very straightforward to write programs to test the various logic circuits attached to the processor. We can test each function individually, greatly simplifying and accelerating any debugging that becomes necessary.

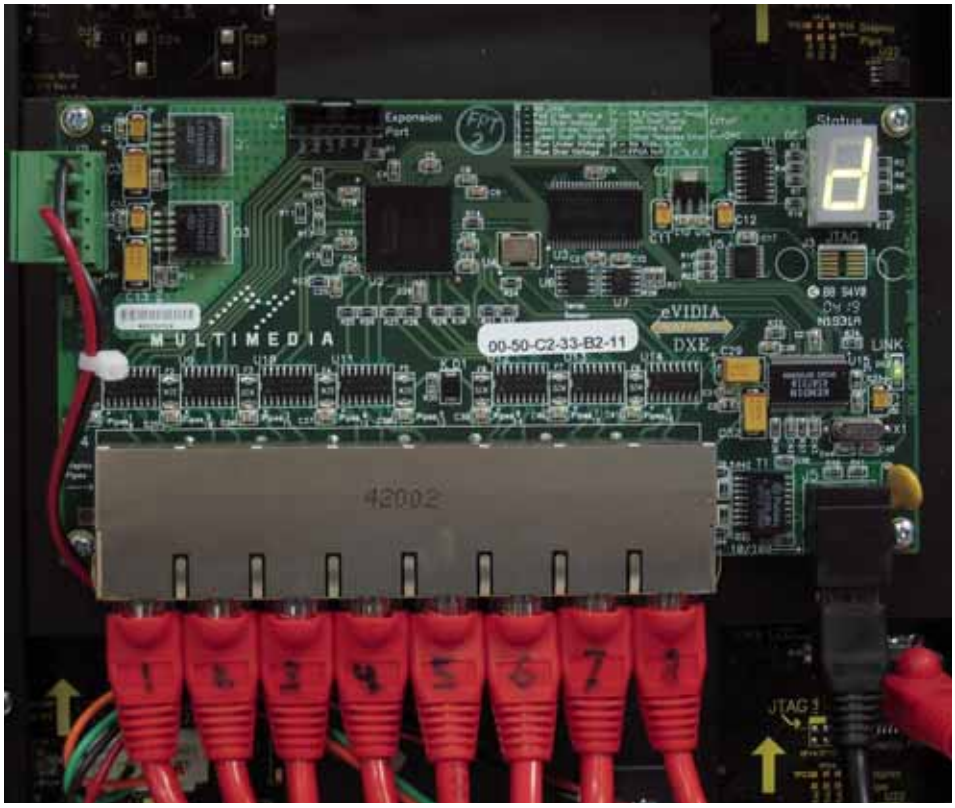


Figure 3 – The video data distribution board is based on an XC3S200 FPGA. It also includes SRAM, a 10/100 Ethernet port, a status display, and numerous connections to display blocks.

A key application of the PicoBlaze processor in this project is the Ethernet controller. As mentioned earlier, we selected Ethernet to distribute data throughout the sign. At each Ethernet connection, we have an Ethernet physical layer transceiver (PHY) device connected directly to an FPGA. We developed a very simple and tiny media access controller (MAC) module, which we use inside the FPGA to connect the PHY to an instantiation of the PicoBlaze processor.

This Ethernet unit is small, requiring less than a quarter of the logic resources in the XC3S200 FPGA. It handles the basic Ethernet layers and protocols, including ARP (address resolution protocol). It also supports the IP (Internet protocol) layer with ICMP (Internet control message protocol), UDP (user datagram protocol), and DHCP (dynamic host configuration protocol). With this Ethernet controller, we can plug an FPGA into our network and it negotiates an IP address. Then we can transfer files and data to and from it.

Conclusion

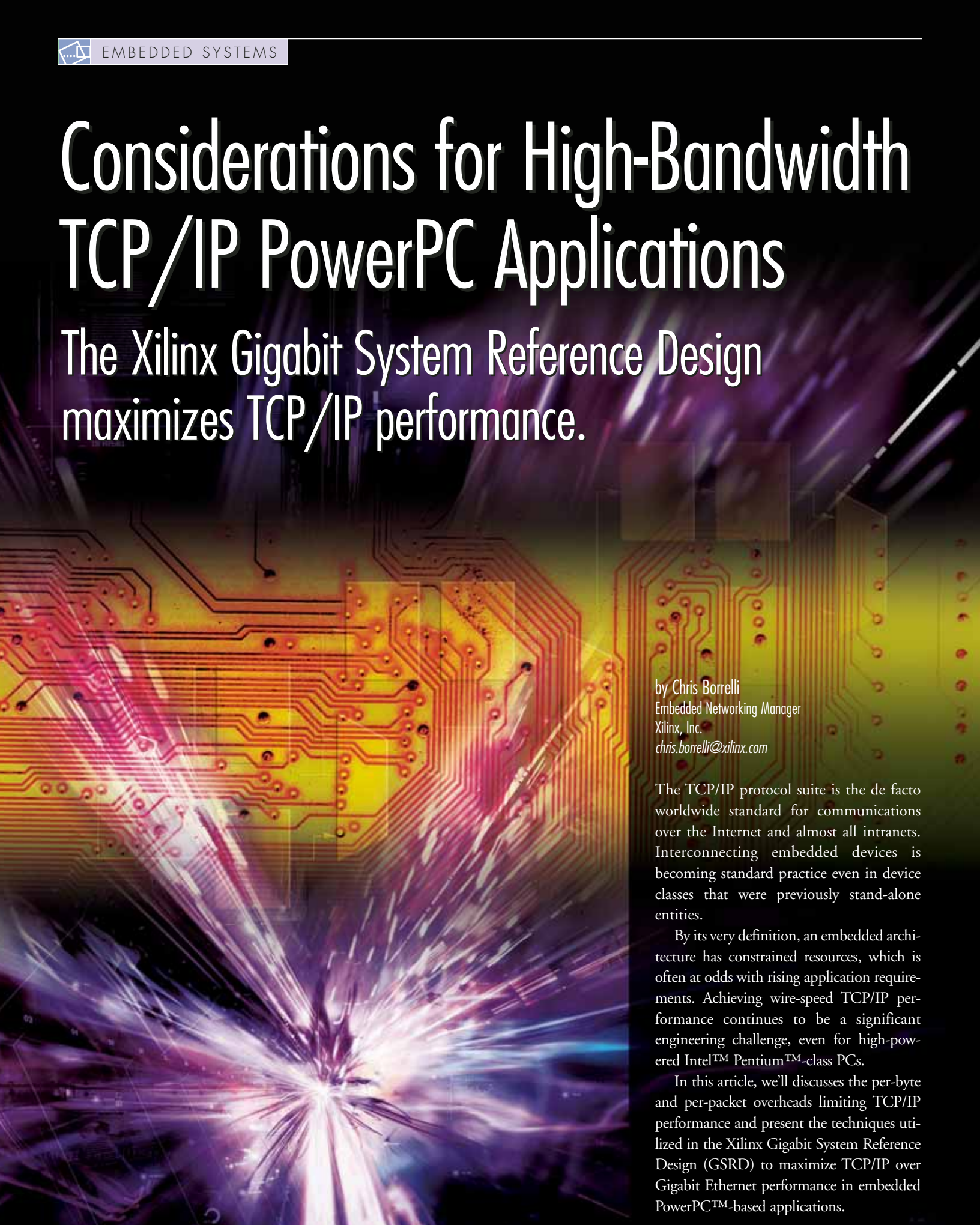
Xilinx devices made the challenge of developing the world's highest definition LED display achievable. These devices are a perfect fit for a complex design because of their flexible nature and powerful feature set. Valuable design components such as the PicoBlaze processor further increase their ease of use and thus their value.

The reconfigurable and flexible nature of the devices allowed us to ship the sign with all first-revision circuit boards, enabling us to develop a very complex system in very little time.

For more information about MultiMedia LED signs, visit *www.multimediaLED.com*. For more information about the engineering provided by AED, visit *www.aedmt.com*.

Considerations for High-Bandwidth TCP/IP PowerPC Applications

The Xilinx Gigabit System Reference Design maximizes TCP/IP performance.



by Chris Borrelli
Embedded Networking Manager
Xilinx, Inc.
chris.borrelli@xilinx.com

The TCP/IP protocol suite is the de facto worldwide standard for communications over the Internet and almost all intranets. Interconnecting embedded devices is becoming standard practice even in device classes that were previously stand-alone entities.

By its very definition, an embedded architecture has constrained resources, which is often at odds with rising application requirements. Achieving wire-speed TCP/IP performance continues to be a significant engineering challenge, even for high-powered Intel™ Pentium™-class PCs.

In this article, we'll discusses the per-byte and per-packet overheads limiting TCP/IP performance and present the techniques utilized in the Xilinx Gigabit System Reference Design (GSRD) to maximize TCP/IP over Gigabit Ethernet performance in embedded PowerPC™-based applications.

GSRD Overview

The GSRD terminates IP-based transport protocols such as TCP or UDP. It incorporates the embedded PowerPC and RocketIO™ blocks of the Virtex-II Pro™ device family, and is delivered as an Embedded Development Kit (EDK) reference system.

The reference system as described in Xilinx Application Note XAPP536 leverages a multi-port DDR SDRAM memory controller to allocate memory bandwidth between the PowerPC processor local bus (PLB) interfaces and two data ports. Each data port is attached to a direct memory access (DMA) controller, allowing hardware peripherals high-bandwidth access to memory.

A MontaVista™ Linux™ port is available for applications requiring an embedded operating system, while a commercial standalone TCP/IP stack from Treck™ is also available to satisfy applications with the highest bandwidth requirements.

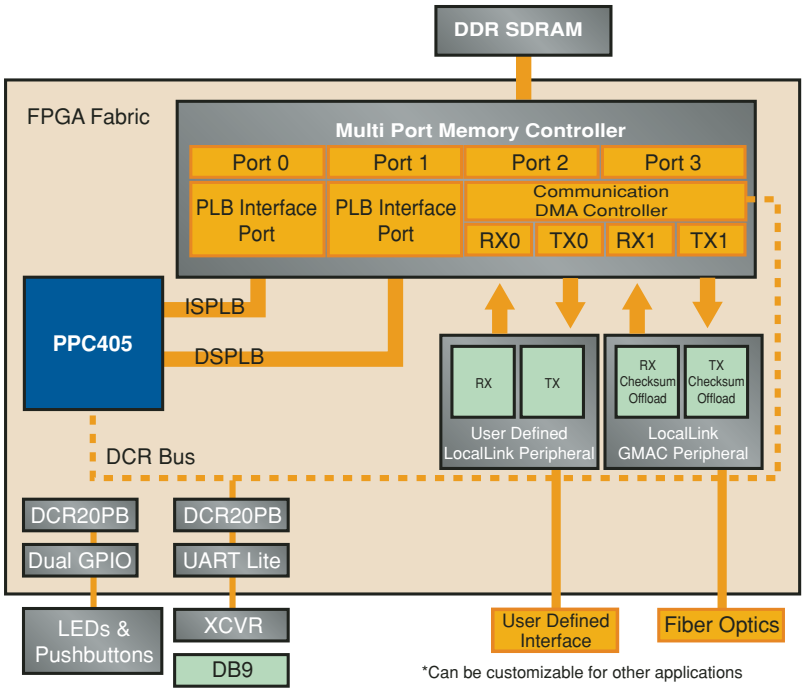


Figure 1 – GSRD system block diagram

System Architecture

Memory bandwidth is an important consideration for high-performance network-attached applications. Typically, external DDR memory is shared between the processor and one or more high-bandwidth peripherals such as Gigabit Ethernet.

The four-port multi-port memory controller (MPMC) efficiently divides the available memory bandwidth between the PowerPC's instruction/data PLB interfaces and a communications direct memory access controller (CDMAC). The CDMAC provides two bi-directional channels of DMA that connect to peripherals through a Xilinx standard LocalLink streaming interface. The CDMAC implements data re-alignment to support arbitrary alignment of packet buffers in memory. A block diagram of the system is shown in Figure 1.

The LocalLink Gigabit Ethernet MAC (LLGMAC) peripheral incorporates the UNH-tested Xilinx LogiCORE™ 1-Gigabit Ethernet MAC to provide a 1 Gbps 1000-BASE-X Ethernet interface to the reference system. The LLGMAC implements checksum offload on both the transmit and receive paths for optimal TCP performance. Figure 2 is a simplified block diagram of the peripheral.

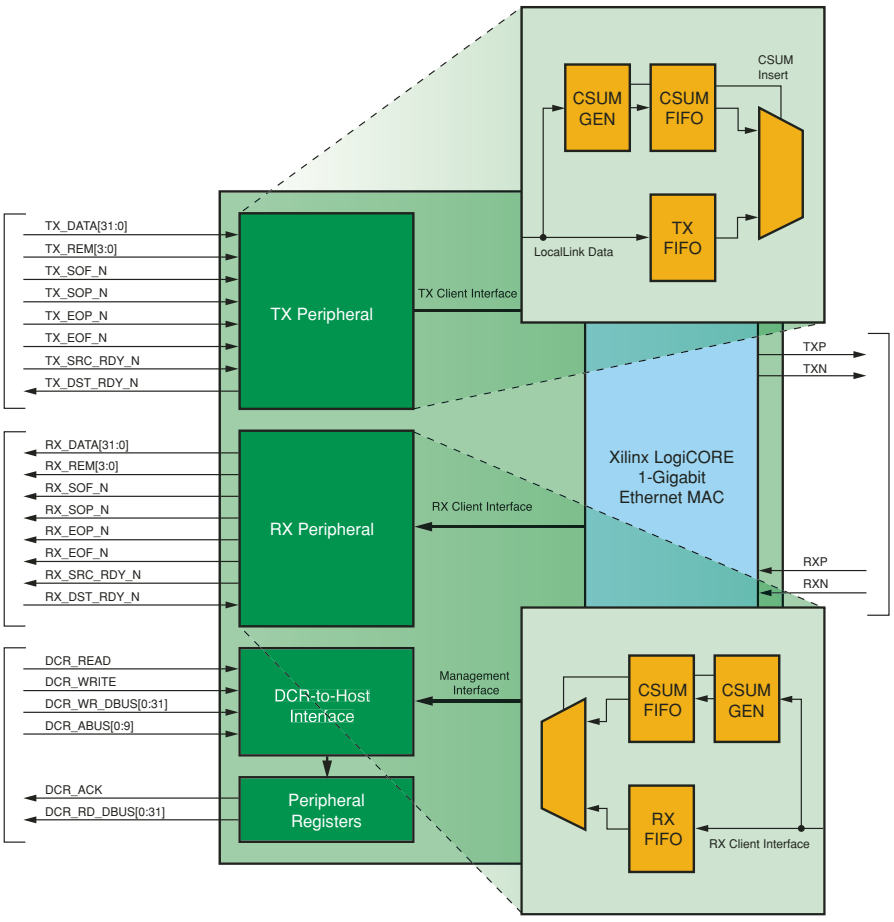


Figure 2 – LocalLink Gigabit Ethernet MAC peripheral block diagram

TCP/IP Per-Byte Overhead

Per-byte overhead occurs when the processor touches payload data. The two most common operations of this type are buffer copies and TCP checksum calculation. Buffer copies represent a significant overhead for two reasons:

1. Most of the copies are unnecessary.
2. The processor is not an efficient data mover.

TCP checksum calculation is also expensive, as it is calculated over each payload data byte.

Embedded TCP/IP-enabled applications such as medical imaging require near wire-speed TCP bandwidth to reliably transfer image data over a Gigabit Ethernet network. The data is generated from a high-resolution image source, not the processor.

In this case, introducing a zero-copy software API and offloading the checksum calculation into FPGA fabric completely removes the per-byte overheads. "Zero-copy" is a term that describes a TCP software interface where no buffer copies occur. Linux and other operating systems have introduced software interfaces like sendfile() that serve this purpose, and commercial standalone TCP/IP stack vendors like Treck offer similar zero-copy features. These software features allow the removal of buffer copies between the user application and the TCP/IP stack or operating system.

The data re-alignment and the checksum offload features of GSRD provide the hardware support necessary for zero-copy functionality. The data re-alignment feature is a flexibility of the CDMAC that allows software buffers to be located at any byte offset. This removes the need for the processor to copy unaligned buffers.

Checksum offload is a feature of the LocalLink Gigabit Ethernet (LLGMAC) peripheral. It allows the TCP payload checksum to be calculated in FPGA fabric as Ethernet frames are transferred between main memory and the peripheral's hardware FIFOs. GSRD removes the need for costly buffer copies and processor checksum operations, leaving the PowerPC 405 to process only protocol headers.

TCP/IP Per-Packet Overhead

Per-packet overhead is associated with operations surrounding the transmission or reception of packets. Packet interrupts, hardware interfacing, and header processing are examples of per-packet overheads.

Interrupt overhead represents a considerable burden on the processor and memory subsystem, especially when small packets are transferred. Interrupt moderation (coalescing) is a technique used in GSRD to alleviate some of this pressure by amortizing the interrupt overhead across multiple packets. The DMA engine waits until there are n frames to process before interrupting the processor, where n is a software-tunable value.

Transferring larger sized packets (jumbo frames of 9,000 bytes) has a similar effect by reducing the number of frames transmitted, and therefore the number of interrupts generated. This amortizes the per-packet overhead over a larger data payload. GSRD supports the use of Ethernet jumbo frames.

The components of GSRD use the device control register (DCR) bus for control and status. This provides a clean interface to software without interfering with the high-bandwidth data ports. The per-packet features of GSRD help make efficient use of the processor and improve system-level TCP/IP performance.

TCP/IP Stack	Ethernet Frame Size	Optimization	TCP Transmit Bandwidth
MontaVista Linux	9000 bytes (jumbo)	None	270 Mbps
MontaVista Linux	9000 bytes (jumbo)	Zero-copy, checksum offload	540 Mbps
Treck, Inc	9000 bytes (jumbo)	Zero-copy	490 Mbps
Treck, Inc	9000 bytes (jumbo)	Zero-copy, checksum offload	780 Mbps

Table 1 – TCP transmit benchmark results

Conclusion

The Xilinx GSRD is an EDK-based reference system geared toward high-performance bridging between TCP/IP-based protocols and user data interfaces like high-resolution image capture or Fibre Channel. The components of GSRD contain features to address the per-byte and per-packet overheads of a TCP/IP system.

Table 1 details the GSRD TCP transmit performance with varying levels of optimization for Linux and standalone Treck stacks.

Future releases of GSRD will explore further opportunities for TCP acceleration using the FPGA fabric to offload functions such as TCP segmentation.

The GSRD Verilog™ source code is available as part of Xilinx Application Note XAPP536. It leverages the MPMC and CDMAC detailed in Xilinx Application Note XAPP535 to allocate memory bandwidth between the processor and the LocalLink Gigabit Ethernet MAC peripheral. The MPMC and CDMAC can be leveraged for PowerPC-based embedded applications where high-bandwidth access to DDR SDRAM memory is required.

For more information about XAPP536 and XAPP535, visit *www.xilinx.com/gsrd/*.

Associated Links:

Xilinx XAPP536, "Gigabit System Reference Design"
http://direct.xilinx.com/bvdocs/appnotes/xapp536.pdf

Xilinx XAPP535, "High Performance Multi Port Memory Controller"
http://direct.xilinx.com/bvdocs/appnotes/xapp535.pdf

Treck, Inc. (*www.treck.com*)

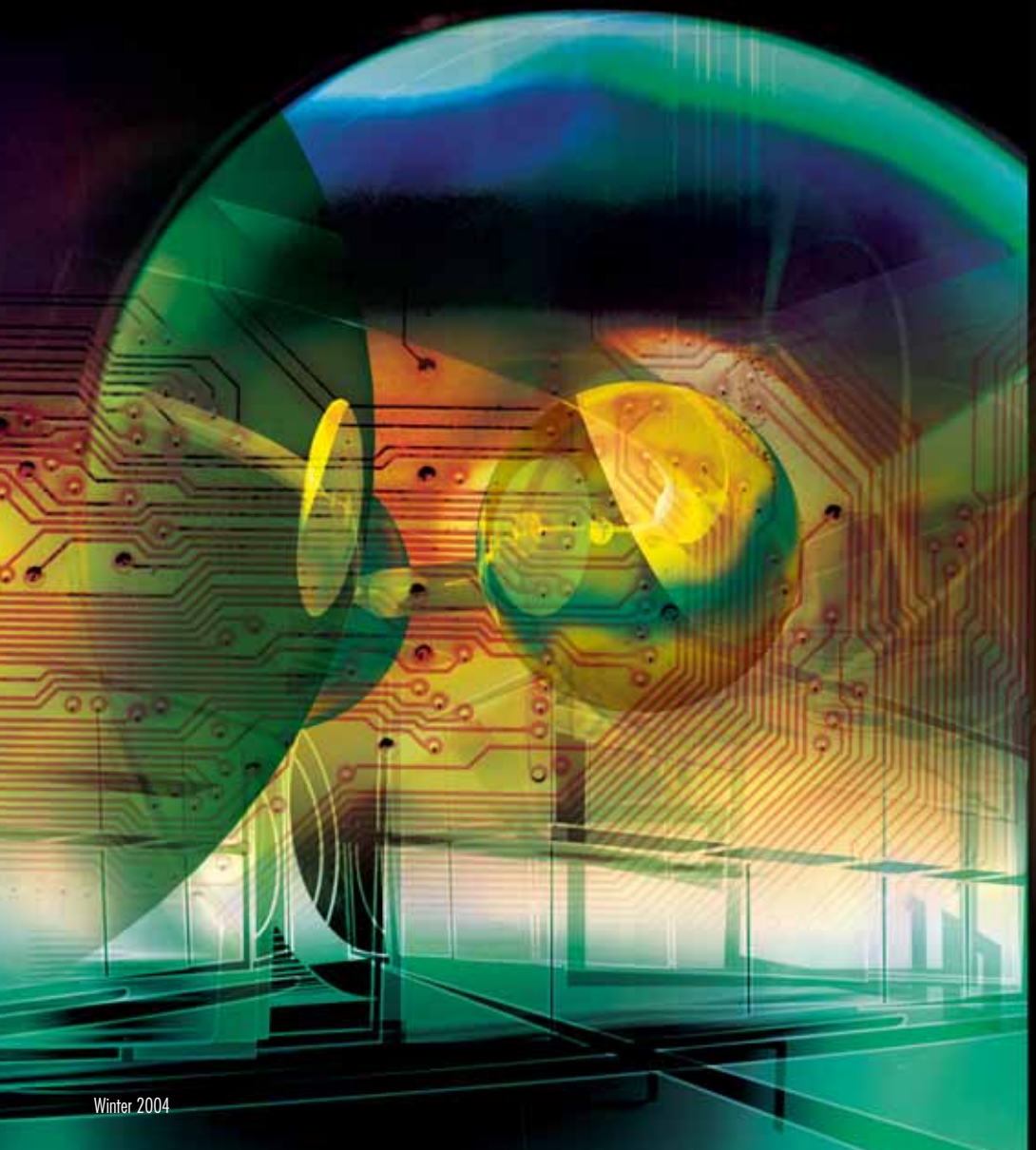
MontaVista Software (*www.mvista.com*)

"End-System Optimizations for High-Speed TCP" (*www.cs.duke.edu/ari/publications/end-system.pdf*)

"Use sendfile to optimize data transfer" (*http://builder.com.com/5100-6372-1044112.html*)

Configure and Build the Embedded Nucleus PLUS RTOS Using Xilinx EDK

Nucleus PLUS RTOS for MicroBlaze and PowerPC 405 processors is now automatically configurable using XPS MLD technology.



by Gordon Cameron
Business Development Manager
Accelerated Technology, Mentor Graphics
gordon_cameron@mentor.com

Accelerated Technology's Nucleus™ PLUS real-time operating system (RTOS) is already available for both the Xilinx® MicroBlaze™ 32-bit soft processor core and the IBM™ PowerPC™ 405 core integrated into Virtex-II Pro™ devices. This deterministic, fast, small footprint RTOS is ideal for "hard" real-time applications.

With the release of the Xilinx Platform Studio EDK 6.3i, configuration of this leading royalty-free RTOS on your newly designed system is as easy as selecting from a pull-down menu. Instead of spending hours modifying your target software to work with your new hardware configuration, you can configure the target software automatically in minutes, without the error-prone possibilities of configuring by hand. This is especially valuable during the earlier design phases when the hardware may be changing frequently. This process was enabled by one of the underlying technologies of Xilinx Platform Studio EDK, called microprocessor library definition, or MLD.

MLD Technology

The Xilinx Platform Studio EDK development system is based on a data-driven code base that makes it extensible and open. MLD is one example of this underlying capability. It was created specifically to allow you to easily create and modify kernel configurations and associated board support packages (BSPs) for partner-supported RTOSs like Nucleus PLUS and its extensive middleware offering.

MLD has two required file types: the data definition file (.MLD) and data generation file (.Tcl). The .MLD contains the Nucleus user-customization parameters, while the .Tcl file is a Tcl script that defines a set of Nucleus-specific procedures for building the final software system (see Figure 1).

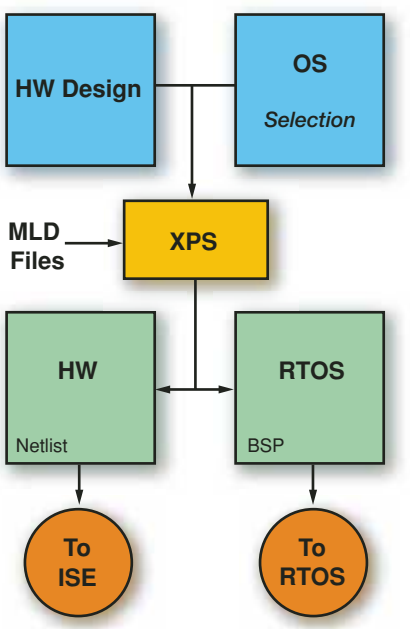


Figure 1 – Outline of an MLD-enabled system design

Installing MLD files in XPS

The installation CD of Nucleus PLUS installs the RTOS, associated drivers, and the two Nucleus-configured MLD files that enable you to use MLD technology within the Xilinx Platform Studio EDK. The default install path for the MLD files is the *\nucleus\bsp* sub-directory, located in *\edk_user_repository*.

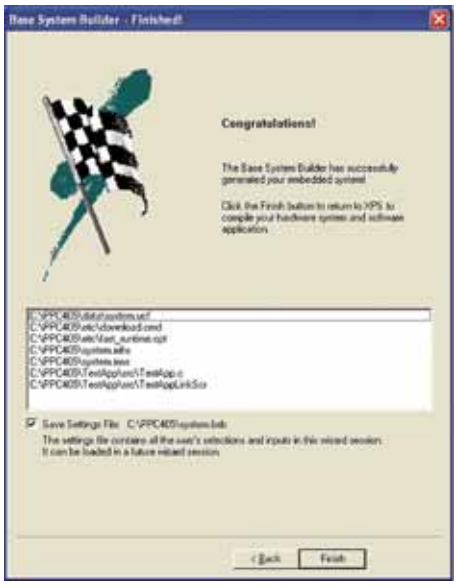


Figure 2 – The hardware design is complete and ready to configure the software.

Accelerated Technology supplies these files and the associated installer as an evaluation disk, included in the latest release of Xilinx EDK 6.3i. Accelerated Technology has also established a website to support and distribute this evaluation. This site contains updates, evaluations, reference designs, and documentation for all of the Accelerated Technology Xilinx offerings and will be updated regularly with new middleware implementations that you can add to the automatic configuration of your application. The website is located at *www.acceleratedtechnology.com/xilinx/*.

To get up and running quickly with your first Nucleus-based system, the installation also includes a sample pre-built reference design with a compiled Nucleus PLUS demonstration. The pre-built reference designs currently support the Memec™ design-based DS-KIT-2VP7FG456 and DS-KIT-V2MB1000 FPGAs. This is the fastest method to employ for a sample Nucleus-based, MLD-enabled Xilinx system.

Use of Xilinx's Base System Builder is also well documented inside the application notes accompanying the installation. With the Base System Builder, you can build a variety of system core configurations to work with the Nucleus PLUS RTOS (see Figure 2).

If you have received your EDK 6.3i update recently or have purchased a seat, please check the contents for this evaluation. After running the Nucleus PLUS evaluation disk installer, the necessary files will be placed into the Xilinx EDK 6.3i and the support of Nucleus PLUS will be automatically added.

The elements of Nucleus PLUS modified by the data generation file (.Tcl) for specific hardware configuration are:

- The number and type of devices used by the hardware designer
- Memory map information
- Locations of memory-mapped device registers
- Timer configuration
- Interrupt controller configuration

Once you have installed these, you can use Xilinx Platform Studio EDK with Nucleus now visible in the RTOS pull-down selection menu. See Figure 3a for the PPC405 and Figure 3b for the MicroBlaze processor.

Evaluating Nucleus PLUS in EDK

The Accelerated Technology Nucleus PLUS evaluation software provided in the EDK Platform Studio 6.3i shipment includes a limited version (LV) of Nucleus PLUS. This is a fully functional version of the RTOS compiled into a library format (rather than the normal source code distribution) with the single restriction that it will stop working after 60 minutes, facilitating evaluation of its full functionality. When you purchase a full license of Nucleus PLUS from Accelerated Technology, you receive the full source code and, obviously, the 60-minute run time restriction is lifted.

The LV version of Nucleus PLUS is configured to execute from the off-chip SRAM or SDRAM module. Once you have a full license to the RTOS, you can configure it to run from any memory in your system.

Nucleus PLUS is a scalable RTOS – only the software you use in your design is included in the downloaded code. This may be contrasted with other larger, more static systems, which consume far more system resources. In some circumstances, the whole RTOS and application can fit in the on-chip memory, thus achieving high performance and low power consumption. Even with larger applications, which may utilize extensive middleware, the efficient use of the relatively small amount of on-chip memory means that the size of the kernel footprint is an important consideration.

You can configure other components of the Nucleus system by hand to work in this environment, such as networking, web server, graphics, file management, USB, WiFi, and CAN bus. Future releases of Nucleus will move these products into full integration with Xilinx Platform Studio EDK and MLD technology.

Nucleus PLUS and Xilinx Devices

As we have said, the process of creating a working BSP for Nucleus PLUS begins with configuring the hardware platform using Xilinx Base System Builder or the supplied sample reference designs. Then you can go to the Project > Software Platform Settings menu item and select the operating system you want to use from the list. Choosing the Nucleus option (Figure 3 a/b) will make available the specific software settings for the RTOS under each of the tabs on the Software Platform Settings menu (Figure 4 shows the user enabling the cache on the PowerPC). Note that in the LV version most of the software options are disabled, but can be changed in the full version.

Once you are satisfied with the software settings, you can use the Generate Netlist and Generate Bitstream commands and download the hardware configuration onto the FPGA using Xilinx XPS or ISE tools.

You can now execute the Tools > Generate Libraries and BSP commands to configure Nucleus PLUS. The application software can be linked with the RTOS. Now you are ready to switch over to the GDB debugger and download the combined RTOS and application image to the FPGA.

Advanced Software Tools

Up to this point, we have bypassed many aspects of application software design, assuming that you have code ready to compile and link and download to the FPGA. In fact, as systems become ever more complex, both hardware and software designers require advanced state-of-the-art tools to help them complete their projects within budget and on time.

The Xilinx EDK-configurable version of Nucleus PLUS uses the standard GNU suite of tools supplied with the Xilinx EDK package. This is more than adequate for many projects for getting systems up and running, but advanced application development often needs more. Accelerated Technology can provide a complete range of tools that encompass all phases of the software design process.

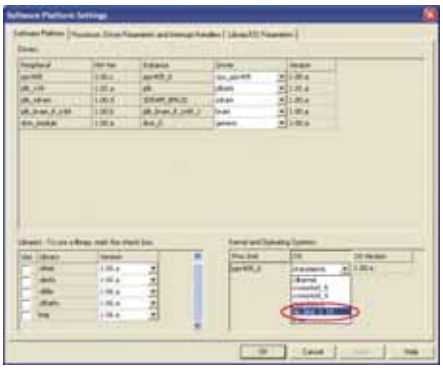


Figure 3a – After installing Nucleus PLUS in EDK, Nucleus appears as an option in the drop-down menu choosing which operating system to use with the PowerPC 405 processor.

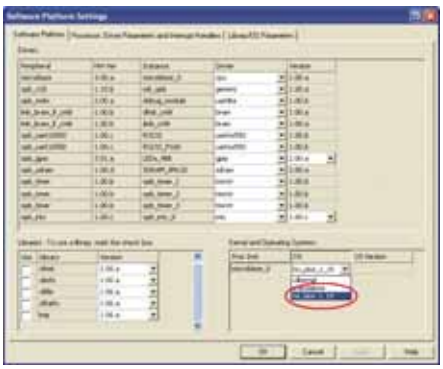


Figure 3b – Nucleus appears as an option in the drop-down menu choosing which operating system to use with the MicroBlaze soft-core processor.

- If code footprint or performance is important, then consider the highly optimizing Microtec compiler for PowerPC Virtex-II Pro devices. This ensures that the code that is shipped is the same as the code that is debugged – a goal not achieved by many compilers.
- Application debugging often needs RTOS awareness, advanced breakpoints, and debugging of fully optimized code. These features are available on PowerPC Virtex-II Pro devices with the industry-standard XRAY debugger.
- To bring software development forward in time so that it can be started before the hardware is complete, software teams can use our advanced prototyping products Nucleus SIM or Nucleus SIMdx. These tools allow the development of the complete application software in a host-based environment.
- UML enables software teams to raise their level of abstraction and produce models of their software. Nucleus BridgePoint enables full code generation by using the xtUML subset of UML 2.0.
- You can verify software/hardware interaction in the Mentor Graphics® Seamless® co-verification environment, which allows combined hardware and software simulation for PowerPC Virtex-II Pro devices.

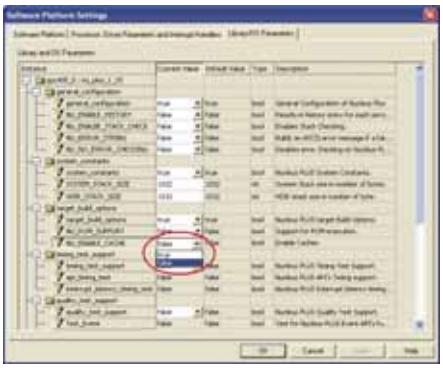


Figure 4 – Enabling the cache in the RTOS configuration parameters

These tools, when combined with the Nucleus PLUS RTOS, are ideal for helping you maximize the functionality and efficiency of your designs.

Conclusion

The latest EDK-configurable Nucleus PLUS RTOS brings a new dimension to systems incorporating high-performance embedded processors from Xilinx. Its small size means that it can use available on-chip memory to minimize power dissipation and deliver increased performance, while its wealth of middleware makes it ideal for products targeted at the networking, telecommunications, data, communication, and consumer markets.

Making this solution easy to configure within Xilinx EDK allows you to easily exploit the benefits of this powerful product. For more information, visit *www.acceleratedtechnology.com* or *www.mentor.com.*

MicroBlaze and PowerPC Cores as Hardware Test Generators

Combining FPGA embedded processors with C-to-RTL compilation can accelerate the testing of complex hardware modules.

by David Pellerin
CTO
Impulse Accelerated Technologies
david.pellerin@impulsec.com

Milan Saini
Technical Marketing Manager
Xilinx, Inc.
milan.saini@xilinx.com

Regardless of whether you are using a processor core in your FPGA design, using a Xilinx® MicroBlaze™ or IBM™ PowerPC™ embedded processor can accelerate unit testing and debugging of many types of FPGA-based application components.

C code running on an embedded processor can act as an in-system software/hardware test bench, providing test inputs to the FPGA, validating the results, and obtaining performance numbers. In this role, the embedded processor acts as a vehicle for in-system FPGA verification and as a complement to hardware simulation.

By extending this approach to include not only C compilation to the embedded processor but C-to-hardware compilation as well, it is possible – with minimal effort – to create high-performance, mixed software/hardware test benches that closely model real-world conditions.

Key to this approach are high-performance standardized interfaces between test software (C-language test benches) running on the embedded processor and other components (including the hardware under test) implemented in the FPGA fabric. These interfaces take advantage of communication channels available in the target platform.

For example, the MicroBlaze soft-core processor has access to a high-speed serial interface called the Fast Simplex Link, or FSL. The FSL is an on-chip interconnect feature that provides a high-performance data channel between the MicroBlaze processor and the surrounding FPGA fabric.

Similarly, the PowerPC hard processor core, as implemented in Virtex-II Pro™ and Virtex-4™ FPGAs, provides high-performance communication channels through the processor local bus (PLB) and on-chip memory (OCM) interfaces, as illustrated in Figure 1.

Using these Xilinx-provided interfaces to define an in-system unit test allows you to quickly verify critical components of a larger application. Unlike system tests (which model real-world conditions of the entire application), a unit test allows you to focus on potential trouble spots for a given component, such as boundary conditions and corner cases, that might be difficult or impossible to test from a system-level perspective. Such unit testing improves the quality and robustness of the application as a whole.

Unit Testing

A comprehensive hardware/software testing strategy includes many types of tests, including the previously-described unit tests, for all critical modules in an application. Traditionally, system designers and FPGA application developers have used HDL simulators for this purpose.

Using simulators, the FPGA designer creates test benches that will exercise specific modules by providing stimulus (test vectors or their equivalents) and verifying the resulting outputs. For algorithms that process large quantities of data, such testing methods can result in very long simulation times, or may not adequately emulate real-world conditions. Adding an in-system prototype test environment bolsters simulation-based verification and inserts more complex real-world testing scenarios.

Unit testing is most effective when it focuses on unexpected or boundary conditions that might be difficult to generate when testing at the system level. For example, in an image processing application that performs multiple convolutions in sequence, you may want to focus your efforts on one specific filter by testing pixel combinations that are outside the scope of what the filter would normally encounter in a typical image.

CoDeveloper generates FPGA hardware from the C-language software processes and automatically generates software-to-hardware and hardware-to-software interfaces. You can optimize these generated interfaces for the MicroBlaze processor and its FSL interface or the PowerPC and its PLB interface. Other approaches to data movement, including shared memories, are also supported.

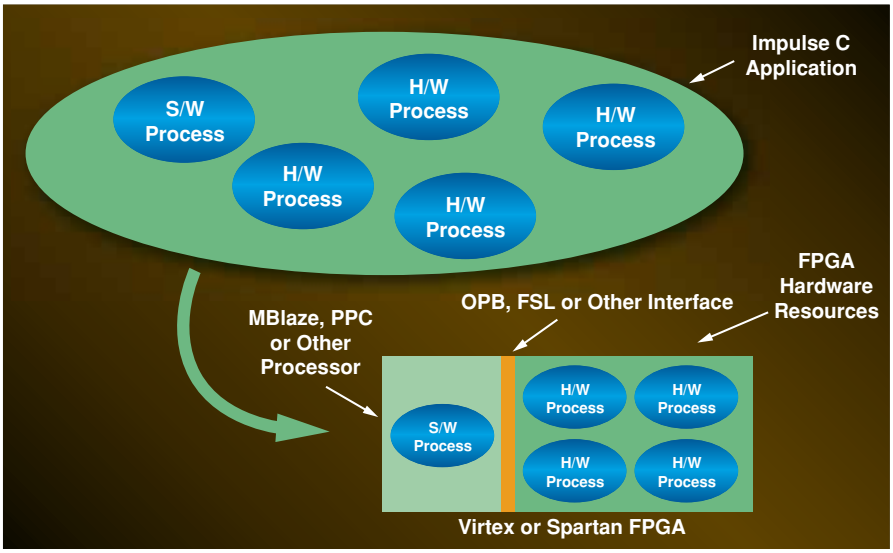


Figure 1 – Hardware and software test components are mapped to the FPGA target and communicate across the OPB or FSL bus to the MicroBlaze or Power PC processor.

It may be impossible to test all permutations from the system perspective, so the unit test lets you build a suite to test specific areas of interest or test only the boundary/corner cases. Performing these tests with actual hardware (which may for testing purposes be running at slower than usual clock rates) obtains real, quantifiable performance numbers for specific application components.

Introducing C-to-RTL compilation into the testing strategy can be an effective way to increase testing productivity. For example, to quickly generate mixed software/hardware test routines that run on the both the embedded processor and in dedicated hardware, you can use tools such as CoDeveloper (available from Impulse Accelerated Technologies) to create prototype hardware and custom test generation hardware that operates within the FPGA to generate sample inputs and validate test outputs.

Desktop Simulation and Modeling Using C

Using C language for hardware unit testing lets you create software/hardware models (for the purpose of algorithm debugging) in software, using Microsoft™ Visual Studio™, GCC/GBD, or similar C development and debugging environments. For the purpose of desktop simulation, the complete application – the unit under test, the producer and consumer test functions, and any other needed test bench elements – is described using C, compiled under a standard desktop compiler, and executed.

Although you can do this using SystemC, the complexity of SystemC libraries (in particular their support for dataflow abstractions through channels) makes the process of creating such test benches somewhat complex. CoDeveloper's Impulse C libraries take a simpler approach, providing a set of functions that allow multiple C processes – representing parallel software or hardware modules – to be described and interconnected using buffered communication channels called streams.

Impulse C also supports communication through signals and shared memories, which are useful for testing hardware processes that must access external or static data such as coefficients.

Data Throughput and Processor Selection

When evaluating processors for in-system testing, you must first consider the fact that the MicroBlaze processor or any other soft processor requires a certain amount of area in the target FPGA device. If you are only using the MicroBlaze processor as a test generator for a relatively small element of your complete application, this added resource usage may be of no concern. If, however, the unit under test already pushes the limits in the FPGA, you may want to target a bigger device during the testing phase or consider the PowerPC core provided in the Virtex-II Pro and Virtex-4 platforms as an alternative.

Synthesis time can also be a factor. Depending on the synthesis tool you use, adding a MicroBlaze core to your complete application may add substantially to the time required to synthesize and map the application to the FPGA, which can be a factor if you are performing iterative compile, test, and debug operations.

Again, the PowerPC core, being a hard core that does not require synthesis, has an advantage over the MicroBlaze core when design iteration times are a concern. The 16 KB of data cache and 16 KB of instructions cache available in the PowerPC 405 processor also makes it possible to run small test programs entirely within cache memory, thereby increasing the performance of the test application.

If a high test data rate (the throughput from the processor to the FPGA) is your primary concern, using the MicroBlaze core with the FSL bus or the PowerPC with its on-chip-memory (OCM) interface will provide the highest possible performance for streaming data between software and hardware components.

By using CoDeveloper and the Impulse C libraries, you can make use of multiple

Figure 2 – A software-based unit test operating on the embedded processor communicates with the hardware unit under test through data streams.

streaming software/hardware interfaces using a common set of stream read and write functions. These stream read and write functions provide an abstract programming model for streaming communications. Figure 2 shows how the Impulse C library functions support streams-based communication on the software side of a typical streaming interface.

Moving Test Generators to Hardware

To maximize the performance of test generation software routines, you can migrate critical test functions such as stimulus generators into hardware. Rather than re-implementing such functions in VHDL or Verilog™, automated C-to-RTL compilation quickly generates hardware representing test producer or consumer functions. These functions interact with the unit under test, using FIFO or other interfaces to implement data streams and supply other test inputs.

The CoDeveloper C-to-RTL compiler analyzes C processes (individual functions that communicate via streams, signals, and shared memories) and generates synthesizable HDL compatible with Xilinx Platform Studio (EDK), Xilinx ISE, and third-party synthesis tools including Synplicity® (Figure 3). The generated RTL is automatically parallelized at the level of inner code loops to reduce process latencies and increase data rates for output data streams.

Automated compilation capability with the ability to express system-level parallelism (creating multiple pipelined processes, for example) makes it possible to generate hardware directly from C language at orders of magnitude faster than the equivalent algorithm as implemented in software on the embedded microprocessor. This creates hardware test generators that generate outputs at a high rate.

Does C-Based Testing Eliminate the Need for HDL Simulators?

C-based test methods such as those described in this article are a useful addition to a designer's bag of tricks, but they are certainly not replacements for a comprehensive hardware simulation. HDL simulation can be an effective way to determine cycle counts and explore hardware interface issues. HDL simulators can also help alleviate the typically long compile/synthesize/map times required before testing a given hardware module in-system. Hardware simulators provide much more visibility into a design under test, and allow single-stepping and other methods to be used to zero-in on errors.

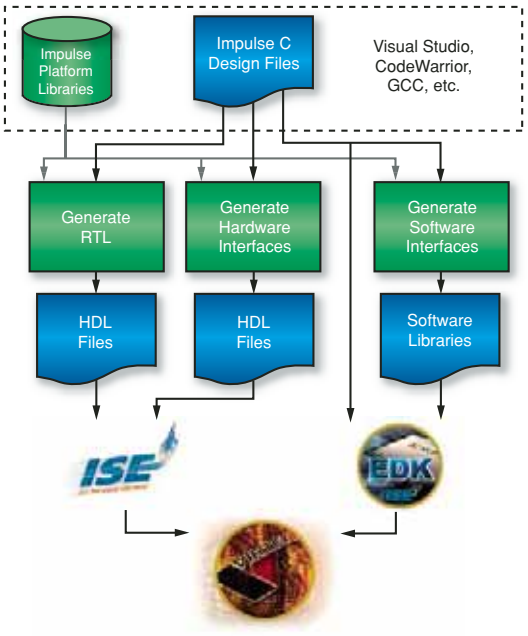


Figure 3 – Hardware and software C code is compiled and debugged in a standard IDE; the interfaces are automatically generated and the generated HDL synthesized in Xilinx tools before downloading into the Virtex-II, Virtex-II Pro, or Virtex-4 device.

If tests require very specific timing, using an embedded processor to create test data will most likely result in data rates that are only a fraction of what is needed to obtain timing closure. In fact, if the test routine is implemented as a state machine on the processor, the speed at which the state machine can be made to operate will be slower than the clock frequency of the test logic in hardware. Hence, for most cases, the hardware portion would need to slow down so the CPU can keep pace – providing test stimulus and measuring expected responses. Alternatively, you can create a buffered interface – a software-to-hardware bridge – to manage the test data using a streaming programming model.

Given the performance differences between a processor-based test bench and the potential performance of an all-hardware system, it should be clear that software-based testing of such applications cannot replace true hardware simulation, in which you can observe, using post-route simulation models, the design running at any simulated clock speed.

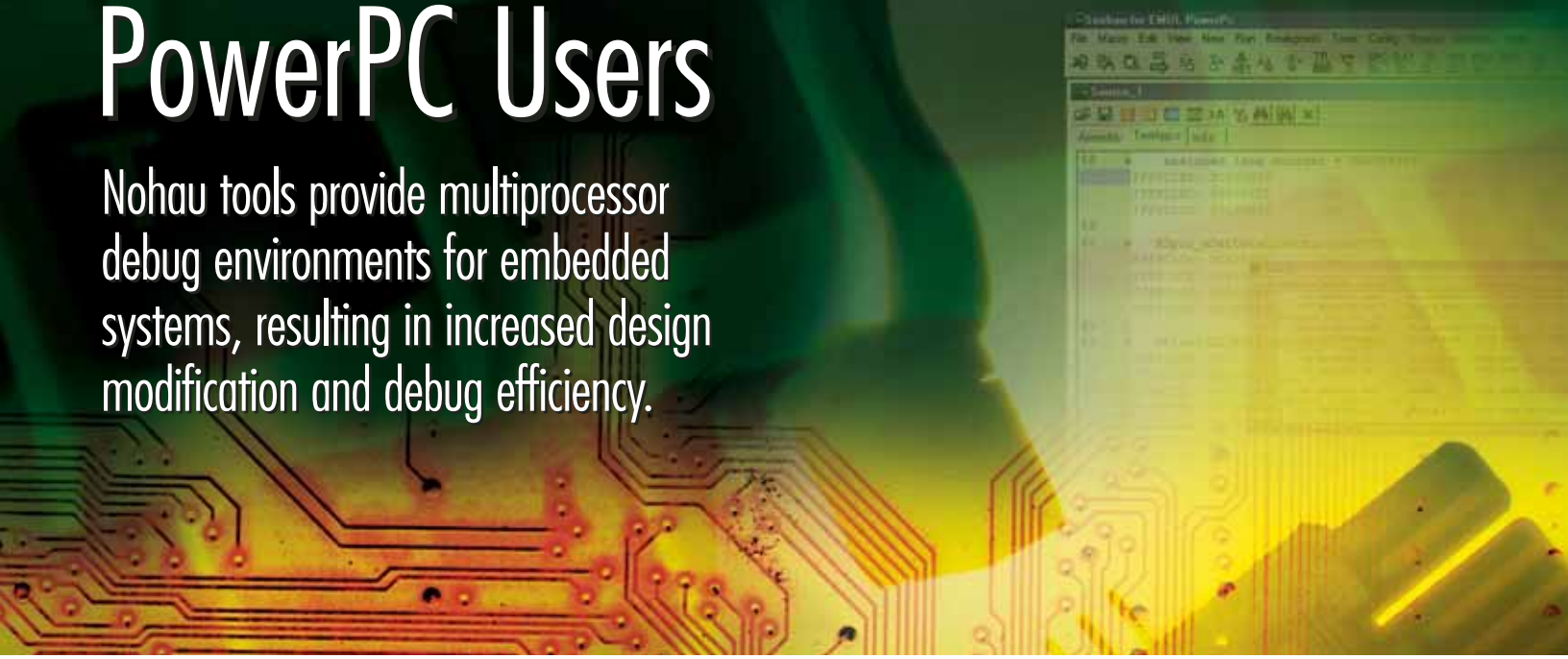
Conclusion

In-system testing using embedded processors is an excellent complement to simulation-based testing methods, allowing you to test hardware elements at lower clock rates efficiently using actual hardware interfaces and potentially more accurate real-world input stimulus. This helps to augment simulation, because even at reduced clock rates the hardware under test will operate substantially faster than is possible in RTL simulation.

By combining this approach with C-to-hardware compilation tools, you can model large parts of the system (including the hardware test bench) in C language. The system can then be iteratively ported to hand-coded HDL or optimized from the level of C code to create increasingly high-performance system components and their accompanying unit- and system-level tests.

For more information, visit *www.impulsec.com*, e-mail *info@impulsec.com*, or call (425) 576-4066.

Nohau Shortens Debugging Time for MicroBlaze and Virtex-II Pro PowerPC Users



Nohau tools provide multiprocessor debug environments for embedded systems, resulting in increased design modification and debug efficiency.

by Darrell Wilburn
President
I.Q. Services, Inc.*
darrell@iq-service.com or *darrellw@nohau.com*

Platform FPGAs can implement a completely configurable system-on-chip by containing one or more microprocessors in a tightly coupled fabric. This delivers very flexible hardware and software, which can change continuously throughout the design and debug cycle.

A powerful set of software debug tools that can properly support sophisticated FPGAs is critical for successful project completion. Debugging and verifying a design from external pins is problematic at best. Reliably measuring 200 to 300 MHz signals (like Fast Simplex Links) over a 3-foot logic cable to an external trace facility is very difficult – and sometimes impossible – to make with sub-nanosecond precision. Furthermore, adding logic paths to provide for external probing is greatly intrusive, which may create new place and route problems as well as timing differences in the final design.

Simulation can still help you overcome the simpler roadblocks, but for real-time or intermittent problems, observing in real time through in-circuit methods quickly becomes a necessity. On-board instrumentation circuits can provide visibility to all system signals as well as executing programs.

The challenges of verification and debug steps are:

- Instrumentation to provide correlated hardware and software measurements
- Needing a broad range of engineering skills
- Extreme flexibility with ever-changing needs for both

Nohau Corporation has developed a compact on-chip development system that enables you to efficiently address these debug issues. The Nohau solution includes compact on-chip debug IP called DebugTraceBlaze that is minimized for size, connects directly to the on-chip peripheral bus (OPB), and utilizes on-chip block RAM for trace storage.

The debug facilities are implemented two ways: through hardware or software. The software-based solution uses a small Xilinx® program called XMD-STUB that resides in the first 1K block of memory. The hardware solution uses programmable logic in the hardware and is transparent to the software. You may choose the solution that is best for you.

Personally, I prefer the software solution because it has less impact on the hardware and is more flexible for customization. Also, the cost of 1K of

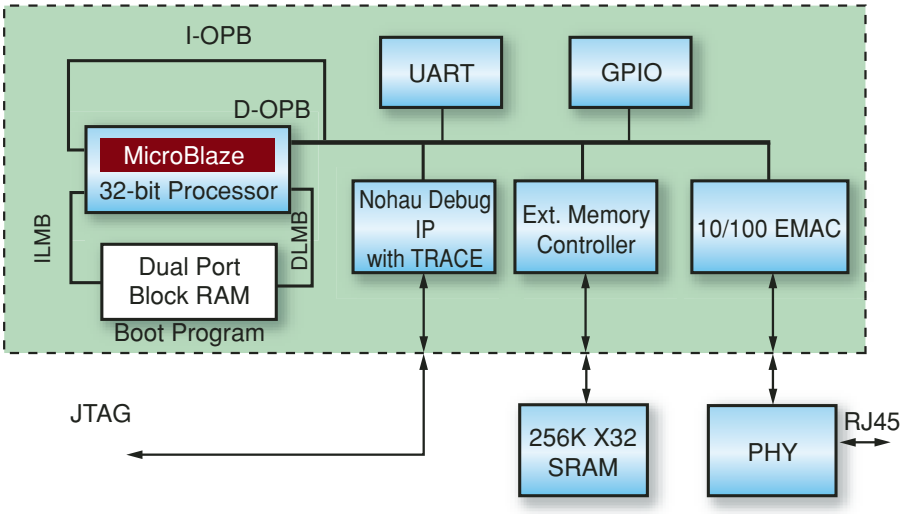


Figure 1 – Nohau Debug IP with trace, shown in a simple five-chip Internet-aware system

memory is usually insignificant in systems that often have 1 MB or more. A block diagram for a typical small system is shown in Figure 1, illustrating placement of the Nohau DebugTraceBlaze module.

