

Xcell journal

SOLUTIONS FOR A PROGRAMMABLE WORLD

Xilinx Unveils ARM-Based Architecture Targeting Software and System Developers

INSIDE

BDTI Study Certifies High-Level Synthesis Flows for DSP-Centric FPGA Design

A Mix of FPGA IP and Resources Makes DisplayPort Compliance Easy

CloudShield Uses Virtex-5 FPGAs to Speed Packet Processing

FPGA-based Control Plane/Data Plane Video Processing Suits Industrial Apps

Design Platforms Accelerate Application Success

Development kits help ramp up new Spartan®-6 or Virtex®-6 FPGA designs

Avnet Electronics Marketing introduces three new development kits based on the Xilinx Targeted Design Platform (TDP) methodology. Designers now have access to the silicon, software tools and reference designs needed to quickly ramp up new designs. This approach accelerates time-to-market and allows you to focus on creating truly differentiated products.

Critical to the TDP methodology is the FPGA Mezzanine Card (FMC) from the VITA standards body. Avnet has collaborated with several industry-leading semiconductor manufacturers to create a host of FMC modules that add functionality and interfaces to the new baseboards, allowing for easy customization to meet design-specific requirements.

Learn more about the new Spartan-6 and Virtex-6 FPGA baseboards and FMC modules designed by Avnet at www.em.avnet.com/drc



DESIGNED BY AVNET

New baseboards for Spartan®-6 and Virtex®-6 FPGAs

- » *Spartan-6 LX16 Evaluation Kit*
- » *Spartan-6 LX150T Development Kit*
- » *Virtex-6 LX130T Development Kit*

New FMC Modules for Baseboards

- » *Dual Image Sensor FMC*
- » *DVI I/O FMC*
- » *Industrial Ethernet FMC*

More are soon to be released!



Quickly see inside your FPGA



Get the most out of your test equipment and shave time out of your next debug cycle with Agilent oscilloscopes and logic analyzers. Our innovative FPGA dynamic probe application allows you to quickly connect to multiple banks of internal signals for rapid validation with real time measurements. Reduce errors with automatic transfer of signal names from the .cdc file from your Xilinx Core Inserter.

Toggle among banks of internal signals for incremental real-time internal measurements without:

- Stopping the FPGA
- Changing the design
- Modifying design timing

Shown with: 9000 Series oscilloscope and 16900 logic analyzer with FPGA dynamic probes



Also works with all InfiniiVision
and Infinium MSO models.

See how you can save time by downloading our
free application note.
www.agilent.com/find/fpga_app_note



Xcell journal

PUBLISHER	Mike Santarini mike.santarini@xilinx.com 408-626-5981
EDITOR	Jacqueline Damian
ART DIRECTOR	Scott Blair
DESIGN/PRODUCTION	Teie, Gelwicks & Associates 1-800-493-5551
ADVERTISING SALES	Dan Teie 1-800-493-5551 xcelladsales@aol.com
INTERNATIONAL	Melissa Zhang, Asia Pacific melissa.zhang@xilinx.com Christelle Moraga, Europe/ Middle East/Africa christelle.moraga@xilinx.com Miyuki Takegoshi, Japan miyuki.takegoshi@xilinx.com
REPRINT ORDERS	1-800-493-5551



www.xilinx.com/xcell/

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124-3400
Phone: 408-559-7778
FAX: 408-379-4780
www.xilinx.com/xcell/

© 2010 Xilinx, Inc. All rights reserved. XILINX, the Xilinx Logo, and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

The articles, information, and other materials included in this issue are provided solely for the convenience of our readers. Xilinx makes no warranties, express, implied, statutory, or otherwise, and accepts no liability with respect to any such articles, information, or other materials or their use, and any use thereof is solely at the risk of the user. Any person or entity using such information in any way releases and waives any claim it might have against Xilinx for any loss, damage, or expense caused thereby.

Xilinx Processor-First Architecture: Catalyst for Mainstream ESL?

An upcoming device launch could be the spark that ignites electronic system-level design industrywide.

It seems that every year for the last 15 years, I've heard at least one analyst or tool vendor declare that this was the year electronic system-level design would go mainstream. Yet with ESL still far removed from the standard hardware (let alone software developer) flow, I had pretty much filed the idea in the same mental category as world peace and flying pigs. But now, thanks to Xilinx's new architecture release (see cover story), I'm seriously starting to think that ESL's day will finally come—and soon.

ESL design—essentially, modeling the behavior of an entire system at a high level of abstraction, using a high-level language—seems to be a perfectly sound answer to the ever-increasing transistor counts, speed and complexity of system-on-chip (SoC) design. The promise of being able to use a language like C to describe desired system functionality—and then have tools partition that functionality into hardware and software, and automatically implement it—could in concept really speed up design innovations. What's more, if a company could find a way to enable software or systems developers to use such a flow, it would tap into a user base several times larger than the traditional user base of hardware engineers. Imagine what system innovations and products these potential millions of engineers could design!

Though ESL's path has been a rocky one, that hasn't deterred a number of players in the electronic-design community—EDA companies, system software companies and even silicon vendors—from pursuing it. They have scored a few big successes, notably high-level synthesis tools (see the article in this issue from research firm BDTI).

However, each of the many ESL vendors is taking a slightly different approach to the complexities of this type of design. Each has its strengths, weaknesses and target markets. Given this diversity, one can't help but see a need for an industrywide focus to bring ESL to fruition—perhaps a technology catalyst, one development that could inspire the many players to hone their efforts and make ESL live up to its promise.

My personal hunch is that the new Xilinx architecture could fit the bill.

You may be saying, well Mike, you work at Xilinx and we expect you to say that. And of course, you are correct. But regardless of my affiliation, I am someone who has heard a lot of pitches in my day. And I'm genuinely impressed with what I have seen of the Xilinx® Extensible Processing Platform thus far. I applaud the folks in charge for taking the time to listen to customers and learn how real products are developed at a system level, and for learning from the mistakes of the past—made by Xilinx and others—before starting to build this well-thought-out architecture.

A device that boots the low-power but very speedy ARM® MPU core first, and right out of the box, rather than requiring that users first program the FPGA logic, is a brilliant move. Also inspired is the support for industry-standard development software and operating systems, and the use of a common industry-standard SoC bus for which customers have already developed oodles of IP.

That said, no one device launch can do the job alone. To truly make the dream of mainstream ESL a reality will take much effort on the part of many companies and individuals. I hope you, our customers and partners, will be among them.



Mike Santarini
Publisher

XCELLENCE BY DESIGN APPLICATION FEATURES

Xcellence in Wired Communications Using Xilinx FPGAs to Accelerate Packet Processing...18

Xcellence in Automotive & ISM Control Plane/Data Plane Video Processing with a Xilinx FPGA...24



12

THE XILINX XPERIENCE FEATURES

Xplanation: FPGA 101 A Xilinx FPGA Route Toward Implementing DisplayPort...30

XTRA READING

Letter From the Publisher Xilinx Processor-First Architecture: Catalyst for Mainstream ESL?...4

Xpert Opinion BDTI Study Certifies High-Level Synthesis Flows for DSP-Centric FPGA Design...12

Xamples A Mix of New and Popular Application Notes...40

Xtra, Xtra A Rundown of New and Upcoming Targeted Design Kits...42

Tools of Xcellence A Bit of News About Our Partners and Their Latest Offerings...44

Xclamations! Supply an Xceptional Caption for Our Artwork and Win an SP601 Evaluation Kit...46

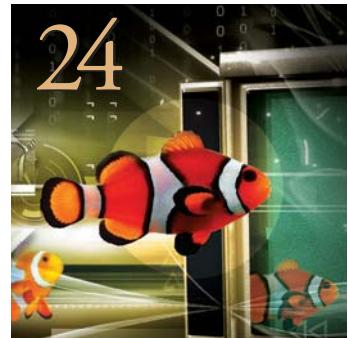


18

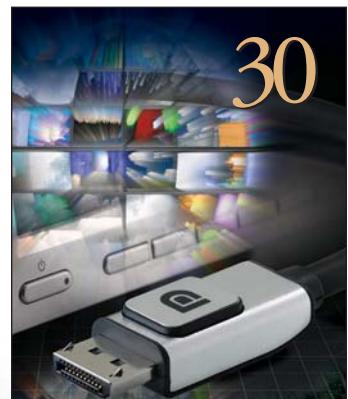
Cover Story

Xilinx Architects ARM-Based
Processor-First, Processor-Centric Device

6



24



30

Xilinx Architects ARM-Based Processor-First, Processor-Centric Device



New architecture targets both software and system developers. Processor boots before programmable logic to speed and simplify system development.



by Mike Santarini
Publisher, Xcell Journal
Xilinx, Inc.
mike.santarini@xilinx.com

For well over a decade, FPGA vendors have been searching for ways to add the vast numbers of embedded-software engineers to a user base composed mainly of hardware design engineers. Software developers outnumber their hardware engineer counterparts by up to 10:1 worldwide, so creating a device that both camps can instantly use has obvious business benefits. Now, Xilinx is architecting a new ARM® microprocessor-based device—an Extensible Processing Platform—tailored to the real way software and system developers work. In breaking through the barrier, this novel device promises to take the company into new markets and to new levels of growth.

As FPGA devices and tools have matured over the last decade to incorporate a greater number of embedded processors (DSPs, microcontrollers and microprocessors) in dominantly programmable-logic architectures, a growing number of embedded-system designers have broadened their skill sets beyond middleware and software development and become competent with hardware design languages. This cross-discipline has allowed these happy few designers to begin using FPGAs to create highly optimized, differentiated architectures that have the right mix of hardware and software for prime system performance, functionality and power consumption.

But while the number of engineers with this mixed skill set has grown steadily but slowly over the last 10 years, the vast majority of systems designers still rely on hardware engineers to create custom hardware functions their systems will need to achieve that optimal mix of performance, functionality, power consumption and system cost.

Processor Comes First

With the Xilinx® Extensible Processing Platform, Xilinx is planning to release a first-of-its-kind device in which a 32-bit ARM Cortex™-A9 processing subsystem running at up to 800 MHz is the dominant

block (see figure). The processor subsystem will be bootable and programmable out of the box. The balance of the new device will consist of a tightly coupled programmable-logic extension block that will allow designers to partition their hardware and software functions based on system requirements. They can implement functions in the programmable-logic extension block to create their own application-specific, highly optimized systems-on-chips (SoCs).

“We’ve put a lot of thought and planning into the architecture of the device and have learned a lot of lessons from our previous devices, like our PowerPC™-based Virtex®-II Pro, Virtex-4 and Virtex-5 FXT FPGAs. We’ve also learned from the missteps of our competitors,” said Vin Ratford, senior vice president for worldwide marketing and business development at Xilinx. “Where all those devices took a hardware design-centric view of system design or simply don’t have enough processing horsepower, our new Extensible Processing Platform takes a processor-first approach, so software designers can start developing with this product right out of the box. They don’t even have to use the extension block if they choose not to.”

But many design teams that have a mix of software and hardware designers will embrace the extension block. Xilinx plans to refine the use model over time with the end goal of offering software and system developers an environment that will enable software gurus to program the programmable-logic extension in addition to the processor without the assistance of hardware designers.

Ratford points out that unlike previous architectures in which the FPGA boots before the processor, this new processor-first platform plays perfectly to how developers actually work when building system architectures.

“Electronic systems and software engineers typically develop what they want their system to do in the software first, and then determine what functions they need to accelerate by implementation in hardware,” he said. “This allows them to fit their design into the appropriate performance, cost and power footprint that ulti-

mately the application is going to need. When they start a project, they are developing a proof of concept. They are less worried about having it tuned to the specific requirements of a specific customer, and are more concerned about having the maximum amount of flexibility in determining what can be done in hardware or software. Through revisions, they decide

the ARM architecture has become the de facto standard choice for designers seeking high-speed, low-power microprocessor cores.

"On all fronts—hardware and software functionality, performance, ecosystem, user familiarity and power—ARM was hands down the best choice for this new architecture," said Ratford. "With power consumption becoming a top-order consideration

and shared memory. By contrast, a typical system that pairs a discrete MPU-based ASSP chip with an FPGA on the same printed-circuit board typically has just over 100 I/Os connecting them.

Further, in the March release of version 4 of its AMBA® bus' Advanced Extensible Interface (AXI), ARM included an extension of the AXI specification optimized for use on programmable logic. The AXI-4 stream protocol extension serves as a bidirectional crossbar communication switch that will take advantage of the abundant I/Os, enabling engineers using the new Xilinx device to achieve new levels of system interblock throughput and at the same time tap into a multitude of hardware peripheral cores that IP vendors and customers have developed for ARM ASIC and ASSP implementations over the last two decades.

This tightly coupled integration between the two domains, along with the new AXI extension, also means that if design teams find a particular function that isn't running optimally on the processor—or if they have a piece of code they want to speed up—they can create hardware for that function and place it in the programmable-logic extension block using an industry-standard interface.

Familiar Software Programming Model

In creating the new architecture, Xilinx was mindful of the requirements and working preferences of its targeted audience.

Because the new device boots the processor system first, at reset, a software developer can program the processor out of the box, side-by-side with the hardware developer. This has the benefit of shortening development cycles by putting these key functions on parallel tracks.

"In fact, someone could buy the part and just use the processor system," said Keith DeHaven, director of processor marketing at Xilinx. "But the value of the device lies in the user's ability to leverage the programmable logic to customize, optimize and differentiate their products while leveraging the command, control and applications features enabled by the ARM-based processor system."

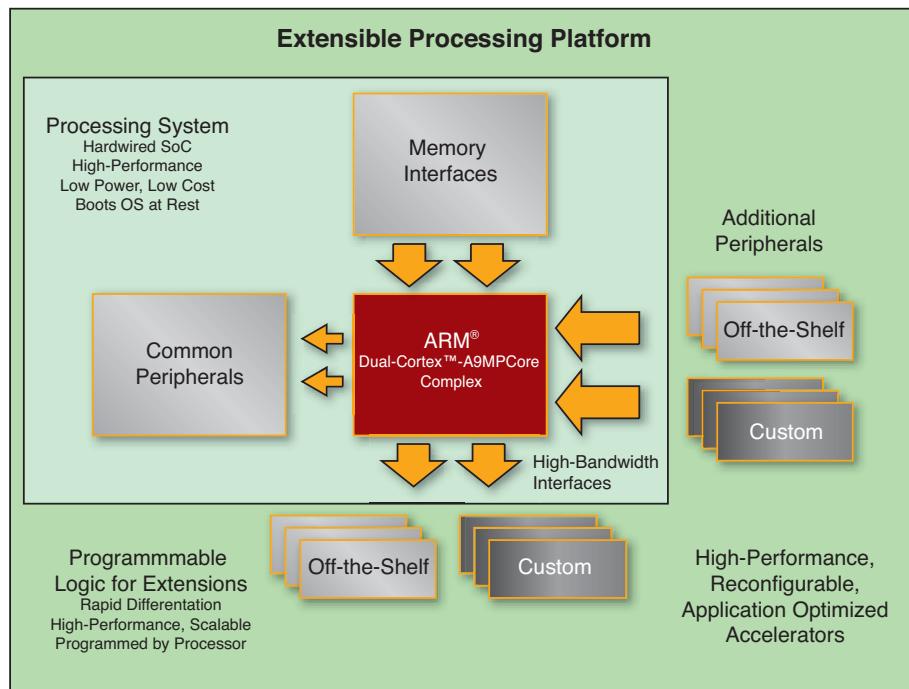


Figure 1 – The Xilinx Extensible Processing Platform pairs an ARM processor with programmable elements.

what functions will go into hardware and what will go in software, and then take steps to refine both to fit their system requirements. Our device will help them do their job faster and better than before."

Xilinx will deliver the new Extensible Processor Platform in the same low-power, high-performance 28-nanometer process technology that it will use for its next-generation FPGAs (see sidebar).

Why ARM?

Xilinx chose to partner with ARM Ltd. because the company is well-established and has an incredible reputation for the quality of both its processor IP and software. Indeed,

for not only wireless but also wired applications, adding the lowest-power processor to the device will give users a mind-blowing number of options for making system trade-offs. They can offload functions to the hardware extension block to increase system performance. And they can create systems that can have screaming performance at one moment, then power down to consume milliamps."

A key feature of the new architecture is the interfaces. Xilinx is interconnecting the full, ARM processor-based system and programmable-logic extension block with a high-bandwidth interface connecting the processor-based system, extension block

The real value of the architecture comes as design teams trade off functions between the processor system and programmable-logic extension block. Now the software developer, not the hardware designer alone, will be dictating how the device will operate.

DeHaven said that the processor system has a constant set of peripherals, switches and memory interfaces, providing the software developer with a consistent programming environment. In addition, developers can get started with existing ARM tools and available hardware (summarized in Table 1) to jump-start their efforts.

But of course, the real value of the architecture comes as design teams trade off functions between the processor system and programmable-logic extension block. Now the software developer, not the hardware designer exclusively, will be dictating from the processor view how the device will operate. For example, the processor system may be using data in the extension block to do peripheral functions, or it may delegate control to the block. Developers will likely run hardware/software co-simulation to see if a given function will run in hardware faster, consume less power or reduce cost. Still others may simply want to take software functions that are not likely to change and offload them to the extension block to free up more room in the processor code for other commands.

Once they solidify what functions will go into hardware and software, they can then have their hardware engineers use Xilinx's ISE® Design Suite to implement those functions with an AMBA-AXI standard interface in the programmable-logic extension block. Meanwhile, the developers can continue creating software while the hardware team programs the extension block.

While the processor-first architecture is unique and the use model more representative of how software developers really work, Xilinx plans to make the flow even more intuitive.

Xilinx and its partners are currently developing a comprehensive set of common and standards-based accelerators and peripherals (IP cores, in hardware speak) and related

drivers and APIs that will further help software and system developers add functions to their designs. Some of these accelerators and peripherals will be ready at the time of launch, allowing users to focus on creating their own custom IP to meet their system needs and differentiate their products.

The accelerators and peripherals will range in size from small functions that

Vendor	OS
eSol	eT-kernel Multicore Edition
Express Logic	ThreadX
Green Hills	INTEGRITY 10
Kernel.org	Linux 2.6+
Mentor Graphics	Nucleus PLUS RTOS
Microsoft	WinCE
MontaVista	MobiLinux 5.0
QNX	Neutrino RTOS
Symbian	Symbian OS 9+
Wind River	VxWorks 6.6 SMP

Table 1 – ARM has a mature and robust ecosystem for operating systems and OS development tools. Here are some of the OSes the ARM Cortex ecosystem supports.

designers can mix and match in the extension block to fully populated extension functions that target specific design domains—connectivity, DSP and processing—and vertical markets such as automotive; industrial, scientific and medical; aerospace and defense; wired and wireless communications, among others.

In the longer term, Xilinx is developing C-to-FPGA compiler flows in an effort to eventually offer software and electronic-system developers a way to readily move functions between software and hardware programming environments to rapidly develop, evaluate and optimize their systems. “The idea is that they will be able to develop in a C-based environment and see results in hardware and software rapidly,” said DeHaven. In fact, Xilinx has been diligently monitoring research conducted by the benchmarking and analysis firm BDTI in evaluating the use models of C-level synthesis tools (see related BDTI article in this issue).

While software developers will be able to use any commercial development tools supporting the ARM Cortex-A9, Xilinx plans to bundle its own tools with its new device to help folks get started. Bundled in the tool kit and PCB will be an Eclipse-based integrated development environment, GNU-based compiler, debugger and drivers. “Users will be able to leverage the environment of their choice,” said DeHaven. “They will be able to develop with industry tools or Xilinx development tools that support the Cortex-A9 and ARM CoreSight™ debug interfaces.”

In addition to ARM native support, Xilinx is also working closely with leading third-party solution providers to develop device-specific software bundles—operating systems and development tools—aimed at engineers using the new device.

