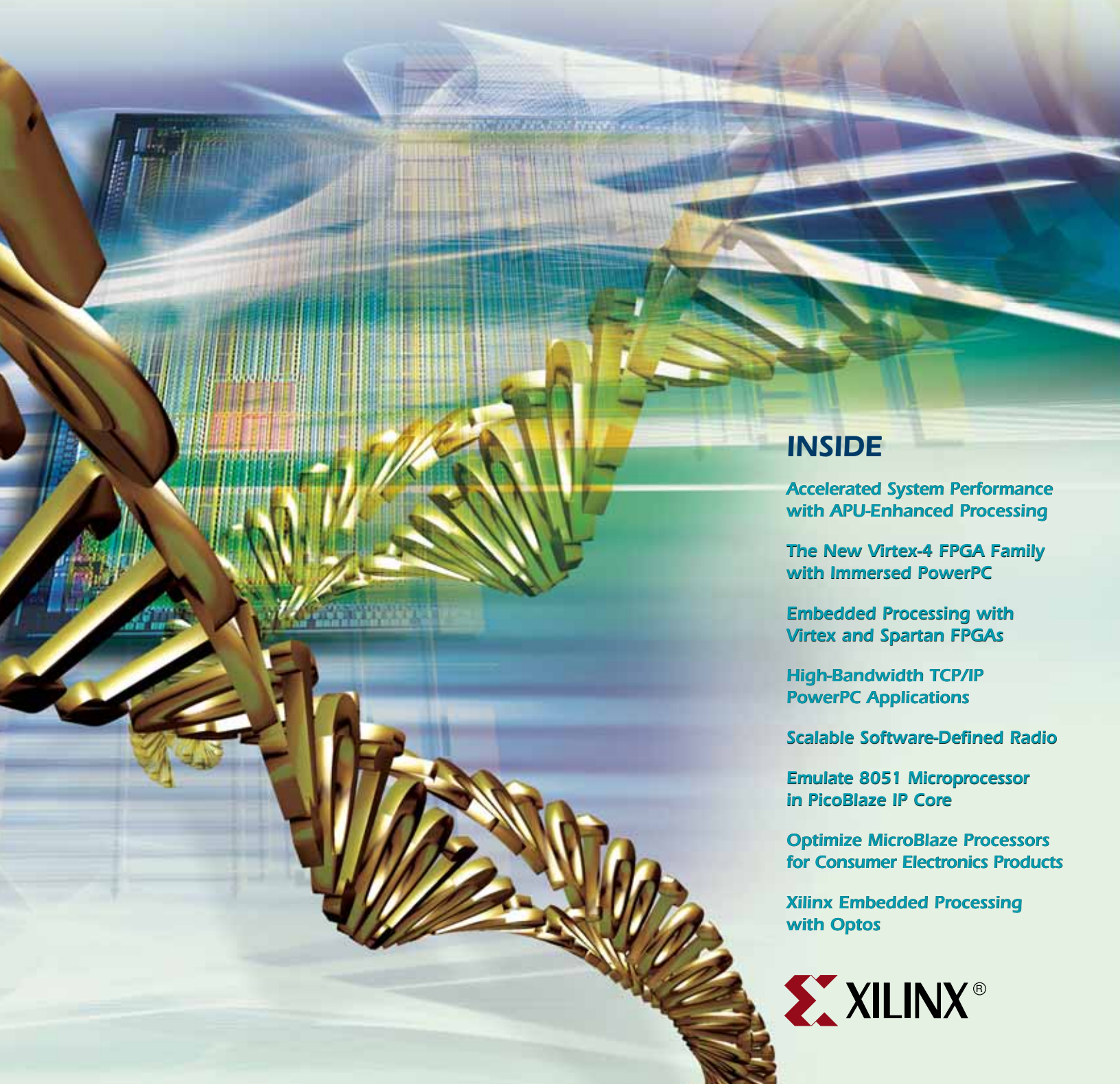
Issue 1
March 2005

Embedded magazine

EMBEDDED SOLUTIONS FOR PROGRAMMABLE PROCESSING DESIGNS



INSIDE

XILINX®

FREE
Online Design with
Spartan-3 2000 FPGAs
for a limited time only

What would you do with *advanced access* to new technology?

SPARTAN-3

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 first 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. We've developed a number of labs that allow you to evaluate the Xilinx XC3S2000 FPGA in different environments. Our labs include: Inverse Discrete Cosine Transform Hardware Acceleration, PWM Controller, and Flash Programming via Serial Interface with Boot Loader.

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

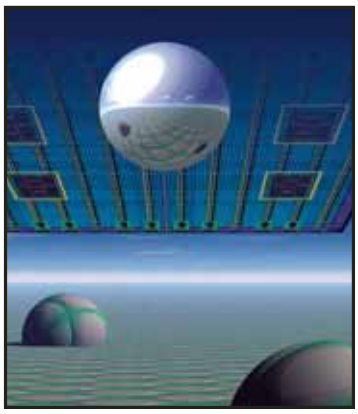
NU HORIZONS
NU HORIZONS ELECTRONICS CORP

To register for your FREE online evaluation, please visit:
www.nuhorizons.com/VirtuaLab

www.techonline.com
TechOnLine

XILINX®
The Programmable Logic Company℠

CONTENTS



Embedded magazine

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

www.xilinx.com/xcell/embedded

Xcell Publications

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

Welcome to the inaugural edition of the new Xilinx *Embedded Magazine* and the Embedded Systems Conference 2005.

In this, our first edition of *Embedded Magazine*, we have assembled a host of articles representing a wide range of embedded processing 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 for FPGA technologies.

Included in this magazine and at the Embedded Systems Conference 2005 are examples and demonstrations of 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.

Last year, we made significant strides in the embedded processing arena. In July, we announced the formation of the Embedded Processing Division, reinforcing our commitment to the increasingly diverse and evolving embedded systems market. This division brings talent and technology together in an organization that intends 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.

In September, Xilinx launched our breakthrough Virtex™-4 family of products, featuring embedded processing solutions with PicoBlaze™ and MicroBlaze™ soft-processor cores and PowerPC™ (available on all Virtex-4 FX devices). The Virtex-4 FX device includes the new PowerPC Auxiliary Processor Unit (APU) controller to easily connect the CPU to the FPGA fabric, enabling the implementation of acceleration hardware for virtually any application. Once only the domain of high-budget ASIC and ASSP design teams, the Virtex-4 FPGA's architectural ability to combine application-specific hardware acceleration with the high-performance PowerPC and MicroBlaze processor shatters the traditional barriers of cost, time to market, and risk.

In February 2005, our Xilinx Platform Studio (XPS) embedded tool suite received top honors at the inaugural International Engineering Consortium (IEC) DesignVision Awards. Judged by independent industry experts, XPS was selected in the FPGA/PLD Design Tools category for its innovation, uniqueness, market impact, customer benefits, and value to society.

We have an exciting lineup of events at the conference, showcasing both Xilinx and our partners' latest solutions. The Xilinx booth (#1525) will feature our award-winning Xilinx Platform Studio, as well as the latest Virtex-4 FX solution; our flexible PicoBlaze and MicroBlaze soft processors with Spartan™-3 devices; and our high-performance DSP solutions.

I hope you find the conference and our first edition of *Embedded Magazine* informative and inspiring. We invite you to unlock the power of Xilinx programmability. The advantages will change the way embedded systems are designed.

Mark Aaldering

Vice President
Embedded Processing
& IP Divisions

Theatre Presentations in Xilinx Booth #1525

A Complete Spectrum of Programmable Processing Solutions
Tuesday, March 8: 1:30 p.m., 4:30 p.m., 7 p.m.
Wednesday, March 9: 12 p.m., 3 p.m., 5:30 p.m.
Thursday, March 10: 10:30 a.m., 1:30 p.m.

Boot a Live Embedded System in Just 15 Minutes
Tuesday, March 8: 2 p.m., 5 p.m., 7:30 p.m.
Wednesday, March 9: 12:30 p.m., 3:30 p.m., 6 p.m.
Thursday, March 10: 11 a.m.

Breakthrough Performance with Virtex-4
Tuesday, March 8: 2:30 p.m., 5:30 p.m.
Wednesday, March 9: 10:30 a.m., 1 p.m., 4 p.m., 6:30 p.m.
Thursday, March 10: 12 p.m.

Eclipse-Based HW/SW Debug
Tuesday, March 8: 3 p.m., 6 p.m.
Wednesday, March 9: 11 a.m., 1:30 p.m., 4:30 p.m.
Thursday, March 10: 9:30 a.m., 12:30 p.m.

Breakthrough DSP Performance at the Lowest Cost
Tuesday, March 8: 3:30 p.m., 6:30 p.m.
Wednesday, March 9: 11:30 a.m., 2 p.m., 5 p.m.
Thursday, March 10: 10 a.m., 1 p.m.

Xilinx Distributor Presentations
Tuesday, March 8: 4 p.m. – Presenter: Insight/Memec
Wednesday, March 9: 2:30 p.m. – Presenter: Avnet
Thursday, March 10: 11:30 a.m. – Presenter: Nu Horizons

Product Demonstrations in Xilinx Booth #1525

Intelligent Tools Accelerate Development
We will use the Xilinx Platform Studio tool suite to create an embedded hardware/software/IP system and boot a real-time operating system. We will show how Platform Studio accelerates embedded processing development for programmable systems through the IDE, as well as powerful design wizards, BSP, and application code generators.

Low Cost and Flexible Processing
Come see the new Spartan-3 SP305 board implementing complex embedded motor control using the MicroBlaze soft 32-bit processor, a floating point unit, PID control loops, PWM, DAC, and ADC to control both a brushless DC and a three-phase AC induction motor. We will also show a frequency generator using the PicoBlaze 8-bit processor, capable of generating a frequency range up to 640 MHz incremented at 1 Hz.

High-Performance Processing
Using a Xilinx ML310 Embedded Development Platform board with multi-OS boot (Linux, VxWorks, and QNX), along with the Xilinx Platform Studio (XPS) tool suite, we will implement a dual-processor system with inter-processor communication. The first processor will be used as a compute engine utilizing the FPU. The second processor will be used as a host processor managing data flow and I/O.

Accelerate Programmable Embedded Platform Performance
Accelerate system performance with the Virtex-4 FX family of devices using the Auxiliary Processor Unit (APU) controller. We will introduce the Xilinx ML403 FX Embedded Development Platform board and demonstrate algorithm code acceleration with APU implementation and graphical display of output. We will utilize the Gigabit System Reference Design (GSRD) with TCP/IP acceleration and Linux eOS.

High-Performance DSP Implementation and Embedded Development Kit Integration Using System Generator
This demo shows how you can use System Generator for DSP 6.3 to build DSP microprocessor-based control circuits for high-performance DSP systems. System Generator for DSP and Xilinx DSP IP cores provide unrivaled DSP design solutions for FPGA-based DSP systems.

Hands-On Workshops

Implementing a Virtex-4 FX PowerPC System with Floating-Point Co-Processor Using Platform Studio
Instructor: Glenn Steiner

Following a brief presentation, this workshop will provide hands-on experience implementing a PowerPC processor system demonstrating code acceleration through an FPU co-processor. You will create the design utilizing the award-winning Platform Studio tool suite and target actual hardware. The application will demonstrate code acceleration via the FPU with graphical display of the output.

Location: Room 256
Tuesday, March 8:
10:00 a.m.–11:30 a.m.
2:00 p.m.–3:30 p.m.
4:00 p.m.–5:30 p.m.

Accelerating an IDCT Software Application Using the Configurable MicroBlaze 32-bit RISC Processor
Instructor: Derek Palmer

Utilizing the new Fast Simplex Link (FSL) port on the Xilinx MicroBlaze soft-core processor, we can add dedicated low-latency co-processing engines tuned specifically for any application at hand. We will demonstrate a 10X performance improvement of an inverse discrete cosine transform (IDCT) algorithm, utilizing a Spartan-3 platform for hands-on experience.

Location: Room 256
Wednesday, March 9:
10:00 a.m.–11:30 a.m.
2:00 p.m.–3:30 p.m.
4:00 p.m.–5:30 p.m.

High-Performance DSP on an FPGA Made Easy
Instructor: Sabine Lam

In this DSP workshop, designers will learn how to implement high-performance DSP designs using the new Xilinx Virtex-4 FPGAs. By leveraging tools such as Xilinx System Generator for DSP, IP cores, and hardware, even designers new to FPGAs will realize the capability and ease-of-use of Xilinx DSP design solutions.

Location: Room 256
Thursday, March 10:
9:00 a.m.–10:30 a.m.
11:00 a.m.–12:30 p.m.
1:00 p.m.–2:30 p.m.

Technical Conference Papers by Xilinx

Virtex-4 FX Platform FPGA Extends PowerPC Instructions
Microprocessor Summit
Session MPS-900
Argent Hotel, San Francisco
Monday, March 7: 9:00 a.m.–10:30 a.m.
Presenter: Dan Isaacs, Xilinx, APD Marketing

Approaches in FPGA Design
Easy Paths to Silicon Design Seminar: EPD-664
Tuesday, March 8: 4:00 p.m.–5:30 p.m.
Presenter: Tim Tripp, Xilinx,
Advanced System Tools Product Marketing

Switching Fabrics at Layers 1 and 2
Network Systems Design Seminar: NSD-707
Wednesday, March 9: 8:30 a.m.–10:00 a.m.
Presenter: Hamish Fallside, Xilinx, Sr. Manager,
Networking Systems Solutions

Hardware/Software Codesign for Platform FPGAs
Embedded Training Program: ETP-423
Thursday, March 10: 11:15 a.m.–12:45 p.m.
Presenter: Don Davis, Xilinx, Sr. Manager, High-Level Tools

Software-Defined Radio with Reconfigurable Hardware and Software
Embedded Training Program: ETP-442
Thursday, March 10: 2:00 p.m.–3:30 p.m.
Presenter: Peter Ryser, Xilinx,
Embedded SW Systems Engineer Manager

Other Xilinx-Related Technical Papers

Designing with Embedded Processor Cores in FPGAs
Easy Paths to Silicon Design Seminar: EPD-524
Monday, March 7: 11:00 a.m.–12:30 p.m.
Presenter: Avnet

Designing with IP in FPGAs
Easy Paths to Silicon Design Seminar: EPD-544
Monday, March 7: 1:30 p.m.–3:00 p.m.
Presenter: Avnet

Implementing DSPs in FPGAs
Easy Paths to Silicon Design Seminar: EPD-604
Tuesday, March 8: 8:30 a.m.–10:00 a.m.
Presenter: Avnet

Designing with Soft-Processor Cores in FPGAs, Part 1
Embedded Training Program: ETP-309
Wednesday, March 9: 8:30 a.m.–10:00 a.m.
Presenter: Avnet

Designing with Soft-Processor Cores in FPGAs, Part 2
Embedded Training Program: ETP-329
Wednesday, March 9: 11:00 a.m.–12:30 p.m.
Presenter: Avnet

FPGA Design for Firmware Engineers, Part 1
Embedded Training Program: ETP-343
Wednesday, March 9: 2:00 p.m.–3:30 p.m.
Presenter: Avnet

FPGA Design for Firmware Engineers, Part 2
Embedded Training Program: ETP-363
Wednesday, March 9: 3:45 p.m.–5:15 p.m.
Presenter: Avnet

FPGA Embedded Processors: Revealing True System Performance
Embedded Training Program: ETP-367
Wednesday, March 9: 3:45 p.m.–5:15 p.m.
Presenter: Memec

Introducing the New Virtex-4 FPGA Family

The latest FPGAs from Xilinx set new records in capacity, capability, performance, power efficiency, and value.

by Erich Goetting
Vice President &
General Manager,
Advanced Products Division
Xilinx, Inc.
erich.goetting@xilinx.com

Welcome to the Xilinx® Virtex-4™ edition of the *Xcell Journal*. We've created this special issue to show you the new Virtex-4 FPGA family, and how its innovations enable the creation of next-generation systems that do more than ever thought possible only a few years ago.

In this article, I'll take you behind the scenes for a guided tour of some of the new technologies, as well as a bit of the inspiration and rationale behind them.

With more than 100 innovations, the Virtex-4 family represents a new milestone in the evolution of FPGA technology. After conducting extensive interviews with leading design engineers worldwide, we knew that they wanted the following things in an advanced next-generation FPGA family:

- Higher performance
- Higher logic density
- Lower power
- Lower cost
- More advanced capabilities

At the heart of the Virtex-4 FPGA is our next-generation 90 nm triple-oxide 10-layer copper CMOS process technology.

It's relatively easy to deliver on one or two of these items – our challenge was to deliver all of them at the same time. We did this through a combination of innovative process and circuit design, process development, the ASMBL architectural approach, and the use of advanced embedded functions.

Development work on the Virtex-4 family (code-named "Whitney" after the highest mountain in the continental United States) began more than two years ago. It represents the creativity and dedication of hundreds of engineers, spanning integrated circuit design and layout, software and IP development, process development, testing and characterization, systems and applications engineering, technical documentation, and product marketing.

One of the most remarkable developments embodied in the new Virtex-4 FPGA family is the ASMBL architecture, which represents a fundamentally new way of constructing the FPGA floor plan and its interconnect to the package. First of all, ASMBL enables I/O pins, clock pins, and power and ground pins to be located anywhere on the silicon chip, not just along the periphery as with previous approaches. This in turn allows power and ground pins to be brought directly into the center of the silicon die, thereby significantly reducing on-chip IR drops that can occur with the largest FPGAs running at the highest frequencies.

Clock input pins are also located in the center of the die, which reduces clock latency. This is because clock networks need to have equal delay to all endpoints (that is, minimum skew), and thus the clock must emanate from the center. In periphery-connected clock input pins, the signal first traverses from the edge of the die to the center, and is then distributed to all regions. The Virtex-4 ASMBL design eliminates this traversal completely, and thus directly reduces the clock network propagation delay.

In addition to its electrical advantages, ASMBL provides another significant benefit in that it allows a more flexible – and thus more precise – allocation of on-chip resources. That in turn has enabled us to offer Virtex-4 devices in three unique platforms, each with a different mix of on-chip resources:

- The LX platform, optimized for logic applications
- The SX platform, optimized for high-end DSP applications
- The FX platform, optimized for embedded processing and high-speed serial applications

A Look Inside the Virtex-4 FPGA

At the heart of the Virtex-4 FPGA is our next-generation 90 nm triple-oxide 10-layer copper CMOS process technology. While that's quite a lot of adjectives, every one of them is incredibly important. The first, 90 nm, refers to the "drawn" gate length of the smallest transistors. As transistors get smaller, they get faster, use less dynamic power, and enable higher complexity at lower price points. Chip designers think in terms of "transistor budgets," which are now in the billion transistor range.

Triple-Oxide 90 nm CMOS Technology

Triple-oxide technology refers to the number of transistor oxide thicknesses available in the process. More oxide thicknesses allow more tuning of performance and power in the device circuitry, and enable Virtex-4 devices to deliver industry-leading performance while dramatically lowering power consumption.

One of our key inputs from many engineers was that performance and power were very important constraints in their systems designs, and that they needed both high performance and low power. With a dual-oxide 90 nm process, we would have had to choose performance or power. This wasn't good enough. By employing a triple-oxide 90 nm process, we achieved high performance and low power.

The 10-layer copper refers to the number of metal interconnect layers and their material, which is copper rather than aluminum (the traditional material). More layers provide more routing in less space and shorter connection distances. Copper reduces resistance compared to aluminum, and thus speeds signal interconnect and reduces on-chip power-distribution IR drop. As clock rates go up and voltages go down, these considerations have become increasingly important, and have driven the industry-wide shift to copper interconnect.

The Virtex-4 logic fabric was completely re-engineered to fully take advantage of the 90 nm triple-oxide CMOS process, resulting in the highest performance fabric ever, with system clock rates in excess of 500 MHz (at three LUT levels). At the same time, static power was cut in half compared to 130 nm Virtex-II Pro™ devices, as was dynamic power.

Thus, while some industry pundits were proclaiming that the future of deep submicron CMOS devices was getting hotter and hotter, with chip temperatures destined to reach that of rocket nozzles and the surface of the sun, the Virtex-4 design's creative approach has turned that conventional wisdom on its head, resulting in overall power reductions of 50% compared to our previous 130 nm generation. In many applications, such as DSP functions, power levels are reduced even more – as much as 90%. No wonder design engineers say that Virtex-4 FPGAs are cool – they literally are.

High-Performance Clocking

Clocks were rated as one of the most important and critical FPGA resources in our surveys of design engineers. Quantity, quality, connectivity, frequency, duty cycle, jitter, and skew all made a big difference.

To take clocking to the next level in Virtex-4 devices, all global clock resources were made fully differential, thereby reducing skew, jitter, and duty-cycle distortion. This marks the first implementation of differential clocking in a programmable logic device. Not only that, but the number of global clocks was increased to 32, for every

device, and internal connectivity options enhanced to allow any region to use any 8 clocks simultaneously.

500 MHz Synchronous Memories and FIFOs

On-chip synchronous block RAM was enhanced to run at 500 MHz. Built-in support for first-in first-out (FIFO) memories was included directly in the block RAM unit, enabling the same 500 MHz operation for FIFOs (approximately a 2X speedup over fabric-based FIFOs), while eliminating the need for any additional logic cells or complex FIFO designs.

If you're designing systems requiring ECC (error checking and correcting) memory, Virtex-4 devices have built-in ECC support, with single-bit correct and double-bit detect. ECC is common in infrastructure equipment in networking, telecom, storage, servers, instrumentation, and aerospace applications, and provides the highest levels of data integrity. Like the integrated FIFO support, the integrated ECC eliminates the cost and delay of fabric-based solutions.

Speaking of on-chip memory, Virtex-4 devices continue to offer SelectRAM™ memory, whereby each LUT is transformed into a 16 x 1 RAM, ideally suited for building high-speed register files and local buffers.

At the other end of the spectrum, interfaces to external memory devices such as DDR, DDR2, QDR-II, and RLDRAM-II are dramatically enhanced through our new ChipSync™ technology, which offers memory interface speeds at rates limited only by the speed of the external memory devices.

The new Virtex-4 ML461 Advanced Memory Development System contains fully functional and hardware-proven reference designs for all of today's most popular memory technologies. If you plan to use external memory, I highly recommend that you check this out.

DSP Performance of 256 GigaMAC/s

In the DSP domain, we incorporated some of the world's fastest multiply accumulate (MAC) technology. The XtremeDSP™ slice can perform an 18 x 18 signed multiply and 48-bit accumulate every 2 ns.