Please note that the Nohau solution requires no external signal pins; all access is through the JTAG port. Furthermore, it does not impact timing because it only interfaces through the OPB bus. The resource utilization in the FPGA for the Nohau IP is very small. Actual requirements are shown in Figure 2.

Debug with no trace:	Debug with trace:
Slices = 84	Slices = 425
LUTs = 86	LUTs = 642
Flops = 148	Flops = 489
Mults = 0	Mults = 0
BRAMs = 0	BRAMs = 4

Figure 2 – Nohau resource usage with and without trace

Design/ Debug Flow

The design flow with Nohau tools present is illustrated in Figure 3. You may build an initial system from scratch or use a platform generator like Nohau BlazeGen or Xilinx Platform Studio (XPS) Base System Builder (BSB).

Simply specify your system with the .MHS, .MSS, .UCF, and project options files, which are generated by the platform builder or user-generated text files. To add the Nohau DebugTraceBlaze IP to a project, you first build it with BSB or BlazeGen and add DebugTraceBlaze IP with a pass through BlazeGen.

The output of the XPS build is a .bit file that contains the bitstream required to program the target FPGA with your system. The Nohau Seehau debugger is a convenient and easy-to-use GUI interface that allows fast and easy updating of system hardware and software as well as test and check-out of software execution. Seehau loads the bit file and programs the FPGA in just a few seconds.

A second path for code development is shown in Figure 3. BlazeGen generates small pre-tested code snippets that fit entirely in one on-board block RAM to provide you with a solid starting place for initial power-up check-out. These snippets are treated just like user code for input to the GNU compiler.

You can enter and compile C/C++ code from inside XPS or from an external editor and compiler using its own make files. For large programs, I recommend using an external GNU make facility. The output from the compile process is an .elf file that contains all code and symbolic information to be loaded directly by Seehau.

As shown in Figure 3, the classic edit/compile/debug loop familiar to embedded system engineers centers around the Seehau debugger. Additionally, a hardware edit/compile/debug loop is now included that loops back through new builds in XPS.

Debugging with Seehau

Seehau provides an intuitive source-level debugger that can be made aware of logic signals in the fabric; RTOS state and variables; correlation of hardware signals to code execution; and Ethernet performance characteristics in Internet-aware applications. Seehau is a full-featured source or assembly debugger with an integral real-time trace facility. It supports either PowerPC™ hard-core or MicroBlaze™ soft-core processors.

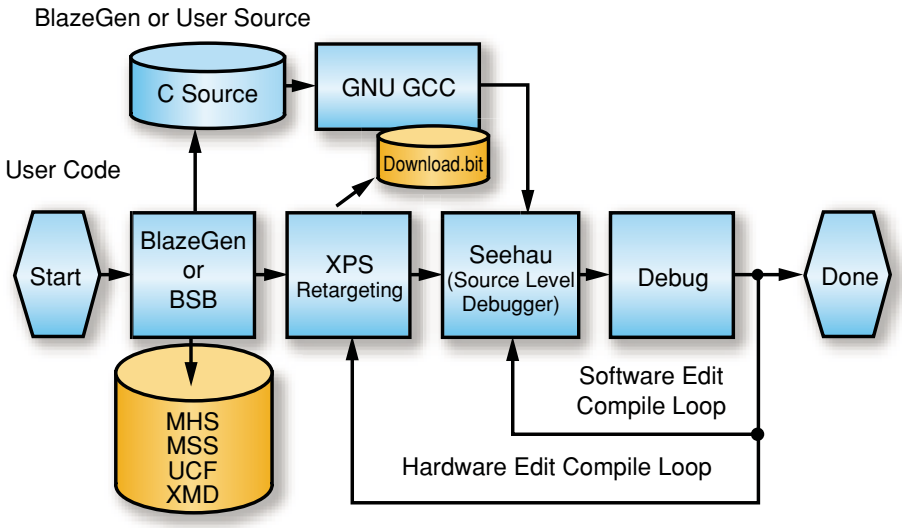


Figure 3 – Development flow with Seehau source-level debugger in place

You can look back in time from any execution to follow the path backward, or you can use the Seehau event configuration system to specify pre- and post-triggering, complex breakpoints, triggers on register reads and writes, and triggers on data from the fabric. Figure 4 shows a typical source-level debug display with processor registers, memory data, program data in source form, and trace and breakpoint status.

Nohau tools are sold as a system, which includes the Seehau debugger, an interface pod to the appropriate JTAG connector, and the IP DebugTraceBlaze configured with trace memory. As a system, it may be ordered as EMUL-MICROBLAZE-PC.

The Nohau EMUL-MICROBLAZE-PC provides a 512-frame or 2K-frame deep trace with a trigger, post-trigger count, and break control. Probe pins may be either 8 or 40 bits wide. It will display data connected to it as specified in the XPS .MHS file.

Figure 5 illustrates a trace display in mixed mode with C source and assembly source intermixed. On this single display, you can correlate the frame at capture time, the execution address, the opcode of the instruction executed, the disassembled MicroBlaze instruction, the C source line that generated that instruction, and 40 bits of data from any logic in the system. Data from logic can include signals from your own logic design.

Multiprocessor System Support

Recently, Nohau completed a joint project with Xilinx, expanding the Seehau system to include support for the hard-core PowerPC processor found in Virtex-II Pro™ devices. As Seehau is a robust, source-level debugger, the user interface and source-level feature set are nearly identical. The only major changes from an embedded system engineer's point of view are the processor-level language on disassembled screens and the register set

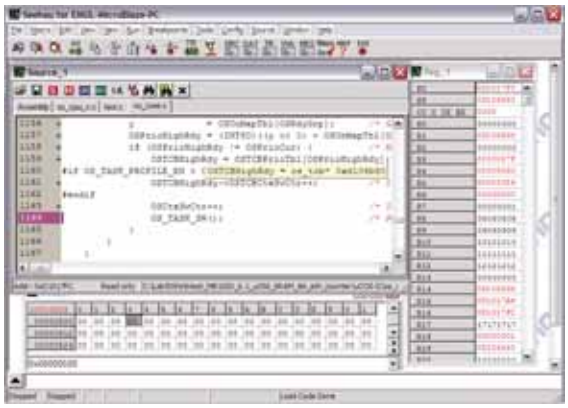


Figure 4 – Full source with breakpoint at context switch in µC/OS-II

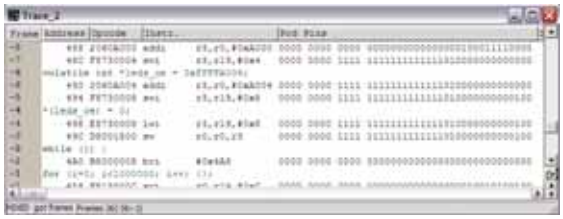


Figure 5 – Trace 40-bit mixed-mode display in binary logic signals from FPGA fabric

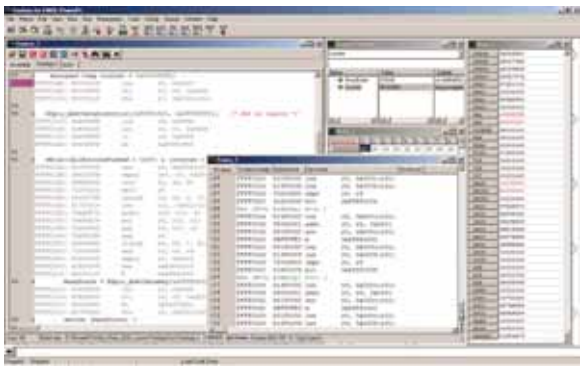


Figure 6 – PowerPC source-level debug with trace

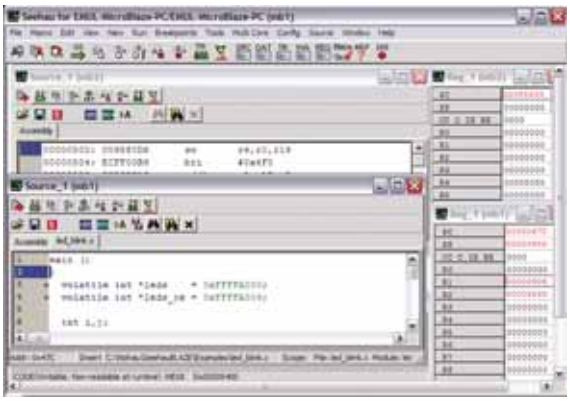


Figure 7 – Multi-core display with two MicroBlaze processors

associated with the PowerPC architecture. Figure 6 shows a source-level debug screen of a typical PowerPC debug session.

Seehau has also been expanded to include support for multiple processors in the same fabric. The processors are run independently. Figure 7 shows a set of screens for a two-processor system.

The Nohau GUI provides a simple, easy-to-use interface that assigns a complete set of control and status windows to each processor. All Seehau windows are available for both processors, and show the name given to each processor in the top banner. In the case shown in Figure 7, the execution sites are named MB1 and MB2.

When you select a command from a pull-down list by clicking on it, the command is directed to the processor assigned to the window in focus. You control the set of windows open for each processor through a pull-down menu. The choice of open windows is controlled by your selection of new windows to view. The result is an easy-to-use, intuitive user interface.

Conclusion

Getting the right tool set and development environment set up for a new FPGA project is critical to the success of the product development cycle. A highly productive development and debug environment based around Nohau tools supports these new multiprocessor systems, with an extension of the same powerful debug and test tools the company has offered for the last 20 years.

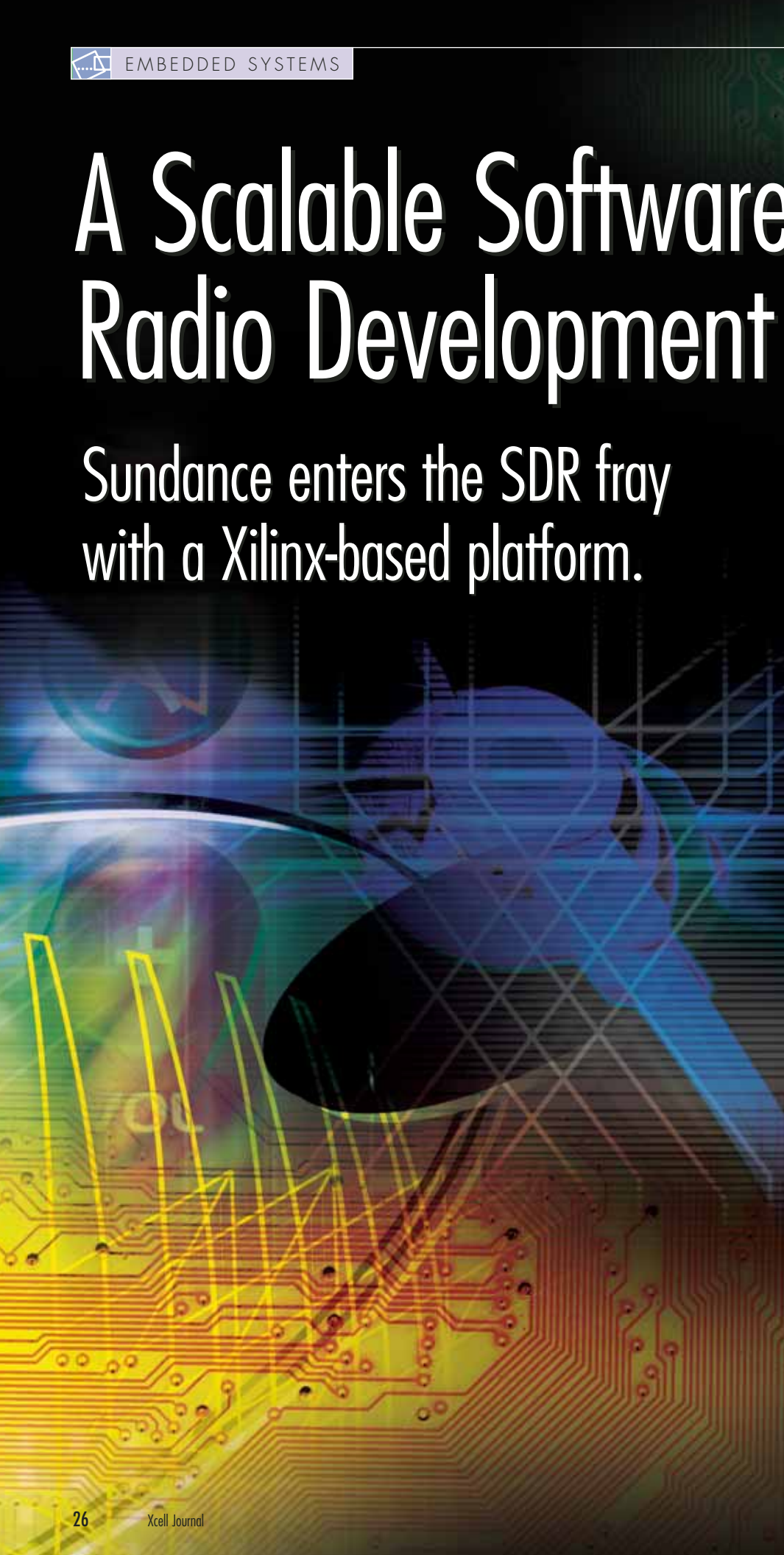
For more information, please visit *www.nohau.com*, *www.iq-service.com*, or e-mail *darrell@iq-service.com* or *darrellw@nohau.com*.

** I.Q. Services is under contract to support and market platform FPGA tools for Nohau Corporation and performs custom start-up engineering for platform FPGA embedded system designs.*

A Scalable Software-Defined Radio Development System

Sundance enters the SDR fray with a Xilinx-based platform.

by Flemming Christensen
Managing Director
Sundance
Flemming.C@sundance.com



There has been a strong push in the past few years to replace analog radio systems with digital radio systems. The Department of Defense Joint Tactical Radio System program shifted the emphasis on the development of software-defined radio (SDR) to the forefront of research and development efforts in the defense, civilian, and commercial fields.

Although it has existed for many years, SDR technology continues to evolve through newly funded ventures. The "holy grail" of SDR is its promise to solve incompatible wireless network issues by implementing radio functionalities as software modules running on generic hardware platforms. Future SDR platforms will comprise hardware and software technologies that enable reconfigurable system architectures for wireless networks and user terminals.

Although new SDR-based systems are purported to be highly reconfigurable and reprogrammable right now, the truth is that SDR hardware platforms are still in their early development stages. Many issues must still be resolved, including reconfigurable signal processing algorithms, hardware and software co-design methodologies, and dynamically reconfigurable hardware. Overall, the main key issues for SDR embedded system platforms are flexibility, expandability, scalability, reconfigurability, and reprogrammability.

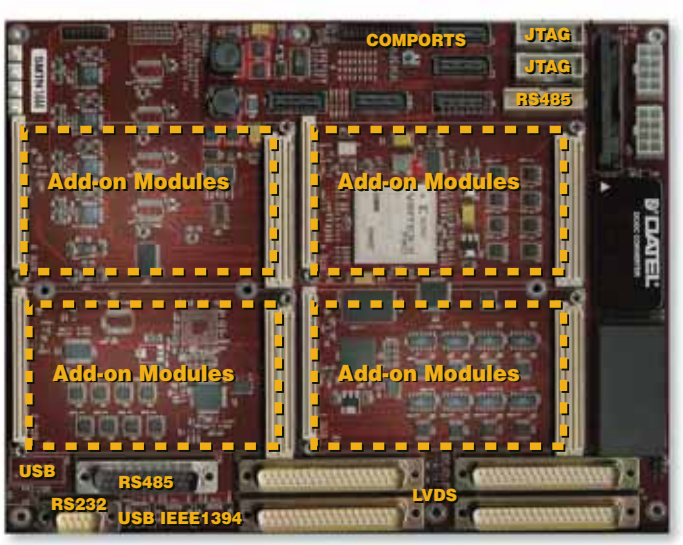


Figure 1 – The SMT148

Meeting the Challenge

SDR is characterized by a decoupling between the heterogeneous execution platform, based on hardware functions together with DSP and MCU processors, and the applications through abstraction layers.

One of the approaches for meeting the complexity demands of third-generation systems is to use multi-core systems where a DSP and a microcontroller work together with an FPGA-based hardware co-processor. With its embedded IBM™ PowerPC™, the Xilinx® Virtex-II Pro™ FPGA is rapidly becoming a solution that embeds and tightly couples reconfigurable logic and a processor in the same device.

Sundance, a developer of advanced system architectures for high-performance signal processing applications, has focused on designing FPGA-based development platforms that address an SDR OEM's wish list. Our challenge was to design a system that would provide the scalability SDR systems require.

The Embedded System Controller

The SMT148 (Figure 1) is one of many development systems Sundance has launched recently. Aimed specifically at SDR, the SMT148 is a fully configurable and expandable waveform development environment that meets the many requirements of SDR developers. This entry-level, stand-alone system enables radio designers to investigate and experiment with the many configurations of multi-channel software programmable and hardware-configurable digital radio.

The SMT148 has at its heart a powerful embedded system controller that leverages the Xilinx Virtex-II Pro FPGA with its embedded PowerPC 405 processor (Figure 2). As an embedded system controller, the role of the Virtex-II Pro device is to manage the reconfiguration of the add-on modules, especially when downloading. Reconfiguration means switching between modes or updating a hardware/software component.

In the global functioning of the SMT148, you can download many kinds of software (high-level applications, protocol stacks, low-level signal processing algorithms) and employ several methods to download software. The eight RocketIO™ transceivers on the Virtex-II Pro device enable high-speed data transfer to additional SMT148 or Virtex-II Pro add-on modules.

Downloads can leverage a powerful I/O architecture that includes the popular FireWire, USB interfaces, LVDS interfaces, and JTAG for debugging and downloads. Data flows into the FPGA and is managed by a Sundance program written for the embedded PowerPC before processing. Figure 3 is the C code comprising the data flow and RocketIO PowerPC program.

Scalable, Reconfigurable Embedded Processors

Scalability is addressed through four add-on module sites, and you can partly resolve the requirements of dynamic reconfiguration by adding additional Xilinx-based FPGA modules. All add-on modules communicate through the Virtex-II Pro device, which also manages two 32-bit microcontrollers that enable communications with most widely used standards.

With the RocketIO transceivers connected to differential pair connectors, you can connect FPGA systems directly

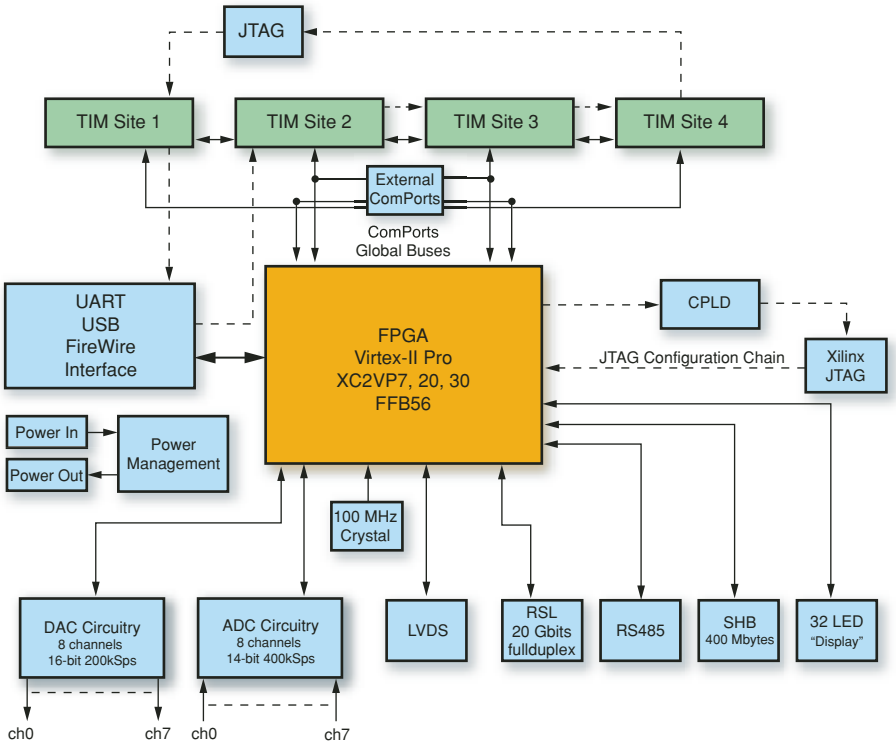


Figure 2 – SMT148 systems architecture

through simple cable connections supporting more than 2 Gbps data rates.

The SMT148 leverages the Xilinx Virtex-II Pro block RAM configuration to generate FIFOs for the RocketIO transceivers, add-on modules communication ports, and a high-speed bus, as well as the embedded PowerPC code. The single high-speed bus allows parallel data transfer to and from a wide range of high-speed ADC/DAC modules.

When we designed the SMT148, we understood that one of the many challenges OEMs would face in developing JTRS-compliant platforms was the availability of interchangeable and networked processing nodes. The availability of processing nodes aims to meet the expandability and scalability requirements in complex waveform applications.

The SMT148 meets this availability challenge with a network of daughter sites that

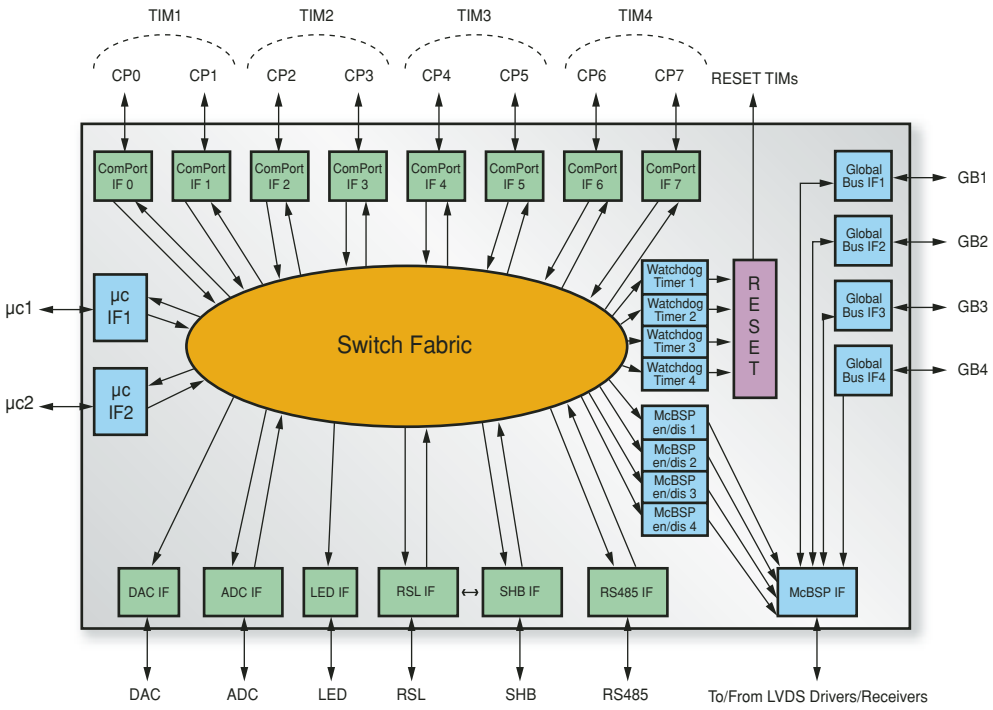


Figure 3 – Interconnections diagram of the digital modules inside the SMT148

Data rates on this port are in excess of 100 MHz (400 Mbps), and are useful for transferring sampled 16-bit I and Q. Processing data streams can take place either in the embedded PowerPC in the Virtex-II Pro device or throughout an array of other add-on Virtex-II Pro FPGA-based modules with embedded PowerPC.

The FPGA on the SMT148 carrier card is connected to many different devices and therefore has many internal interfaces that allow it to exchange data or commands with the external world. All interfaces are reset at power on when applying a manual reset. Figure 4 shows the interconnections between the digital modules inside the FPGA.

you can use for additional resources such as signal processors, reconfigurable computing modules, and Sundance's large family of add-on modules. These add-on modules include a variety of embedded system options such as reconfigurable modules with tightly coupled Virtex-II Pro FPGAs and DSPs, digital and analog converters, data conversions, transceivers, and I/Os of all types.

Designed for Developers

Powered by an external supply, the SMT148 platform has an impressive topology that accepts input signals from various sources through a network of multi-pin connectors. High-speed I/O channels support the additional nodes in a network topology neatly harmonized to the Xilinx architecture.

Fully interconnected and configurable through their communication ports, these add-on sites are also connected to the embedded Virtex-II Pro FPGA. The availability of a network of add-on sites removes the main expandability restrictions often associated with other platforms, and offers the OEMs a highly compact design and development tool.

More importantly, this scalable system architecture makes the SMT148 a perfect development platform with which to resolve the many issues related to the implementation of multiple radio functionalities in a single environment. These can be addressed as multiple software modules running on Sundance's reconfigurable hardware platform.

IP Cores

Sundance takes advantage of the high-performance DSP acceleration capabilities and flexible connectivity that the Virtex-II Pro FPGA provides by supporting developers with a family of software tools and IP cores.

The SMT148 I/O flexibility enables you to rapidly investigate and experiment with features of the Virtex-II Pro FPGA as well as those of developed IP cores from Sundance. These include multi-tap complex filters, Viterbi decoders, encoders, a complete transmitter, QAM mapper, multi-phase pulse shaping filter, multi-phase cascaded integrator comb (CIC) interpolation filter with a fixed interpolation rate, multi-phase numerical-controlled oscillator (NCO), and multi-phase digital mixer.

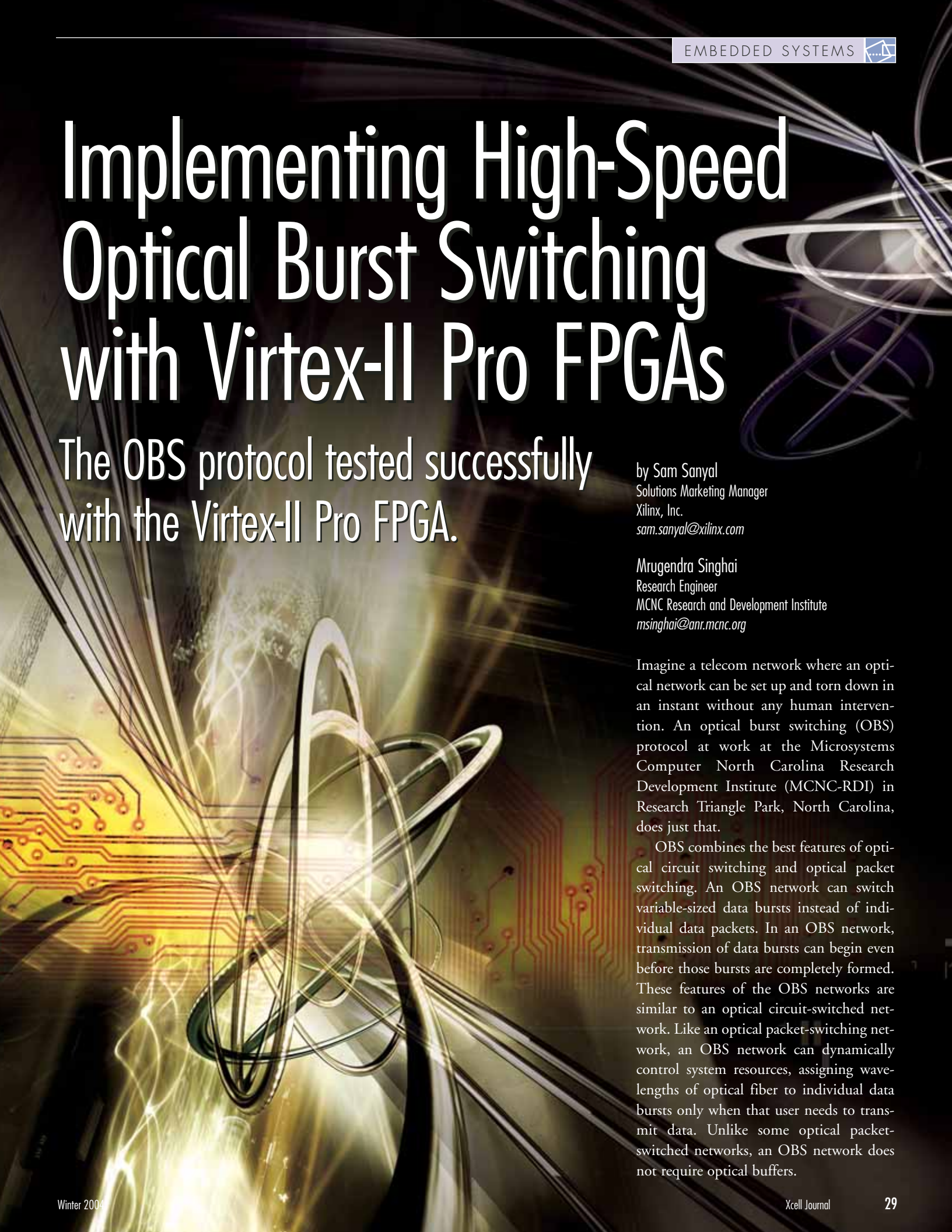
Conclusion

Developing, testing, and implementing SDR IP cores is simplified with the Sundance SMT148 platform. You can now focus on developing additional IPs without worrying about peripheral processing or I/O devices, as these are simply off-the-shelf add-on IP blocks.

For more information, please visit *www.sundance.com*.

Implementing High-Speed Optical Burst Switching with Virtex-II Pro FPGAs

The OBS protocol tested successfully with the Virtex-II Pro FPGA.



by Sam Sanyal
Solutions Marketing Manager
Xilinx, Inc.
sam.sanyal@xilinx.com

Mrugendra Singhai
Research Engineer
MCNC Research and Development Institute
msinghai@anr.mcnc.org

Imagine a telecom network where an optical network can be set up and torn down in an instant without any human intervention. An optical burst switching (OBS) protocol at work at the Microsystems Computer North Carolina Research Development Institute (MCNC-RDI) in Research Triangle Park, North Carolina, does just that.

OBS combines the best features of optical circuit switching and optical packet switching. An OBS network can switch variable-sized data bursts instead of individual data packets. In an OBS network, transmission of data bursts can begin even before those bursts are completely formed. These features of the OBS networks are similar to an optical circuit-switched network. Like an optical packet-switching network, an OBS network can dynamically control system resources, assigning wavelengths of optical fiber to individual data bursts only when that user needs to transmit data. Unlike some optical packet-switched networks, an OBS network does not require optical buffers.

The MCNC-RDI has developed a NASA-funded OBS protocol implementation, called JIT (Just-In-Time), which recently achieved successful testing in an ATDnet (Advanced Technology Demonstration network) testbed. Established by the Defense Advanced Research Projects Agency (DARPA) for demonstrating advanced networking technology, the all-optical ATDnet runs at 2.5 Gbps through six sites using eight wavelengths and wavelength division multiplexing (WDM) switches. The testbed included applications in multiple areas like optical networking, network security, and networked information systems.

Technology Overview

WDM is a method of transmitting data from different sources over the same fiber-optic link at the same time; each data channel is carried on its own unique wavelength. The result is a link with an aggregate bandwidth that increases with the number of wavelengths employed. In this way, WDM technology can maximize the use of the available fiber-optic infrastructure – what would normally require two or more fiber links will now require only one.

WDM technologies primarily differ in the number of available channels. Coarse wave division multiplexing (CWDM) combines as many as 16 wavelengths onto a single fiber; dense wave division multiplexing (DWDM) combines as many as 64 wavelengths onto a single fiber.

With DWDM technology, the wavelengths are closer together than CWDM, meaning that transponders are generally more complex and expensive than CWDM. However, with DWDM, the advantage is a much higher density of wavelengths, and also longer distance. DWDM is emerging as a preferred solution for providing scalable and efficient optical networking technologies of the future.

The key objective of the hardware-based OBS protocol implementation is to dynamically manage commercially available WDM switches. An OBS network comprises OBS network controllers and clients with OBS network interface cards (NICs). OBS network controllers direct the optical data bursts received from a source-client OBS NIC to a destination-client OBS NIC.

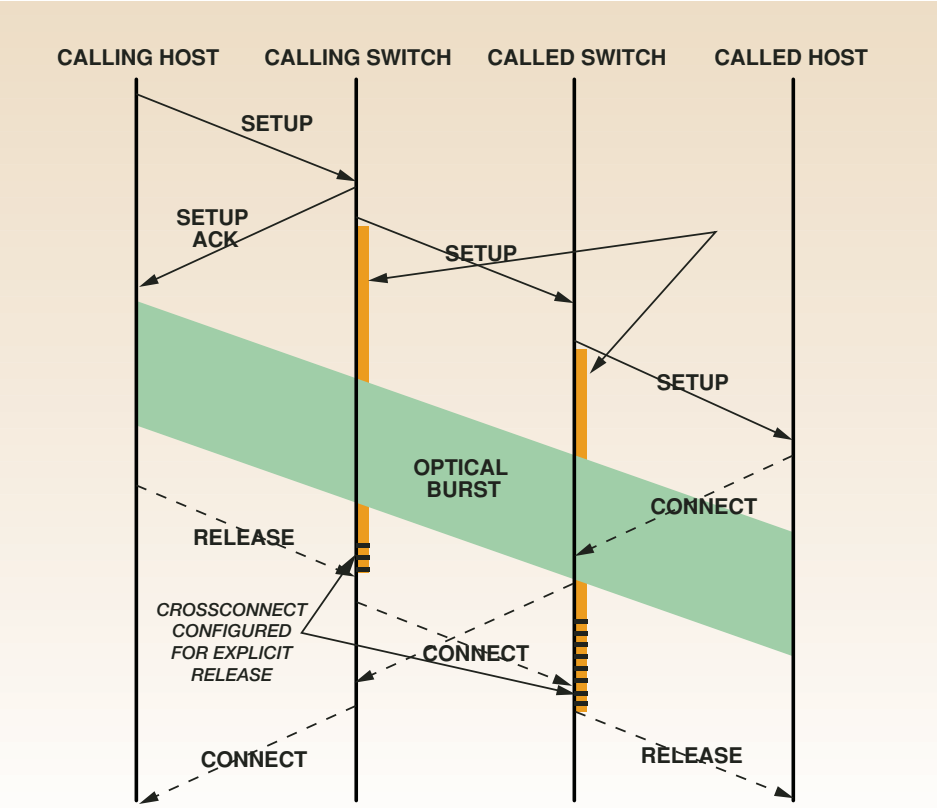


Figure 1 – JIT signaling scheme

Advances in Xilinx FPGA technology have made it possible for the MCNC-RDI to build a NIC that implements the JIT signaling protocol for an OBS network. The OBS NIC uses DWDM technology to transmit and receive data optically on specific wavelengths and is capable of handling data rates as high as 1.25 Gbps. The NIC card can be tuned dynamically to as many as eight different DWDM wavelengths.

In the JIT protocol, a control packet reserves a wavelength channel in the network for a period of time ***L*** equal to the burst length, starting at the expected arrival time ***R*** (this can be adjusted by the number of hops that a burst needs to travel and the processing time at each intermediate node).

If the reservation is successful, the control packet adjusts the offset time for the next hop and forwards it on. If the reservation is not successful, the burst will be blocked and the packet will be discarded. Because JIT is a one-way reservation protocol, buffering does not occur at the node level, thus reducing any latency. Implementation of JIT with an efficient scheduling algorithm can further decrease the probability of burst loss.

The JIT protocol uses a SETUP message to announce a burst in the OBS network. Each optical burst of data, comprising some number of contiguous packets destined for a specific destination, is sent immediately after the node receives a SETUP ACK from the ingress OBS node. An out-of-band SETUP message is sent across all switches before this step to prepare all path switches for the burst data. OBS does not use any optical buffering or packet parsing. For a long burst, a KEEPALIVE message may be required to keep all switches in active state. The JIT signaling scheme is shown in Figure 1.

The Role of the FPGA

The development of the OBS NIC was enabled by the availability of integrated high-speed multi-gigabit RocketIO™

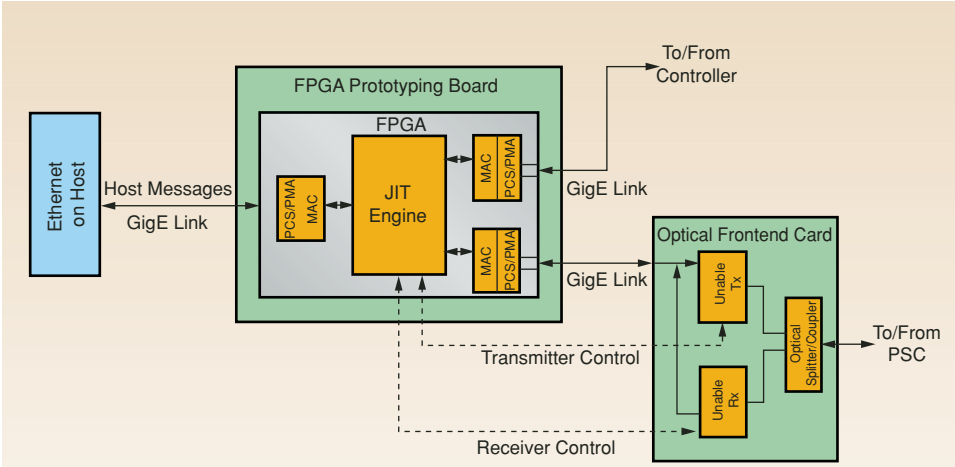


Figure 2 – Architecture of OBS NIC

transceivers in the Virtex-II Pro™ FPGA, allowing high-speed data streams (1-10 Gbps) to directly reach the core of the FPGA for processing. Dense FPGA logic available in the Virtex-II Pro FPGA facilitates implementation of complex state machines of the JIT protocol. The availability of embedded IBM™ PowerPC™ 405 processors in the Virtex-II Pro FPGA allows implementation of complex scheduling algorithms and timers associated with the JIT protocol.

The OBS NIC contains a Virtex-II Pro XC2VP20 FPGA. Three Gigabit Ethernet channels are used in this implementation of the OBS NIC. The first channel on the OBS NIC connects to an off-the-shelf Gigabit Ethernet card plugged into the host. This channel carries data and host messages between the OBS NIC and the host. The second channel is for signaling and connects to the OBS network controller; it carries the JIT OBS signaling messages. The third channel is used as the data channel and is connected to the optical front-end card.

The optical front-end card consists of an optical tunable transmitter and receiver. The OBS NIC generates the tuning commands for the laser and optical receivers on the optical front-end card. Figure 2 illustrates the architecture of the OBS NIC.

The Virtex-II Pro FPGA on the OBS NIC uses a PCS/PMA core and a MAC layer to connect the external gigabit channels to the JIT engine.

The JIT engine implements the JIT OBS protocol in the OBS NIC. Functionalities for both the source and destination state machines of the JIT OBS client are implemented in the JIT engine. The JIT engine processes three kinds of messages – messages from the host, signaling messages from the network, and internally generated timing messages.

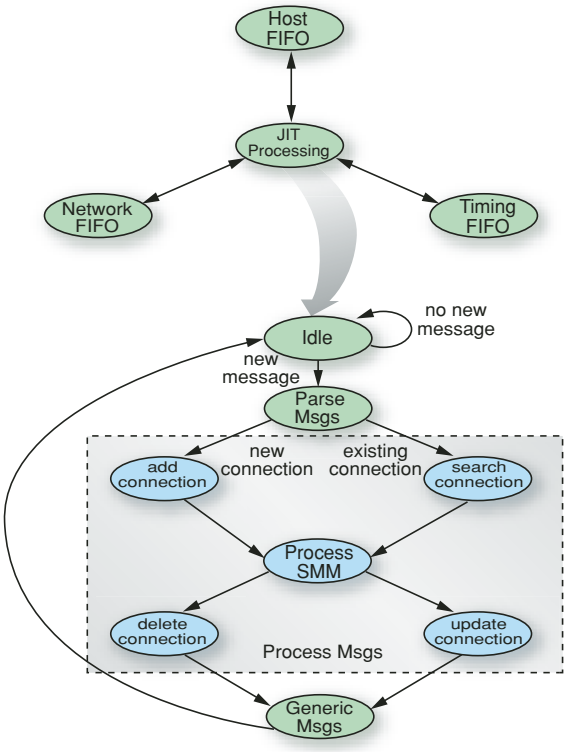


Figure 3 – FSM system flow diagram

The JIT engine uses two functional state machines (FSM): the scheduling FSM, using a round-robin scheme, picks up a message from one of the three message queues (for different types of messages) and dispatches them for further processing, while the processing FSM is responsible for taking a message and processing that message. Several processing sub-modules can be activated by processing FSM as needed, such as a hashing module or a state machine module. Figure 3 diagrams the processing of messages in the JIT engine.

Conclusion

We believe that communications will be bi-modal within the next 25 years. All land lines will be optically based, with optical access to the user or device that is a client of the network. All backbone connections will be across optical trunks. Networking will be predominantly implemented in the "optical layer," with little or no additional layering above it. Optical networks will be mostly a transparent transport media for applications.

To meet the increasing demands of bandwidth and cost reduction, several technologies in the optical communications paradigm have been under intensive research.

Just-In-Time signaling applied to the optical burst switching paradigm has the promise of being able to provide either circuit- or packet-switched services. JIT OBS implements the best of optical circuit switching and optical packet switching but avoids their shortcomings. JIT signaling aims to better utilize the variable parameters that can exist within both an optical and a wireless network, such as frequency availability and data-rate differences.

For more information on the research conducted by MCNC-RDI in the field of optical networks, visit *www.mcnc-rdi.org.*

So Many Gates

So Few Dollars

The hard work is done! Now ASIC and FPGA designers can prototype logic designs for a fraction of the cost of existing solutions. Here are 6+ million gates (measured the ASIC way) on an easy to use, stand-alone, USB2.0-hosted board (a PCI/PCI-X interface is coming soon). The DN6000k10 supports up to 9, 2vp100 VirtexII-Pro FPGA's, with an incredible amount of FPGA to FPGA interconnect for easy logic partitioning. FPGA's are interconnected with rocket I/O's, enabling the movement of data between them at 100's of GB/s. In addition to 6M+ gates, the DN6000k10 also packs on-board:

- 2 PowerPC cores per FPGA (400MHz)
- Up to 8MB embedded RAM, 444, 18x18 multipliers — per FPGA
- 12 external 133MHz 32M x 16 DDR SDRAM's, 5 4Mx16 FLASH
- 480+ connections for daughter card and logic analyzer interfaces

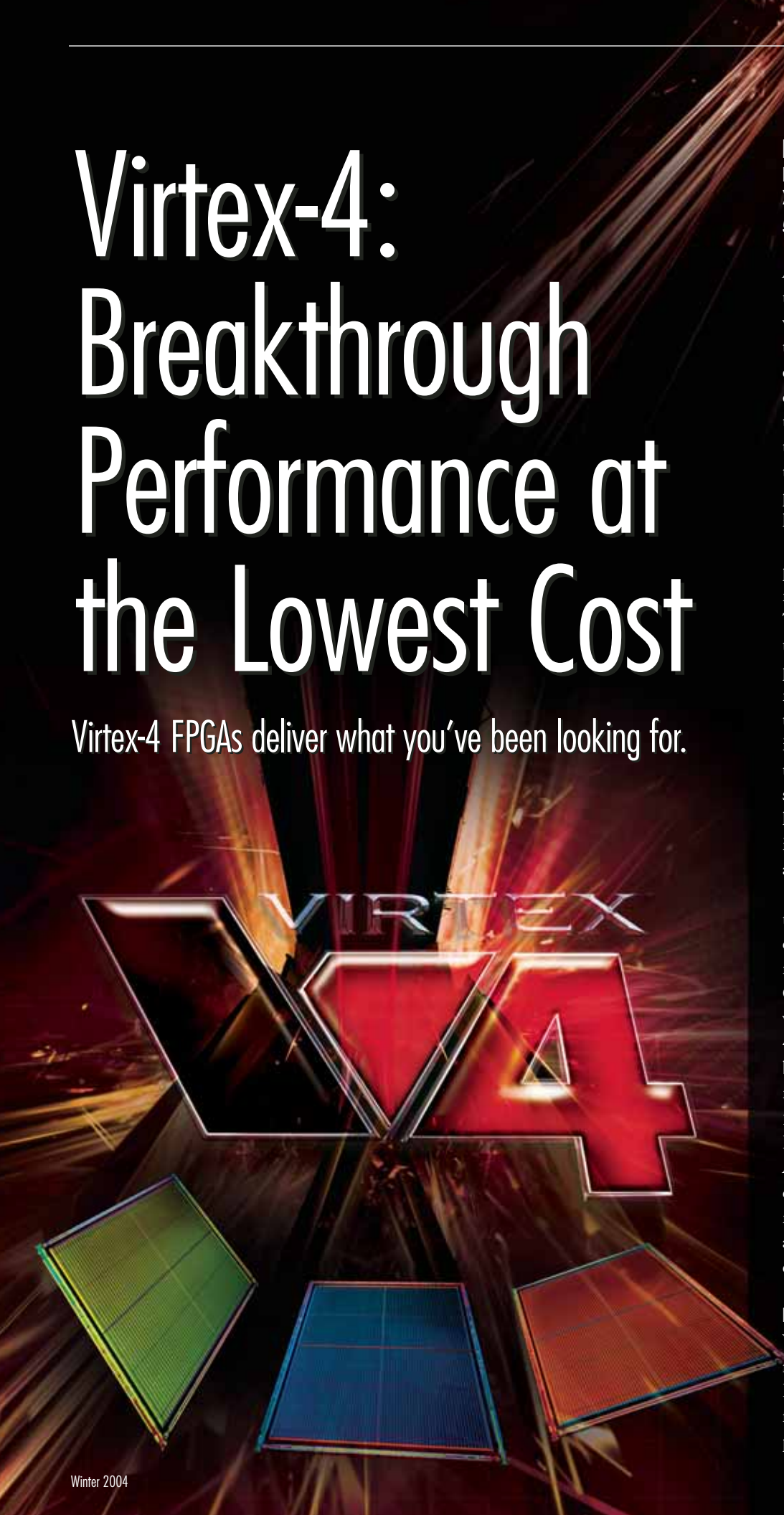
Configuration is fast, easy, and robust using a SmartMedia-based FLASH card or, via the USB interface. Every tool, utility, driver, and support application that The Dini Group could imagine you might need is included. Please contact us for complete specifications, we are eager to show you how our hard work can make your job easier.

The DiNI Group

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

Virtex-4: Breakthrough Performance at the Lowest Cost

Virtex-4 FPGAs deliver what you've been looking for.



by Greg Lara
Product Marketing Manager – Virtex Solutions
Xilinx, Inc.
greg.lara@xilinx.com

As Xilinx® began to define the capabilities of the fourth-generation of Virtex™ devices, we set out to address the performance, functionality, and cost requirements of next-generation electronic systems, and to increase our customers' productivity by easing system design challenges. We interviewed more than 800 customers, including system architects and experts in logic design, embedded processing, high-performance DSP, and high-speed connectivity.

Despite the differences in their end products, these high-end FPGA users had a number of common key requirements. They asked for higher system performance to meet the demands of their leading-edge products; lower power consumption to meet stringent power budgets driven by system cost and reliability requirements; help in reducing system cost to enable them to thrive in a competitive marketplace; and solutions to simplify complex design challenges, such as building source-synchronous interfaces to the latest high-speed memories and advanced components.

We achieved these goals by enhancing the features proven popular in earlier Virtex devices and developing new capabilities never before available in FPGAs. Combining advanced processing technology with greater integrated functionality, Virtex-4™ FPGAs provide 2x more density, and boost performance as much as 2x, while reducing power consumption by as much as 50% compared with previous-generation FPGAs (see sidebar, "Features at a Glance"). At the same time, Virtex-4 FPGAs cut the cost of programmable system platforms by more than 50%, enabling developers to adopt high-performance FPGAs in an extraordinary range of products.

Higher Performance

Viretx-4 FPGAs attack the requirements for higher performance on several fronts. First, designers can improve system performance, thanks to the advanced 90 nm process and optimized FPGA fabric.

The second approach is to include dedicated, performance-tuned circuitry for implementing key system functions, such as integrated processors, DSP slices, Ethernet MACs, and serial transceivers. For example, the embedded Virtex-4 XtremeDSP™ slice delivers up to 500 MHz performance and the RocketIO™ serial transceiver ranges from 0.6 to 11.1 Gbps – unprecedented in the industry.

The third approach is the incorporation of powerful clock management capability, enabling engineers to extract the maximum performance from the programmable logic fabric. Xesium clocking technology addresses designers' demands for more flexible clocking with abundant resources – up to 32 global clocks in each device and up to 20 digital clock manager (DCM) circuits.

Xesium DCM circuits enable flexible generation of multiple clock domains with differential signaling supporting frequencies of up to 500 MHz performance and 40% less jitter than previous circuitry. In addition, Virtex-4 devices are the only FPGAs to provide differential clocking networks, a key advantage in implementing precision clocks with minimal skew and jitter.

Virtex-4 FPGAs further enhance clock management with phase-matched clock dividers (PMCD) that provide improved handling of multiple synchronous clock domains. These circuits, together with enhanced software support, give designers precise edge control and frequency synthesis capabilities, enabling the generation of high-quality clock networks.

Power Advantage

Virtex-4 FPGAs reduce power with a combination of techniques. By using a triple oxide technology, Xilinx can make trade-offs between speed and leakage that reduce static power consumption by 40% as we build transistors with different gate oxide thicknesses for configuration, interconnect, and I/O. This technology enables us to offset, and even reverse, the increase in leakage current inherent in the migration to finer geometry nodes and is exclusive to Xilinx in the FPGA industry.

In addition, dynamic power consumption decreases by 50% because of lower supply voltage and lower capacitance in the 90 nm process. Finally, extensive use of abundant embedded IP provides valuable functionality in circuits optimized to consume as little as one-tenth the power of an equivalent implementation in programmable logic fabric.

Lower System Cost

Xilinx addressed the requirements for lower system cost on three fronts:

- 90 nm, 300 mm process leadership produces the lowest FPGA price.

  Xilinx manufactures Virtex-4 FPGAs using the same 90 nm, 300 mm processing technology we use to build the world's lowest-cost FPGAs, Spartan-3™ devices. The combination of finer geometries and larger 12 inch wafers produces approximately five times as many die per wafer, compared to building an equivalent chip with 130 nm process on 200 mm (8 inch) wafers. This lowers cost per die significantly.

- Multiple platforms deliver cost-optimized feature sets.

  With each generation of Virtex FPGAs, Xilinx has taken advantage of the latest process node to fabricate devices that offer greater capacity, higher performance, and lower price. For the Virtex-4 family, we went even further to achieve cost reduction.

  As we strive to expand the use of Virtex FPGAs into new markets and geographies, we see that our customers have different requirements that vary with the complexity and target price for the systems they are creating. Using our propriety ASMBL (pronounced "assemble") architecture (see Figure 1 and sidebar, "ASMBL Architecture Enables Cost-Optimized Platforms"),

ASMBL Architecture Enables Cost-Optimized Platforms

With traditional FPGA architectures, increasing the size of the devices to meet the demands for greater logic capacity and more memory typically results in parallel scaling of all the advanced features on the die, rapidly increasing cost.

To solve this inefficiency, Xilinx introduced a radical new architecture that enables us to offer a new generation of Virtex FPGAs providing the broadest range of capabilities in three unique platforms with feature mixes optimized to meet the requirements of different application domains. The ASMBL (Advanced Silicon Modular Block) architecture enables Xilinx to scale the capabilities and capacity of Virtex FPGAs independently of one another and rapidly assemble multiple platforms.