Suited for Multiple Vertical Markets

Xilinx created the new architecture at the urging of customers who are looking for a device that is scalable, flexible, upgradeable and will allow them to quickly create derivatives for their needs. The Extensible Processing Platform will allow them to further differentiate their products from competing systems based on fixed-function

ASSPs and ASICs. "We've previewed this to several customers and they are eager to get their hands on the device," said Ratford. "I predict the market reach of this device will be incredible."

For example, Xilinx expects any vertical market that incorporates intelligent video will see immediate benefits from using the new device. Intelligent video requires multiple processing steps such as preprocessing pixel-leveling, which is computationally intensive and thus well served by the parallel-processing capabilities of programmable

logic. Intelligent video also requires analytic processing at the element level, which is served using a combination of compatible parallel (programmable-logic) and serial (MPU-based) functions. Meanwhile, application processing at the frame level requires decision, control and communications processing typically performed by MPUs.

Among the specific video markets that stand to gain are automotive driver assistance; consumer multiclass, multifunctional printers; general embedded systems that use scanners; industrial smart cameras,

including Internet Protocol surveillance cameras and machine vision, DVRs, medical imaging systems, broadcasting studio cameras and transcoders; and defense-grade night-vision equipment.

One intelligent-video application that will see immediate benefit from the new architecture is automotive driver assistance. Key customers in this space have been urging Xilinx for years to create an ARM-based extensible platform.

Automotive customers will be able to program the device to control and analyze

28-nm FPGAs will target the right mix of power consumption and performance

In February 2010, Xilinx announced it will manufacture its next-generation FPGAs in 28-nanometer high-k metal gate (HKMG) high-performance, low-power process technologies and will use a new, unified ASMBL™ architecture for the devices.

Xilinx evaluated several technologies at the 28-nm node before concluding that the HKMG high-performance, low-power process offered the ideal mix of power consumption and performance for Xilinx's next-generation FPGAs.

For the last four generations of IC processes, static power caused by transistor leakage has become increasingly problematic. At 28 nm, it becomes a first-order effect. Where dynamic power refers to the power a device consumes when it is performing the tasks it was designed to do, static power is wasted juice that increases overall power consumption and produces more heat.

If left unchecked at the 28-nm node, static power draw can account for well over 50 percent of a device's total power draw, drastically taxing power and thermal budgets. A key to the HKMG high-performance, low-power process is the material it uses as a gate dielectric (thermal insulator): instead of silicon dioxide, the technology employs hafnium dioxide. Where a 40-nm silicon dioxide material only afforded a k value (or gate dielectric constant) of 3.9, the 28-nm HKMG high-performance, low-power process will have a k value of 25, which is much more resistant to leakage.

By pairing HKMG with Xilinx's ASMBL (Advanced Silicon Module Block) architecture—which enables Xilinx to rapidly and cost-effectively assemble multiple domain-optimized platforms with an optimal blend of features—Xilinx believes its next-generation devices will have 50 percent lower static power consumption than devices implemented using alternative 28-nm high-performance processes. At the same time, Xilinx is including architectural innovations enabling a system-level performance increase of up to 50 percent over previous-generation FPGAs.

Reducing power consumption also enables increased device capacity. Using this new process technology, Xilinx will introduce a new ultra-high-end 28-nm FPGA to enable applications that require increased capacity in keeping with Moore's Law.

In addition, Xilinx's design team is also introducing extra measures to help designers optimize their designs for the right mix of functionality and performance, as well as power.

Xilinx worked closely with customers to identify and understand the architectural bottlenecks in their systems. One of the biggest areas constraining performance is in interfacing the FPGA with other devices. To address this issue, Xilinx will introduce new clocking technology and will harden critical datapath components. The company will also introduce fine-grained clock-gating features and new place-and-route algorithms in the ISE Design Suite to further reduce power. Fine-grained clock-gating technology is a patented algorithm that analyzes the logic equation and disables wasted logic transitions that do not contribute to the final result. This methodology will help customers effectively reduce the power consumption of their designs by as much as 30 percent.

For more information, visit <http://www.xilinx.com/technology/roadmap/28nm-technology.htm>.

— Mike Santarini

The new device will be processor centric, rather than FPGA centric.

'In a lot of applications you want the controlling software to reprogram the FPGA depending on the application you are running, and there are times where you want the processor to run autonomously from the FPGA fabric.'

the data from multiple sensors positioned 360 degrees around a vehicle, with each sensor performing multiple functions simultaneously. The intelligent control sensor could, for example, have these sensors monitor lanes painted in the road, detect swerving vehicles in adjacent lanes, sync the automobile's speed to that of the vehicle ahead, detect pedestrians and monitor spaces between parked cars to locate adequate parking spots—all simultaneously. A system like this might warn the driver instantly if it detects threats; it could even automatically slow the vehicle down to avoid a collision.

Because such a device would be hardware and software programmable, tier-one vendors could make derivatives of this controller for various car manufacturers and various product lines of each of a carmaker's vehicles—without having to change the overall configuration of the control unit. That capability will save OEMs vast amounts of time, effort and money. Further, the software and hardware programmability means that these devices can be serviced or upgraded in the field.

Similarly, in industrial controls, users can create systems that will manage and analyze data from a series of sensors and motors that in real time can identify defective products flying down an assembly line, detect cracks in the machinery or power down motors when they are running too hot or are not in use to reduce factory costs, optimize operations and even save the lives of workers.

The new device also holds great promise for applications in the wired and wireless communications markets, especially in wireless LTE radio, baseband and enterprise femtocell markets, and in routers, switches and multiplexers in the wired communications arena.

Xilinx anticipates the device will also find multiple uses in the military-aerospace business, especially in cockpit control, munitions and communications equipment supporting the Global Information Grid (see the cover story in *Xcell Journal*, Issue 69).

National Instruments, a longtime customer of Xilinx and alpha customer of the new device, has been closely monitoring the development process and giving feedback to Xilinx. Currently, National Instruments pairs on a PCB a real-time processor with a Xilinx FPGA in its NI Reconfigurable I/O (RIO) LabVIEW FPGA embedded platform (<http://www.ni.com/fpga>). The platform offers a wide range of peripheral I/O and predefined software libraries that customers can mix and match to create unique embedded systems for various vertical markets. By offloading some of the functions from the standalone processor and instead running them in the FPGA, LabVIEW FPGA can run much faster and with the determinism required for instrumentation, measurement and control applications. The LabVIEW FPGA environment also enables the typical LabVIEW user or market-domain expert to accomplish this without knowing anything about the details of FPGA design.

National Instruments expects embedded products based on the new Xilinx architecture will run much faster, with the added benefit of consuming much less power, said NI R&D fellow Keith Odom.

"Now NI will have a very high-performance processor capable of running our highly productive graphical design environment, and we have very high bandwidth connecting the processor system to the programmable-logic fabric," said Odom. "The amount of data we can transfer between the processor and the pro-

grammable-logic fabric is much higher than you would typically see in an FPGA that has an embedded processor or even a microcontroller-based ASSP. Because the bandwidth is so high, you can move beyond mechanical and audio and into more electrical-, radio- or vision-related applications, along with enabling much more sophisticated algorithms to be applied to the data in all applications. This device opens up a lot of new possibilities."

Odom notes that because it essentially integrates two devices in one, data communication on the new device will consume less power. "Because these blocks are connected with so many I/Os, you don't have to burn the power you normally would in high-speed chip-to-chip communication. You'll also be able to power it down into standby modes," he said.

Odom also likes the fact that the new device will be processor centric, rather than FPGA centric. "It is absolutely key," said Odom. "In a lot of applications you want the controlling software to reprogram the FPGA depending on the application you are running, and there are times where you want the processor to run autonomously from the FPGA fabric. So our application constantly switches out what's on the FPGA based on your current processing needs, and this architecture is ideal for that."

"It will be incredible to see what customers will be able to do with device once we've released it," said Ratford of Xilinx. "We're very excited, but still have some work to do for the device to reach its full potential."

Xilinx will announce pricing and availability of the new device in early 2011. To learn more details about the new architecture so you can start development today, visit www.xilinx.com/technology/roadmap/processing-platform.htm. 

BDTI Study Certifies High-Level Synthesis Flows for DSP-Centric FPGA Design

Advances in high-level synthesis tools make it easier for DSP developers to implement their designs in FPGAs, benchmarking firm finds.



by Jeff Bier
President
BDTI
bier@bdti.com

Jennifer Eyre White
DSP Analyst
BDTI
eyre@bdti.com

In recent years, high-level synthesis tools have become something of a holy grail for engineers who use—or want to use—FPGAs in their designs. These tools take as their input a high-level representation of an application (written in C or MATLAB’s M language, for example) and generate a register-transfer-level HDL description of a hardware implementation targeting an FPGA.

High-level synthesis tools (HLSTs) are particularly interesting to two groups of prospective users: FPGA users who are implementing demanding digital signal-processing (DSP) applications, and high-performance DSP processor users doing the same thing. That’s because taxing signal-processing workloads—which typically have high data rates and high levels of parallelism—are often well suited for implementation on FPGAs using HLSTs.

For current FPGA users, these tools hold the promise of an easier and faster design process. For current DSP processor users, HLSTs offer a different, yet equally tantalizing prospect: the ability to migrate to a more powerful processing engine (an FPGA) without having to wrangle RTL code. So, what’s not to like?

One key problem is that, in the past, high-level synthesis tools have not been able to generate efficient RTL code (in terms of resource usage). Most engineers were unwilling to sacrifice the performance and efficiency of hand-coded RTL, and the tools never achieved significant market penetration. Emerging anecdotal evidence suggests, however, that newer HLSTs for Xilinx® FPGAs are very efficient and usable. Given this conflicting information, how is a prospective user to judge whether high-level synthesis tools are worth considering?

To answer this question, we at independent benchmarking and analysis firm BDTI developed the BDTI High-Level Synthesis Tool Certification Program in 2009. Our aim was to provide objective, credible data and analysis of HLSTs for FPGAs so as to enable potential users to quickly understand their capabilities and limitations in demanding signal-processing applications. We conducted the evaluation from the perspective of experienced DSP software engineers who had no previous experience in FPGA development—a background that is characteristic of many of the processor users who stand to benefit from HLSTs.

The first two HLSTs the program evaluated were Synfora’s PICO and AutoESL’s AutoPilot. In early 2010, we released the first results from the evaluation program—including a few results that will surprise many FPGA and DSP processor users.

Implementation Using HLSTs

Our process for implementing test applications using HLSTs starts with a high-level-language description of the desired functionality, from which the high-level synthesis tools generate RTL implementations. Then Xilinx’s RTL tools (the Integrated Synthesis Environment, or ISE®, and the Embedded Development Kit, or EDK) transform that RTL implementation into a complete FPGA implementation in the form of a bitstream for programming a specific Xilinx FPGA on a specific hardware platform with I/O and memory. In this case, the platform was a Xilinx XtremeDSP™ Video Starter Kit—Spartan®-3A DSP Edition, a Targeted Design Platform based on a Spartan-3A DSP FPGA.

We could have limited our evaluation to the high-level synthesis tools alone, ignoring the RTL-to-bitstream portion of the design flow. But we believe that potential users need to know what it takes to get from the high-level application description all the way to an FPGA implementation—a job that requires the RTL tools in addition to the high-level synthesis tool. For this reason, we evaluated the entire implementation flow—

not just the C-to-RTL portion, but also the Xilinx RTL tool chain.

Typically, the first step in implementing an application on any hardware target is to restructure the initial C code. By “restructuring,” we mean rewriting the initial C code (which is typically coded for clarity and ease of understanding rather than for performance) into a format more suitable for the target processing engine. On a DSP processor, for example, it may be appropriate to rearrange an application’s control flow so that intermediate data always fits in cache. With a high-level synthesis tool targeting an FPGA, restructuring typically provides a representation of the application that allows the tool to extract potential parallelism, resulting in a streaming pipelined implementation.

In general, high-level synthesis tools do not handle restructuring automatically. Instead, the restructuring is done by hand. In fact, designers can do the restructuring entirely independently of the high-level synthesis tool. In our evaluation, for example, we used Microsoft Visual Studio for restructuring and reverifying the C code. Compared with handwritten RTL code, where restructuring and language translation occur as a single combined step, restructuring entirely in C is easier and less error-prone—a key advantage for high-level synthesis tools.

After restructuring the high-level code, the user directs the HLST to synthesize a hardware implementation of the specified functionality in the form of RTL HDL code. Then Xilinx’s RTL tools (ISE and EDK) take the RTL code the HLST has generated, perform synthesis and place-and-route tasks, report the resource utilization of the implementation and alert the user to any timing issues.

BDTI’s Tool Certification Program

BDTI’s goal in creating the High-Level Synthesis Tool Certification Program was to enable two key points of comparison in order to serve the two classes of potential HLST users. First, we wanted to compare the efficiency (in terms of resource usage) of HLST-based FPGA application implementations vs. an implementation based

Historically, demanding applications implemented in handwritten RTL code on an FPGA typically achieved relatively good quality of results but poor productivity, while applications implemented on DSP processors provided good productivity but relatively poor quality of results.

on handcrafted RTL code. This information is critical for current FPGA users pondering whether to adopt HLSTs to accelerate their development times. Second, we wanted to gauge the performance and development effort associated with using HLSTs on an FPGA relative to implementing the same workload using a DSP processor with its associated software development tools. This comparison enables DSP processor users to assess how difficult it would be to switch technologies and migrate to an FPGA-based design.

We evaluated the high-level synthesis tool flows (including the associated RTL tools) using two well-defined sample applications, or “workloads.” These applications (described briefly in the next section) are representative of demanding digital signal-processing applications that designers often implement on FPGAs, with high data rates and high computational demands. Other types of applications would likely yield different results than those presented here.

We implemented the two applications using several approaches. First, we implemented a given workload on the target FPGA using each high-level synthesis tool in conjunction with the Xilinx RTL tools. We then implemented the same workload on the same FPGA using a traditional RTL design approach, or on a DSP processor using its associated development tools (depending on the workload under consideration). In this manner, we were able to compare the quality of results and productivity associated with using various tool-plus-chip combinations.

Evaluation Workloads

The two applications we used for evaluation purposes were the BDTI Optical

Flow Workload and the BDTI DQPSK Receiver Workload.

The term “optical flow” (or “optic flow”) refers to a class of video-processing algorithms that analyze the motion of objects and object features (such as edges) within a scene. The BDTI Optical Flow Workload operates on a 720p resolution (1,280 x 720 progressive scan) input video sequence and produces a series of two-dimensional matrices characterizing the apparent vertical and horizontal motion within the sequence. In designing this workload, we incorporated dynamic, data-dependent decision making and array indexing in order to ensure a challenging test case for the tools.

There are two operating points associated with the BDTI Optical Flow Workload, each of which uses the same algorithm but is optimized for a different metric. Operating Point 1 is a fixed workload defined as processing video with 720p resolution at 60 frames per second. The objective for Operating Point 1 is to minimize the resource utilization required to achieve the specified throughput. (Resource utilization refers to the fraction of available processing-engine resources required to implement the workload.)

The objective for Operating Point 2, meanwhile, is to maximize the throughput (measured in frames per second) using all available device resources.

The second workload, the BDTI DQPSK Receiver Workload, is a wireless communications receiver baseband application that includes classical communications blocks found in many types of wireless receivers. It is a fixed workload with a single operating point defined as processing an input stream of complex, modulated data at 18.75 Msamples/s, with

the receiver chain clocked at 75 MHz. The receiver produces a demodulated output bitstream of 4.6875 Mbits/s. The objective for this workload is to minimize the FPGA resource utilization needed to achieve the specified throughput.

Memory usage and memory bandwidth requirements vary significantly between the workloads. The BDTI DQPSK Receiver Workload requires minimal memory usage (and therefore, no external memory chip). The BDTI Optical Flow Workload, however, requires storing a history of four video frames (1,280 x 720 pixels per frame), and thus requires an external memory chip to accompany the Spartan-3A DSP FPGA. Optical Flow Workload Operating Point 1 requires a single external memory chip and interface (with a bandwidth of approximately 450 Mbytes/s), while Optical Flow Workload Operating Point 2 typically requires two external memory chips and interfaces with a combined bandwidth of approximately 1.4 Gbytes/s.

For the BDTI Optical Flow Workload, typical FPGA implementations process one pixel per clock cycle for Operating Point 1 and two pixels per clock cycle for Operating Point 2. BDTI DQPSK Receiver Workload implementations process one input sample every four clock cycles.

Description of Metrics and Platforms

Historically, demanding applications implemented in handwritten RTL code on an FPGA typically achieved relatively good quality of results (that is, performance and efficiency) but poor productivity, while applications implemented on DSP processors provided good productivity but relatively poor quality of results. High-level synthesis tools targeting FPGAs seek to provide the best of both worlds: good qual-

ity of results achieved with high productivity levels. Therefore, in our evaluation we consider two categories of metrics: quality of results and usability.

Quality-of-results metrics assess the performance and resource utilization of the workload implementation. The BDTI Optical Flow Workload reports quality-of-results metrics for the HLST-Xilinx flow and for the DSP processor flow. The BDTI DQPSK Receiver Workload delivers quality-of-results metrics for the HLST-Xilinx flow and for a traditional FPGA implementation using a handwritten RTL design that Xilinx developed in accordance with typical industry design practices, including the use of Xilinx CORE Generator™ intellectual-property blocks where appropriate.

Usability metrics assess the productivity and ease of use associated with the HLST-Xilinx design flow, and are based on our experience in implementing the BDTI Optical Flow Workload. These metrics compare the productivity and ease of use associated with the HLST-plus-Xilinx tool flows targeting an FPGA relative to using a DSP processor with its associated software

development tool chain. We evaluated usability metrics qualitatively based on nine aspects of tool use, including out-of-the-box experience, ease of use, completeness of tool capabilities, efficiency of overall design methodology and quality of documentation and support.

For this evaluation, the target FPGA is the Xilinx Spartan-3A DSP 3400 (XC3SD3400A). For the BDTI Optical Flow Workload, the Xilinx XtremeDSP Video Starter Kit—Spartan-3A DSP Edition is the target platform. We used Xilinx RTL tools, including the ISE and EDK tool suites (version 10.1.03, lin64), along with the high-level synthesis tools.

The target DSP processor for this project is the Texas Instruments TMS320DM6437. This video-oriented processor includes a 600-MHz TMS320C64x+ DSP core along with video hardware accelerators. (The hardware accelerators are not applicable to the BDTI Optical Flow Workload, and therefore we didn't use them.) The evaluation used the Texas Instruments DM6437 Digital Video Development Environment as the target platform, and used the Texas Instruments Code Composer Studio tools

suite (version V3.3.82.13, Code Generation Tools version 6.1.9).

Implementation, Certification Process

We distributed the job of implementing the two workloads on the two chips among the high-level synthesis tool vendors, Xilinx and BDTI based on the chip and tool chain used. The HLST vendors implemented both workloads using their tools along with the Xilinx tools, and submitted performance and resource utilization results to BDTI for verification and certification. We used these certified results to generate the quality-of-results metrics presented in this article.

In parallel, our engineers received training from the HLST vendors and independently implemented portions of the BDTI Optical Flow Workload using the high-level synthesis tools and Xilinx tools. This process provided BDTI with first-hand insight into the usability of the tool chains and the quality of results they generated. We also implemented the BDTI Optical Flow Workload on the DSP processor, while Xilinx implemented the handwritten RTL FPGA version of the

Performance (Frames/Second)
Higher is Better



Figure 1 – A Spartan-3A DSP FPGA using HLSTs achieved 195 frames/s at 720p resolution on the BDTI Optical Flow Workload, a video application, whereas a C64x+ DSP processor attained just 5.1 frames/s.

Cost/Performance (Frames/Second)
Lower is Better



Figure 2 – Cost/Performance for the BDTI Optical Flow Workload (720p) on a Spartan-3A DSP FPGA using HLSTs was far better than on a 600-MHz TI C64x+ architecture DSP.

BDTI DQPSK Receiver Workload (which BDTI then verified and certified).