The Virtex-4 LX, FX, and SX platforms include the breakthrough XtremeDSP technology. With the new SX platform we did something completely new – we dramatically increased the ratio of DSP units to logic cells. Given the highly integrated nature of XtremeDSP slices, they need only small amounts of logic fabric to implement most common DSP functions, and thus increasing the ratio provides a significant increase in DSP compute power per unit silicon area. In fact, SX devices provide a 10X performance increase per unit cost over previous solutions.

Power is dramatically reduced as well, with more than a 10X reduction for multiply/add functions from previous FPGA solutions. The Virtex-4 SX55 contains 512 XtremeDSP slices, providing an aggregate DSP compute performance of 256 GigaMAC/s, making it one of the most powerful DSP devices ever manufactured.

The state-of-the-art XtremeDSP slice employs new "silicon algorithms" developed by a company called Arithmatica™. Many different architectures exist for implementing multiplication, and the Arithmetica architecture is truly a breakthrough. We are excited to see it available for the first time to FPGA users. For more information, visit Arithmatica's website at *www.arithmatica.com*.

The Evolution of Advanced I/O Technology

I/O continues to be a critical success factor for today's systems designers. During the last decade, we have seen four major changes in I/O. First was the shift away from 5V, the result of the need to scale voltages as we scaled the transistor. This in turn led to the plethora of I/O standards that we are all familiar with today: SSTL, HSTL, LVDS, and LVCMOS 1.5. The Virtex-4 SelectIO™ resource continues to lead the industry, supporting virtually every I/O standard in use today on every pin.

XCITE On-Chip Termination

The second major change was the transition from lumped loads to transmission line loads – again the direct result of Moore's Law. As transistors got faster and clock rates increased, I/O edge rates increased as well. But because the propagation speed of signals is a constant, dictated by the speed of light, we entered the realm in which a signal on one end of a wire was no longer the same as the signal on the other end of the same wire. This is what transmission lines are all about, and their appearance during the last few years has driven a sea change in all aspects of signal interconnect and I/O design.

To make sure that these signal "waves" don't start "splashing" uncontrollably, transmission lines need to be driven, built, and received using proper signal integrity approaches, the most critical of which is termination. Traditionally implemented with discrete resistors on the PCB, termination layouts can become exceedingly difficult around high-density pinouts like those used in FPGAs. This often dictates more PCB layers and thus more system cost.

Virtex-4 FPGAs include our third-generation of XCITE™ integrated digitally controlled termination technology. Offering a precisely controlled source impedance at the output drive pin, it is designed to enable the driving of transmission lines without external components, with maximum speed and signal integrity, and with straightforward PCB layout and layer stack-ups.

Likewise, on inputs, XCITE offers parallel termination for single-ended inputs and true differential termination for differential inputs. Termination occurs on the end of the transmission line at the die, not on the way there on the PCB, offering maximum signal integrity. Many customers report that the XCITE technology has saved them many PCB layers, increased PCB packing density, and saved them substantial dollars in their bill of materials.

Source-Synchronous Interfaces

The third major change was the shift from system-synchronous to source-synchronous interfaces. Traditional system-synchronous interfaces work by distributing a single clock to all transmitters and receivers in the system, and transmitting data between source and destination within a single clock cycle. This makes the data rate inversely proportional to the sum of clock-

to-out, transmission line delay, and input setup time.

Typically, system synchronous interfaces top out at speeds in the range of 100 MHz. To go faster, source-synchronous interfaces transmit a clock along with the data, and the receiver uses this clock to capture the data. Using this technique, along with double-data-rate transmissions, enables parallel I/O data rates in excess of 1 Gbps.

The challenge of source-synchronous interfaces is that each interface generates a new clock domain at the receiver. On top of this, to operate at high speeds, the precise alignment of clock and data at the receiver is paramount. To address this new world of source-synchronous interfaces, Virtex-4 devices include the breakthrough ChipSync technology. ChipSync units lie between the SelectIO technology and the core FPGA fabric, are available on every I/O pin on the device, and serve to transmit and receive high-speed source-synchronous data and clocks, achieving speeds of 1 Gbps per pin pair.

On the receiver, precise digital delay lines work internally to align data signals to each other, and then to align these to the received clock. The captured data is synchronized and transferred to the selected FPGA core clock domain.

To operate at maximum data rates, the transmit and receive units include parallel-to-serial and serial-to-parallel conversion units, respectively. Using ChipSync technology is virtually automatic for most designs, as it is utilized automatically in the various Xilinx IP cores and reference designs.

Networking interfaces such as SPI-4.2 and HyperTransport™, and memory interfaces such as DDR, DDR2 SDRAM, and QDR II SRAM, all employ the Virtex-4 ChipSync technology. And if you're designing your own source-synchronous interface, the ChipSync wizard gives you complete control and an easy-to-use GUI that lets you dial in exactly what you want to build.

Multi-Gigabit Serial Interfaces

The fourth major change in I/O has been the rapid adoption of high-speed serial interfaces. For years, serial interfaces were limited to long-distance communications, such as those used in fiber-optic links in the SONET/SDH world and the Ethernet links like 100BASE-T.

A key breakthrough occurred in the late 1990s, in which high-speed serial transceivers (which traditionally had been designed using complex process technology such as GaAs [Gallium-Arsenide]) were for the first time created using advanced design techniques using standard CMOS. Once implemented in CMOS, these transceivers had lower cost and much lower power, and could even be integrated into complex CMOS chips.

Virtually overnight, gigabit serial technology changed from a rare, expensive, and power-hungry technology to a common, low-cost, and very power-efficient technology. This has been the economic and technical impetus behind the industry's "Serial Tsunami," in which interface after interface has shifted from parallel to gigabit serial links. Two common examples are visible in today's computer architectures, with the shift from parallel PCI to 2.5 Gbps serial PCI-Express™, and the shift from the parallel ATA drive interface to the Serial ATA interface.

There are more than a dozen multi-gigabit serial interfaces in widespread use today, with more being introduced every year. The Virtex-4 FX family provides our third-generation RocketIO™ multi-gigabit serial transceiver technology. Spanning speeds from 622 Mbps to more than 10 Gbps, each Virtex-4 RocketIO transceiver is programmable and can implement a myriad of speeds and serial standards. Link-layer IP is available for such standards as PCI Express, Serial-ATA, FibreChannel, Gigabit Ethernet, and Aurora, to name a few.

In addition, Virtex-4 FX devices each include multiple embedded tri-mode (or 10/100/1000) Ethernet MACs, making implementation of compliant Ethernet devices simpler and faster than ever.

Application-Specific Embedded Processing

Virtex-4 embedded processing solutions include full support for both MicroBlaze™ 32-bit soft CPUs on all devices, and embedded PowerPC™ 32-bit RISC CPUs on all Virtex-4 FX devices.

The number of CPUs in one device is limited only by your imagination, and of course by the available logic cells. The high-performance PowerPC CPU runs at clock rates up to 450 MHz and delivers up to 702 DMIPS each. The first PowerPC processor available by any manufacturer on 90 nm, the PowerPC processor is incredibly power-efficient, using only 0.29 mW/DMIPS. This makes it among the lowest power microprocessors available from any manufacturer worldwide.

New Auxiliary Processor Unit (APU) technology connects the CPU to the FPGA fabric, enabling implementation of acceleration hardware for virtually any application. Once only the domain of high-budget ASIC and ASSP design teams, the Virtex-4 FPGA's architectural ability to combine application-specific hardware acceleration with high-performance RISC CPUs shatters traditional barriers of cost, time-to-market, and risk.

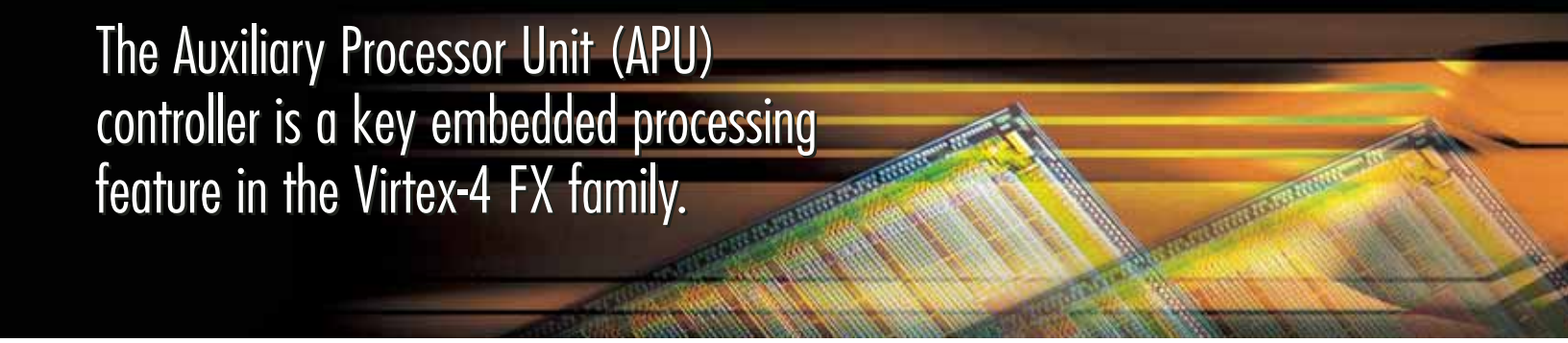
During the next few years, I expect to see more and more instances of application-specific acceleration, as it truly offers the ability to deliver very high performance at low cost and low power. A recent research program completed within Xilinx Research Labs, led by Dr. Kees Vissers, demonstrated a 20-fold speedup for anencryption/decryption (IPSEC) application over the base PowerPC processor. Using only 135 mW, it outperforms a 3.2 GHz Pentium™-4, while at the same time reducing power by 99%. That, in my opinion, is what state-of-the-art embedded processing is all about.

Conclusion

I hope that you've enjoyed reading a bit about the Virtex-4 Platform FPGA and the factors that drove its design. From the breakthrough ASMBL architecture and the triple-oxide 90 nm CMOS process technology, to the world's most capable embedded processing and multi-gigabit serial solutions, Virtex-4 devices offer an unparalleled set of enabling technologies for your next-generation systems designs. I look forward to seeing the creativity of the world's designers in tomorrow's products.

Accelerated System Performance with APU-Enhanced Processing

The Auxiliary Processor Unit (APU) controller is a key embedded processing feature in the Virtex-4 FX family.



by Ahmad Ansari
Senior Staff Systems Architect
Xilinx, Inc.
ahmad.ansari@xilinx.com

Peter Ryser
Manager, Systems Engineering
Xilinx, Inc.
peter.ryser@xilinx.com

Dan Isaacs
Director, APD Embedded Marketing
Xilinx, Inc.
dan.isaacs@xilinx.com

The APU controller provides a flexible high-bandwidth interface between the reconfigurable logic in the FPGA fabric and the pipeline of the integrated IBM™ PowerPC™ 405 CPU. Fabric co-processor modules (FCM) implemented in the FPGA fabric are connected to the embedded PowerPC processor through the APU controller interface to enable user-defined configurable hardware accelerators. These hardware accelerator functions operate as extensions to the PowerPC 405, thereby offloading the CPU from demanding computational tasks.

APU Instructions

The APU controller allows you to extend the native PowerPC 405 instruction set with custom instructions that are executed by the soft FCM; the primary capabilities are shown in Figure 1. This provides a more efficient integration between an application-specific function and the processor pipeline than is possible using a memory-mapped coprocessor and shared bus implementation.

The instructions supported by the APU are classified into three main categories:

- User-defined instructions (UDI)
- PowerPC floating-point instructions
- APU load/store instructions

The UDIs are programmed into the controller either dynamically through the PowerPC 405 device control register (DCR) or statically when the FPGA is configured through its bitstream. The APU controller allows you to optimize your system architecture by decoding instructions either internally or in the FCM.

The floating-point unit (FPU) is an example of an FCM. The PowerPC floating-point instruction set is decoded in the APU controller, whereas the computational functionality is implemented in the FPGA fabric. To support FPUs with different complexities, the APU controller allows you to select subgroups of the PowerPC floating-point instructions. These instructions are executed in the FCM while other subgroups of instructions are either computed through software FPU emulation or ignored completely. This fine-tuning optimizes FPGA resources while accelerating the most critical calculations with dedicated logic.

The APU controller also decodes high-performance load and store instructions between the processor data cache or system memory and the FPGA fabric. A single instruction transfers up to 16 bytes of data – four times greater than a load or store instruction for one of the general purpose registers (GPR) in the processor itself. Thus, this capability creates a low-latency and high-bandwidth data path to and from the FCM.

APU Controller Operation

Figure 2 identifies the key modules of the APU controller and the 405 CPU in relation to the FCM soft coprocessor module implemented in FPGA logic. To explain the operation of the APU controller and the processor interactions related to the execution units in soft logic, we can trace the step-by-step sequence of events that occur when an instruction is fetched from cache or memory.

Once the instruction reaches the decode stage, it is simultaneously presented to both the CPU and APU decode blocks. If the instruction is detected as a CPU instruction, the CPU will continue to execute the instruction as it would normally. Otherwise, within the same cycle, the CPU

- **Extends PPC 405 Instruction Set**
  - Floating Point Support (with soft auxiliary processor)
  - User-Defined Instructions
- **Offloads CPU-Intensive Operations**
  - Matrix Calculations
    - Video Processing
  - Floating-Point Mathematics
    - 3D Data Processing
- **Direct Interface to HW Accelerators**
  - High Bandwidth
  - Low Latency

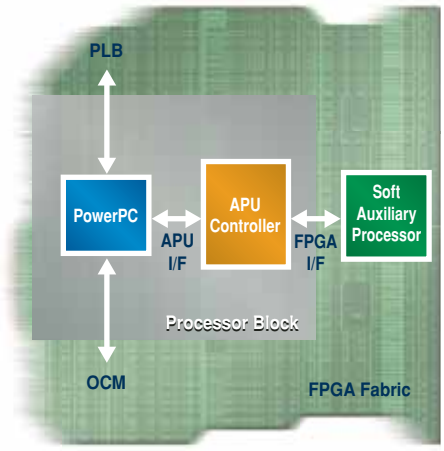


Figure 1 – APU expanded processing capabilities

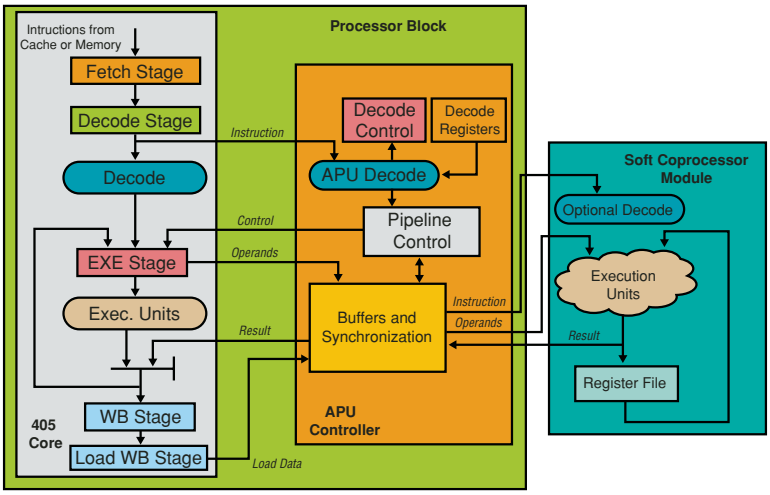


Figure 2 – APU controller processing operative block diagram

will look for a response from the APU controller. If the APU controller recognizes the instruction, it will provide the necessary information back to the CPU.

If the APU controller does not respond within that same cycle, an invalid instruction exception will be generated by the CPU. If the instruction is a valid and recognized instruction, the necessary operands are fetched from the processor and passed to the FCM for processing.

Because the PowerPC processor and the FCM reside in two separate clock domains, synchronization modules of the APU controller manage the clock frequency difference. This allows the FCM to operate at a slower frequency than the processor. In this instance, the APU controller would receive the resultant data from the coprocessor and at the proper execution time send the data back to the processor. The APU controller knows in advance, based on instruction type, if or when it will get the result.

Autonomous and Non-Autonomous Instructions

Two major categories of instructions exist: autonomous and non-autonomous. For autonomous instructions, the CPU continues issuing instructions and does not stall while the FCM is operating on an instruction. This overlap of execution allows you to achieve high performance through techniques such as software pipelining.

On the other hand, during the synchronized execution, the CPU pipeline stalls while the FCM is operating on an instruction. This feature allows you to implement synchronization semantics to pace the software execution with the hardware FCM latency.

Non-autonomous instruction types are further divided into blocking and non-blocking. If blocking, asynchronous exceptions or interrupts are blocked until the FCM instruction completes. Otherwise, if non-blocking, the exception or interrupt is taken and the FCM is flushed.

Software Description

Software engineers can access the FCM from within assembler or C code. On one side, Xilinx has enabled the GCC compiler (which is contained in the Embedded Development Kit) to generate code that uses an FCM floating-point unit to calculate floating-point operations. Furthermore, assembler mnemonics are available for UDIs and the pre-defined load/store instructions, enabling you to place hardware-accelerated functions into the regular program flow. For the ultimate level of flexibility, you can define your own instructions designed specifically for the hardware functionality of the FCM.

You can easily use the pre-defined load/store instructions through high-level C macros. For example, in an application where the FCM is used to convert pixel data into the frequency domain, 8 pixels of 16 bits are transferred from main memory to an FCM register with a simple program:

```
unsigned short pixel_row[8];  // 8 pixels,
each pixel has a size of 16 bits
lqfcm(0, pixel_row);  // transfer a row of
pixels to FCM register zero
```

The quadword load operation maintains cache coherency as the data is moved through the cache, if caching is enabled for the corresponding address space.

The FCM operation on the pixel data can start on an explicit command; for example, a UDI. However, for many applications the operation starts immediately after the FCM hardware detects the completion of the load instruction.

The latter approach has many advantages:

- Simple software – A load operation moves the data from the memory to

the FCM and starts the operation. A subsequent store instruction retrieves the result of the operation and stores it back to main memory.

- High data transfer rates – Quadword load and store operations take just a few cycles to complete. A single operation moves 16 bytes within that timeframe.
- Low latency – FCM load operations are simple to use. The processor completes the operation in a single cycle.

The principle of the RISC architecture uses a number of simple instructions on data stored in general-purpose registers (GPR) to compute complex operations. User-defined instructions fall into this category but take the concept a step further in that the system architect defines the complexity of the operation on data stored in GPRs and FCM registers (FCR). Again, from a software point of view, the engineer codes user-defined instructions through C macros. GCC recognizes mnemonics such as udi0fcm as a user-defined operation of the general form:

```
udi0fcm<FCRT5/RT5>,<FCRA5/RA5/imm>,
<FCRB5/RB5/imm>
```

The target of the operation is either a GPR or an FCR. The operands are either GPRs, FCRs, immediate values, or a combination. As you can see, the semantics are not defined by the instruction and depend on your intentions and the implementation in the FCM.

This code sequence demonstrates the use of a user-defined instruction as an example of a complex add operation:

```
struct complex {
        int r, i;     // 32 bit integer for real
        and imaginary parts
};
complex a, b, r;
ldfcm(0, &a);     // load complex number a
into FCM register 0
ldfcm(1, &b);     // load complex number b
into FCM register 1
udi0fcm(2, 1, 0);// udi0fcm computes r = a
+ b, where r is stored in FCM register 2
stdfcm(&r, 2);    // store complex result
from FCM register 2 to variable r
```

To increase the readability of the code, you can redefine the user-defined instruction with regular C preprocessor constructs. Instead of using the udi0fcm() macro, you can redefine it to a more comprehensible complex_add() macro with `#define complex_add(r, a, b) udi0fcm(r, a, b)` and change the listing to call complex_add(2, 1, 0) instead of udi0fcm(2, 1, 0).

Therefore, system architects can partition their tasks into hardware- and software-executed pieces that are efficiently and precisely interfaced to one another through the use of the APU controller. This partitioning can be done statically during the initial system configuration or dynamically during the program execution. Using the direct processor/FPGA coupling presented by the APU controller and its high throughput interfaces, hardware/software synchronization is greatly simplified and performance significantly improved.

Accelerating System Performance

The following examples showcase key advantages the APU provides based on two different scenarios. The first scenario is essentially a benchmarking comparison of a finite impulse response (FIR) filter using a soft FPU core, implemented as an FCM attached directly to the APU controller (as compared to software emulation used to calculate the filter function). The second scenario implements a two-dimensional inverse discrete cosine transform (2D-IDCT) typically used as one of the processing blocks in MPEG-2 video decompression, again compared to emulating the 2D-IDCT function in software.

The two use cases are different in that the FPU implements a set of registers in the FPGA fabric upon which the FPU instructions operate. The 2D-IDCT only requires load and store operations, while the functionality of the operation on the data stream is fixed. In either case the operations are complex enough to justify offloading into the FPGA fabric.

Thus, the combination of using the APU and FPGA hardware acceleration clearly provides a significant performance advantage over software emulation – or the conventional method involving the processor and processor local bus architecture with a soft co-processing function.

FIR Filter

The implementation of floating-point calculations in hardware yields an improvement by a factor of 20 over software emulation. Connecting the FPU as an FCM to the APU controller provides performance improvement because the latency to access the floating-point registers is reduced and dedicated load and store instructions move the operands and results between the FPU registers and the system memory.