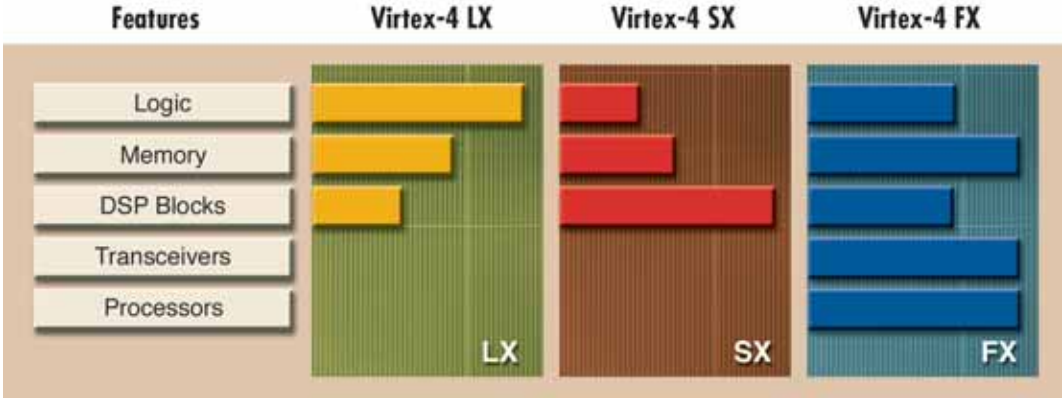


Figure 1 – ASMBL architecture

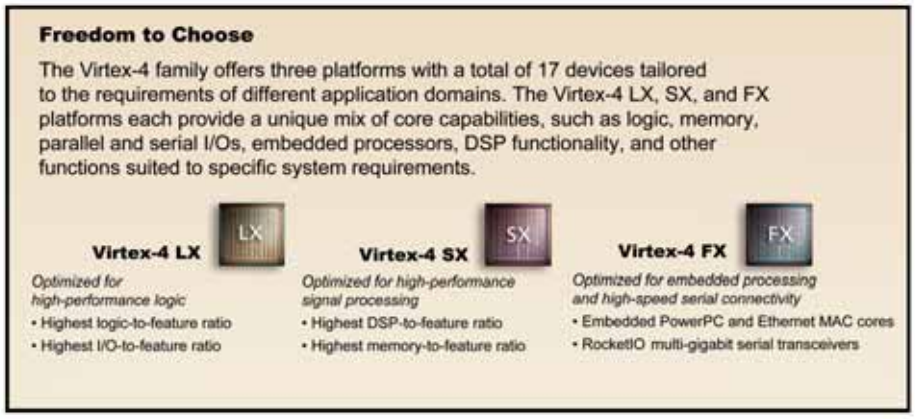


Figure 2 – One family, multiple platforms

Features at a Glance

- Largest logic capacity
  - Up to 200,000 logic cells
- Largest memory capacity
  - Up to 10 Mb block RAM
- Highest performance
  - 500 MHz Xesium clocking technology
  - Expanded clocking resources
  - Enhanced clock precision
  - Reduced clock jitter and skew
- Simplified source-synchronous interfacing
  - ChipSync technology
- Complete serial connectivity solution
  - 622 Mbps – 11.1 Gbps RocketIO transceivers
- Higher performance, low-power DSP
  - 500 MHz XtremeDSP slice
- Simplified processor acceleration
  - PowerPC 405 processor with auxiliary processor unit (APU) controller interface
- Integrated Ethernet MAC
- Fourth-generation design security

we have assembled three different platforms (Figure 2) with an initial offering of 17 devices that deliver cost-optimized solutions for the widest range of high-performance electronic systems.

- Integrated IP reduces the customer's bill of materials and saves FPGA resources.

Virtex-4 FPGAs reduce system cost with abundant integrated IP. By incorporating many functions that find use in a broad range of applications, Virtex-4 FPGAs replace a number of discrete components commonly found on system boards.

Designers can take advantage of embedded PowerPC™ processors, up to 10 Mb of embedded dual-port RAM/FIFO, integrated Ethernet MACs, sophisticated DSP circuitry, and on-board serial transceivers, among other features. This helps our customers lower system cost in several ways: by reducing component count and streamlining logistics with a smaller bill of materials; by simplifying the design and manufacturing of system hardware; by easing PCB design and manufacturing; and by improved system reliability through the reduction of solder joints.

In addition, building dedicated circuits on the FPGA provides required functionality efficiently, while preserving the programmable logic fabric for customers to add the value of their proprietary designs. The result is more capability within a single package at a given price point.

Up to 80% Additional Cost Reduction with EasyPath

The EasyPath™ program further lowers system cost for customers who are ready to take their finished design to volume production. Xilinx creates customized test programs for EasyPath customers that exercise only the device resources used in the specific design. This approach shortens test time and increases yield to reduce FPGA unit price up to 80%.

Source Synchronous Interfacing

To ensure reliable data transfer between a new generation of high-speed devices, hardware designers are turning to source-synchronous design techniques, in which the component sending the data generates and issues its own clock signal along with the data that it transmits. This technique eliminates one set of problems associated with parallel interfaces, but introduces its own circuit design challenges. ChipSync technology significantly simplifies component interface design with critical built-in circuitry that is available in every Virtex-4 I/O (see sidebar, "Virtex-4 Solves Source-Synchronous Design Challenges").

Embedded Processing

Embedded developers have already used Xilinx processor solutions to create thousands of designs. As we talked to these developers about the requirements for their next-generation systems, several common themes emerged.

A Full Range of Processing Solutions

Engineers need a range of processing solutions to match the requirements of different tasks, ranging from simple control functions to advanced algorithms and high-speed calculation. In addition, they want the different solutions to share a common design environment.

Xilinx satisfies these requirements with a range of processors that includes the

Virtex-4 FX devices include built-in Ethernet connectivity, enabling seamless chip-to-chip connections without consuming programmable logic resources.

PicoBlaze™ eight-bit microcontroller soft core, the MicroBlaze™ 32-bit general purpose processor soft core, and the industry-standard PowerPC architecture, in the form of a performance-optimized hard core.

Efficient Hardware Acceleration

Using an FPGA with an embedded processor as a platform for programmable system design enables flexible partitioning of functionality into hardware and software. Immersing the processor in the FPGA logic fabric opens the door to the additional flexibility of creating custom hardware to accelerate the execution of critical software. Hardware acceleration enables designers to apply logic resources to achieve performance exactly where needed.

Creating hardware (tightly coupled to the CPU) to act on a set of operands can accelerate the execution of key software by performing in a single cycle calculations that take many cycles on a processor. This performance boost is achieved by tuning the hardware design to provide the degree of parallelism required by the algorithm.

High Performance, Flexible Hardware Acceleration

Creating accelerators for FPGA-based processors requires three elements: programmable logic fabric for building the custom hardware; unassigned address space for the new instruction; and a low-latency path between the processor and the acceleration hardware. Xilinx provides the most efficient integration of microprocessor and FPGA fabric with dedicated interfaces that save clock cycles by eliminating bus overhead; are decoupled from the CPU to enable implementation of multiple accelerators; and do not stall the pipeline crucial to RISC performance.

All Virtex FPGAs have abundant programmable logic resources suitable for building acceleration hardware. Xilinx enables efficient accelerator integration for the MicroBlaze soft processor core with the Fast Simplex Link (FSL). The MicroBlaze processor supports up to 32 input and 32 output FSL, and code development is easy with simple programming for blocking and non-blocking instructions.

Virtex-4 FX devices include up to two PowerPC hard processor cores. Xilinx first introduced the immersed PowerPC 405 core in the Virtex-II Pro™ family. For the Virtex-4 family, Xilinx has increased processor performance to 680 DMIPS at 450 MHz and reduced power consumption to 0.44 mW/MHz while maintaining compatibility with all software and IP created for the first-generation core.

A new auxiliary processor unit (APU) controller simplifies the integration of acceleration hardware for the PowerPC core by providing a direct interface between the CPU pipeline and the FPGA logic fabric. This ultra-low-latency architecture enhances performance by reducing, by a factor of ten, the number of bus cycles needed to access the accelerator hardware. The net result is a 20-fold increase in processor-accelerator efficiency.

High-Speed Connectivity

When we asked system developers to describe their connectivity requirements, they highlighted the need for performance to support emerging standards and flexibility to upgrade today's designs to meet future bandwidth requirements. They are looking for solutions that offer bandwidth greater than 3.125 Gbps, provide complete support for multiple communication standards, and maintain the highest possible signal integrity.

Our third-generation RocketIO multi-gigabit transceiver satisfies these requirements with the industry's broadest operating range and other enhancements. Virtex-4 FX FPGAs enable bridging between just about any serial or parallel connectivity standard. For example, the third-generation RocketIO multi-gigabit transceivers provide compliance with the PCI Express standard, with support for out-of-band signaling (electrical idle and beaconing) and spread-spectrum clocking.

To address the challenges of backplane and other high-speed connectivity designs, RocketIO multi-gigabit transceivers provide comprehensive equalization techniques to ensure signal integrity in a wide variety of applications (Table 1). These advanced equalization techniques enable engineers to give new life to old systems by upgrading legacy backplanes.

- Third-generation multi-gigabit transceivers
- Operating range: 622 Mbps — 11.1 Gbps
- Channels: up to 24
- Transmit pre-emphasis
- Receive linear and decision feedback equalization (DFE)
- 8b/10b and 64b/66b encode/decode
- Sonet jitter compliant at OC-12 and OC-48 line rates

Table 1 – RocketIO features at a glance

In addition, Virtex-4 FX devices include built-in Ethernet connectivity, enabling seamless chip-to-chip connections without consuming programmable logic resources. The Ethernet MAC core supports 10/100/1000 Mbps data rates with UNH-verified standards compliance and interoperability.

High-Performance DSP

Developers told us they need to achieve higher DSP performance targets to implement next-generation applications such as MPEG-4 video compression/decompression and multi-channel mobile communications. Scaling existing DSP implementations to meet these targets with multiple programmable DSPs or dedicated ASIC hardware can be prohibitively expensive. Designers also need to control system power consumption as they squeeze more functionality into smaller form factors.

To address new DSP performance requirements, Xilinx crafted the versatile XtremeDSP slice, providing twice the DSP performance of previous implementations while drawing less than 1/7th of the power. Although all Virtex-4 FPGAs contain XtremeDSP slices, the Virtex-4 SX platform provides the highest ratio of XtremeDSP slices to other resources.

The largest SX device, the XC4VSX55, has 512 slices. Using these 500 MHz XtremeDSP slices with 18 x 18-bit multiplier and 48-bit accumulator exclusively, this device can achieve 256 GMAC/s performance at a very aggressive price point, providing the most powerful DSP capabilities of any FPGA in the industry. Demonstrating the revolutionary flexibility of the multi-platform approach enabled by the ASMBL architecture, the DSP-optimized SX55 offers ten times the DSP value, as measured in GMACs/dollar, compared with previous-generation FPGAs.

Xilinx is helping DSP developers close the gap between the performance of programmable single-MAC DSPs and the requirements of advanced algorithms with Virtex-4 SX platform FPGAs. Virtex-4 FPGAs can serve alongside programmable DSPs as pre-processors or co-processors to offload compute-intensive tasks.

Conclusion

To learn more about how you can take advantage of the breakthrough capabilities and performance of Virtex-4 FPGAs in your next system, please visit our website at *www.xilinx.com/virtex4/*.

Virtex-4 Solves Source-Synchronous Design Challenges

Source-synchronous interfaces typically send signals at bandwidths of up to 1 Gbps or higher on each channel. FPGA logic circuitry has difficulty processing incoming signals at that speed, so the frequency must be reduced by converting serial data on each channel to parallel data as it enters the device. Conversely, transmission requires converting parallel data to serial format. Traditionally, this process involves multiple stages of dividing down or multiplying up the speed. The steps required to meet the setup and hold requirements are laborious and time-consuming.

ChipSync technology simplifies design and boosts performance with an embedded SERDES that serializes and de-serializes parallel bus interfaces to match the data rate to the speed of the internal FPGA circuits. ChipSync technology enables data rates greater than 1 Gbps for differential I/O, and over 600 Mbps for single-ended I/O. This ability simplifies the design of interfaces such as SPI-4.2, XSBI, and SFI-4, as well as RapidIO™ and HyperTransport™.

Each channel and clock follows a slightly different route through the printed circuit board. Ensuring reliable data capture requires satisfying the setup and hold times of each channel. With communication interfaces of eight channels and higher, and with memory buses up to 144 bits wide, this can be an extremely challenging task.

ChipSync technology simplifies the implementation of communication and high-speed memory interfaces (including DDR 2 SDRAM, QDR II SRAM, FCRAM II, and RLDRAM II) by compensating routing issues that produce skew between data and clock signals. Built-in circuitry enables the delay of each data and clock channel within the SelectIO™ block, in 78 ps increments, to meet the setup and hold requirements for reliable data capture.

For extreme levels of skew, the misalignment might be greater than a bit interval. Aligning bits helps read the data reliably, but some channels might be out of step with others. To address extreme levels of skew, greater than a bit interval, ChipSync technology provides a bitslip capability. An optional training pattern simplifies the task of aligning data words across all channels.

With source-synchronous design, each interface has its own clock. As multiple interfaces and memories are connected to the same FPGA, the need for numerous flexible clock resources grows. With clock-aware I/Os, ChipSync technology enables simultaneous implementation of multiple source-synchronous interfaces.

Xesium clocking makes this possible with up to 24 clock regions per device. Each region can have up to six I/Os acting as clock sources for data capture. Up to 95 I/Os can be clocked by a single I/O clock, providing great clock flexibility and a large number of clocks.

What will happen when your FPGA and PCB designs meet?

Integrated Systems

Integrated Systems | The potential for disaster is enormous if your designers work independently of one another. Mentor Graphics eliminates that potential with the only integrated systems design solution in EDA. We have superior design solutions in both PCB and FPGA, so it stands to reason that we'd create the only truly integrated system flow. Our unique class of tools empowers designers to concurrently design FPGA's and PCB's. Increase your system performance and design team productivity. Get the systems integration white paper at **www.mentor.com/techpapers** or call **800.547.3000.**

Mentor Graphics®

Taking Digital Signal Processing to the Extreme

In this series on digital signal processing, the *Xcell Journal* spotlights the challenges of and solutions to developing extremely high-performance DSP applications.

by Omid Tahernia
Vice President and General Manager, DSP Division
Xilinx, Inc.
omid.tahernia@xilinx.com

Programmable DSP processors are growing tremendously and are being implemented in a variety of applications. To keep pace with this explosive growth, Xilinx® is expanding and enhancing its XtremeDSP™ processing solution. We've added dedicated DSP elements into our FPGAs to make it easier and more cost- and power-efficient to achieve performance levels previously only possible in custom ASICs.

The Xilinx solution is the perfect match with programmable DSP processors. The Xilinx tool suite allows you to very easily develop massively parallel digital signal processing engines that can do the "heavy lifting" in a complementary fashion to programmable DSPs.

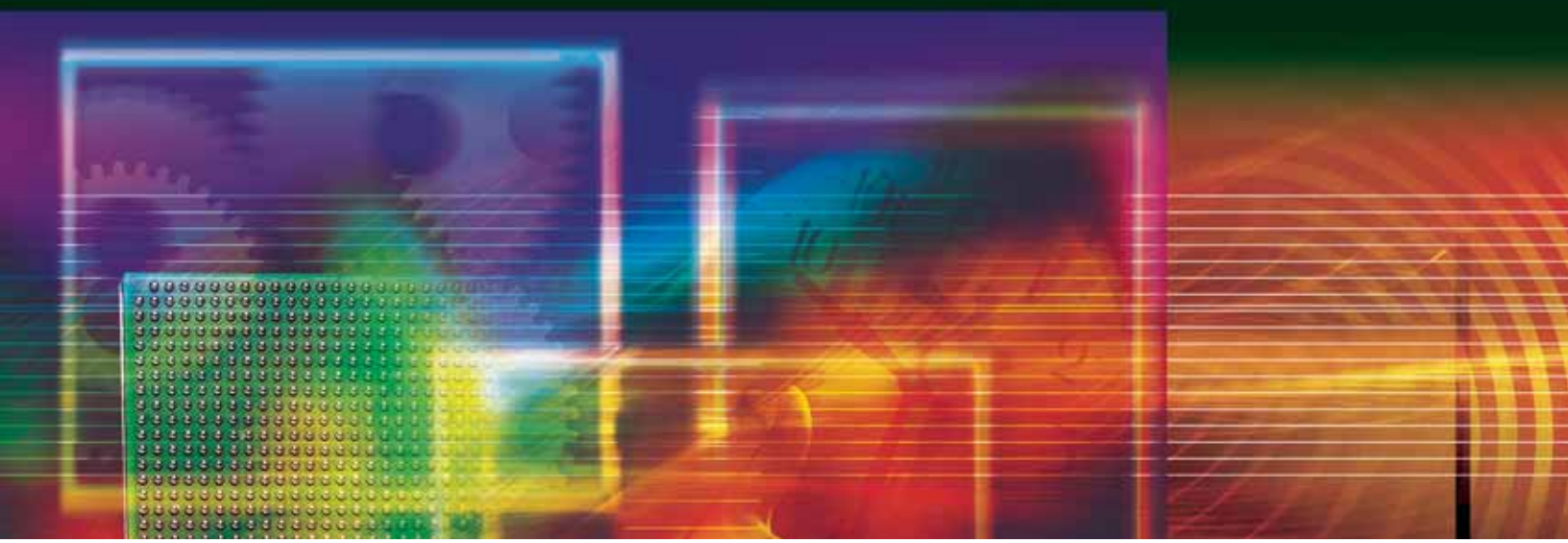
Providing extremely high-performance DSP solutions has become so important at Xilinx that we have recently created a DSP division. We are consolidating our DSP resources and creating focused development platforms and reference designs to help designers of high-performance DSPs get up to speed on our solution quickly and cost effectively.

In the following series, you will find articles on prototyping, developing, implementing, and analyzing in the context of real-world, high-performance DSP applications, using tools from Xilinx and our partners.

Table of Contents

Implementing the H.264/AVC Video Coding Standard on FPGAs

Xilinx Virtex FPGAs provide excellent co-, pre-, and post-processing hardware acceleration solutions.



by Wilson C. Chung
Senior Staff Video and Image Processing Engineer
Xilinx, Inc.
wilson.chung@xilinx.com

H.264/AVC is the latest international video coding standard in a series of such standards: H.261, MPEG-1, MPEG-2, H.263, and MPEG-4 visual, or part 2. It was approved by the ITU-T (International Telecommunications Union Telecommunication Standardization Sector) as recommendation H.264 and by ISO/IEC as International Standard 14 496-10 (MPEG-4 part 10) Advanced Video Coding (AVC) in May 2003.

Despite H.264/AVC's promises of improved coding efficiency over existing video coding standards, it still presents tremendous engineering challenges to system architects, DSP engineers, and hardware designers. The H.264/AVC standard brought in the most significant changes and algorithmic discontinuities in the evolution of video coding standards since the introduction of H.261 in 1990.

The algorithmic computational complexity, data locality, and algorithm and data parallelism required to implement the H.264/AVC coding standard often directly influences the overall architectural decision at the system level. In turn, this determines the ultimate cost of developing any commercially viable H.264/AVC system solution in the broadcasting, video editing, teleconferencing, and consumer electronics fields.

Complexity Analysis

To achieve a real-time H.264/AVC standard definition (SD) or high definition (HD) resolution encoding solution, system architects often employ multiple FPGAs and programmable DSPs. To illustrate the enormous computational complexity required, let's explore the typical run-time cycle requirements of the H.264/AVC encoder based on the software model provided by the Joint Video Team (JVT), comprising experts from ITU-T's Video Coding Experts Group (VCEG) and ISO/IEC's Moving Picture Experts Group (MPEG).

Using Intel™ VTune™ software running on an Intel Pentium™ III 1.0 GHz general-purpose CPU with 512 MB of memory, achieving H.264/AVC SD with a main profile encoding solution would require approximately 1,600 BOPS (billions of operations per second).

Table 1 illustrates a typical profile of the H.264/AVC encoder complexity based on the Pentium III general-purpose processor architecture. Notice that in Table 1, motion estimation, macroblock/block processing (including mode decision), and motion compensation modules are the primary candidates for hardware acceleration.

Functional Blocks	% of Run-Time Total Cycles
mv_search.c	67.31 %
block.c	8.19 %
refbuf.c	6.95 %
macroblock.c	3.48 %
rdopt.c	3.37 %
biariencode.c	3.21 %
cabac.c	2.98 %
memcpy.asm*	2.91 %
abs.c*	0.57 %
image.c	0.54 %
rdopt_coding_state.c	0.46 %
loopFilter.c	0.03 %

Table 1 – H.264/AVC encoder complexity profile by files

However, computation complexity alone does not determine if a functional module should be mapped to hardware or remain in software. To evaluate the viability of software and hardware partitioning of the H.264/AVC coding standard implementation on a platform that consists of a mixture of FPGAs, programmable DSPs, or general-purpose host processors, we need to look at a number of architectural issues that influence the overall design decision.

- Data locality. In a synchronous design, the ability to access memory in a particular order and granularity while minimizing the number of clock cycles due to latency, bus contention, alignment, DMA transfer rate, and the types of memory used (such as ZBT memory, SDRAM, and SRAM) is very important. The data locality issue is primarily dictated by the physical interfaces between the data unit and the arithmetic unit (or the processing engine).
- Data parallelism. Most signal processing algorithms operate on data that is highly parallelizable (such as FIR filtering). Single instruction multiple data (SIMD) and vector processors are particularly efficient for data that can be parallelized or made into a vector format (or long data width). FPGA fabric exploits this by providing a large amount of block RAM to support numerous very high aggregate bandwidth requirements. In the new Xilinx Virtex-4™ SX device family, the amount of block RAM matches closely with the number of Xtreme DSP™ slices (SX25 – 128 block RAM, 128 DSP slices; SX35 – 192 block RAM, 192 DSP slices; SX55 – 320 block RAM, 512 DSP slices).
- Signal processing algorithm parallelism. In a typical programmable DSP or a general-purpose processor, signal processing algorithm parallelism is often referred to as instruction level parallelism (ILP). A very long instruction word (VLIW) processor is an example of such a machine that exploits ILP by grouping multiple instructions (ADD, MULT, and BRA) to be executed in a single cycle. A heavily pipelined execution unit in the processor is also an excellent example of hardware that exploits the parallelism. Modern programmable DSPs have adopted this architecture (including the Texas Instruments™ TMS320C64x).

  However, not all algorithms can exploit such parallelism. Recursive algorithms like IIR filtering, variable-length coding (VLC) in MPEG1/2/4, context-adaptive variable length coding (CAVLC), and context-adaptive binary arithmetic coding (CABAC) in H.264/AVC are particularly sub-optimal and inefficient when mapped to these programmable DSPs. This is because data recursion prevents ILP from being used effectively. Instead, dedicated hardware engines can be built efficiently in the FPGA fabric.
- Computational complexity. Programmable DSP is bounded in computational complexity, as measured by the clock rate of the processor. Signal processing algorithms implemented in the FPGA fabric are typically computationally intensive. Some examples of these are the sum of absolute difference (SAD) engine in motion estimation and video scaling.

  By mapping these modules onto the FPGA fabric, the host processor or the programmable DSP has the extra cycles for other algorithms. Furthermore, FPGAs can have multiple clock domains in the fabric, so selective hardware blocks can thus have separate clock speeds based on their computational requirements.
- Theoretic optimality in quality. Any theoretic optimal solution based on the rate-distortion curve can be achieved if and only if the complexity is unbounded. In a programmable DSP or general-purpose processor, the computational complexity is always bounded by the clock cycles available. FPGAs, on the other hand, offer much more flexibility by exploiting data and algorithm parallelism by means of multiple instantiations of the hardware engines, or increased use of block RAM and register banks in the fabric.

  A programmable DSP or general-purpose processor is often limited by the number of instruction issues per cycle, the level of pipeline in the execution unit, or the maximum data width to fully feed the execution units. Video quality is often compromised as a result of the limited cycles available per task in a programmable DSP, whereas hardware resources are fully allocated in FPGA fabric (three-step vs. full-search motion estimation).

Implementing Functional Modules onto FPGAs

Figure 1 shows the overall H.264/AVC macroblock level encoder with major functional blocks and data flows defined. One of the primary successes of the H.264/AVC standard is its ability to predict the values of the content of a picture to be encoded by exploiting the pixel redundancy in different ways and directions not exploited previously in other standards. Unfortunately, when comparing to previous standards, this increases the complexity and memory access bandwidth approximately four-fold.

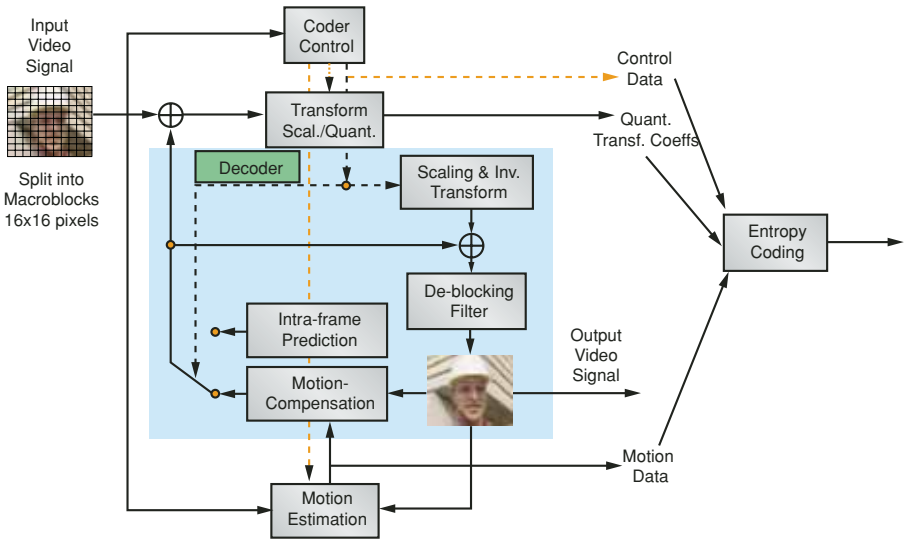


Figure 1 – H.264/AVC macroblock encoder with functional blocks and data flows

Improved Prediction Methods

Let's highlight some of the main features of the H.264/AVC video coding standard design that enable its enhanced coding efficiency, evaluating these functional modules based on the design criteria discussed in the previous section.

- Quarter-pixel-accurate motion compensation. Prior standards use half-pixel motion vector accuracy. The new design improves on this by providing quarter-pixel motion vector accuracy. The prediction values at half-pixel positions are calculated by applying a one-dimensional six-tap FIR filter [1, -5, 20, 20, -5, 1]/32 horizontally and vertically.

  Prediction values at quarter-pixel positions are generated by averaging samples at the full- and half-pixel positions. These sub-sampling interpolation operations can be efficiently implemented in hardware inside the FPGA fabric.

- Variable block-sized motion compensation with small block size. The standard provides more flexibility for the tiling structure in a macroblock size of 16 x 16 pixels. It allows the use of 16 x 16, 16 x 8, 8 x 16, 8 x 8, 8 x 4, 4 x 8, and 4 x 4 sub-macroblock sizes.

  Because of the increasing combinations of tiling geometry with a given 16 x 16 macroblock, to find a rate distortion optimal tiling solution is extremely computationally intensive. This additional feature places an enormous burden on the computational engines used in motion estimation, refinement, and mode decision process.

- In-the-loop adaptive deblocking filtering. The deblocking filter has been successfully applied in H.263+ and MPEG-4 part 2 implementations as a post-processing filter. In H.264/AVC, the deblocking filter is moved inside the motion-compensated loop to filter block edges resulting from the prediction and residual difference coding stages of the decoding process. The filtering is applied on both 4 x 4 block and 16 x 16 macroblock boundaries, in which two pixels on either side of the boundary may be updated using a three-tap filter. The filter coefficients or "strength" are governed by a content-adaptive non-linear filtering scheme.

- Directional spatial prediction for intra coding. In cases where motion estimation cannot be exploited, intra-directional spatial prediction is used to eliminate spatial redundancies. This technique attempts to predict the current block by extrapolating the neighboring pixels from adjacent blocks in a defined set of directions. The difference between the predicted block and the actual block is then coded.

  This approach is particularly useful in flat backgrounds where spatial redundancies exist. There are a total of nine prediction directions for Intra_4x4 prediction, and four prediction directions for Intra_16x16 prediction. Note that the data causality imposes quick memory access to the neighboring 13 pixel values to the above and left of the current block in the case of Intra_4x4. For the Intra_16x16, 16 neighboring pixels on each side are used to predict a 16 x 16 block.

- Multiple reference picture motion compensation. The H.264/AVC standard offers the option for multiple reference frames in the inter-frame coding. Unless the number of the referenced pictures is one, the index at which the reference picture is located inside the multi-picture buffer has to be signaled. The multi-picture buffer size determines the memory usage in the encoder and decoder. These reference frame buffers must be addressed correspondingly during the motion estimation and compensation stages in the encoder.

- Weighted prediction. The JVT recognizes that in encoding certain video scenes that involve fades, having a weighted motion-compensated prediction dramatically improves the coding efficiency.

Improved Coding Efficiency

In addition to improved prediction methods, other parts of the standard design were also enhanced for improved coding efficiency. Two additional features are most likely to impact the overall system architecture based on our design criteria for software and hardware partitioning:

- Small block size, hierarchical, exact-match inverse, and short word-length transform. The H.264/AVC, like other standards, also applies transform coding to the motion-compensated prediction

residual. But, unlike previous standards that use an 8 x 8 discrete cosine transform (DCT), this transform is applied to 4 x 4 blocks, and is exactly invertible in a 16-bit integer format. The small block helps reduce blocking and ringing artifacts, while the precise integer specification eliminates any mismatch

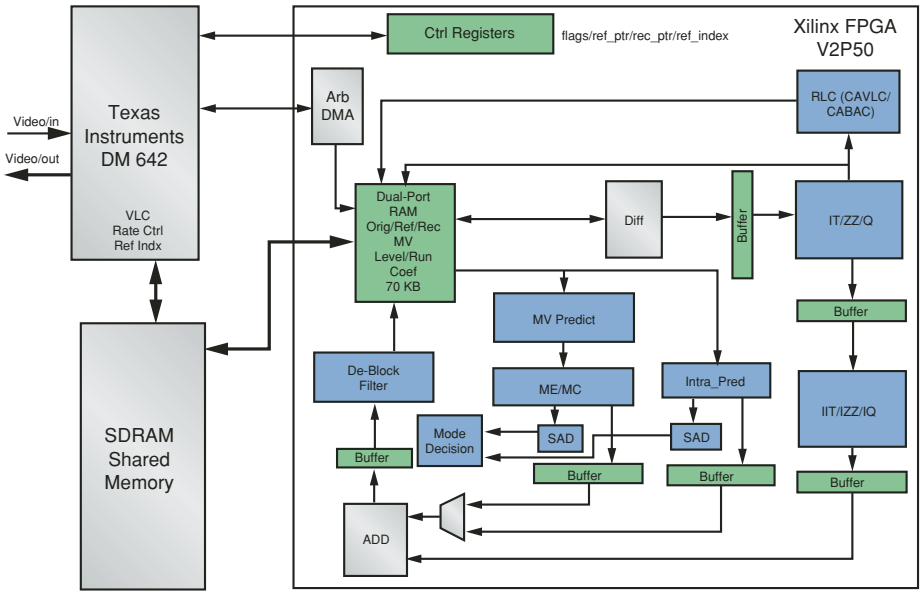


Figure 2 – Typical H.264/AVC hardware/software functional block partition
Diagram courtesy of W&W Communications

issues between the encoder and decoder in the inverse transform.

Furthermore, an additional transform based on the Hadamard matrix is also used to exploit the redundancy of 16 DC coefficients of the already transformed blocks. Compared to a DCT, all applied integer transforms have only integer numbers ranging from -2 to 2 in the transform matrix. This allows you to compute the transform and the inverse transform in 16-bit arithmetic using only low-complexity shifters and adders.

- Arithmetic and context-adaptive entropy coding. Two methods of entropy coding exist: a low-complexity technique based on the use of context-adaptively switched sets of variable length codes (CAVLC) and the computationally more demanding algorithm of context-based adaptive binary arithmetic coding (CABAC). CAVLC is the baseline entropy coding method of H.264/AVC. Its basic coding tool consists of a single VLC of structured Exp-Golomb codes, which by means of individually customized mappings are applied to all syntax elements except those related to the quantized transform coefficients. For the CABAC, a more sophisticated coding scheme is applied. The transform coefficients are first mapped into a 1-D array based on a predefined scan pattern. After quantization, a block contains only a few significant non-zero coefficients.

  Based on this statistical behavior, five data elements are used to convey information of the quantized transform coefficients for a luminance 4 x 4 block. The efficiency of entropy coding can be improved further if using CABAC.

  There are two parts in CABAC. The arithmetic coding core engine and its associated probability estimation are specified as multiplication-free low-complexity methods using only shifts and table look-ups. The use of adaptive codes allows it to adapt to non-stationary symbol statistics. By using context modeling based on switching between conditional probability models that are estimated from previous coded syntax elements, CABAC can achieve a reduction in bit rate between 5-15% compared to CAVLC.

Figure 2 depicts a typical system-level functional block partition of the H.264/AVC SD video codec. The solution is implemented based on the Spectrum Digital EVM DM642 evaluation module for the Texas Instruments TMS320DM642 DSP, together with the Xilinx XEVM642-2VP20 Virtex-II Pro™ or XEVM642-4VSX25 Virtex-4™ daughtercard.

Conclusion

When used in an optimized fashion, the coding tools of the H.264/AVC standard increase coding efficiency by about 50% compared to previous video coding standards (like MPEG-4 part 2 and MPEG-2) for a wide range of bit rates and resolutions. Currently, it is the most likely successor to the widely used MPEG-2. However, the algorithm is quite complex, at a resolution greater than source input format (SIF).

The DVD-Forum, with its HD-DVD initiatives, has selected H.264/AVC together with WMV-9 and MPEG-2 as the standard video coding formats. The European DVB consortium has also selected H.264/AVC as the next format after MPEG-2. These announcements, plus endorsements from Hollywood studios, content distributors, and broadcast infrastructures, have further validated the importance of the H.264/AVC video coding standard for the next few years.

For more comprehensive studies and technical details of the H.264/AVC video coding standard, please see the References.

References

"Draft ITU-TU Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC)," in Joint Video Team (JVT) of ISO/IEC MPEG and ITU-T VCEG, JVT-G050, 2003.

A. Luthra, G.J. Sullivan, and T. Wiegand. July 2003. "Special Issue on The H.264/AVC Video Coding Standard." *IEEE Trans. Circuits System Video Technology* 13(7): 557-725.

XtremeDSP

Developing a GSM Modem on a DSP/FPGA Architecture

Using System Generator and Simulink, you can create a seamless simulation-to-implementation FPGA design flow.

by Louis Belanger
Product Development Manager
Lyrtech Signal Processing, Inc.
louis.belanger@lyrtech.com

GSM (Global System for Mobile) is the most widely-used cellular phone technology. Having begun mostly as a European standard, it has spread throughout the world to become ubiquitous.

GSM was one of the first digital cellular systems, and as such, represented another level of magnitude in terms of complexity, with support for growth features such as GPRS (General Packet Radio Service), which provides data capabilities to GSM phones. In this context, designing a GSM system is a complex task and could benefit from advanced design flow techniques where initial system simulation phases can be seamlessly carried over to the implementation phases.

In this article, we'll describe such a design flow for GSM development, starting from research and model simulation/implementation in The MathWorks MATLAB® to FPGA implementation through simulation phases in The MathWorks Simulink® and Xilinx® System Generator.

Project Context

Our implementation is in the general context of wireless application development examples to showcase our DSP/FPGA platforms. A previous project, the implementation of a SSB (single side band) AM radio (also with The MathWorks) featured a simpler analog radio and was showcased at Xilinx Programmable World 2003.

We wanted to implement a much more complex radio, and so we selected the GSM digital cellular standard. In doing so, we are getting closer to our goal to design ever-more-complex wireless systems for our customers.

We wanted to have a simplified system that still operates as a GSM system, while demonstrating a Model-Based (also known as system-level) Design flow. The target platform is our SignalMaster DSP-FPGA, with high-speed sampling boards to sample the intermediate frequency (IF) coming from a special-purpose radio frequency (RF) front end. We performed IF processing in the Xilinx Virtex-II™ FPGA and developed the design using System Generator.

In a complementary manner, baseband processing implementation occurs in the DSP, using a similar Simulink design flow with the The MathWorks Real-Time Workshop™ C-code generator, Embedded Target for TI DSP toolbox, and LYRtech's GSM DSP libraries.

We designed protocol-oriented transactions using The MathWorks Stateflow™, a

tool that allows the graphical design of states and transitions, as well as the production of associated code. This event-driven code can run on the DSP itself or on a companion RISC processor.

GSM Processing

Figure 1 depicts a GSM physical channel. There are 124 200-kHz channels that are frequency multiplexed in a 25-MHz-wide RF spectrum, one for each downlink and uplink path. Figure 1 also shows in more detail how the "bursts" of each GSM channel are constructed.

Basically, each burst is part of an 8-slot time division multiplex frame, forming a 200-kHz-wide spectrum. Each one has tail bits and an extended guard interval to avoid interference, as long as the mobile station (MS) is within 35 km of the base station (BS). Some fixed training-bit sequences allow synchronization between the MS and the BS.

Using the GSM as a design example corresponds very well to our platform's segmented architecture. The main uplink physical layer elements of a GSM speech and data transmission chain are shown in Figure 2. Figure 2 also illustrates how to partition the processing.

We performed IF processing on the FPGA with functions such as polyphase DDC (digital down converter), DDS (direct digital synthesis), and GMSK (Gaussian Minimum Shift Keying) modulation. DSP-based baseband processing can tackle the tasks of encoding, encrypting, and interleaving, as well as burst building functions. Finally, communication protocol handling occurs at the RISC processor level (or in the DSP for simplified protocols, such as in our case).

Simulink DSP Model

The DSP section of the GSM model contains the following elements:

- All higher level protocol layers
- Baseband processing modules
- Data transfer protocol for mapping data onto 32-bit frames, before sending to the FPGA through the parallel bus interface

Figure 3 displays the DSP model, which contains the baseband processing block and Stateflow diagrams. This figure shows a combination of The MathWorks' MATLAB off-the-shelf functions and target-specific library blocks.

This is very typical of a Model-Based Design flow, in which target-specific block performance is compared with pure simulation blocks. At target compilation time, the associated DSP library of these last blocks can build the DSP code, providing very efficient code generation and performance.

FPGA Processing Model-Based Design

Figure 4 illustrates the main FPGA-based Simulink model for the base station. On the transmit side, the FPGA receives 148-bit GSM frames from the DSP as input, modulates them to get a GMSK burst, and then frequency-shifts the resulting signal in the 70 MHz IF band through SSB modulation, using the Xilinx direct digital synthesizer (DDS) block.

Two FDM transmit channels are simulated in this model. These two signals are then mixed together on the same physical channel and sent to the digital-to-analog converter (DAC); that block is located in the Signal I/O and Mixer subsystem.

On the receiver side, signals feeding the FPGA come from the analog-to-digital converter (ADC). Because the digitized signal is 25 MHz wide and can contain as many as 124 GSM channels, channel selection is required. This is performed by a

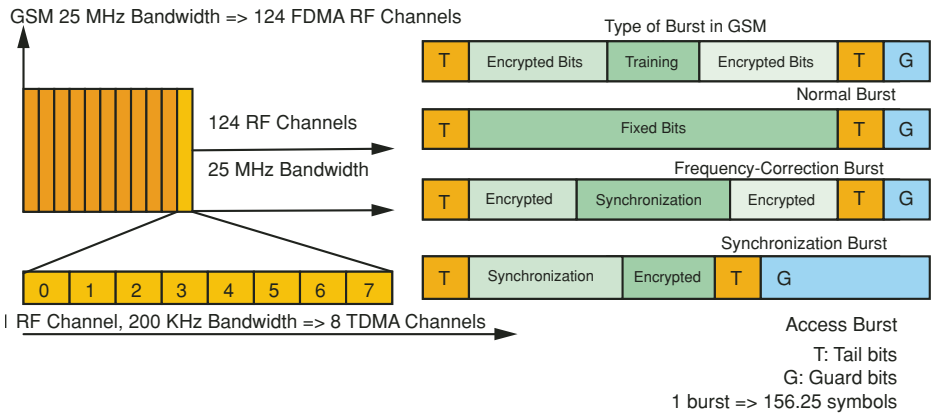


Figure 1 – GSM FDM, TDM, and burst structure

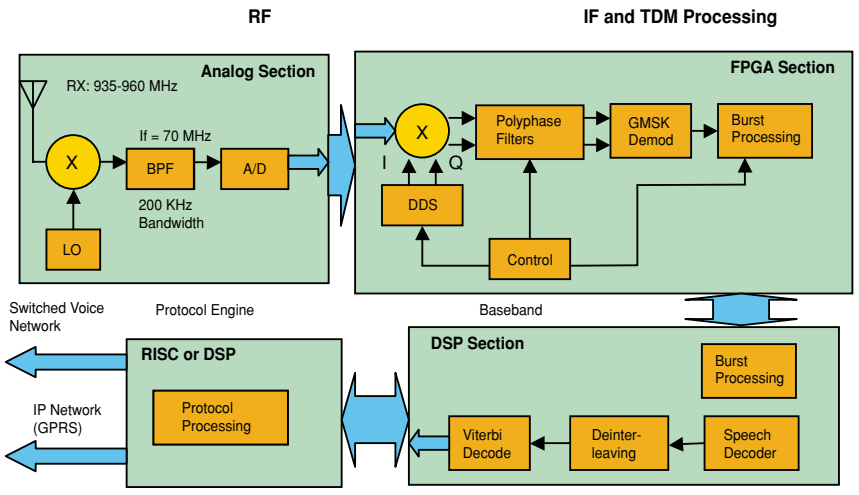


Figure 2 – Partitioning of the GSM processing between the FPGA, the DSP, and a RISC processor

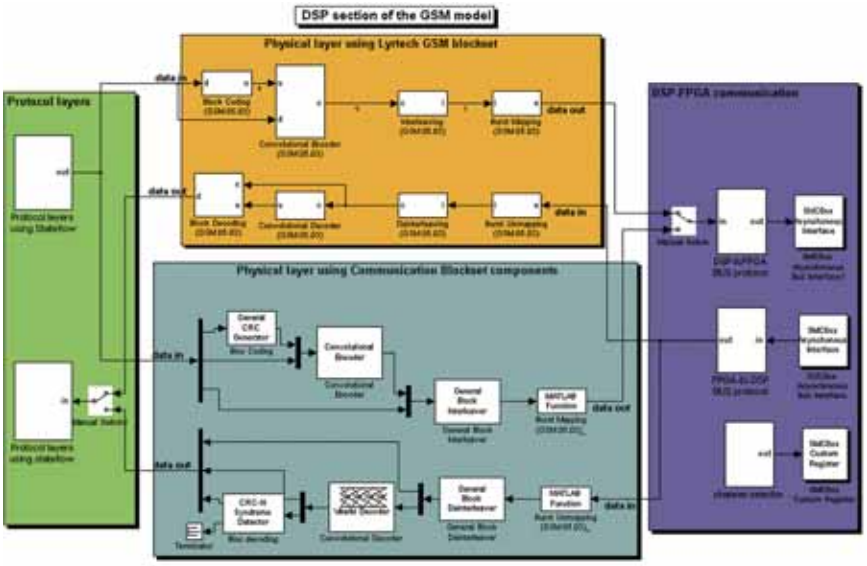


Figure 3 – DSP model of baseband GSM processing with comparative host/target blocksets

DDS in the IF demodulator subsystem; that frequency is dictated by a value coming from the DSP. The baseband signal can now be GMSK demodulated and sent for further processing.

This model runs in one of three different ways:

- Normal mode (simulation)
- Co-simulation mode (hardware-in-the-loop)
- Real-time mode (100% hardware)

In normal mode, the data comes from the block located in the DSP section. This block contains the DSP functions shown in Figure 2, in order to provide frames to the transmitter (the same applies for the receiver side).

Simulink performs all processing during simulation. The gateway blocks have no effect, except for converting the signal from one format to another (such as from double to fixed-point). The Signal I/O and Mixer subsystem then produces a loopback of the IF signal and adds white Gaussian noise to it.

The co-simulation mode basically performs in the same way as the normal mode, except that the processing function between the gateways will be executed in the FPGA as a hardware-in-the-loop simulation.

In real-time mode, all the blocks outside the gateways are ignored. These gateways establish the communication between the different hardware entities. In the case of our GSM model, the frames travel from DSP to FPGA and vice-versa through the parallel 32-bit data bus.

The IF signal now travels through the DAC and can either be in loopback mode, if the output of the DAC is connected to the input of the ADC, or fed to the front-end to produce an RF signal.

Co-Simulation Benefits

We designed the first version using only standard communication and DSP blocksets from Simulink, running in double precision from start to end. As a second step, we gradually replaced Simulink blocks with ones from Xilinx, and tested the model in normal mode, which resulted in hybrid models and simulations that were easier to debug.

After producing the Xilinx model, we could test it in co-simulation to verify that hardware computation was as expected. As a final step, we targeted the whole process to hardware.

Demodulator Subsystem Complexities

The subsystem presented in Figure 5 is the IF-to-baseband demodulator, which is the most complex part of the design. Before

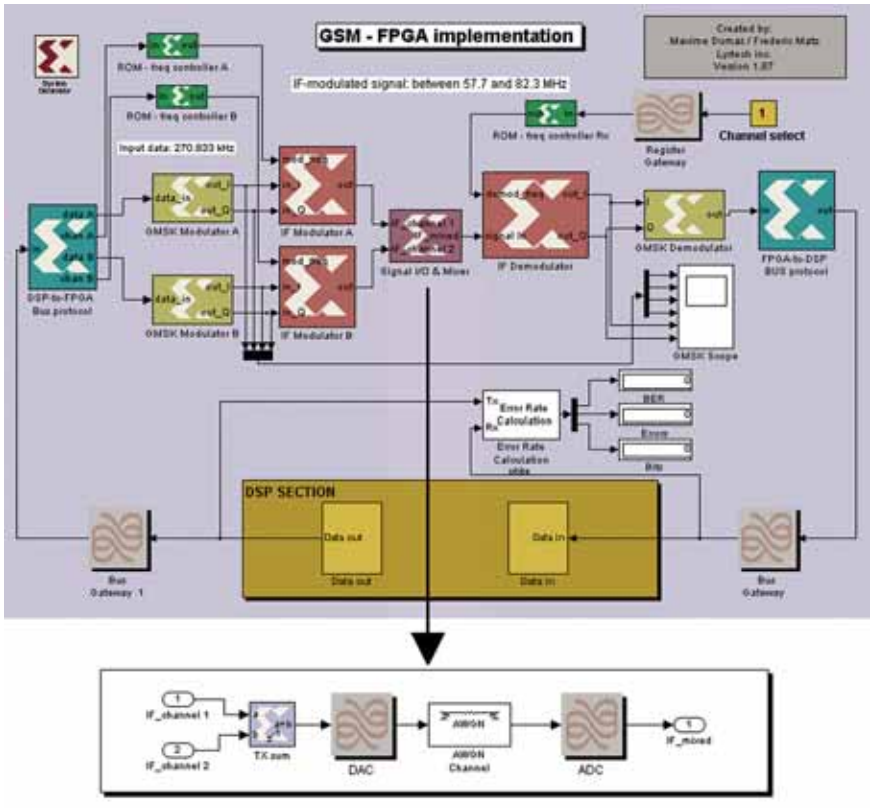


Figure 4 – FPGA model/IF processing of GSM

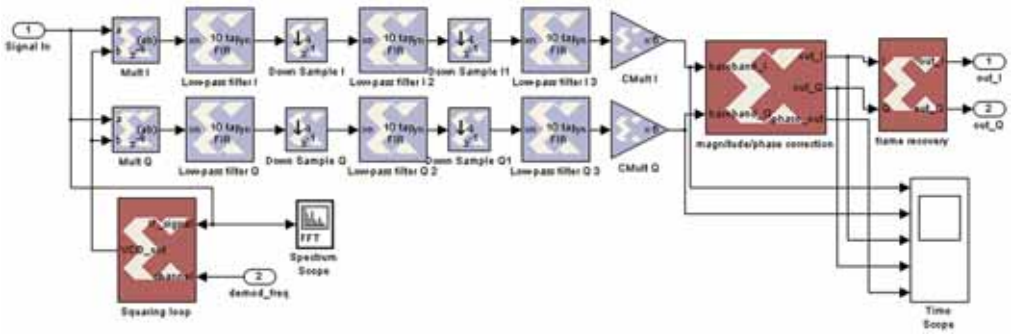


Figure 5 – IF FPGA model/demodulator subsystem

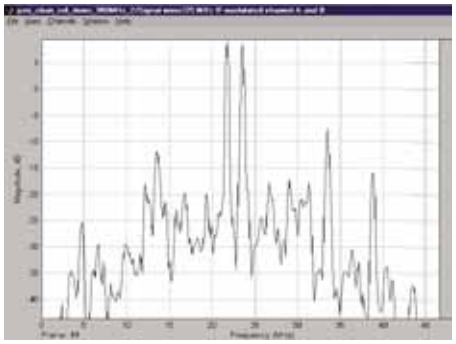


Figure 6 – Spectrum scope displaying the two GSM FDMA channels

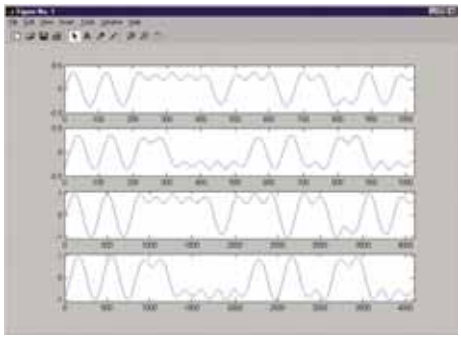


Figure 7 – Time scope displaying I/Q signals before and after the magnitude/phase correction operation

the GMSK demodulator receives the signal, the receiver has more to do than just blindly shift and down-sample the signal from IF to baseband. It must compensate for the effects of the channel on the signal, which means it has to perform the following:

- Carrier frequency recovery
- Carrier phase recovery
- Amplitude adjustment
- Timing recovery

We created low-pass filters using the digital filter design block from the MATLAB DSP blockset, which were later replaced by the Xilinx FIR filters that use the same taps generated by the Simulink block. Once the demodulator was functional for ideal signals, we added correction blocks to cope with non-ideal signals.

You can see in Figure 5 that a squaring loop deals with carrier recovery, while the magnitude/phase correction block takes care of the remaining amplitude and phase errors with trigonometric properties of a quadrature signal. We performed a cross-correlation with the expected training sequence to recover the timing and send a complete frame to the GMSK demodulator.

Hybrid FPGA/Simulink modeling was very useful in the development of this subsystem, because it was possible to visualize the signals at every step of the processing, both in time and frequency. Figure 6 displays the spectra of the two FDMA channels before demodulation and channel selection, while Figure 7 shows the waveform obtained before and after the magnitude/phase correction block.

FPGA Resource Estimation

Table 1 shows the resources used in the Virtex-II FPGA.

The IF-baseband demodulation is based mainly on a three-stage decimation and filtering applied to both I and Q signals, each using three 10-tap FIR filters.

Basically, the implementation of the GMSK demodulator brings together three major components: the phase recovery module, the timing recovery module, and the MLSE (maximum likelihood sequence estimation). The phase recovery module uses some dividing and square-root operators, which are costly to implement. The timing recovery is based mostly on the correlation of large data sequences that demand many embedded multipliers, and also some large data buffers made out of block RAM.

The implementation of an MLSE comprises a modified version of the Viterbi algorithm and demands considerable resources. This full-feature demodulator can be optimized, simplified, or targeted at an ASIC, but as a first-pass iteration, it provides a good estimation of the resources needed for its implementation.

	Slices	Total Slices (%) Virtex-II XC2V3000
GMSK modulator	190	1.33 %
IF-baseband modulation/demodulation	6,567	45.80 %
GMSK demodulator	4,775	33.30 %
BUS gateway and protocol	382	2.67 %
TOTAL :	**11,914**	**83.10 %**

Table 1 – Resources used in the FPGA

Conclusion

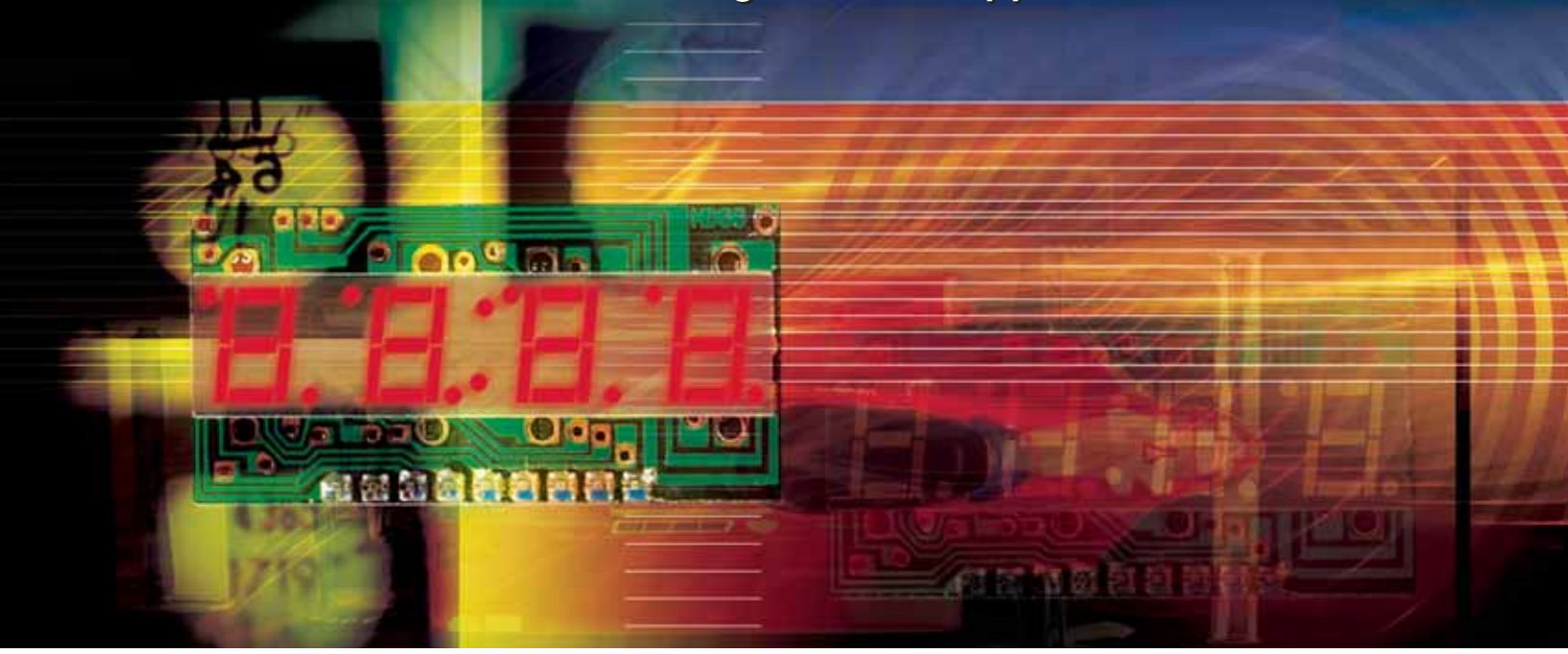
A Model-Based Design approach in the development of a complex wireless application for a DSP/FPGA architecture is very effective for thoroughly testing a design while implementing it for target hardware. Nonetheless, the use of FPGA cores and DSP libraries also allows the implementation to be quite efficient, even with a high-level design approach. Combined with flexible platforms such as the SignalMaster, these tools and approaches really help today's designers tackle the difficult challenges of designing state-of-the art wireless systems.