Quality of Results: Performance and Efficiency

As shown in Figure 1, the FPGA implementations of the BDTI Optical Flow Workload created using high-level synthesis tools achieved roughly 40 times the performance of the DSP processor implementation. Bringing chip cost into the analysis, Figure 2 shows the corresponding cost/performance advantage, which is roughly 30x in favor of the FPGA implementation. Clearly, FPGAs used with high-level synthesis tools can provide compelling performance and cost advantages for some types of applications. (More detailed results are available on www.BDTI.com.)

We also evaluated the efficiency of the HLST-based FPGA implementations of the BDTI DQPSK Receiver Workload vs. the same workload implemented using hand-coded RTL. Here, too, the HLSTs performed extremely well. As shown in

Platform	Chip Resource Utilization (Lower is better)
HLST plus Xilinx RTL tools targeting Xilinx XC3SD3400A FPGA	5.6% - 6.4%
Handwritten RTL code using Xilinx RTL tools targeting Xilinx XC3SD3400A FPGA	5.9%

Table 1 – Resource utilization for BDTI DQPSK Receiver Workload, at 18.75 Msamples/s input data with a 75-MHz clock speed



Table 1, both AutoPilot and PICO were able to generate RTL code that was comparable in efficiency (that is, resource usage) to handwritten RTL code. The similarity of HLST and handwritten RTL results is probably not coincidental; we provided AutoESL and Synfora with the resource utilization figure for the hand-coded RTL implementation at the outset of the evaluation process. Those compa-

nies likely used this figure as a target in optimizing their implementations. (We should note, however, that such information is not required for effective use of the high-level synthesis tools, and that the HLST vendors did not get the handwritten RTL design.)

We also interviewed designers who have used the AutoESL and Synfora high-level synthesis tools, who confirmed this

					Efficiency of Design Methodology				
	Out-of-Box Experience	Ease of Use	Completeness of Capabilities	Quality of Documentation and Support	Learning to Use the Tool	Design and Implementation (First Compiling Version)	Design and Implementation (Final Optimized Version)	Platform Infrastructure Development	Extent of Modifications Required to Reference Code
Combined HLST + Xilinx RTL tools rating	Fair	Good	Good	Good	Very Good	Very Good	Good	Good	Good
Texas Instruments software development tools rating	Good	Very Good	Very Good	Very Good	N/A (assuming already familiar)	Excellent	Good	Good	Fair

Table 2 – Usability metrics of HLST and FPGA tools vs. DSP development software



Development time has been a key impediment for many system designers trading off the use of a programmable DSP processor vs. an FPGA. Our evaluation indicates that this new approach involving high-level synthesis tools largely eliminates this barrier.

resource usage finding. Indeed, they reported that the tools produce excellent results comparable to those obtained via handwritten RTL code, with much less design and verification effort—a significant achievement.

Usability Metrics

Our usability metrics assess how easy it is to use the high-level synthesis tool flow compared with the DSP processor tool chain. For each usability metric, we assign a score of excellent, very good, good, fair or poor. In assigning these scores, we consider the overall design methodology for a complete project—starting with a C-language application specification and ending with a real-time implementation on the target chip (either an FPGA or a DSP processor). Table 2 presents the usability metrics.

In general, both PICO and AutoPilot were easy to install and use, even without expertise in FPGA design. In contrast, we had difficulty installing and using Xilinx's RTL tools, and we ultimately decided to bring in an experienced FPGA engineer to help get the design running on the FPGA. We needed the FPGA engineer to interpret error messages from the Xilinx RTL tools, for example, and to interface the HLST-generated RTL modules with I/O and memory modules to create a complete design that would run on the FPGA. In general, we found it challenging to resolve design problems uncovered outside the scope of the high-level synthesis tools. If the HLST user does not have RTL design and tools skills (as we didn't), at this stage of the design flow he or she will need the assistance of an engineer who does have them.

Even given the challenges associated with the RTL-to-bitstream portion of the flow, however, Table 2 indicates that the HLST-Xilinx tool chains yielded usability and productivity results that were nearly as good as those of the DSP processor flow. Overall, we found that it took a similar level of effort to implement the BDTI Optical Flow Workload on the TI DSP processor as on the Xilinx FPGA using either of the two HLSTs, assuming the availability of an experienced FPGA engineer to assist with some portions of the flow.

This is a significant finding, and may surprise many DSP software engineers. Development time has been a key impediment for many system designers trading off the use of a programmable DSP processor vs. an FPGA, and our evaluation indicates that this new approach largely eliminates this barrier for applications such as the BDTI Optical Flow Workload.

HLSTs: Game Changers?

Our earlier benchmarking of FPGAs and DSP processors (published in the 2007 report *FPGAs for DSP*) showed large performance and cost/performance advantages for FPGAs on some applications when the FPGA implementations were created using traditional RTL design techniques. The new analysis presented here confirms this performance advantage (for example, a 40x speed gain and 30x cost/performance advantage on the BDTI Optical Flow Workload), and shows that FPGAs can achieve similar performance and cost advantages when used with high-level synthesis tools. In addition, we found that the two high-level synthesis tools evaluated thus far—Synfora's PICO

and AutoESL's AutoPilot—achieved a level of resource-use efficiency comparable to that attained using handwritten RTL code. While we did not directly evaluate the time savings that came from using the HLSTs rather than writing RTL code by hand, we believe they will prove to be substantial, in part based upon our interviews with current HLST users.

FPGA designs created using traditional handwritten RTL coding typically take much more effort than the equivalent application implemented in software on a DSP processor—which is one good reason why many DSP processor users are reluctant to switch horses. Perhaps the most surprising outcome of this project, therefore, is that it took roughly the same effort to implement the evaluation workload on the FPGA (using either AutoPilot or PICO, plus the Xilinx tools) as it took on the DSP processor.

For FPGA users, our study indicates that HLSTs offer improved productivity with no significant downsides. For DSP processor users, it's clear that FPGAs are worth considering—with HLSTs quickly becoming a game-changing technology.

The authors wish to thank the many individuals at AutoESL, Synfora, Xilinx and BDTI who contributed to the evaluations described here. For more information on BDTI's evaluation programs and detailed results for the High-Level Synthesis Tool Certification Program, visit www.BDTI.com. For news and analysis of signal-processing technologies and tools, sign up for BDTI's monthly newsletter at InsideDSP.com.

Using Xilinx FPGAs to Speed Packet Processing

Virtex devices enable the programmable FAST processors to decode, inspect and modify packets with minimal CPU involvement.



by Andy Norton
 Distinguished Engineer, Office of the CTO
 CloudShield Technologies, an SAIC company
 ANorton@CloudShield.com

Next-generation network infrastructures are emerging as 10-Gigabit Ethernet matures and the industry looks ahead to 40GbE and 100GbE. Converged networks create new challenges for scalable open platforms to process the traffic. Common components in converged next-gen infrastructure chassis include high-performance terabit switching fabrics and programmable content processors capable of handling tens of gigabits of traffic at the application layer amid constantly increasing complexity and burgeoning applications. CloudShield has created a new class of programmable packet processors able to inspect, classify, modify and replicate packets, integrating dynamic interaction with the application layer.

Our Flow Acceleration SubsysTem (FAST) uses Xilinx® Virtex®-class FPGAs to preprocess packets for CloudShield Deep Packet Processing and Modification blades. These FPGAs include 10-Gbit Ethernet MACs with per-port ingress processors for classification and key extraction, egress processors for packet modification, packet queues using quad-data-rate (QDR) SRAMs, Xilinx Aurora-based messaging channels and a search engine based on ternary content-addressable memory (TCAM). Our FPGA chip set provides caching and processing of packets with minimal CPU involvement for high-performance processing at up to 40 Gbits per second. Featuring Layer 2-7 field-based lookups, it uses dynamically reconfigurable rules to modify packets at wire speed in a flexible and deterministic manner.

FAST Packet Processor Core Functions

Our currently deployed deep-packet processing blades use two blade-access controller FPGAs and a packet-switch FPGA, all implemented using LX110T Virtex-5 FPGAs. In each of our blade-access controllers we provide data-plane connectivity using two Xilinx 10GbE MAC/PHY cores, chip-to-chip interfaces based on Xilinx ChipSync™ technology and pack-

et processing using our intellectual-property (IP) cores. Our packet-switch FPGA uses a standard Xilinx SPI-4.2 IP core for interfacing to our network processor (NPU) and our IP core search engine.

We used standard Xilinx IP cores wherever possible in order to focus our system-on-chip design efforts on our packet-processing capabilities. We chose a Xilinx 10-Gbit Ethernet MAC core with dual GTP transceivers to implement the 4 x 3.125-Gbps XAUI physical-layer interface. For the NPU interface, we used a Xilinx SPI-4 Phase 2 core supporting up to 1 Gbps per LVDS differential pair with dynamic phase alignment and ChipSync technology. Our key packet-processing IP cores are as follows:

- **FAST Packet Processor:** Our FPP's Ingress Packet Processor (FIPP) is responsible for first-level packet parsing, key and flow ID hash generation, and Layer 3-4 checksum verification on a per-port basis. The FPP's Egress Packet Processor (FEPP) performs egress packet modifications and Layer 3-4 checksum recalculation.
- **FAST Search Engine:** Our FSE maintains a flow database in TCAM and QDR SRAM that we use to determine the processing actions that need to be performed on ingress packets. The FSE receives key messages from each port's FIPP, determines the processing actions for the packet and sends a result message back to the originating queue.
- **FAST Data Queue:** Our FDQ stores incoming packets in an "unscheduled" holding buffer. When the ingress packets are written to QDR SRAM, the queue forwards a key message from the FIPP to the FAST Search Engine. The FSE uses this key to determine how to process the packet, and sends a result message back to the FDQ. Based on the result message, the queue can forward, replicate or drop each buffered packet. The queue may additionally perform packet modification on the forwarded and replicated packets independently.

Ins and Outs of Data Flow

Figure 1 shows the data flow through our flow-acceleration subsystem. Core FPGA functions are shown in green, packet flow in yellow, control messages in blue, external devices in gray.

First, we identify customer traffic beginning with a received packet from a 10GbE network port. Packets on each port enter a FAST Ingress Packet Processor for packet parsing and analysis (No. 1 in the figure). After classifying protocols and encapsulations, the FIPP locates the Layer 2, 3 and 4 header offsets. Next come flow hashing and key extraction (flow selection lookup rules such as a 5-tuple using source and destination Internet Protocol addresses, source and destination ports and protocol).

At this point, our queue manager buffers receive packets to free memory pages in external QDR SRAM. Received packets at this stage are considered unscheduled. We park them in external QDR SRAM while waiting for FAST scheduling. The FAST Data Queue (No. 2 in the figure) assigns a packet ID and dispatches key messages to the FAST Search Engine (No. 3), which uses the key to identify the flow. A matched flow entry in external TCAM provides an index into our Flow Action Table in associated SRAM. The matched flow action is based on our customer's provisioned application subscriptions.

Our FAST Search Engine replies with a result message to the FDQ (No. 4), where our action scheduler assigns packets to an

output queue according to its specified action. We then de-queue packets from the packet queue to the indicated destination output port (No. 5), where our FAST Egress Packet Processor (No. 6) handles packet modification according to a provisioned rule in the Flow Modification Table as indicated by the specified action.

When our FAST Search Engine matches a customer flow, the specified action occurs, while a miss would result in the execution of a default rule (drop or send to NPU). We allow basic actions such as dropping packets, forwarding packets directly to a network port, forwarding them to our exception-packet-handling NPU or replicating and forwarding packets with independent rules. Our extended actions

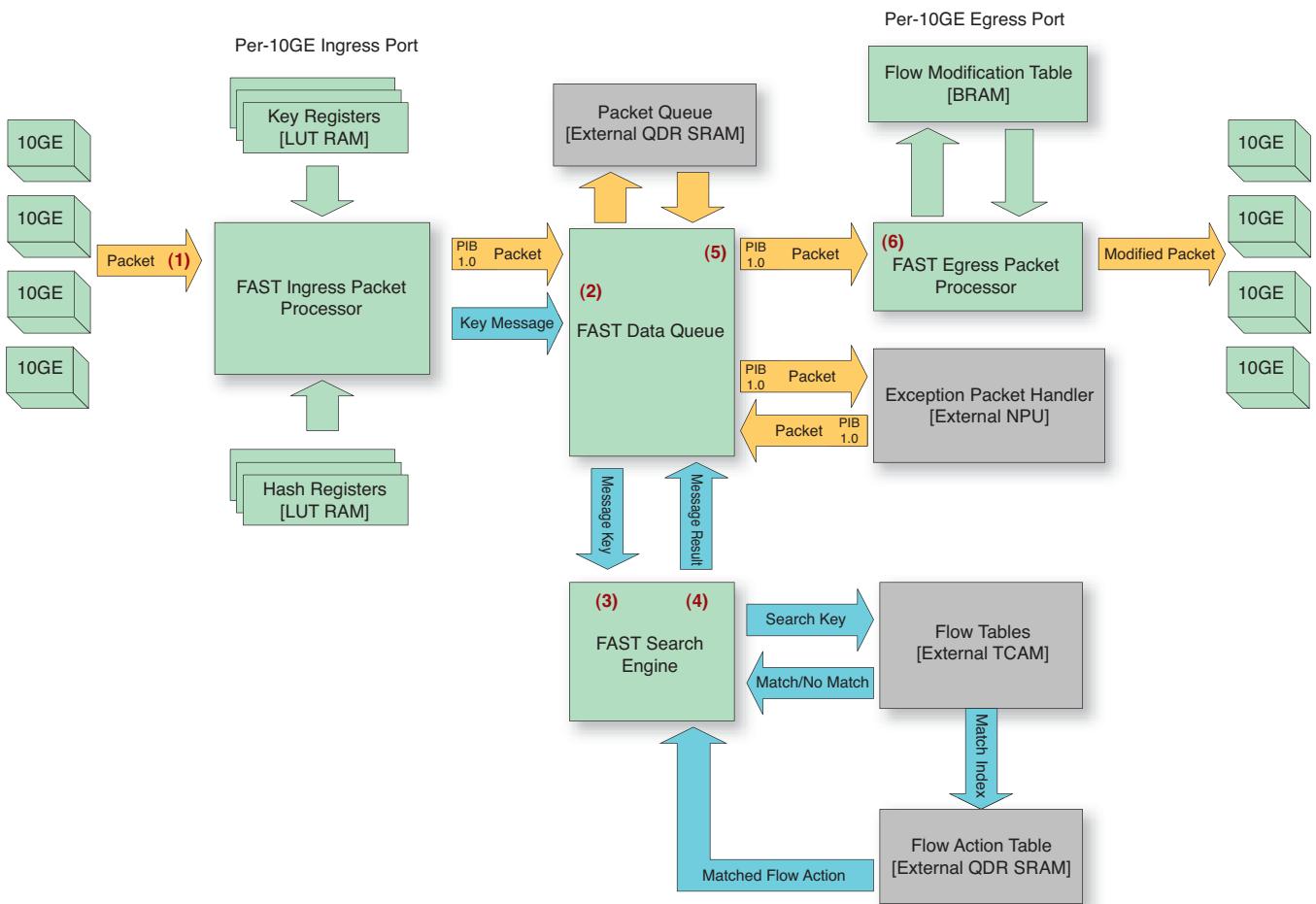


Figure 1 – Data flow in the Flow Acceleration Subsystem

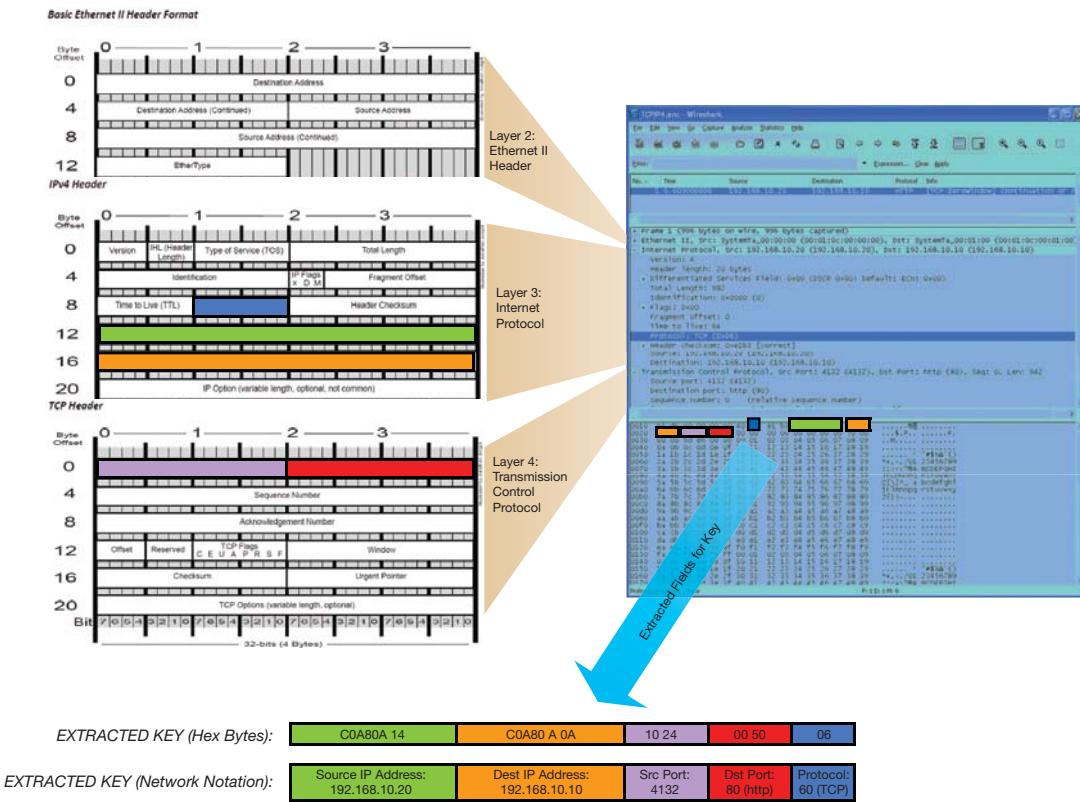


Figure 2 – 5-Tuple key extraction for TCP/IP packet over Ethernet Type II

include packet collapse (deleting a portion of a packet), packet expand/write (inserting a series of bytes into a packet) and packet overwrite (modifying a series of bytes) combinations. An example of an overwrite rule might be modifying a MAC source or destination address, modifying a VLAN inner or outer tag, or changing a Layer 4 header flag. An insert/delete example might be as simple as deleting an existing EtherType and inserting MPLS labels or VLAN Q-in-Q tags, or as complex as inserting an IP header as a GRE delivery header followed by the GRE header (Generic Routing Encapsulation is a tunneling protocol; see Internet RFC 1702).

FAST Packet Processors

Our FAST Ingress Packet Processor decodes all packets to determine the contents of Layers 2, 3 and 4, if present. After our initial Ethernet Layer 2 decoding, the packet may undergo further Layer 2 pro-

cessing. Next we proceed with Layer 3, processing either an IPv4 or IPv6 packet. Assuming we find one of these valid Layer 3 types, we continue processing at Layer 4.

At the same time the packets are being decoded, our key extraction unit is locating and storing key fields to produce a search key that our FAST Search Engine will use for flow lookup later. Figure 2 shows the format of a TCP/IP packet over Ethernet Type II and a standard 5-tuple key to be extracted. It also shows the resultant key extracted for this example.

We also perform IP, TCP, UDP and ICMP checksum calculations on all classified packets for both ingress and egress processors. Two Virtex-5 FPGA DSP48E slices provide the adders needed to calculate and verify the checksums. Our first DSP sums the data stream on 32-bit boundaries, and the second DSP folds the resulting sum into a 16-bit checksum value at the end of the associated layer being calculated. We then calculate the

checksums; for recalculation, we zero out the checksum byte positions of the incoming data stream and reinsert the inverse of the checksum result back into the data stream using a storage buffer. The pseudo header bytes required for the Layer 4 checksum are multiplexed into the incoming data stream for inclusion in the final calculation.