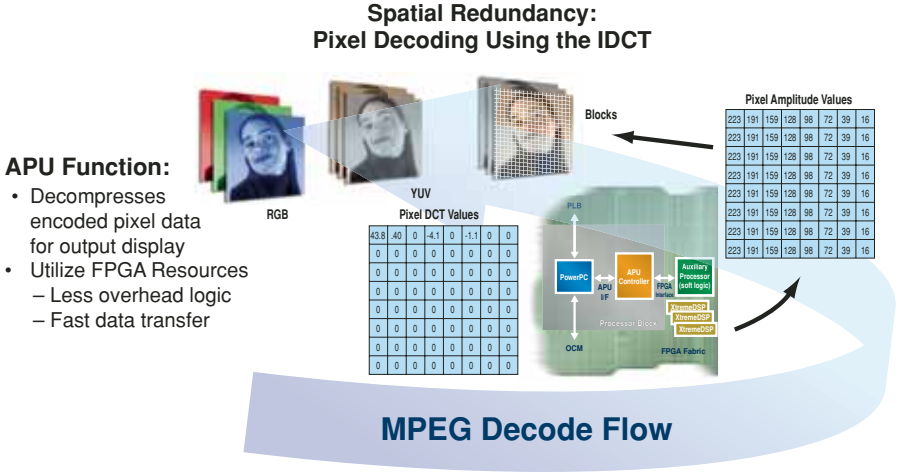


Figure 3 – Utilizing APU to decode pixel data for display output

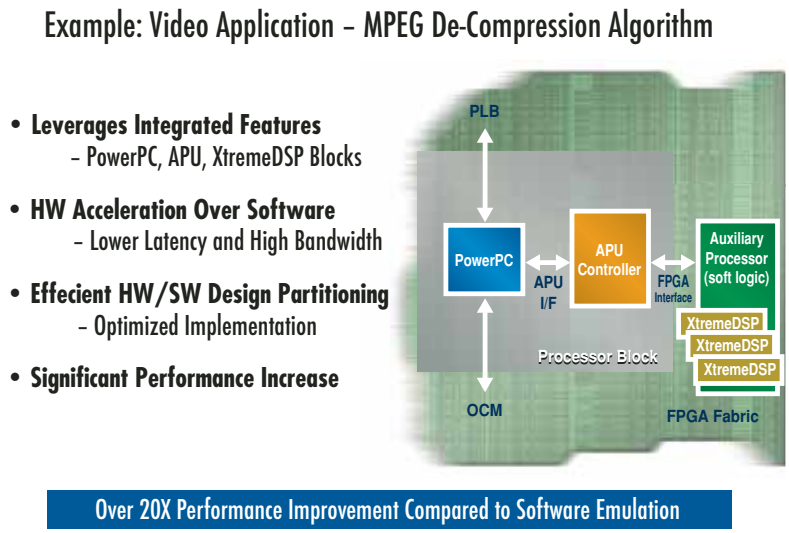


Figure 4 – Accelerated system performance with APU

2D-IDCT

The 2D-IDCT transforms a block of 8 x 8 data points from the frequency domain into pixel information. A high-level diagram depicting the pixel decode by the APU controller, along with advantages, is shown in Figure 3. In this example, each data point has a resolution of 12 bits and is represented as a 16-bit integer value. The data structure is defined where each row of 8 pixels consumes 16 bytes. This is an ideal size that allows optimal use of the FCM load and store instructions described earlier. In other words, eight FCM quadword load instructions are needed to load a data block into the 2D-IDCT hardware. Eight FCM quadword store instructions are sufficient to copy the pixel data back into the system memory.

The calculation of the 2D-IDCT in the FCM starts immediately after the first load, and the pixel data is available shortly after the last load operation. As shown in Figure 4, the 2D-IDCT makes uses of the new XtremeDSP™ slices in the Virtex-4 architecture that offer multiply-and-accumulate functionality.

A software-only implementation of a 2D-IDCT takes 11 multiplies and 29 additions together with a number of 32-bit load and store operations, while the hardware-accelerated version takes 8 load and 8 store operations. The reduced number of operations results in a speed-up of 20X in favor a 2D-IDCT FCM attached through the APU controller.

By comparison, if you connect the 2D-IDCT hardware block to the processor local bus, as it is done conventionally, the system performance will be reduced. This increased latency is mainly caused by the bus arbitration overhead and the large number of 32-bit load and store instructions. This is illustrated schematically in Figure 5.

Conclusion

The low-latency and high-bandwidth fabric coprocessor module interface of the APU controller enables you to accelerate algorithms through the use of dedicated hardware. Where operations are complex enough to justify the offloading into the FPGA fabric, or when acceleration of a specific algorithm is desired to achieve optimal performance, the combination of the APU controller and FPGA hardware acceleration provides a definitive performance advantage over software emulation or the conventional method of attaching coprocessors to the processor memory bus.

Generating the accelerated functions called by user-defined instructions is easily performed through GUI-based wizards. This functionality will be included in subsequent releases of the powerful Embedded Development Kit or Platform Studio.

If you are more comfortable working at the source code or assembly level, the APU controller allows you to define your own instructions written specifically for the hardware functionality of the FCM, or you can easily use the pre-defined load/store instructions through high-level C macros.

The APU controller provides a close coupling between the PowerPC processor and the FPGA fabric. This opens up an entire range of applications that can immediately benefit customers by achieving increases in system performance that were previously unattainable.

For additional details on the APU controller in Virtex-4-FX devices, including detailed descriptions and timing waveforms, refer to the Virtex-4 PowerPC 405 Processor Block Reference Guide at *www.xilinx.com/bvdocs/userguides/ug018.pdf.*

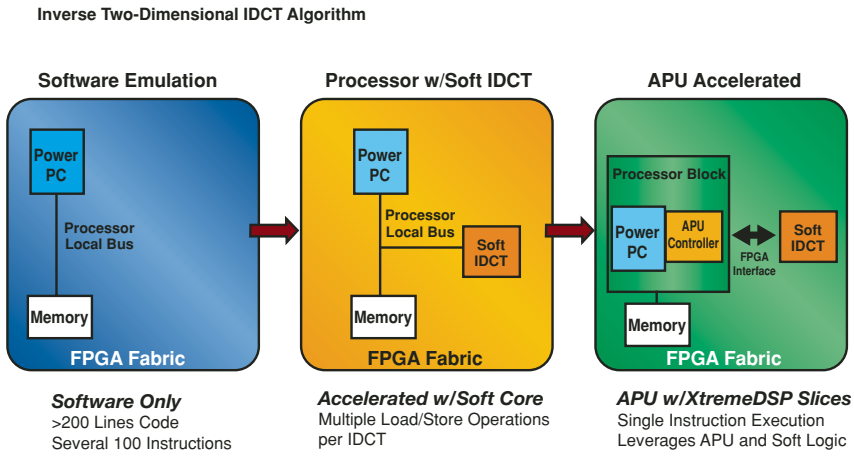


Figure 5 – Comparison of implementation models for 2D-IDCT

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.

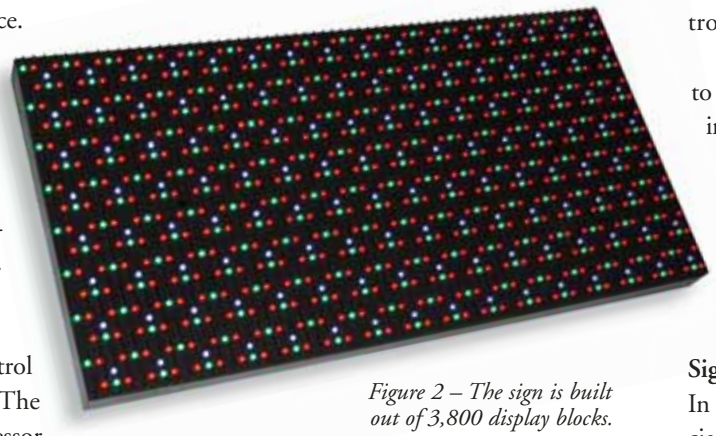


Figure 2 – The sign is built out of 3,800 display blocks.

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.

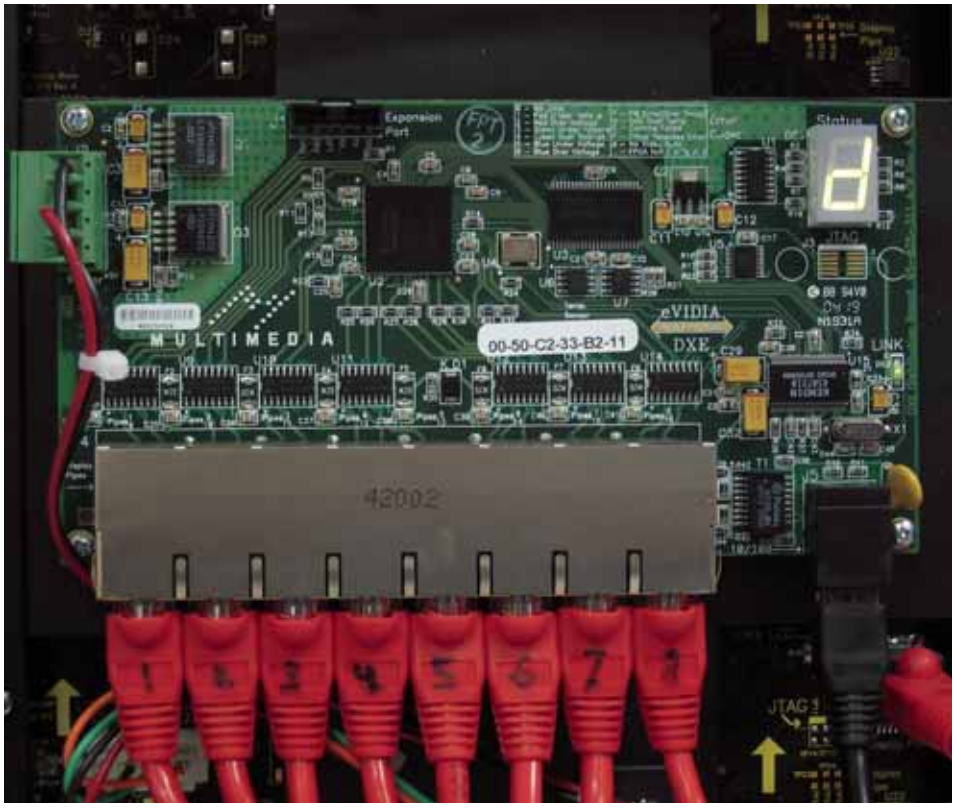


Figure 3 – The video data distribution board is based on an XC3S200 FPGA. It also includes SRAM, a 10/100 Ethernet port, a status display, and numerous connections to display blocks.

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.*

Xilinx Teams with Optos

Xilinx Design Services assists in the development of a new eye care technology.

by Barrie Mullins
Integrated Solutions Manager, Xilinx Design Services
Xilinx, Inc.
barrie.mullins@xilinx.com

Optos PLC, a revolutionary eye care technology company, selected Xilinx Virtex-II Pro™ FPGAs to be at the heart of their wide-field visual imaging system. To lower the risk, reduce the learning curve, and meet their time-to-market requirements for this new technology, Optos engaged Xilinx Design Services (XDS) to help them plan, design, implement, and deliver their complex solution.

The Optos Panoramic System comprises two IBM™ PowerPC™ 405 microprocessors and multiple proprietary interfaces, as well as industry standard interfaces. The design utilizes the latest technology, tools, and software provided by Xilinx and gave XDS the opportunity to apply its system, hardware, and embedded software knowledge, along with its project management expertise, to meet the time-to-market requirements.

The Design Challenge

For Optos, the challenge was to find a technology that provided them with the technical features to take their system from concept to reality. The system is used by a qualified operator to take a scan of the patient's eye while they look at a target presented on an LCD panel. This image capture uses Optos' patented technology in wide-field visual imaging and transfers the data to a microprocessor. The processor is used for storage of patient data and processing by the operator.

The challenge for XDS was to design, develop, test, simulate, integrate, and bring up the hardware and software on a single platform to enable a complete connectivity solution addressing all layers of the Optos application. The solution also had to be scaleable for future features and upgrades. XDS was also required to meet budget constraints.

The Teams

The Xilinx Design Services team worked with the Optos engineering team to set the system requirements and to agree on a schedule and deliverables for the project. Over the period of the project, the XDS project manager worked with the Optos project management team to ensure that milestones were delivered and that information flowed smoothly between the two teams.

Optos

Optos was founded to develop a technology that will provide ophthalmologists and optometrists with improved image capture and analysis capability for the early detection and prevention of eye disease. Optos' patented technology in wide-field visual imaging is unique. Their business strategy is to make their visual imaging system available to the greatest number of practitioners, thus allowing an overall improvement in eye care and early disease detection for a far greater number of people.

The Optos headquarters in Dunfermline, United Kingdom, is the base for research and development of the Panoramic diagnostic ophthalmic apparatus, the technology behind Optomap Retinal Images.

Xilinx Design Services

Optos contracted XDS to design and deliver the Virtex-II Pro 2VP20 design. By doing so, they reduced the risk and learning curve for this new technology, and they ensured they would meet their time-to-market goal. XDS gathered a team of experienced co-design engineers with in-depth knowledge of project management, embedded software

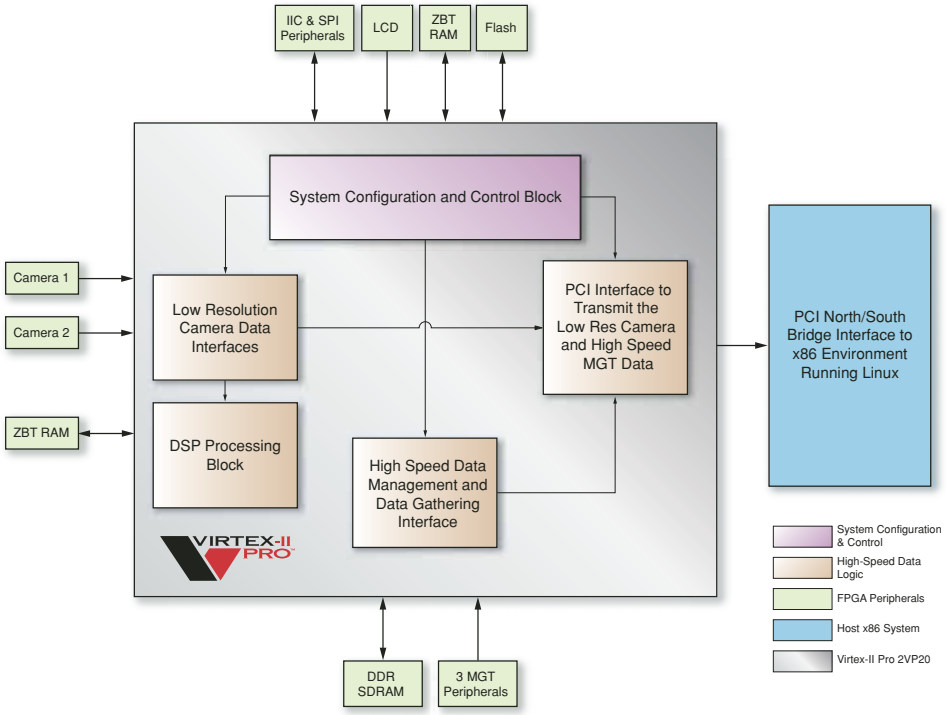


Figure 1 – Block diagram of Virtex-II Pro design in the Panoramic system

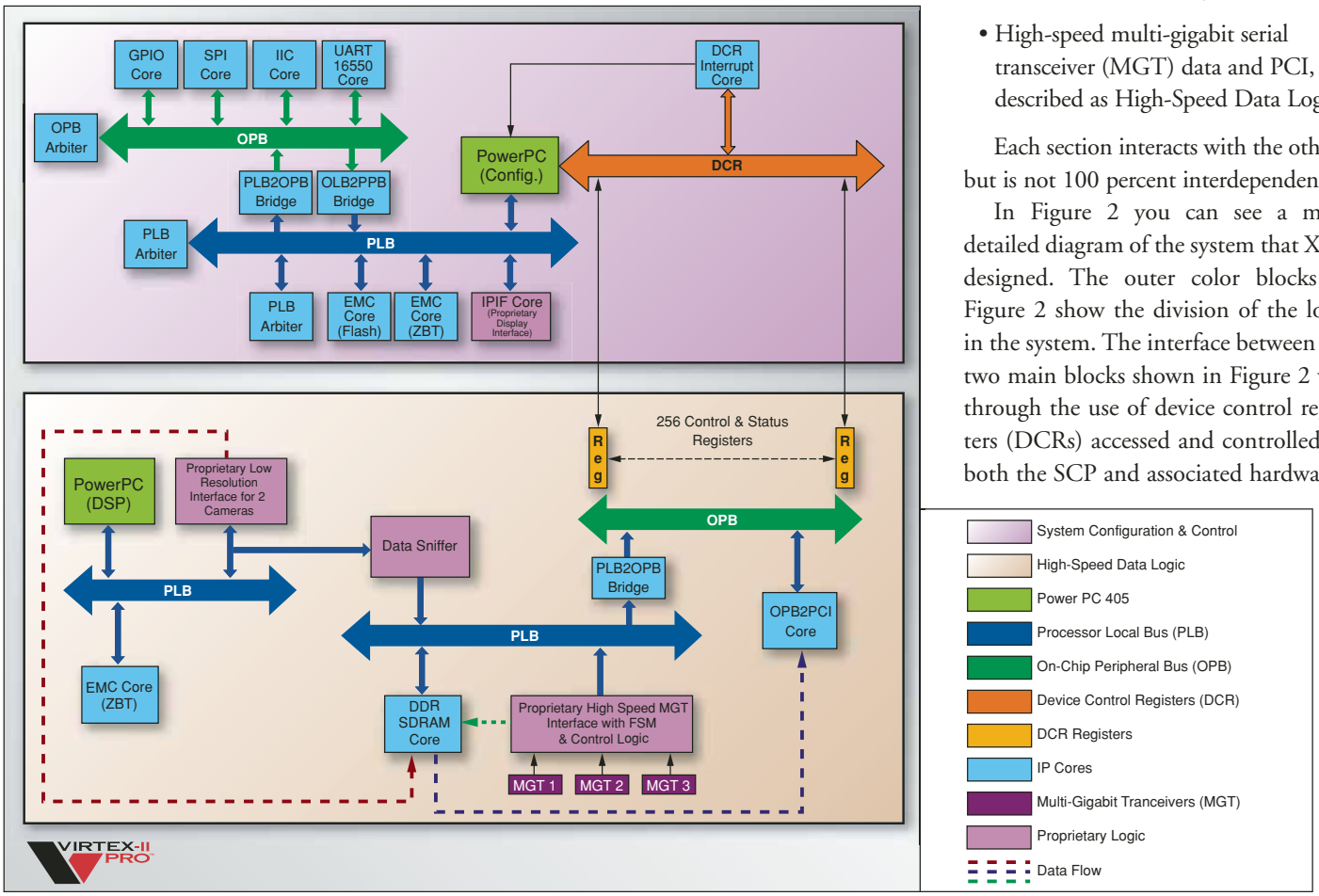


Figure 2 – Detailed view of Virtex-II Pro design in the Panoramic system

design, FPGA design, co-design tools, IP cores, and platform FPGAs.

The final delivery to Optos included the source code for the embedded test software, HDL designs, design scripts, HDL test vectors, simulation test software, build scripts, and associated testbenches. The delivery specifications also included all associated project documentation: project plan, design specifications, testbench strategy, software specifications, memory map, and verification plan.

The Panoramic System

Figure 1 shows the basic Panoramic system design, which includes:

- System control processor (SCP), described as System Configuration and Control in the diagram legend
- Camera and digital signal processing (DSP) components, described as DSP and Camera Logic
- High-speed multi-gigabit serial transceiver (MGT) data and PCI, described as High-Speed Data Logic.

Each section interacts with the others, but is not 100 percent interdependent.

In Figure 2 you can see a more detailed diagram of the system that XDS designed. The outer color blocks in Figure 2 show the division of the logic in the system. The interface between the two main blocks shown in Figure 2 was through the use of device control registers (DCRs) accessed and controlled by both the SCP and associated hardware.

System Control Processor

The main function of the SCP is system configuration management, coordination, status reporting, and control of the timing of events based on inputs. It starts image capture, and controls LCD display and other logic through registers in the system.

The main technical blocks of the SCP perform the following functions:

- Communicate with the Intel™ x86 operating system environment over PCI and RS-232 connections
- Configure the LCD panel and camera interfaces via IIC
- Display images on the LCD panel
- Communicate with peripheral subsystems via SPI
- Boot from internal Virtex-II Pro block RAM and load application from external flash memory to ZBT RAM before running it
- Schedule and control events in the application software via interrupts and GPIO
- Use DCR registers to gather status and control operations in other sections.

DSP and Camera Logic

The camera interfaces are required to support 30 frames per second (fps) of 640 x 480 pixels, requiring bandwidths greater than 9 Mbps for each camera. The data is stored locally in ZBT RAM so that a DSP algorithm running on the second PowerPC 405 microprocessor can be applied to the data. While the data is being received from the cameras, it is also copied to DDR SDRAM for transmission over the PCI to the host x86 memory space.

High-Speed Data Logic

RocketIO™ MGTs are used to receive high-speed data from three peripheral boards, each transmitting data from a single MGT on a Virtex-II Pro 2VP7. The MGT data transfers are initiated by an external system triggered by the operator. This event is synchronized using interrupts to the SCP. The data from the three MGTs is formatted in the Virtex-II Pro device and used to form a larger data packet, which is then stored in DDR SDRAM for transmission over PCI.

The PCI section is used to transfer all the data stored in the DDR SDRAM to the x86 host processor environment. XDS designed a number of proprietary hardware blocks to control the flow of data from DDR SDRAM to the OPB-PCI core and over to the host x86 memory space. Configuration and status of the PCI core is carried out by the SCP through DCR registers.

Tools

During this project, XDS used several software tools to create the Panoramic system, to simulate the design, and to manage the development environment to ensure traceability and quality releases. The tools used are listed in Table 1.

Function	Tool Used	Vendor
VHDL Design, Compile, Test	ISE 5.2i	Xilinx
Software Design, Compile, Test	EDK 3.2	Xilinx
Logic and Software Simulation	MTI 5.6d	Mentor Graphics
Version Management	CVS	Open Source

Table 1 – Software used to develop, simulate, and manage the development environment of the Panoramic system

Test and Verification

When designing a project of this size, test and verification are of paramount importance to the successful completion of the project. The XDS team focused a large number of resources to ensure the design met Optos' requirements.

The team developed a full testbench that would fit around the design (shown in Figure 2). This testbench was designed to simulate every interface in the design. It consisted of a PowerPC microprocessor with peripheral cores to test protocols and communication interfaces. Both the design and the testbench used the swift model for the PowerPC microprocessor to run the software during simulation. A swift model is a cycle-accurate simulation model that can interpret PowerPC instructions and act accordingly.

Two software applications were used to test the system in simulation. The first ran on the testbench and the second ran on the system under test (SUT).

The function of the SUT application is to interact with the peripheral cores and to drive the inputs and outputs of the SUT. The application in the testbench interacts with the SUT in the following manner:

- Data received from the SUT is stored in memory.
- Protocol data is decoded, stored, and replied to, if appropriate.

The application for the SUT allows hardware design engineers to control many different features and to run a suite of tests in any order required. The XDS team also designed tests to verify the pure hardware interfaces without interaction with the applications.

All functionality was tested at unit, functional, and system level. The tests were all documented in the test and verification plan, which was approved by Optos before the system design phase began.

Conclusion

With Xilinx Design Services, you get a highly skilled and experienced team that will help get you to market faster by bridging the learning curve, delivering quality on-time designs, and working in close contact with your development team.

In a letter to the engineers at Xilinx Design Services, David Cairns, technical director at Optos, wrote, "I have been impressed with Xilinx Design Services. Their commitment to quality and high standards has eased transition into production. XDS went the extra mile, and we could not have achieved our design goals without their help."