For additional information on this project and our SignalMaster line of DSP/FPGA development platforms, visit *www.lyrtech.com.*

XtremeDSP

Hardware Design Kits Turbo-Charge DSP Co-Processing Applications

Xilinx and Avnet have released new design kits that reduce time to market for a wide range of DSP applications.



by Warren Miller
VP of Marketing, Avnet Design Services
Avnet
warren.miller@avnet.com

Traditionally, designs for a variety of applications used dedicated digital signal processing (DSP) chips or application-specific standard products (ASSPs) to process digital information using signal processing algorithms. Filtering, video processing, and audio processing were just a few of the many applications using digital signal processors.

Now, with performance and capacity improvements to FPGAs, as well as the improved efficiency of common arithmetic operations usually found in most DSP applications, FPGAs doing DSP functions are becoming more common. In many cases both processors and FPGAs are used in the same application, in a co-processing architecture where the FPGA does pre- or post-processing to accelerate processing speed.

DSP applications are usually difficult to verify via software simulation because of the enormous number of cycles required to process a meaningful data stream; thus, it is usually better to use a hardware development platform to prove out the key parts of a new design. The new DSP design kits from Avnet provide a powerful, flexible, and expandable platform to validate even the most complex signal processing designs that use both FPGAs and DSPs.

Avnet DSP Design Kits

Avnet Design Services has created a variety of DSP-oriented design kits for use with Xilinx® FPGAs and Texas Instruments™ (TI) DSPs. The Spartan-3™-based design kit is optimized for simple video applications, while the Spartan-IIE™-based kit is targeted at audio applications.

The Virtex-II Pro™ kit features an adaptor card that interfaces to TI DSPs and is meant for co-processing applications where the FPGA is offloading significant processing and control functions from the digital signal processor. Each of these design kits also includes a variety of software tools from Xilinx, The MathWorks, and TI. Let's describe these kit components in more detail.

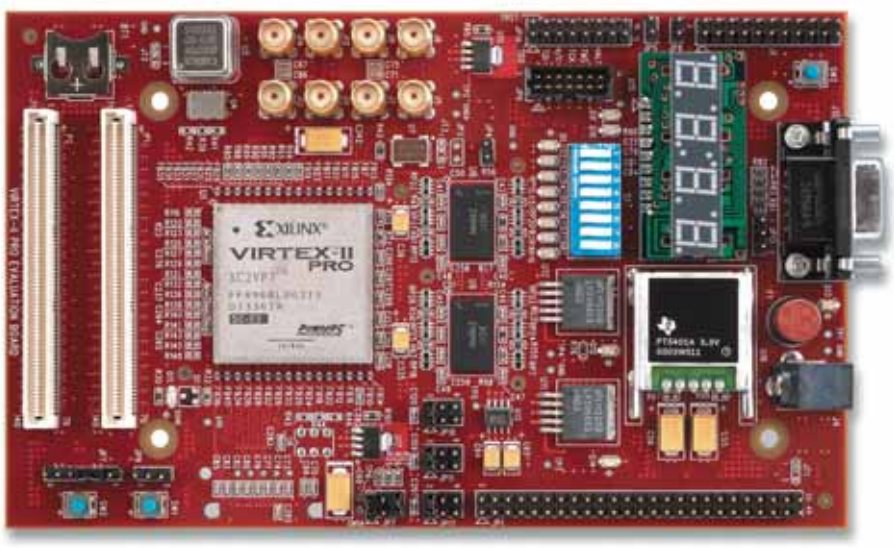


Figure 1 – DSP Co-Processing Design Kit hardware platform

DSP Co-Processing Design Kit

The DSP Co-Processing Design Kit features a Virtex-II Pro evaluation board, shown in Figure 1. This board contains a Xilinx XC2VP7-FF896 FPGA, eight SMA connectors for high-speed I/O, on-board DDR SDRAM (64 MB), up to 30 LVDS pairs, user I/O switches/LEDs, and several expansion connectors.

Two of the expansion connectors are compatible with the TI adaptor daughtercard (shown in Figure 2) and can connect to TI DSPs. Example designs show how to interface directly with the TI processor using the Xilinx EDK toolset and a direct memory interface approach.

A co-processing-oriented application can use the hardware platform, demonstration designs, and included tools as a great starting point for prototype design and algorithm development. DSP applications are often very difficult to simulate in software, so the ability to quickly create a hardware/firmware/software platform can cut development time significantly. Using the co-simulation tools available in the Xilinx tool suite through The MathWorks Simulink™ and the target hardware is one technique that can dramatically reduce design time.

Additionally, deciding what portions of the algorithm to process in the DSP and which portion to process in the FPGA can often best be done with a trial-and-error approach, using real hardware to quickly evaluate the performance of various options. For example, the number of data streams that can be pre-processed by an FPGA before post-processing by a DSP will depend on many factors – the "burstiness" of the incoming data, the "accept" response rate of the DSP, the size of the buffer memories, the bandwidth of the system bus, and the amount of pre-processing allocated to the FPGA. These are all difficult decisions to make without doing some detailed hardware prototype-based analysis.

The DSP Co-Processing Design Kit also includes the following software tools, as evaluation versions, from the Xilinx XtremeDSP™ Software Evaluation CD Kit: Xilinx ISE 6.2 Foundation™, ChipScope™ Pro, Xilinx System Generator for ISE 6.2, The MathWorks MATLAB™, and Simulink.

Video DSP Design Kit

The Video DSP Design Kit targets simple DSP-oriented video applications in the industrial security, consumer, and automotive markets. Algorithms for video processing like image recognition, video encode, video decode, and video image enhancement are all very difficult to prototype and evaluate without actual hardware on which to run the software or firmware. Using a DSP Design Kit, with some simple video capabilities, can make it much easier and quicker to prototype and evaluate various algorithms and architecture alternatives.

The Video DSP Design Kit features a Xilinx Spartan-3 XC3S400-FG456 or XC3S1500-FG456 FPGA, Platform Flash configuration PROM, expansion connectors, 32-bit PCI edge connector, 10/100 Ethernet port, video DAC, RS-232 console, PS2 keyboard and mouse ports, simple analog I/O, 1 MB SRAM, 256 Kb serial EEPROM, and a variety of user switches and LEDs.

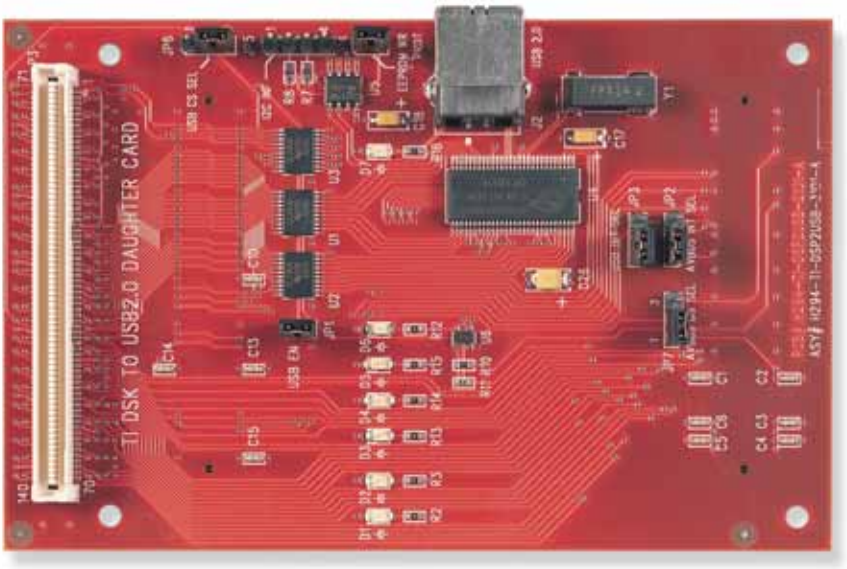


Figure 2 – DSP processor adaptor module

The kit also includes example designs and user documentation to make it easy to get started on a new video DSP design. Several Xilinx application notes and reference designs (some using Xilinx IP cores available from the DSP System Generator tool) are available online to provide even more of a head start (see Table 1). The Xilinx DSP Central website (*www.xilinx.com/products/design_resources/dsp_central/grouping/index.htm*) has a complete list.

Audio DSP Design Kit

The Audio DSP Design Kit is similar to the Video DSP Kit, but is optimized for audio processing applications. The kit features a Spartan-IIE hardware board with an XC2S200E-6FT256 FPGA, a TI TLV320AIC23 16-bit audio CODEC, RS-232 port, LEDs and switches, and several expansion connectors.

Customize Your Platform

If the Audio and Video DSP Design Kits are not quite what you need for your design, you can add more hardware, firmware, or software to create a custom platform. Avnet has a variety of hardware add-in modules that can serve as extensions to the basic platform.

Audio/Video Add-in Module

The Audio/Video Module provides additional functionality for DSP applications targeting audio and video processing applications. It interfaces to a host through a standard AvBus connector and provides multiple video interfaces to accommodate RGB monitors, LCD panels, and standard definition television monitors. The module also captures composite video and includes a CODEC to facilitate audio processing. A PS2 keyboard/mouse interface is included as well as a touchscreen controller.

Key elements of the module are:

- Philips™ SAA7113H video input processor
- Philips UCB1400 stereo 20-bit audio CODEC
- Philips SAA7121H digital video encoder
- Analog Devices ADV7123 140 MHz triple video DAC
- Interface for OmniVision™ OV6630AA CMOS color digital camera
- Interface for Fujitsu™ MB86S02A CMOS color digital camera
- AvBus expansion connector interface for Sharp™ LQ057Q3DC02 color TFT LCD module
- X/Y touchscreen controller
- PS2 keyboard and mouse interfaces

Communications/Memory Add-in Module

The Communications/Memory Module is an expansion daughtercard for use with Avnet Avenue Solutions offerings. The daughtercard interfaces through AvBus connectors and provides general-purpose resources to complement Avnet Avenue Solutions-based modules. The daughtercard provides all necessary resources for implementation of Xilinx MicroBlaze™ processor core designs.

Key elements of the module include:

- 64 MB SDRAM
- 16 MB Flash
- 1 MB SRAM
- IrDA
- 10/100/1,000 Ethernet PHY
- USB 2.0
- PC card interface

Conclusion

For a wide variety of DSP applications, it makes sense to start your design with a hardware-based development platform. You can pick and chose from three main platforms and customize by mixing and matching a variety of IP cores, daughtercards, firmware, and software. Visit *www.em.avnet.com/dspstartingline/* for current information on all Xilinx DSP-related tools from Avnet.

You can order any of the kits described in this article from your local Avnet sales office, or obtain additional information from the Avnet DSP Startingline website at *www.em.avnet.com/dspstartingline/*.

Digital Communications	• 16.QAM demodulator for software-defined radio • A QAM system with packet framing and FEC for telemetry channels • Concatenated FEC codec for DVB standard • Costas loop carrier recovery • Digital down converter for GSM applications
Signal Processing	• A/D and delta-sigma D/A conversion • FFT/IFFT in streaming mode • LMS-based adaptive equalization • Custom FIR filter reference library • Polyphase 1:8 MAC-based FIR using SRL16ES • IIR filtering: multi-channel, folded implementation • IIR filtering: 2nd-order Direct Form I implementation
Image Processing	• 2D DWT filter • 2D filtering using a 5 x 5 operator • Color space converter
Mathematical Operators	• CORDIC-based rectangular-to-polar coordinate converter • CORDIC-based divider circuit • CORDIC-based sine and cosine function
Control Logic	• Debugging a PicoBlaze™ microcontroller design

Table 1 – DSP demos in Xilinx System Generator

Using System Generator for DSP to Create the J.83 Cable Modulator

System Generator enables the rapid development of multi-channel cable head-end modulators to provide a true low-cost solution.

by Veena Kumar
Staff Design Engineer
Xilinx, Inc.
veena.kumar@xilinx.com

Hemang Parekh
Senior Design Engineer
Xilinx, Inc.
hemang.parekh@xilinx.com

The increased capability and capacity of video, audio, data, and interactive services through cable distribution has spurred much interest. Applications such as video-on-demand and cable telephony are natural extensions of these services.

The ITU-T (International Telecommunications Union – Telecommunication Standardization Sector) has established the J.83 specification to standardize the physical layer transmission of audio, video, and data services over cable networks. These cable transmission networks as they apply to Europe, North America, and Japan are detailed in Annex A, B, and C of this standard, respectively.

Xilinx addresses this interest with the J.83 Cable Modulator IP, a flexible, scalable, and cost-effective solution. In this article, we'll discuss the use of Xilinx J.83 cores in the downstream modulator at the head-end (Figure 1), while focusing on the physical layer implementation.

The Xilinx J.83 IP solution provides flexibility to parameterize the modulator; scalability to allow you to select any number of channels on a single FPGA; and ease of use in the System Generator for DSP visual programming environment as the design and delivery mechanism.

This programming interface allows you to work at a suitable level of abstraction from the target hardware platform and use the same model.

System Generator for DSP

The Xilinx System Generator tool suite was employed to implement a majority of the J.83 modulator design. System Generator is a visual dataflow design environment based on The MathWorks Simulink® visual modeling tool set. This programming interface allows you to work at a suitable level of abstraction from the target hardware platform and use the same model – not only for simulation and verification but also for FPGA implementation.

System Generator blocks are bit- and cycle-true behavioral models of FPGA intellectual property components, or library elements. A library-based approach results in design cycle compression in addition to generating area-efficient high-performance circuits. Together with model features such as data-type propagation and the extensive virtual instruments that are part of the Simulink libraries, the environment facilitates rapid design space exploration, together with powerful mechanisms for model debugging.

MATLAB® scripts from The MathWorks programmatically generate custom VHDL and project files based on user-defined parameters.

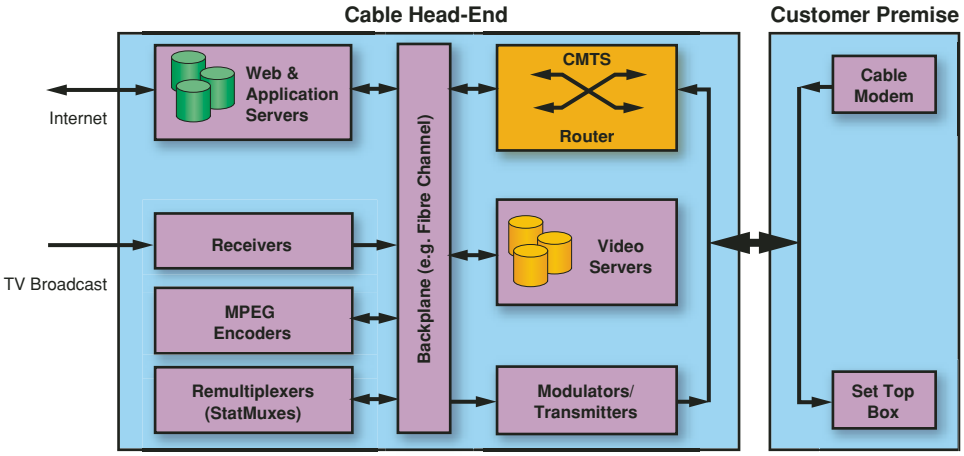


Figure 1 – Cable network (J.83 modulator fits in the cable head-end modulator/transmitter block)

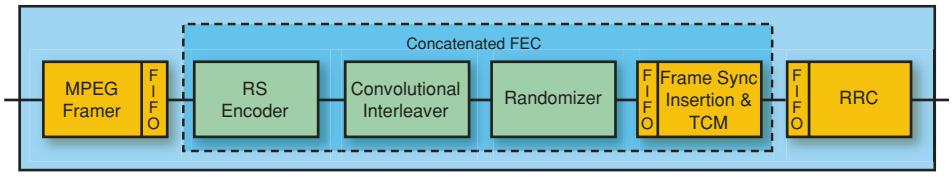


Figure 2 – J.83 Annex B functional block diagram

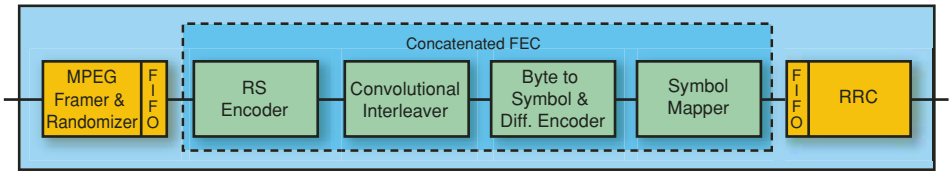


Figure 3 – J.83 Annex A/C functional block diagram

J.83 in System Generator for DSP

The J.83 specification defines the forward error correction (FEC) and baseband modulation with pulse-shaping characteristics. The J.83 Annex B FEC section (Figure 2) uses a concatenated coding technique with four processing layers, comprising an RS encoder, convolutional interleaver, randomizer followed by a frame sync insertion block, and trellis-coded modulation (TCM). The J.83 Annex A and Annex C (Figure 3) have identical FEC processing stages, comprising an RS encoder, convolutional interleaver, and a byte-to-symbol differential encoder, followed by a symbol mapper.

The System Generator Xilinx library, or block set, is abundantly populated with IP that enables rapid design and simulation of such a system. The tokens required to construct the J.83 FEC section – as well as the filter blocks required to construct pulse-shaping filters – are available within the library browser. The underlying circuit of each of these tokens is optimized in area and speed to suit the Xilinx family of devices.

Each of these elements is conveniently customizable to be compatible with the precise specification of the J.83 standard. It is then a simple matter of using these customized library elements to build out the circuit required.

For example, you can obtain the (204,188) RS encoder required for J.83 Annex A/C by using the Xilinx Reed Solomon encoder block, with the Code Specification parameter set to DVB. Similarly, the Xilinx interleaver deinterleaver block is directly used in the design, with the mode set to Interleaver and the Number of Branches and Length of Branches set to 12 and 17, respectively. This results in an exact match to the requirements of the interleaver in the J.83 A/C specification. Using the visual graphic means of design entry in System Generator, these blocks are easily connected to each other and to the control circuitry that is part of the design.

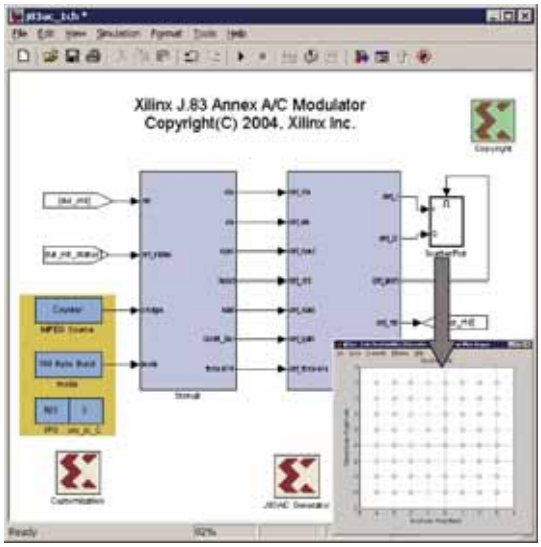


Figure 4 – J.83 Annex A/C modulator with scatter plot

IP Simulation in Simulink

It takes a lot of time to simulate and test the functionality of a complex system. You can use the same J.83 circuit built in System Generator for simulation and verification, as well as the FPGA implementation. Within the same environment, using Simulink for simulation, the design is stimulated with MPEG transport packets and the appropriate QAM, reset, synchronization, and other control inputs.

As shown in Figure 4, this stimulus is shown in the block labeled "Stimuli." The source, inter-packet gap, and burst nature of the MPEG transport packet may be chosen at random at the top level, allowing a test of the full suite of possibilities. Figure 4 also shows a discrete time scatter plot of the output of the baseband section of the modulator.

Simulation of this complex system in an HDL simulator for a meaningful number of clock cycles (such that several frames of data may be processed) imposes a huge penalty in the time taken to complete a simulation. This makes it an impractical choice, but sometimes it is the only option when the design source is in an HDL format.

This simulation time is drastically reduced when simulating the model in Simulink. What might take days to simulate in a gate-level simulator could be accomplished in a matter of hours. This savings in time is highly valuable – not only do you benefit from superior simulation speed in Simulink but you also reap the benefits of a shortened design cycle, allowing for overall rapid IP delivery.

Single and Multi-Channel Designs

The modulator is constructed out of two primary footprints or granularity: a single-channel implementation and a four-channel implementation. A block diagram of the four-channel granularity Annex B and A/C are shown in Figure 5 and Figure 6, respectively.

Each instance of the single-channel footprint provides for exactly one independent channel; the four-channel footprint, however, is optimized to efficiently support four channels at a time, using resource-sharing techniques. You select the granularity, and with that selection, make a trade-off between resource utilization and individual channel control.

The trade-off is essentially in the area (resource) utilization; the optimized four-channel group solution results in a very efficient and compact design requiring fewer FPGA resources. However, it imposes the restriction that the four channels must share the same controls. The single-channel solution imposes no such restriction; the trade-off here is the linearly increasing FPGA resources used, which is directly proportional to the number of channels required.

Multi-channel modulators are automatically constructed through the use of multiple copies (also referred to as groups) of the single- or four-channel implementation. For example, a four-channel modulator may be constructed with four copies of single-channel granularity or a single copy of the optimized 4-channel granularity design. Similarly, a 12-channel modulator may comprise 12 copies of the single-channel granularity design or 3 copies of the optimized 4-channel design.

The ease of use is evident in that the

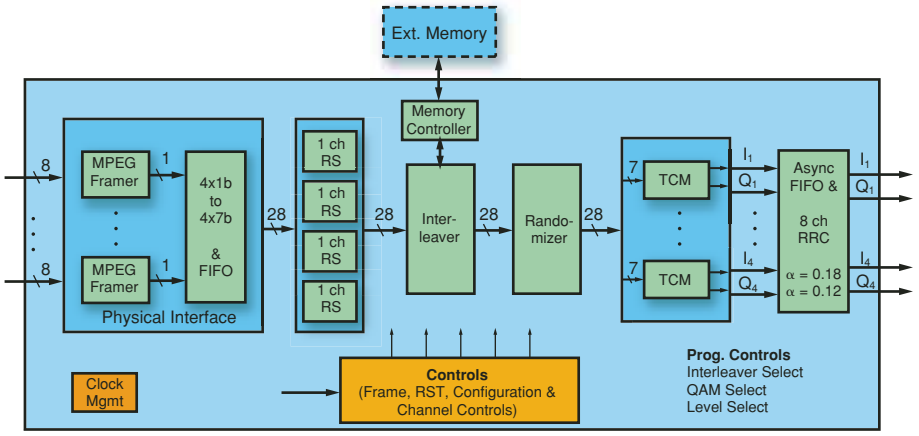


Figure 5 – J.83 Annex B four-channel granularity design

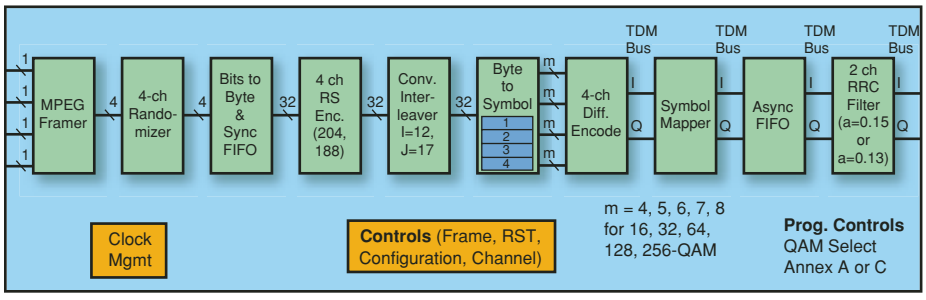


Figure 6 – J.83 Annex A/C four-channel granularity design

only requirement is for you to specify the parameters; the multiple instantiations of the basic footprints and the required connections between them are automatically generated, leaving you with a core design tailored to those exact specifications.

Usage

The Xilinx J.83 modulator implementation is available as a module that plugs into Xilinx System Generator for DSP, or as a netlist that may be directly referenced by another design. The design of the J.83 core in System Generator allows for generation with a simple push button solution.

Through a GUI constructed in the familiar Simulink environment, the core provides you with a convenient means of supplying design specifics such as the granularity desired, the number of channels required, and clock rates, as shown in Figure 7.

Figure 7 – J.83 Annex B generator GUI screenshot

During parameterization and generation, the core is automatically configured to the specifications and deposited into the target directory. Along with the netlist, the core also includes behavioral and timing simulation script files (.do) for Mentor Graphics® ModelSim™ and an ISE Project Navigator project file (.npl). From this point on, you can bring the core into the ISE Project Navigator environment for synthesis, place and route, and bitstream generation.

Resource Sharing

The Xilinx FPGA implementation of the J.83 modulator specification capitalizes on a particular architectural feature to construct efficient multi-channel implementations: the shift register logic 16 (SRL16) primitive, found in Virtex-II™, Virtex-II Pro™, and Spartan-3™ devices. You can think of SRL16 as a series concatenation of 16 flip-flops with a programmable tap point. This unique aspect of Xilinx FPGAs is extremely powerful for building very efficient time-division multiplexed (TDM) hardware that you can use, for example, to process multiple channels of data.

Because they run the design at a faster rate, TDM processing structures save resources. This has been notably exploited during the design of an optimized multi-channel group of modulators. For example, in the design of the optimized four-channel granularity of a group, all channels share a common control structure in the MPEG framer, RS encoder, interleaver, randomizer, and TCM. As the interleaver controls are shared, the data path into and out of the interleaver effectively becomes wider.

Resource Utilization

Using the resource sharing techniques we've described thus far, you can realize significant savings in the implementation of modulators constructed out of optimized four-channel granularity designs compared to the equivalent constructed out of single-channel granularity designs.

Table 1 and Table 2 show a comparison of the resources used in the design of various sizes of J.83 modulators using single- and four-channel granularity footprints. They also show the resources used to implement 4, 8, and 12 channels of J.83 Annex B and J.83 Annex A/C solutions on a Spartan-3 device.

Although Table 1 details the resources on an implementation that does not contain the optional root-raised cosine filter, the details in Table 2 are specific to an implementation that contains the option. Using the 12-channel case as an example, the scales are favorably tipped towards a four-channel granularity implementation of the J.83 Annex B and J.83 Annex A/C, as the savings achieved are significant.

Number of Channels	J.83 Annex B		J.83 Annex A/C	
	Slices/BRAM/External Memory		Slices/BRAM	
	One Channel Granularity	Four Channel Granularity	One Channel Granularity	Four Channel Granularity
4	3372/8/1	1866/2/1	1574/4	1049/3
8	6764/16/2	3644/4/1	3130/8	2088/6
12	10049/24/2	5405/6/1	4683/12	3304/9

Table 1 – Resource utilization comparison between one- and four-channel granularity J.83 Annex A/B/C designs without RRC (Spartan-3 FPGAs)

Number of Channels	J.83 Annex B		J.83 Annex A/C	
	Slices/BRAM/External Memory		Slices/BRAM	
	One Channel Granularity	Four Channel Granularity	One Channel Granularity	Four Channel Granularity
4	8014/20/1	3748/7/1	4829/8	2444/4
8	16024/40/2	7402/14/1	9661/16	4877/8
12	23924/60/2	11057/21/1	14449/24	7483/12

Table 2 – Resource utilization comparison between one- and four-channel granularity J.83 annex A/B/C designs with RRC (Spartan-3 FPGAs)

Design Example Usage

The J.83 modulator design provides control configuration that you can control using a PowerPC™ or the MicroBlaze™ processor in Virtex-II Pro FPGAs. The processor can not only control the (J.83 Annex B) configurations such as QAM, interleaver control word, and interleaver level, but also the reset sequence of the design. It may be shared to control other user logic such as the MAC layer implementation for cable communication at the head end and baseband-to-IF digital upconversion. The functional block diagram in Figure 8 depicts how you can leverage the capabilities of the Virtex-II Pro architecture for the J.83 design.

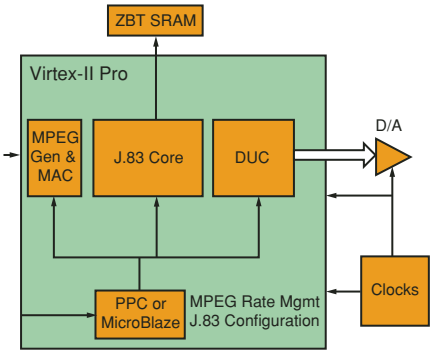


Figure 8 – J.83 single chip system design

Conclusion

Xilinx System Generator enables the rapid development and simulation of high-performance systems on Xilinx FPGAs. SRL16s allow you to design a 16-channel granularity modulator without using 16 times the resources of a 1-channel granularity modulator or 4 times the resources of a 4-channel granularity modulator.

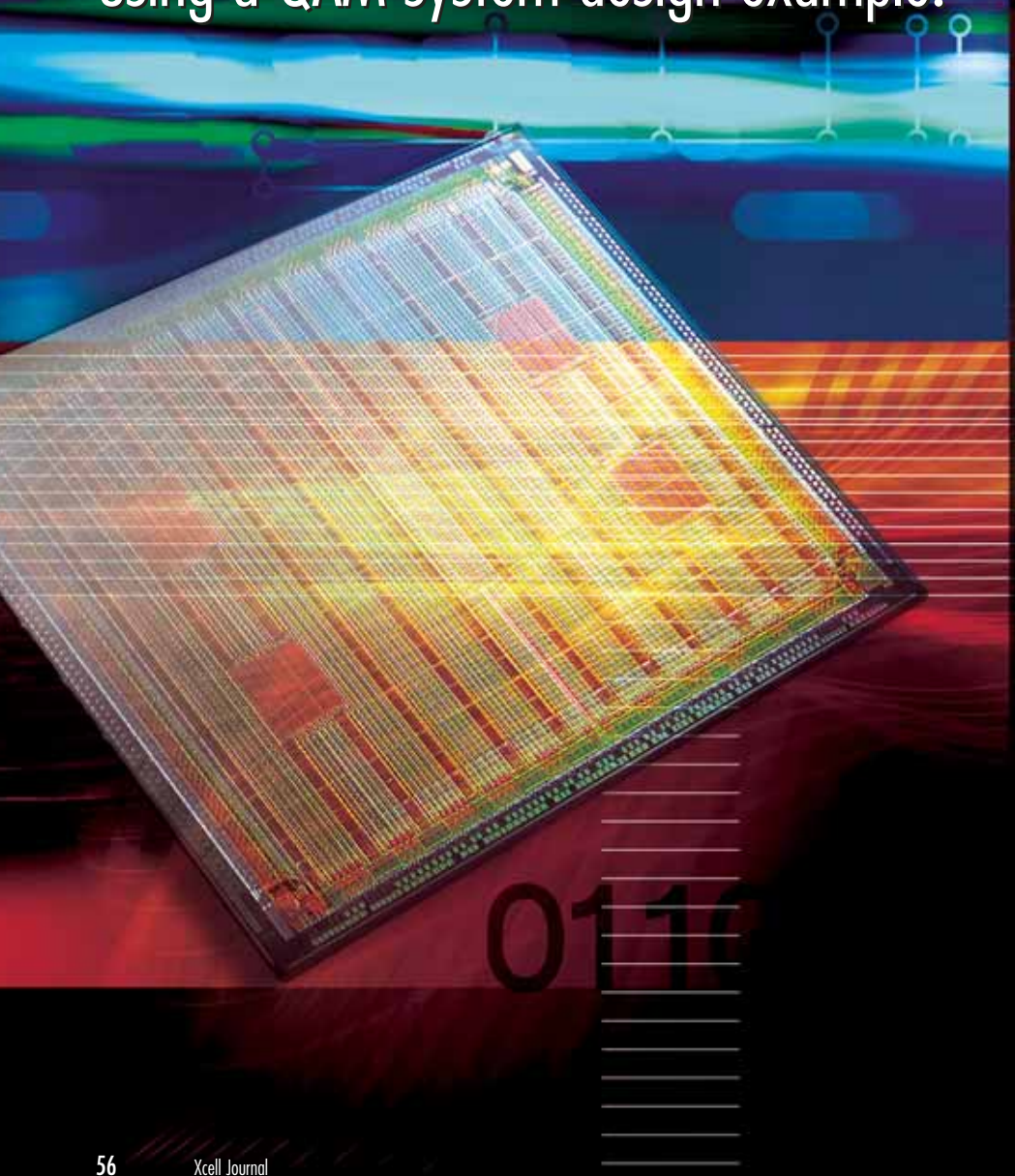
You can build various standard-compliant modulators for video broadcast for transmission over terrestrial links (DVB-T) via satellite (DVB-S2) or to handheld devices (DVB-H) quickly and efficiently using System Generator for DSP and various library blocks available from Xilinx. SRL16s in Xilinx FPGAs allow efficient time-multiplexed dataflow structures, offering significant resource savings.

For more information about the Xilinx J.83 Modulator IP, visit *www.xilinx.com/ipcenter/j83_mod/*.

XtremeDSP

Implementing DSP Algorithms in FPGAs

A detailed overview of Xilinx System Generator for DSP, using a QAM system design example.



by Sabine Lam
DSP Technical Marketing Engineer
Xilinx, Inc.
sabine.lam@xilinx.com

Today, the role of FPGAs in the DSP market is well understood – there simply is no better way to create ultra-fast DSP applications. Yet combining these two technologies can be a challenge – DSP designers primarily use The MathWorks MATLAB™ or C/C++ to specify systems, whereas FPGA designers use VHDL or Verilog™. The only common approach between these two is that they often start with a block diagram.

DSP architects and FPGA designers have two completely different backgrounds, yet they must work together to create an optimum product. Their focus and expertise do not overlap, and as a result, they often have difficulty communicating. The team must verify that the FPGA implementation does indeed match the original specification given by the DSP architect, and usually they must modify the DSP algorithm to obtain the best possible implementation in the FPGA. This requires a constant exchange of information about simulation results, design size, design performance, DSP algorithm changes, and implementation results throughout the design process.

The tool provides high-level abstractions that are automatically compiled into an FPGA at the push of a button.

Deciding on a single tool and a language that meets the requirements of the whole design team can be difficult, especially when budgets are low and turnaround times are short. What could be better than an environment understood by the entire team, using a single source code? without writing a single line of HDL code or even looking at the Xilinx ISE tools.

To highlight the System Generator design flow, we will use a Quadrature Amplitude Modulator (QAM) system design example (Figure 1), implemented according to the specifications provided by the Consultative Committee for Space Data Systems for telemetry channel coding specification (CCSDS 101.0 B-5).

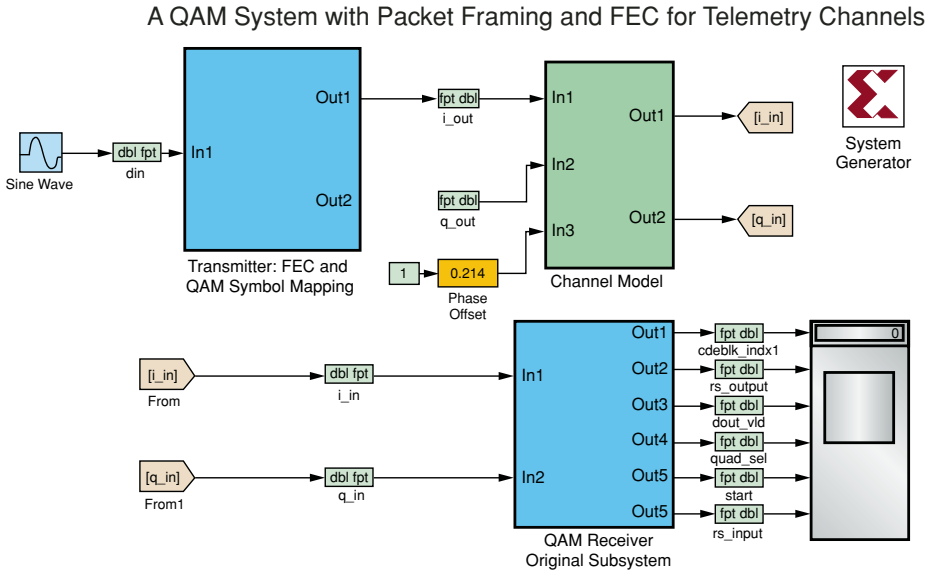


Figure 1 – QAM system (System Generator model)

System Generator

Xilinx® System Generator is a system-level modeling tool that facilitates FPGA hardware design. It extends The MathWorks Simulink™ in many ways to provide a modeling environment well suited to hardware design.

The tool provides high-level abstractions that are automatically compiled into an FPGA at the push of a button. The tool also provides access to underlying FPGA resources through lower-level abstractions, allowing the construction of highly efficient FPGA designs. It is delivered along with a predefined Xilinx blockset library, but also allows access to other languages with which most FPGA designers are familiar. Finally, it offers the ability to design at a system level, and allows simulation, implementation, and verification within the same environment, usually

Introduction to the QAM System Design Example

In our example, the overall QAM system starts with the transmitter. This subsystem accepts data from an input source, where forward error correction (FEC) is applied and an attached synchronization marker (ASM) is inserted into the data before modulation. The modulated data is then driven to the channel model, where inter-symbol interference, Doppler content, and additive white Gaussian noise are introduced into the signal. Finally, the receiver employs a 16-QAM demodulator that performs adaptive channel equalization and carrier recovery. The ASM is stripped from the demodulated data before applying error correction.

A PicoBlaze™ microcontroller controls the Reed-Solomon decoder, maintains frame alignment of the received packets, and performs periodic adjustments of the demapping QAM-16 quadrant reference. Both the transmitter and the receiver are targeted to the FPGA, whereas the channel is a Simulink model used for simulation and verification.

Although not all System Generator features are used in this design, it is a good example showing a combination of powerful features, using Xilinx blockset elements, legacy HDL code, a PicoBlaze processor, and hardware verification, resulting in a very elegant, efficient, and quick way to implement a complex design and qualify it in a single environment.

Design Implementation with System Generator

A System Generator design always starts and finishes with gateways to convert the Simulink double-precision data into a Xilinx fixed-point format. These gateways define the boundaries of your design; you can convert them into I/O ports for top-level designs or an I/O interface to import into a higher-level system. Between these gateways, you must use blocks from the Xilinx library blockset or import your own code through the black box interface.

The Xilinx blockset library comprises basic elements, math functions, DSP functions, communications blocks, control logic, and other useful elements. Each block is fully parameterizable, and a tight integration with the MATLAB workspace allows you to enter parameters based on complex equations or variables defined in the workspace.

Equations such as this one:

```
acc_nbits = ceil(log2(sum(abs(coef*2^coef_
width_bp)))) + data_width + 1
```

define the precision required for a filter as a function of the filter taps (coefficients), the number of taps, and the coefficient width. Because these are fixed at design time, it's possible to tailor the hardware resources to

the filter specification. As you change coefficients, no modification will be required to the design, because the output precision will automatically be recalculated.

This capability is only available with tools tightly integrated with MATLAB.

The Xilinx blockset library will not always meet your needs, and some functions are better suited for HDL implementation, such as complex state machines or complex legacy code. The System Generator black box allows you to bring VHDL, Verilog, and EDIF netlists into a design.

In the QAM design example, you can imagine replacing one of the FEC blocks (Reed-Solomon or Viterbi decoder) readily available from the Xilinx blockset library with your own HDL implementation through a black box. The black box behaves like other System Generator blocks – it is wired into designs, participates in simulations, and is compiled into hardware.

When System Generator compiles a black box, it automatically wires the imported module and associated files into the surrounding netlist. System Generator simulates black boxes by automatically launching an HDL simulator, generating additional HDL as needed (analogous to an HDL test bench), compiling HDL, scheduling simulation events, and handling the exchange of data between the Simulink and the HDL simulator. This is called "HDL co-simulation."

At this point, you can also envision bringing MATLAB or C-code previously compiled with the appropriate MATLAB-to-gate and C-to-gate tools into the Xilinx black box.

Simulation and Verification

You can couple Simulink blocks with System Generator models to produce robust, flexible test bench environments. The convenience of this capability is that implementation and verification design phases can take place in the same environment. Such is the case with the QAM system design, where a mixture of Simulink and System Generator blocks were used to implement the design test bench.

This allows you to verify the functionality of your design at any critical probing points in a format well understood by the DSP architect (such as sinewave or constellation). You can also automatically generate this exact same test bench and export it to an HDL simulation tool, in this case in a format well understood by the FPGA designer (binary waveform). Also automatically generated is "golden data" confirming similar functionality in the Simulink and HDL environments.

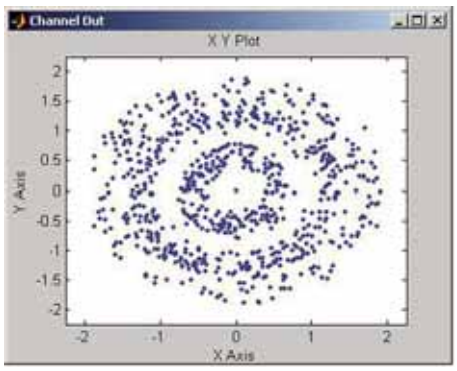


Figure 2 – Input (distorted channel) of the QAM demodulator

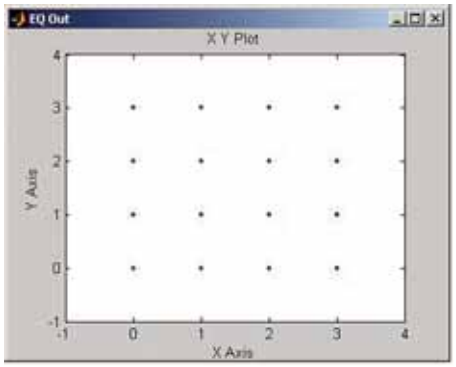


Figure 3 – Output (constellation) of the QAM demodulator

Figure 4 – System Generator GUI interface

Figure 5 – Phase offset slider bar

Note that the automatic generation of the test bench and golden data saves an enormous amount of time over other design flows.

Proceeding with the design implementation, you can now generate the netlist through the System Generator token (Figure 4).

At the push of a button, you can decide to generate a VHDL netlist, which will then have to be synthesized and place and routed, or you can decide to go straight to a bitstream or even target a hardware demonstration board.

Selecting hardware co-simulation makes it possible to incorporate a design running in an FPGA directly into a Simulink simulation. Hardware co-simulation compilation targets automatically create bitstreams and associate them with blocks. When the design is simulated in Simulink, results for the compiled portion are calculated in hardware. This allows the compiled portion to be tested in actual hardware, and can speed up simulation dramatically.

In addition to supporting specialized interfaces such as the XtremeDSP™ kit, System Generator provides a generic interface that uses JTAG and the Parallel Cable IV to communicate with FPGA hardware. This takes advantage of the ubiquity of JTAG to extend System Generator's hardware-in-the-simulation-loop capability to numerous other FPGA platforms.

System Generator also allows you to run hardware co-simulation on your own development board. You can make design decisions and changes earlier in the design

process and accelerate the design cycle directly from the Simulink environment.

Hardware Acceleration

The complexity and size of the QAM system resulted in lengthy simulation times. To accelerate the simulation and confirm the functionality of this system, the transmitter and the receiver were downloaded to two separate FPGA boards and brought back into Simulink using System Generator's hardware co-simulation feature. You are now simulating an entire system from Simulink, with 10 to 100 times faster simulation time, instantaneously validating the functionality in the FPGA.

Interactively Control Hardware

You can also go one step further by accessing the FPGA while running the simulation. In this QAM example, the amount of Doppler content introduced by the channel can be controlled interactively during simulation; a slider bar shifts the carrier phase of the modulated signal (Figure 5). A significant adjustment to the slider invariably causes the receiver to lose lock. Interactive control over Doppler provides a simple yet powerful way to test the functionality of the receiver's control systems.

Designers now have a simple way to simulate complex designs that require millions of samples that without hardware in the loop would take months to simulate. This, among other features, is something that only System Generator can offer. With other DSP design methodologies, you are required to verify designs in multiple design environments – a complicated process resulting in significantly slower simulation times.

Conclusion

System Generator is a mature tool that allows algorithm development, implementation, simulation, and verification in an environment understood by most designers. Although there are other design flows, no other tool offers features such as HDL co-simulation, hardware co-simulation, and integration with the ChipScope™ Pro tool and EDK, which are invaluable and only available in Xilinx System Generator for DSP.

XtremeDSP

New XtremeDSP Slices Deliver As Much As 10X More GMACs Per Dollar

Virtex-4 FPGAs offer breakthrough DSP performance at the lowest cost.

by Narinder Lall
Senior Manager – DSP Product and Solutions Marketing
Xilinx, Inc.
narinder.lall@xilinx.com

Xilinx® Virtex™ Series FPGAs have been the preferred choice for high-performance signal processing and DSP co-processing in many digital communications and video/imaging applications. With algorithmic complexity on the rise, the pervasive need for flexibility, and increasing downward pressures on price/power per channel, designers often face tough tradeoffs and difficult choices to employ either FPGA- or ASIC-centric systems for challenging signal processing applications.

New XtremeDSP™ slices on the Virtex-4™ device family extend FPGA signal processing capability beyond 256 GMACs, which represents DSP performance two times greater than previous generation Virtex-II Pro™ FPGAs.

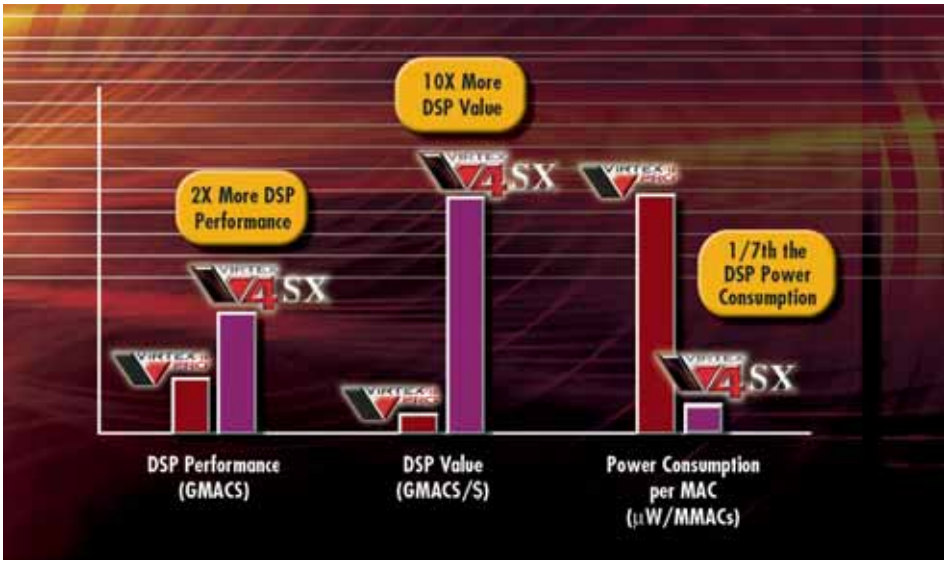


Figure 1 – The Virtex-4 FPGA offers breakthrough DSP performance at new low-cost points.

Even more revolutionary, Virtex-4 system designers need not employ the largest family member to achieve this performance, as has previously been the case. Virtex-4 FPGAs now deliver this signal processing capability in a medium-density device, providing you with as much as a staggering 10X increase in the available GMACs per dollar (see Figure 1). This dramatically extends production volumes where it makes economic sense to use an FPGA for performance-centric signal processing applications. The new DSP slices also dramatically reduce power consumption, allowing you to drive down both cost and power per channel.

DSP to Logic Resources

At the heart of the Virtex-4 FPGA's signal processing resources are new highly integrated XtremeDSP slices, sometimes referred to as DSP48s (Figure 2). Depending on the family member, you can utilize as many as 512 XtremeDSP slices, each capable of providing 500 MHz throughput.

Each slice contains a dedicated 2's complement signed, 18 x 18 bit multiplier, and a three-input adder/subtracter with feedback for accumulation modes. The addition of a seven-bit op mode multiplexer allows you to dynamically configure the XtremeDSP slice for one of more than 40 operating modes, such as addition, multiplication, accumulation, MACC functions, MACC cascading, wide (48-bit) addition, and wide multiplexing. Configuration wizards in Xilinx ISE or System Generator for DSP allow you to simply apply the desired function.

The addition of new XtremeDSP slices allows you to implement many such functions within the slice and without the need for external logic slices, which can thus be allocated to other tasks. XtremeDSP slices can also be cascaded directly without accessing logic fabric or any loss in speed.

Xilinx's new ASMBL architecture enables us to alter the mix of XtremeDSP slices and logic slices. The SX platform with in the Virtex-4 family offers the highest ratio of XtremeDSP slices to logic slices at one XtremeDSP slice for every 108 logic slices. The SX platform is ideally suited for multiplier or MAC-intensive tasks such as software radios. The LX platform offers the highest ratio of logic to other features, and is suited for many traditional FPFA applications that may also require some DSP capability.

Reduced Power Consumption

In today's infrastructure applications, driving down cost per channel is not the only goal diligently pursued. Wireless infrastructure manufacturers are under increasing pressure to stay within power limits imposed by governing telecom standards bodies. Power consumption is also becoming a key concern for some military applications, such as Joint Tactical Radio Systems radios.

The integrated XtremeDSP slices on Virtex-4 FPGAs eliminate the need to use logic slices for many signal processing and arithmetic tasks, reducing the need for power-consuming routing resources. Initial

The integrated XtremeDSP slices on Virtex-4 FPGAs minimize the need to use logic slices for many signal processing and arithmetic tasks…

- **High Performance**
  - 500 MHz, fully pipelined
- **High Integration**
  - 40+ DSP/arithmetic operation modes
  - Directly cascadeable
- **Easy to implement**
  - Software configuration wizards

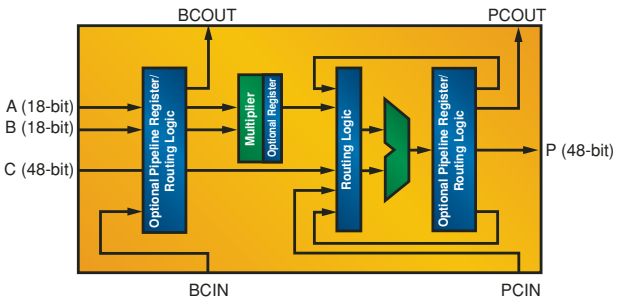


Figure 2 – New XtremeDSP slices feature 18 x 18 bit multiply, 48-bit accumulator

Virtex-4 FPGAs bring a new revolutionary era in the XtremeDSP initiative that provides you with economic incentives to use FPGAs and get your design to market faster than ever before.

power estimates show that XtremeDSP slices consume only 57 µW/MMAC, representing one-seventh the power for an equivalent function implemented using Virtex-II Pro FPGAs. Although not the ultimate goal, this reduction in power goes some way in addressing the power concerns of infrastructure equipment providers.

Another way to reduce system power consumption in such applications is to use the embedded processor capabilities available on the FX platform. You have the option to trade gates for processor cycles for sequential control tasks using FX platform devices. Examples of such implementations include software communication architectures or real-time operating systems.

High Compute Density Using SRL16s

Shift Register Logic (SRL16) is a unique feature in Xilinx FPGAs. A popular feature for increasing compute density in multi-channel implementations, SRL16s are included in all Virtex-4 platforms.

To demonstrate SRL16 usage, let's take a look at a simple Reed-Solomon encoder example. Implementing a single-channel Reed-Solomon encoder in a Virtex-4 device consumes 56 logic slices. For a 16-channel implementation, one approach would be to replicate this 16 times, resulting in a consumption of 16 x 56 slices.

Figure 3 shows another implementation of the 16-channel solution using SRL16s. This consumes only 86 logic slices, representing only 10% of the 16X replicated version. SRL16s can substantially pack more signal processing into a smaller area, allowing you to potentially target a much smaller device than is possible with other FPGA architectures.