A FAST Egress Packet Processor at each output port performs rule-table-based (rules stored in internal BRAM) packet modifications and Layer 3-4 checksum recalculation and insertion. The FEPP expands beyond traditional packet modification “fixed-function” techniques, enabling packet modifications to include overwriting, inserting, deleting or truncating packets based on a specified modification rule number. Our flow modification rules allow specification of an opcode indicating the type of operation, with the OpLoc indicating the starting location, an OpOffset, Insert Size, Delete Size, whether to perform Layer 3 and 4

Our next-generation implementation will boost performance, extend caching capabilities and add new functionality. Migrating our FAST chip set into a single Xilinx Virtex-6 FPGA, we will take our next-gen FAST functionality, interfaces and performance to a new level while decreasing board space and power requirements.

checksum calculation and insertion as well as mod-rule chaining.

We can use packet overwrite features to simply modify existing fields such as a MAC destination address, MAC source address, VLAN tag or even a single TCP flag.

In order to modify only the MAC destination address, the “action” that the FEPP receives along with the packet would be to use, for example, Rule 2 in the Flow Modification Table (Figure 3). Rule 2 would have been preprovisioned to specify

Another example of overwrite is shown for Rule 6, where we want to modify a particular TCP flag, possibly the ACK, SYN or FIN (Figure 4). This rule would use opcode (overwrite), OpLoc (Layer 4), OpOffset (0 offset from Layer 4), mask type (use byte 14) and BitMask (which bits within a byte to mask). We could have specified multiple fields for the overwrite, using mask types to include or exclude specific bytes.

Our overwrite capabilities are not limited to only what we can store in the Flow

Our insert/delete capabilities can achieve even more-complex packet modifications. The example for Rule 5 (Figure 5) uses our insert/delete capability. The various actions—including opcode (insert/delete), OpLoc (Layer 2), OpOffset (start at byte 12), ISize (insert size = 22 bytes), DSize (delete size = 2 bytes) and Insert Data (0x8847, MPLS labels)—associated with Rule 5 will result in the deletion of the existing EtherType and insertion of the new EtherType=8847, indicating that the new packet will be an MPLS unicast packet, followed by a stack of MPLS labels as specified by the insert data.



Figure 3 – Simple MAC destination address overwrite modification



Figure 4 – Overwrite modification of TCP flags

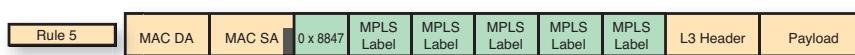


Figure 5 – MPLS label insertion modification

the opcode (overwrite), OpLoc (location within the packet, for example Layer 2), OpOffset (offset from starting location), mask type (which bytes to use) and modification data (actual overwrite data). The result would be to overwrite the 6 bytes, starting at location Layer 2, with the pre-provisioned modification data.

Modification Table, but also include what is stored in the Flow Action Table as associated data. A rule could specify the use of associated data that passes to the FEPP as part of the action, significantly expanding the range of data that can be used for the modification. The result is to allow, for example, overwrites for an entire VLAN tag range.

Floorplanning and Timing Closure

The greatest challenge we faced in designing our novel packet processor arose from increasing FPGA design complexity, higher routing and utilization density, the integration of various IP cores, use of multiple hard logic objects (BRAMs, GTPs, DSPs and the like) and insufficient data-flow planning back at the earliest project phases. Our released Phase 1 Virtex-5 FPGA bit files were based on lower utilization density, particularly lower BRAM usage, resulting in relatively easy timing closure. Adding significant new functionality with BRAM utilization approaching 97 percent at a later phase, we became painfully aware of just how critical optimal floorplanning can be, and how decisions made early in a product life cycle can affect later phases.

The primary goal of floorplanning is to improve timing by reducing route delay. In order to accomplish this, design analysis considering data flow and pinout is of paramount importance. The Xilinx PlanAhead™ tool, now integrated into

ISE® as the point tool for floorplanning and timing analysis, provided the interactive analysis and visualization we needed to navigate the complex web of how to achieve timing closure on a high-utilization design. PlanAhead supplied the insight into our design that we needed to provide the minimal number of constraints so as to guide the map, place and route tools to meet our timing requirements. We have found that this often means optimally placing a key set of BRAMs in addition to block-based design area constraints.

In retrospect, had we spent more time at the earliest project stages, using PlanAhead to try what-if scenarios and help us visualize the optimal data-flow and pin selections, our task at this late design phase would have been less difficult.

Dynamically Adaptable Packet Processing

Our Flow Acceleration SubsysTem's ability to inspect and modify packets at wire speed with maximum flexibility, combined with the ability to dynamically interact with application-layer services, results in highly adaptable packet processing. Virtex-class FPGAs have been a key enabler, providing the system-on-chip platform to accelerate content-based routing and implement key packet-processing functions previously unseen in prior-generation FPGAs.

Our next-generation implementation will boost performance, extend caching capabilities and add new functionality. Migrating our FAST chip set into a single Xilinx Virtex-6 FPGA, we will take our next-gen FAST functionality, interfaces and performance to a new level while decreasing board space and power requirements, resulting in a single-chip deep-packet-processing coprocessor unit. 

Acknowledgements

No complex FPGA design effort such as ours could happen without a great team. I'd like to acknowledge our FPGA dream team: Greg Triplett, FPGA team lead and search engine design lead; Scott Stovall, data queue design lead; Scott Follmer, packet-processing design lead; Steve Barrett, verification team lead; and Isaac Mendoza, SystemVerilog whiz and verification engineer.

GET PUBLISHED



WOULD YOU LIKE TO WRITE FOR XCELL PUBLICATIONS?

It's easier than you think!

Submit an article draft for our Web-based or printed publications and we will assign an editor and a graphic artist to work with you to make your work look as good as possible.

For more information on this exciting and highly rewarding program, please contact:

Mike Santarini
Publisher, Xcell Publications
xcell@xilinx.com

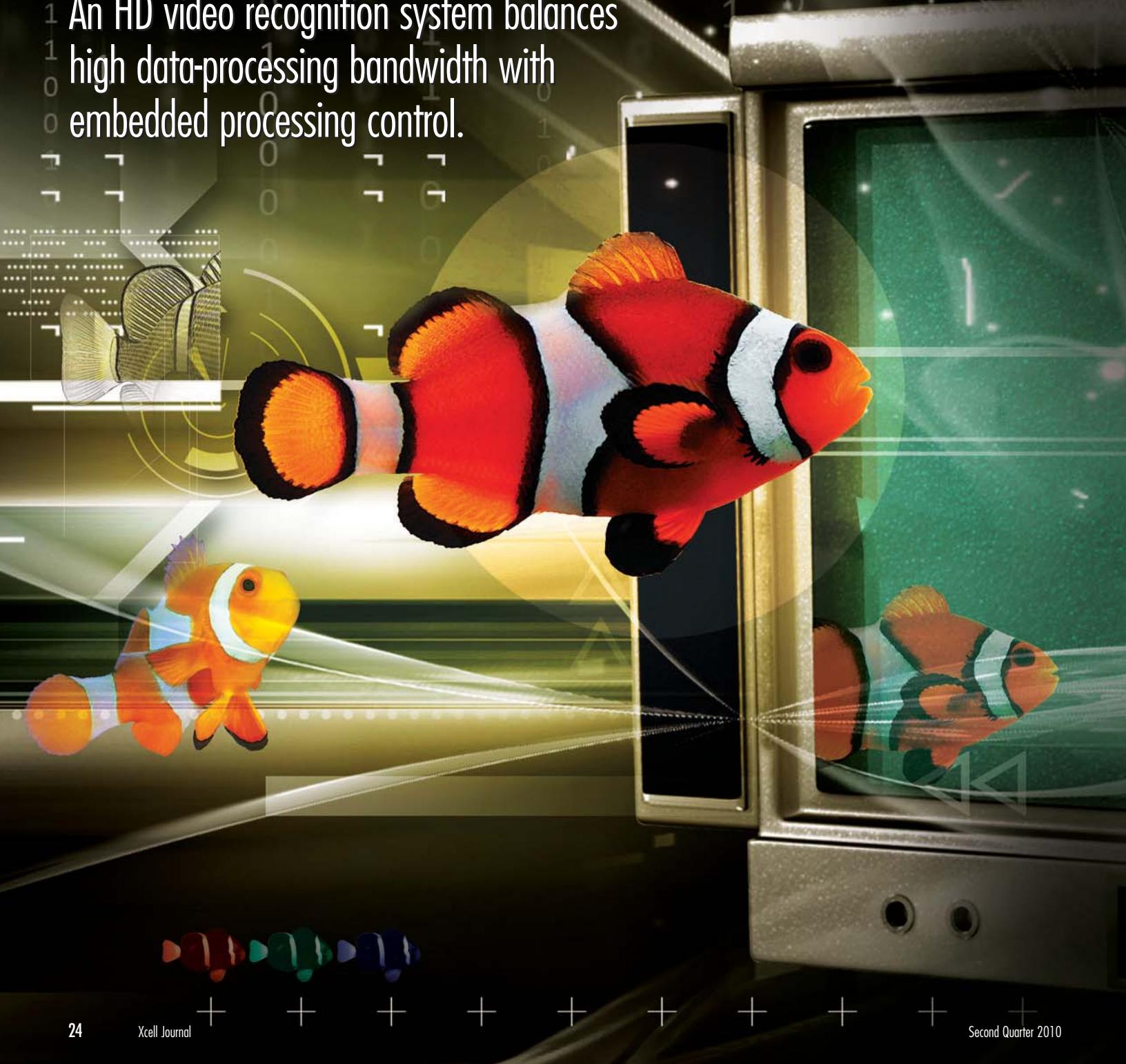


See all the new publications on our website.

www.xilinx.com/xcell

Control Plane/Data Plane Video Processing with an FPGA

An HD video recognition system balances high data-processing bandwidth with embedded processing control.



by Glenn Steiner

Senior Manager for Embedded Processing Solutions
Xilinx, Inc.

Dan Isaacs

Director of Technical Marketing
Xilinx, Inc.

David Pellerin

Founder and Chief Technology Advisor
Impulse Accelerated Technologies

One of the greatest challenges facing an embedded-systems designer is scoping the system's performance requirements. The information needed to identify the true performance requirements may not be available or may be difficult to obtain. The best estimates sometimes fail due to additional unanticipated computational burdens. At other times, the analysis can indicate that an embedded processing system is not cost-effective for the data-processing demands. As a result, a systems designer finds it highly desirable to have an architecture that is scalable to match the potentially changing performance requirements and to handle high-performance data processing. A control plane/data plane processing architecture implemented inside an FPGA can meet these requirements.

What is control plane/data plane processing, and why might you need it for your next embedded system?

In systems where it's impossible or impractical to do all the processing in software, designers can obtain additional performance in a variety of ways. They can use multiple processors in symmetric or asymmetric processing configurations; implement hardware coprocessors; or split the data-processing tasks off completely into one or more dedicated processing elements—as in control plane/data plane processing.

In this method of programming, the data processing is divided into two distinct planes. The control plane represents algorithm elements that are not performance critical, such as administrative tasks, user interfaces and operating system functionality. Meanwhile, the data plane represents the movement of data through the system, for example, a video or audio stream, as well

as its processing. In the data plane, designers use techniques such as pipelining to increase data throughput. Typical applications for control plane/data plane processing include streaming video, network packet processing and high-speed signal processing.

Let's look more closely at a control plane/data plane application that involves real-time processing of streaming data. To

process. A highly parallel processing element, an FPGA in this example, handles the video processing while a medium-performance processor inside the FPGA manages the video-processing pipeline. The processor might be dedicated to a single application, or it might be running an operating system such as Linux. The resulting mixed hardware/software implementa-

High-definition video processing represents a common real-world application that is solved very effectively by splitting the problem into control plane and data plane processing.

do that, we confront the challenge of identifying a unique pattern in a high-definition (HD) video stream. This example is representative of many applications that require a mix of high-performance data processing and control functions involving an embedded microprocessor.

A 720p, 60-Hz HD video stream operates at a 74.25-MHz pixel rate. This represents a required processing rate of 222.75 Mbytes/second. If a hypothetical dual-core, dual-issue processor operating at 2.5 GHz were to process this data, the optimal instruction rate would be 10 giga-instructions per second. Such a processor would be capable of executing 22.4 instructions per byte of data processed. For some applications this may be sufficient, but 22.4 instructions represents very little data processing. Complex video-processing functions such as kernel convolutions, noise reductions and other filtering takes many more instructions per second. The solution is to create parallel or pipelined processing elements in the data plane.

HD video processing represents a common real-world application requirement that is solved very effectively by splitting the problem into control and data plane

distribution distributes the processing to where it can be best handled and results in a low-cost, high-performance data-processing solution. Figure 1 shows a typical control plane/data plane system.

FPGAs Enable Computation Load Balancing

Short of using an expensive ASIC, FPGAs are the highest-performance and most cost-effective method of implementing streaming data-processing elements. FPGAs, due to their flexible architecture, enable hardware designers to implement processing systems consisting of elements that are both parallelized and pipelined. This allows designers to tune a system for both performance and latency.

Designers can then couple this data plane solution to an external, discrete microprocessor for control. Having that processor inside an FPGA presents several advantages. An internal processor dramatically reduces the control latency between the processor and the data plane elements. Such latency reduction can amount to many processor cycles. An external processor must communicate with the data plane. The communication channel may be 32 or more bits with additional wires for address

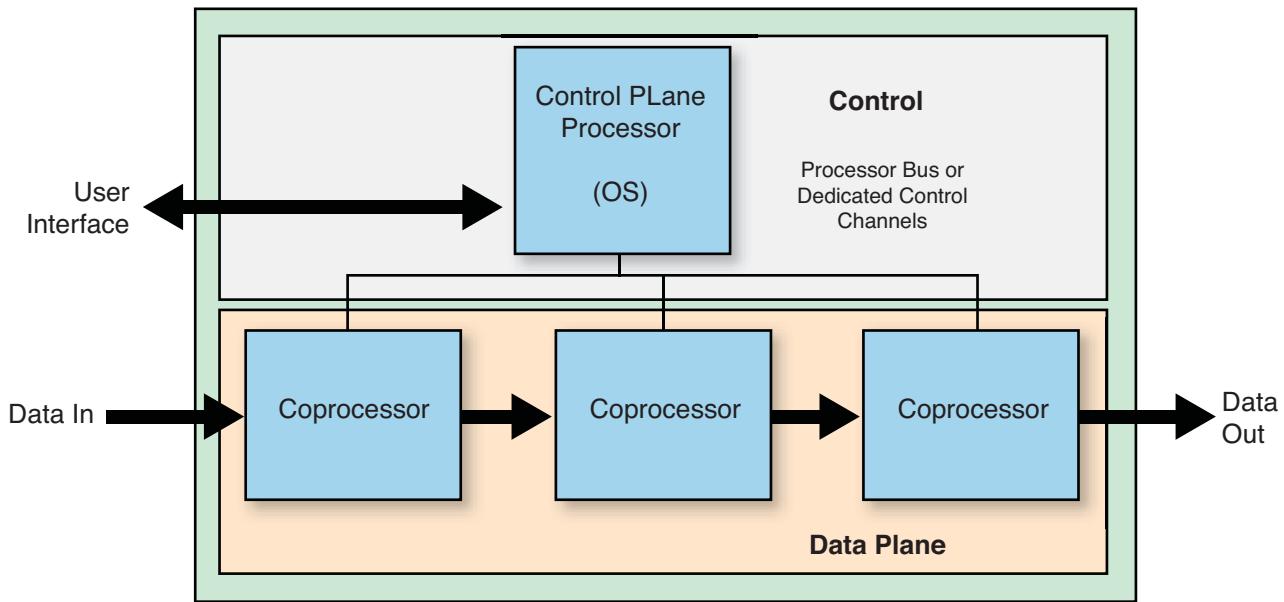


Figure 1 – Typical control plane/data plane processing system

and control. These additional wires may necessitate a larger package for both the processor and FPGA, driving up system cost. Alternatively, PCI Express® (PCIe) can provide a dramatic reduction in the number of pins. Unfortunately, not all processors and FPGAs support this relatively new interface and even when available, PCIe components are more costly than equivalent parts without PCIe.

Implementing a control plane processor and the data plane within an FPGA reduces parts count, board space and, frequently, power. The result can be a lower-cost solution. Hardened implementations of processors such as the PowerPC®, or soft implementations such as the Xilinx MicroBlaze™ processors, are available in FPGAs. FPGA-based processors may also be configured based upon application requirements. FPGA-based systems enable system-level tuning by providing the ability to move decision and computational functions between the processor and the FPGA logic.

Implementation of a Control/Data Plane System

Certain tools can simplify the implementation of an FPGA-based control plane/data plane system. Two common approaches are

to assemble the system using wizards or to do so by modifying an existing reference design.

FPGA tools allow the rapid assembly of a microprocessor system via wizards. Using drop-down lists or checkboxes, you simply specify the targeted part and the desired processor and peripherals. Similarly, you may use tools such as MATLAB® software to rapidly assemble a signal-processing pipeline with interfaces to the processor bus for control. Alternatively, you can construct a digital signal-processing pipeline via C-to-HDL tools. The control plane/data plane system can be connected by simply matching bus interfaces. Figure 2 presents the introductory window to start the wizard as well as the final system the wizard builds.

The second method involves modifying an existing reference design. FPGA reference designs continue to evolve and are becoming market focused. The one we used in our example case study has a complete microprocessor system with memory and peripherals, as well as a 720p HD digital signal-processing pipeline. Thus, this system represents a complete control/data plane solution. The reference design demonstrates the processor controlling the gain and FIR filter in the pipeline. Using C-to-FPGA tools to create the object detection and highlighting module,

the complete system was functional in less than 20 hours of effort.

The processor may control the data pipeline by using supplied drivers that board support packages provide. Drivers are now available for Linux, enabling processors to directly control the data-processing pipeline. Linux calls consist of opening the I/O device from the Linux application and then reading or writing to the device.

Case Study of an HD Video Recognition System

Object detection and recognition are of interest in industries ranging from surveillance to medical imaging and factory automation. The higher the image resolution, the more accurate the object recognition. Thus, HD video cameras and the associated HD video stream processing capabilities are highly desirable. Our case study starts with the question (inspired by a famous animated movie): can we detect and highlight a clown fish in a 720p HD video stream?

The design requires a color pattern match across 16 bits to recognize the clown fish stripe pattern. Once recognized, the fish is to be highlighted with a moving spotlight on the display. Further, the size of this spotlight is designed to grow or shrink depending on the likelihood of a match (in

reality, the system reduces image brightness in all areas except for the spotlight around the fish). The spotlight sizing and shape calculations, and the comparisons required to search for the fish at each pixel location, represent a significant amount of computation to perform at every 74.25-MHz clock cycle. It is quickly determined that the processing requirements are well beyond the capabilities of a typical embedded processor.

In such situations, it is best to offload the streaming-data processing to a coprocessor. Implementing a coprocessor in an FPGA provides the flexibility to architect a solution that meets performance requirements at the lowest system cost. As a result, an FPGA-based control/data plane architecture is the best choice. An FPGA embedded processor controls, via a bus interface, the digital signal-processing pipeline that is responsible for receiving the video data, detecting the fish, highlighting the fish and outputting the video data for display.

Thus, for this object detection and highlighting example, we chose a MicroBlaze embedded processor operating at 50 MHz to manage and control a data-processing pipeline operating at 74.25 MHz, and to manage the user interface. Freed of

the burden of performing actual video processing, the processor can handle many other functions, such as host data communication via Ethernet, graphical user interface management and fine control of the data-processing pipeline (for example, gain control on a frame-by-frame basis).

An operating system such as Linux is ideal for providing the multitasking capabilities, network stack and language support for user interfaces. Figure 3 shows the block diagram of the implemented system. This solution enables an optimized balance between the need for high data-processing bandwidth and software control of how the data is processed.