For more information about Xilinx Design Services, visit *www.xilinx.com/xds.*

Learn how the latest Xilinx technology can help you design cost effective solutions faster.

Gain hands-on experience to speed up your next development cycle.

What Are You Waiting For?
Register now for the event nearest you.

Visit www.memec.com/xfest-2005

XILINX®

Memec Insight

ASIA · CANADA · EUROPE · JAPAN · UNITED STATES

(MG0084-04) 12.20.04

What Customers and Partners are Saying about Xilinx Embedded Processor Solutions

Real breakthroughs. Real stories.

PowerPC

"We designed our latest Nova identity4 broadcast production video switcher with what we consider a truly disruptive technology utilizing only a single Virtex™-II Pro FPGA. This approach reduced our part count 10-fold over prior systems while providing a per-channel cost at a fraction of our competition.

"We are very excited about the new Virtex-4 FX family of devices. The higher performance processor and integrated APU will allow us to rapidly create hardware modules for accelerating specific software functions. By leveraging these capabilities with the integrated EMAC cores and enhanced RocketIO™ transceivers available in Virtex-4 FX devices, our next-generation systems will further widen the distance between us and our competition."

Xilinx Customer
Roger Smith, Chief Engineer, **Echolab**

"Leveraging a high degree of IP design re-use developed with our first Virtex-II Pro-based system, we can rapidly migrate our high-performance storage networking architecture to the Virtex-4 FX family of devices. By utilizing many of the FX features, we are able to increase both functionality and performance while reducing system cost.

"Key elements include the enhanced PowerPC™ processor running Linux OS to manage and control our high throughput multi-channel switching matrix and the integrated dual tri-mode Ethernet MAC and RocketIO transceivers for high-speed data transfer. The APU controller can also offload some tasks via custom-defined instructions, resulting in further computing and performance headroom."

Xilinx Customer
Sandy Helton, Chief Technology Officer,
SAN Valley Systems

MicroBlaze

"The MicroBlaze™ solution has proven to be far more than a companion microprocessor to our FPGA compute engines. It has become quite an integral part in shaping the way we are solving problems and is helping reduce our engineering costs.

"Currently, we use the MicroBlaze processor in our 68-billion-color LED display system to manage calibration and setup routines for the video processor module. The ability to easily modify these functions without completely changing the design allows us to offer a cutting-edge product."

Xilinx Customer
Ricardo Ramos, President and CEO,
Interativa Paineis Eletronicos

"Incorporating the MicroBlaze processor inside a single FPGA allowed us to use a low pin-count device and reduce PCB complexity and cost. But the most significant impact that the MicroBlaze processor has offered is the ability to almost completely change a design without needing to change the physical design whatsoever. As requirements change, we no longer need to remap functionality into different peripherals, as is often the case when moving to a larger microcontroller. We can add, delete, or move peripherals; the Embedded Development Kit (EDK) makes this effortless."

Xilinx Partner
Erik Widding, President,
Birger Engineering

"The storage module was a nice application for the MicroBlaze embedded processor. We wrote routines in C for the MicroBlaze core for asynchronous transfer, including the packet handler and interpretation. The routines that needed acceleration were re-coded in RTL and moved to hardware implementations on the same Virtex-II device. The storage elements needed to take data streams in, process them, and put the data out to disk. We could accomplish that 100% with the MicroBlaze core, just not at speed.

"The MicroBlaze processor allowed us to update the program without reprogramming the device, drop in a test program in C, set breakpoints, and functionally debug by examining memory, variables, and registers. This helped us debug the functionality very quickly. We also were able to do performance analysis and profiling that helped us decide what modules needed acceleration."

Xilinx Partner
Derek Stark, Senior Software Engineer,
Nallatech

PicoBlaze

"The favorite toy in our Xilinx bag-of-tricks is the PicoBlaze™ processor. We could not have completed the sign 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. 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."

Xilinx Partner
Jason Daughenbaugh, Sr. Design Engineer,
Advanced Electronic Designs (AED)

"We do four channels of PID + feedforward control + profile generation + host interface, all with 32-bit position resolution and 32-bit breakpoints at a sample rate of 10 KHz, with all four axes in motion. Not bad for an 8-bit microcontroller!"

Xilinx Customer
Peter Wallace, Partner, **Mesa Electronics**

"We needed the control that a CPU provides without the overhead of extra peripherals and external hardware. The PicoBlaze processor was easy to implement and worked very well in our custom protocol application."

Xilinx Customer
Scott Orangio, Senior Hardware Engineer,
Polycom Inc.

"The PicoBlaze processor made our transition towards embedding a software-based application into our well-known FPGA environment seamless and surprisingly simple."

Xilinx Customer
Moti Cohen, Hardware Design Engineer,
TeraSync

Xilinx Platform Studio

Regarding Xilinx Platform Studio winning the DesignVision Award:

"It is inspiring to see Platform Studio recognized in this way. The platform approach to next-generation embedded systems will become the only viable solution to meet the economic and technological challenges of the future. Xilinx continues to prove its undeniable leadership in developing world-class technologies that make FPGAs a compelling system platform for an expanding range of embedded processing applications."

Jerry Worchel, Principal Analyst,
In-Stat/MDR

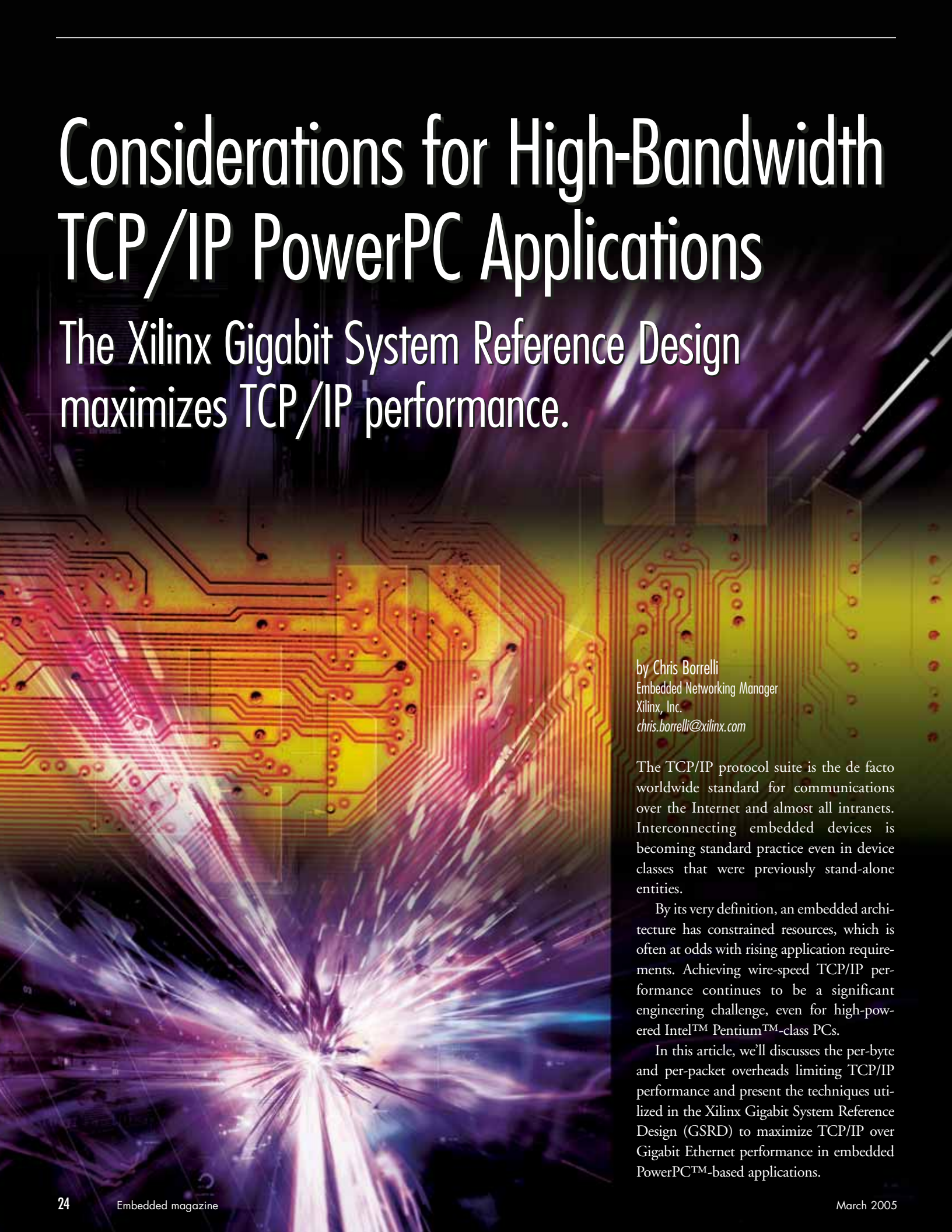
Regarding Xilinx Platform Studio 6.3i product release:

"The platform approach will not only find a permanent home in the FPGA world, but in the ASIC and ASSP worlds as well. With this announcement, Xilinx continues to prove its undeniable leadership in this market."

Max Baron, Principal Analyst,
In-Stat/MDR

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.

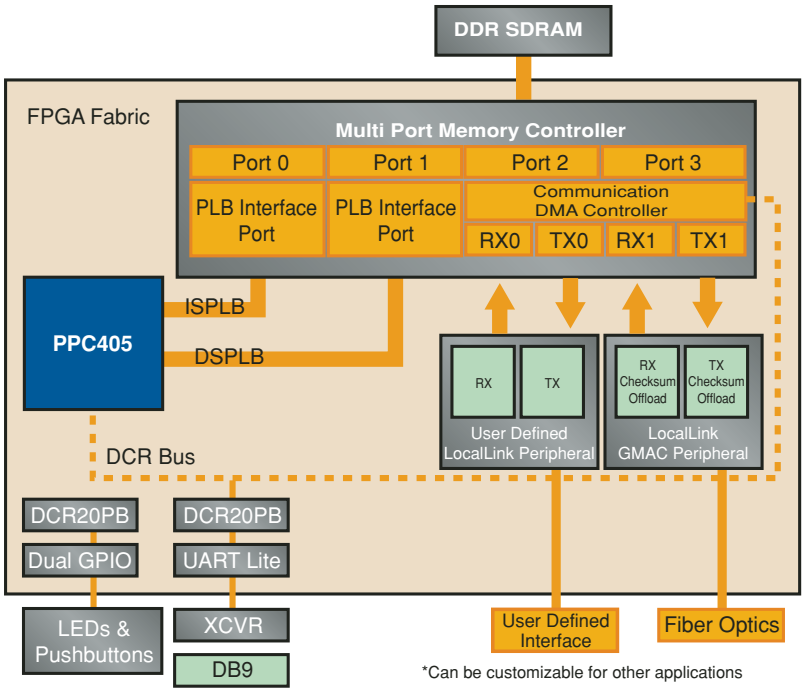


Figure 1 – GSRD system block diagram

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.

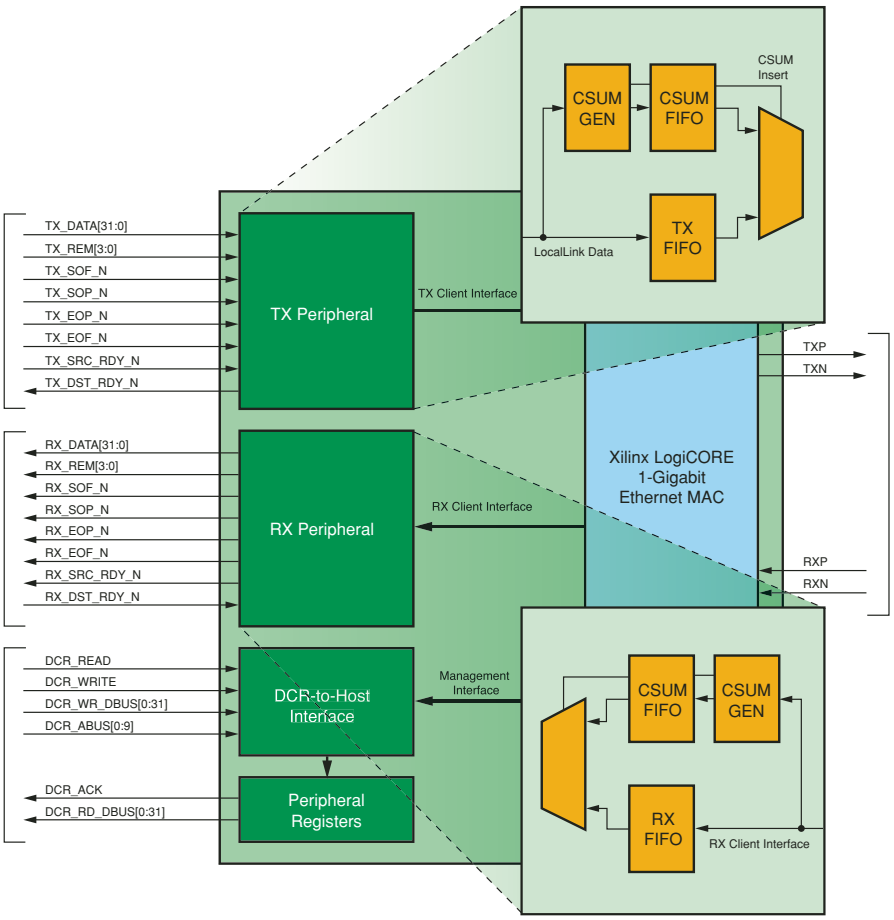


Figure 2 – LocalLink Gigabit Ethernet MAC peripheral block diagram

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*)

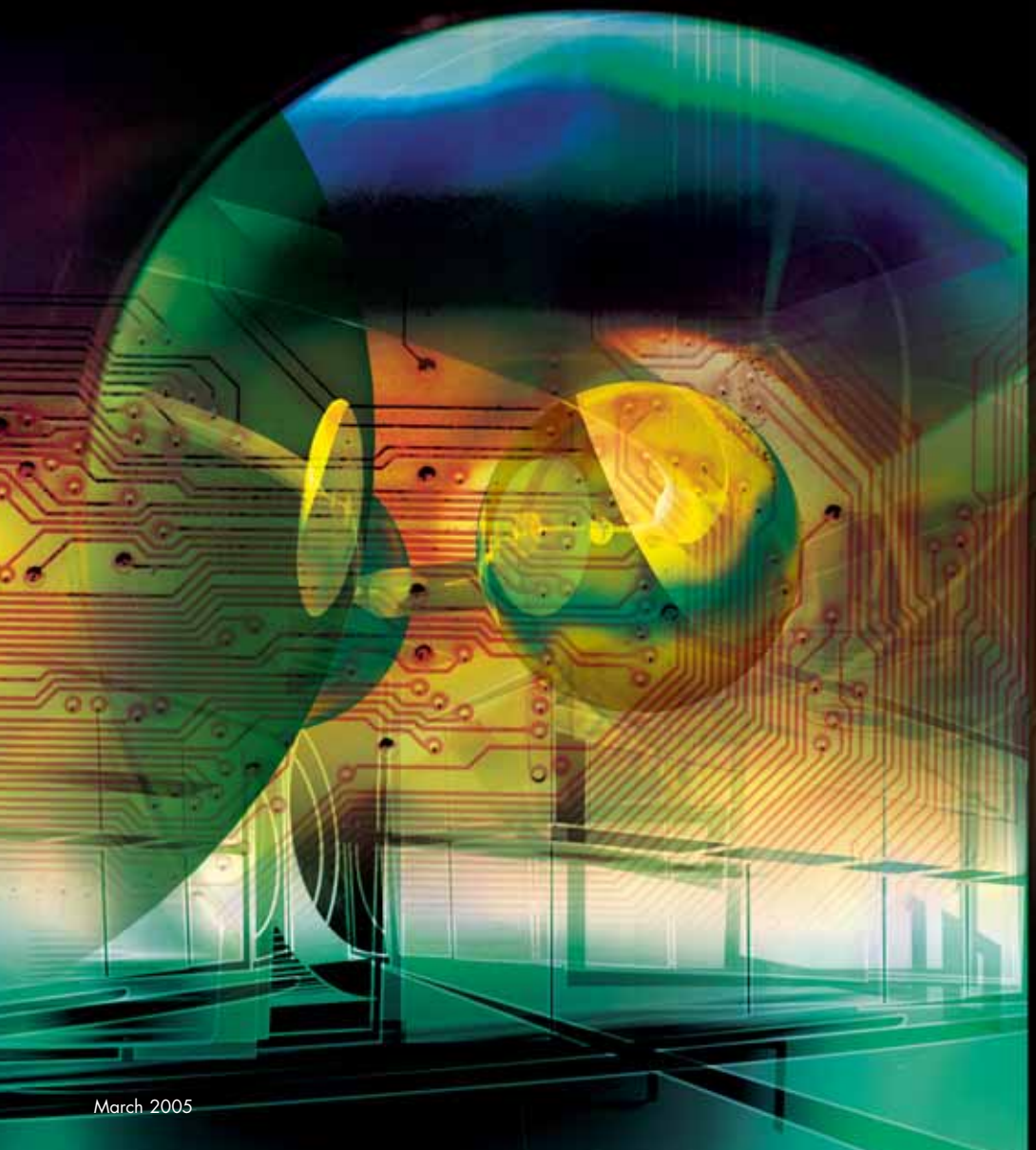
MontaVista Software (*www.mvista.com*)

"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).

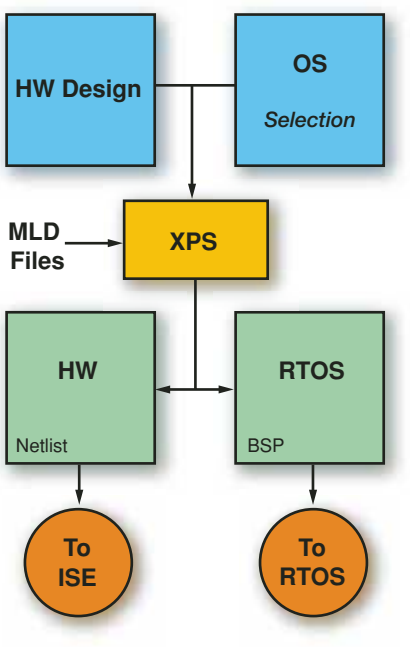


Figure 1 – Outline of an MLD-enabled system design

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*.

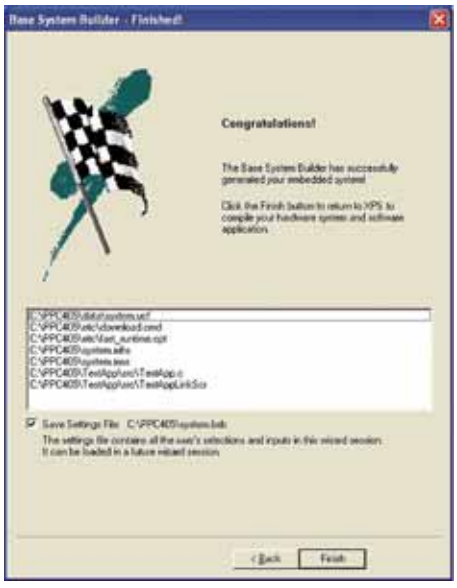


Figure 2 – The hardware design is complete and ready to configure the software.

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.

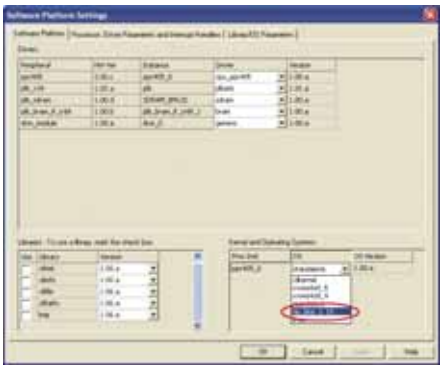


Figure 3a – After installing Nucleus PLUS in EDK, Nucleus appears as an option in the drop-down menu choosing which operating system to use with the PowerPC 405 processor.

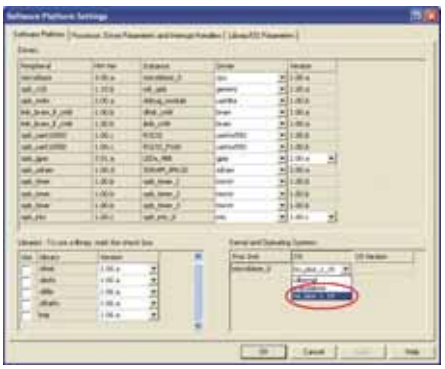


Figure 3b – Nucleus appears as an option in the drop-down menu choosing which operating system to use with the MicroBlaze soft-core processor.

- 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.

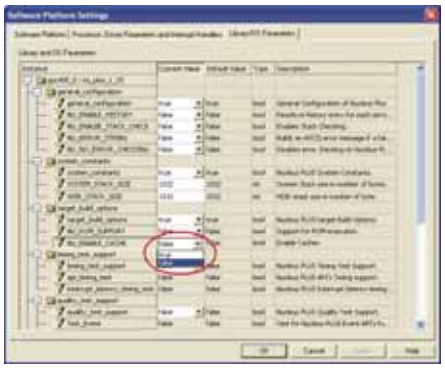


Figure 4 – Enabling the cache in the RTOS configuration parameters

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.

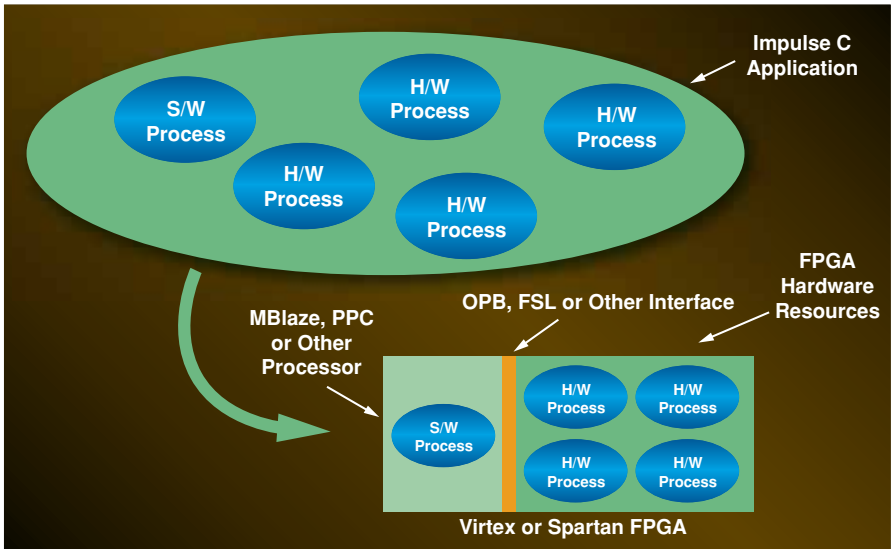


Figure 1 – Hardware and software test components are mapped to the FPGA target and communicate across the OPB or FSL bus to the MicroBlaze or Power PC processor.