Serial/Parallel Connectivity

In addition to embedded processors, the FX platform also includes 3.125 Gbps multi-gigabit transceivers that are particularly suited for interfacing to other DSP processors. One such example is high-speed serial connectivity using the serial RapidIO™ interface, which is gaining momentum with DSP vendors. With 1 Mbps LVDS interfaces for interfacing to high-speed A/D converters and a host of DRAM and SRAM memory interfaces for hooking up to frame buffers, the Virtex-4 family is an ideal platform for interfacing to other DSP devices that will form part of the system data flow.

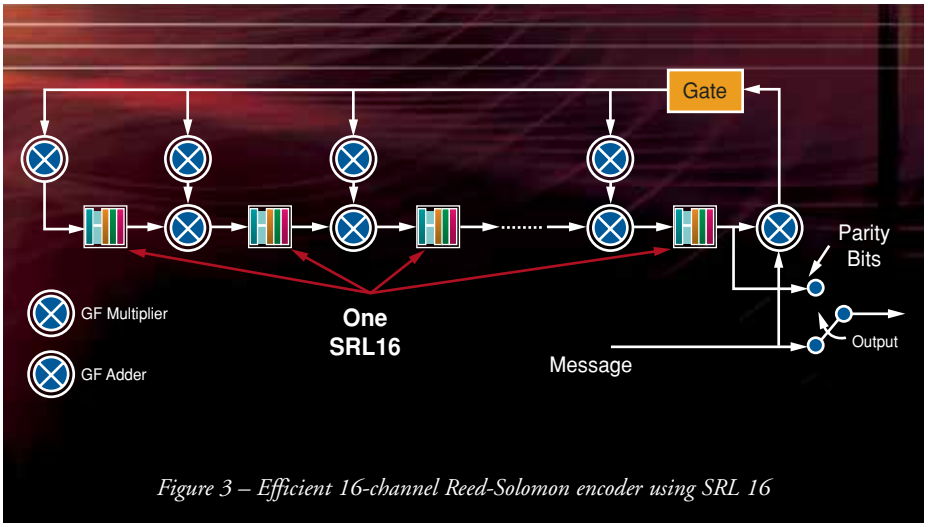


Figure 3 – Efficient 16-channel Reed-Solomon encoder using SRL 16

Virtex-4 DSP Design Solutions

The Virtex-4 family includes a beefed-up set of DSP design resources.

- System Generator for DSP allows you to model your design in The MathWorks Simulink® and, through powerful capabilities like hardware-in-the-loop, verify and debug that design from the same environment. System Generator also includes a new block that allows you to instantiate an XtremeDSP slice and configure it for one of its many operating modes.
- Hardware-in-the-loop is supported for any Virtex-4 development environment with a JTAG header. Other new capabilities introduced in System Generator 6.3 include the ability to generate VHDL or Verilog™ netlists.
- The Xilinx DSP library now supports Virtex-4 FPGAs, allowing you to develop designs faster.
- A range of services are now available as you implement your DSP design onto Virtex-4 FPGAs. These include DSP design services, education classes, and platinum/technical support.

Conclusion

FPGA-based DSP has always been associated with high performance when hundreds of GMACs/s rates are needed. Virtex-4 FPGAs bring a new revolutionary era in the XtremeDSP initiative that provides you with economic incentives to use FPGAs and get your design to market faster than ever before.

To understand how to use the new XtremeDSP slices in your next design, attend the Virtex-4 session in the DSP track at Programmable World 2004, or watch the demo-on-demand that will follow the event at *www.xilinx.com/dsp/*.

Developing Image Processing Algorithms with System Generator

System Generator allows you to easily implement and analyze any image processing algorithm.

by Daniel E. Michek
Staff Field Applications Engineer
Xilinx, Inc.
daniel.michek@xilinx.com

Converting image processing algorithms to FPGA implementations can be tedious. The algorithm may be proven in software but with no direct link to actual implementation. Additionally, it can be difficult to subjectively verify the implementation.

Using a mathematical simulator to verify and create HDL implementation files bridges the gap from the algorithm architect to the FPGA engineer. Xilinx® System Generator for DSP allows for high-level mathematical verification and converts the heart of the algorithm into ready-to-use HDL.

Simulation inside of The MathWorks Simulink® tool enables you to easily verify the image algorithm qualitatively and subjectively when used with the Image Processing Toolbox, also from The MathWorks. Using System Generator to develop and implement image processing algorithms allows for a thoroughly verified and easily executed design; plus, you save time on subjective analysis of the HDL. The high-level block diagram allows for easy communication between team members, resulting in less time spent crossing skill boundaries when determining implementation trade-offs.

The Basics

The Image Processing Toolbox is an excellent starting point with which to develop image processing algorithms for FPGAs. It allows you to easily load and view many image types. Although not directly usable within System Generator, it can also rotate, resize, filter, convert to and from the frequency domain, and operate on the image mathematically and morphologically. You can use these latter functions as a qualitative measure against the actual System Generator implementation.

Exchanging Data

Images are often stored and manipulated as two-dimensional arrays that can be quite large. For example, a 1,024 x 1,024 x 8-bits-per-pixel image is 1 MB. Typically, a portion of the image is moved into the FPGA one pixel at a time and aggregated into a larger unit (often a line), with manipulations performed on these points.

An example pre-processing MATLAB® m-file script might contain:

- Reading in a source test image using the IMREAD function in the Image Processing Toolbox.
- Analyzing variables such as width, height, and color depth of the image to pass as arguments to Xilinx block set tokens. This enables easy parameterization and scaling of the application.
- Storage and creation of other variables necessary to the application. Examples include rotation angle, resizing percentages, and bit precision within the algorithm.
- Converting the matrix data from an m x n array to a 1 x (m*n) with "for loops" and concatenation. This allows The MathWorks DSP block set "Signal From Workspace" token to pass elements as samples to the Xilinx block set "Gateway In" token.
- Viewing the source test image for later subjective analysis using the IMSHOW function in the Image Processing Toolbox.

An example post-processing m-file might contain:

- Conversion of "ToWorkspace" variables from a 1 x (p*q) array to a p x q array for easy manipulation by using "for loops."
- Displaying the resulting matrices using IMSHOW for subjective analysis.
- Computing qualitative analysis of the results versus the original image or an algorithm developed with the Image Processing Toolbox.

Using M-Files

You can use m-files to model the algorithm before implementation with System Generator. Although this step is optional, it can be a highly effective aid in the slight paradigm shift from matrix operations in software to raster scan operations in a highly parallel architecture.

Some of the key benefits include:

- Deconstruction proofs of Image Processing Toolbox functions as an aid to understand the algorithm.
- Creation of intermediate variables to assist in debugging the System Generator version of the design. Try to set variables similar to how they will appear in the design. Two-dimensional matrices should be used as memory, while raster order works better between blocks.
- Making algorithm trade-offs. Ask yourself if you should place the bi-linear interpolation stage before or after the sending of data to memory.
- A fast proof that the algorithm is on the right track.
- Qualitative analysis from the deconstruction m-file to the corresponding Image Processing Toolbox function.

Design Considerations – An Example

Real-time image rotation has many challenges when going from the algorithm architect at a high level to the FPGA engineer at the HDL and board level. The algorithm level choices that you will make about how to implement the design in the FPGA will affect device utilization. Conversely, board and system requirements may place limitations on your design, such as number, width, and type of frame buffer memories.

Design Choices

A key decision about the architectural style of the rotation algorithm is quality. For example, when moving the pixels of the original image grid to fractional locations on the rotated image grid, is a nearest neighbor selection adequate? Should you perform bi-linear interpolation on the image to reduce shear (the tendency to see discontinuities along object edges)? Is bicubic interpolation (determining the new pixel value based on weighting and summing its closest 16 neighbors) the preferred method? Your choice of quality will impact resources in both the FPGA and the frame buffer.

Our example is based upon the Hotelling transform to determine the original image versus the rotated image pixel addresses. You have two choices as to where the rotation engine occurs with respect to frame buffering:

1. Place the rotation engine after the frame buffer. This has the effect of sequential raster scan address writes into the frame buffer and allows for raster scan format of the output data by reading from the frame buffer in non-sequential address form. The Hotelling transform with basis vectors cos(t) and sin(t) for rotation angle (t), {Sx,Sy} representing source x and y coordinates, and {Dx,Dy} representing destination x and y coordinates in the 2-D image are:

$$Sx = Dx * \cos(t) + Dy * \sin(t)$$

$$Sy = Dy * \cos(t) - Dx * \sin(t)$$

2. Place the rotation engine before the frame buffer. In effect, this method predetermines and weights the values stored through non-sequential addressing into the frame buffer. The output data is read in raster scan format from the frame buffer. The Hotelling transforms with similar representation as above are:

$$Dx = Sx * \cos(t) - Sy * \sin(t)$$

$$Dy = Sy * \cos(t) + Sx * \sin(t)$$

In their paper, "Real Time Image Rotation and Resizing, Algorithms and Implementations" (*www.xilinx.com/products/logicore/dsp/rotation_resize.pdf*), my Xilinx colleagues Robert Turney and Chris Dick showed that this approach is more economical in terms of memory bandwidth. Because the data is stored to the frame buffer in non-sequential order, you must take care to ensure that all valid pixel locations are written to.

Also, if the rotation angle is changed, you must clear artifacts in the corners from the previous frame from the frame buffer. A nearest neighbor rotation of 45 degrees clockwise demonstrates the void artifacts, as shown in Figure 1, that occur due to multiple input pixels being written to the same frame buffer address.

Figure 1 – Void artifacts when rotation engine occurs before the frame buffer

To avoid a complex control path and concentrate mostly on the data path, our example goes with the first choice previously described (place the rotation engine after the frame buffer). The net effect will be an increase in memory bandwidth of 4 or 16 times for bi-linear and bi-cubic interpolation, respectively. The choice of how to deal with the increase in bandwidth requirements is design- and memory-dependent.

You can increase the number of read accesses from the memory by a factor of four to minimize the total amount of memory for bi-linear interpolation. This will decrease the overall incoming pixel rate by a factor of four unless the memory access speed is sufficient to handle the offset. Alternatively, you can store the four pixels necessary for bi-linear interpolation at the

same address location, thus increasing the total amount of memory by a factor of four. This methodology can be mitigated based on pixel bit width – that is, 8-bit-wide pixels can use this approach to fit into a 32-bit-wide memory footprint.

Design Implementation with System Generator

Our example will implement image rotation with the following assumptions:

- Input image size and pixel widths are known at the time of implementation
- Use the Hotelling transform of the form:

  $$Sx = Dx * \cos(t) + Dy * \sin(t)$$

  $$Sy = Dy * \cos(t) - Dx * \sin(t)$$

- Bi-linear interpolation
- Increased memory bandwidth is accounted for by reading the data out of the frame buffer four times faster than the incoming pixel rate

Frame Buffer Read Address Generation

Because the data will be output in raster scan format, you can create the destination address with two counters representing the two dimensions. These values must be center-adjusted around zero before being transformed. At this point, any adjustment changes the center of rotation for the output image.

The following two examples demonstrate manipulating the output image by using minimal extra logic:

- Two adders are all that are required to add pan control for the output image.
- Using two multipliers, you can scale the destination addresses to zoom in or out from the center point.

As interpolation occurs at a later stage, zooming and panning at this point will result in an output image with each pixel uniquely interpolated, versus a simple copy or additional interpolation procedure later. Figure 2 demonstrates a 45-degree counter-clockwise rotation with 8x zoom.

Once the destination address is correct for center, pan, and zoom, the transform is applied. The resulting addresses are reverse-corrected for the centering function. The numbers are of the form integer with a decimal portion. The integer portions are the required source address locations to read from the frame buffer. The decimal portions are used to create weighting factors used for the bi-linear interpolation.

Figure 2 – Zoom and pan incorporated into the algorithm

Figure 3 – Original picture for subjective comparison

Figure 4 – Image rotated counter-clockwise 7.5 degrees, followed by rotation clockwise 7.5 degrees.

Weighting the Data

Four pixels are fetched from the frame buffer to create a 2 x 2 matrix containing the source address pixel and its original neighbors to the right, below, and below-right, which we will identify as XY, XpY, XYp, and XpYp, respectively. The source address transform uses the decimal portions of {Sx,Sy} represented as {Rx,Ry} to create the following weighting equations:

- Weighting(XY) = Wxy = (1 – Rx) (1 – Ry)
- Weighting(XpY) = Wxpy = (Rx) (1 – Ry)
- Weighting(XYp) = Wxyp = (1 – Rx) (Ry)
- Weighting(XpYp) = Wxpyp = (Rx) (Ry)

These equations can be shown to be equal to the following equations to reduce the number of multipliers necessary from four to one:

- Wxy = (1 – Rx) – Wxyp
- Wxpy = (Rx) – Wxpyp
- Wxyp = (Ry) – Wxpyp
- Wxpyp = (Rx) (Ry)

The interpolated pixel value is the summation of the 2 x 2 matrix pixels multiplied by their respective weighting functions.

Compare the original image in Figure 3 with an image that was rotated through 7.5 degrees, followed by a -7.5 degree rotation in Figure 4.

Conclusion

In this article, I've shown how to use System Generator to explore and implement image processing algorithms.

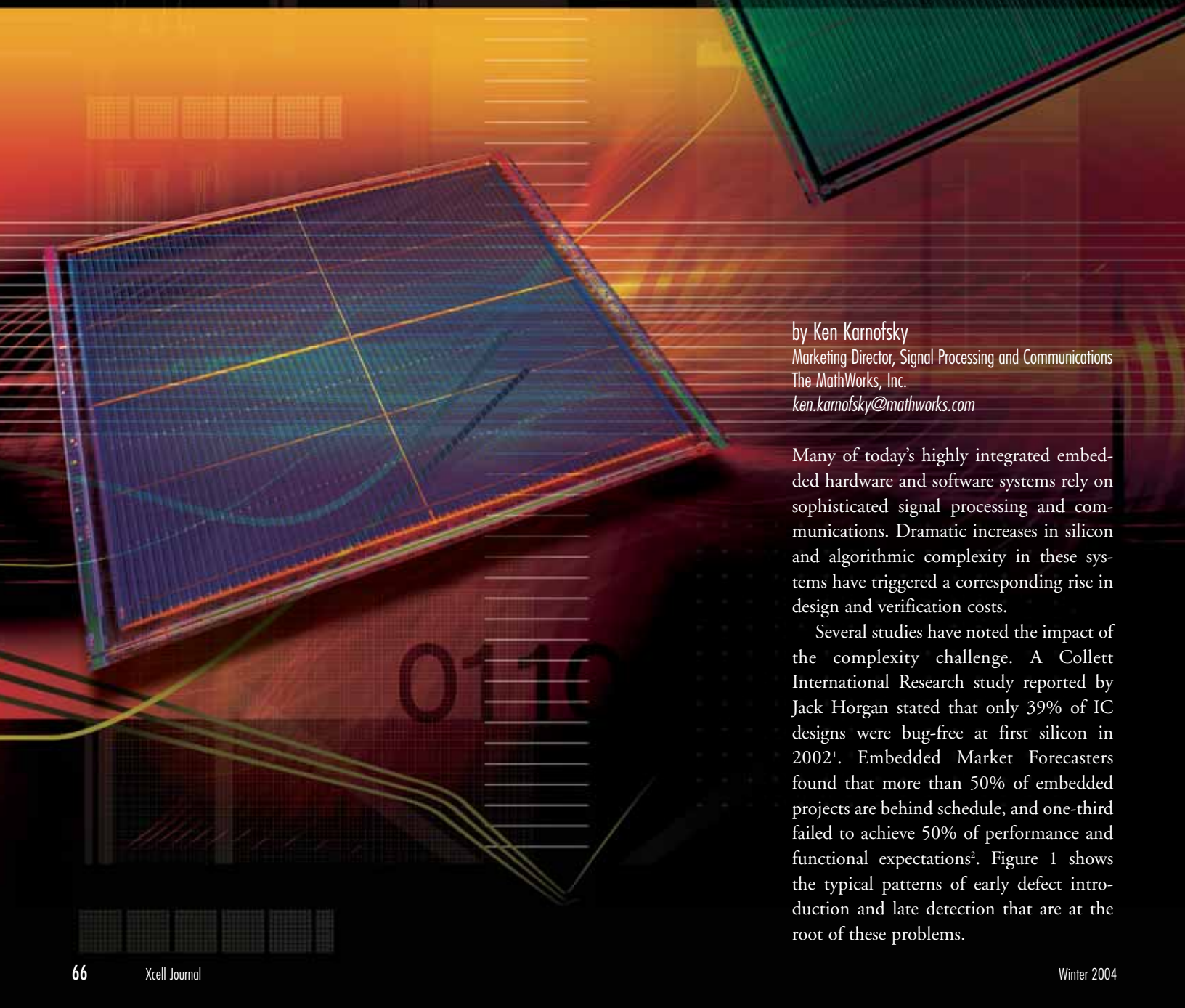
The use of a mathematical simulation tool with image handling capabilities allows you to easily investigate various options in an intuitive capacity. Although this example shows an image rotation implementation, these same methodologies can help you develop any image processing algorithm.

If you have any questions or suggestions, call me, Daniel Michek, at (858) 431-5901, or send me an e-mail at *daniel.michek@xilinx.com.*

XtremeDSP

Simulink Brings Model-Based Design to Embedded Signal Processing

The complexity of FPGA-based signal processing systems drives the need for new development approaches.



by Ken Karnofsky
Marketing Director, Signal Processing and Communications
The MathWorks, Inc.
ken.karnofsky@mathworks.com

Many of today's highly integrated embedded hardware and software systems rely on sophisticated signal processing and communications. Dramatic increases in silicon and algorithmic complexity in these systems have triggered a corresponding rise in design and verification costs.

Several studies have noted the impact of the complexity challenge. A Collett International Research study reported by Jack Horgan stated that only 39% of IC designs were bug-free at first silicon in 2002[1]. Embedded Market Forecasters found that more than 50% of embedded projects are behind schedule, and one-third failed to achieve 50% of performance and functional expectations[2]. Figure 1 shows the typical patterns of early defect introduction and late detection that are at the root of these problems.

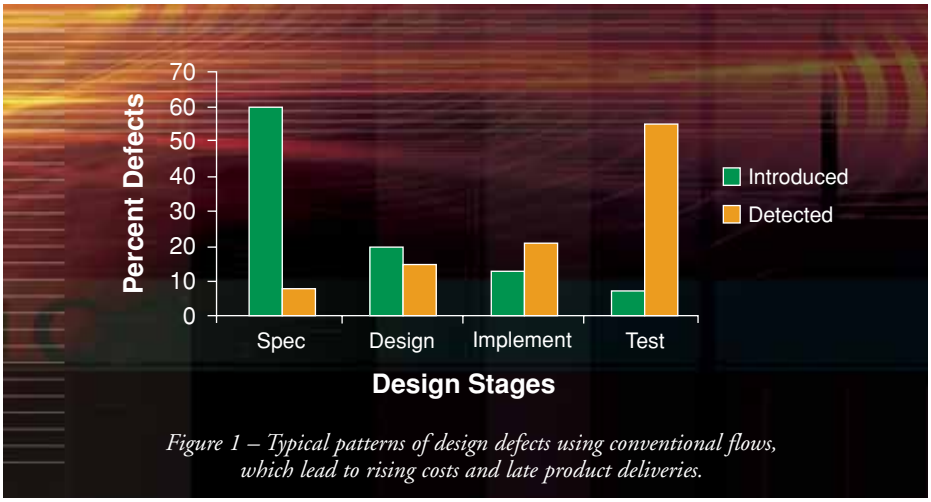


Figure 1 – Typical patterns of design defects using conventional flows, which lead to rising costs and late product deliveries.

Tacking new verification techniques or language extensions onto traditional design tools and flows is not enough to effectively improve the development process. These incremental improvements do not eliminate the aspects of traditional flows – ambiguous text-based specifications, manual implementation, and after-the-fact testing – that produce expensive errors and jeopardize delivery timelines.

In contrast, The MathWorks has demonstrated that Model-Based Design with Simulink® produces dramatic reductions in development time, cost, and risk. These benefits have been documented in the aerospace, automotive, communications, and semiconductor industries – wherever the application requires real-time signal processing, communications, and control logic.

Reinventing Development

The use of FPGAs for high-performance DSP is a natural application for Model-Based Design. Getting the most out of FPGA hardware requires insight into algorithmic and architectural complexities at the same time. To do this, architects need tools that offer direct access to hardware – tools that hardware and software engineers can also use to refine and implement the designs. A design environment like Simulink makes it possible to quickly and accurately simulate system behavior, and provides a direct path to implementation using automatic code generation.

Xilinx® recognized early on that Simulink was a platform that could make FPGA-based DSP design practical. Today, a complete design flow is available from The MathWorks and Xilinx that includes third-party development hardware for real-time prototyping and deployment. Many organizations now enjoy order-of-magnitude returns on their investment in Model-Based Design for Xilinx FPGAs.

Comprehensive, system-level mathematical models form the basis of Model-Based Design. Such modeling was once only in the realm of technology researchers, not mainstream product developers. But facing the limitations of traditional software and hardware description languages for large-scale projects, many design leaders recognize that modeling and simulation is necessary to handle the complexity of today's systems, not only for system design but for hardware and software development as well.

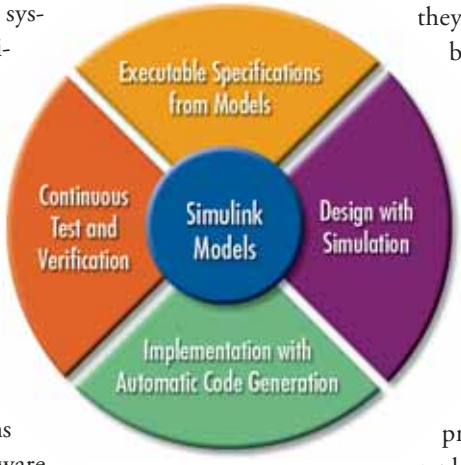


Figure 2 – The elements of Model-Based Design

Hardware description languages, even with "system-level" extensions, do not support the rapid modeling and design iteration needed for algorithmically intensive, large-scale embedded hardware systems – that is, virtually all of today's communications and multimedia systems.

Similarly, C-based tools and design flows will not address the software explosion in these systems. In fact, automotive companies and others facing the rapid growth of software-intensive embedded systems have turned to Model-Based Design. Manually developing code in C is no longer an option, because companies cannot hire enough programmers or test engineers to develop and verify the code.

The Elements of Model-Based Design

With Model-Based Design, specification, design, implementation, and verification can be accomplished – and accelerated – using a single Simulink model. Figure 2 depicts these elements, which are described below.

Executable Specifications from Models

Simulink models serve as executable specifications for system and component behavior, replacing ambiguous text documents. These models can span digital and analog hardware as well as software, and they facilitate clear, unambiguous communication between engineering teams.

Design with Simulation

Simulink is a platform for multi-domain simulation of dynamic systems. The Simulink product family provides an interactive, graphical block diagram environment with a customizable set of block libraries for signal processing, communications, and control. You can create comprehensive system specifications, model channels, and other environmental effects.

These tools also simplify system analysis using quantitative measures such as signal-to-noise ratio and bit-error rate. Simulink is integrated with MATLAB®, providing access to an extensive range of tools for algorithm development and data analysis.

Simulink models are hierarchical; you can partition them easily into subsystems or components. This simplifies comprehension of the design and interaction of

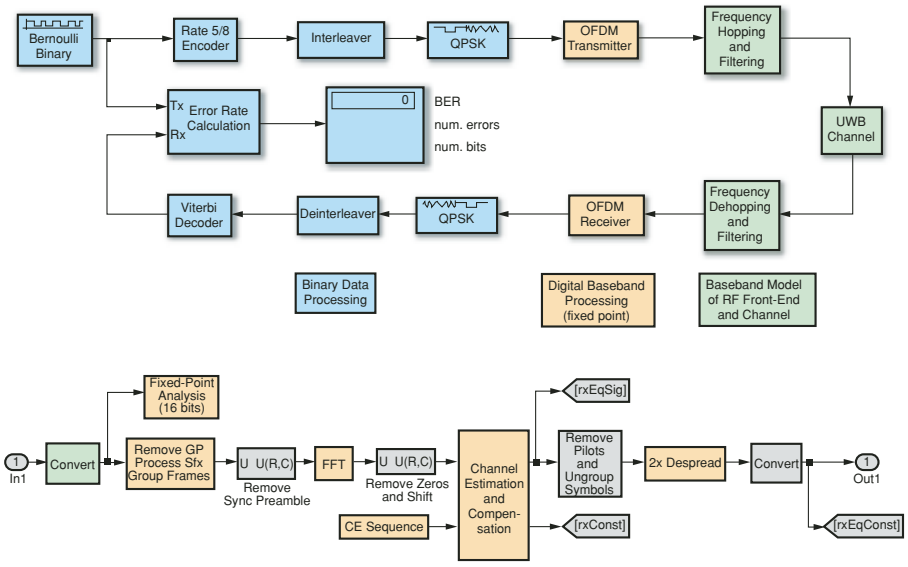


Figure 3 – Simulink model of UWB system (top) with fixed-point OFDM receiver (bottom). Fixed-point computations are shown in yellow.

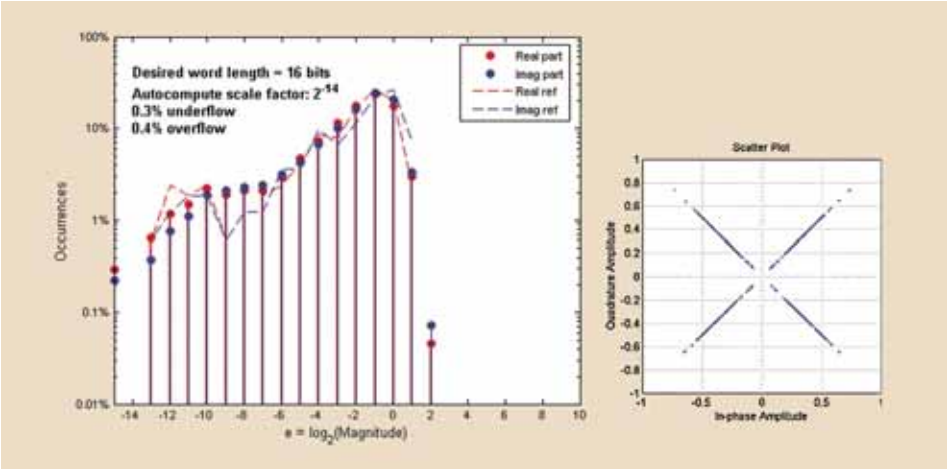


Figure 4 – Optimal fixed-point scaling of OFDM transceiver obtained through iterative simulation and analysis.

subsystems, including software, digital hardware, and RF/analog hardware. You can rapidly simulate and iterate to identify flaws, and refine the model to validate behavior against the requirements.

The Simulink model defines all the information needed to implement the software or hardware, including bit-true fixed-point processing and cycle-accurate timing and synchronization of multi-rate systems. Simulation is used to show that the executable specification defined by the model is complete and works correctly.

You can use model components to create well-defined subsystem interfaces, which simplify reuse in subsequent design efforts, even when those projects employ different target hardware or hardware/software partitioning.

Implementation with Automatic Code Generation

Once you have refined and validated the design, you can automatically generate code from the model, eliminating the need for hand-coding and the errors that manual coding can introduce. You can then use the code for real-time prototyping and deployment in the target system. The strategic partnership between Xilinx and The MathWorks has brought automatic hardware code generation capabilities to Xilinx FPGAs.

Continuous Test and Verification

You can ensure quality throughout the development process by integrating tests into the models at any stage and quantifying test coverage of the model. This continuous verification and simulation helps identify errors early, when they are easier and less expensive to fix, and streamlines the final verification stage.

The system model, or "golden reference," can serve as the test bench for the hardware or software implementation, which you can verify through software or hardware-in-the-loop co-simulation.

Applications of Model-Based Design

Model-based design can accelerate and simplify the development of many technologies. These examples are a small subset of the many applications available on The MathWorks website.

UWB Wireless

The range of ultra wideband (UWB) links is limited by the requirements for low-power, high-speed, and low-cost implementation. Fixed-point word length and scaling have a direct impact on hardware size, cost, and signal-to-noise ratio (SNR) degradation.

Using Simulink, the 10-bit orthogonal frequency division multiplexing (OFDM) transceiver for UWB shown in Figure 3 was designed in a few days. The receiver operates with a 0.5 dB degradation in signal-to-noise ratio, relative to a floating-point reference model.

The optimal word length was determined through simulation over a range of word lengths and channel conditions to evaluate trade-offs between chip size and wireless range. The results are shown in Figure 4. The transceiver operates within a complete end-to-end system model that serves as both an executable specification and a test harness for verifying downstream implementation.

Digital Down Converter for Software-Defined Radio

FPGAs are being used to perform high-data-rate signal processing in many emerging software-defined radio applications. A

typical application is the digital down converter (DDC), which is a sequence of multi-rate filters that decimate the RF signal down to the baseband rate. The design challenge is to design an architecture for each filter stage that optimizes the trade-offs among word length, computational delays, and accuracy of the overall filter response to avoid aliasing and other unwanted numeric effects.

The Simulink model of the cascaded integrator comb (CIC) filter used in the DDC, shown in Figure 5, was automatically generated from the MATLAB filter specification. Models such as this can provide a reference design for developing optimized Xilinx FPGA implementations with Xilinx System Generator.

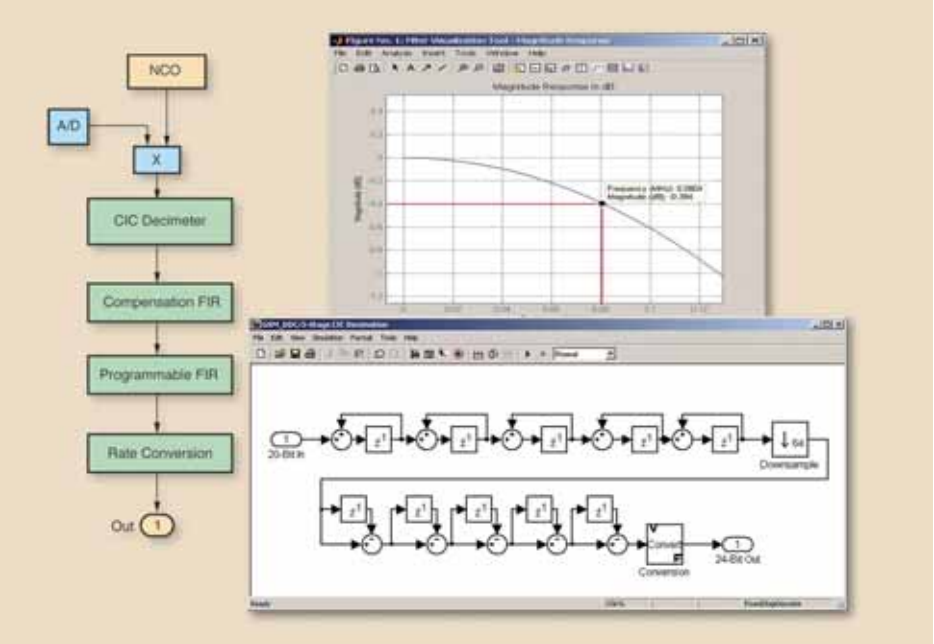


Figure 5 – Frequency specification (top) and Simulink model of a fixed-point CIC filter (bottom) for a digital down converter (left) in a software-defined radio receiver front end.

Reconfigurable Encryption System

Nallatech, a provider of high-performance FPGA systems, used Simulink and Xilinx System Generator for DSP to design a reconfigurable video encryption system in less than two weeks. The system enables their customers to re-verify an entire system when changing components and interfaces – without any knowledge of VHDL. "With this design flow, we efficiently implemented our system and algorithms with a significant improvement on traditional design times, without sacrificing performance," says Daniel Denning, a research engineer with Nallatech. "Coding in VHDL would have taken us three times as long."

Simulink 6

In June 2004, The MathWorks introduced Simulink 6, which increases performance, responsiveness, modeling fidelity, and workflow efficiency when modeling large systems. Simulink 6 also extends the scope of Model-Based Design to new domains and applications. These enhancements include:

- Component-based modeling for large-scale systems, including the ability to simulate, test, and implement each design component independently
- Unified data management for model and signal parameters across component models, including a graphical model explorer tool
- Simulink Verification and Validation, which links models to requirements and test cases, and identifies untested portions of models
- Ability to include a subset of the MATLAB language in Simulink models, and automatically generate embeddable C code
- New products for Model-Based Design of signal processing and communications systems, including:
  - Filter Design HDL Coder, for generation of VHDL and Verilog™ code for fixed-point filters
  - Fixed-Point Toolbox, for design and verification of fixed-point algorithms and analysis of fixed-point data in MATLAB
  - RF Toolbox, for design and analysis of networks of RF components
  - RF Blockset, for design and simulation of RF system and component behavior in end-to-end Simulink wireless system models
  - Video and Image Processing Blockset, for design and simulation of embedded video and image processing systems
  - Link for ModelSim™, for co-simulation and verification of VHDL and Verilog using Mentor Graphics® ModelSim

Conclusion

There is growing acceptance of Model-Based Design as the way to handle complexity in embedded hardware and software systems. The MathWorks/Xilinx alliance has enabled the design and implementation of high-performance DSP systems within the Simulink environment, reducing design and schedule risk while capitalizing on the potential of FPGAs for advanced signal processing applications.

For more information, visit *www.mathworks.com/products/dsp_comm/* for technical literature, webinars, and demonstrations.

References

1. Horgan, J. March 29, 2004. "Hardware/Software Co-verification," EDA Café Weekly.
2. Krasner, J. January 2004. "Model-Based Design and Beyond: Solutions for Today's Embedded Systems Requirements," Embedded Market Forecasters, American Technology International, Framingham, Mass.

XtremeDSP

Interfacing Simulink to the Analog World

The P160 Analog Module adds analog I/O to Memec/Xilinx development boards.

by Luc Langlois
DSP Specialist
Memec Insight
luc_langlois@ins.memec.com

If your product performs digital signal processing, it probably must interface to real-world analog signals. To avoid surprises, it's best to introduce analog I/O early in the design process; the ideal point is in the modeling phase with Xilinx® System Generator for DSP, under Simulink® from The MathWorks.

Consider the development of a digital QPSK demodulator in a software-defined radio. The FPGA performs several signal processing tasks, including carrier recovery DPLL (digital phase-locked loop), down conversion to baseband, down-sampling, pulse-shaping, and symbol timing recovery. You may wish to compare simulated demodulator performance against the real thing by injecting a heterodyned analog signal with noise through an ADC (analog-to-digital converter) to the FPGA running at full clock speed.

We need a framework to receive digitally sampled analog input signals from an ADC as stimuli into the Simulink model, either as real-time streaming data or as captured data for repeatable playback. We also need our framework to deliver processed data from the Simulink model to a DAC (digital-to-analog converter) to produce an analog output signal.

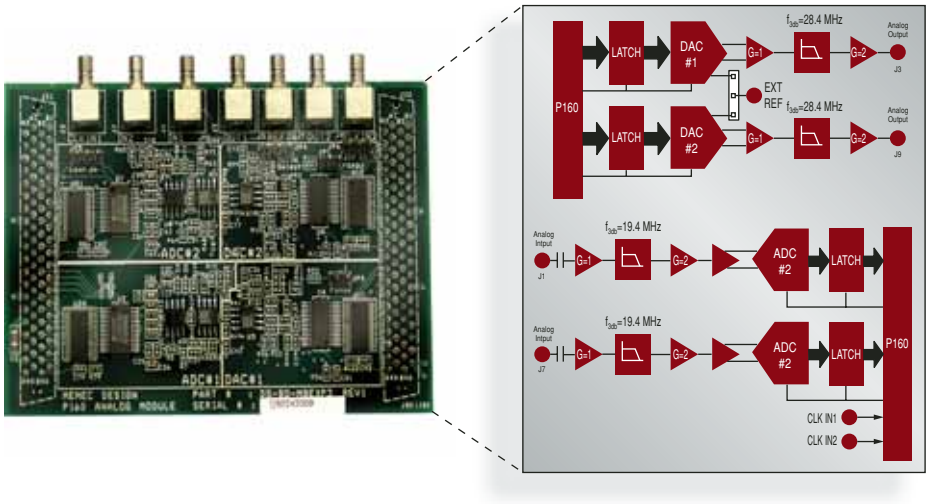


Figure 1 – Memec P160 Analog Module

The Memec™ P160 Analog Module is a daughtercard to interface external analog signals to Memec's wide assortment of Xilinx FPGA development boards. In this article, we'll present design techniques using the Memec P160 Analog Module in Simulink DSP models, making creative use of several new blocks from Xilinx System Generator version 6.3. If you are an FPGA designer, these techniques offer practical starting points, providing you with a head start on your DSP development using external analog signals.

Memec Simulink Library

The Memec P160 Analog Module is shown in Figure 1. It provides two channels of analog I/O through 12-bit data converters from Texas Instruments™:

- Two 165 megasamples per second DAC902 DACs, driving single-ended analog outputs.
- Two 53 megasamples per second ADS807 ADCs. The digital data out of the ADCs is latched into external buffers and then passed to the FPGA through the P160 interface.

The Memec P160 Analog Module DAC and ADC blocks are delivered as a Simulink library (shown in Figure 2). It offers the following features:

- Drag-and-drop P160 analog components from the Simulink library browser
- Supports HDL co-simulation from Simulink
- Supports common compilation types in Xilinx System Generator for DSP 6.3
  - Hardware co-simulation type: generates board-specific I/O ports to FPGA pins connected to the P160 Analog Module
  - HDL netlist type: generates top-level FPGA I/O pins connected to the P160 Analog Module
- Automatically detects target part/package from the System Generator token
- Supports all Memec FPGA development boards with P160 expansion connector
- Installer for automatic Simulink library creation

Interfacing to External Analog Signals

Let's describe three design techniques using various features of the P160 Analog Module to interface to analog signals during development of a DSP design in Simulink.

Memec P160 Analog DAC in HDL Co-Simulation

Our first design technique uses HDL co-simulation, a Xilinx System Generator feature that lets you incorporate your HDL code into Simulink though a black box. Simulink doesn't interpret HDL directly; rather, it invokes the Mentor Graphics® ModelSim™ HDL simulator, with which it exchanges I/O data during simulation.

When the design is compiled to hardware, the HDL code is included for synthesis. This technique is invaluable if your engineering staff wishes to preserve any existing investment in proven, re-usable HDL code.

System Generator blocks don't expose the system clock directly. Nevertheless, as digital designers, we sometimes prefer to see digital waveforms referenced to the clock, especially for logic that drives signals onto FPGA pins to off-chip. For this reason, the Memec P160 analog DAC block is built using a black box.

Figure 3 shows a model in which we drive the DAC block with a sinusoid signal stored in a ROM look-up table. The ModelSim waveform window opens automatically during simulation, displaying all inputs and outputs of the black boxes and all clock and clock enable signals supplied by System Generator. The signal display can be customized with an auxiliary Tcl script.

Figure 2 – Memec Xilinx DSP library in the Simulink library browser

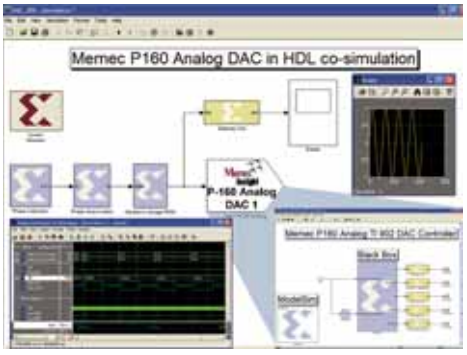


Figure 3 – Memec P160 Analog DAC in HDL co-simulation

Hardware Co-Simulation with the ADC

You can synchronize a System Generator hardware co-simulation block with its associated FPGA hardware in one of two clock modes. In single-step mode, the FPGA is clocked from Simulink; in free-running mode, the FPGA runs off an internal clock, and is sampled asynchronously when Simulink awakens the hardware co-simulation block. Let's examine a design that demonstrates switching between free-running and single-step modes.

The Memec P160 analog ADC controller from the Simulink library is built with gateway-ins defined as board-specific I/O ports. When used in a hardware co-simulation block, the Xilinx implementation tools bring these gateways to FPGA input pins, which connect to the P160 analog daughtercard, according to location constraints for the target device on the board. Consequently, sampled data from the P160 analog ADC can reach the FPGA.

As Figure 4 illustrates, we first compile the model on the left for hardware co-simulation and then connect it, as shown on the right. We use the Xilinx pause simulation block to switch clock modes of the co-simulation block.

Hardware co-simulation starts in free-running mode as we sample an analog input signal through the P160 ADC at a sampling rate derived from the system clock on the Memec development board. Sampled data fills a FIFO, while the Simulink model polls the FIFO's full flag asynchronously. When the full flag goes high, simulation pauses; the clock mode is switched to single-step and captured data samples are read out from the FIFO to Simulink. The captured data can be saved as stimuli for subsequent simulation.

ChipScope Validation of DAC-to-ADC Loopback

Our third design technique is useful to validate P160 analog DAC and ADC functionality in a loopback configuration. As shown in Figure 5, we generate a waveform that drives the P160 analog DAC to produce a continuous analog output signal. The waveform, stored in FPGA block RAM, is defined as an expression from The MathWorks MATLAB®; either a periodic function such as a sinusoid or arbitrary data from a MATLAB array.

When compiling the model to a bitstream, the DAC and ADC output ports are mapped to FPGA I/O that connects to the P160 Analog Module, according to the selected target FPGA.

In operation, a loopback cable connects the analog output signal from the P160 Analog Module DAC back into the ADC input. The Xilinx ChipScope™ tool – available as a block in System Generator – captures both the generated waveform and the sampled loopback version.

The design runs in the FPGA at the system clock rate – 100 MHz on most Memec development boards. The data-sampling rate is set in the model to a convenient sub-multiple of the system clock rate, as the ADC can operate at a maximum 53 megasamples per second.

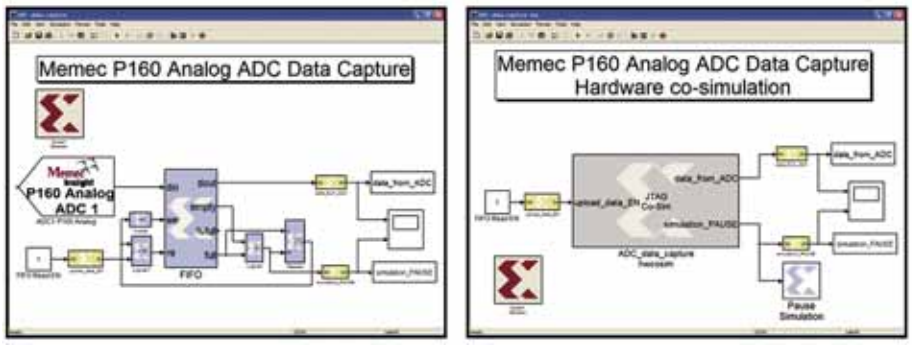


Figure 4 – Simulink model for data capture through the P160 ADC

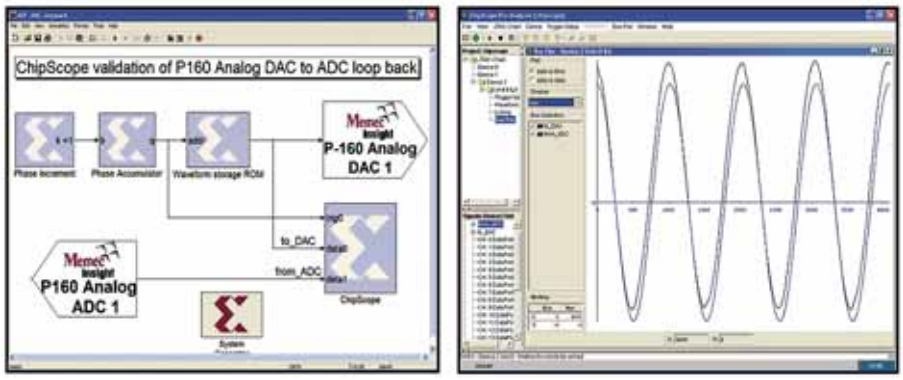


Figure 5 – ChipScope validation of P160 analog DAC-to-ADC loopback

Conclusion

The Memec P160 Analog Module is ideal for interfacing FPGAs to external analog signals. We have shown example techniques to quickly get you started to capture, process, and produce analog signals in your DSP applications under Simulink.

As your needs evolve to meet customer demand for ever-increasing analog I/O performance, expect Memec's next generation of its analog module to deliver faster sampling rates and higher resolution within a consistent support framework in System Generator for DSP and Simulink.

The P160 Analog Module specs, Memec Xilinx DSP Simulink library, and reference designs are available to all current P160 analog customers. You can obtain the free download through the Memec Reference Design Center at *http://legacy.memec.com/solutions/reference/xilinx/*.

X-ray vision for your designs

Agilent 16900 Series logic analysis system with FPGA dynamic probe

- *Increased visibility with FPGA dynamic probe*
- *Intuitive Windows® XP Pro user interface*
- *Accurate and reliable probing with soft touch connectorless probes*
- *16900 Series logic analysis system prices starting at $21,000*

Agilent Direct

Get a quick quote and/or FREE CD-ROM with video demos showing how you can reduce your development time.

U.S. 1-800-829-4444, Ad# 7909
Canada 1-877-894-4414, Ad# 7910
www.agilent.com/find/new16900
www.agilent.com/find/new16903quickquote

Now you can see inside your FPGA designs in a way that will save days of development time.

The FPGA dynamic probe, when combined with an Agilent 16900 Series logic analysis system, allows you to access different groups of signals to debug inside your FPGA—without requiring design changes. You'll increase visibility into internal FPGA activity by gaining access up to 64 internal signals with each debug pin.

You'll also be able to speed up system analysis with the 16900's hosted power mode—which enables you and your team to remotely access and operate the 16900 over the network from your fastest PCs.

The intuitive user interface makes the 16900 easy to get up and running. The touch-screen or mouse makes it simple to use, with prices to fit your budget. Optional soft touch connectorless probing solutions provide unprecedented reliability, convenience and the smallest probing footprint available. Contact Agilent Direct today to learn more.

Agilent Technologies
dreams made real

Build Custom Real-Time Video Applications Quickly and Easily

You can use a board equipped with all required video I/O and a Virtex-II FPGA to rapidly develop custom video processing functions.

by John L. Smith
Principal Engineer
Titan Corp., AP&D Division
john.l.smith@titan.com

Visually guided tele-operation is becoming ubiquitous in a variety of fields, including medicine, defense, and industry. A key requirement is low latency – there should be minimum delay between capturing video at the sensor and displaying it at the remote viewer. With training, people can get used to as much as a half-second of delay, but often the result is vehicle oscillation, as the operator over-corrects controls without having the intuitive immediate feedback.

Titan Corporation's Advanced Products and Design Division works with aerospace defense primary contractors, who provide unmanned aerial vehicles (UAVs) to the DoD. Figure 1 shows a General Atomics™ Predator UAV, and Figure 2 shows a ground station used for UAV remote control.

Figure 1 – Predator UAV

Figure 2 – Ground station console

In surveillance missions, MPEG-2 encoded video from a pan-tilt-zoom (PTZ) camera mounted on the UAV is transmitted to a ground station. There the imagery is presented on a console to the operators. For the most effective control of the camera and vehicle, we had to reduce delay through the MPEG-2 decoder to less than 75 ms.

To accomplish this task, we used our commercial off-the-shelf multimedia video processing board, the VigraWATCH™. VigraWATCH (VW) is equipped with a Xilinx® Virtex-II™ FPGA and an IBM™ PowerPC™ 440GP (PPC) processor. This

provides more than enough processing power to easily implement a customized MPEG-2 I-frame decoder, which far surpasses the minimum latency requirement.

With the overhead of circuit board development and a basic software framework in place, and by taking advantage of IP included in the Xilinx development toolset, we were able to get the job done in four months.

MPEG-2

MPEG-2 is a widely used video compression standard rich with diverse encoding methods. Its diversity includes three distinct techniques for coding individual video frames as either intra (I-frames), predicted (P-frames), or bi-directionally interpolated (B-frames). P-frames and B-frames introduce additional latency, both encoding and decoding. To cut latency to the absolute minimum, we used only I-frame encoding and decoding. Intra-frame encoding consists of a pipelined set of functions.

Basics

The three MPEG-2 coding methods are:

- I-frame – Intra-frame encoding is based solely on information within a single frame. Furthermore, the I-frame encoding and decoding process may begin as soon as the first 16 lines of a frame are received.
- P-frame – Predictive encoding uses a previous frame and encodes only the differences between that frame and the current frame to be encoded.
- B-frame – Bi-directionally encoded frames use both a previous (I or P) frame and a future (I or P) frame, forming a "best match" interpolation between those two frames and the current frame and encoding the resulting differences.

B-Frames Impede Low Latency and P-Frames Don't Help

Because B-frames use a future frame to encode the current frame, B-frame encoding and decoding impose a delay; the encoder (or decoder) must wait for the future frame to arrive before coding the current frame. Thus, B-frames must be tossed in the quest for minimum latency.

The P-frame's principal contribution to MPEG-2 is in improving compression ratios, as they are smaller than I-frames. A greater compression ratio means reduced transmission bandwidth. However, because low latency is the primary concern, the bandwidth needs to be enough to accommodate I-frames without buffering delays. We also had another latency issue – development time. Thus, we developed an I-frame-only decoder.

(Without P- and B-frames, MPEG-2 video becomes essentially the same as motion JPEG. In this case, we were constrained to MPEG because that was the source format.)

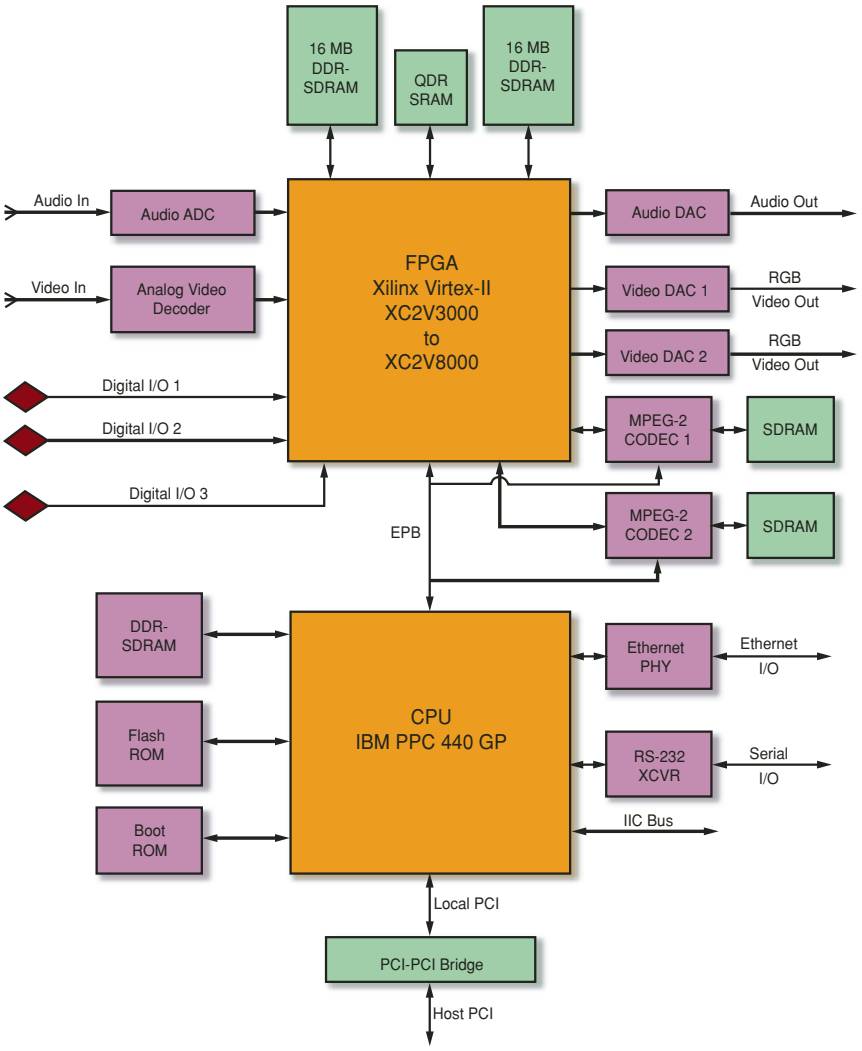


Figure 3 – VigraWATCH block diagram

The VigraWATCH Video Processor

The VW platform allows you to rapidly develop high-performance audio, video, and image processing functions using the XC2V3000, the microprocessor, or both. Figure 3 illustrates the VW's primary components, peripherals, and available I/O.

Primary Components

The VW contains five large ICs: an IBM PPC440GP, a Xilinx XC2V3000 FPGA, two Cirrus Logic™ MPEG-2 codecs, and a PCI-PCI bus bridge.

The PPC provides general-purpose processing. It has a dual-issue superscalar RISC core with 64-way associative I- and D-caches. It also manages PCI, RS-232, IIC, and Ethernet I/O. Chain-controlled DMA units are available in the PPC for

moving data between the PCI bus, the external peripheral bus (EPB), the PPC DRAM, and I/O registers.

The FPGA handles raw (uncompressed) audio, raw video, and raw and compressed I/O for the MPEG codecs. Part of the FPGA fabric is dedicated to video generators and mixers, I/O multiplexers, standard video processing such as scaling, and RAM interfaces. About 10% of the XC2V3000 is dedicated to a basic 2-D graphics engine. You can use the remainder of the FPGA fabric for custom processing functions. The default FPGA internal clock is 100 MHz, which matches the clock used for the DRAMs.