HW/SW Co-design with C-to-FPGA tools

C-to-FPGA compilers let developers tackle software/hardware development using a new set of development tools and new techniques. Developers can start by coding their algorithm in software. Experience tells us that developing algorithms in software is more efficient than developing in hardware for several reasons. First, a software language such as C enables programmers to develop algorithms at a higher level than they can when using hardware definition languages like Verilog or

VHDL. Next, the debug and test tools for C tend to run faster and more effectively, and typically are easier to use, than corresponding hardware development tools. C algorithms run at full speed on a target processor compared with hardware algorithms, which are initially tested and debugged on a simulator. Finally, C development tools tend to be significantly less expensive than their hardware counterparts. Thus, engineers generally prefer to develop algorithms in C or a similar high-level language.

Once an algorithm has been proven using a software language such as C, designers must measure its performance and determine whether the algorithm can run entirely on an embedded processor or entirely in hardware, or if a mixed hardware/software coprocessing implementation is best. Performance analysis tools are available to aid in such determination. If the code must be converted to hardware, the designer must either convert the algorithm by hand or use a C-to-FPGA tool.

C-to-FPGA tools enable developers to rapidly convert algorithms to HDL, optimize the generated hardware processor and perform what-if scenarios balancing performance and FPGA resources. They

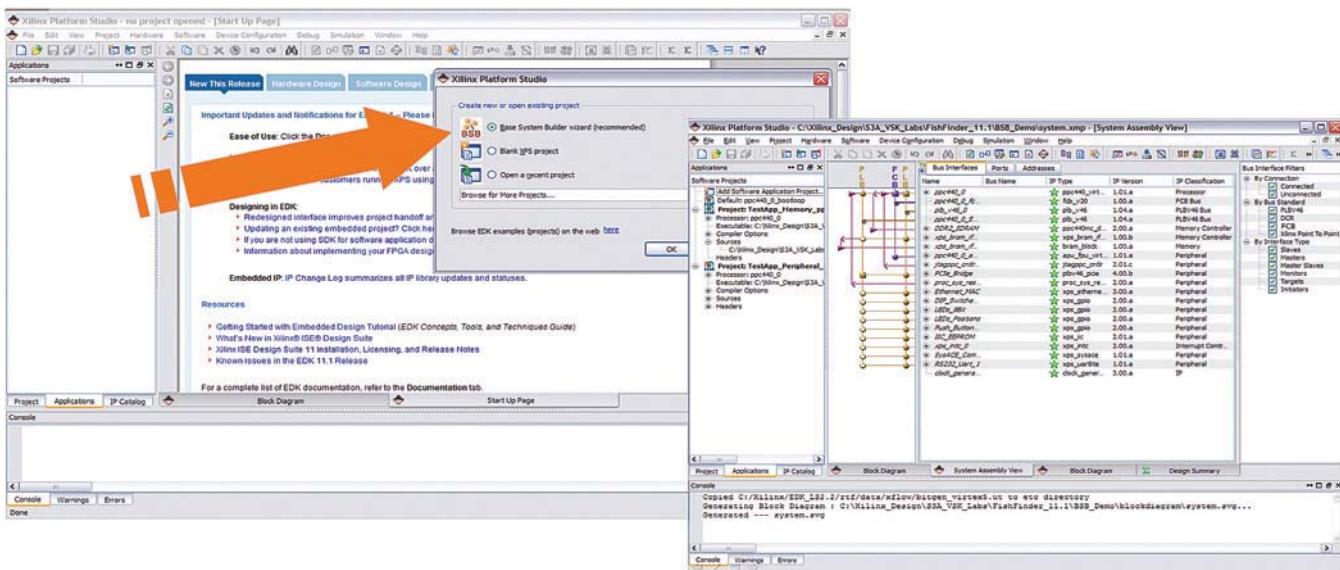


Figure 2 – Wizard start screen and completed system

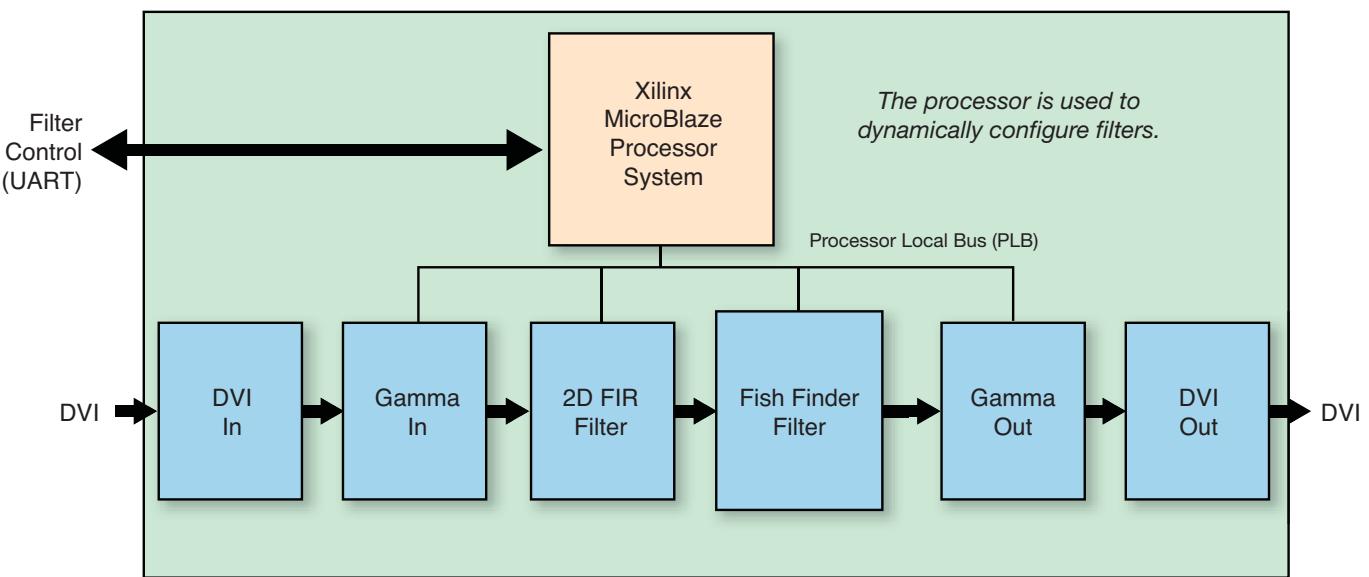


Figure 3 – Clown fish finder control plane/data plane system

also enable software engineers to become hardware engineers, giving them access to the high-performance data-processing logic within an FPGA.

Connecting Processor to FPGA with Linux

Linux suppliers working with FPGA manufacturers have developed drivers that enable processors to communicate with and control FPGAs. First, you must configure Linux for the I/O device. This is a two-step process. To begin, load the custom driver into the Linux kernel:

```
module_init(xll_example_init);
```

then, register the driver to a specific device number (for example, 253):

```
err =
register_chrdev_region(devno, 1,
"custom_io_example");

bash# mknod /dev/custom_io_exam-
ple0 c 253 0
```

Communication is accomplished by opening the I/O device and then reading or writing to the device, as shown by these sample code segments:

```
// Open custom I/O device from
```

Linux application

```
int custom_io_ex_open(struct inode
*inode, struct file *filp);

// Read / Write to custom peripheral I/O using standard Linux
read/function function calls

ssize_t custom_io_ex_read(struct
file *filp, char __user *buf,
size_t count, loff_t *f_pos);

ssize_t custom_io_ex_write(struct
file *filp, const char __user *buf,
size_t count, loff_t *f_pos);
```

The FPGA Advantage

Signal-processing systems frequently have data bandwidth requirements exceeding that which can be economically achieved via general-purpose processors. In such cases, designers commonly split their data-processing system into two processing functions, using a general-purpose processor for control processing and a hardware accelerator, such as an FPGA, for the data processing. This constitutes a control plane/data plane processing system.

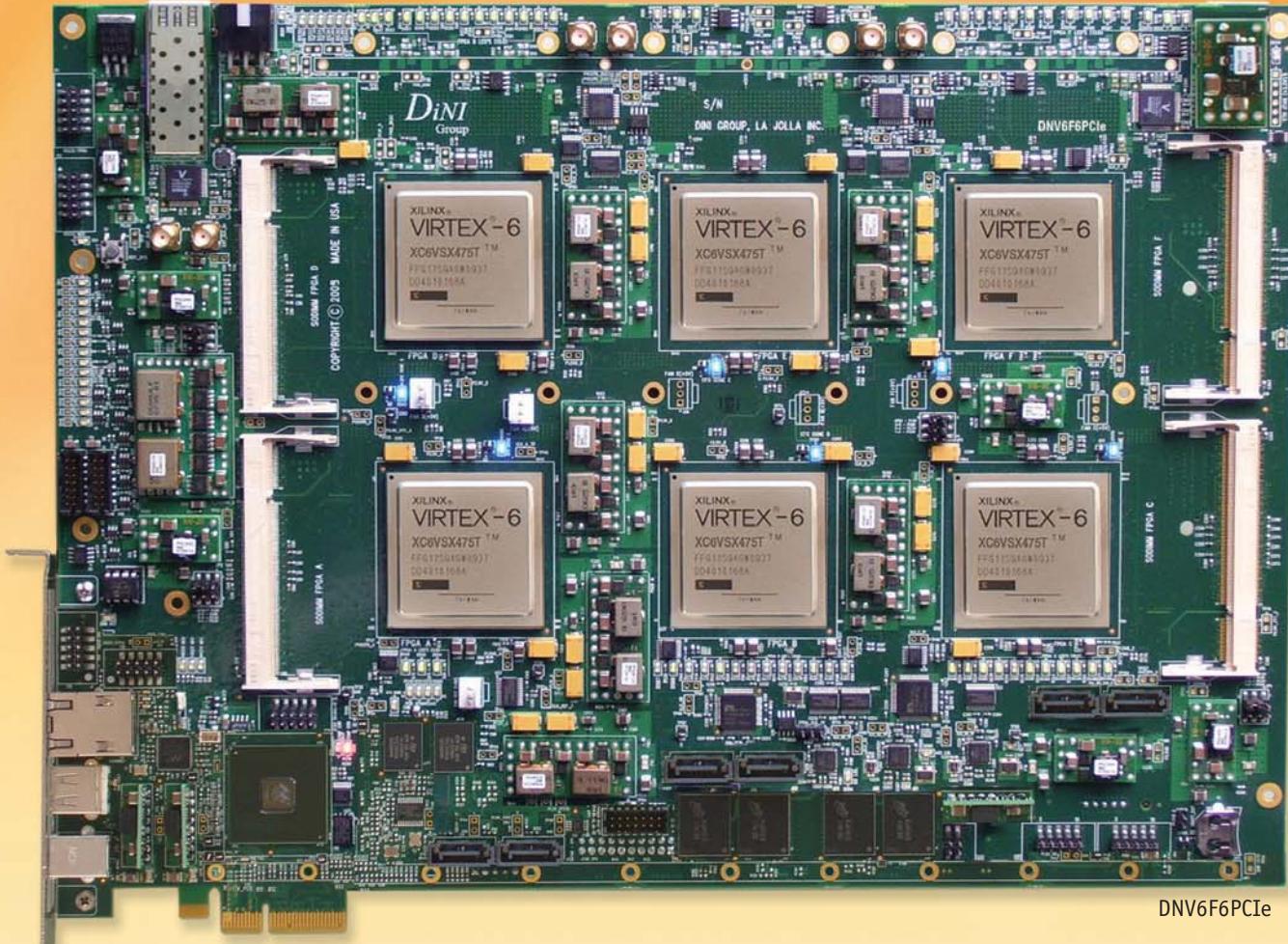
FPGAs are ideal for implementing both control and data plane functions.

An FPGA may contain one or more soft processors such as the MicroBlaze and/or hard processors such as the PowerPC. Integrating them into the FPGA enables low-latency and high-bandwidth communication between the control plane processor and the data plane processing system.

Assembling such systems for both the embedded and data-processing functions is straightforward with wizards and pre-built reference designs. C-to-FPGA tools help streamline this process by converting algorithms prototyped in C to high-performance hardware processing elements. Finally, Linux drivers are now available, enabling easy coding of communication and control between the processor and the FPGA signal-processing pipeline.

Our case study is a typical example of an application where processing an HD video stream is not practical with a low-cost general-purpose processor but is easily accomplished via a signal-processing pipeline inside an FPGA. The processor is then freed up to provide user interface, networking and system management functions while monitoring and controlling the signal-processing pipeline.

NEW Virtex-6X6 — More Power and Speed for ASIC Prototyping and High Performance Computing



DNV6F6PCIe

Six powerful Virtex®-6 FPGAs, up to 24 Million ASIC gates, clock speeds to 710 Mhz: this new board races ahead of last generation solutions. The Dini Group has implemented new Xilinx V6 technology in an easy to use PCIe hosted or stand alone board that features:

- 4 DDR3 SODIMMs, up to 4GB per socket
- Hosted in a 4-lane PCIe, Gen 1 slot
- 4 Serial-ATA ports for high speed data transfer
- Easy configuration via PCIe, USB, or GbE
- Three independent low-skew global clock networks

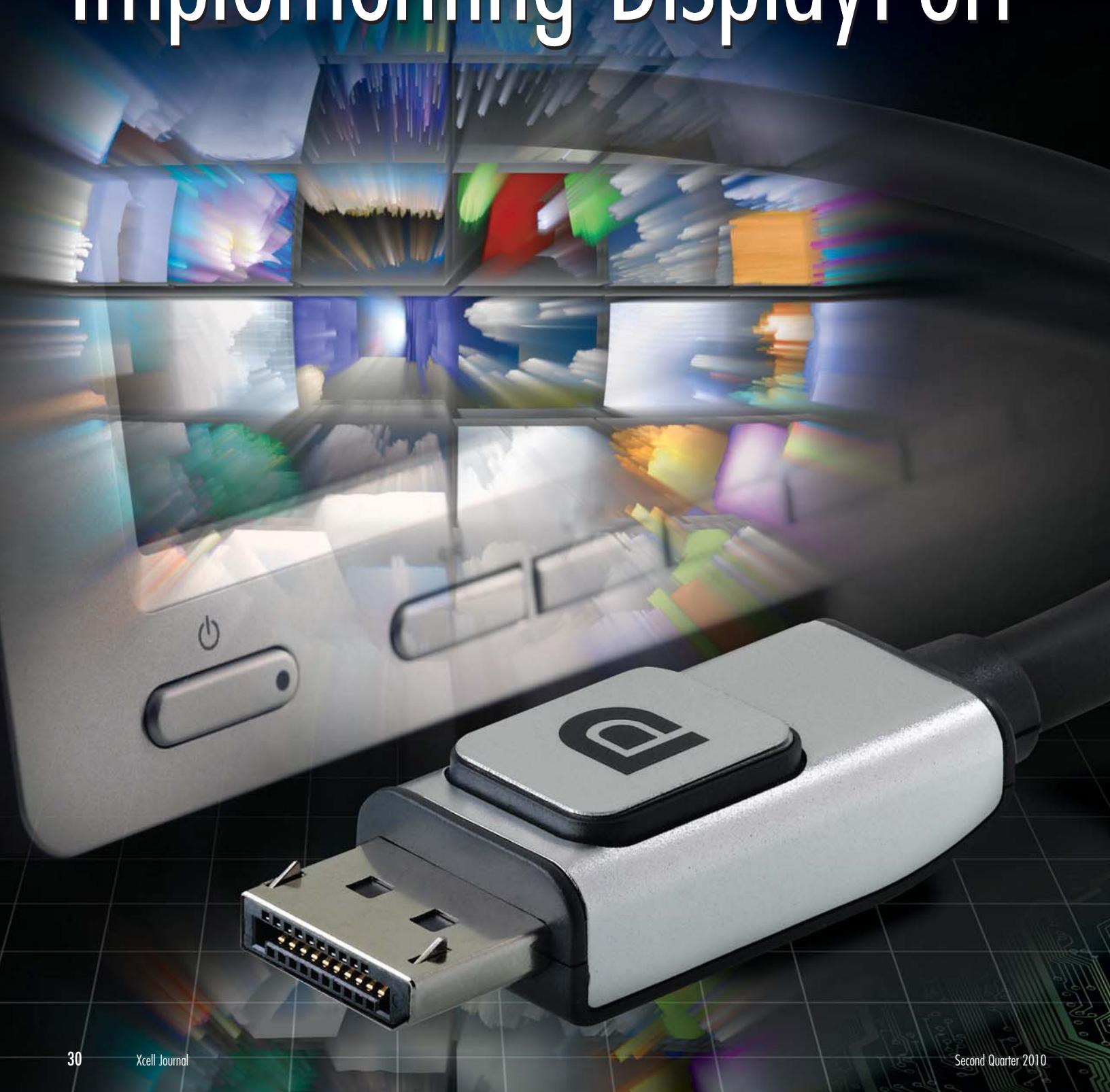
The higher gate count FPGAs, with 700 MHz LVDS chip to chip interconnects, provide easier logic partitioning. The on-board Marvell Dual Sheeva processor provides multiple high speed interfaces optimized for data throughput. Both CPUs are capable of 2 GFLOPS and can be dedicated to customer applications.

Order this board stuffed with 6 SX475Ts—that's 12,096 multipliers and more than 21 million ASIC logic gates—an ideal platform for your DSP based algorithmic acceleration and HPC applications.

Don't spin your wheels with last year's FPGAs, call Dini Group today and run your bigger designs even faster.

D
N
I
G
R**O****U****P**

An FPGA Route Toward Implementing DisplayPort



There's a simpler way to design whizzy new 3-D TVs. The Xilinx Spartan-6 FPGA Consumer Display Kit and IP will get you there.

by Carol Fields

Senior Staff Product Marketing Manager

Xilinx, Inc.

Carol.Fields@xilinx.com

Neal Kendall

Marketing Manager

Quantum Data, Inc.

At the Consumer Electronics Show in January, several major flat-panel TV and display companies introduced high-definition 3-D-enabled TVs and monster 4K x 2K LCD monitors, dramatically upping the ante on the number of bits that will be flying between your TV, display and other electronics in your home, car or mobile device. With these new TVs, sports fans will be leaping for joy over features like a 176-degree field of view, 1,200:1 contrast ratio and 450 nits of brightness—more than enough to penetrate even the darkest man cave.

But for design engineers creating these TVs or electronics that will connect to them, all the new features translate into one hefty bandwidth requirement. For example, a quad 4K x 2K HDTV with 8 megapixels (providing digital cinema quality for the home) would consume four times the bandwidth required for optimal operation of today's top-line TVs and monitors. That translates to a lot of bits racing between the set-top box and the HDTV.

Nor is this desire for more bandwidth limited to the consumer market. The broadcast equipment, digital display, scientific and medical markets continue to drive the need for bandwidth for displays in application such as MRI and CT scans, command and control, daisychained displays, electronic billboards and 3-D rendering of DNA, aircrafts, weather and various parts of the human body.

To help meet this bandwidth demand while controlling the cost, the Video Electronics Standards Association introduced DisplayPort into the market in 2007 and has since worked diligently with partners to refine it. Today, VESA DisplayPort 1.1a supports a data rate of 2.7 Gbits/second per channel with up to four channels in a single cable, while DisplayPort 1.2 doubles the data rate to 5.4 Gbps (enough to handle, for example, 3,840 x 2,400 pixels at 60 Hz, or four monitors at 1,920 x 1,200 or, alternatively, a 3-D display at 120 Hz and 2,560 x 1,600 pixels). DisplayPort supports both embedded displays, such as those within a laptop computer, and box-to-box connections between a video "source" device (set-top boxes, DVD players, PC graphic cards, laptops) and a separate display device, commonly referred to as "sinks" in HDMI and DisplayPort standards documents.