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.

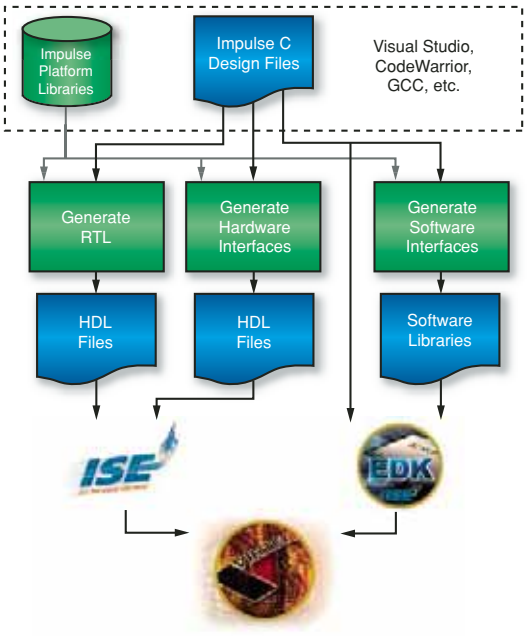


Figure 3 – Hardware and software C code is compiled and debugged in a standard IDE; the interfaces are automatically generated and the generated HDL synthesized in Xilinx tools before downloading into the Virtex-II, Virtex-II Pro, or Virtex-4 device.

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.

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.

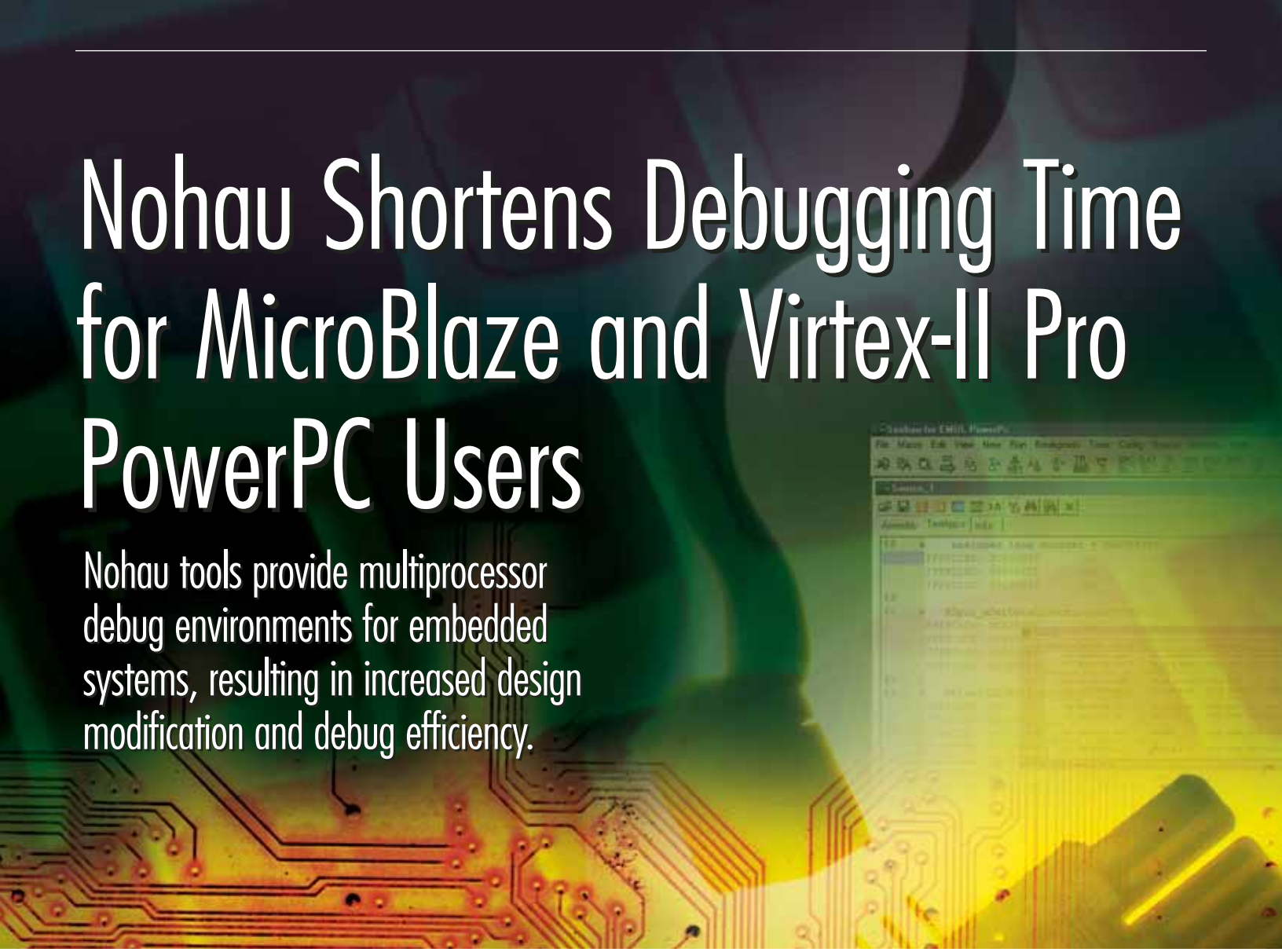
The FPGA dynamic probe, when combined with an Agilent 16900 Series logic analysis system, allows you to access different groups of signals to debug inside your FPGA—without requiring design changes. You'll increase visibility into internal FPGA activity by gaining access up to 64 internal signals with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

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

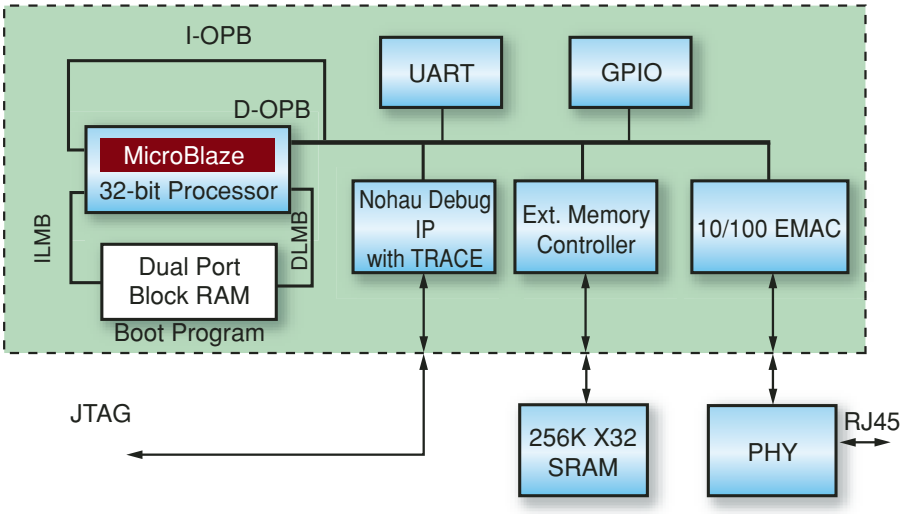


Figure 1 – Nohau Debug IP with trace, shown in a simple five-chip Internet-aware system

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.

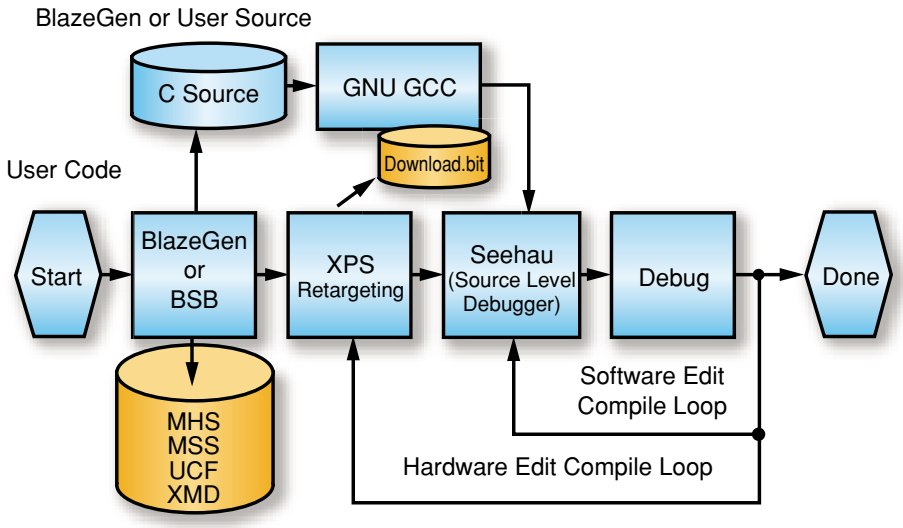


Figure 3 – Development flow with Seehau source-level debugger in place

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

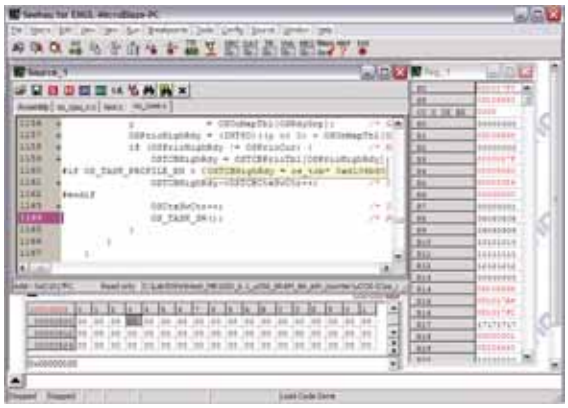


Figure 4 – Full source with breakpoint at context switch in µC/OS-II

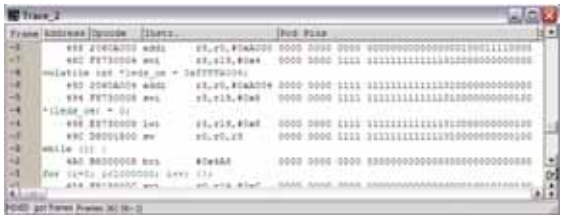


Figure 5 – Trace 40-bit mixed-mode display in binary logic signals from FPGA fabric

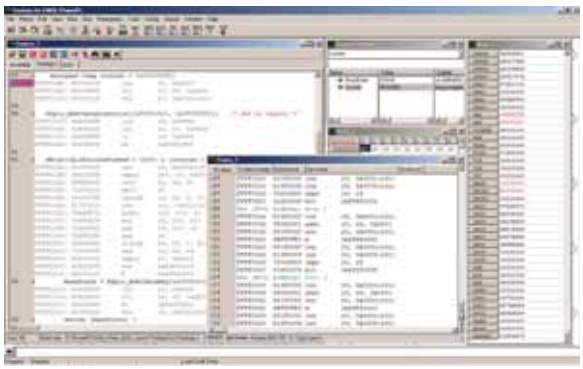


Figure 6 – PowerPC source-level debug with trace

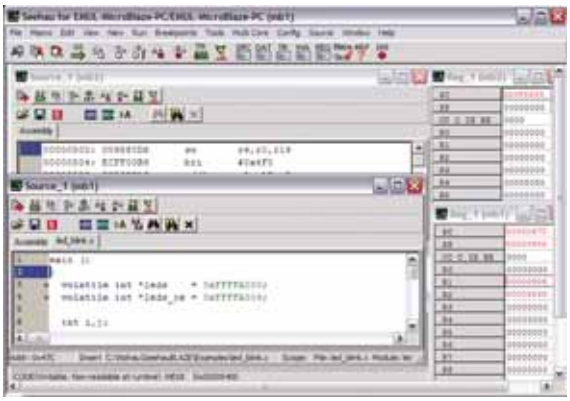


Figure 7 – Multi-core display with two MicroBlaze processors

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.

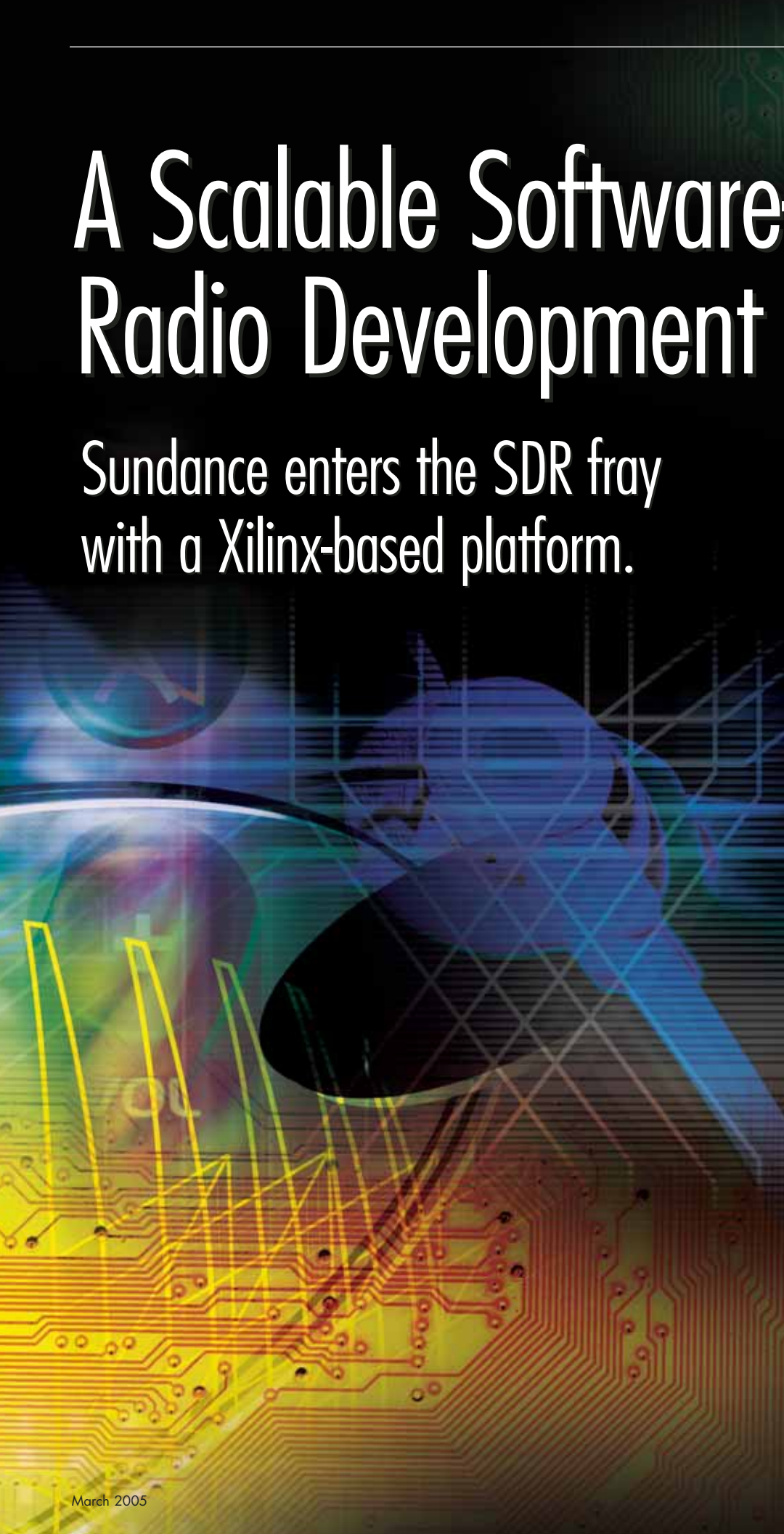
For more information, please visit *www.nohau.com*, *www.iq-service.com*, or e-mail *darrell@iq-service.com* or *darrellw@nohau.com*.

** 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.

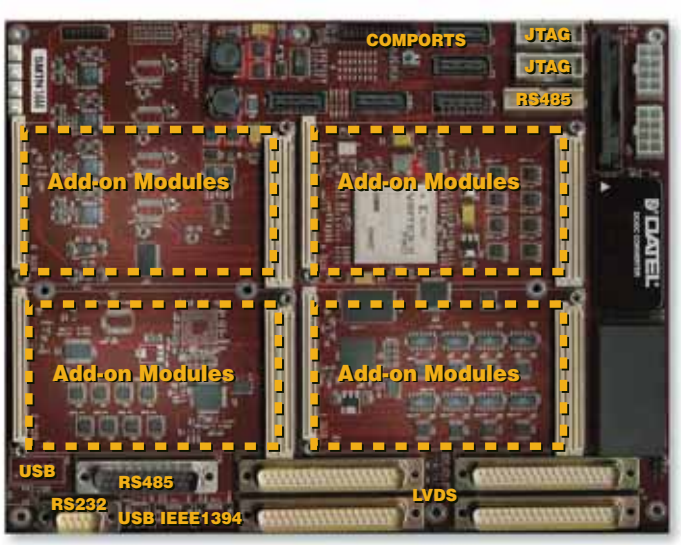


Figure 1 – The SMT148

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

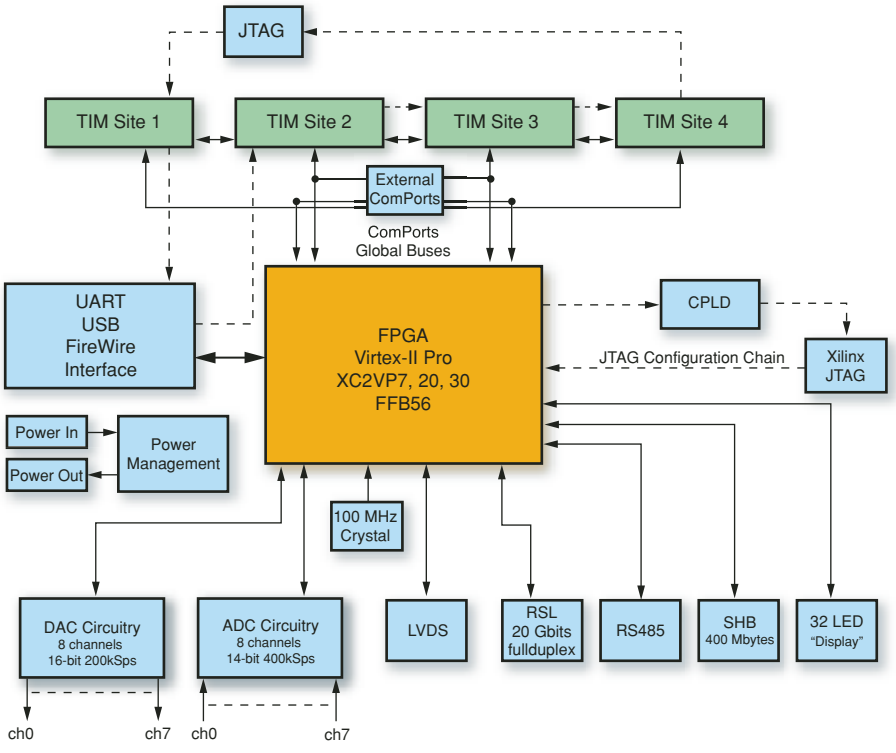


Figure 2 – SMT148 systems architecture

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

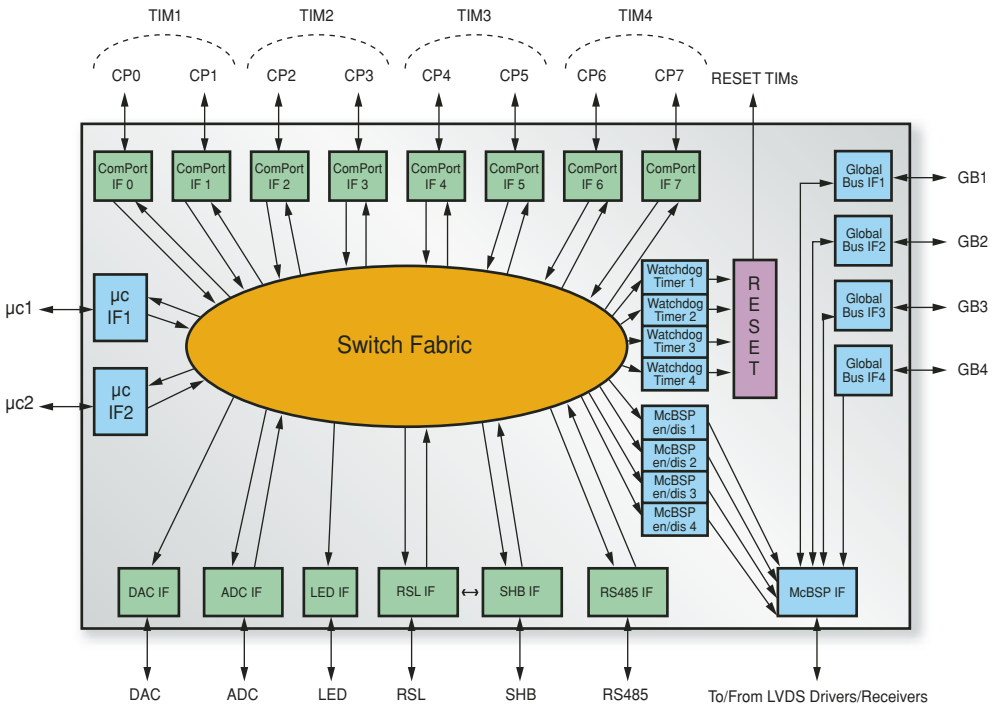


Figure 3 – Interconnections diagram of the digital modules inside the SMT148

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.