Each of the two MPEG-2 codecs is capable of encoding or decoding elementary streams. They are independent of each other. For example, in a video application where the raw video is enhanced by the FPGA, you can compress both the original and the enhanced video. In a communications scenario, one codec may be compressing local video for transmission, while the other is de-compressing remote video. Or you can use the two codecs to decompress video from two distinct remote sources.

The PCI-PCI bridge allows you to install VW in either 3.3V PCI or 5V PCI systems (the PPC is not 5V I/O tolerant).

Peripherals

Inputs to the VW FPGA include:

- A stereo audio digitizer
- A video digitizer/decoder

The video decoder accepts standard-definition NTSC and PAL format analog video from one of four composite sources or one of two S-video sources.

Outputs from the VW FPGA include:

- Two SVGA DACs, capable of driving independent displays
- An audio DAC producing standard line-level stereo audio output

The two DRAM banks attached to the FPGA are independent; each is capable of 1.6 Gbps peak bandwidth. One is associated with the graphics engine in the FPGA; the other is typically used by video processing functions.

Digital I/O connectors 1 and 2 each support 22 bi-directional LVTTL signals, as well as a few auxiliary connections. You can use the digital I/O to connect another board directly to the FPGA, or to connect two VW boards together. Digital I/O connector 3 has 16 LVTTL pins and can be used for a video interface port or as a convenient place to bring out de-bugging test points.

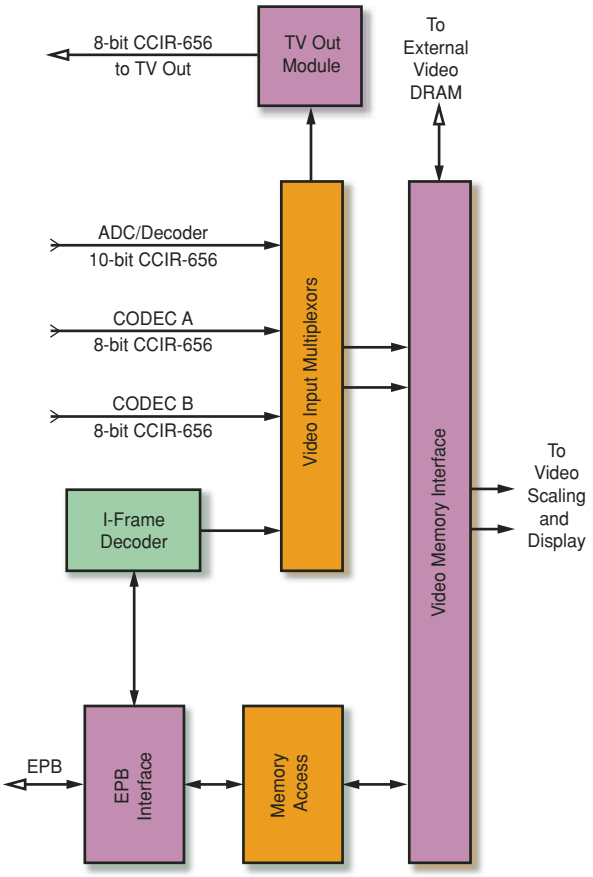


Figure 4 – Section of FPGA internal structure including I-frame decoder

Software

The PPC runs MontaVista™ Linux™, an embedded Linux supporting real-time functionality, multi-processes, and multi-threading. You can operate VW standalone, independent of any host computer, or as an add-in board driven by a host system. On Sun™ Solaris™-, Microsoft™ Windows™-, Wind River Systems™ VxWorks™-, or Linux-based host systems, graphic drivers allow VW to function as primary or secondary display. An API provides control of basic VW functions.

Building the I-Frame Decoder

We had a "clean room" software decoder developed from the MPEG specification available in-house at the start of the project. We partitioned the I-frame decoding functions into modules and did software profiling and hardware simulation to determine how to distribute the modules across the FPGA hardware and PPC software.

Integration with VW FPGA Internals

We connected the I-frame decoder inside the FPGA as a standard video input. Figure 4 shows a portion of the VW FPGA internals and how the decoder's two ports connect to the pre-existing circuitry. The EPB port carries encoded data, tables, and control register setup data from the PPC. The CCIR-656 video out port connects to a video multiplexer that selects between all of the video inputs. This allows us to re-use the existing design's video storage circuitry to move frame data into video memory, and ultimately to the display. Because the I-frame is processed sequentially, we can use internal block RAM to assemble macro blocks; a port to connect to external RAM is not required.

Decoding Modules

The pipeline layout of the decoder is shown in Figure 5. Input on the left is fed by the PPC. Output on the right is CCIR-656 format 4:2:2 YCbCr 8-bit video. This format matches the output from the VW peripheral analog video decoders. The layout was designed to allow progressive incremental design, integration, and testing of the modules.

The input buffer uses a 512-deep x 32-bit-wide FIFO to receive all data from the

PPC. This FIFO allows the relatively slow 66 MHz EPB bus to operate at full speed, without having to implement low-level hardware handshakes. A high-level handshake is implemented by making the FIFO's fill level available for read-back by the PPC.

The PPC core can keep track internally of the FIFO fill level and make decisions as to whether to work on filling the FIFO or perform other useful functions. The input buffer also contains an auto-incrementing register used to generate indirect addresses for rapidly filling tables in other modules, to keep the decoder's I/O address range on the EPB bus small.

The variable length (VL) decoder decodes the Huffman-encoded block coefficients according to MPEG-2 tables B-14 and B-15. State machines to traverse the Huffman code trees and a look-up table to extract run/level value pairs from the leafs both fit into a single Virtex-II block RAM configured as 1K deep x 16 bits wide.

We used some extra FPGA fabric for shift registers to handle escape codes for run/level values not included in the Huffman code tables. The ISDSM block handles the functions of inverting zigzag scanning, dequantization, and scaling.

The iDCT was the easiest block to design: it is included as a standard core in the Xilinx ISE CORE Generator™ package.

The format converter assembles the Y, Cb, and Cr sample blocks into slices in a slice-assembly RAM buffer comprising 16 block RAMs. The slices are then scanned out line by line and the lines are wrapped in CCIR-656 start and end active video (SAV/EAV) marker codes. We used an address rotation technique so new blocks can be assembled in the buffer as soon as a single line is removed, allowing the pipeline to run continuously without having to double-buffer the slice assembly RAM.

If we had to develop the board, all of the associated software and all of the IP that went into the low-latency decoder and display system would have taken years instead of months.

Results

The original unoptimized MPEG-2 codec chip external to the FPGA had a latency of ~1800 ms. Working with the codec chip manufacturer, we reduced their latency to 45 ms. The I-frame decoder we developed using the Xilinx FPGA and PPC has a latency of less than 2 ms.

Conclusion

We saved a lot of time and effort using prebuilt boards and IP in the development process. If we had to develop the board, all of the associated software and all of the IP that went into the low-latency decoder and display system would have taken years instead of months.

You can rapidly develop other video processing functions, including:

- Other codecs – H.264, MPEG-4, Motion JPEG2000
- Enhancement – linear and non-linear filters, super-resolution, histogram equalization/specification, de-convolution, warping
- Stabilization and mosaicing

For more information on MPEG-2, read the book, "MPEG Video Compression Standard," edited by Joan L. Mitchell et al. And for more information on the VigraWATCH system, visit *www.titan.com*, keyword search "VigraWATCH."

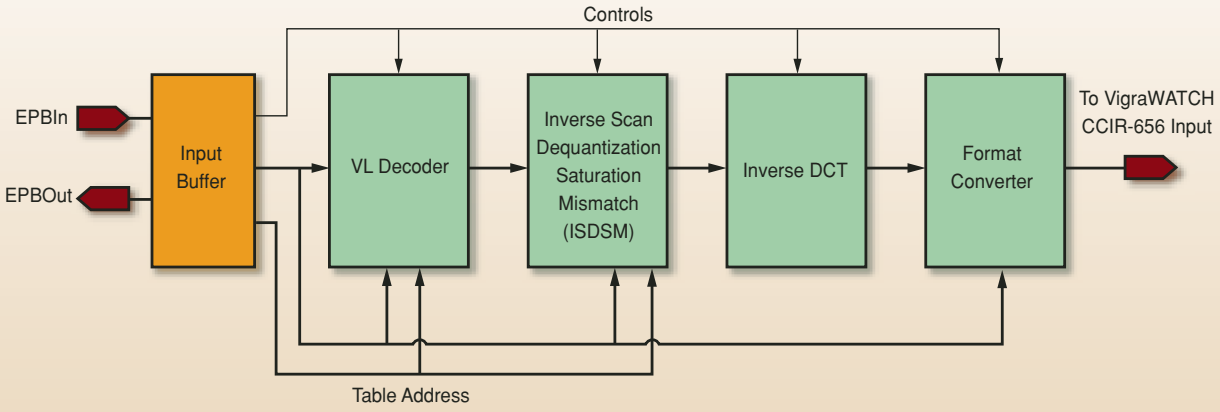


Figure 5 – I-frame decoder block diagram

Let System Generator Do the Handshaking

You can use the state control capabilities of Xilinx System Generator for synchronous digital DSP realization.

by T. Justin Campbell
FPGA Programmer
UVM
tcampbel@uvm.edu

If you are a DSP circuit designer, you should not feel restricted by the basic Xilinx logic blocks when building your design. Custom logic in a DSP circuit may not seem possible within the abstract world of Xilinx® System Generator, but on the contrary, you can easily realize custom logic by configuring a Xilinx MCode block.

A MATLAB® M-file from The MathWorks configures the block to emulate the algorithm realized in the file. You can attain custom control – and more specifically, state machines – with System Generator through configuration of a Xilinx black box, with code generated from Xilinx StateCAD. The VHDL code generated by StateCAD is emulated within the Xilinx black box block.

With advanced design and control logic, synchronization in DSP circuits also becomes an issue. You can realize handshaking (the exchange of control and status information between two blocks) in System Generator through delays and enable signals.

I would like to see Xilinx System Generator offer more flexibility (with the addition of output enables and other input parameters) to offer dynamic configuration of blocks. This flexibility comes at the expense of maintaining abstraction if you would prefer not to immerse yourself in the details of digital VLSI design.

Using a Xilinx MCode Block

If you are implementing a straightforward logic algorithm, configuring the MCode block is an easier solution than building the logic together through Xilinx blockset logic. Let's describe an example implementation of custom logic; in this case, a switching circuit used in a filter. The algorithm in MCode is shown in Figure 1.

```
function [tone_q, noise_q, filter_finish] =
switch_cir(d, BBRX_Start, tone_bin, BBRX_End, enable, BBRX_Finish)

if(BBRX_Finish==false)
  if((d>=BBRX_Start) &(d <=tone_bin-8))
     tone_q = false; nooise_q = true;
  elseif((d>=tone_bin-7)&(d<=tone_bin+7))
     tone_q = true; noise_q = false;
  elseif((d>=tone_bin+8)&( d<=BBRX_End))
     tone_q = false; noise_q = true;
  else
     tone_q = false; noise_q = false;
  end
else
  tone_q = false;
  noise_q = false;
end
if(( d==BBRX_End)&(enable==true))
  filter_finish = true;
else
  filter_finish = false
end
```

Figure 1 – MCode for a switching circuit

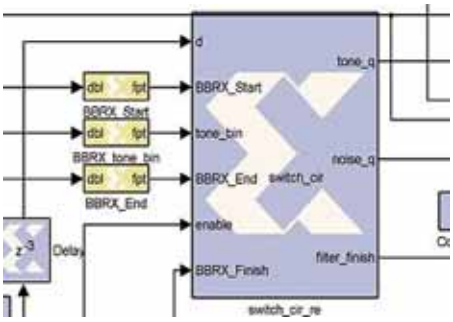


Figure 2 – Configured MCode switching circuit block

```
old_filter_finish=filter_finish;
if (enable = false)
      filter_finish = false;
elseif BBRX_end
      filter_finish = true;
else
      filter_finish = old_filter_finish;
end
```

Figure 3 – Pseudo code for the switching algorithm

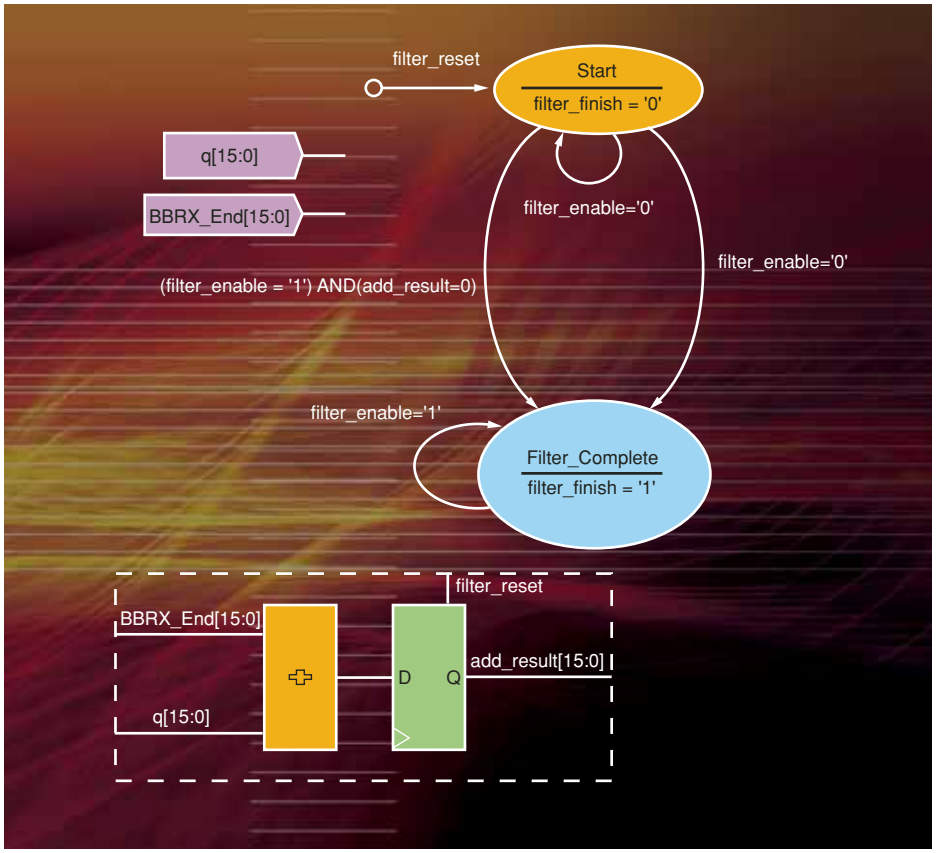


Figure 4 – StateCAD implementation for BBRX filter

You must place the M-file in the same directory as The MathWorks Simulink® model file, followed by selection and placement of the MCode block in the model file. In the parameter listing for the MCode block, a MATLAB function parameter exists; here you would type "switch_cir" (the name of the M-file). The block will then configure itself and emulate the logic of the file, as shown in Figure 2.

Currently, the Xilinx MCode block cannot hold an internal state. But if you would like to implement a state machine (capable of holding an internal state), there are other alternatives, such as generating VHDL code to emulate the state machine and implementation through a Xilinx black box configuration.

Xilinx StateCAD Configuration

A simple state machine is given with the algorithm shown in Figure 3.

The requirements of the signal *filter_finish* are described as follows: If the block is enabled, when the counter reaches the value of "BBRX_end," *filter_finish* should go high and stay high until enable goes low. The state machine shown in Figure 4 was generated with StateCAD to emulate this logic.

Note that the default state in the VHDL code must be changed manually to "start." This is because StateCAD's default state in terms of VHDL code is that whose name is first alphabetically. Thus, to avoid complications, you should always create a default state in which to start called "aaa."

You can then generate VHDL code (bbrx.vhd) for this state machine. The VHDL code can then be modified for configuration of a Xilinx black box. If a blank

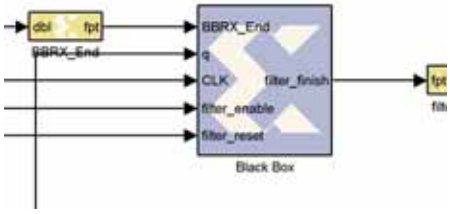


Figure 5 – Implementation of state machine algorithm through Xilinx black box configuration

Xilinx black box is in a model file, and the VHDL code to configure it resides in the folder in which the model file is saved, then the configuration wizard for the black box will automatically generate an .m file to describe the functionality of the black box. With System Generator 3.1, you can configure the black box manually.

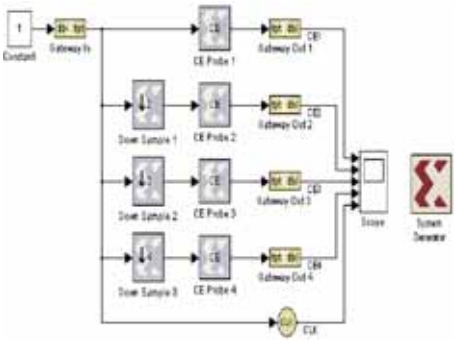


Figure 6 – Example model demonstrating clock enable probe use

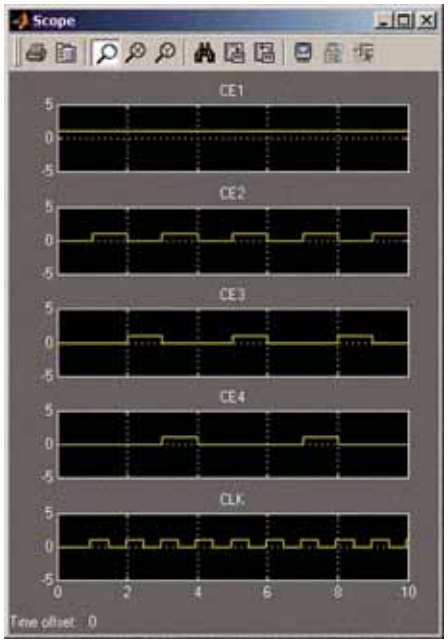


Figure 7 – Scope output from clock enable probe example

A problem exists when generating the .m file through the wizard. The configuration wizard for the black box cannot realize multiple entities in the .vhd file. The VHDL file bbrx.vhd contained multiple entities because of inefficient VHDL code generation through Xilinx StateCAD. Thus, you must manually manipulate the code to reduce it to one entity. You can then use the modified VHDL code in conjunction with the Xilinx black box wizard, creating a block shown like that in Figure 5.

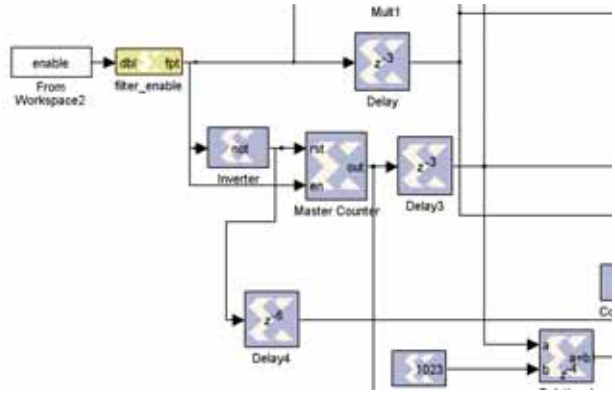


Figure 8 – Propagation of enable signal through delay blocks

System Generator 6.1 Features

The Fast Fourier Transform (FFT) implementation through configuration of a Xilinx black box block with M-file and VHDL wrapper file had problems in the past with execution in Simulink. These problems arose from the fact that System Generator did not appear to allow for multiple sampling rates (for instance, a clock and its respective down-sampled version).

This problem has since been alleviated with the addition of a clock enable probe system generator block. This block lets you effectively up- and down-sample a clock rate such that multiple clock rates are allowed within the same model. Figures 6 and 7 illustrate an example of the clock enable probe.

DSP Circuitry Synchronization

Synchronous design goes hand in hand with the development of DSP circuitry. Therefore, it is important to be able to realize synchronous design in the high-level abstraction that System Generator provides.

Note that Xilinx delay blocks are used for "delaying" enable signals for a duration that matches computation effort time. You can use the output of this delay as an effective "output enable," as shown in Figure 8. These delays are of such importance that before enabling the second block in a chain, you want to make sure the first block has completed its computation.

Exploring "FFT_power.mdl" demonstrates that latency requirements increase when the precision of the inputs and output of the multiplier block increase. Thus, the delays need to be modified when greater computational effort and thus greater time requirements in terms of Xilinx block latency result from overall design changes.

You could add greater flexibility to the Xilinx blockset with the addition of extra input parameters to some of the blocks in the set. For instance, the count-limited counter does not offer a count-to value as a possible input parameter. Therefore, dynamic configuration of the counter threshold is unrealizable.

Output enables allow one stage to signify it is complete, and thus for the next stage to start. Currently, most of the blocks in the blockset do not provide signals telling you when the block is finished its computation. This would be helpful in the handshaking for several Xilinx blocks.

However, you can realize output enable signals and counter threshold input signals by generating a VHDL file using the Xilinx Core Generator™ tool, and then configuring a Xilinx black box. But this requires more time from the designer and greater engineering effort as well.

We have to consider flexibility (the addition of output enables and other input parameters) to offer dynamic configuration of blocks versus a trade-off in maintaining the abstraction desired by DSP designers lacking strong digital design skills. It is therefore important to consider these ideas in future versions of System Generator.

Conclusion

A custom logic design may seem like a daunting task, but with the flexibility offered by Xilinx System Generator, it is quite achievable. Xilinx MCode and black box block configuration offer viable solutions for implementing custom logic.

System Generator is a very powerful and abstract tool, but we would like to see greater flexibility in terms of achieving synchronous design within System Generator.

Early Access: The Designer's Edge

Nu Horizons and Xilinx provide high-performance DSP technology tools with the online Nu Horizons Spartan-3-2000 platform.

by Steven Erck
Director Technical Marketing
Nu Horizons Electronics Corp.
serck@nuhorizons.com

Brian Seymour
Chief Technical Officer
TechOnLine
brian.seymour@techonline.com

Today's engineering environments are fraught with limited resources, tight schedules, and dramatic learning curves. One of the most important customer responsibilities we have as suppliers or distributors is to help simplify or lessen these constrictions.

Development boards, EDA software tools, and reference designs have long been the traditional tools for design engineers. But compiling these configurations does not relieve stress in critical areas, and more than likely can intensify the problem. Reducing time is the common critical element, and accelerating the development cycle should become an engineer's prime objective.

Nu Horizons Electronics Corp. has formed an exclusive partnership with Xilinx® to provide a complete online evaluation and development environment using TechOnLine™ VirtuaLab™ technology. Rather than waiting for parts and boards, you can now can evaluate the latest Xilinx technology, learn new tools, and take real measurements at the click of a button.

How VirtuaLab Works

TechOnLine has served the educational needs of the engineering community since 1995. Their solutions are designed to build awareness, educate and train audiences, and enable the real-time evaluation of hardware and software products over the Internet.

VirtuaLab goes beyond the software simulation usually associated with web-based design; you can access hardware and software with just a browser. From virtually anywhere in the world, design engineers with an Internet connection can experience the advantages of designing Xilinx DSP solutions with the Xilinx/Nu Horizons VirtuaLab.

When you begin a VirtuaLab session, you can be assured of a consistent experience and a controlled environment. You can schedule time on VirtuaLab 24/7 according to your local time zone, and the scheduled date can be integrated into your Microsoft™ Outlook calendar.

The first user interface is very familiar: a Windows™ or Linux™ desktop on which locally stored files can be moved onto an LDAP (lightweight directory access protocol) environment on TechOnLine's servers. In this way, benchmarks, applications under development, and other proprietary software can be run on the target board, within a personalized file structure. Security is always a top priority.

From the Windows desktop, you move to the remote target control interface, which provides remote control of the hardware with the same capabilities you would have as if the board was connected to your local PC. Additional functionalities include the ability to:

- Reset the platform
- Power-cycle the platform
- View a Web-cam display of the VirtuaLab
- Control voltage
- View a component description
- Observe LEDs
- Utilize a virtual touch screen

The remote target control gives you the ability to stimulate and observe the hardware, and can best be described as providing access to all the features of the hardware by having it on the Windows desktop. During a VirtuaLab session, you will have exclusive use of the evaluation platform and access to a complete Integrated Development Environment (IDE) that includes sample Xilinx high-performance DSP application software, as well as the ability to compile and upload code or any other application.

An innovative support feature called "shadowing" allows Nu Horizons' field applications engineers to log into your session, providing you with technical assistance if needed.

The Nu Horizons/Xilinx VirtuaLab

Architects and designers can reserve exclusive blocks of time by simply logging onto *VirtuaLab.TechOnLine.com* to register. You can select one of three types of interactive VirtuaLab experiences:

1. **New Xilinx user** – Although technically astute, you have not been exposed to either the development tool flow or the Xilinx FPGA architecture.
2. **Xilinx-aware user** – You have designed with previous Xilinx products but may not know about the latest solutions.
3. **Expert Xilinx user** – You are proficient in both standard and advanced products architecture, and are licensed in most or all of the EDA design tools available for development.

The opportunity to return and continue your studies or design work is always possible with VirtuaLabs' LDAP environment, which provides secure storage; however, you can always transfer your files if you prefer.

Once in the VirtuaLab environment, you can compile code, measure board performance, and set breakpoints – anything that you could do if the evaluation board was being controlled by your own PC.

What Can I Evaluate?

Working with Xilinx, the Nu Horizons engineering teams offer a full portfolio of educational aids, as well as direct access to Xilinx design flow tools and high-speed test equipment.

Our first laboratory consists of the Nu Horizons Spartan-3™-2000 evaluation platform (Figure 1), which is a very flexible testing platform that allows you to evaluate the Xilinx XC3S2000 FPGA and develop a multitude of applications. One of the Nu Horizons/Xilinx VirtuaLab applications is focused on high-performance FPGA DSP functionality.

You can develop advanced algorithms and perform complex measurements through a full complement of test equipment connected to the Spartan-3-2000 environment. Both a signal/pattern generator and high-speed oscilloscope are connected to high-speed analog-to-digital converter (ADC) and digital-to-analog converter (DAC) modules, and each piece of Agilent™ test equipment is placed in the host mode so you can remotely manage the equipment's front panel controls. You can save settings and scripts easily within your I- or H-drive private folders to re-use for future sessions.

Having real signal insertion and being able to measure real output means that you can validate your algorithms, transforms, and functions, and simultaneously have confidence that all results are both precise and authentic.

Reference DSP designs are provided within the VirtuaLab. These designs allow you to evaluate the Spartan-3-2000 FPGA in a pre-verified environment. Reference designs include:

- Existing System Generator tutorial to introduce engineers to other features such as the ChipScope™ tool, HDL co-simulation, hardware co-simulation, and the PicoBlaze™ processor.
- Simple FFT with a 256-tap FIR filter and interpolation by three. Ease of use and reasonably high performance allow you to evaluate the tool interface as well as the hardware. The filter design is provided using the FDA tool to generate the coefficients, allowing you to modify the coefficients and view the results. Simulation can be run in System Generator, as well as live in hardware.
- Equalized 16-QAM demodulator, including the adaptive filter. The receiver architecture provides subsystems that demonstrate adaptive channel equalization and carrier tracking on a random QAM data source.

The Spartan-3-2000 platform delivers acquisition/conversion capability through two high-performance plug-in modules and two mid-range performance platform solutions.

ADC Platform Solution

The Nu Horizons Spartan-3-2000 evaluation platform includes a mid-range ADC

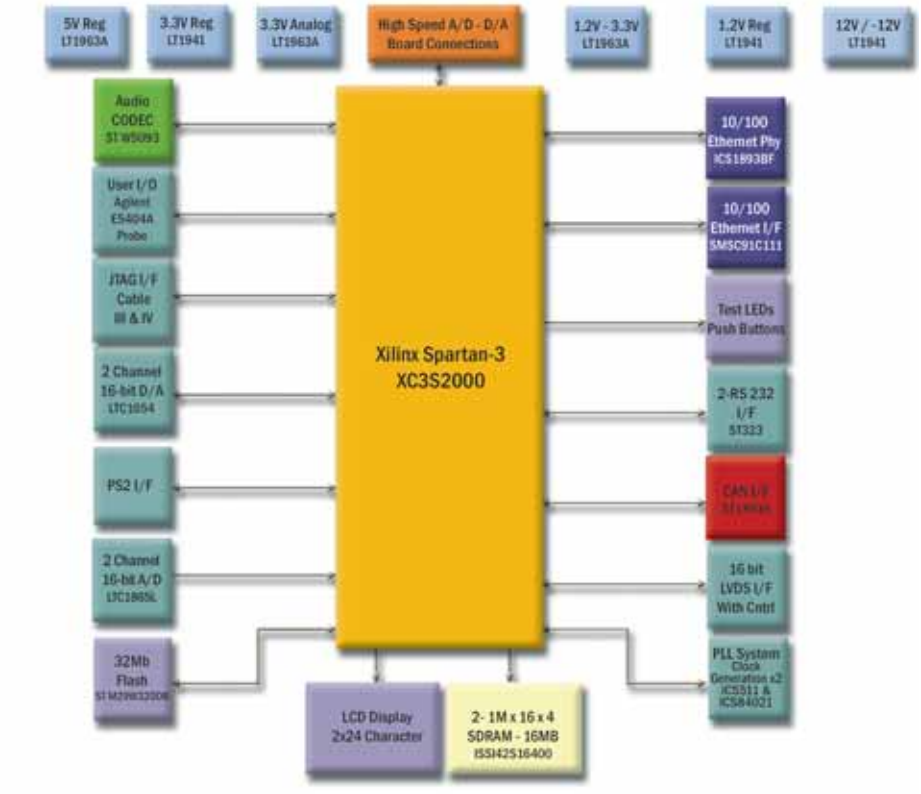


Figure 1 – Nu Horizons Spartan-3-2000 evaluation platform

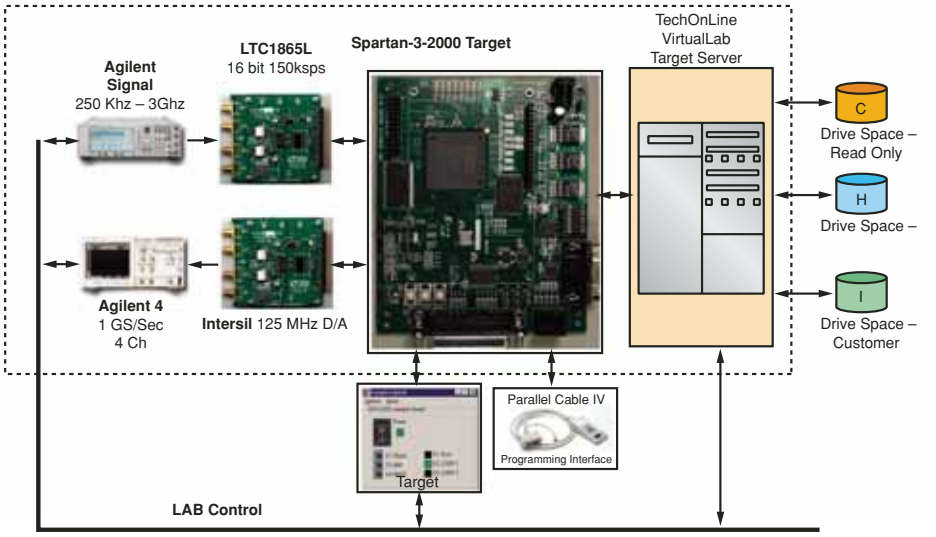


Figure 2 – TechOnLine VirtuaLab block diagram

on the main platform. You can use the Linear Technology™ LTC1865L 16-bit 150 ksps ADC in ratio metric applications, or with external references. The high-impedance analog inputs and the ability to operate with reduced spans down to 1V full scale allow direct connection to signal sources in many applications – eliminating the need for external gain stages.

High-Performance A/D Acquisition Module

The high-performance ADC acquisition modules interface directly to the Nu Horizons Spartan-3-2000 evaluation platform also provided by Linear Technology. These ADC development boards offer 12-14 bits of resolution, and sampling rates from 25-40 MHz.

The Linear Technology ADC development boards (see Table 1) are designed to digitize high-frequency, wide-dynamic range signals. These boards target applications such as telecommunications, digital imaging, spectrum analysis, and cellular base stations.

520B-A	LTC1748	14 bit	80 Msps	Ain < 40 MHz
520B-B	LTC1748	14 bit	80 Msps	Ain > 40 MHz
520B-C	LTC1745	12 bit	25 Msps	Ain < 40 MHz
520B-D	LTC1746	14 bit	25 Msps	Ain < 40 MHz
520B-E	LTC1747	12 bit	80 Msps	Ain < 40 MHz
520B-F	LTC1747	12 bit	65 Msps	Ain > 40 MHz
520B-G	LTC1742	14 bit	65 Msps	Ain < 40 MHz
520B-H	LTC1742	14 bit	65 Msps	Ain > 40 MHz
520B-I	LTC1741	12 bit	65 Msps	Ain < 40 MHz
520B-J	LTC1741	12 bit	65 Msps	Ain > 40 MHz
520B-K	LTC1743	12 bit	50 Msps	Ain < 40 MHz

Table 1 – Linear Technology high-speed acquisition modules 25-80 Msps

ISL5629EVAL1	2 x 8	210 MHz
ISL5729EVAL1	2 x 10	210 MHz
ISL5829EVAL1	2 x 12	210 MHz
ISL5929EVAL1	2 x 14	210 MHz

Table 2 – Intersil high-speed conversion modules

DAC Platform Solution

The Spartan-3-2000 evaluation platform includes Linear Technology's LTC1654L 14-bit, 8 conversion rate DAC. The LTC1654 is a dual rail-to-rail voltage output 14-bit DAC that includes output buffer amplifiers and a flexible serial interface. The LTC1654L has two programmable speeds: a fast and slow mode with ±1 LSB settling times of 3.5 ms or 8 ms, respectively, and supply currents of 750 mA and 450 mA in the two modes. The LTC1654 also has shut-down capability, power-on reset, and a clear function to 0V.

High-Performance DAC Module

The Spartan-3-2000 evaluation platform includes an interface to Intersil™ high-speed DAC ISL5x29EVAL1 evaluation modules (see Table 2).

You can develop high-performance DSP applications, such as quadrature transmit with an IF range of 0-80 MHz and medical/test instrumentation and equipment. You can evaluate the Intersil technology in conjunction with the Xilinx Spartan-3 FPGA in high-performance DSP designs.

Figure 2 illustrates how you can load a DSP application, provide signal insertion, and measure output waveform.

Conclusion

Today's semiconductor industry is largely driven by new technologies that are barely available, and aggressive development schedules that utilize these new technologies. Suppliers and distributors typically deliver boxes of building blocks, and if the design engineer works many extra hours assembling the pieces, they may wonder if this new technology can even perform the task at hand.

Nu Horizons and Xilinx have looked closely at the challenge of limited resources, tight schedules, and dramatic learning curves and built an approach with TechOnLine's VirtuaLab that provides you with a time-saving, innovative, and zero-cost solution to evaluating technology.

In addition to its primary use by engineers, the VirtuaLab can help other departments as well. Online product evaluation, sales demonstrations, and internal training are example applications of VirtuaLab technology. Each VirtuaLab is delivered with a range of tools including a product tutorial or quick start script – enabling novices to exercise the hardware without writing any code, or a complete IDE for the expert expecting a full-featured experience.

For more information, visit the Nu Horizons/Xilinx VirtuaLab at *VirtuaLab.TechOnLine.com* or contact your local Nu Horizons sales representative.

What would you do with *advanced access* to new technology?

Online Design with Spartan-3 2000 FPGAs for a limited time only

Evaluate the new Xilinx Spartan-3 2000 FPGA today!

Today's engineering environments consist of limited resources, tight schedules, and dramatic learning curves. Waiting for high volumes of Silicon production can add to the frustration of completing a design and being fi st to market with your solution. That's why Nu Horizons and Xilinx have partnered to provide engineers with a complete online evaluation and development environment using TechOnLine's VirtuaLab℠ technology.

Save time. Be innovative... with a zero-cost solution.

Check it out. TechOnLine's VirtuaLab goes beyond the software simulation usually associated with Web-based design. Hardware and software can be accessed over the Web with only a browser. From virtually anywhere in the world with an Internet connection, you will experience the advantages of designing with the newest products from Xilinx. Our fi st Laboratory allows you to evaluate the Xilinx XC3S2000 FPGA, and develop a multitude of applications. One of several Nu Horizons / Xilinx Virtual Lab applications is centered on high-performance FPGA DSP functionality. Others include Embedded Processing and high-speed Serial I/O.

Why wait?

For a limited time... evaluate the latest Xilinx technology, learn new tools and take real measurements, without any initial investment. What would you do with advanced access to new technology?

DISCOVER THE POSSIBILITIES OF ONLINE DESIGN

NU HORIZONS ELECTRONICS CORP

For more information visit
www.nuhorizons.com/VirtuaLab

TechOnLine

FPGA Power Solutions

High Performance Analog Solutions from Linear Technology

Dual Output DC/DC Converter Solutions for Xilinx FPGA-Based Systems

Xilinx FPGAs require at least two power supplies: V_{CCINT} for core circuitry and V_{CCO} for I/O interface. For the latest Xilinx FPGAs, including Virtex-II Pro, Virtex-II and Spartan-3, a third auxiliary supply, V_{CCAUX} may be needed. In most cases, V_{CCAUX} can share a power supply with V_{CCO}. The core voltages, V_{CCINT}, for most Xilinx FPGAs, range from 1.2V to 2.5V. Some mature products have 3V, 3.3V or 5V core voltages. Table 1 shows the core voltage requirement for most of the FPGA device families. Typical I/O voltages (V_{CCO}) vary from 1.2V to 3.3V. For Virtex-II Pro and Spartan-3, the auxiliary voltage V_{CCAUX} is 2.5V. It is 3.3V for Virtex-II.

Each FPGA family has a specific quiescent supply current, ranging from under 100mA to about 2A. For applications with multiple FPGAs, the core supply current can be higher than 10A.

With multiple voltage rails in today's systems (FPGA, DDR memory, data converter ICs, etc.), supply sequencing and tracking are quite important for proper start-up and shutdown. Ramp time requirement should also be satisfied. For example, the recommended ramp time (t_{CCPO}) for the core voltage V_{CCINT} is less than 50ms during power-on. Some Xilinx FPGA families also have minimum V_{CCINT} ramp time requirements.

New dual output DC/DC regulators from Linear Technology, the LTC®3407, LTC3736 and LTC3708, greatly simplify the design of an optimal power supply solution for systems using Xilinx FPGAs.

LTC3407: Dual Synchronous, 600mA, DC/DC Regulator

The LTC3407 is a dual synchronous step-down DC/DC converter with integrated power switches. It provides a compact and high efficiency power solution for FPGAs with supply currents up to 600mA. The switching regulator operates from a 2.5V to 5.5V input voltage range and has an adjustable output range from 0.6V to 5V. Its internal 1A switches provide up to 96% efficienc , eliminating the need for external MOSFETs and Schottky diodes. Figure 1 is an application example for 2.5V/600mA and 1.8V/600mA supplies.

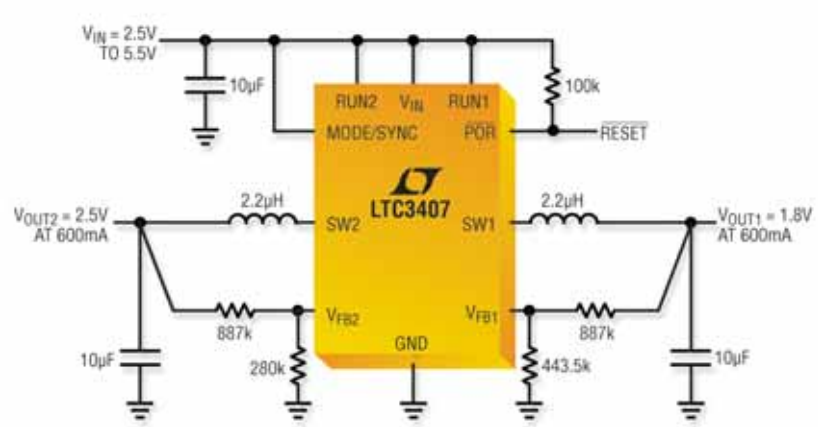


Figure 1. High Efficiency 2.5V/600mA and 1.8V/600mA Regulators

Table 1. Core Voltage Requirement for Xilinx FPGA Families

	Virtex-II Pro	Virtex-II	Virtex-E Extended Memory	Virtex-E	Virtex	Spartan-3	Spartan-IIE	Spartan-II	Spartan-X	Spartan
V_{CCINT}	1.5V	1.5V	1.8V	1.8V	2.5V	1.2V	1.8V	2.5V	3.3V	5V

FPGA Power Solutions

Figure 2 shows the efficiency curves of the circuit vs load current.

The LTC3407 has a constant 1.5MHz switching frequency, allowing the use of tiny inductors and capacitors. Selectable Burst Mode® operation provides high eff ciency at light loads. The IC has short-circuit protection and a power-on reset (power good) output. It is available in small thermally enhanced 10-lead MSOP and 3mm x 3mm DFN packages.

LTC3736: 2-Phase, Dual Synchronous, DC/DC Controller for 5A Loads

The LTC3736 is a 2-phase dual synchronous step down DC/DC controller. Power supplies using the LTC3736 can provide 5A at both outputs with a 5V input, meeting the load current requirements for most FPGA applications. The LTC3736 receives input from 2.7V to 9.8V and produces output voltages ranging from 0.6V to 9.5V. Figure 3 shows that up to 95% eff ciency is achieved. An application example is shown in Figure 4. In contrast to single-phase operation, the two channels of a 2-phase switching converter are operated 180° out of phase. This technique interleaves the current pulses coming from the topside MOSFET switches, greatly reducing the total RMS input ripple current. This in turn allows the use of smaller and lower cost input capacitors, reduces the EMI attenuation requirement and improves operating eff ciency.

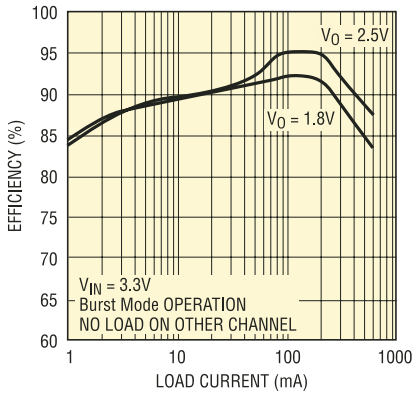


Figure 2. LTC3407 Efficiency Curve

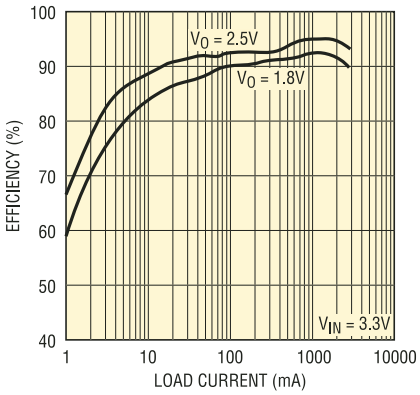


Figure 3. Efficiency vs Load Current for the LTC3736 Converter

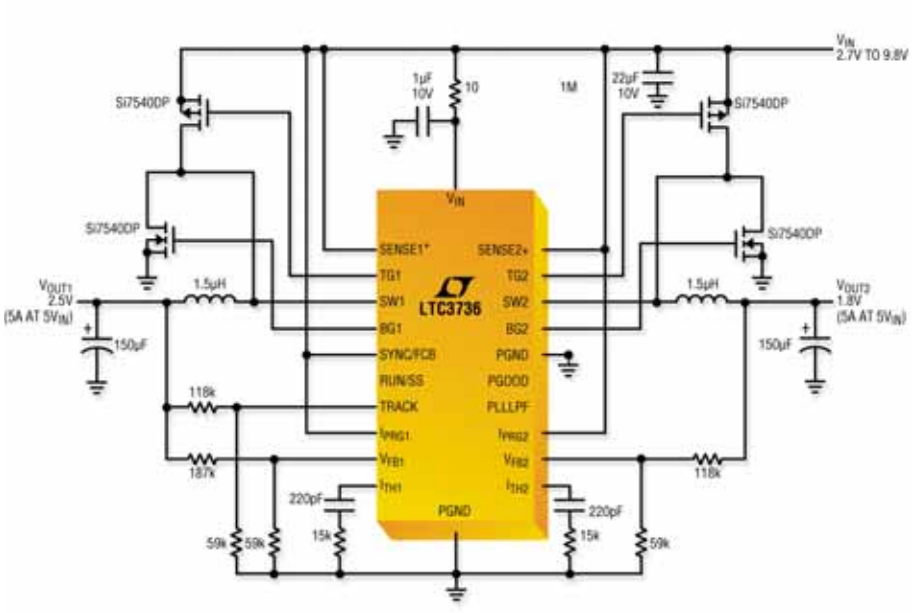


Figure 4. High Efficiency 2.5V/5A and 1.8V/5A Dual Output Converter with Output Tracking

Figure 5 compares the input waveforms for a representative single-phase dual switching converter to the 2-phase dual switching converter. Figure 6 shows how the RMS input current varies for single phase and 2-phase dual controllers with 2.5V and 1.8V outputs over a wide input voltage range. For most applications, 2-phase operation will reduce the input capacitor RMS current requirement to that of just one channel operating at maximum current and 50% duty cycle. The LTC3736 has a default switching

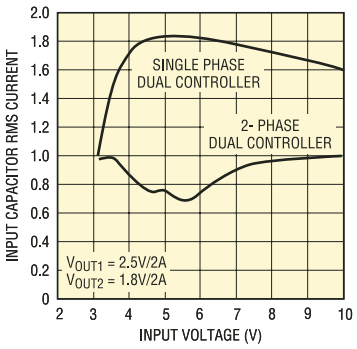


Figure 6. RMS Input Current Comparison

frequency at 550kHz, but can be externally synchronized from 300kHz to 750kHz. The LTC3736 provides output tracking for controlled ramp-up of two supply rails, programmable current limit, output overvoltage protection, power good output and selectable Burst Mode operation for high effciency light load operations. The device is available in the tiny (4mm x 4mm), thermally enhanced QFN package or the 24-lead SSOP package.

LTC3708: 2-Phase, Dual Synchronous, DC/DC Controller for 15A Loads

The core supply voltages of the latest Xilinx FPGAs have decreased towards 1V. The Virtex-II pro family requires 1.5V V_{CCINT} and the Spartan-3 family

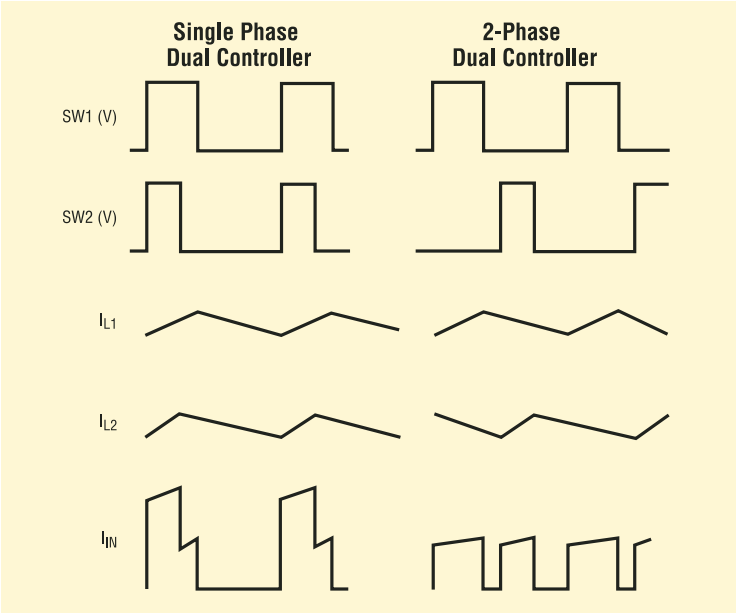


Figure 5. Example Waveforms for a Single-Phase Dual Converter vs the 2-Phase LTC3736

needs only 1.2V V_{CCINT}. In the meantime, these FPGAs demand more current from the power supplies. Some systems use more than ten FPGAs/board so the resulting total supply current can easily exceed 10A. The LTC3708-based dual output supply is an ideal choice for such applications.

The LTC3708 is a 2-phase dual synchronous step-down DC/DC controller with a wide input voltage range from 3.3V up to 36V. Its output voltage can be programmed down to 0.6V. Figure 7 shows the schematic of a dual output 2.5V/15A and 1.8V/15A converter.

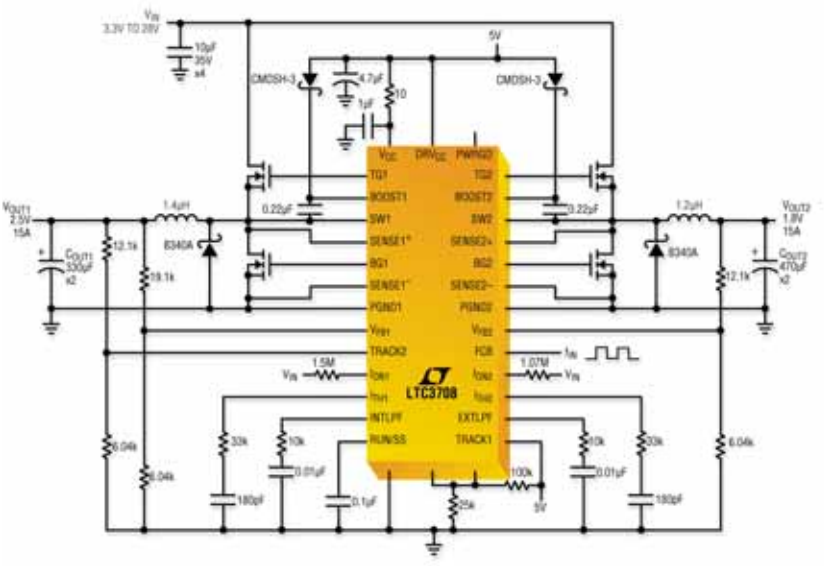


Figure 7. High Efficiency 2.5V/15A and 1.8V/15A Dual Output Converter with Output Tracking

FPGA Power Solutions

As shown in Figure 8, up to 95% efficiency can be achieved. The LTC3708 has output voltage up/down tracking capability. It allows both coincident and ratiometric tracking, as shown in Figures 9 and 10. The ramp rate can be selected by a soft-start capacitor from RUN/SS pin to ground. Multiple LTC3708s can easily be daisy-chained in applications requiring more than two voltages to be tracked. The 2-phase operation of the LTC3708 reduces power loss and noise and lowers the input-filtering requirement.

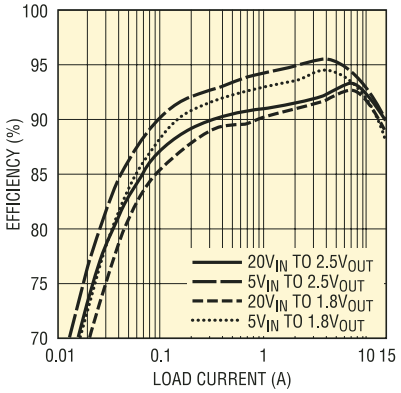


Figure 8. Efficiency vs Load Current for the LTC3708 Converter

The constant on-time, valley current mode control in the LTC3708 allows fast transient response, minimizing the number of output capacitors. An internal phase locked loop allows the IC to be synchronized to an external clock for applications with more than two output rails. The LTC3708 also features programmable current limit, output overvoltage protection and power good output. The device is available in the 5mm 5mm QFN package.

An optimal power solution for multirail supply systems incorporating the latest Xilinx FPGAs should provide multiple outputs with supply tracking/sequencing. As board real estate becomes more expensive, the power supply must be more efficient and smaller while supplying higher current in high-end applications. Linear Technology's latest dual output power management ICs (LTC3407, LTC3736 and LTC3708) successfully address these challenges. For data sheets and additional information on other power solutions for Xilinx FPGAs, visit Linear Technology's website at www.linear.com.

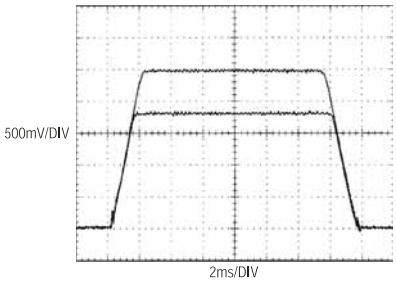


Figure 9. Up/Down Coincident Tracking

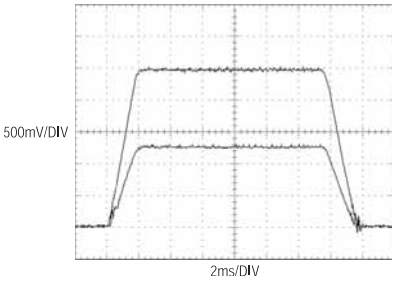


Figure 10. Up/Down Ratiometric Tracking

Implementing 70 High-Speed Channels with 9 FPGAs

Using nine Xilinx XC2VP7 circuits on a data concentrator card greatly reduced costs and PCB design effort and increased board reliability.

by Jose C. Da Silva
Design Engineer
LIP (Laboratorio Instrumentacao e Particulas) – Lisbon
jc.silva@cern.ch

Adarsh Jain
Design Engineer
LIP (Laboratorio Instrumentacao e Particulas) – Lisbon
adarsh.jain@cern.ch

Implementing 70 high-speed differential pairs on a 9U PCB using regular off-the-shelf deserializers can be a nightmare; high-speed PCB design, noise, clock jitter, and signal integrity are the main challenges. Even the smallest deserializer packages would occupy roughly two-thirds of a 9U board, on which you would still need space for the logic – configuration, memories, access interfaces, and local control.