Figure 1 – TED Spartan-6 FPGA Consumer Video Kit

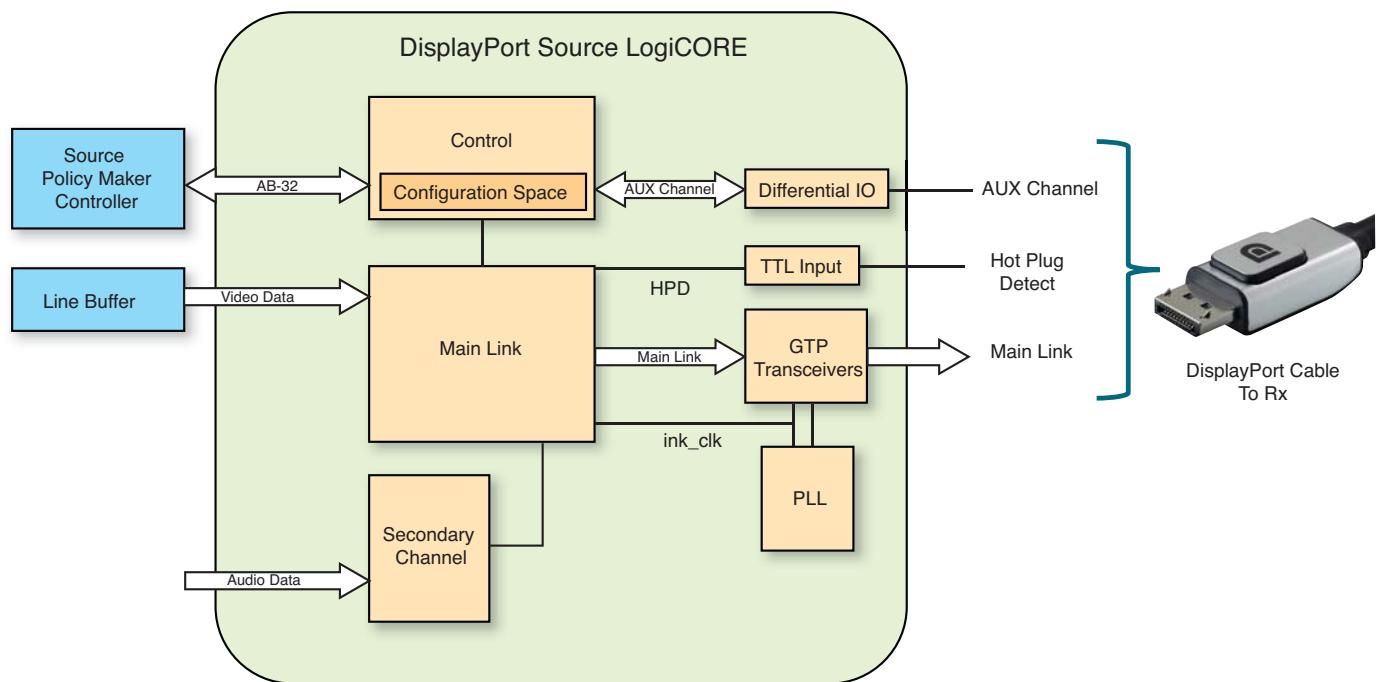


Figure 2 – DisplayPort Source Policy Maker Controller System Reference Design and LogiCORE Source High-Level Block Diagram

While some chip makers have introduced standard off-the-shelf transmitters and receivers for these applications, Xilinx has launched a flexible programmable version of the VESA DisplayPort v1.1a solution called the Xilinx LogiCORE™ DisplayPort v1.1 (Version 1.2 is coming in IDS 12.1). This IP is readily available for Xilinx customers, but before you get started, it's wise to review some additional background information about some of the key functions of the standard, such as the Policy Maker, and how you can implement them in Xilinx® FPGAs using our upcoming XAPP "Implementing a DisplayPort Source Policy Maker Controller System Reference Design using a MicroBlaze™ Embedded System" in the Tokyo Electron Devices (TED) Spartan®-6 Consumer Video Kit (<http://www.teldevice.co.jp/engl/>).

Policy Maker—A Key Difference

The DisplayPort protocol marks a significant change in connectivity technology for the display market. This transition is similar

to the Intel-led PC market's migration from the parallel PCI bus to serial PCI Express. For the display market, VESA is leading the migration from protocols like VGA, DVI and HDMI to a high-speed serial-transceiver, packet-based layer architecture protocol with DisplayPort. Unlike parallel protocols, serial packetized protocols have an added degree of complexity to achieve and maintain the connection or link. In the VESA DisplayPort 1.1a specification, the control functions are grouped as the Link Policy Maker and the Stream Policy Maker. The Link Policy Maker manages the link and is responsible for keeping the link synchronized. Its jobs include link discovery, initialization and maintenance. The Stream Policy Maker manages the transport initialization and maintains the isochronous stream by controlling sequences of actions by the underlying hardware.

These elements of the Policy Maker are implementation specific and may be handled within the operating system, software driver, firmware or FPGA logic. Many

commercial DisplayPort ICs hide the Link and Stream Policy Maker implementation from the designer to simplify usage. If your display requirements match an off-the-shelf DisplayPort ASSP, the price and ease of use are hard to beat. However, designers looking to differentiate their products from the competition turn to FPGAs.

Source Policy Maker Reference Design

The DisplayPort Source Policy Maker Controller System Reference Design using a MicroBlaze Embedded System implements functionality similar to that of a commercial off-the-shelf DisplayPort chip with the added advantage of being available in source code for customization. By using the Source Policy Maker Controller System Reference Design application note, you don't need to learn the details of the Policy Maker to get started, just connect up the example design and go.

In addition to this source code design, the DisplayPort transmit (Tx) or source

For the display market, VESA is leading the migration from protocols like VGA, DVI and HDMI to a high-speed serial-transceiver, packet-based layer architecture protocol with DisplayPort.

core comes with additional example designs for a finite state machine (FSM) controller,

The DisplayPort Tx FSM controller example design (which has the top-level file name dport_tx_fsm_cntrl) comes with a DisplayPort LogiCORE source design example. This simple proof-of-concept design contains an RTL-based finite state machine to implement a simple Policy Maker that demonstrates the proper start-up procedure. The dport_tx_fsm_cntrl design example has the benefit of requiring less time to simulate than the other sample designs.

The Source Policy Maker Controller System Reference Design using a MicroBlaze Embedded System XAPP, which is due out in late May, has the top-level ISE® project name dport_source_ref_design.xise (you will find it soon at <http://www.xilinx.com/products/ipcenter/EF-DI-DISPLAYPORT.htm>). This design allows you to modify the Source Policy Maker Controller source code for your own needs. It works in conjunction with the DisplayPort LogiCORE v1.2 (IDS 12.1) release and the Spartan-6 TED Consumer Video Kit.

These two example designs contain the basic procedure for setting up the cores and maintaining the link and stream. Note that the TED Spartan-6 Consumer Video Kit does not include a DisplayPort cable.

Functional Overview

Both source and sink/display specifications use the Policy Maker; however, Xilinx has implemented them differently with respect to the DisplayPort LogiCORE. The Policy Maker function on the sink (Rx) side is much simpler than that of the source (Tx) side. Xilinx LogiCORE implemented most

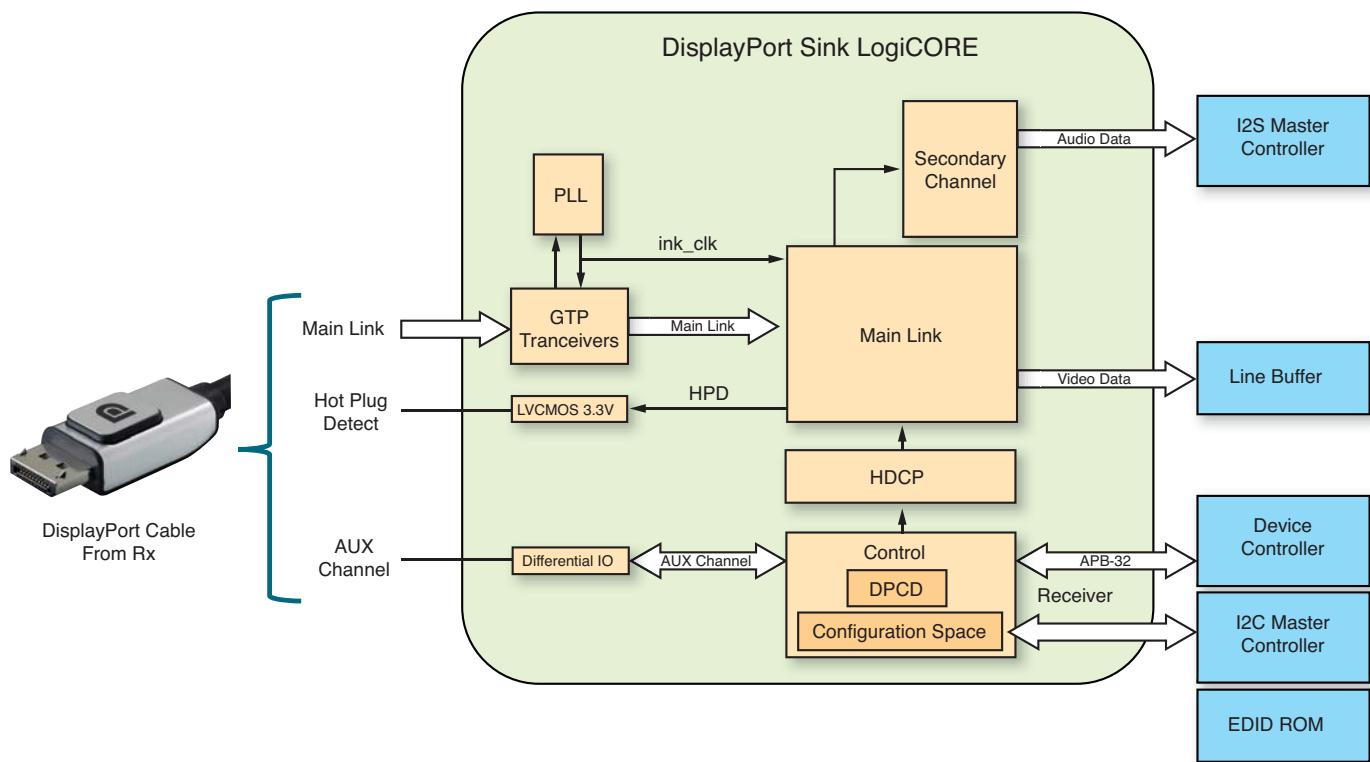


Figure 3 – DisplayPort Rx High-Level Block Diagram

What if EDIDs Did not Exist?

Designers take for granted the automation that Enhanced Display Identification Data structures provide. One way to get a sense of their importance is to imagine what life would be like if EDIDs did not exist. In modern home theater environments, the absence of EDIDs in a sink device such as an HDTV, audio/video receiver or video processor would mean that the consumer would need to review the specifications of these devices and learn their capabilities. The user would then have to set the audio and video formats to ensure that the source device's sound and image output did not exceed the capabilities of the audio system or display. If the specifications were not available or were not intelligible, the user would have to employ trial-and-error methods to discover the optimal settings.

For PCs, the absence of an EDID in the monitor would mean that the graphics card would have a default resolution. If the resolution had to be changed, users would have to set it manually for audio in a manner similar to that described above for home theater systems.

What Information Does an EDID Contain?

EDIDs provide a variety of information describing the capabilities of a display device or audio system. The data is arranged in 128-byte blocks. The VESA standard requires just a single block for VGA, DVI and DisplayPort. However, DisplayPort EDIDs will be enhanced to support an option for an extension block in order to define additional capabilities not covered in Block 0. The Consumer Electronics Association requires both the initial VESA block (Block 0) and one or more extension blocks; therefore, HDMI display devices have both the VESA block and the CEA extension block.

EDIDs are stored in an EPROM of an audio or video rendering device. Because of the limited storage space, EDID data is stored in a very compact manner using bit- or byte-oriented storage. In some cases the values are truncated or abbreviated to conserve space.

The list of display capabilities in the base EDID block is a long one. The block includes a header with 8 bytes of fixed data; vendor, product and version information; basic display parameters (video input definition, screen size and gamma) and color characteristics such as chromaticity and white point, along with timing information. The latter includes established and standard timings; a timing formula; and detailed timing descriptors. The VESA E-EDID standard requires that the first detailed timing descriptor be the "preferred" video format, with subsequent ones listed in order of decreasing preference. Consumer electronics equipment with HDMI interfaces requires both the VESA block and at least one extension block that defines the more important audio and video capabilities for HDTVs or audio systems.

EDID Operation

A source device reads the EDID of a sink in response to a connection event—called a hot plug—downstream at the display. The EDID is transmitted over the Display Data Channel (DDC) for consumer electronics products using VGA, DVI and HDMI, or over the auxiliary channel for monitors with DisplayPort interfaces (see Figure A). In the simple case of a source directly connected to a display device, the EDID is read when the hot-plug lead is asserted.

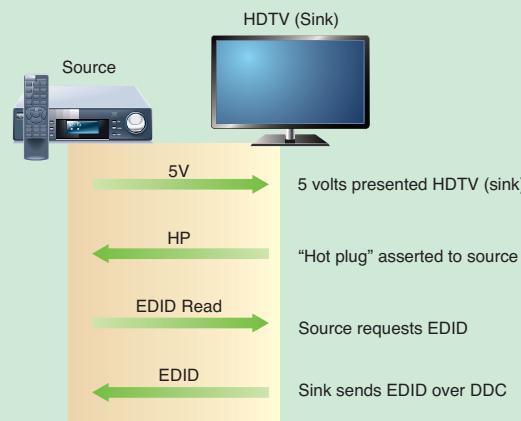


Figure A – Typical EDID operation involving a source device (set-top box) and sink (HDTV)

In cases where there is a repeater device between the source and the sink—common in home theaters—the EDID is read when the audio system transmits a hot-plug pulse in response to a connection event downstream at the sink. A repeater will forward the EDID directly to the source device or, in the case of an audio system, will substitute its audio EDID to the source, as shown in Figure B.

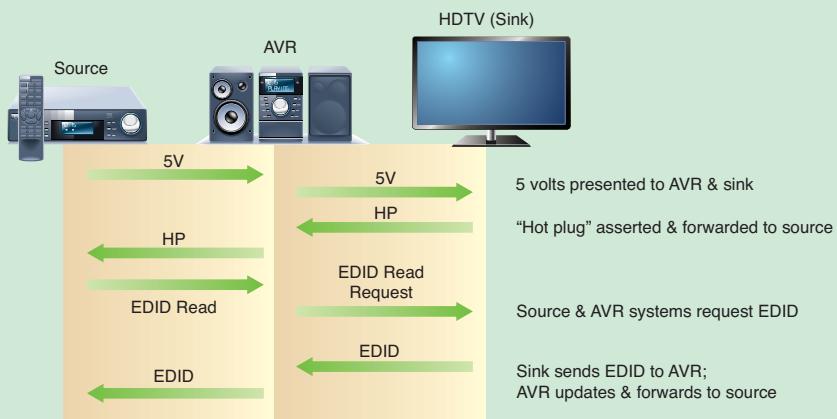


Figure B – Typical EDID operation with an audio system (AVR)

The ecosystem is not complete without considering validation of an EDID implementation. Given the importance of EDIDs in simplifying and optimizing the user experience, it is critical to get an EDID implementation correct. Various types of tests are necessary to ensure proper operation, such as the one shown in Figure C.

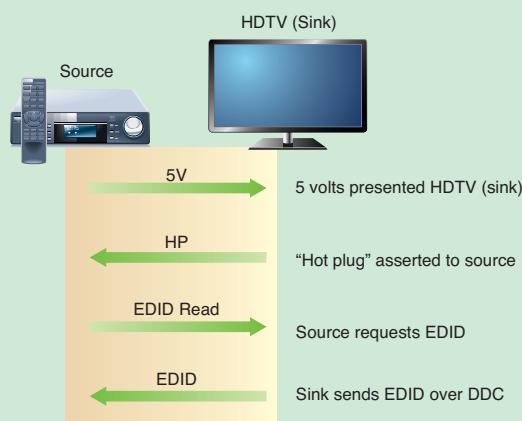


Figure C – Setup for testing a source device with test equipment emulating a display device

Many Levels of Testing

The most basic type of testing conducted in a development lab environment is functional testing, or simulating a properly operating device with test equipment. The test equipment interoperates with the sink device whose EDID is being verified. Conversely, when developing a source device, functional testing is used to verify that a source responds properly to a known-good EDID that the test equipment emulates. In most cases, a developer will want to test the source device against a variety of known-good EDIDs—old and new—to verify proper operation. Testing the EDID of a repeater device is more challenging and requires test equipment that can emulate both a known-good source and a known-good sink device.

Developers should also conduct fault testing on their EDID-equipped devices. Fault testing is a more rigorous type of testing in which the test equipment is configured to simulate a series of anomalous behaviors to verify that a device operates as desired in suboptimal conditions. Fault testing is important to ensure interoperability.

For testing source devices, test equipment emulates a rendering device and is provisioned with a flawed EDID. Test engineers can introduce a variety of anomalies to ensure that a source device responds appropriately. To emulate a flawed EDID or a series of them, it is essential to have an EDID editor utility in the test equipment. With the editor's help, developers can quickly construct a variety of modifications to existing EDIDs to verify that a source responds in the proper way.

EDID fault testing of a sink device or the input side of an audio system involves requesting EDID data in a way that is not typical. For example, although permissible, reading the EDID one byte at a time may result in an undesirable response from the display.

Developers of repeater devices, such as an audio or video processor, may want to mix functional testing and fault testing. For example, a developer could

use test equipment to emulate a known-bad EDID on the sink device and emulate a known-good source device.

Since improper EDIDs can cause significant interoperability problems, an HDMI device has to pass compliance testing in an authorized test center (ATC) in order to use the HDMI logo. VESA recently approved a similar compliance test for DisplayPort devices and it will soon be required for DisplayPort logo use.

Compliance test specifications define the series of tests necessary to ensure that a device operates correctly in accordance with a standard. Compliance testing of an EDID begins by defining the intended capabilities of the sink device. The capabilities are entered or imported into a compliance test application residing on a piece of test equipment. The pass/fail results for each distinct test in the series are shown in a report. Designers must take care to ensure that a failure is genuine and not the result of an improper configuration.

Since it's expensive and time-consuming to submit and resubmit a commercial product for testing, the best approach is for developers to procure test equipment for their lab to conduct precompliance testing. In many cases developers can obtain the same equipment used in the ATCs for their own lab. This significantly reduces the chance of failure in the ATCs. In the case of EDID compliance testing, the approved test tool for HDMI is commercially available from Quantum Data. A VESA-approved EDID compliance test tool for DisplayPort is anticipated to be commercially available soon from Quantum Data as well.

Although the goal of compliance testing is to ensure that devices interoperate, it is often not sufficient in and of itself because of the diversity of equipment and suppliers. Therefore, additional interoperability testing is often needed. One such area where interoperability testing has a high value is in ensuring backward compatibility.

To support backward compatibility with existing source devices, new EDIDs must include all fields and blocks that past versions of EDIDs have had. The CEA extension blocks have length fields so that older sources can skip newer, unsupported data blocks. Backward compatibility can be verified with the sink device under development but it is often more convenient to use test equipment because it enables developers to more quickly update EDIDs in the emulated sink for testing.

In developing a new source device, engineers may want to verify that it interoperates with the EDIDs of older sink devices. Having test equipment that can emulate a variety of older EDIDs is essential.

During all types of testing in the lab—functional, fault, compliance and especially interoperability—the EDID transactions can be monitored. This can help identify the root cause of some EDID-related interoperability problems, particularly those related to timing and responses to hot-plug events.

EDIDs consist of complex data sets that are critical in simplifying and optimizing the user experience in both PC and consumer electronics environments. Ensuring that EDIDs are implemented correctly is a key part of the development process. Quantum Data is a recognized authority on the verification of EDID implementations. Its test equipment and associated test applications are used in the authorized test centers and by developers all over the world.

— Carol Shields and Neal Kendall

of the sink Policy Maker function inside the LogiCORE. The RTL-based sink controller provides the remaining portion. The Policy Maker's function on the source side, being much more complex, is available as a source code reference design.