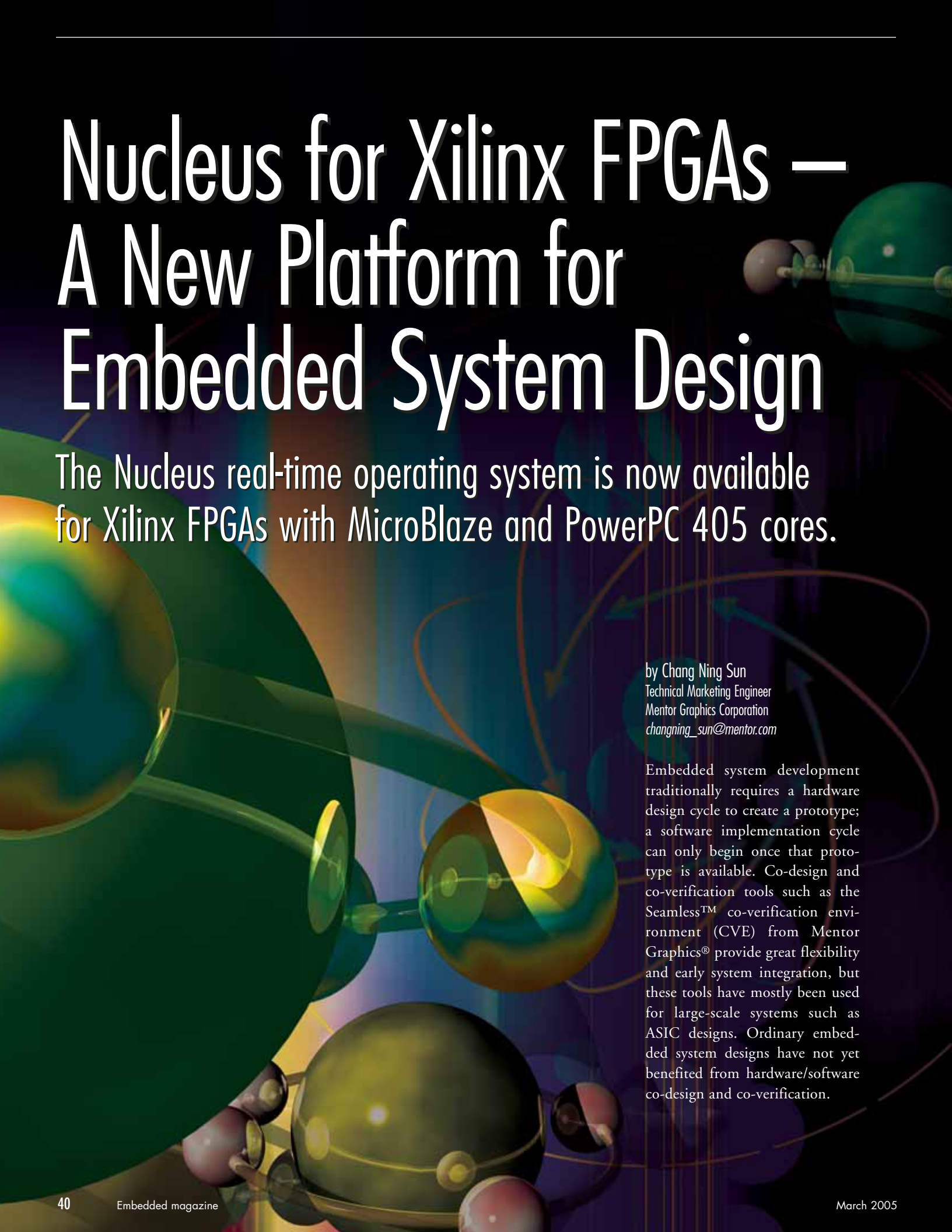
Conclusion

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.*

Nucleus for Xilinx FPGAs — A New Platform for Embedded System Design

The Nucleus real-time operating system is now available for Xilinx FPGAs with MicroBlaze and PowerPC 405 cores.



by Chang Ning Sun
Technical Marketing Engineer
Mentor Graphics Corporation
changning_sun@mentor.com

Embedded system development traditionally requires a hardware design cycle to create a prototype; a software implementation cycle can only begin once that prototype is available. Co-design and co-verification tools such as the Seamless™ co-verification environment (CVE) from Mentor Graphics® provide great flexibility and early system integration, but these tools have mostly been used for large-scale systems such as ASIC designs. Ordinary embedded system designs have not yet benefited from hardware/software co-design and co-verification.

In recent years, innovations in FPGA technology have shifted the use of these devices from supporting logic to forming a central part of the embedded system. With their high logic density and high performance, modern FPGAs allow you to implement almost an entire embedded hardware system in a single FPGA device.

Xilinx® Spartan-II™, Spartan-3™, and Virtex-II™ Platform FPGAs (combined with the MicroBlaze™ soft processor core) and Virtex-II Pro™ Platform FPGAs (combined with the PowerPC™ 405 hard processor core) offer new hardware platforms and facilitate new methodologies in embedded system design.

The Nucleus™ real-time operating system (RTOS) from Accelerated Technology (Mentor Graphics' Embedded Systems Division) fully supports Xilinx Platform FPGAs. Together with our FPGA design flow and Seamless CVE tools, the Nucleus RTOS is a complete hardware/software co-design and co-verification environment.

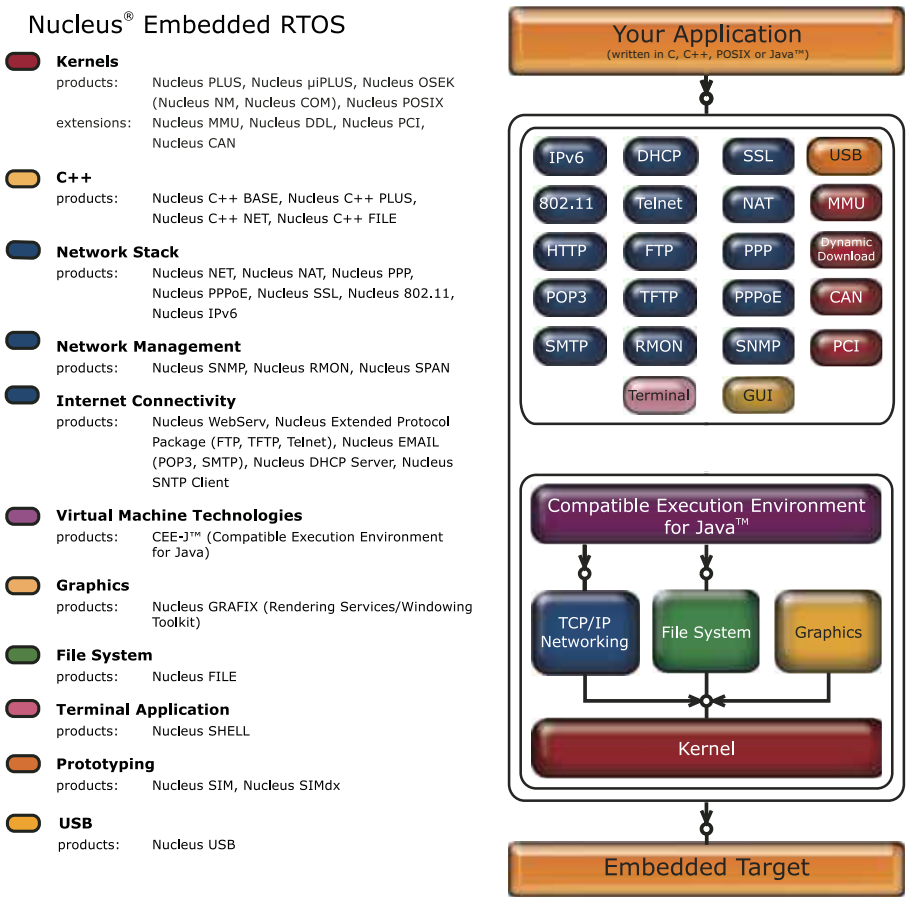


Figure 1 - Nucleus system architecture

The Nucleus RTOS

Very few embedded software developers start their development on bare hardware; most choose a commercial embedded RTOS like Nucleus as the base environment and develop their applications on top of the operating system.

Generally, an RTOS provides a multi-tasking kernel and middleware components such as a TCP/IP stack, file system, or USB stack. Application developers build their application software by using the system services provided by the RTOS kernel and the middleware components. A number of commercial RTOSs are available: some have hard real-time kernels, some have extensive middleware support, and some have good development tools. The Nucleus RTOS has all of these features from a single vendor.

Figure 1 shows the Nucleus system architecture.

Nucleus PLUS

Nucleus PLUS, a fully preemptive and scalable kernel suitable for hard real-time applications, is designed specifically for embedded systems. It has a very small footprint; the kernel itself can be as small as 15-20 kB.

All Nucleus kernel functions are provided as libraries, so only the required kernel functionality is linked with the application code in the final image. Nucleus PLUS provides complete multi-tasking kernel functions, including task management, inter-task communication, inter-task synchronization, memory management (MMU), and timer management. Recently, we added a dynamic download library (DDL) into Nucleus PLUS to allow dynamic downloading and allocation of system modules.

Nucleus Middleware

The middleware available for an RTOS has become an important factor when selecting a commercial RTOS for a particular application. As embedded applications become increasingly more complicated, application developers find implementing industry standards like TCP/IP, WiFi, USB, and MPEG more difficult. These standards are often implemented as middleware components and provided together with the RTOS.

The Nucleus RTOS has an extensive library of middleware support (Table 1), which covers every vertical market of the embedded industry. Nucleus NET utilizes a zero-copy mechanism for transferring data between user memory space and the TCP/IP stack, which significantly improves data transmission efficiency and reduces dynamic memory allocation.

Nucleus software fully supports the POSIX standard for software portability. POSIX interface libraries are available for all Nucleus middleware components.

Nucleus Device Drivers

The Nucleus RTOS has an extensive list of off-the-shelf device drivers. When you select a new hardware IP or peripheral to use in your design, the driver source code may already be available. If not, a driver template lets you easily develop a functional driver.

Nucleus software has proven to be a robust RTOS and has been widely used in every vertical market of the embedded industry…

Mentor Graphics will continue to add device drivers to the Nucleus library for new hardware IP blocks for Xilinx FPGAs as well as for third parties, such as its own IP Division.

Nucleus Development Tools

The Nucleus RTOS has the most integrated development environment (IDE) in the industry. Combined with our Microtec compiler tools, the codelab embedded development environment (EDE) and XRAY debuggers provide a complete software design flow, from compilation to debugging.

Our C/C++ source-level debuggers support both run-mode and freeze mode debugging with complete Nucleus kernel awareness. The XRAY debugger supports both homogeneous and heterogeneous multi-core debugging, and a wide range of processors and DSPs.

In addition to the Microtec compilers, the EDE, IDE, and debuggers are also integrated with many third-party compiler tools such as GNU.

For system-on-chip (SoC) users, integration of the Nucleus RTOS and XRAY debugger with other products in the Mentor Graphics lineup (such as Seamless CVE for hardware/software co-verification) offer a complete hardware/software co-design and co-verification platform.

Royalty-Free with Source Code

The royalty-free with source code business model of the Nucleus RTOS, combined with the high scalability and small footprint architecture, provides great value to customers developing large-volume products such as cell phones, PDAs, and cameras. A single up-front license fee covers all production rights for a single product using Nucleus software, with no additional royalty fees and no need to track shipment numbers.

Nucleus software has proven to be a robust RTOS and has been widely used in every vertical market of the embedded industry, including consumer electronics, telecom, defense/aerospace, automotive, and telematics.

Nucleus Product	Function
Nucleus NET	Complete embedded TCP/IP stack
Nucleus Extended Protocols	Telnet, FTP, TFTP, Embedded Shell
Nucleus SNMP version 1,2,3	Network management protocols
Nucleus RMON (1-9 groups)	Network remote monitoring protocols
Nucleus Residential Gateway	NET, PPP, PPPoE, DHCP server
Nucleus WebServ	Embedded HTTPD web server
Nucleus EMAIL	POP3 and SMTP
Nucleus FILE	MS DOS compatible file system
Nucleus GRAFIX	Graphics-rendering engine with windowing toolkit
Nucleus USB	Complete USB stack for USB specifications 1.1, 2.0, and OTG
Nucleus 802.11 STA (WiFi)	802.11b and 802.11g protocol stack
Nucleus IPv6	IP version 6 protocol stack
CEE-J Java VM for Nucleus	CLDC, MIDP, Embedded Java, and Personal Java

Table 1 - Primary Nucleus middleware components

Nucleus for Xilinx FPGAs

The scalable and configurable nature of Nucleus products, combined with the flexibility and versatility of FPGAs, provides great potential for hardware/software co-design, co-verification, and early integration of your software and hardware projects.

Nucleus for MicroBlaze

The MicroBlaze soft processor core features a Harvard-style RISC architecture with 32-bit instruction and data buses. A MicroBlaze soft core can be programmed into Spartan-II, Spartan-3, or Virtex-II Platform FPGAs.

Nucleus software fully supports the Xilinx MicroBlaze soft processor core with GNU compiler tool and GDB debugger.

Nucleus for Virtex-II Pro FPGAs/PowerPC 405

Many embedded system developers are already familiar with the PowerPC 405 processor. Xilinx Virtex-II Pro FPGAs provide fully configurable hardware platforms with the PowerPC 405 hard processor core.

Nucleus software fully supports Virtex-II Pro FPGAs with the PowerPC 405 hard core, providing a Microtec PowerPC compiler and XRAY kernel-aware debugger, as well as GNU compiler tools and GDB debugger.

FPGA Reference Design for Nucleus Software

To give embedded system developers a quick start with Nucleus software for Xilinx FPGAs, we worked with Xilinx and Memec™ Insight to develop reference designs for running Nucleus software. Each reference design is a sample FPGA hardware configuration based on a Memec Insight FPGA development board.

You can build and implement all of the reference designs into the FPGA device by

using Xilinx EDK and ISE software. For each reference design, we provide a Limited Version (LV) of our Nucleus software, including the Nucleus PLUS kernel and the relevant middleware (such as Nucleus NET) to help you evaluate Nucleus software with the FPGA.

The LV version of Nucleus software functions exactly the same as a full version, but is provided as binary only and runs for a limited time. Currently, we have FPGA reference designs for the following Memec Insight boards:

- Spartan-IIE LC MicroBlaze development board
- Spartan-3 LC MicroBlaze development board
- Spartan-3 MB MicroBlaze development board
- Virtex-II MB MicroBlaze development board
- Virtex-II Pro PowerPC development board

We have also created a website (*www.acceleratedtechnology.com/xilinx*) to support Nucleus software for Xilinx FPGAs, where you can download FPGA reference designs and Nucleus software updates.

FPGA Design Flow

The FPGA reference designs for the Nucleus RTOS can be used as your starting point to build a custom FPGA-based hardware platform. All of the reference designs are provided as Xilinx Platform Studio (XPS) projects, so you can directly open these projects in XPS and edit, build, and implement the design with Xilinx EDK and ISE software. Typically, building a Nucleus system using an FPGA and configurable cores involves the following:

- Configuring a hardware platform with FPGA, processor core, hardware IP, and memory
- Building the design with EDK/ISE software to generate an FPGA bitstream
- Building software libraries with EDK to generate low-level device driver code
- Developing Nucleus device drivers and Nucleus applications
- Programming the FPGA device
- Downloading Nucleus into the FPGA hardware board

Once you have reached this point, you can start evaluating your Nucleus software and begin debugging your application.

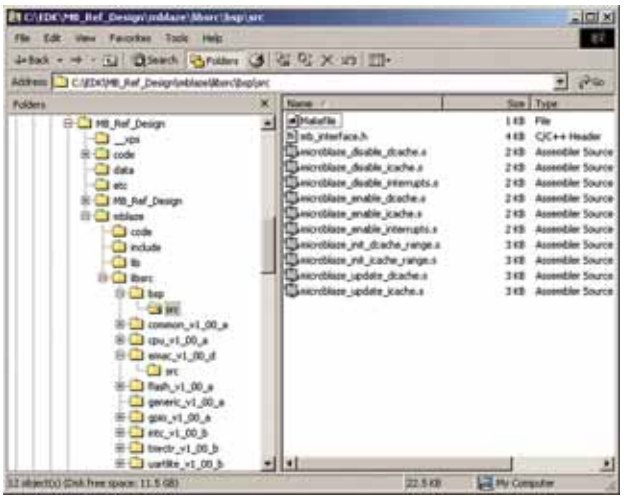


Figure 2 - Xilinx EDK-generated source code directories

Xilinx EDK and ISE software provide a complete environment for configuring and implementing an FPGA-based hardware platform. At the same time, EDK also generates software libraries and C header files for the hardware design. These libraries provide basic processor boot-up code and low-level hardware IP device driver function code. These low-level functions can be treated as a "BIOS" layer for Nucleus.

Figure 2 shows the source code directories generated by EDK in an XPS project. You will find one subdirectory for each piece of hardware IP, such as "emac_v1_00_d" for Ethernet and "uartlite_v1_00_b" for UART. Each subdirectory contains the driver code for that hardware IP. These driver functions are included with the hardware IP from the vendor and have been integrated into the EDK installation.

Nucleus software provides an OS adaptation layer that is an interface layer between the EDK-generated "BIOS" function layer and the Nucleus high-level device drivers. This OS adaptation layer serves as a hardware abstraction layer, which will significantly simplify the Nucleus device driver porting over FPGA-based hardware platforms.

Nucleus Software Debugging

Both MicroBlaze soft-core and PowerPC 405 hard-core designs provide a standard JTAG debug interface. So after implementing the hardware design in the FPGA, you can take full advantage of our JTAG-based codellab EDE or XRAY debugger to facilitate application debugging. Both codellab and XRAY have complete Nucleus kernel awareness, which can help you analyze complex real-time behaviors in a multi-tasking system.

Co-Design and Co-Verification

As we mentioned earlier, Nucleus software provides a viable platform for hardware/software co-design and co-verification. As one of the only EDA companies with an embedded software focus, Mentor Graphics' Nucleus RTOS and XRAY debugger are fully integrated with our Seamless hardware/software co-verification software platform.

The combination of the Nucleus RTOS and Seamless co-verification tool provides you with one of the most comprehensive hardware/software co-design and co-verification tools possible.

Conclusion

Embedded systems are becoming increasingly complicated. To meet the challenge, integrated solutions including hardware/software co-design and co-verification are required. FPGAs with configurable processor cores bring a new, innovative approach to modern embedded system designs and open the door to hardware/software co-design and co-verification.

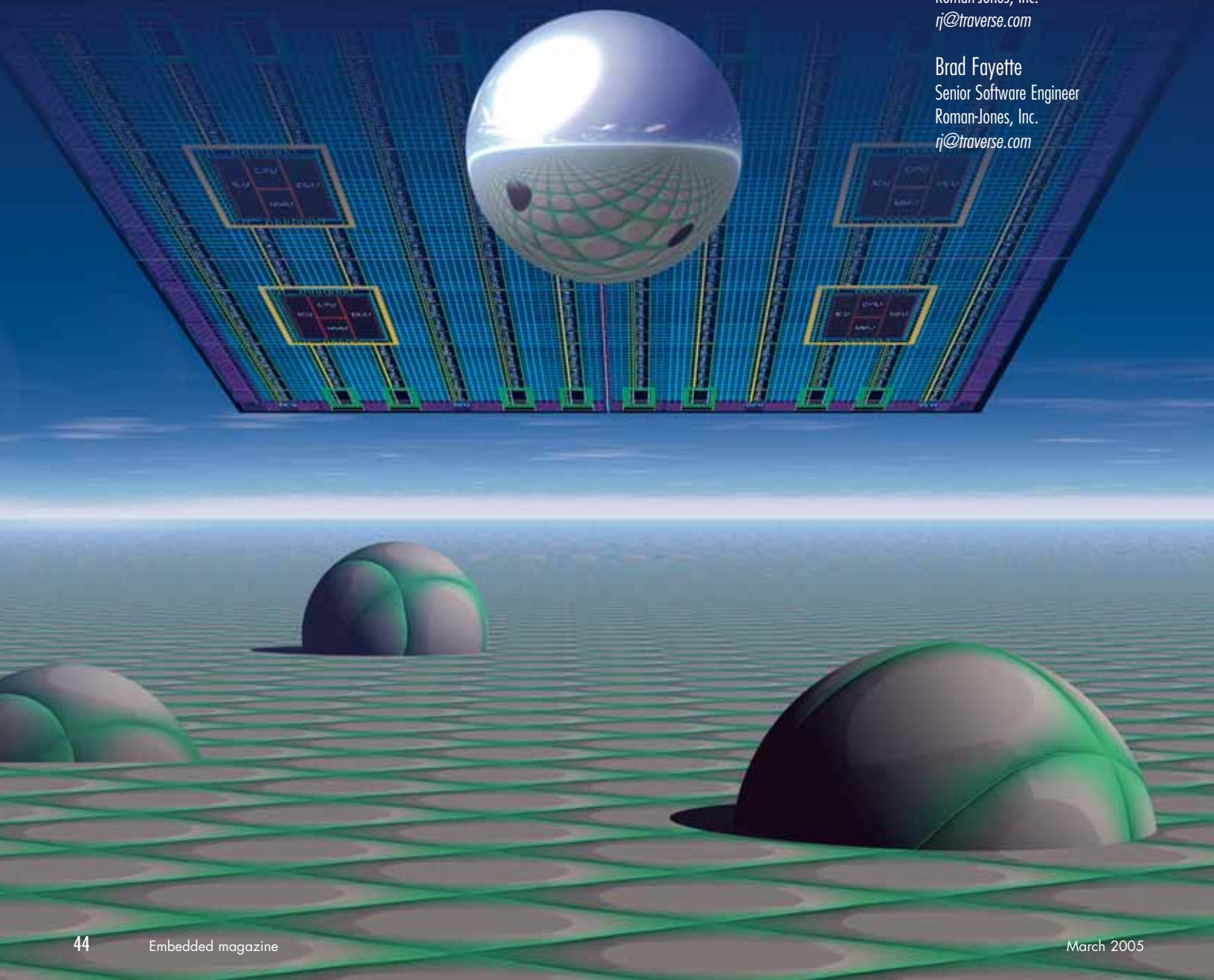
Nucleus software for FPGAs can significantly accelerate your hardware/software development cycle and improve your hardware/software integration quality. For more information, please visit *www.acceleratedtechnology.com.*

Emulate 8051 Microprocessor in PicoBlaze IP Core

Put the functions of a legacy microprocessor into a Xilinx FPGA.

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

Brad Fayette
Senior Software Engineer
Roman-Jones, Inc.
rj@traverse.com



How do you put a one-dollar Intel™ 8051 microprocessor into an FPGA without using 10 dollars' worth of FPGA fabric?

The answer is emulation. Using software emulation, Roman-Jones Inc. has developed a new type of 8051 processor core built on a Xilinx 8-bit, soft-core PicoBlaze™ (PB) processor. This "new" PB8051 is more than 70% smaller than competing soft-core implementations – without sacrificing any of the performance of this legacy part. The PB8051 is a Xilinx AllianceCORE™ microprocessor built through emulation.

The Legacy of the 8051

The Intel 8051 family of microprocessors – probably one of the most popular architectures around – is still the core of many embedded applications. This processor just refuses to retire. Many designers are using legacy code from previous projects, while others are actually writing new code.

The 8051 architecture was designed for ASIC fabric. It is not efficient in an FPGA, resulting in excess logic usage with marginal performance.

FPGA microprocessor integration is a solution for older 8051 products undergoing redesign to eliminate obsolesce, lower costs, decrease component count, and increase overall performance.