Our design concerns a data concentrator card (DCC), part of a large high-energy physics experiment at the European Organization for Nuclear Research (CERN) in Geneva. A very large particle accelerator called the Large Hadron Collider (LHC) is being constructed near the Franco-Swiss border west of Geneva. A number of experiments will be conducted to observe and measure the various properties of several existing, and possibly new, fundamental particles.

We picked (9 Xilinx Virtex-II Pro devices), as it meant a significant savings in device count (from 105 to 9).

One such experiment is called the Compact Muon Solenoid (CMS), which is based on a large superconducting magnet system. The CMS will have a number of sub-detectors, including an Electromagnetic Calorimeter (ECAL). The ECAL will use about 80,000 crystals to capture the energy of the photons and electrons. The data collected from these crystals will be captured, processed, and transmitted by the DCCs (about 60 of them) for further analysis.

Design Overview

The DCC includes 70 high-speed optical receiver channels (6 blocks of 12 channels each) implemented on a 9U VME board (36 cm x 40 cm) working at 800 Mbps using a 2-byte 8b/10b protocol.

For the implementation of the transceivers, we had two choices:

1. As many as 70 discreet deserializers, along with 35 FPGAs for the required control (this number was based on cost considerations), for a total device count of 105. This would have given us more granularity and a lower cost, but more components and hence higher debug and testing times.
2. Only nine Xilinx® Virtex-II Pro™ devices with eight embedded RocketIO™ transceivers on each (only the XC2VP7-FG456 part was available at the time). We would lose some granularity, but the PCB would be much less dense and easier to test.

We picked the second choice, as it meant a significant savings in device count (from 105 to 9). And because the DCCs will be in operation for four to five years, it will have a huge impact on overall PCB design and the final cost of production and maintenance from a long-term perspective.

Also, after deserialization, we will need to verify the integrity of received data and reformat it for downstream processing and analysis. We found that the remaining resources in the selected device were enough for most purposes. Of the 72 transceivers available, we use 70 and leave the other two unconnected. The use of 800 Mbps per channel is a system choice, but the design could work at 1.6 Gbps or higher.

PCB Design Issues

The DCC PCB is a 12-layer board with four power planes and eight routing layers.

We have mostly followed the main rules for high-speed design and analog considerations from Chapter 4 of the Xilinx RocketIO™ Transceiver User Guide, such as:

- All high-speed traces are impedance controlled and routed manually in "microstrip-edge couple differential pair," with impedance matched to 50 Ohms and as close as possible to the source (respecting the crosstalk rules). No other lines were designed in the same area as the high-speed layout, where the immediate layer was the ground power plane.
- All high-speed differential pair signals were AC coupled with 100 nf capacitors and internally terminated to 50 Ohms.
- All of the transceivers' power supply pins were filtered with an individual LC filter and a separate power plane for the "analog" supply, also with specific filters. No transceiver power supply was left unconnected, regardless of whether it was used or not. We used the same type of LC filters on the optical receivers.
- Approximately 350 power supply decoupling capacitors of three different values (to match the main clock frequencies in use on the board) were placed as close as possible to the central power pins of the Xilinx FPGAs. Other capacitors were placed nearby each FPGA.
- Each FPGA received one high-quality reference clock (low jitter – 100 ps peak-to-peak) differential pair from an individual buffer. We recommend using two independent reference clock sources to ease the internal usage of this clock on the FPGA if using all of the RocketIO transceivers.

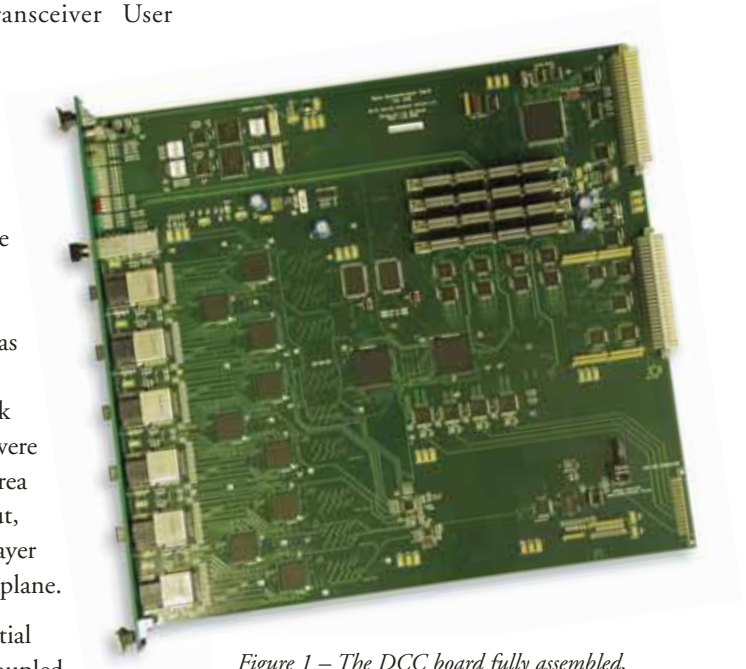


Figure 1 – The DCC board fully assembled, with the nine Virtex-II Pro FPGAs on the left.

RocketIO Implementation and Issues

Virtex-II Pro devices provide the first stage of processing for the front-end data (received from the on-detector electronics) on the DCC board. Each device receives 800 Mbps of serial data on each of its eight channels from the optical receivers, for a total of 6.4 Gbps per device. In a nutshell, the purpose of the Xilinx FPGAs is to process this data and prepare it for readout.

RocketIO transceivers are used to deserialize the received data and perform 8b/10b decoding. The 16-bit data is then

written in a programmable latency buffer to match the trigger latency. A number of data verification checks are carried out. The data is finally formatted into 64-bit words and written into FIFOs. From there, it is read out by the event builder on the board.

Without going into the details of the functionality, we will focus on the various issues we faced (and solved) in making the real hardware churn out correct data, with a focus on the use of RocketIO transceivers. Much of what we learned was on a trial-and-error basis. The main issue was related to the reference clock, which we'll describe in detail in the next section.

The other significant issue that we faced was the alignment of the K character within the 2-byte data path of the received data. We were initially using the Gigabit_Ethernet primitive in half-rate mode for a 2-byte data path. But we observed that not all of the channels were putting the K character in the same place within the 2-byte word and there was no way to force this alignment in the Gigabit_Ethernet primitive (the ALIGN_COMMA_MSB parameter of this primitive is set to FALSE by default).

Because our protocol expected the K to always appear on the LSB of the word, we switched to the GT_CUSTOM primitive, where we could force the alignment and subsequently swap the position of K to the LSB of the data. The simulations showed perfect alignment – but in real hardware, some of the channels were getting misaligned.

A colleague of ours referred us to the design note about 32-bit word comma alignment in the RocketIO transceiver user guide. Although this is usually needed only for a 4-byte data path, we implemented a similar scheme for our 2-byte data path and this fixed our misalignment problem.

Clock, Programming, and JTAG

We cannot over-emphasize the need for a high-quality reference clock. Besides satisfying all of the criteria specified in the RocketIO user manual, we made sure that our reference clock was as clean as we could possibly get (see Figure 2).

> … this is a flexible approach, as the FPGAs are reprogrammable and a more economical solution in the long term.

We used a quartz-based phase-locked loop (QPLL) circuit developed at CERN for our system to provide the best jitter-free clock source (100 ps peak-to-peak). We found that a lot of problems in the performance of the RocketIO devices could be traced to a noisy/jittery reference clock. If you are using RocketIO transceivers on both halves of the chip, then it's much better to have two reference clocks. We believe this helps even if you are running the RocketIO transceivers in half-rate mode (which is our case).

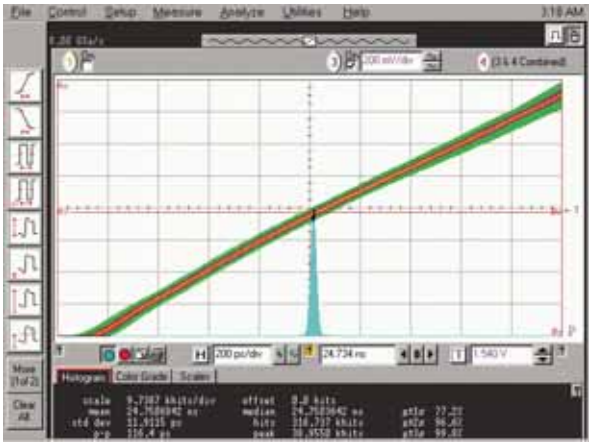


Figure 2 – Clock jitter measurement

Another aspect of the clocking scheme that we used was to pass the reference clock through a global clock buffer after an input global differential clock buffer. We observed improved stability and a more uniform distribution of the reference clock with the FPGA editor.

Also, though not directly related to the high-speed transceivers, we found that an independent post-configuration DCM reset logic (usually recommended if you have an external feedback clock) is useful even when using internal feedback. This solved a problem we were having with the DCMs where they were sometimes not locking after reconfiguration. Xilinx Technical Support helped us find the solution (Xilinx Answer Record 14425).

As for programming and JTAG, we used the same group of EPROMs to configure eight of the nine FPGAs. One of the FPGAs is the master and provides the clock for all the devices in the chain. The ninth FPGA has a different pinout and a separate EPROM for itself.

All circuits are connected in the same JTAG chain, which improved reprogramming time mainly during the "test" stages. We found that a need exists for a pull-up resistor on the TDO output of each Xilinx device, something that we hope Xilinx will add in future devices. The JTAG is used also to check the board interconnections after assembly.

Conclusion

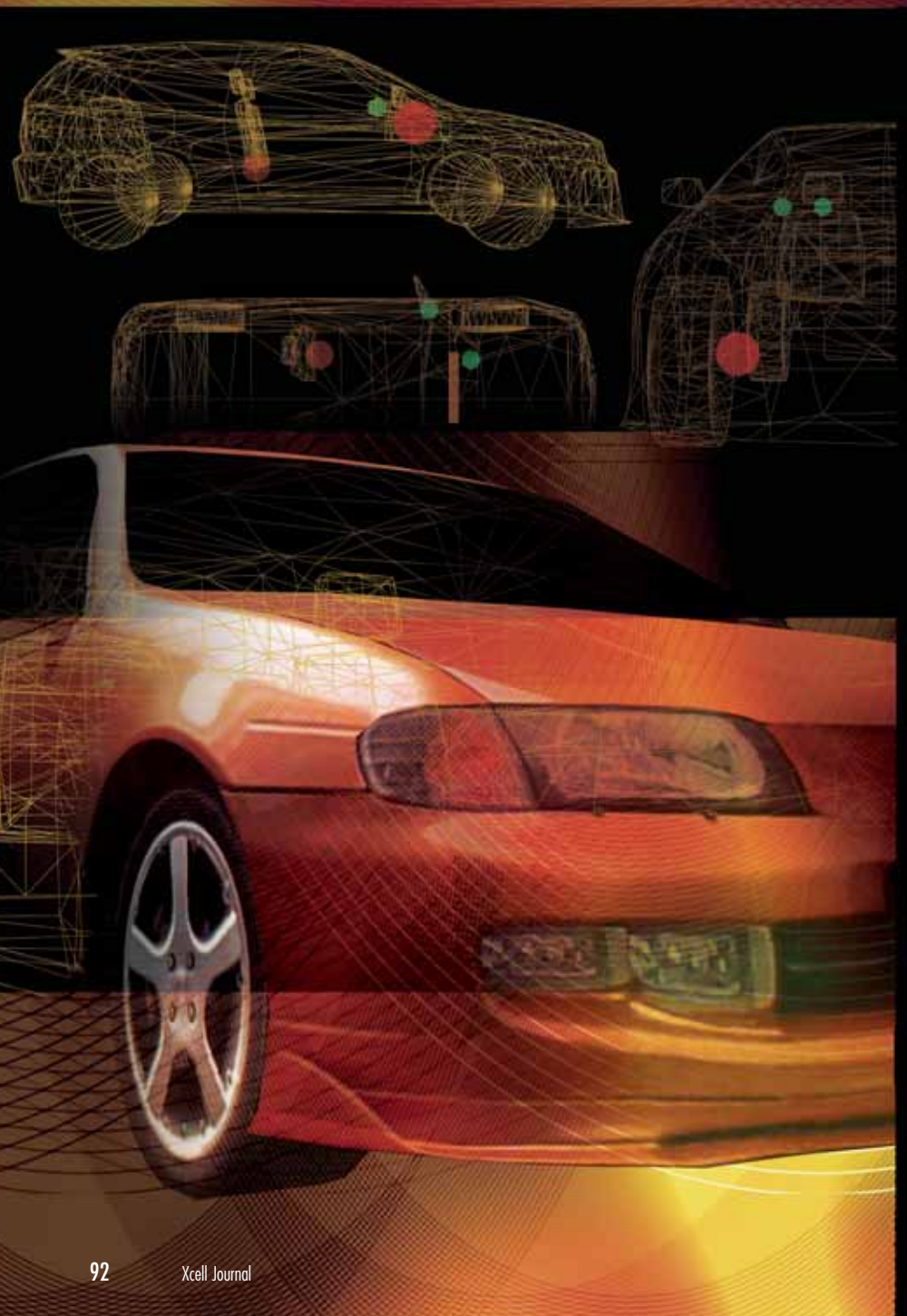
In this article, we've shown the advantages of using embedded deserializers instead of discrete components on a large project. By using nine 456-pin FPGAs to do the same job as 105 TQFPs, we saved time, both in the design and debugging phases. Plus, this is a flexible approach, as the FPGAs are reprogrammable and a more economical solution in the long term.

We are currently considering migrating to a bigger Xilinx device as our processing requirements from the FPGAs increase. Therefore, we are studying the new devices available and how such a migration will affect our PCB design in terms of the routing of the high-speed lines.

We believe that by following the design rules concerning high-speed design, like clean clock distribution, power supply filtering, and good routing of the internal reference clocks, it is possible to obtain a successful design in good time. For more information, please write to us at *jc.silva@cern.ch* or *adarsh.jain@cern.ch*.

LIN Bus – A Cost-Effective Alternative to CAN

PLDs are ideal for implementing LIN buses, offering fast time to market, flexible design options, low cost, and low power consumption.



by Karen Parnell
Automotive Product Marketing Manager
Xilinx, Inc.
Karen.parnell@xilinx.com

The automotive industry is constantly striving to reduce costs but at the same time introduce new and innovative comfort and convenience features to meet customer demand. Almost all automotive companies have adopted various busing systems to reduce wiring complexity and weight, and hence overall costs. This also results in increased fuel efficiency.

Although flexible topologies are ideal, the need exists for global standards to offer better business cases to suppliers, which would ultimately lead to greater competition and lower prices. J1850 (in the U.S.) and the ubiquitous Bosch™-defined Controller Area Network (CAN) (in Europe) are the most popular standards to date, but in some applications can be considered overkill.

In such applications you could consider using LIN as an alternative. The Local Interconnect Network (LIN) is a single-wire UART-based networking architecture originally developed for automotive sensor and actuator networking applications. The LIN master node connects the LIN network to higher-level networks like CAN, extending the benefits of networking all the way to the individual sensors and actuators.

In addition to CAN, LIN also complements Media Oriented Systems Transport (MOST) for high-speed data rates and FlexRay for safety-critical applications such as steer- and brake-by-wire. Figure 1 shows the relative cost per node and speed of various automotive networks.

Conceived in 1998, the LIN consortium comprises car manufacturers Audi™, BMW™, DaimlerChrysler™, Volvo™, and Volkswagen™. LIN is an inexpensive serial bus used for distributed body control electronic systems in vehicles. It enables effective communication for smart sensors and actuators where the bandwidth and versatility of CAN is not required. Typical applications are door control (window lift, lock, and mirror control), seats, climate regulation, lighting, and rain sensors. In these units the cost-sensitive nature of LIN enables the introduction of mechatronic elements such as smart sensors, actuators, or illumination. They can be easily connected to the car network and become accessible to all types of diagnostics and services. Outside the automotive sector, LIN is used for machine control as a sub-bus for CAN.

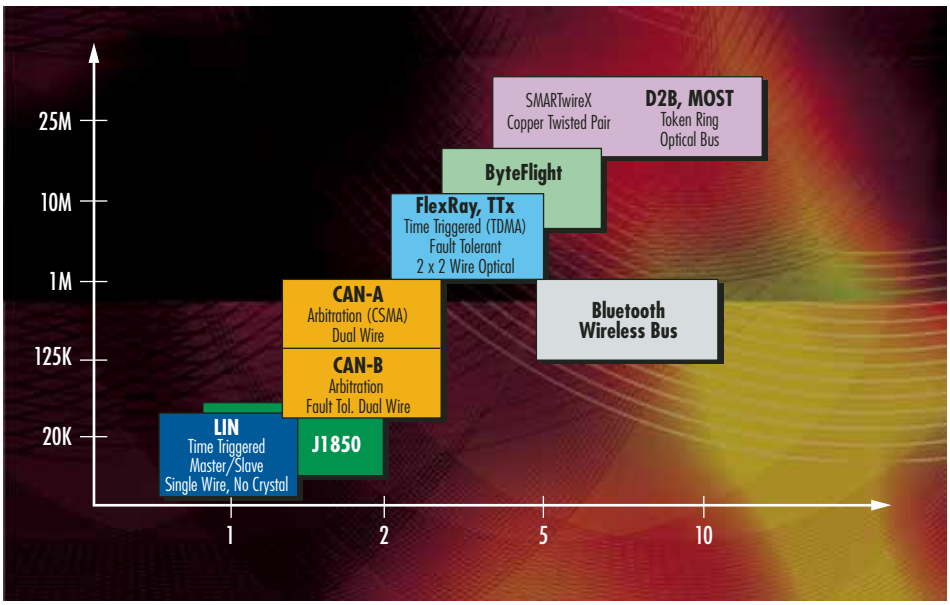


Figure 1 – Relative cost per node of automotive networks

A LIN network comprises one master node and one or more slave nodes. All nodes include a slave communication task that is split into a transmit and a receive task, while the master node includes an additional master transmit task. The communication in an active LIN network is always initiated by the master task: the master sends out a message header that comprises the synchronization break, synchronization byte, and message identifier.

Exactly one slave task is activated upon reception and filtering of the identifier, which starts the transmission of the message response. The response comprises two, four, or eight data bytes and one checksum byte. The header and the response part form one message frame.

The identifier of a message denotes the content of a message but not the destination. This communication concept enables the exchange of data in various ways: from the master node (using its slave task) to one or more slave nodes, and from one slave node to the master node and/or other slave nodes. It is possible to communicate signals directly from slave to slave without the need for routing through the master node, or broadcasting messages from the master to all nodes in a network. The sequence of message frames is controlled by the master and may form cycles including branches.

Programmable logic has long been accepted as an effective way to bring designs to market quickly and also allow design flexibility...

Flexible LIN Solution

Programmable logic has long been accepted as an effective way to bring designs to market quickly and also allow design flexibility right up to production and beyond. Historically, this time-to-market advantage and flexibility had to be balanced with higher component costs.

But times have changed. PLDs cost much less and can now be used in high-volume, cost-sensitive applications such as mobile phones, PDAs, and automotive infotainment systems. To enable designs to be brought to market quickly, some Xilinx AllianceCORE™ third-party IP providers have developed robust and fully verified IP cores aimed at FPGA and CPLD architectures. One example is their LIN core, which occupies a fraction of a low-cost FPGA (for example, 13% of a 200,000 system-gate device), thus leaving space for additional LIN nodes, CAN nodes, UARTs, soft core processors, or simply glue logic.

The LIN interface – whether implemented in programmable logic, ASIC, or ASSP – is approximately half the cost of a CAN node.

LIN Bus Benefits

The reliability of LIN is high, but it does not have to meet the same levels as CAN. A LIN bus is designed to be a logical extension to CAN. It is scalable and lowers the cost of satellite nodes. No crystal oscillator or resonator is required. It is easy to implement, has a low reaction time (100 ms max), and predictable worst-case timing.

The LIN bus can be implemented using just a single wire, while CAN needs two wires. This means that a LIN network can also be lower in cost through simpler connectors and wiring – thus also reducing the

Automotive Network	Speed	Cost Per Node	Requirements	Size in Programmable Logic
CAN	Up to 1 Mbps	$2	Crystal oscillator, two wires, 5V bus supply	348 slices (FPGA)
LIN	20 Kbps	$1	Single wire 40M line length	256 slices (FPGA) or 216 macrocells (CPLD)

Table 1 – CAN versus LIN

weight of wiring, increasing fuel efficiency, and reducing handling time and manufacturing costs. CAN also needs a 5V supply for the bus, whereas LIN only requires 2V. Table 1 shows the relative merits of LIN versus CAN.

In summary, LIN offers these benefits:

- Complementary to CAN as an ultra-low-cost sub-network
- Self-synchronization mechanism means no quartz oscillator required
- Low-cost silicon implementation using on-chip UART or SCI
- Single wire + low baud rate = reduced harness cost
- No protocol license fee

Microcontroller Implementation

There are many ways to implement LIN in semiconductors:

- Software: bit bashing
- Software: UART implementation
- Hardware: MCU with dedicated LIN port
- Hardware: PLD

Let's look at each way and explore the benefits and pitfalls of each.

Software: Bit Bashing

A LIN node can be implemented in many microcontrollers (MCUs) with no additional hardware except for a physical layer driver device. It can be implemented using existing on-chip MCU resources such as timers, GPIO, and interrupts – effectively "bit bashing."

This type of implementation does have restrictions – designers must adhere to strict real-time programming constraints to meet the full LIN specification. This is expensive with respect to MCU timing and on-chip resources and leaves very little bandwidth for other application code.

LIN nodes based purely on "bit bashing" may also be complicated to test, particularly when integrated with existing RTOSs. With this type of implementation, it would be very difficult to achieve accurate bit timing measurement and control and may not be power efficient or practical.

Software: UART Implementation

LIN was originally conceived to make use of existing UARTs within standard MCUs, along with on-chip timers, GPIO, interrupts, and serial ports. This is a better way of implementing than simply "bit bashing" but may have certain limitations in designs that already use the on-chip serial port for other tasks.

This implementation may also burden the application code with LIN protocol requirements and will complicate the design and testability of the code. This method also needs to be complemented with GPIO functionality for error checking and synchronization purposes and requires CPU activity throughout LIN message exchange. Therefore, it is not the most power-efficient solution.

Hardware: MCU with Dedicated LIN Port

An MCU with dedicated LIN port may appeal to more designers, as it uses off-the-shelf verified silicon. Thus, it will not burden the software application with LIN protocol processing, as shown in the previous examples. This type of micro is well suited for CAN-to-LIN bus bridging applications where a need exists to pass data between the two networks. This implementation also tends to be less power hungry than the equivalent software solution.

As with most emerging networks, however, the availability of silicon and relatively high cost may be an issue and create long lead times – so forward planning is a must with respect to ordering devices. One of the potential downfalls of using these devices is when more than one LIN is needed. For example, in an ECU gateway, you may need to use more than one MCU – which will impact part costs, manufacturing costs, stocking costs, and PCB complexity.

If your design requires something outside of the specification provided by the silicon vendor, this may also cause issues, as these fixed function parts allow little or no flexibility for customization. The devices still require an external bus transceiver chip and a degree of real-time processing in the MCU.

Distributed MCU solutions can also result in complex design and test issues associated with software-based designs; designers may need to explore all potential fault and interrupt loop states so that no strange indeterminate states occur. Exhaustive testing is costly, however, and the test vectors can take longer to write than the design code itself.

A LIN node can be implemented in many microcontrollers (MCUs) with no additional hardware except for a physical layer driver device.

The ability to switch between master and slave in the same device means that inventory and stocking costs are reduced …

Hardware: Programmable Logic Device (PLD)

LIN implemented in PLDs offers similar benefits to LIN implemented in an MCU-dedicated hardware peripheral. They do benefit from being implemented in generic devices that are off-the-shelf, low cost, and low power. This means that time to market is extremely quick and easy.

The LIN implemented in PLD hardware does not suffer from complicated test issues, as testing is much simpler and determinant than software-based designs. PLD LIN does not burden the software application with LIN protocol processing. It allows for accurate LIN timing control and does not require a crystal oscillator in slave mode, thus saving costs, board space, and power consumption.

PLDs are generic devices. They do not incur non-recurring engineering charges and can be used across many projects. One of the key advantages is the ability of the devices to be programmed in-system, so changing the hardware from master to slave is a breeze. As with MCU designs, the PLD needs an external transceiver device to drive the line.

The main downside to using PLDs? You may not be conversant with the design flow, so this may not be your most natural design route – but it is certainly worth trying. In more integrated higher end designs you will still need some sort of processor support, but this can be achieved by using an embedded soft-core processor such as MicroBlaze™, a low-cost 32-bit RISC processor.

LIN System Development

Automotive designers have a dilemma when adopting a new bus standard: Should they wait for standard silicon devices or try to develop an ASIC with a semiconductor supplier in advance of a final agreed and verified protocol specification? Some specifications take years to be agreed upon, verified, and ratified, so many semiconductor suppliers are loath to start designing devices before the specification is frozen.

To take advantage of new busing networks in advance of fixed specifications, designers are turning to soft IP cores embedded within programmable logic devices. This allows designers to try out new ideas risk-free and add in customized solutions within the bounds of the protocol. This approach also allows cut-down versions of the full interface if not all of the features are required – thus saving even more silicon area.

Now that programmable logic prices have dramatically dropped, they can even be considered a viable way of designing production solutions as well as prototype builds. A key benefit of having a LIN interface embedded within a PLD in the form of an IP core is that it can be reconfigured remotely to be either a master or a slave node, thus aiding greatly the test and design phases. Even in field fault diagnosis and vehicle maintenance, the ability to make nodes either master or slave may be beneficial. In the case of a non-volatile CPLD, reconfiguring the node is simply a matter of erasing the device and re-programming it with a new personality.

The ability to switch between master and slave in the same device means that inventory and stocking costs are reduced – plus there is only the need to qualify one device rather than two, thus saving the lengthy device qualification time and costs associated with it.

PLDs from Xilinx are offered in the extended temperature range of -40°C to +125°C for automotive applications. PLDs come in two main types: the larger FPGAs and simpler, low-power CPLDs.

Conclusion

The LIN bus can be used as a cost-effective alternative to CAN in low-speed automotive and industrial networks. To add even more flexibility to the network, the LIN interface can be implemented in reconfigurable logic, which is not only low power but can be reconfigured remotely to be either a master or slave in the device.

The ability to reconfigure the device to either node can help with fault diagnosis in the field, test in development, and also cut down on inventory by only stocking one device. This also reduces device qualification time and costs.

For more information, visit these websites: CAN – *www.can.bosch.com*; LIN – *www.lin-subbus.org*; LIN IP core – *www.intelliga.co.uk*; Xilinx automotive devices – *www.xilinx.com/automotive/*.

LIN IP Cores and LIN Application Note

Xilinx currently has two AllianceCore partners that offer fully verified LIN IP cores: Intelliga Integrated Design Ltd. and CAST™ Inc. Further details of these IP cores can be found at *www.xilinx.com/ipcenter/*.

You can download Xilinx application note XAPP432, "Implementing a LIN Controller on a CoolRunner-II CPLD," to use in an existing CoolRunner-II design, or simply to understand how to design your own LIN network. The application note is available at *www.xilinx.com*.

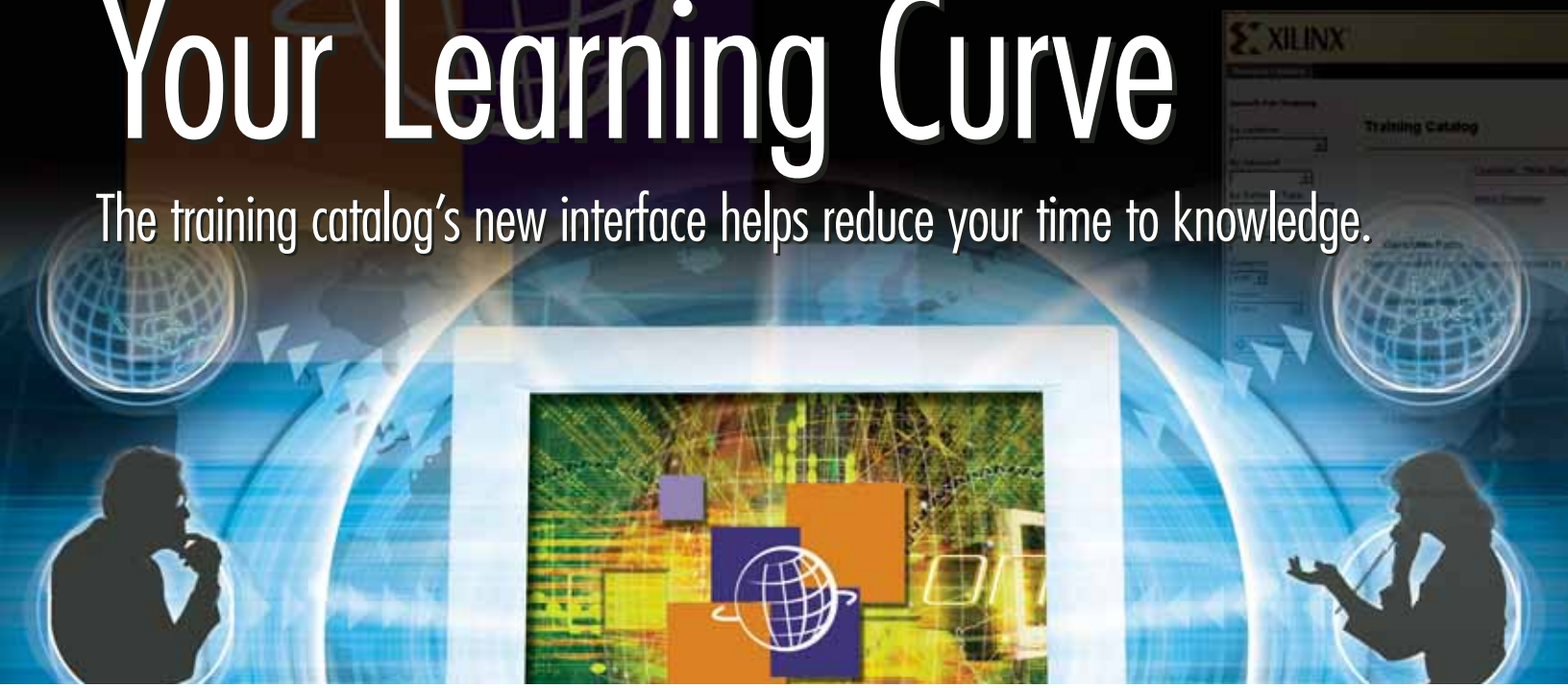
For more information, please e-mail the automotive team at *automotiveteam@xilinx.com*.

Note: The LIN IP Core from Intelliga Integrated Design Ltd. and CAST Inc. are fully supported for use in automotive designs.

The LIN implementation in XAPP432 is a reference design and should be used for evaluation purposes only.

Education Services Trims Your Learning Curve

The training catalog's new interface helps reduce your time to knowledge.



by Cindy Andruss
Instructional Designer
Xilinx, Inc.
cindy.andruss@xilinx.com

Designers have a need for speed – not only in their designs but also in their learning curves. Keeping up with the latest information about Xilinx products and services is a critical part of getting your product out of design and into the market.

"Knowing what and how you need to learn in order to increase performance is vital in today's global economy, where lack of knowledge can spoil your competitive edge," said Patrice Anderson, Xilinx® Education Services manager. "It's no secret that specialized training on Xilinx software tools can help you reduce your time to knowledge and gain an advantage over your competitors."

Education Services understands that effective training is critical to a designer's improved productivity and performance. Although our website already addressed many customer training needs, we realized that we could make improvements. Xilinx formed a team of experts to analyze the website and as a result determined that a self-service portal was the best solution. The "extreme makeover" included features to reduce your time to knowledge more than ever.

The new training catalog on the Education Services website gives you a speedy, personalized, low-cost, flexible, and quality training solution at your fingertips anytime, anywhere.*

Improved Navigation

The makeover team, which included web developers and designers, programmers, system administrators, usability experts, instructional designers, and technical writers, put their heads together to increase the usability of the aging and content-heavy Education Services website. Since its last redesign three years ago, the site was beginning to show wrinkles, and the navigation had become complicated.

The team wanted to create a quicker route to course descriptions and registration. They also wanted to augment training services and course offerings to help decrease knowledge gaps among Xilinx designers.

In addition to streamlining the navigation, the team also set out to redesign the online enrollment process, changing keywords in the search menus to provide a more intuitive registration experience. Now you can find instructor-led, live, and recorded e-Learning courses more easily by curriculum paths based on your design specialty. From course description to enrollment, fewer clicks get you where you need to go faster (see Figure 1).

"Our usability studies showed us where we could make a significant impact on our customers' training experience," said Rohan Thompson, web redesign project manager. "The subsequent overhaul to the Education Services website reflects what our customers told us they wanted."

Easy Payment Methods

The training catalog now offers more payment methods to offer flexibility and quicken the enrollment process. Simply use your credit card when registering for courses in the training catalog, or apply your company's training credits or purchase order number toward training services.

Be sure to check with your company representative to find out whether you have low-cost payment options through a Xilinx Productivity Advantage (XPA) agreement. The XPA offers all Xilinx software, education, support services, and IP cores in one package customized to meet your needs. Quite a number of training modules in the training catalog require no payment at all.

New Services

Education Services currently offers more than 14 instructor-led courses, five live online courses that together include 17 hands-on labs, and 14 recorded e-Learning modules to help you maximize your productivity and get your designs to market faster. Register for any of these learning opportunities in the training catalog or take advantage of new online services such as skills assessments, personalized learning plans, and more free recorded e-Learning modules.

You can preview classroom courses from your desktop and meet the Xilinx instructors, all experts in their fields, via the "Presenter's Bio" link in the training catalog. Offered at no charge, these online services are perfect for every budget and require absolutely no travel at all.

Philip Nowe, a hardware designer and consultant in Canada, said he is a big fan of online services, especially recorded e-Learning modules, because they maximize his time. "The modules give me the ability in one hour or less to get most of the information I need about a particular tool or software," he said. "I usually get answers to any very specific questions I might have from my local FAE, or I register for a full course on the subject."

Figure 1 - Xilinx Education Services redesigned its training catalog to reveal new online services and navigation features.

Management and Business Workflow Processes

You can maximize your productivity and time with a solution that lets you or your engineers get Xilinx training right when you need it for immediate application on the job. The training catalog is open 24/7 and allows training to be an integral part of the design process.

Education Services' recorded e-Learning fits especially well into the workflow learning model. The recorded modules are typically less than an hour in length and help to familiarize you with Xilinx technologies and short topics. In some cases, the modules offer you a preview of the content covered in multi-day, instructor-led courses.

Some recorded e-Learning modules cover topics that simply did not fit within the time constraints of an instructor-led course provided in the classroom. Aibing Zhou, a design engineer at Juniper Networks™ in Sunnyvale, California, recently took the ChipScope™ Pro free recorded e-Learning module during a training break at work.

"The ChipScope Pro recorded e-Learning was very helpful to me to catch up with the tool," Zhou said. "I was able to apply the learning immediately to gain field experience. When I have a problem, I look for e-Learning courses for more in-depth training on that tool."

Reduce Time to Knowledge with the Xilinx Training Catalog

- Search courses by curriculum paths
- Maximize training expenses
- Assess your skills and take only the training you need when you need it
- Redeem training credits purchased via the XPA program
- Learn from Xilinx-certified instructors

Empowerment

Managing your own training is like driving a car. You have the freedom to go whenever and wherever your wish. You get that kind of power everytime you visit the training catalog.

You can take control of your training with the self-directed features in the training catalog, such as self-registration; self-paced, recorded e-Learning; and self-managed training plans. Feel vested in your training by completing online course evaluations to let Xilinx know how it can continue to improve its services and products.

Nowe said that he prefers self-registering for courses online. "I like to search for courses on my own without the help of a registrar because it's faster," he said. "The training catalog also makes it easier for me to keep track of courses I've already taken and find new ones that interest me."

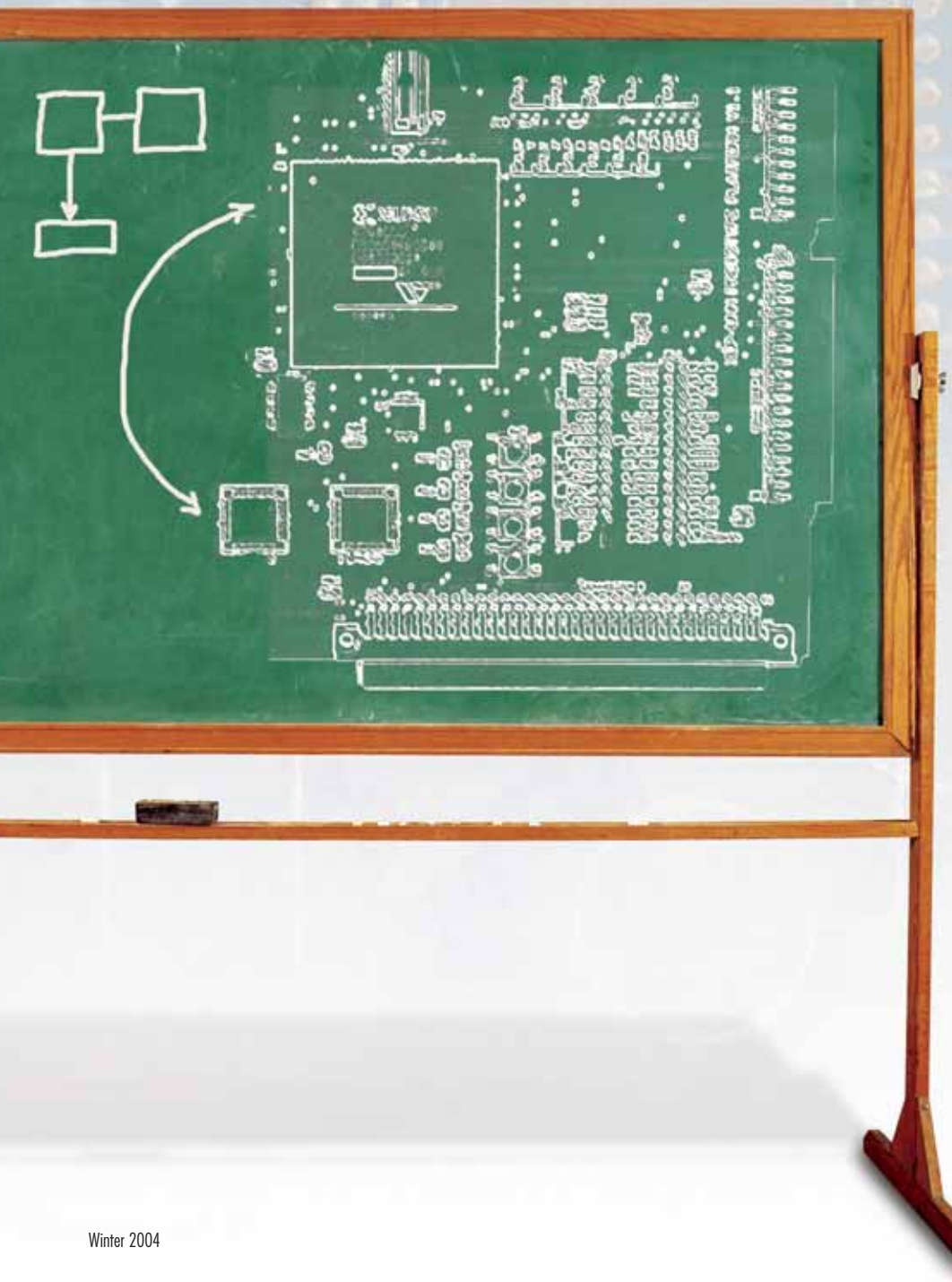
Conclusion

The training catalog is an empowering solution that meets your needs for a personalized, speedy, flexible, low-cost, and quality experience. Find out how Xilinx training can trim your learning curve. Contact Education Services at 877-XLX-CLAS or *registrar@xilinx.com*, or visit *www.xilinx.com/education/*.

** The training catalog is available to all Xilinx customers; however, registration for instructor-led courses is currently available in the North American region only.*

Board of Education

The UNM FPGA prototype project shows how the Xilinx University Program helps students learn about programmable logic.



by Craig J. Kief
Graduate Student
University of New Mexico
kiefc@ece.unm.edu

Both universities and corporations share a desire for students and engineers who can easily integrate into high-technology professions after graduation. Collaborative efforts are the key to developing the necessary skill sets.

One such collaboration is the recent Xilinx® University Program – University of New Mexico (XUP-UNM) prototype board project. When the XUP needed someone to develop a prototype board for donated Virtex™ 1000 devices, UNM jumped at the chance. The quality of this project clearly shows the high level of interaction between the local Xilinx facility, the XUP, and the university.

UNM students had to meet several key design criteria before beginning the project. The most important was that they had to design the board using donated Virtex 1000-BGA560 FPGAs. These million-system-gate devices are ideal for university projects. Their functionality and size make them suitable for a wide range of projects with the Xilinx Integrated Software Environment (ISE), System Generator, or Embedded Development Kit (EDK).

Project Requirements

As shown in Figure 1, the primary goal was a platform on which students could complete entire projects, as well as one that would allow easy interfacing to multiple external options for increased capability.

Another requirement for the prototype board was to allow a maximum number of inputs and outputs; students need to be able to get signals into and out of the board. Where possible, it was beneficial to interface with Digilent™ series circuit cards already available.

Because Digilent input/output boards are available to schools through the XUP donation program, this allows students a wide range of input and output options by switching between plug-in modules.

The board provides a multitude of output connectors:

- Standard 40-pin 0.1 inch connectors on Digilent I/O boards, which make it possible for students to obtain low-cost push pins that allow interfaces to logic analyzers and other test equipment.
- Hirose 140-pin connectors that mate to the new high-speed bus series of Digilent boards, featuring memory and analog-to-digital conversion.
- Standard 96-pin connectors that allow basic interfacing to a 6U-VME mounting platform.
- Standard 9-pin serial and 25-pin parallel connectors and their associated interfacing devices.

A nice feature is the ability to switch the 25-pin connector between enhanced parallel port and JTAG modes. By flipping a switch, students can JTAG program the board using a standard parallel printer cable. This is a very useful feature in a student environment because it allows professors to mount the boards in a stationary position; students can program the board without vertically mounting a programming cable to the header pins, reducing the possibility of the header pins breaking off.

From an academic standpoint, another great benefit is the placement of dual FPGAs on a single platform. The design team connected 100 pins from one FPGA directly into the second FPGA. The ability to easily develop projects that process data between dual FPGAs adds an entirely new level of project capabilities.

Power Supply

Because the XUP-UNM board consists of dual Virtex FPGAs, three XC18V04 in-system programmable configuration PROMs, and a wide variety of other parts, we needed a power system that could provide several amperes of filtered power over an extended period of time. The heart of the power system is a Texas Instruments™ TPS54616 buck switching power supply. This supply provides a stable 3.3 volt output to a maximum 6 amp range.

Figure 1 – XUP-UNM prototype platform

Because everything else on the board is driven by this source, 6 amps was a good supply level. Students designed a large portion of the power filtering using Xilinx application note XAPP623, "Power Distribution System (PDS) Design: Using Bypass/Decoupling Capacitors."

> "This board has great potential within an undergraduate curriculum. We are looking forward to seeing what it can do."
>
> – Colonel Bryan Goda, West Point

Both students and professors are currently evaluating prototypes of these platforms at the University of Texas (Austin), University of Texas (El Paso), and West Point. According to Colonel Bryan Goda, an academy instructor at West Point, "This board has great potential within an undergraduate curriculum. We are looking forward to seeing what it can do."

Conclusion

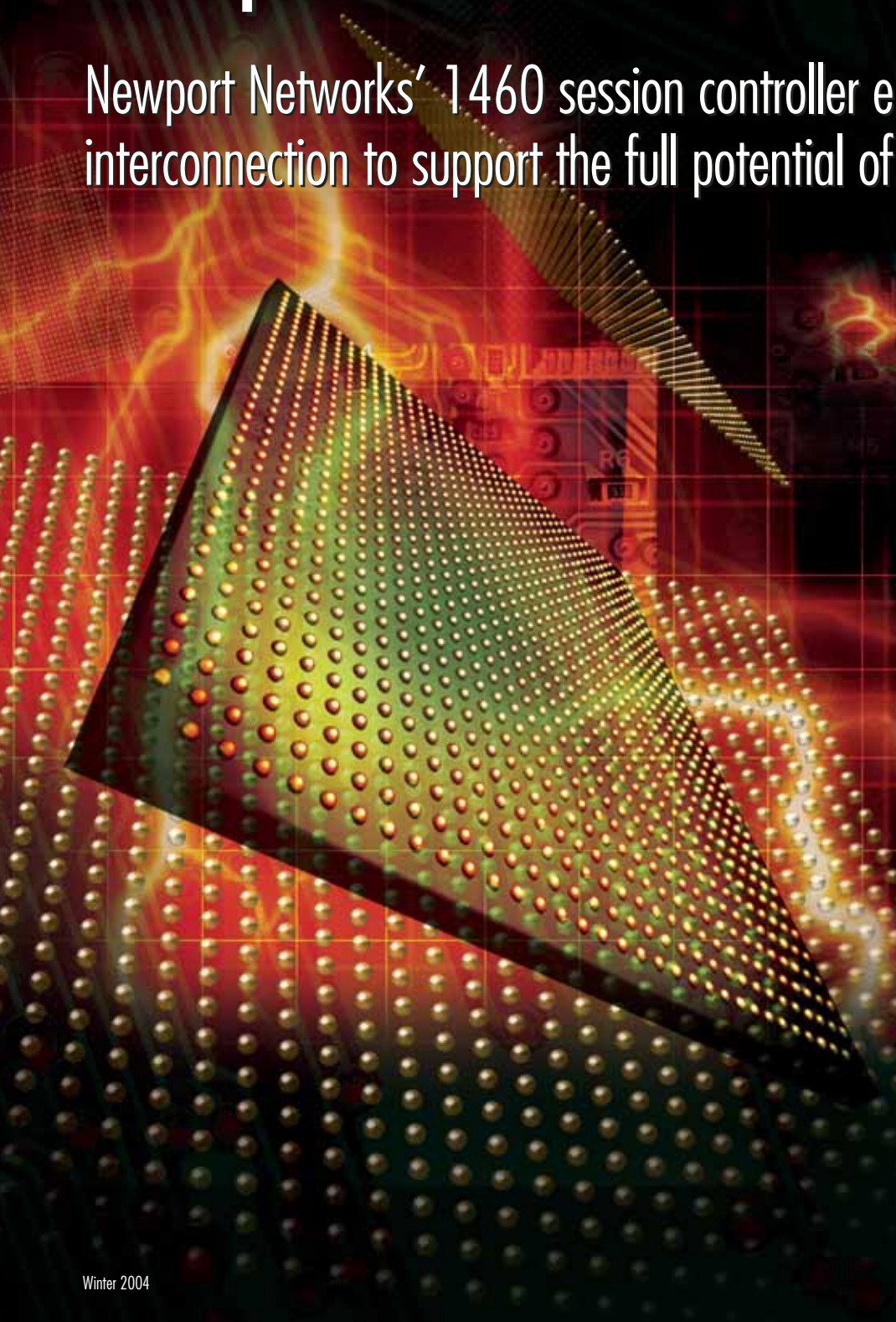
The XUP-UNM prototype platform is a tremendous example of how academia and industry can work together to accomplish a common goal. Experiences learned in these types of endeavors pay great benefits by allowing students to learn from both real-world practice and theoretical classroom experiences.

UNM has been a long-time supporter of Xilinx software, hardware, and training, and the school has developed a number of online tutorials on many topics. One series includes ISE, VHDL, Floorplanner, System Generator, and XPower (*www.eece.unm.edu/vhdl/*). Another series includes EDK and System Generator (*www.eece.unm.edu/xup/* and *www.eece.unm.edu/signals/*). And yet another indication of the interactive efforts between academia and industry are the annual professors workshops (*www.eece.unm.edu/xup/workshops.htm*).

Dr. Howard Pollard, Dr. Marios Pattichis, Alonzo Vera, and Jorge Parra of UNM were essential to the success of this project. Frank Wirtz and Reno Sanchez of Xilinx Albuquerque, Rick Ballantyne of Xilinx Canada, and Jeff Weintraub of XUP were also a tremendous help. For more information on any of these topics, e-mail Craig Kief at *kiefc@ece.unm.edu* or Alonzo Vera at *alonzo@ece.unm.edu.*

FPGAs Ensure Flexible and Adaptable IP Interconnection

Newport Networks' 1460 session controller enables direct IP-IP interconnection to support the full potential of IP services.



by David Vant
VP of Marketing
Newport Networks
david.vant@newport-networks.com

IP transport has the potential to unlock an enormous variety of communication opportunities. Voice over Internet Protocol (VoIP) is just the first in an avalanche of powerful IP-based services. These will include sophisticated messaging; storefront and customer relationship management applications; and complex and personalized services for mobile workers, home workers, and "hot deskers."

To secure the critical mass of subscribers that will allow this powerful new age to take off, IP network owners need a cost-effective and flexible interconnect that will support the full diversity of IP services both now and in the future. Carrier-class robustness is also mandatory.

Newport Networks chose the Xilinx Virtex-II™ FPGA architecture to ensure those qualities in its next-generation IP-IP interconnect solution, the Newport Networks 1460 session controller.

For scalability and robustness... Newport Networks decided to implement a significant proportion of the functionality in custom hardware.

Interconnecting IP Networks

The earliest all-IP networks relied on conventional public switched telephone network (PSTN) gateways to interconnect with other networks, even those with similar IP infrastructures. But a PSTN gateway cannot support cutting-edge IP services. Carriers will depend on these services to open new revenue streams, secure profitability, and differentiate their offerings.

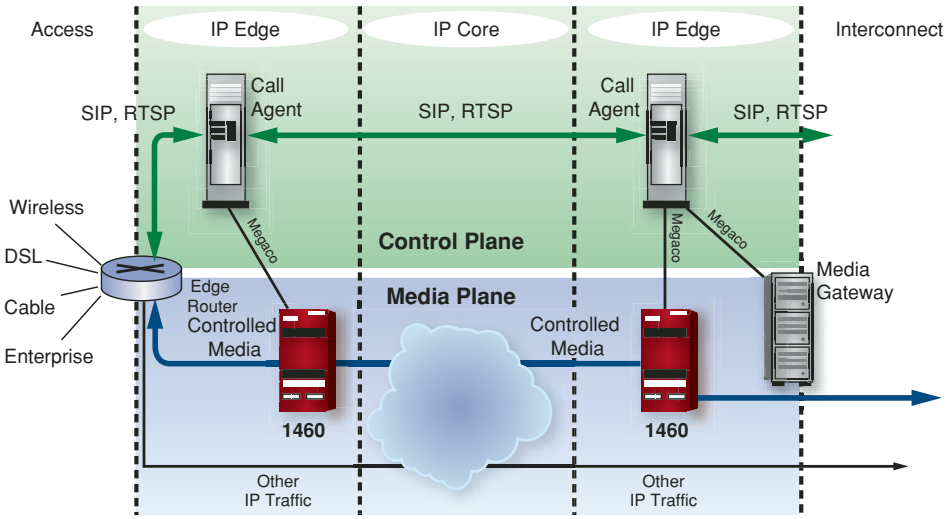


Figure 1 – The 1460 session controller supports direct IP-IP network interconnection.

PSTN gateways are also quite expensive for making IP interconnections.

The Newport Networks 1460 session controller solves this challenge. It sits at the edge of the carrier network to enable service providers to interconnect at the IP level. Figure 1 shows how the 1460 supports an IP-IP interconnect, controlling signaling and media streams as they enter and exit the network. Benefits include broadband multimedia interconnection and lower peering costs.

Any solution designed to enable direct IP-IP interconnections must be extremely flexible. This flexibility will ensure maximum interoperability between network owners while IP standards and protocols continue to change quickly. Some protocols, such as the SIP (Session Initiation Protocol) family, are now quite well defined. Others are more esoteric and continue to evolve. And as some standards achieve de facto status, each new IP service seems to precipitate a flood of competing and complementary protocols. We expect the many IP standards to consolidate in the foreseeable future. Flexibility is therefore paramount.

Scalability must also be built into the infrastructure to support the subscriber growth that IP carriers are targeting. Easy management is also a prerequisite. This includes 99.999% availability, fully resilient operation, and the ability to modify key functions remotely and apply upgrades without powering down equipment.

Further basic requirements of an IP-IP interconnect include features that preserve network security, ensure the availability of accounting information for accurate billing, and control Quality of Service (QoS) mapping and media translations.

Building the Next Generation

Choosing whether to implement the functions of a device like the 1460 session controller in software or hardware depends on the carrier's business model and future plans. For example, you can bring a software-centric solution to market quickly, which is also very flexible. But drawbacks include a relative lack of scalability, and robustness also falls as subscriber numbers increase. These interconnects are also mission critical; because session controller servers are located in-line with the call parties, a software crash or other failure can result in dropped calls.