Let's take a closer look at the source-side Policy Maker. This allows the system



Figure 4 – VESA DisplayPort v1.1a Main Link

Device Controller in Figure 3 on the sink side is part of the sink design example that is available from CORE Generator™.

MicroBlaze Processor Plays Central Role

Xilinx designed the Source Policy Maker Controller to be used with the core so that it functions in much the same way as

control-plane processors. Designers can add this source code to the existing control software inside or outside of the FPGA. The license agreement provides the option of implementing the controller portion outside of the FPGA (that is, in an external processor) as long as the code is used in conjunction with the core.

Designers can modify the XAPP design using the Xilinx embedded hardware design kit with Xilinx Platform Studio (the EDK) or Xilinx embedded software design kit with the SDK. In general, FPGA designers typically use the EDK and software developers rely on the SDK.

The EDK flow produces an intermediate net file (NGC) that you can incorporate into the top-level ISE® project prior to implementing the design. The NGC file contains the MicroBlaze code as part of the BRAM initialization.

Fast Turnaround

The EDK flow generally takes more time when you have modified the software; however, once you've generated a netlist, you no longer need the EDK or SDK. The SDK flow modifies the FPGA bitstream, updating only the contents of the MicroBlaze code in BRAM. This SDK flow provides a faster turnaround time for software modification; however, in this scenario, you must use the SDK every time you generate a bitstream. The XAPP white paper on this topic contains detailed instructions on how to use the Xilinx FPGA Embedded Software Development Kit to run this design.

The “logic” portion of the Source Policy Maker Controller design sits on top of a MicroBlaze processor and uses I2C commands to control the link, stream and configuration space. The C code implements the Policy Maker instruction controls, the top-level instantiation files and the Embedded Development Kit (EDK). Meanwhile, Xilinx provides Software Development Kit (SDK) project files to give designers maximum implementation flexibility. Xilinx also provides Policy Maker C source code for applications with existing

designer maximum flexibility in functionality and implementation. The top level of the example design contains two high-level component instantiations of the core: the XAPP Implementing a DisplayPort Source Policy Maker Controller System Reference Design using a MicroBlaze Embedded System and the DisplayPort core source (Tx) design. Xilinx divided the core's implementation into atomic link functions called the Main Link, the Secondary Channel and the AUX Channel protocol. The Main Link provides for the delivery of the primary video stream. The Secondary Channel, which Xilinx will include in a future release of the core, will integrate the delivery of audio information into the Main Link during a blanking period. Meanwhile, the AUX Channel establishes the dedicated source to the sink communication channel (see Figure 2).

Xilinx added a line buffer to the user data interface so users can easily implement the example design in an FPGA (see Figures 2, 3 and 4). The Policy Maker or

an ASSP DisplayPort source device. We recommend that you use a MicroBlaze embedded or external processor to properly initialize and maintain the link. The XAPP contains a preconfigured version of the Policy Maker Reference Design implemented in a MicroBlaze processor inside the FPGA, allowing you to immediately turn this design into hardware. When it becomes available, the reference design will contain source code designers can modify.

The “logic” portion of the Source Policy Maker Controller design sits on top of a MicroBlaze processor and uses I2C commands to control the link, stream and configuration space. The C code implements the Policy Maker instruction controls, the top-level instantiation files and the Embedded Development Kit (EDK). Meanwhile, Xilinx provides Software Development Kit (SDK) project files to give designers maximum implementation flexibility. Xilinx also provides Policy Maker C source code for applications with existing

In DisplayPort, VESA has added intelligence to the system to negotiate capabilities between the source (for example, a set-top box, DVD player or PC graphics card) and sink device (such as a display monitor) and to optimize communication parameters.

The Policy Maker on the source side contains a state machine that connects to the processor interface through an AMBA® APB port or a 32-bit PLBv46 bus using an AMBA-to-PLB bridge. Xilinx has stored an instruction set in Block RAM that you can modify. The C++ code Xilinx used to train the link was compiled using the GNU C++ compiler and has been fully tested on a soft MicroBlaze processor implemented inside the FPGA using Xilinx's EDK Platform Studio processor design suite. The reference design contains a complete Xilinx SDK project. The sample test bench connects a 135-MHz clock to a VID clock, and a 100-MHz clock to the APB clock. Xilinx has checked that it has appropriately connected all the inputs. Reset is available at the top-level block.

Extended Display Identification

An extremely important part of DisplayPort is interfacing different devices via VESA's Enhanced Display Identification Data structures. EDIDs are really not new. In fact, for a number of years designers have been using a variety of video interfaces to read sink device parameters from EDIDs to interface devices. However, these early EDIDs and related interface technologies generally didn't include a sophisticated configurable communications channel. Now, however, with DisplayPort, VESA has added intelligence to the system to negotiate capabilities between the source (for example, a set-top box, DVD player or PC graphics card) and sink device (such as a display monitor) and to optimize communication parameters (for a tutorial on VESA EDIDS, see the sidebar). The variables negotiated for DisplayPort v1.1a include

the number of lanes (one, two or four), the data rate per lane (1.62 or 2.7 Gbps), the voltage swing (0.2, 0.6, 0.8, 1.2 V), four levels of channel pre-emphasis and a down-spreading of the link clock.

The Rx sink example design provided with the LogiCORE from the CORE Generator provides an example EDID (see Figure 3) that is intended to be read by a source device in order to ensure that the user's viewing experience is optimal.

The sink example design implements the EDID data structure in BRAM inside the FPGA. The DisplayPort source code enables an I2C protocol over the AUX Channel. Figures 3 and 4 show a block diagram of a DisplayPort sink connected to a source. The Link and Stream Policy Maker on the sink side is part of the sink core; however, the Link Policy Maker on the source side is more complex and will be provided as a source code with the reference design. The EDID interfaces to the Rx sink side through the I2C interface.

The I2C protocol is ideal for bolting onto the EDID data structure and is commonly used for this type of application. The I2C controller locates and manages the data found in the EDID and passes it to the sink core through the I2C interface protocol (over the AUX Channel) through a serial interface. In operation mode, you do not need to be aware that the EDID is being accessed. By probing the I2C bus, you can monitor the content of the ROM. In debug mode, you can modify the I2C controller and override the 3-bit content found in the EDID ROM. The I2C supplies the control signals, which when tied to the proper open collector output provide the I2C master interface.

The sink includes a data structure called the DisplayPort Configuration Data (DPCD), which stores configuration data and acts as a communications mailbox that both the sink and source can read and write to. The source generally consumes the contents of the DPCD across the AUX Channel (see Figures 3 and 4).

Policy Maker Link Training

The process of establishing communications on the DisplayPort link is called "link training." During link training, the core at the start of communication will try to optimize the link speed and power used while minimizing errors. If there are problems during data transfer, the core will automatically repeat link training to adapt to the changing conditions. Communications between the source and sink packets take place over the bidirectional half-duplex 1-Mbps AUX Channel. Video and audio data is transferred on the main link lanes (1, 2 or 4), which are high-speed gigabit transceiver channels going from the source to the sink.

The core performs link training in two phases: clock recovery followed by channel equalization, symbol lock and interlane alignment. During the first phase, the receiver's PLLs are locked to the incoming signal and a link clock is recovered. During the second phase, the system optimizes channel equalization and establishes symbol lock and interlane alignment.

Here is a typical operation sequence of the Policy Makers on both the source and sink sides:

1. The Tx Link Policy Maker monitors hot-plug detect and, when detected, gives notice to the Stream Source Policy Maker. The Stream Source

Policy Maker reads the sink's EDID over the AUX Channel.

2. The Tx Link Policy Maker reads the DisplayPort Configuration Data from the sink via the AUX Channel.

Depending on source and sink capabilities, it writes the configuration parameters to the link configuration field of the sink DPCD and starts link training by writing to the TRAINING_PATTERN_SET byte of the sink DPCD. It then initiates sending the training patterns.

3. The Tx Link Policy Maker controls the clock recovery sequence by adjusting the voltage swing and bit rate if necessary with feedback from the Rx Link Policy Maker. Once the core achieves clock recovery, link training moves on to the channel equalization stage, where pre-emphasis is adjusted if called for by the Rx Link Policy Maker. The

receiver also establishes symbol lock and interlane alignment.

4. Once the core passes link training (that is, the system achieves bit lock and symbol lock), it is indicated with in the DPCD. The Tx Link Policy Maker reports the training status to the Tx Stream Policy Maker, which enables the isochronous stream along with stream attribute data transfer.

Policy Maker Additional Functionality

Besides involvement in link training, the Tx Link Policy Maker also uses the IRQ HPD signals from the receiver to monitor for sink event notification and checks the link-status fields of the DPCD to understand the reason for the interrupt. If it detects that the link has lost lock, the Tx Link Policy Maker must retrain the link. It can also reconfigure the link for increased or reduced main-link lane count if the receiver calls for it.

The Link Policy Maker also determines the order of multiple AUX request transac-

tions, since one transaction ends before another can be initiated. Since a sink may reply with a NACK or DEFER, the Policy Maker must decide on the follow-up action for those cases. AUX transactions are limited to 16 bytes of data, so the Policy Maker must divide larger transactions into multiple transactions with none larger than 16 bytes.

Thanks to this capability to negotiate and optimize link settings, DisplayPort can achieve optimal results over varying conditions. The Link and Stream Policy Makers are the control functions that coordinate the process, allowing modern high-speed video and audio transport. Xilinx's Source Policy Maker Controller System reference design using a MicroBlaze Embedded System is designed to help you exploit these new capabilities to the fullest to bring to market your feature-rich display product. The Xilinx DisplayPort LogiCORE offers a flexible source and sink solution with example EDIDs and source code ready to download into the TED Spartan-6 Consumer Video Kit. Evaluation versions of this IP are available at no charge. You can find everything you'll need to get started as well as a link to the XAPP Implementing a DisplayPort Source Policy Maker System Reference Design using MicroBlaze Embedded System at <http://www.xilinx.com/products/ipcenter/EF-DI-DISPLAYPORT.htm>, coming in late May 2010.

References

1. Xilinx IP Center – DisplayPort LogiCORE: <http://www.xilinx.com/products/ipcenter/EF-DI-DISPLAYPORT.htm>
2. Quantum Data 882E Video Test Instruments, http://www.quantumdata.com/pdf/882E_DP_DS_RevI.pdf
3. VESA DisplayPort Standard, v1.1a, January 2008
4. I2S Bus Specification, Philips Semiconductors, June 1996 (more on the I2S bus is at http://www.nxp.com/acrobat_download/varioust/12SBUS.pdf)
5. UG196 and UG 198, Virtex®-5 FPGA GTP Transceiver User Guides
6. UG386, Spartan-6 FPGA GTP Transceiver User Guide

The authors would like to thank Carl Rohrer, Matt Ouellette, Tom Strader and Chris Arndt of Xilinx and Craig Bezek of Quantum Data for their reviews and input.

~Imagine.Combine.Implement.Today~

IP Cores for Your Low-Volume FPGA Products

you have an **idea?** looking for the perfect graphics controller? choose & configure IPs implement your FPGA w/o coding logicBRICKS™ make your ideas come true

Get Online, Plug and Play!

Prices start at €500

Visit our web shop today:
www.logicbricks.com

logicBRICKS™
Designed by **XYLON**

logicBRICKS is a registered trademark of Xylon d.o.o.
All other trademarks are the property of their respective owners

Drive the Automotive Market with a Distinct Advantage: Xilinx Spartan-6 FPGAs

By NICK DIFIORE

JUST 10 YEARS AGO, you could count on one hand the number of electronic systems in an automobile. Today, it is difficult to find a single system in a car that isn't electronic or at least electromechanical, as car manufacturers look beyond engine block size and body design to electronics to differentiate their offerings. Xilinx® Automotive (XA) Spartan® FPGAs are playing an increasingly important role in this auto electronics revolution. And this is just the beginning.

In the past, auto electronics suppliers mainly used FPGAs to prototype their designs. To keep costs down, they would almost always move the design to an ASIC or leverage an ASSP for final production, despite the long arduous development process. This is rapidly changing.

In 2004, Xilinx reached an important milestone with the release of its 90nm XA Spartan-3 FPGAs and its commitment to full automotive qualification. That FPGA family achieved a cost level that allowed customers not happy with the spiraling costs of ASICs or limited differentiation of ASSPs to use FPGAs in production automobiles, not just for prototyping.

In fact, today's most advanced and successful luxury automobiles have as many as 18 Xilinx devices across driver assistance, driver information and

infotainment systems. Now, with the new XA Spartan-6 FPGAs, automotive companies can bring more advanced electronics to all of their vehicle lines, not just their luxury models.

The XA Spartan-6 FPGA Advantage

Xilinx's new 45nm XA Spartan-6 FPGAs have more than double the effective logic of XA Spartan-3 devices, while also increasing clock speeds, on-chip memory and parallel DSP processing power. New hard block functions like multiport memory controllers and PCIe®, combine with flexible Gigabit rated SelectIO™ technology and available Multi-Gigabit Transceivers to give designers what they need to create the industry's most advanced automotive applications at affordable price points.

For example, right out of the box, the new XA Spartan-6 FPGAs support faster DDR3 memories for camera, video, or high resolution display-based applications. Xilinx has also

made PCIe host interfacing available for infotainment systems requiring high throughput for multimedia storage, playback, and transport. The dedicated PCIe hard block and high speed I/Os eliminate the need for an external interface chip and keep more programmable logic free for application-specific features.

XA Spartan-6 FPGAs do all of this while exceeding AEC-Q100 qualification standards to meet the quality needs of the most demanding automakers worldwide.

In the coming months, the XA Spartan-6 value proposition will become even more compelling as Xilinx rolls out its XA Targeted Design Platforms, allowing designers, in turn, to further accelerate the pace of automotive innovation.

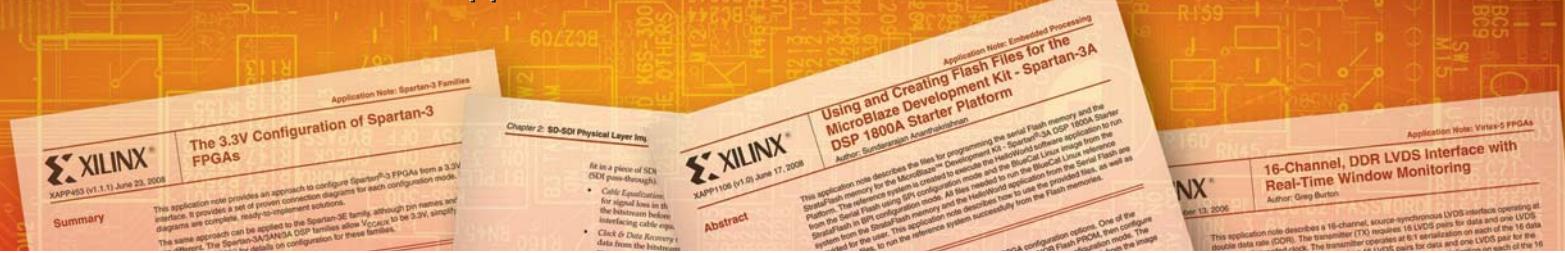
To learn more about the XA advantage, visit www.xilinx.com/automotive.



About the Author: Nick DiFiore is the Director of Automotive System Architecture & Platforms based in Xilinx, Inc.'s Detroit offices. Contact him at more_info@xilinx.com

Application Notes

If you want to do a bit more reading about how our FPGAs lend themselves to a broad number of applications, we recommend these notes.



XAPP1065: Spread-Spectrum Clock Generation in Spartan-6 FPGAs

http://www.xilinx.com/support/documentation/application_notes/xapp1065.pdf

Consumer display applications commonly use high-speed LVDS interfaces to transfer video data. Designers can use spread-spectrum clocking to address electromagnetic-compatibility issues within these consumer devices. This application note by Jim Tatsukawa uses Spartan®-6 FPGAs to generate spread-spectrum clocks by means of the DCM_CLKGEN primitive.

Spartan-6 FPGAs can generate a spread-spectrum clock source from a standard fixed-frequency oscillator. The DCM_CLKGEN primitive can use either a fixed spread-spectrum solution without any logic, providing the simplest implementation, or a soft spread-spectrum solution using a state machine. The soft implementation adds flexibility but requires additional control logic to generate the spread-spectrum clock.

While the application note focuses specifically on LVDS displays, it's applicable to other applications with similar DCM usage that also have need of spread-spectrum clocks.

XAPP1144: Virtex-6 Embedded Trimode Ethernet MAC Hardware Demonstration Platform

http://www.xilinx.com/support/documentation/application_notes/xapp1144.pdf

This application note describes a system using the Virtex®-6 FPGA Embedded Trimode Ethernet MAC Wrapper core on a Xilinx Virtex-6 FPGA ML605 development board. The embedded system is under the control of a PC-based graphical user interface that provides access to the Ethernet MAC's features along with several data flow options.

The hardware demonstration platform shows how to integrate the Ethernet MAC, its wrapper core and the Ethernet Statistics v3.3 core into a system, generate the required clock resources, handle the Ethernet data flow using packet FIFOs and flow control, and connect to a physical interface. An embedded microprocessor manages the embedded system and the Ethernet MAC. Data flow occurs in the fabric logic; the microprocessor does not handle it in real time.

XAPP878: MMCM Dynamic Reconfiguration

http://www.xilinx.com/support/documentation/application_notes/xapp878.pdf

This application note by Karl Kurbjun and Carl Ribbing provides a method to dynamically change the clock output frequency, phase shift and duty cycle of the Virtex-6 FPGA mixed-mode clock manager (MMCM) through its dynamic reconfiguration port (DRP). The authors explain the behavior of the internal DRP control registers and offer a reference design that uses a state machine to drive the DRP, ensuring that the registers are controlled in the correct sequence. Due to its modular nature, the design can be used as a full solution for DRP or can be easily extended to support additional reconfiguration states. The design also uses minimal Virtex-6 FPGA resources, consuming only 24 slices.

The reference design supports two reconfiguration state addresses and can be extended to support additional states. Each state does a full reconfiguration of the MMCM, making it possible to change most parameters. The design does not support outputs configured with fractional divider values. Nor does it support reconfiguring with fine phase shifting enabled.

XAPP876: Virtex-5 FPGA Interface to a JESD204A-Compliant ADC

http://www.xilinx.com/support/documentation/application_notes/xapp876.pdf

In this application note and accompanying reference design, author Marc Defossez describes how to interface the Virtex-5 LXT, SXT, TXT and FXT devices featuring GTP/GTX transceivers to an analog-to-digital (ADC) converter compliant to JEDEC Standard No. 204A (JESD204A) Serial Interface for Data Converters. With some restrictions, this design can also be used for ADC devices that comply with the older JESD204 standard.

The JESD204A standard describes a serialized interface between data converters and logic devices. It contains normative information to enable the implementation of designs that communicate with JESD204A-compliant devices.

This application note shows how to implement a two-lane dual ADC with each lane having a 14-bit resolution and running at 125 Msamples/second. It provides an overview of how to implement the serial data interface and the link protocol described in the JESD204A standard.

The JESD204A standard describes the protocol for implementation with general high-speed serdes devices. The Virtex-5 TXT device contains GTX transceivers. The JESD204A standard is interpreted accordingly, and a compliant interface is delivered for GTX transceivers. The implementation described is for a single device containing two converters, using one link of two lanes connected to the FPGA.

XAPP873: Virtex-5 FPGA Interface for Fujitsu Digital-to-Analog Converters with LVDS Inputs

http://www.xilinx.com/support/documentation/application_notes/xapp873.pdf

In a second application note, Marc Defossez describes how to interface a Fujitsu MB86064 or MB86065 digital-to-analog converter (DAC) with parallel low-voltage differential signaling (LVDS) inputs to a Virtex-5 FPGA utilizing the dedicated I/O functions of the FPGA family. The note and accompanying reference design also illustrate a basic LVDS interface for connecting to any DAC with high-speed parallel interfaces.