The new FPGA-embedded PB8051 designs allow you to take advantage of existing in-house software tools and your own architecture familiarity to quickly implement a finished design. The integrated PB8051 can be customized on the FPGA to exact requirements.

Processor Emulation

Programmers have used microprocessor emulation for many years as a software development vehicle. It allows programmers to write and test code on a development platform before testing on target hardware.

This same concept can be practical when the target microprocessor architecture does not lend itself to efficient implementation and use of FPGA resources.

The features of our PB8051 emulated processor include:

- Smaller Size – Traditional 8051 implementations range from 1,100 to 1,600 slices of FPGA logic. The PicoBlaze 8051 processor requires just 76 slices. Emulation hardware requires 77 slices. Add another 158 slices for two timers and a four-mode serial port, and you have a total of a 311 slices. This is a reduction of more than two-thirds of the FPGA fabric of competing products.
- Faster – At 1.3 million instructions per second (MIPS), the PB8051 is faster than a legacy 8051 (1 MIPS) running at 12 MHz. Compare this to 5 MIPS with a 40 MHz Dallas version or 8 MIPS with a traditional FPGA implementation – which takes up more than three times as much FPGA fabric as the PB8051. The PicoBlaze processor itself runs at a remarkable 40 MIPS using an 80 MHz clock.
- Software Friendly – You can write C code or assembler code with your present software development tools to generate programs. You can also run legacy objects out of a 27C512 EPROM.
- Easy Hardware – You can use VHDL or Verilog™ hardware description languages to instantiate the 8051-type core. Hook up block RAM or off-chip program ROM, and you're ready to go.
- Low Cost – The PB8051 is $495 with an easy Xilinx SignOnce IP license.

> This "new" PB8051 is more than 70% smaller than competing soft-core implementations – without sacrificing any of the performance of this legacy part.

Architecture of Emulated 8051

As shown in Figure 1, the architecture of an emulated processor has several elements. Each element is designed independently, but together, they act as a whole.

PicoBlaze Platform

The PicoBlaze host processor is the heart of the emulated system and defines the architecture of the PB8051 emulated processor core. It comprises:

- PicoBlaze Code ROM – Block ROM contains the software code to emulate 8051 instructions.
- Internal Address/Bus – The PicoBlaze peripheral bus is a 256-byte address space accessed by PicoBlaze I/O instructions. It allows the PicoBlaze processor to interface with RAM, emulation peripherals, timers, and the serial port. This bus is internal to the core and thus hidden from the user.
- Emulation Peripherals – Not all emulation tasks can be done in software and still meet the performance requirements for your particular application. Specialized hardware assists the PicoBlaze code to perform tasks that are time-consuming, on a critical execution path, or both.

A good example is the parity bit in the PSW (program status word) register that reflects 8051 accumulator parity. This function must be performed for every 8051 instruction executed, and would take several PicoBlaze instruction cycles to perform. It is, therefore, done in hardware.

- Instruction Decode ROM – This is a key emulation peripheral that is used to decode 8051 instructions to set PicoBlaze routine locations and emulation parameters.

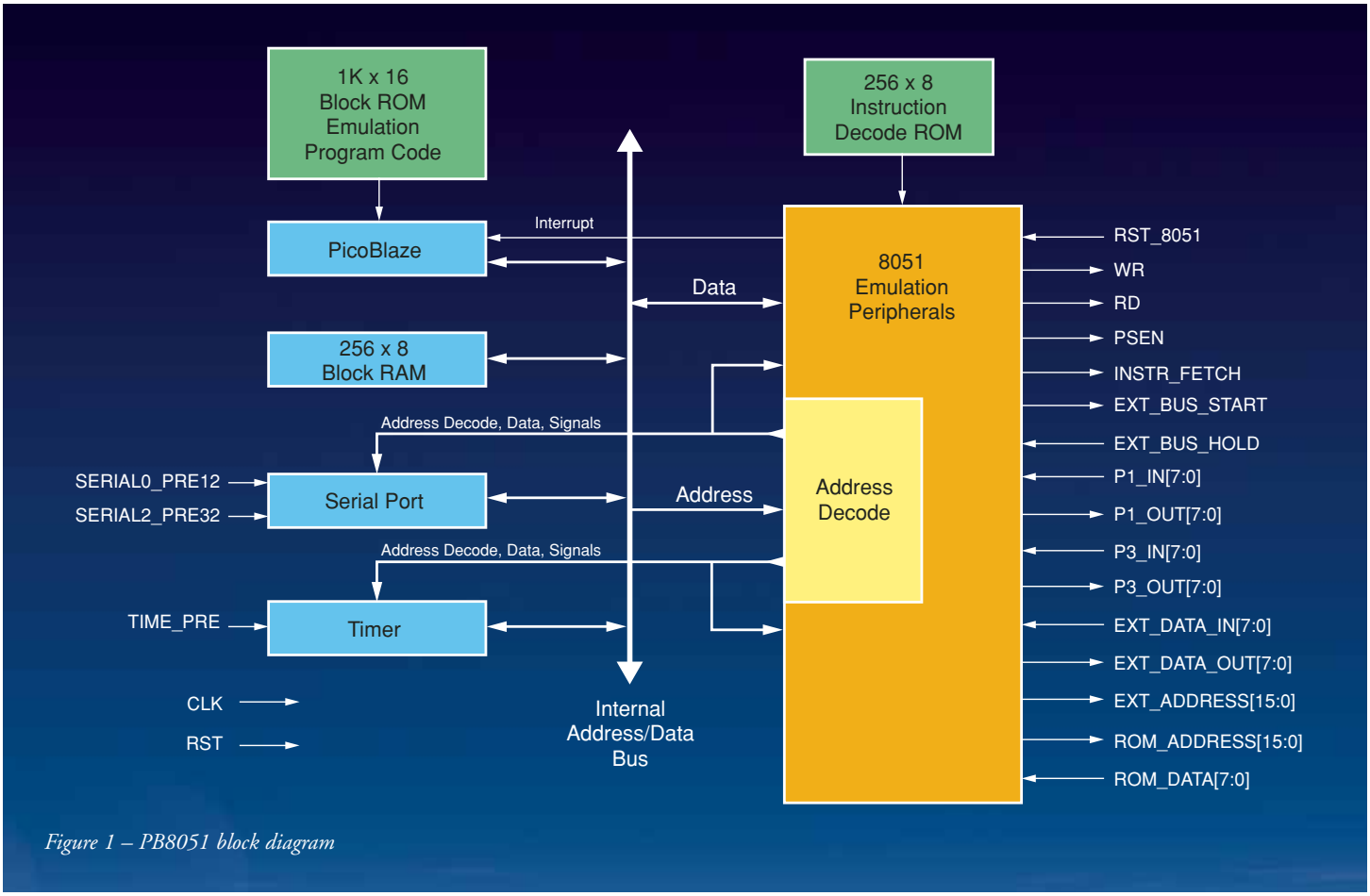


Figure 1 – PB8051 block diagram

- Address Decode – A significant amount of the emulation peripheral hardware is dedicated to simple address decode of the PicoBlaze peripheral bus. This address space is for the PicoBlaze processor only and is insulated from the 8051 application.
- Block RAM – 256 bytes are available to 8051 internal RAM and some 8051 registers. You can access this block RAM via 8051 instructions.
- Serial Port and Timer – The actual 8051 timer and multi-mode serial port were best done in hardware instead of trying to implement these functions in software. Clock prescaling inputs are provided so that these functions can run at a clock rate independent of the emulated system.

PicoBlaze Emulation Software

A 1K x 16 block ROM holds the PicoBlaze code that performs the actual emulation. The emulation program is carefully constructed in very tight PicoBlaze assembler code, optimized for speed and efficiency.

The emulation program is divided into several segments:

- Instruction Fetch – An 8051 bus cycle is simulated to fetch the next instruction from 8051 program memory (64 Kb size), which may be on-chip block RAM or off-chip EPROM (such as the 27C256).
- Instruction Decode – Fetched instructions are decoded to determine the addressing mode and operation.
- Fetch Operands – Depending upon the addressing mode, additional operands are fetched for the instruction from program memory, internal RAM, or external RAM.
- Instruction Execution – An instruction performs the desired operation and updates emulated register contents, including affected 8051 PSW flags.
- Scan Interrupts – This function determines if an interrupt is pending, and if so, services it. An interrupt window is generated at the end of every emulated instruction when required.

In addition to PicoBlaze code, Java™ software utilities process symbols taken from PicoBlaze listings (.LOG files) into a ROM table. These .LOG files are used to decode 8051 opcodes.

This program also produces the .COE files used by the Xilinx CORE Generator™ system to create PicoBlaze code ROM. All of this is transparent to designers integrating with the PB8051.

User Back-End Interface

What gives the PB8051 its "hardware flavor" is the user back-end interface, where you interface your logic design with the emulated 8051 processor. The back-end interface is part of the emulation peripherals, controlled by the PicoBlaze platform.

What gives the PB8051 its "hardware flavor" is the user back-end interface, where you interface your logic design with the emulated 8051 processor.

Just as the 8051 processor family has many derivatives to define port, functionality, features, and pinout, the back-end interface serves the same function. The PB8051 has a back-end interface that resembles the generic 8031, the ROM-less version of the 8051.

Roman-Jones Inc. customizes back-end interfaces to meet your exact 8051 needs, such as removing an unused serial port or adding an I^2C port to emulate the 80C652 derivative.

Designing with the PB8051

Incorporating the PB8051 into the rest of your design is easy, because it comes with reference designs and examples of Xilinx integrated software design (ISE) projects.

Hardware Considerations

Figure 1 illustrates the signal names available on the user back-end interface. A VHDL or Verilog template provides the exact signal names – many of which are already familiar to 8051 designers. A few new types of signals exist, including:

- Pre-scales used by the timer and serial port to set counting and baud rates.
- One-clock-wide read/write strobes to read and write external memory space. There is also a program store enable (PSEN) strobe for the 8051 code memory.
- Bus start and hold signals used to insert wait states for external memory cycles that cannot be completed in one system clock cycle.

The PB8051 is instantiated as a component into your top-level design. You determine if the 8051 program resides in on-chip block RAM or off-chip EPROM. Peripherals can be hooked up to either the P1/P3 port lines or to the external address and data buses. For convenience, the address and data lines for program and data memory spaces are separated, so conventional multiplexer circuitry is not needed.

If your entire design, including the 8051 program, resides on the FPGA, simply set the EXT_BUS_HOLD to "low" to take full advantage of running at clock speed. If you elect to use an off-chip EPROM or have slow peripherals, wait states can be inserted by asserting EXT_BUS_HOLD at "high." One of our reference designs illustrates wait state generation.

Xilinx Implementation Considerations

There are few implementation considerations other than the need to place the PB8051 design netlist into your project directory and instantiate it as a component in your VHDL or Verilog design; ISE software will do the rest. ISE schematic capture is also supported.

Simulation Considerations

We've included a behavior simulation model of the PB8051 for adding (along with the rest of your design) to your favorite simulator. Modeltech and Aldec™ simulators have been tested for correct operation. Post place-and-route or timing simulations follow a conventional design flow.

You will enjoy watching your 8051 instruction execution flow go by on the simulation waveforms. This makes behavioral debugging straightforward, fast, and easy. We've provided a reference test bench with example waveform files.

8051 Software Considerations

To generate your 8051 programs the way you always have, use your favorite C compiler or assembler and linker (if necessary) to produce the same Intel hex format file that you would use to burn an EPROM. You can even use an existing hex file, because the PB8051 looks like a regular 8051 as far as software code is concerned. An Intel hex to .COE utility is included for those designs that put the 8051 software into on-chip block RAM. On-chip program storage provides maximum speed performance.

Test and Debug Considerations

We recommend you use design tools to quickly and easily test and debug your design. The most useful tool will be an HDL simulator, such as Modeltech or Aldec programs. Most problems and bugs can be solved at the behavioral level. For interactive debugging, the Xilinx ChipScope™ integrated logic analyzer has proved to be the tool of choice. At the current time, no source code debugger tools are available for the PB8051.

Designer's Learning Curve

Designers should have some experience in 8051 hardware/software and FPGA design before attempting to consolidate the two. The PB8051 core is designed for ease of use and integration.

Your 8051 hardware and software expertise should include hardware understanding of the part and experience in writing 8051 code using software development tools. The basic design flow using the PB8051 is identical to the normal packaged processor flow.

Integrating the PB8051 onto the Xilinx part is much the same as instantiating a core using the Xilinx CORE Generator tool. If you are familiar with VHDL or Verilog language, and have a couple of Xilinx designs under your belt, you're good to go.

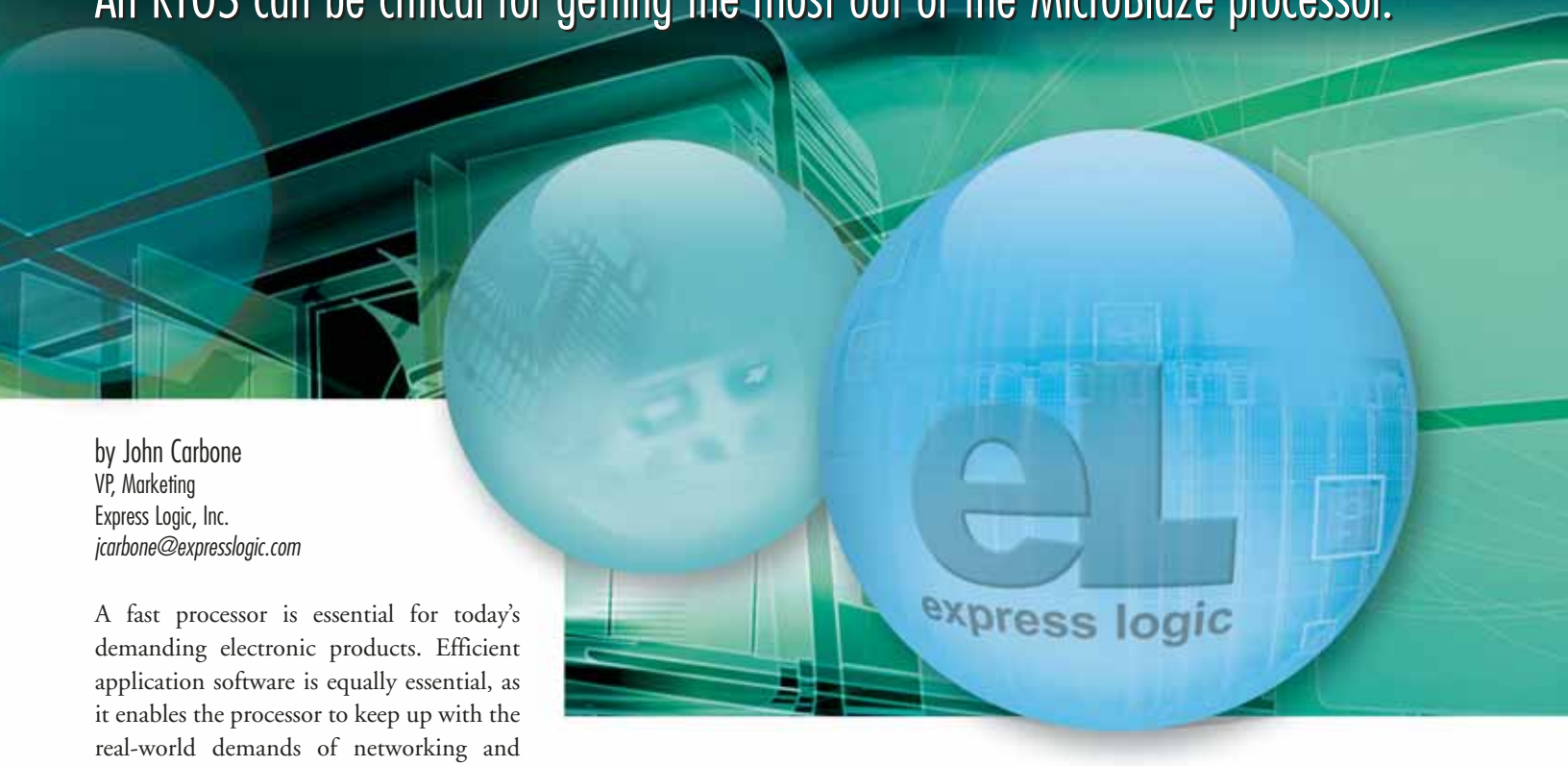
Conclusion

Integrating microprocessors onto FPGAs through emulation, as illustrated with the PB8051, is a viable alternative to a full hardware functional design. The advantage of FPGA fabric savings correlates to reduced parts cost.

Integrating the PB8051 processor into your design yields lower component count, easier board debugging, less noise, and optimum performance of the peripheral/8051 micro-interface, because both are in an FPGA. For more information about the PB8051 microcontroller, visit *www.roman-jones.com/rj2/PB8051Microcontroller.htm.*

Optimize MicroBlaze Processors for Consumer Electronics Products

An RTOS can be critical for getting the most out of the MicroBlaze processor.



by John Carbone
VP, Marketing
Express Logic, Inc.
jcarbone@expresslogic.com

A fast processor is essential for today's demanding electronic products. Efficient application software is equally essential, as it enables the processor to keep up with the real-world demands of networking and consumer products.

But what about the real-time operating system (RTOS) that handles your system interrupts and schedules your application software's multiple threads?

If it's not optimized for the processor you're using, and isn't efficient in its handling of external events, you'll end up with only half a processor to do useful work.

An efficient RTOS should do more than just handle multiple threads and service interrupts. It must also have a small footprint and be able to deliver the full power of the processor used by your application software. Express Logic's ThreadX® real-time operating system is just such an RTOS.

The ThreadX RTOS has now been optimized to support the Xilinx MicroBlaze™ soft processor, and is available off-the-shelf for your next development project.

A Good Fit

Let's look at some of the reasons why the ThreadX RTOS is a good fit with the MicroBlaze processor.

Size

The ThreadX RTOS is only about 6 KB - 12 KB in total size for a complete multitasking system. This includes basic services, queues, event flags, semaphores, and memory allocation services.

Its small size means that more of your memory resources will be preserved for use by the application and not absorbed by the RTOS. This will result in smaller memory requirements, lower costs, and lower power consumption. Table 1 shows code sizes for various optional components of the ThreadX RTOS on the MicroBlaze processor.

Efficiency

The ThreadX RTOS can perform a full context switch between active threads in a mere 1.2 µs on a 100 MHz Xilinx Virtex-II Pro™ FPGA. This is particularly critical in applications with high interrupt incidences, such as network packet processing.

The ability of the ThreadX RTOS to handle interrupts efficiently means that your system can handle higher rates of external events such as TCP/IP packet arrivals, delivering greater throughput.

Speed

The ThreadX RTOS responds to interrupts and schedules required processing in less than 1 microsecond. The single-level interrupt processing architecture of the ThreadX RTOS eliminates excess overhead found in many other RTOSs.

This keeps the ThreadX RTOS from gobbling up valuable processor cycles with housekeeping tasks, thus leaving more processor bandwidth for your application. Thus, you can configure a less expensive processor or add more features to your application without increasing processor speed and cost.

Table 2 shows execution times for various ThreadX services measured according to the following definitions:

- **Immediate Response** (IR): Time required to process the request immediately, with no thread suspension or thread resumption.
- **Thread Suspend** (TS): Time required to process the request when the calling thread is suspended because the resource is unavailable.
- **Thread Resumed** (TR): Time required to process the request when a previously suspended thread (of the same or lower priority) is resumed because of the request.
- **Thread Resumed and Context Switched** (TRCS): Time required to process the request when a previously suspended thread with a higher priority is resumed because of the request. As the resumed thread has a higher priority, a context switch to the resumed thread is also performed from within the request.
- **Context Switch** (CS): Time required to save the current thread's context, find the highest priority ready thread, and restore its context.
- **Interrupt Latency Range** (ILR): Time duration of disabled interrupts.

Ease of Use

All services are available as function calls, and full source code is provided so you can see exactly what the ThreadX RTOS is doing at any point.

Many other commercial RTOS products are massive collections of unintelligible system calls with non-intuitive names. The ThreadX RTOS uses service call names that are orderly, simple in structure, and ultimately easier to use. This enables shorter development times and faster time to market.

Fast Interrupt Handling

The ThreadX RTOS is optimized for fast interrupt handling, saving only scratch registers at the beginning of an interrupt service routine (ISR), enabling use of all services from the ISR, and using the system stack in the ISR.

ThreadX Instruction Area Size (in bytes)	
Core Services (required)	4,804
Queue Services	3,200
Event Flag Services	1,720
Semaphore Services	932
Mutex Services	2,540
Block Memory Services	1,144
Byte Memory Services	1,876

Table 1 – Code size of various ThreadX services

ThreadX Service Call Execution Times				
Service	**IR**	**TS**	**TR**	**TRCS**
tx_thread_suspend	1.0 µs	2.0 µs		
tx_thread_resume			1.0 µs	2.8 µs
tx_thread_relinquish	0.5 µs			1.9 µs
tx_queue_send	0.9 µs	2.4 µs	1.6 µs	3.4 µs
tx_queue_receive	0.9 µs	2.3 µs	2.5 µs	4.2 µs
tx_semaphore_get	0.3 µs	2.3 µs		
tx_semaphore_put	0.3 µs		1.3 µs	3.0 µs
tx_mutex_get	0.4 µs	2.4 µs		
tx_mutex_put	1.0 µs		2.0 µs	3.6 µs
tx_event_flags_set	0.6 µs		1.7 µs	3.4 µs
tx_event_flags_get	0.4 µs	2.5 µs		
tx_block_allocate	0.4 µs	2.3 µs		
tx_block_release	0.4 µs		1.4 µs	3.1 µs
tx_byte_allocate	1.4 µs	2.9 µs		
tx_byte_release	0.9 µs		3.0 µs	4.7 µs
Context Switch (CS)	1.2 µs			
Interrupt Latency Range (ILR)	0.0 µs-0.8 µs			

Table 2 – Execution times of various ThreadX services

Efficient interrupt processing delivers the best performance from MicroBlaze processors, giving your product a boost over the competition.

Full Source Code

By providing full source code, you have complete visibility into all ThreadX services – no more mysteries are hidden within the RTOS "black box."

This full view of source code enables you to avoid delays in development caused by an incomplete understanding of RTOS services. With the ThreadX RTOS, full source code is there any time you need it.

Conclusion

Express Logic's ThreadX RTOS is well matched for networking and consumer applications based on MicroBlaze processors. It's been optimized to deliver the processor's inherent performance directly through to the application.