On the other hand, a number of off-the-shelf computing platforms are capable of supporting cutting-edge services for a reasonable number of subscribers. But they lack the robustness required of a true carrier-class solution.

For scalability and robustness, therefore, Newport Networks decided to implement a significant proportion of the functionality in custom hardware. But the new gateway also had to retain that crucial flexibility to remain protocol-agile and easy to manage – key attributes in delivering a low overall cost of ownership for IP carriers.

For a network to be easily managed, operators must be able to perform routine maintenance and apply periodic upgrades without visiting on-site to change cards, introduce additional logic, or implement hardware links to support test functions. In the IP world, new services emerge and evolve quickly, calling for frequent functional upgrades.

The imminent widespread adoption of Internet Protocol version 6 (IPv6) will also

bring great implications for IP system flexibility. Adoption of IPv6 has already begun, predominantly by carriers in the Far East.

Standard Processing Hardware

Newport Networks has introduced the 1460 session controller to enable network operators to capitalize on the opportunities presented by the IP services revolution. At its heart are three distinct functional cards that perform line interfaces, application processing, and switching management functions, respectively.

Interestingly, a standard processing block is implemented on each card type. Around this hardware block, we can quickly configure a line interface card (LIC) by simply adding network processing blocks. The LIC processor performs header and packet stripping, packet analysis, traffic classification, and other processing. The session controller accommodates as many as 12 LICs, allowing easy scaling to support rapid subscriber growth. The Newport Networks 1460 is capable of supporting as many as 100,000 simultaneous, toll-quality VoIP calls.

Alternatively, by combining the processing core with switching blocks instead of network processing blocks, we can quickly configure a switching card. These are dual redundant cards that also include the switching fabric on board.

The processing hardware is implemented in an array of four PowerPC™ processors, each accompanied by high-performance FPGAs that deliver hardware acceleration and provide the flexibility to react to future changes in IP protocols, services, and business models. Hardware-accelerated functions implemented in the FPGAs include:

- Data plane integrity checking and statistical gathering
- Packet segmentation and reassembly on either side of the switch fabric
- Checksum assist
- Time-critical functions such as packet analysis are unloaded to the FPGAs. The power of this configuration means that hardware assist, such as payload string search, is also an option.

The IP protocol environment is unlikely to settle down for some time. Quite apart from competition among protocols supporting IP services that we know of today, new IP-based services are quickly emerging.

Virtex-II Benefits

When looking for a suitable FPGA to take on these intensive processing tasks for the 1460 session controller cards, Newport Networks chose the Virtex-II FPGA. Valuable features include high I/O count, greater than 100 MHz bus speed operation, plentiful on-board RAM, digitally controlled impedance (DCI), and I/O banking. In particular, the internal RAM-based FIFOs enable a convenient software interface. This allows for smooth interaction between the PowerPC and the hardware-accelerated functions executed in the FPGA.

The Virtex-II on-chip delay-locked loop (DLL) circuits also proved useful for generating low-skew internal and external clock domains. These can be referenced to an incoming clock signal such as the common switch interface (CSIX), an open standard commonly used to interface a network processor with a physical switch fabric.

The DLLs also allow output clocks to be phase-adjusted to meet setup and hold times for devices such as SDRAM. Virtex devices provide as many as eight fully digital dedicated DLLs on-chip.

Alongside the Virtex-II devices that add raw processing power, Xilinx XC9500 CPLDs perform MAC layer functions and other custom functions. These include proprietary data, control, and alarm interfaces to the backplane. The XC9500 CPLDs provide plenty of gates to implement these functions, with predictable routing and high I/O.

In the future, Newport Networks may move the 1460's computing platform into the Virtex-II Pro™ architecture, subject to CPU bandwidth requirements. Virtex-II Pro FPGAs integrate PowerPC processing blocks directly on the chip, enabling cost and real estate savings and easing manufacturing demands.

Conclusion

The IP protocol environment is unlikely to settle down for some time. Quite apart from competition among protocols supporting IP services that we know of today, new IP-based services are quickly emerging. These are supported by legions of new protocols.

While the industry works toward greater standardization among the applicable protocols, equipment providers need to deliver solutions that have the power to meet today's challenges as well as flexibility for the future. The Newport Networks 1460 session controller exploits the high-performance Virtex-II FPGA architecture to achieve each of these goals.

For more information about the Newport Networks 1460 session controller, visit *www.newportnetworks.com.*

Simplify FPGA Application Design with DIMEtalk

Remove the difficulties of system integration for single or multiple FPGA designs.

by Craig Sanderson
Systems Applications Engineer
Nallatech
c.sanderson@nallatech.com

Developing processing systems to implement high-performance applications is an extremely demanding task for engineers today. Increasingly, the demands of space, weight, and power have led designers away from traditional processor-based systems to FPGA-based solutions. This trend has led to significant advances in design flows, tools, and awareness of how to program FPGAs, which in turn has made developing the algorithmic portions of a design easier.

Designers must then begin to integrate the various elements of the overall system with one another and interface them to the outside world. In a microprocessor system this is generally simple, utilizing system-level libraries and operating system features. In an FPGA design flow it is generally much more complicated, especially if you are using more than one device.

Evidence suggests that developing this inter-process communications structure can consume as much as 80% of the development time on a typical project. This element of the design is generally not addressed by algorithmic design tools.

Having experienced first-hand how time-consuming implementing communications in FPGA applications can be, at Nallatech we looked for a way to make the process easier, developing design tools that we used internally for a few years. These early tools and principles formed the basis of DIMEtalk™, which is now available commercially.

DIMEtalk allows you to design custom inter-process communications networks within and between FPGAs at a conceptual level and automatically generate synthesizable FPGA code to represent them. This significantly reduces the time spent designing the communications element of an application, enabling you to concentrate your efforts on the parts of an application where your expertise lies, delivering solutions to customers faster.

System Communications

Developing applications to run in FPGAs has become easier, in part because of the advances made in design flows, tools, and general awareness of how to program FPGAs. Tools such as Xilinx® System Generator and other high-level implementation methodologies enable developers to quickly translate their algorithms from math-level functions into working FPGA algorithm blocks.

Once developed, connecting these algorithm blocks together is a complex and error-prone task. Even more complex is the connection of algorithms in multiple FPGA applications and the communication with external interfaces and backplanes.

This interconnectivity inside, the area outside and between FPGAs is termed "system communications" and can consume the vast majority of design time in many applications, distracting developers from their key expertise in the application being implemented. High-level algorithm design tools do not generally make provisions for implementation of system communications, so although the algorithm implementation has been made easier, system communications remains complex, error prone, and time consuming.

DIMEtalk: The Concept

Looking at the needs of FPGA application developers, we established key requirements for a system communications tool:

- Scalability, catering to designs of all sizes and designs distributed across multiple FPGAs
- Flexibility, tailored to the needs of the application
- Easy algorithm interfacing, complementing algorithm implementation
- Easy implementation, ideally through a software tool
- Resource-friendly, minimizing hardware resource requirements

Looking beyond the FPGA world, the majority of data communications take place across some form of data network. Data networks are appealing because of the flexibility and scalability they provide. When we developed the early DIMEtalk tool at Nallatech, we intended it to be primarily network-based for these reasons.

So in essence, DIMEtalk is network-based and meets identified needs by providing:

- A high-level software tool to enable users to develop communication networks
- An intelligent packet-based network – routing tables automatically defined by software
- Easy user node interfaces – block RAM, FIFO, memory map
- Automatic FPGA-synthesizable code to represent the network
- "Small footprint" network elements for efficient use of resources within Xilinx FPGAs

Physical Communications Infrastructure

We had to define the physical infrastructure of the tool – the network elements that would exist within the FPGA. We analyzed a number of data networking standards to assess their viability for use within FPGAs, but existing standards lacked the required flexibility and resource efficiency. For this reason, we developed a dedicated simplified network protocol and network infrastructure. The network elements used in DIMEtalk are as follows:

- "Routers" direct data around the network
- "Nodes" are the user interface to the network and can be connected to user application designs
- "Bridges" move data between physical devices across a defined physical media (for example, between FPGAs)
- "Edges" are used for protocol conversion to another data transfer standard (such as PCI, VME, or USB on Nallatech systems)

From a users' perspective, nodes within the network are the most important; these are the points within the network where you connect your algorithm blocks. The available interfaces are block RAM, FIFO, and memory map-based, which makes developing compatible interfaces within algorithm blocks and connecting these to the network easy.

In runtime, packet-based transfers are used across the network, enabling the transfer of data between nodes within FPGAs and also to backplane interfaces and host systems. Because of the highly optimized network protocol and low overhead implementation used in DIMEtalk, the efficiency of data transfers is as high as 97%.

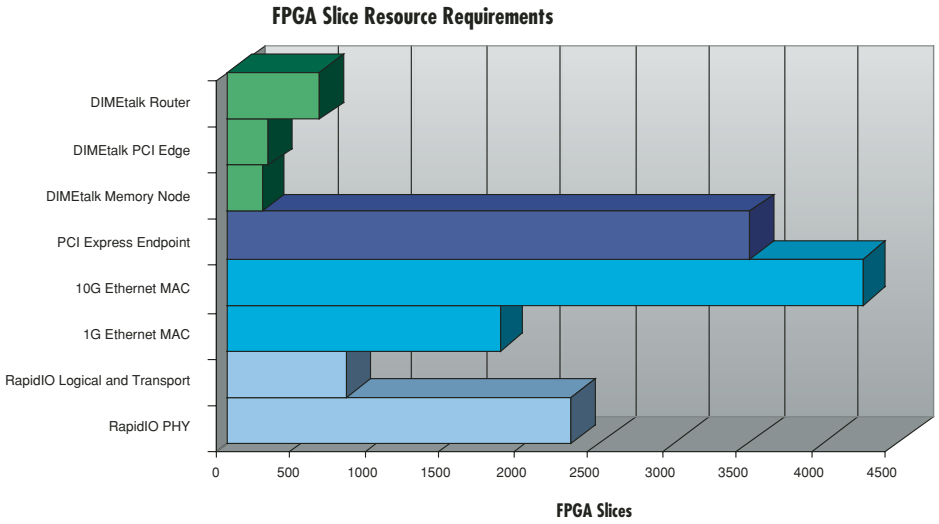


Figure 1 – DIMEtalk network element resource usage comparison

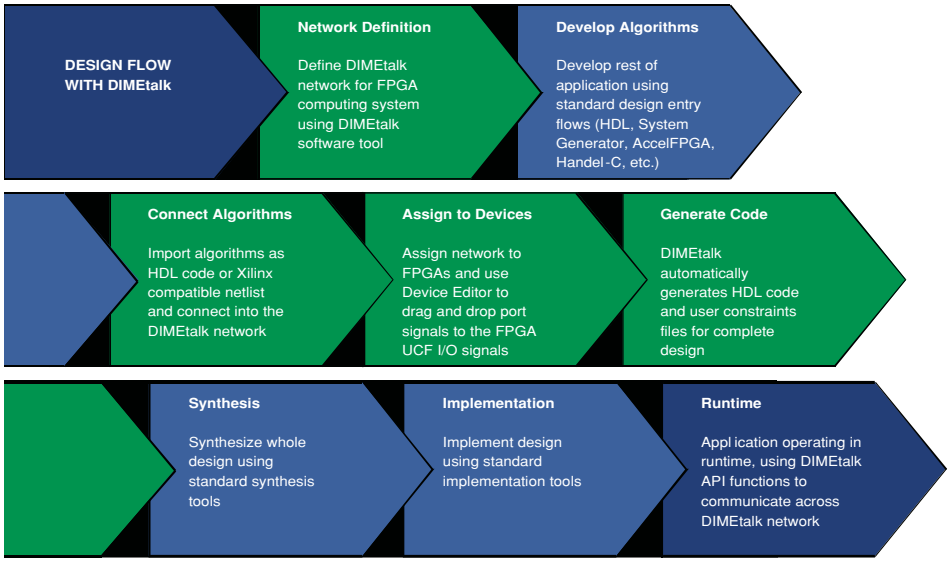


Figure 2 – DIMEtalk design flow

We are not suggesting that DIMEtalk networks should completely replace other types of data networks. However, within FPGAs and going between FPGAs on the same card, a low-resource, easy-to-implement network such as DIMEtalk makes sense – as demonstrated by the resource usage shown in Figure 1.

DIMEtalk is intended to be used alongside other data network and backplane types – that's why the edge components are so important. The edges enable you to use low-resource DIMEtalk networks where it is right to do so and interface directly to other protocols off-card.

Using DIMEtalk

DIMEtalk is designed to make life easier for developers to deploy applications on an FPGA computing platform. The intuitive design flow shown in Figure 2 enables easy network implementation and you can use the design-entry tool of your choice for algorithm blocks. The stages of the design flow are:

- Network Definition – conceptually design the network to provide communications links and interface points to algorithms as required across the FPGAs
- Develop Algorithms – use HDL or other tools to develop algorithm blocks using HDL or other design flows connected to the interface nodes of the DIMEtalk network
- Connect Algorithms – connect the completed algorithm blocks to the network; at this stage, the network and application are functionally complete
- Assign to Devices – assign the whole design to FPGAs using a drag-and-drop feature
- Code Generation – automatic code generation for design
- Synthesis – using standard synthesis tools
- Implementation – using the Xilinx ISE software tool flow

Figure 3 – Hardware for example system

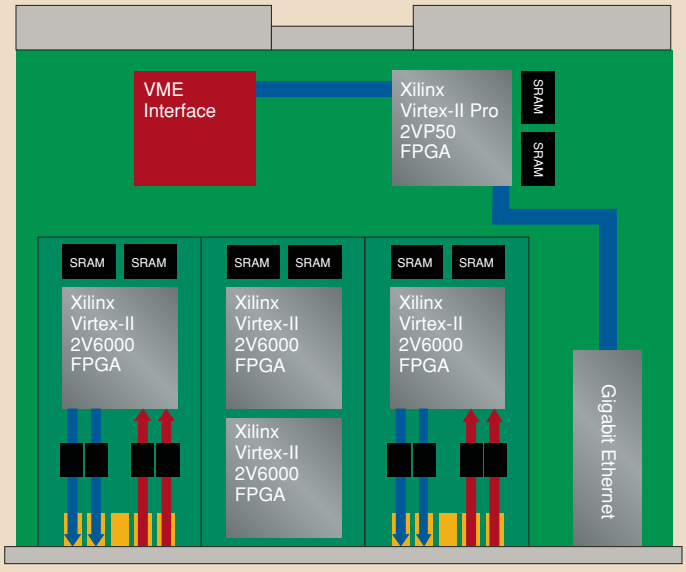


Figure 4 – Hardware block diagram for example system

Forthcoming developments in future releases of DIMEtalk will include additional interface support and links directly into algorithm development tools…

Application Architecture Example

Let's look at DIMEtalk in an application context. The easiest approach is a very high-level one to avoid getting caught up in the details of a potential system – the focus being on the overall architecture rather than low-level functionality. A typical application might include the following:

- VME form-factor
- Multiple high-density platform FPGAs
- High-speed external analog interfaces
- High-speed synchronous SRAM memory
- Gigabit Ethernet interface

This relatively complex hardware configuration is shown in Figures 3 and 4. In this case, the system comprises commercial off-the-shelf hardware products. From a functional perspective, the algorithm processing blocks that would perform the function of this application reside within the FPGAs. You can develop these algorithm blocks using the design entry flow of your choice, including HDL, commercial IP cores, and high-level languages and tools.

An example DIMEtalk network for this system is shown conceptually and in the DIMEtalk software tool in Figures 5 and 6. The network spans across all five FPGAs in the system, with router and bridge elements in place as appropriate to enable the network to operate. Each FPGA has algorithm blocks(s) associated with it – these are connected to the network at user nodes. The user node type and location can be defined to fit your application requirements.

In this example, the network is also connected through a DIMEtalk edge to the VMEbus host interface on the system – enabling direct data communications from the host to specific algorithm blocks inside the FPGAs.

What is clear from this example is the value DIMEtalk adds to the system. You can take an off-the-shelf system along with your algorithm blocks and rapidly connect all of these together and to the VMEbus.

Conclusion

Using DIMEtalk, you can efficiently implement the systems communications infrastructure required for FPGA computing applications. The generated network is flexible and provides a complete communications solution to connect together algorithm blocks, interfaces, backplane links, and host system. This type of infrastructure would have taken significantly longer to implement using traditional methods.

Forthcoming developments in future releases of DIMEtalk will include additional interface support and links directly into algorithm development tools, making application development even easier.

For further information about DIMEtalk, visit *www.nallatech.com/dimetalk/*.

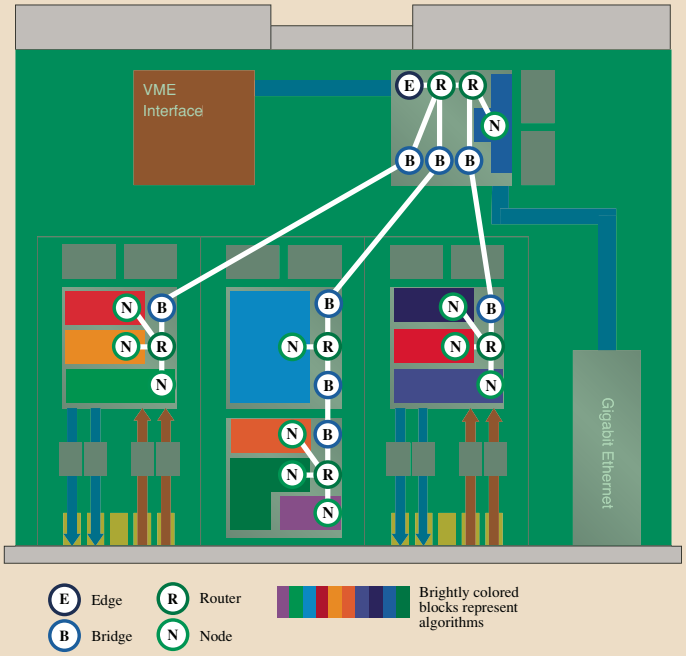


Figure 5 – DIMEtalk network shown for example system

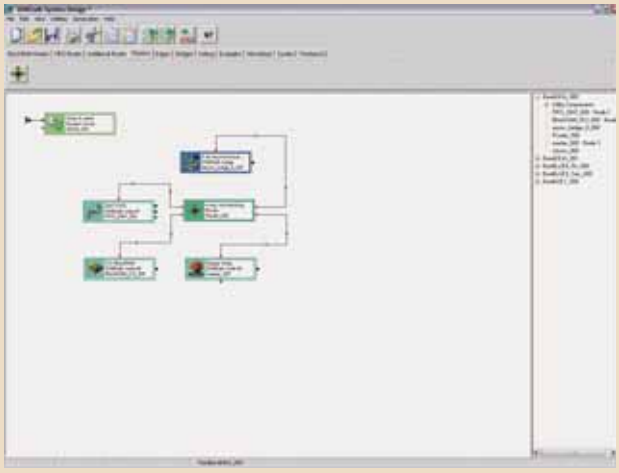
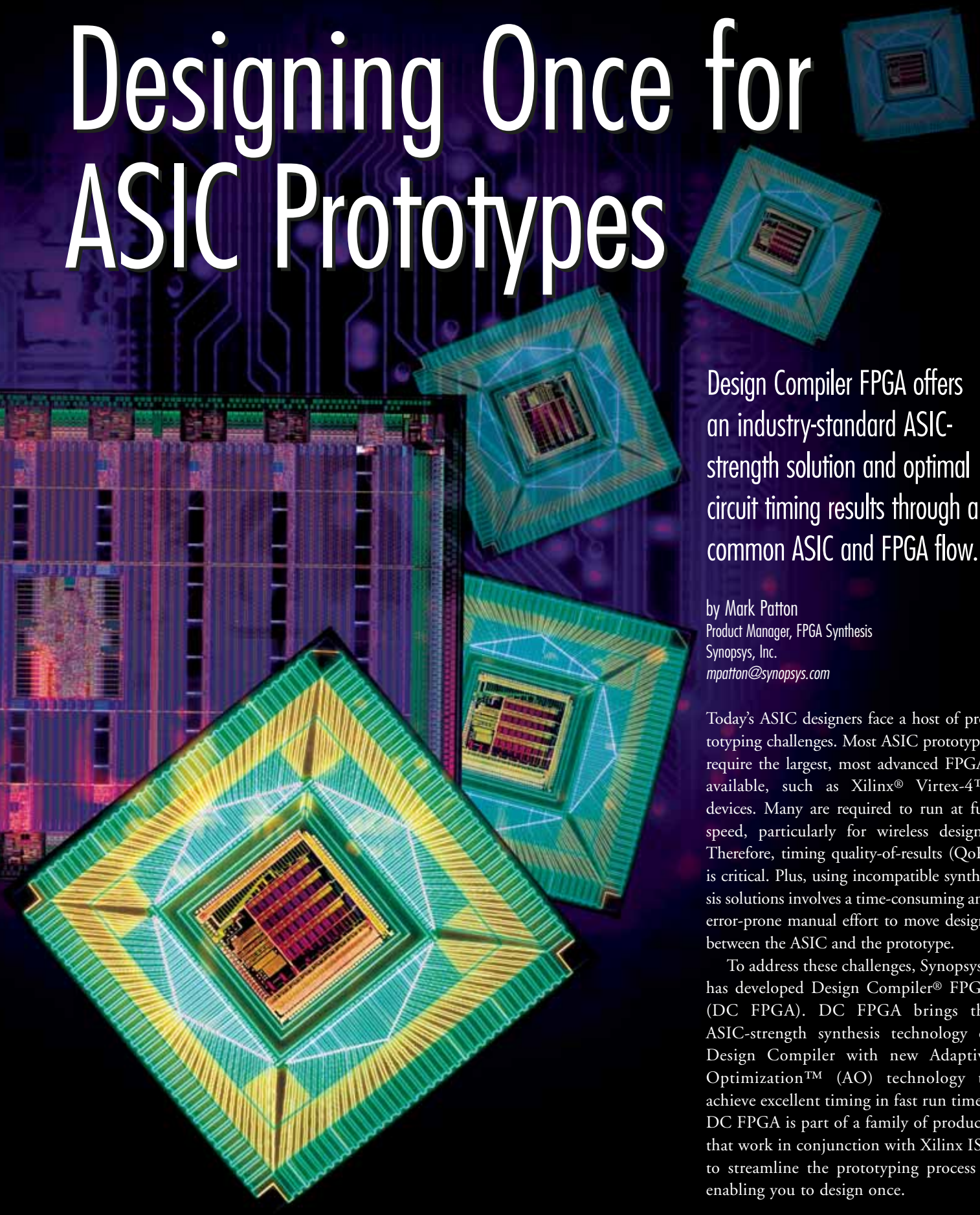


Figure 6 – Screenshot of DIMEtalk network portion for example system

Designing Once for ASIC Prototypes



Design Compiler FPGA offers an industry-standard ASIC-strength solution and optimal circuit timing results through a common ASIC and FPGA flow.

by Mark Patton
Product Manager, FPGA Synthesis
Synopsys, Inc.
mpatton@synopsys.com

Today's ASIC designers face a host of prototyping challenges. Most ASIC prototypes require the largest, most advanced FPGAs available, such as Xilinx® Virtex-4™ devices. Many are required to run at full speed, particularly for wireless designs. Therefore, timing quality-of-results (QoR) is critical. Plus, using incompatible synthesis solutions involves a time-consuming and error-prone manual effort to move designs between the ASIC and the prototype.

To address these challenges, Synopsys® has developed Design Compiler® FPGA (DC FPGA). DC FPGA brings the ASIC-strength synthesis technology of Design Compiler with new Adaptive Optimization™ (AO) technology to achieve excellent timing in fast run times. DC FPGA is part of a family of products that work in conjunction with Xilinx ISE to streamline the prototyping process – enabling you to design once.

Why Prototype?

The complexity associated with ASIC development has led to a significant increase in the number of design teams choosing to prototype their designs using an FPGA. According to Gary Smith of Gartner/Dataquest User Wants and Needs 2003, as well as our own surveys (as part of the Galaxy Technical Seminar), more than 40% of all ASIC designs have been prototyped in an FPGA. This trend is increasing over time.

Prototyping provides several benefits. Primarily, it offers a way to prove the design before undertaking an expensive ASIC manufacture. A physical prototype also enables the design to be rigorously verified using real data.

Industry analyst data gathered from the 2003 Synopsys Verification Seminar indicated that a majority (70%) of design re-spins still occur because of functional errors. Rapid verification of the programmable prototype can go a long way toward ensuring that the ASIC design is right the first time. Additionally, a prototype enables earlier integration of the complete system, providing a platform for software development that can continue in parallel with ASIC development and manufacturing.

The benefits of FPGA prototyping are clear – you will have more confidence in your design, which ultimately enables the development of a "right-first-time" ASIC in less time.

Prototyping Challenges

Ideally, the source register transfer language (RTL) for the design would be identical for both ASIC and FPGA. But in practice, you must make modifications to the RTL to get the best results from FPGA synthesis, or, in some cases, to even synthesize a design.

FPGA synthesis tools typically require you to write code in a certain style, following recommended coding guidelines, and each synthesis tool will have its own subset and variation of language support. Unless the ASIC and FPGA synthesis tools use the same compilers and directives, the RTL for the FPGA and ASIC implementations will likely be different.

Designers often use the Synopsys DesignWare® Library building blocks in the ASIC implementation of the design. Using a synthesis tool that does not support DesignWare requires you to write the specific elements yourself, which can potentially introduce errors in the design.

Meeting timing is often one of the most challenging issues in prototyping the design. Often designers are forced to use a fixed optimization strategy in traditional FPGA synthesis tools to try and meet timing. If the fixed method does not provide the required results, your only option is to make manual modifications to the RTL and try again – often with the same poor result.

These manual modifications are time consuming and error prone. They can lead to RTL "drift," where the two descriptions become so diverse that the functional equivalence is jeopardized. Even small differences, such as a single signal being tied high or low in the FPGA, can spell disaster if carried through to ASIC manufacture.

The complexity of a typical design implemented in devices such as Virtex-4 FPGAs will almost certainly demand a team effort. Many existing tools restrict the design process to a top-down flow guided by a single user. FPGA designers want the flexibility to choose the design flow – just like their ASIC counterparts.

Although none of these flow differences in isolation represent an insurmountable challenge, collectively they can add up to a major overhead in the time and effort required to develop the prototype, affecting the design integrity between the FPGA and ASIC implementations. Unless you can easily migrate your design between FPGA and ASIC implementations, the benefits of prototyping are lost.

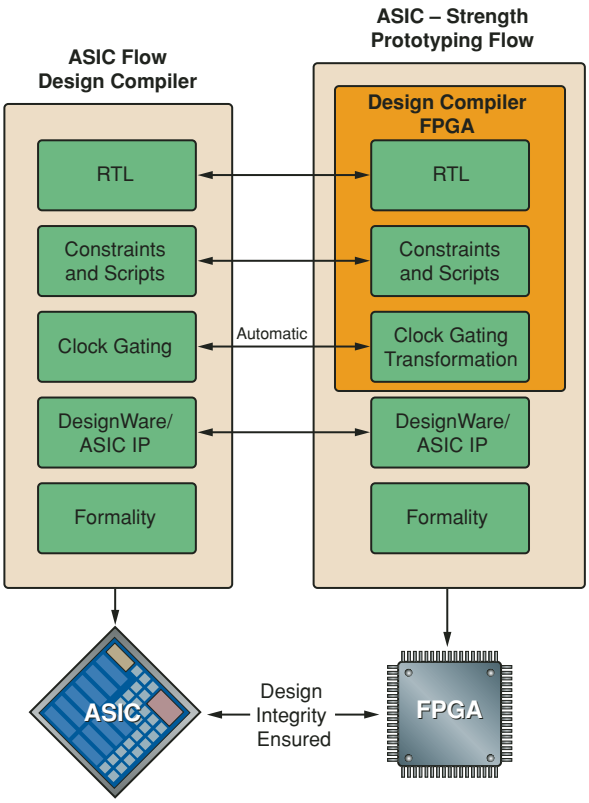


Figure 1 – Design Compiler FPGA offers a fast path to prototype.

A Unified ASIC/FPGA Design Flow

Clearly, a common design flow that supports development of both ASIC and FPGA, without the manual intervention typical in the current approach to prototyping, would provide major advantages in ensuring design integrity and minimizing development time.

Design Compiler FPGA is an FPGA synthesis product intended for design teams who prototype ASICs using high-end FPGAs. DC FPGA is built on Design Compiler's industry-leading ASIC synthesis technology and is then customized to include FPGA-specific features. DC FPGA inherits DC's reliability – proven through the development of more than 125,000 ASIC designs.

As shown in Figure 1, DC FPGA shares the same compilers, scripting language, and SDC (Synopsys Design Constraints) as Design Compiler. Because they use the same compilers, DC and DC FPGA will interpret the RTL the same way. This eliminates the manual changes in the RTL required when using different synthesis tools for FPGA and ASIC design.

Manually transforming gated clocks to the FPGA equivalent is not only very time consuming but also error prone. DC FPGA

can automatically transform gated clocks in the ASIC design to the FPGA equivalent. This capability preserves clock gating functionality while improving timing and eliminates manual design modification. These features, along with full DesignWare Library building blocks support, allow you to easily migrate designs between ASIC and FPGA implementations.

To address the sheer complexity of today's FPGA designs, DC FPGA supports top-down and bottom-up methodologies for team-based designs, enabling you to choose the appropriate methodology.

DC FPGA's new AO technology automatically selects the best synthesis algorithm for the design. The algorithms are dynamically controlled given the nature of the design and the applied constraints. AO technology will also reorder the sequence in which the synthesis algorithms are run. The result is that AO technology provides the best timing in fast run times.

For designers who want even more flexibility, DC FPGA allows you to fully control the synthesis process on a block level. This level of control is very useful, particularly when you are trying to gain the last bit of performance from the design or want to carefully control the implementation.

Formal verification is a key part of a unified design flow because it mathematically proves that the RTL matches the implementation. DC FPGA supports formal verification with our Formality® solution. Both DC FPGA and Xilinx ISE output automatic setup files for Formality, which greatly simplifies the formal verification task. The formal verification flow with Formality is shown in Figure 2.

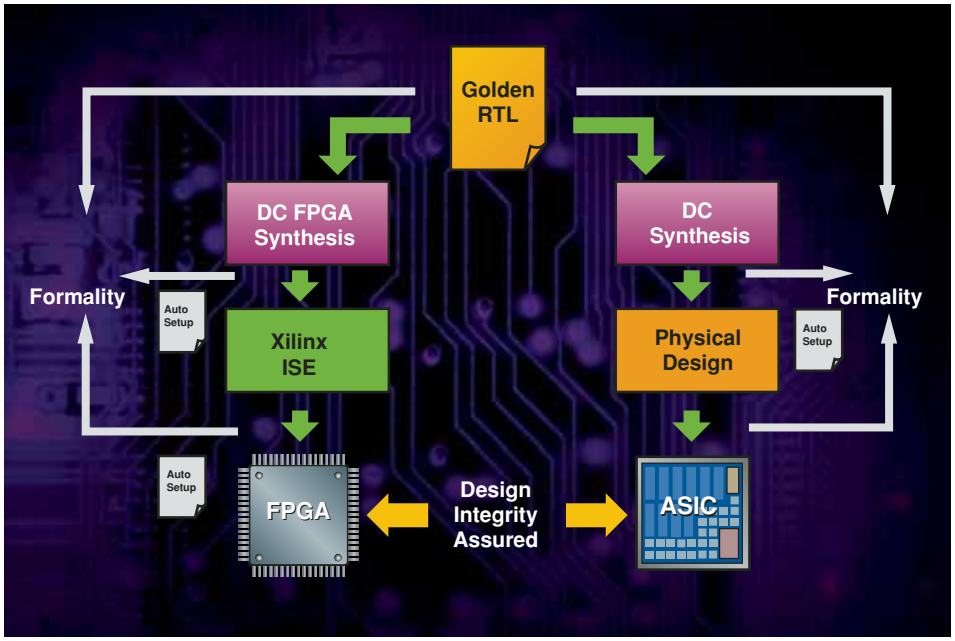


Figure 2 – ASIC and FPGA formal verification flows with Formality

Conclusion

The goal of effective prototyping is to have your design up and running at the desired speed with the least possible effort, while maintaining design integrity with the ASIC implementation.

DC FPGA will help you reach this goal. With DC FPGA, you can use common RTL for both the FPGA and the ASIC implementations to maintain design integrity, allowing you to design once. The timing performance of DC FPGA with AO technology, combined with the flexibility in synthesis, will help you meet your most difficult design challenges, getting to prototype quickly.

DC FPGA is just part of the complete ASIC-strength prototyping solution from Synopsys. Other tools supported in the Xilinx flow are Formality for formal verification, DesignWare Library IP, Leda® for RTL design and code checking, PrimeTime® for static timing analysis, VCS® for simulation, Module Compiler™ for datapath synthesis, and HSPICE® for analysis of multi-gigabit serial I/Os.

Although it is a new product, DC FPGA has a rapidly growing base of more than 80 customers. For more information about Design Compiler FPGA, visit *www.synopsys.com/products/dcfpga/dcfpga.html.*

"As a customer-centric designer and manufacturer of microprocessors, flash memory devices, and system-on-chip solutions for the computer and communications industry, AMD is pushing the speed limits of today's FPGA device technology. Using DC FPGA from Synopsys, we were able to meet the 40 MHz wireless LAN 802.11g ASIC prototyping chip performance target – a significant speed increase over what we were able to achieve with other FPGA synthesis tools. DC FPGA's compatibility with Design Compiler and the flexibility to run on a Linux™-based platform significantly accelerates our design flow process by giving us access to a common design environment for both ASIC and FPGA design."

– Dirk Haentzschel, Sr. Design Engineer, AMD Dresden Design Center

"Design Compiler FPGA impressed us because it was the only FPGA synthesis solution that had a working formal verification flow. In addition, DC FPGA was able to handle gated-clock transformations that are critical for our low-power mobile products, as well as a 23% timing improvement over our existing FPGA synthesis solution."

– Dr. Michiel Lotter, Co-Founder and VP of Engineering, Zyray Wireless

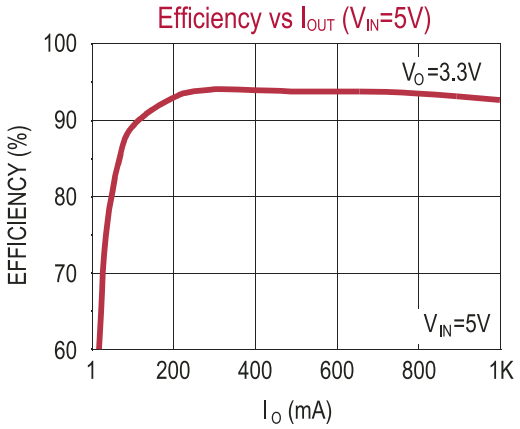
High Efficiency, 0.6A to 2A DC-DC Buck Regulators

Intersil Power Management Solutions

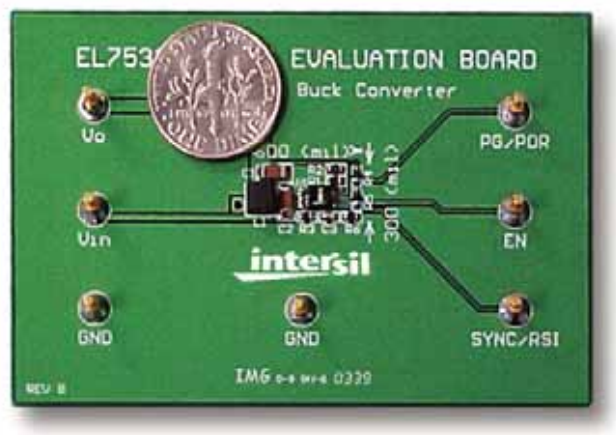
0.6A to 2A Family of Integrated FET Buck Regulators

The EL7530, EL7531, EL7532, EL7534 and EL7536 family of DC-DC buck regulators with integrated MOSFETs are simple to use, compact and full-featured. Their high efficiency makes them especially well suited for battery-operated products.

The EL7530/EL7531 devices include pulse frequency mode (PFM) and pulse width modulation (PWM) for high efficiency in standby or at full load.



EL7536 High Efficiency 1A Integrated FET Regulator



EL753X Evaluation Board with 300 mils x 600 mils Footprint

Learn more and get samples at http://www.intersil.com/EL753X

Features

- Tiny 0.97cm^2 total BOM footprint
- Extremely small 0.6A - 2A synchronous buck regulators
- V_{IN} = 2.5V - 5.5V
- 95% maximum efficiency
- 100% duty cycle (V_o close to V_{IN})
- 1.4MHz fixed PWM
  - Small passives
- PFM/PWM auto-switchable available in EL7530 and EL7531
  - 120µA quiescent current
- Power good signal for EL7530 and EL7531
- Power-on-reset for EL7532, EL7534 and EL7536
- External frequency synchronizable (EL7532, EL7534 and EL7536)
- Internal soft start

Applications

- Core Power Supply
- Communications Equipment
- Storage Systems
- WLAN
- Pocket PC
- Wireless Web Browsers
- GSP Navigators
- Digital Cameras
- Barcode Scanners
- Portable Instruments
- Language Translators

Get more technical info on Intersil's complete portfolio of High Performance Analog Solutions at

www.intersil.com/info

Amplifiers | Converters | DataComm | DCPs/DCCs | Display | Interface | **Power Management** | Switch/MUX | Timing | Video

intersil®
HIGH PERFORMANCE ANALOG

Product Selection Matrix

	Virtex-4 LX (Logic)								Virtex-4 SX (Signal Processing)			Virtex-4 FX (Embedded Processing & Serial Connectivity)					
	4VLX15	4VLX25	4VLX40	4VLX60	4VLX80	4VLX100	4VLX160	4VLX200	4VSX25	4VSX35	4VSX55	4VFX12	4VFX20	4VFX40	4VFX60	4VFX100	4VFX140
EasyPath™ Solutions	—	—	XCE4VLX40	XCE4VLX60	XCE4VLX80	XCE4VLX100	XCE4VLX160	XCE4VLX200	—	—	XCE4VSX55	—	—	XCE4VFX40	XCE4VFX60	XCE4VFX100	XCE4VFX140
Logic Cells	13,824	24,192	41,472	59,904	80,640	110,592	152,064	200,448	23,040	34,560	55,296	12,312	19,224	41,904	56,880	94,896	142,128
Total Block RAM (kbits)	864	1,296	1,728	2,880	3,600	4,320	5,184	6,048	2,304	3,456	5,760	648	1,224	2,592	4,176	6,768	9,936
Digital Clock Managers (DCM)	4	8	8	8	12	12	12	12	4	8	8	4	4	8	12	12	20
Phase-matched Clock Dividers	0	4	4	4	8	8	8	8	0	4	4	0	0	4	8	8	8
Max SelectIO™	320	448	640	640	768	960	960	960	320	448	640	320	320	448	576	768	896
Max Differential I/O Pairs	160	224	320	320	384	480	480	480	160	224	320	160	160	224	288	384	448
XtremeDSP™ Slices	32	48	64	64	80	96	96	96	128	192	512	32	32	48	128	160	192
System Monitor Blocks	0	1	1	1	1	1	1	1	0	1	1	0	0	1	1	1	1
Analog-to-Digital Converters (ADC)	0	0	0	0	1	1	1	1	0	0	0	0	0	0	1	1	1
PowerPC™ Processor Blocks	—	—	—	—	—	—	—	—	—	—	—	1	1	2	2	2	2
10/100/1000 Ethernet MAC Blocks	—	—	—	—	—	—	—	—	—	—	—	2	2	4	4	4	4
RocketIO™ Serial Transceivers	—	—	—	—	—	—	—	—	—	—	—	0	8	12	16	20	24
Configuration Memory Bits	4,875,392	8,037,312	12,647,680	18,315,520	24,101,440	31,818,624	41,863,296	50,648,448	9,651,072	14,476,608	24,088,320	5,017,088	7,641,088	15,838,464	22,262,016	35,122,240	50,900,352

Package	Area	MGT	Pins	4VLX15	4VLX25	4VLX40	4VLX60	4VLX80	4VLX100	4VLX160	4VLX200	4VSX25	4VSX35	4VSX55	4VFX12	4VFX20	4VFX40	4VFX60	4VFX100	4VFX140
SF363	17 x 17 mm	—	240	240	240										240	240				
FF668	27 x 27 mm	—	448	320	448	448	448					320	448		320	320				
FF1148	35 x 35 mm	—	768			640	640	768	768	768				640						
FF1513	40 x 40 mm	—	960						960	960	960									
FF672	27 x 27 mm	12	352													320 (8)[1]	352 (12)[1]	352 (12)[1]		
FF1152	35 x 35 mm	20	576														448 (12)[1]	576 (16)[1]	576 (20)[1]	
FF1517	40 x 40 mm	24	768																768 (20)[1]	768 (24)[1]
FF1760	42.5 x 42.5 mm	24	896																	896 (24)[1]

Pb-free solutions are available. For more information about Pb-free solutions, visit www.xilinx.com/pbfree/.

1. Number of available RocketIO Multi-Gigabit Transceivers

Important: Verify all data in this document with the device data sheets found at http://www.xilinx.com/partinfo/databook.htm

XILINX®
The Programmable Logic Company℠

Xilinx Spartan™-3 FPGAs

http://www.xilinx.com/devices/

Product Selection Matrix

	CLB Resources					Memory Resources			DSP	CLK Resources				I/O Features				Speed		PROM
SPARTAN-3	System Gates (see note 1)	CLB Array (Row x Col)	Number of Slices	Logic Cells (see note 2)	CLB Flip-Flops	Max. Distributed RAM Bits	# Block RAM	Block RAM (bits)	Dedicated Multipliers	DCM Frequency (min/max)	# DCMs	Frequency Synthesis	Phase Shift	Digitally Controlled Impedance	Number of Differential I/O Pairs	Maximum I/O	I/O Standards	Commercial Speed Grades (slowest to fastest)	Industrial Speed Grades (slowest to fastest)	Configuration Memory (Bits)
Spartan-3 Family – 1.2 Volt (see note 3)																				
XC3S50	50K	16 x 12	768	1,728	1,536	12K	4	72K	4	24/330	2	YES	YES	YES	56	124	Single-ended LVTTL, LVCMOS3.3/2.5/1.8/1.5/1.2, PCI 3.3V – 32/64-bit 33MHz, SSTL2 Class I & II, SSTL18 Class I, HSTL Class I, III, HSTL1.8 Class I, II & III, GTL, GTL+ Differential LVDS2.5, Bus LVDS2.5, Ultra LVDS2.5, LVDS_ext2.5, RSDS, LDT2.5, LVPECL	-4 -5	-4	.4M
XC3S200	200K	24 x 20	1,920	4,320	3,840	30K	12	216K	12	24/330	4	YES	YES	YES	76	173		-4 -5	-4	1.0M
XC3S400	400K	32 x 28	3,584	8,064	7,168	56K	16	288K	16	24/330	4	YES	YES	YES	116	264		-4 -5	-4	1.7M
XC3S1000	1000K	48 x 40	7,680	17,280	15,360	120K	24	432K	24	24/330	4	YES	YES	YES	175	391		-4 -5	-4	3.2M
XC3S1500	1500K	64 x 52	13,312	29,952	26,624	208K	32	576K	32	24/330	4	YES	YES	YES	221	487		-4 -5	-4	5.2M
XC3S2000	2000K	80 x 64	20,480	46,080	40,960	320K	40	720K	40	24/330	4	YES	YES	YES	270	565		-4 -5	-4	7.7M
XC3S4000	4000K	96 x 72	27,648	62,208	55,296	432K	96	1,728K	96	24/330	4	YES	YES	YES	312	712		-4 -5	-4	11.3M
XC3S5000	5000K	104 x 80	33,280	74,880	66,560	520K	104	1,872K	104	24/330	4	YES	YES	YES	344	784		-4 -5	-4	13.3M

Note: 1. System Gates include 20-30% of CLBs used as RAMs
2. For Spartan-3, a Logic Cell is defined as a 4-input LUT + flip-flop
3. Automotive Q-Grade Solutions for Spartan-3 will be available 2H2004.

Package Options and UserI/O[1]

		Spartan-3 (1.2V)								
			XC3S50	XC3S200	XC3S400	XC3S1000	XC3S1500	XC3S2000	XC3S4000	XC3S5000
Pins	Area[2]	I/O's	124	173	264	391	487	565	712	784
PQFP Packages (PQ) – wire-bond plastic QFP (0.5mm lead spacing)										
208	30.6 x 30.6 mm		124	141	141					
VQFP Packages (VQ) – very thin TQFP (0.5mm lead spacing)										
100	16.0 x 16.0 mm		63	63						
TQFP Packages (TQ) – thin QFP (0.5mm lead spacing)										
144	22.0 x 22.0 mm		97	97	97					
FGA Packages (FT) – wire-bond fine-pitch thin BGA (1.0 mm ball spacing)										
256	17 x 17 mm			173	173	173				
FGA Packages (FG) – wire-bond fine-pitch BGA (1.0 mm ball spacing)										
320	19 x 19 mm				221	221	221			
456	23 x 23 mm				264	333	333			
676	27 x 27 mm					391	487	489		
900	31 x 31 mm							565	633	633
1156	35 x 35 mm								712	784

Note 1: Numbers in table indicate maximum number of user I/Os
Note 2: Area dimensions for lead-frame products are inclusive of the leads.

Pb-free solutions are available. For more information about Pb-free solutions visit www.xilinx.com/pbfree/.

Important: Verify all data in this document with the device data sheets found at http://www.xilinx.com/partinfo/databook.htm

For the latest information and product specifications on all Xilinx products, please visit the following links:

FPGA and CPLD Devices
http://www.xilinx.com/devices/

Configuration and Storage Systems
http://www.xilinx.com/configsolns/

Packaging
http://www.xilinx.com/packaging/

Software
http://www.xilinx.com/ise/

Development Reference Boards
http://www.xilinx.com/board_search/

IP Reference
http://www.xilinx.com/ipcenter/

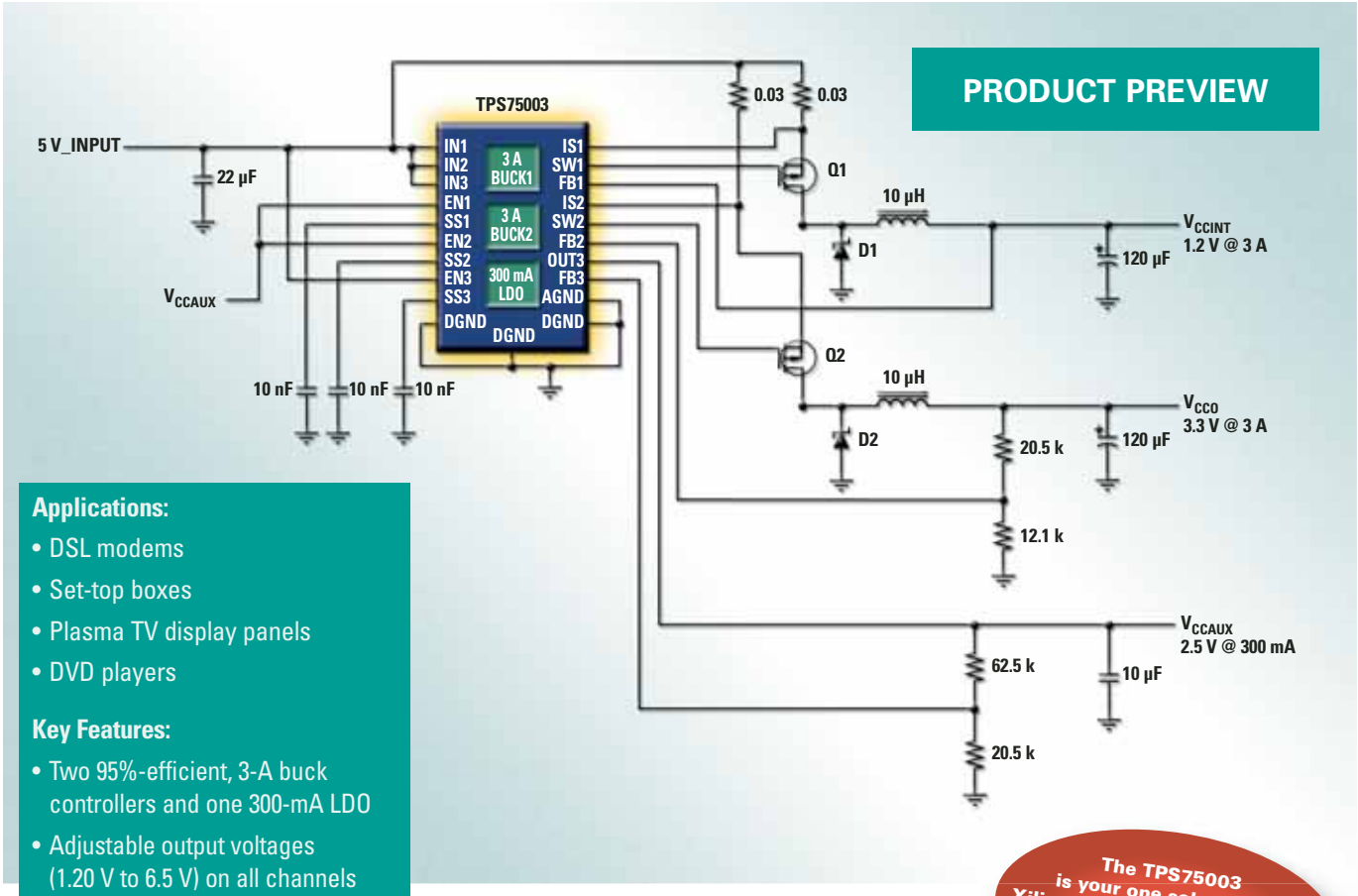
Global Services
http://www.xilinx.com/support/gsd/

POWER MANAGEMENT

TI Power Solutions: Power Behind Your Designs

Highly-integrated, Triple Supply from TI Powers Spartan™-3 Core, I/O and V_{CCAUX} Rails

PRODUCT PREVIEW



Applications:

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

Key Features:

- Two 95%-efficient, 3-A buck controllers and one 300-mA LDO
- Adjustable output voltages (1.20 V to 6.5 V) on all channels
- Input voltage range of 2.2 V to 6.5 V
- Independent soft-start for all three power supplies
- LDO stable with small ceramic output capacitor
- Independent enable for each supply for flexible sequencing
- 4.5 mm x 3.5 mm x 0.9 mm 20-pin, QFN package
- 1 ku price: $1.90

The TPS75003 is your one solution for Xilinx FPGA power management—tested, endorsed, and preferred by Xilinx.

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

For information on TI's complete line of power management solutions for Xilinx FPGAs, visit **www.ti.com/xilinxfpga**. You'll find a library of reference designs tested and endorsed by Xilinx, along with schematics and BOMs for all designs.

Questions? Need samples or an Evaluation Module? Contact us at: **fpgasupport@list.ti.com**

REAL WORLD SIGNAL PROCESSING™

TEXAS INSTRUMENTS

The Right Power Supply. The Right Partners.

Optimizing your FPGA power requirements at the beginning of your design cycle eliminates last-minute surprises and delays. Xilinx industry-leading partners–Texas Instruments, National Semiconductor, Linear Technology, and Intersil–make it fast and easy with simple, comprehensive reference guides and support. It's a powerful first step in developing more robust and reliable Xilinx FPGA designs

National Semiconductor
The Sight & Sound of Information

LINEAR TECHNOLOGY

TEXAS INSTRUMENTS

intersil
HIGH PERFORMANCE ANALOG

www.xilinx.com/powercentral

ONE FAMILY.
MULTIPLE PLATFORMS.
ENDLESS POSSIBILITIES.

LX — OPTIMIZED FOR LOGIC
SX — OPTIMIZED FOR DSP
FX — OPTIMIZED FOR PROCESSING/CONNECTIVTY

VIRTEX-4

Introducing the world's first multi-platform, domain-optimized FPGA family—delivering breakthrough capabilities and performance at every price point.

THE FREEDOM TO CHOOSE

For the first time ever, you can select from multiple FPGA platforms, optimized for application domains. You choose the exact capabilities you want. You pay only for what you need. Virtex-4 FPGAs are built upon our unique ASMBL™ (Advanced Silicon Modular Block) architecture, enabling Xilinx to *assemble* logic, memory, I/O, DSP, processors and more, giving you complete freedom of choice.

Easiest to Use Software

A SOLUTION FOR EVERY SYSTEM DESIGN CHALLENGE

The three Virtex-4 platforms—LX, SX, and FX—offer you up to 200,000 logic cells, and 500 MHz tuned performance. Our new ChipSync™ technology simplifies source-synchronous interfaces. You can implement serial protocols at any speed from 600 Mbps to 11.1 Gbps with RocketIO™ multi-gigabit transceivers. Hardware acceleration for the embedded PowerPC™ is easy with our auxiliary processing unit. And with XtremeDSP™ delivering 256 GMACS, you can solve those ultra-high performance DSP challenges.

All of your design possibilities just became realities. See for yourself at ***www.xilinx.com/virtex4***.

XILINX®
The Programmable Logic Company™
www.xilinx.com

LOWEST SYSTEM COST · HIGHEST SYSTEM PERFORMANCE

PN 0010828