What's next? Express Logic is working to port NetX™, its TCP/IP network stack, to MicroBlaze. With so many of today's consumer electronics devices accessing the Internet, efficient TCP/IP software is increasingly critical. The NetX TCP/IP stack will provide further support for MicroBlaze in consumer electronics products, and will enable developers to concentrate on their application code, getting their product to market faster.

To develop a successful, high-performance consumer, networking, or other electronic product around MicroBlaze processors, use an efficient RTOS like ThreadX by Express Logic. For further information on the ThreadX RTOS and other embedded software products from Express Logic, please visit *www.expresslogic.com*.

Use an RTOS on Your Next MicroBlaze-Based Product

An RTOS called µC/OS-II has been ported to the MicroBlaze soft-core processor.

by Jean J. Labrosse
President
Micrium, Inc.
Jean.Labrosse@Micrium.com

Designing software for a microprocessor-based application can be a challenging undertaking. To reduce the risk and complexity of such a project, it's always important to break the problem into small pieces, or tasks. Each task is therefore responsible for a certain aspect of the application. Keyboard scanning, operator interfaces, display, protocol stacks, reading sensors, performing control functions, updating outputs, and data logging are all examples of tasks that a microprocessor can perform.

In many embedded applications, these tasks are simply executed sequentially by the microprocessor in a giant infinite loop. Unfortunately, these types of systems do not provide the level of responsiveness required by a growing number of real-time applications because the code executes in sequence. Thus, all tasks have virtually the same priority.

For this and other reasons, consider the use of a real-time operating system (RTOS) for your next Xilinx MicroBlaze™ processor-based product. This article introduces you to a low-cost, high-performance, and high-quality RTOS from Micrium called µC/OS-II.

What is µC/OS-II?

µC/OS-II is a highly portable, ROM-able, scalable, preemptive RTOS. The source code for µC/OS-II contains about 5,500 lines of clean ANSI C code. It is included on a CD that accompanies a book fully describing the inner workings of µC/OS-II. The book is called *MicroC/OS-II, The Real-Time Kernel* (ISBN 1-5782-0103-9) and was written by the author (Figure 2). The book also contains more than 50 pages of RTOS basics.

Although the source code for µC/OS-II is provided with the book, µC/OS-II is not freeware, nor is it open source. In fact, you need to license µC/OS-II to use it in actual products.

Micrium's application note AN-1013 provides the complete details about the porting of µC/OS-II to the MicroBlaze soft-core processor. This application note is available from the Micrium website at *www.micrium.com*.

µC/OS-II was designed specifically for embedded applications and thus has a very small footprint. In fact, the footprint for µC/OS-II can be scaled based on which µC/OS-II services you need in your application. On the MicroBlaze processor, µC/OS-II can be scaled (as shown in Table 1) and can easily fit into Xilinx FPGA block RAM.

µC/OS-II is a fully preemptive real-time kernel. This means that µC/OS-II always attempts to run the highest-priority task that is ready to run. Interrupts can suspend the execution of a task and, if a higher-priority task is awakened as a result of the interrupt, that task will run as soon as the interrupt service routines (ISR) are completed.

µC/OS-II provides a number of system services, such as task and time management, semaphores, mutual exclusion semaphores, event flags, message mailboxes, message queues, and fixed-sized memory partitions. In all, µC/OS-II provides more than 65 functions you can call from your application. µC/OS-II can manage as many as 63 application tasks that could result in quite complex systems.

µC/OS-II is used in hundreds of products from companies worldwide. µC/OS-II is also very popular among colleges and universities, and is the foundation for many courses about real-time systems.

	Code (ROM)	Data (RAM)
Minimum Size	5.5 Kb	1.25 Kb (excluding task stacks)
Maximum Size	24 Kb	9 Kb (excluding task stacks)

Table 1 – Memory footprint for µC/OS-II on the MicroBlaze processor

RTOS Basics with µC/OS-II

When you use an RTOS, very little has to change in the way you design your product; you still need to break your application into tasks. An RTOS is software that manages and prioritizes these tasks based on your input. The RTOS takes the most important task that is ready to run and executes it on the microprocessor.

With most RTOSs, a task is an infinite loop, as shown here:

```
void MyTask (void)
{
  while (1) {
    // Do something (Your code)
    // Wait for an event, either:
    //  Time to expire
    //  A signal from an ISR
    //  A message from another task
    //  Other
  }
}
```

You're probably wondering how it's possible to have multiple infinite loops on a single processor, as well as how the CPU goes from one infinite loop to the next. The RTOS takes care of that for you by suspending and resuming tasks based on events. To have the RTOS manage these tasks, you need to provide at least the following information to an RTOS:

- The starting address of the task (MyTask())
- The task priority based on the importance you give to the task
- The amount of stack space needed by the task.

With µC/OS-II, you provide this information by calling a function to "create a task" (OSTaskCreate()). µC/OS-II maintains a list of all tasks that have been created and organizes them by priority. The task with the highest priority gets to execute its code on the MicroBlaze processor. You determine how much stack space each task gets based on the amount of space needed by the task for local variables, function calls, and ISRs.

In the task body, within the infinite loop, you need to include calls to RTOS functions so that your task waits for some event. One task can ask the RTOS to "run me every 100 milliseconds." Another task can say, "I want to wait for a character to be received on a serial port." Yet another task can say, "I want to wait for an analog-to-digital conversion to complete."

Events are typically generated by your application code, either from ISRs or issued by other tasks. In other words, an ISR or another task can send a message to a task to wake that task up. With µC/OS-II, a task waits for an event using a "pend" call. An event is signaled using a "post" call, as shown in the pseudo-code below:

```
void EthernetRxISR(void)
{
  Get buffer to store received packet;
  Copy NIC packet to buffer;
  Call OSQPost() to send packet to task
  (this signals the task);
}

void EthernetRxTask (void)
{
  while (1) {
    Wait for packet by calling OSQPend();
    Process the packet received;
  }
}
```

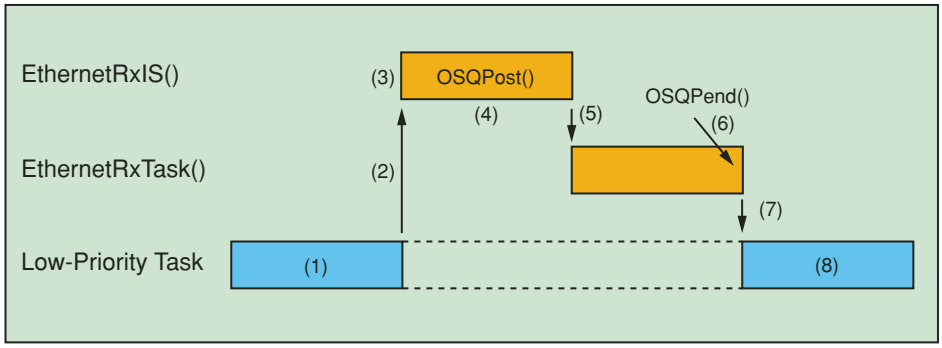


Figure 1 – Execution profile of an ISR and µC/OS-II

The execution profile for these two functions is shown in Figure 1.

Let's assume that a low-priority task is currently executing (1). When an Ethernet packet is received on the Network Interface Card (NIC), an interrupt is generated. The MicroBlaze processor suspends execution of the current task (the low-priority task) and vectors to the ISR handler (2).

With µC/OS-II, the ISR starts by saving all of the MicroBlaze registers, taking a mere 300 nanoseconds at 150 MHz. The ISR handler then needs to determine the source of the interrupt and execute the appropriate function (in this case, EthernetRxISR()) (3).

This ISR obtains a buffer large enough to hold the packet received by the NIC. It copies the contents of the packet received into the buffer and sends the address of this buffer to the task responsible for handling received packets (EthernetRxTask() in our example) using µC/OS-II's OSQPost() (4).

When this operation is completed, the ISR invokes µC/OS-II (calls a function) and µC/OS-II determines that a higher-priority task needs to execute. µC/OS-II then resumes the more important task and doesn't return to the interrupted task (5). With µC/OS-II on the MicroBlaze processor, this takes about 750 nanoseconds at 150 MHz.

When the packet is processed, EthernetRxTask() calls OSQPend() (6). µC/OS-II notices that there are no more packets to process and suspends execution of this task by saving the contents of all MicroBlaze registers onto that task's stack. Note that when the task is suspended, it doesn't consume any processing time.

µC/OS-II then locates the next most important task that's ready to run (the "Low-Priority Task" in Figure 1) and switches to that task by loading the MicroBlaze registers with the context saved by the ISR (7). This is called a *context switch*. µC/OS-II takes less than 1 microsecond at 150 MHz on the MicroBlaze processor to complete this process. The interrupted task is resumed exactly as if it was never interrupted (8).

The general rule for an application using an RTOS is to put most of your code into tasks and very little code into ISRs. In fact, you always want to keep your ISRs as short as possible. With an RTOS, your system is easily expanded by simply adding more tasks. Adding lower-priority tasks to your application will not affect the responsiveness of higher-priority tasks.

Safety-Critical Systems Certified

µC/OS-II has been certified by the Federal Aviation Administration (FAA) for use in commercial aircraft by meeting the demanding requirements of the RTCA DO-178B standard (Level A) for software used in avionics equipment. To meet the requirements for this standard, it must be possible to demonstrate through documentation and testing that the software is both robust and safe.

This is particularly important for an operating system, as it demonstrates that it has the proven quality to be usable in any application. Every feature, function, and line of code of µC/OS-II has been examined and tested to demonstrate its safety and robustness for safety-critical systems where human life is on the line.

µC/OS-II also follows most of the Motor Industry Software Reliability Association (MISRA) C Coding Standards. These standards were created by MISRA to improve the reliability and predictability of C programs in critical automotive systems.

Your application may not be a safety-critical application, but it's reassuring to know that the µC/OS-II has been subjected to the rigors of such environments.

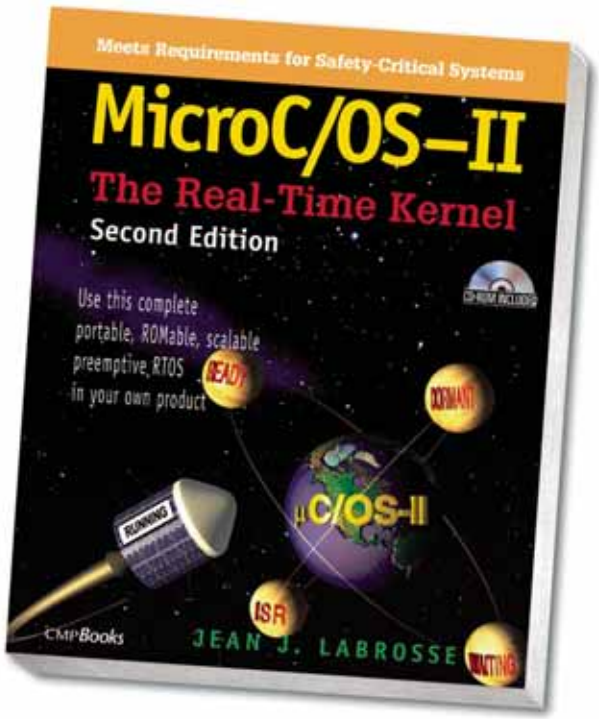


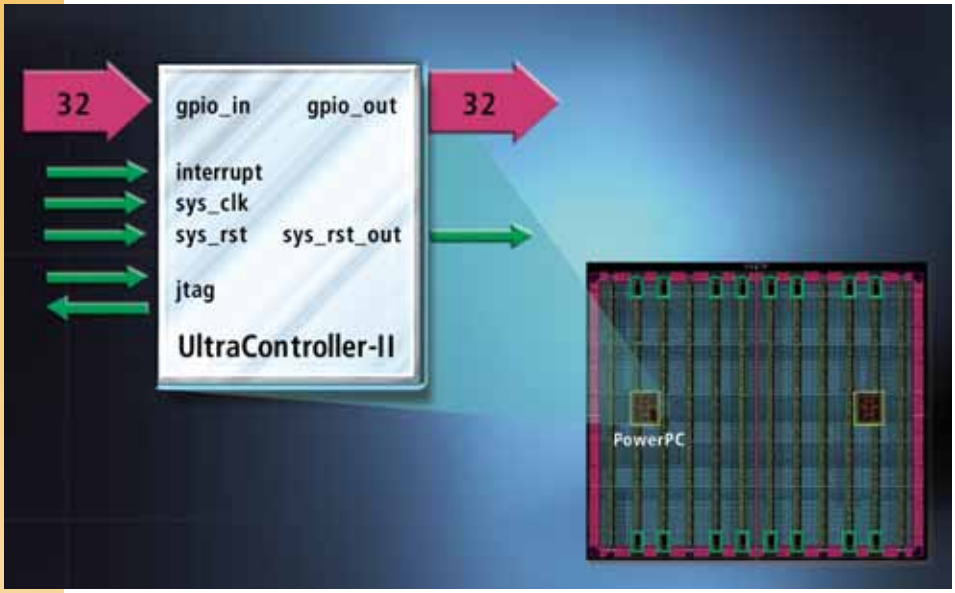
Figure 2 – MicroC/OS-II book cover

Conclusion

An RTOS is a very useful piece of software that provides multitasking capabilities to your application and ensures that the most important task that is ready to run executes on the CPU. You can actually experiment with µC/OS-II in your embedded product by obtaining a copy of the MicroC/OS-II book. You only need to purchase a license to use µC/OS-II for the production phase of your product.

UltraController™-II

Ultra-Fast, Ultra-Small PowerPC-Based Microcontroller Reference Design



Embedded applications present a variety of design challenges, requiring both hardware and software to accomplish real-time processing. This is because non-performance critical functions and complex state machines are handled more efficiently in software, while timing-critical, parallel structures are handled more efficiently in hardware. The Virtex-II Pro and Virtex-4 FX FPGAs meet these requirements by providing fast FPGA fabric and immersed PowerPC™ processors on the same device.

The UltraController-II, an enhanced version of the popular UltraController, provides the easiest way to use the embedded PowerPC processor in the Virtex-II Pro and Virtex-4 FX FPGAs. This minimal footprint embedded processing engine maximizes performance and minimizes FPGA resources by running code strictly from the integrated PowerPC caches. System designers can easily incorporate the UltraController-II design by using the award winning Xilinx Platform Studio tool suite.

PowerPC Made Easy

The UltraController-II, an enhanced version of the popular UltraController, is a minimal footprint embedded processing engine based on the PowerPC 405 processor core in the Virtex-II Pro and Virtex-4 FX FPGAs. With exciting new features, the UltraController-II further simplifies PowerPC based FPGA designs.

Ease-of-Use for Hardware and Software Designers – The UltraController-II reference design is a pre-verified processing engine with 32 bits of user-defined general purpose input and output as well as interrupt handling capability. It enables FPGA logic centric designers to partition design functions between FPGA logic and software while leveraging the embedded PowerPC core in the Virtex-II Pro and Virtex-4 FX FPGAs, using the award winning Xilinx Platform Studio design flow.

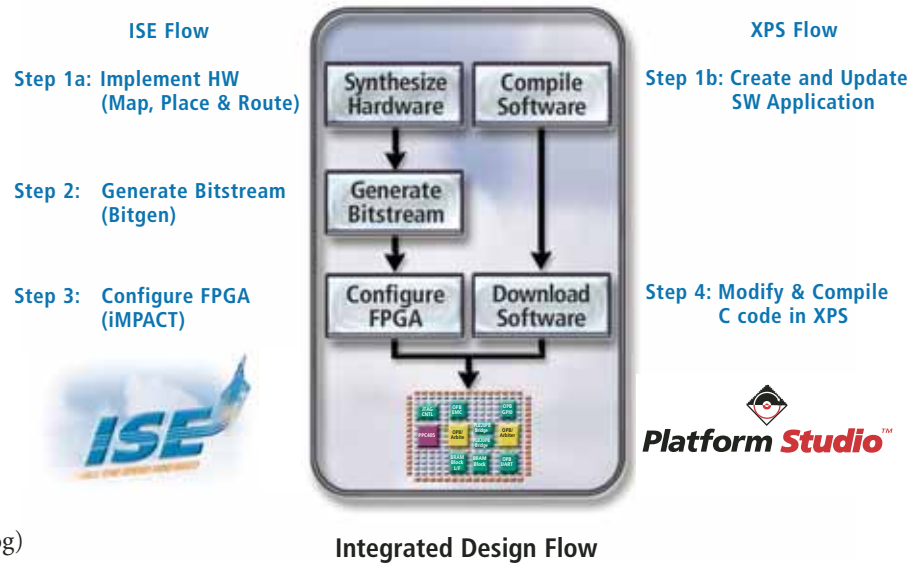
Maximum Performance – Processing power is determined by program execution speed. The UltraController-II can be clocked at the maximum PowerPC clock frequency (up to 400 MHz in Virtex-II Pro and 450 MHz in Virtex-4 FX FPGA) which far exceeds any soft core processor implementation. It also delivers up to 702 DMIPs with a low 0.45 mW/MHz power consumption in Virtex-4 FX.

Minimum FPGA Resource – Code now executes faster by using the 16 KB instruction and 16 KB data-side cache memory. By utilizing cache memory to hold instructions and data no additional BRAM resources are required. As a result, the UltraController-II occupies only 10 logic cells – 1/5 the logic size of the original UltraController!

XILINX®

Key Features

- Utilizes the PowerPC in the Virtex-II Pro and Virtex-4 FX
- Occupies only 10 logic cells
- Runs at up to 400 MHz in Virtex-II Pro and 450 MHz in Virtex-4 FX
- Code and data execute from the integrated 16KB instruction and 16 KB data caches
- 32-bit input and 32-bit output ports
- External interrupt support
- Timer functionality support (FIT, PIT, Watchdog)
- Easy to integrate using the award winning Xilinx Platform Studio (XPS) tool flow
- Fully observable with ChipScope Pro
- Verilog and VHDL source code provided
- "C" source code examples provided



Integrated Design Flow

Design Flow

The UltraController-II development leverages the proven hardware flow of ISE and simplified software flow within the Xilinx Platform Studio (XPS) and Embedded Development Kit (EDK) to provide rapid prototyping and debug abilities. The simplified software flow enables a direct code-to-download option with zero hardware impact, yet maintains the standard ISE development tool flow for full synthesis, simulation and PAR capabilities.

UltraController Features Comparison

Feature	UltraController in Virtex-II Pro	UltraController-II in Virtex-II Pro	UltraController-II in Virtex-4 FX
Performance			
Max fmax	200 MHz	400 MHz	450 MHz
Max DMIPS	220 DMIPs	624 DMIPs	702 DMIPs
DMIPS/MHz	1.1	1.56	1.56
Logic Cells	49 Logic Cells	10 Logic Cells	10 Logic Cells
Power mW/MHz	0.9	0.9	0.45
BRAM Usage	UC 4x4: 4 UC 8x8: 8	0	0

Take the Next Step

Visit **www.xilinx.com/ultracontroller** to download the free UltraController II reference design. Contents include:

- Complete VHDL/Verilog Source Code
- IP and software included
- Collateral Included
  - QuickStart Tutorial
  - HDL Sources & Examples
  - "C" Source Code Examples

The UltraController II application note (XAPP575) is available at ***www.xilinx.com/bvdocsappnotes/xapp575.pdf***

The UltraController application note (XAPP672) is available at ***www.xilinx.com/bvdocsappnotes/xapp672.pdf***

Corporate Headquarters
Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Tel: 408-559-7778
Fax: 408-559-7114
Web: www.xilinx.com

Europe Headquarters
Xilinx, Ltd.
Citywest Business Campus
Saggart,
Co. Dublin
Ireland
Tel: +353-1-464-0311
Fax: +353-1-464-0324
Web: www.xilinx.com

Japan
Xilinx, K. K.
Shinjuku Square Tower 18F
6-22-1 Nishi-Shinjuku
Shinjuku-ku, Tokyo
163-1118, Japan
Tel: 81-3-5321-7711
Fax: 81-3-5321-7765
Web: www.xilinx.co.jp

Asia Pacific
Xilinx, Asia Pacific Pte. Ltd.
No. 3 Changi Business Park Vista, #04-01
Singapore 486051
Tel: (65) 6544-8999
Fax: (65) 6789-8886
RCB no: 20-0312557-M
Web: www.xilinx.com

Distributed by

FORTUNE 2005 100 BEST COMPANIES TO WORK FOR

XILINX®
The Programmable Logic Company℠

Printed in U.S.A.
PN 0010751-1

Support Across the Board.™

XLerated Solutions Seminar from Avnet Electronics Marketing Showcases the Latest Innovations from Xilinx®

Avnet Electronics Marketing introduces a new FREE, one-day seminar that will detail how to utilize Xilinx FPGA technologies in the areas of programmable logic, connectivity, embedded processing, and design tools. It will include presentations, case studies, and live demonstrations to help you assess how these pioneering offerings can be leveraged in your company's product plans.

Seminar attendees will qualify for reduced pricing on special bundles of training classes, software tools, development hardware and the Embedded Systems Starter kit. Free Avnet Design Services evaluation kits (choice of Virtex or Spartan-based) will also be given away.

Seminar Topics

- The Virtex-4 FX family of FPGAs, which delivers breakthrough performance at the lowest cost and offer a compelling alternative to ASICs and ASSPs.
- The Spartan-3 FPGA family, which delivers an unmatched combination of low-cost and full-feature capability.
- Detailed coverage of the capabilities of these two families using the Integrated Software Environment (ISE) and Embedded Development Kit (EDK).

Registration: em.avnet.com/xleratedsolutions

Locations	Dates
Minneapolis, MN	March 22
San Jose, CA	March 22
Chicago, IL	March 23
Ottawa, ON	March 24
Toronto, ON	March 25
Boston, MA	March 30
Rochester, NY	March 31
Atlanta, GA	April 4
Huntsville, AL	April 5
Los Angeles, CA	April 5
Irvine, CA	April 6
Baltimore, MD	April 6
San Diego, CA	April 7
Philadelphia, PA	April 7
Seattle, WA	April 7
Phoenix, AZ	April 8
Salt Lake City, UT	April 18
Denver, CO	April 19
Raleigh, NC	April 19
Orlanda, FL	April 20
Ft. Lauderdale, FL	April 21
Austin, TX	April 27
Dallas, TX	April 28

AVNET® electronics marketing

Enabling success from the center of technology™

1.800.408.8353
www.em.avnet.com

ONE FAMILY. MULTIPLE PLATFORMS. ENDLESS POSSIBILITIES.

LX — Optimized for Logic
SX — Optimized for DSP
FX — Optimized for Processing/Connectivity

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

Platform Studio™

LOWEST SYSTEM COST · HIGHEST SYSTEM PERFORMANCE

PN 0010862