The author demonstrates the implementation in hardware using the DK86065-2 Fujitsu development kit and the Xilinx ML550 and ML555 demonstration boards. Fujitsu has developed a passive interface adapter module for this purpose.

All three of the implementations described make use of the OSERDES I/O features of the Virtex-5 FPGA. Designers can extend the reference applications in resolution width and speed for use in a wide range of applications. At the same time, you will find techniques you can use to interface the Virtex-5 FPGA to other types of DAC components.

XAPP1064: Source-Synchronous Serialization and Deserialization (up to 1050 Mb/s)

http://www.xilinx.com/support/documentation/application_notes/xapp1064.pdf

Spartan-6 FPGAs perform in a wide variety of applications requiring various serdes factors up to 16:1, at speeds up to 1,050 Mbits/second, depending on the application, speed grade and package.

The input and output serdes blocks—ISERDES and OSERDES, respectively—in Spartan-6 devices simplify the design of serializing and deserializing circuits, while allowing higher operational speeds. This application note by Nick Sawyer discusses how to efficiently use the ISERDES and OSERDES primitives in conjunction with the input delay blocks and phase detector circuitry.

Each Spartan-6 FPGA input/output block contains a 4-bit ISERDES and a 4-bit OSERDES. You can cascade the serdes from two adjacent blocks (master and slave) to make an 8-bit block. This gives the possibility of serdes ratios from 2:1 to 8:1 on both output and input for both single- and double-data-rate I/O clocks.

XAPP880: SFI-4.1 16-Channel SDR Interface

with Bus Alignment Using Virtex-6 FPGAs

http://www.xilinx.com/support/documentation/application_notes/xapp880.pdf

The 16-channel single-data-rate interface design described in this application note targets a Virtex-6 FPGA, taking advantage of ChipSync™ technology, available for every I/O in all Virtex-6 devices. The design features the ability to dynamically adjust the delay of the clock path in the receiver with 78-picosecond resolution. Using this dynamic delay feature, the receiver escapes the limitations of static setup/hold timing by creating its own dynamic setup/hold timing. The interface calibrates out process variations by finding the optimal setup/hold timing for each individual device.

Author Vasu Devunuri shows a Virtex-6 FPGA SFI-4.1 interface talking to an SFI-4.1 interface in another device—either an ASIC or an FPGA—that supports a 16-channel SDR interface. Because the interface is a source-synchronous link, the receivers of both devices get their clock from the Tx side of the other device. The transmitter clocks originate in the back-end systems and can be provided from a variety of sources, such as an oscillator on the printed-circuit board. Because each of the 16 data channels on the serial side of the interface runs at 4:1 serialization, the data width on the parallel side of the SDR interface is 64 bits.

Based on the design and characterization described, the SFI-4.1 16-channel SDR reference design exceeds its performance targets under all conditions of process, voltage and temperature. However, designers should be aware that any deviation from the timing budget can degrade the maximum performance.

XAPP1088: Correcting Single-Event Upsets in Virtex-4 FPGA Configuration Memory

http://www.xilinx.com/support/documentation/application_notes/xapp1088.pdf

Designers of high-reliability applications must be concerned with the effect of single-event effects (SEEs) on FPGA configuration memory. Changes to configuration memory can cause changes in the functionality and performance of the device. This application note by Carl Carmichael and Chen Wei Tseng describes the use of configuration scrubbing and readback in the Virtex-4 family of FPGAs for detecting and correcting single-event effects (SEUs) induced by cosmic rays.

In-orbit, high-energy charged particles (radiation) can have an effect on the electronic components used in space-based and extraterrestrial applications. In particular, SEEs can alter the logic state of any static memory element (latch, flip-flop, routing pip or RAM cell). Because the user-programmed functionality of an FPGA depends on the data stored in millions of configuration latches within the device, SEUs in the configuration memory array might have adverse effects on the expected functionality.

Mitigation techniques, such as using the TMRTTool to implement triple-module design redundancy, or triple FPGA deployment, can harden the application against SEUs. However, designers must correct the upsets so that errors do not accumulate.

Toward this end, the Virtex-4 FPGA SelectMAP interface provides the most efficient, post-configuration read/write access to the configuration memory array. This application note focuses on configuration mitigation operation performed via SelectMAP.

Xilinx's Latest Platforms

by Mark Goosman

Senior Marketing Manager, Xilinx Platform Solutions

Early last year, Xilinx introduced Targeted Design Platforms, a tremendous step forward in helping designers address what has been termed the “programmable imperative”—the necessity to do more with less, remove risk wherever possible, and differentiate in order to survive. The aim was to provide customers with simpler, smarter and more strategically viable design platforms for the creation of world-class FPGA-based solutions in a wide variety of industries (see cover story, *Xcell Journal*, Issue 68). In support of these platforms, Xilinx has launched a family of evaluation and development kits, each of which combines all the individual components required to enable quick development right out of the box.

Development kits from Xilinx® combine “targeted” and “tested” resources in the form of hardware, development tools, IP and reference designs, enabling designers to immediately begin developing, integrating and debugging system logic, interfaces and software. FPGA Mezzanine Card (FMC) daughterboards from Xilinx and third parties enable expansion with domain-specific, market-specific and customer-specific capabilities.

Whether for the general development needs that the Base Platform Kits address, or the specific needs of DSP, embedded or high-speed connectivity development with the Domain-Specific Kits, Xilinx has worked to deliver complete solutions that will help designers get off the ground faster and, thus, get their products to market in much less time.

BASE PLATFORM KITS

The base-level Virtex®-6 and Spartan®-6 FPGA evaluation kits provide all the elements that engineers need to develop basic FPGA-based systems, including support for I/O, memory interfacing and more. These evaluation kits provide a flexible environment for higher-level system design, including applications that need to implement features such as Ethernet or DDR2. Equipped with industry-standard FMC connectors, these kits provide the capabilities for scaling and customization for specific applications and markets.

The base platform comprises a robust set of well-integrated, tested and targeted elements that enable customers to immediately start a design. These elements include:

- Xilinx FPGA silicon
- ISE® Design Suite Logic Edition
- Reference designs common to many applications, such as memory interface and configuration designs
- Development boards that run the reference designs
- A host of widely used IP, such as GigE, Ethernet and memory controllers

EMBEDDED, DSP AND CONNECTIVITY DOMAIN KITS

The more you know in advance about your demanding design requirements, the more Xilinx can provide to help accelerate your FPGA system design. Domain-specific platform kits provide additional integrated tools and IP optimized for the three primary Xilinx FPGA user profiles, or domains: the embedded processing developer, the digital signal-processing (DSP) developer and the logic/connectivity developer.

These platforms save time by providing an enhanced design infrastructure, enabling engineers to jump ahead in the development schedule to focus more time on creating their own unique product differentiation.

Domain-specific kits extend the base platform with a predictable, reliable and intelligently targeted set of integrated technologies, including:

- Xilinx FPGA silicon
- ISE Design Suite System Edition, supporting higher-level design methodologies
- Targeted Reference Designs optimized for embedded processing, connectivity and DSP

- User interface enabling fast bring-up and detailed evaluation of key capabilities
- Base development boards with additional functionality provided via FMC daughtercards
- Domain-specific IP including MicroBlaze™, PCI Express® (gen1 and gen2), XAUI, DSP filters and other functions required for embedded, DSP and connectivity-based design
- Operating systems (required for embedded processing) and software

Every element in these platforms is tested, targeted and supported by Xilinx and our ecosystem partners. Starting a design with the appropriate domain-specific platform can cut weeks, if not months, off your development time.

Digital Signal Processing

For DSP designers, the Xilinx DSP kits offer an easy way to explore the ultrahigh (GMACs/s) performance in Virtex-6 FPGAs or the superior high-performance, low-cost advantages enabled by Spartan-6 FPGAs. With support for design flows that leverage MATLAB® and Simulink from

The MathWorks, or RTL examples, seasoned DSP algorithm developers can get started quickly. The kits include versions of the reference designs that support either flow, allowing you to begin your design in the manner you find most familiar.

Embedded Design

The Xilinx embedded kits enable rapid software application development as well as easy customization of the processor hardware subsystems. Software developers can begin building applications right out of the box using the Xilinx SDK (an Eclipse-based IDE) and lightweight real-time operating systems like Micrium uC/OS-II. Or you can develop applications using a full-fledged embedded Linux board support package from PetaLogix. The Targeted Reference Design for the Xilinx embedded kits includes a MicroBlaze soft-core processor subsystem design.

Connectivity

Designing systems that incorporate high-speed serial I/O, especially those that work with multiple protocols and different line rates, can be challenging. The Xilinx connectivity kits address these challenges, leveraging proven, high-performance transceiver technology available in the Virtex-6 family along with the industry's first transceivers available in an FPGA opti-

mized for low power and cost, in Spartan-6 FPGAs. The ISE Design Suite software integrates state-of-the-art serial connectivity tools to allow implementation of PCIe® (gen1 and gen2), XAUI and Gigabit Ethernet bridges that have been optimized for maximum bandwidth using DMA. The kits support designs that utilize both established as well as new serial protocols. This complete and comprehensive approach accelerates development for experienced users and simplifies the adoption of FPGAs for new users.

TARGETED REFERENCE DESIGNS

Xilinx has dramatically accelerated designer productivity by including Targeted Reference Designs with each of these kits. Tuned to the specific domain, each of the Targeted Reference Designs can be used as is or modified and extended. These preverified designs offer significant benefits to a wide range of designers. For new users, the Targeted Reference Designs reduce the learning curve by demonstrating FPGA implementations of real-world concepts. Experienced designers can use them to jump-start their application development or to quickly evaluate platform architecture and capabilities.

Even without purchasing the specific base platform or domain-specific kit, users can download documentation and reference designs. This allows both greater evaluation of the capabilities of the Xilinx kits and the abil-

ity to upgrade the features and functionality from one kit to another. In most cases, you can find information on upgrading kits on the documentation page for the specific evaluation or development kit in the Xilinx Boards and Kits area at www.xilinx.com/kits. For example, if you have purchased the Spartan-6 FPGA SP605 Evaluation Kit and wish to upgrade to the Spartan-6 FPGA Embedded Kit, there's a three-step process detailed at the bottom of the page at www.xilinx.com/products/boards/s6embd/reference_designs.htm.

Caught between the growing complexities of technical advances and shrinking design schedules, it's becoming more and more vital for designers to find new ways to streamline development. The primary driver behind the launch of Targeted Design Platforms was the need to accelerate customers' design cycles and enable them to spend less time developing the infrastructure of an application and more time creating their unique value in the design. With the broad portfolio of development kits, Xilinx is helping customers leap ahead in their development to focus on the portion of their application that creates a unique difference from their competition. 

Visit www.xilinx.com/kits to see more details on all these products as well as to learn about market-specific offerings such as the new Spartan-6 FPGA Industrial Video Processing Kit. From time to time, Xilinx offers special promotional pricing on select boards and kits.

Kits for Virtex-6 FPGAs	Virtex-6 FPGA ML605 Evaluation Kit	A full-featured, highly scalable development environment for high-performance applications
	Virtex-6 FPGA Connectivity Kit	Enables designers to build high-speed serial and other connectivity applications
	Virtex-6 FPGA ML623 MGT Characterization Kit	For comprehensive characterization of Virtex-6 FPGA high-speed serial transceivers
	Virtex-6 FPGA Embedded Kit	Gives hardware and software developers a basic platform for building high-performance embedded applications
	Virtex-6 FPGA DSP Kit	Accelerates DSP development with reference designs for building digital signal-processing-based applications
Kits for Spartan-6 FPGAs	Spartan-6 FPGA SP601 Evaluation Kit	Entry-level environment for evaluating the Spartan-6 family
	Spartan-6 FPGA SP605 Evaluation Kit	Full-featured kit for developing low-cost applications requiring high-speed serial connectivity
	Spartan-6 FPGA Embedded Kit	Gives hardware and software developers a basic platform for building low-cost embedded applications
	Spartan-6 FPGA DSP Kit	Kick-starts development of DSP applications with reference designs for digital signal-processing-based applications
	Spartan-6 FPGA Connectivity Kit	Enables designers to build high-speed serial and other connectivity applications
	Spartan-6 FPGA SP623 MGT Characterization Kit	For comprehensive characterization of high-speed serial transceivers

Visit www.xilinx.com/kits for technical details and ordering information.

Tools of Xcellence

News and the latest products from Xilinx partners

by Mike Santarini
Publisher, *Xcell Journal*
Xilinx, Inc.
mike.santarini@xilinx.com

Altium Facilitates FPGA-Based System Development with NanoBoard 3000

EDA vendor Altium Ltd.'s advanced FPGA-based development board, the Altium NanoBoard 3000, targets design groups using FPGAs with on-board processors at the heart of their system designs. Based on a Xilinx® Spartan®-3AN FPGA, the board is a feature-packed programmable design environment that Altium provides to customers complete with hardware, software, ready-to-use royalty-free IP and a dedicated Altium Designer Soft Design license—all for \$395.

"We call it an instant FPGA prototyping environment," said Bob Potock, director of technical marketing at Altium (Sydney, Australia). "It targets the mainstream company that doesn't have the luxury of having a team of FPGA development specialists but has systems engineers focused on developing new products. A growing number of design teams want to make FPGAs central in their system designs and are adding processor cores to those FPGAs."

With the NanoBoard 3000, Potock said, electronics designers can construct soft-processor-based systems inside the FPGA without any prior FPGA design expertise. Nor do NanoBoard 3000 developers need any VHDL or Verilog skills, he said, because they can use their existing board-layout and systems-design know-how to construct, test and implement FPGA-based embedded systems.

"Users can add processors, memory controllers, peripheral blocks and software stacks, and create a system without having to write HDL or even low-level driver code," said Potock.

Potock said the key to the NanoBoard 3000 is tight integration among the development board, the Altium Designer tools and the IP.

In addition to a Spartan-3AN, the prototyping board contains an LCD panel with touchscreen capabilities, along with flash RAM, static RAM and dynamic RAM memory. It supports USB, SVGA, Ethernet, PS2 and MIDI, and also has programmable clocks and A/D and D/A converters on board.

Meanwhile, Altium has added a number of graphical editors and scripts to Altium Designer, streamlining many design tasks between FPGA programming and PCB design. "Altium Designer is really a development platform that incorporates printed-circuit-board design, FPGA design and embedded-software development, all integrated together," said Potock.

The system works hierarchically. "If I were building a new product, the PCB would be the top-level project," he said. "Below that would be the FPGA project and inside that project would be the embedded hardware, such as the MicroBlaze™ processor, and then the software project running on that processor. These are all linked in Altium Designer."

Specifically for the FPGA design step of the flow, Altium Designer includes a schematic-entry, HDL entry or even C-to-HDL compiler

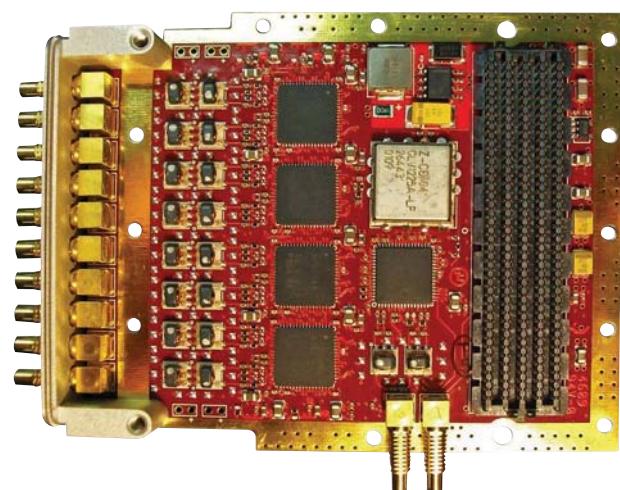
to program the FPGA. "We also have a technology called 'open bus,' which creates a layer of abstraction so users can quickly integrate IP blocks that we provide," said Potock. "They can be quickly downloaded into the FPGA."

Altium provides customers with a fairly extensive library of royalty-free IP. "You can construct your FPGA design with these blocks and also use the vertical market cores provided by Xilinx to develop very specific systems," said Potock.

Altium has also built functionality into Altium Designer that allows the tool to control Xilinx's ISE® Design Suite. "After a user creates an FPGA design in schematics, HDL or C, Altium Designer wraps around ISE Design Suite so that when you say 'program or compile the design,' it can call Xilinx's synthesis tool or Altium's own synthesis tool," said Potock. "From a customer perspective, they don't see the Xilinx tools, they see Altium Designer. I think folks appreciate that they only have to work in one environment for all these steps."

The company also provides drivers and software development scripts for each of the IP blocks to facilitate embedded-software development. "We have something called Software Platform Builder that detects all the cores you have put into your FPGA, and it will literally build up a software stack based on your IP choices," said Potock.

For more information on the Altium NanoBoard 3000, visit <http://altium.com/products/altium-designer/en/altium-designer-home.cfm>.



The Altium NanoBoard 3000 and associated tools aim to simplify FPGA design for the masses.

4DSP Fields Eight A/D and D/A FMC Boards, New FM680 PMC-XMC Mezzanine Card

Defense electronics company 4DSP LLC (Reno, Nev.) has released eight new analog-to-digital and digital-to-analog boards based on the FPGA Mezzanine Card (FMC) standard, along with a new version of its flagship FM680 PMC-XMC mezzanine card equipped with a Virtex®-6 FPGA.

The latest 4DSP FMC boards are based on the new open industry standard developed by a consortium of companies working through the ANSI/VITA organizations, as defined in the ANSI/VITA 57.1 2008 specification. FMC modules are designed to connect to FMC-compliant carrier cards in the CompactPCI, VPX or PCI Express® form factors. Xilinx has given FMC a large role in its Targeted Design Platforms and the Virtex-6 and Spartan-6 FPGA associated development kits (see cover story, *Xcell Journal*, Issue 68).

Pierrick Vulliez, CTO at 4DSP, said the company has offered very high-speed analog cards for many years. Now, with both companies' cards supporting FMC, the 4DSP boards can run optimally with Xilinx's new kits. "The cards range in performance from 60 Megasamples per second to 5 Gsps," said Vulliez. "FMC allows us to offer our products to a much larger audience."

The 4DSP FMC Series cards include a number of unique features, such as a user-selectable option to have data sampled by an internal clock source (optionally locked to an external reference) or to use an externally supplied sample clock. A trigger input for customized sampling control is also available. I/O connections are on the front panel as per VITA 57.1. Cascading multiple FMC boards for synchronized high channel count is possible.

Equipped with power supply and temperature monitoring, the 4DSP Series FMC cards have several power-down modes to switch off unused functions and reduce system-level power and heat. Vulliez said these features are well suited for software-defined radio (SDR) and similar applications requiring batteries or other low-power sources. The FMC cards are also well suited for man-pack applications, ground mobile vehicles, UAVs and other airborne applications where limited power sources affect mission range and on-station mission time. The boards are also available with Mil-I-46058c-compliant conformal coating for use in hostile environments.

The products include:

- FMC103, a four-channel, 210-MspS FMC-LPC board with 12-bit A/D converter
- FMC104, a four-channel, 250-MspS FMC-LPC board with 14-bit ADC
- FMC107, an eight-channel, 65-MspS FMC-LPC board with 12-bit ADC
- FMC108, an eight-channel, 250-MspS FMC-HPC board with 14-bit ADC
- FMC110, which provides two channels of 12-bit A/D at 1 Gsps and two channels of 16-bit D/A at 1 Gsps, enabling simultaneous sampling at a maximum rate of 1 Gsps



The 4DSP analog-to-digital / digital-to-analog board line offers users a wide range of performance options.

- FMC122, a single-channel 2.5-Gsps at 8 bits FMC-HPC ADC board that can alternatively be configured as a dual-channel 1.25-Gsps at 8 bits converter board
- FMC125, a quad-channel, trimode 8-bit FMC-HPC ADC board that can sample at 1.25, 2.5 or 5 Gsps
- FMC126, a quad-channel trimode 10-bit FMC-HPC ADC board that samples at 1.25, 2.5 or 5 Gsps

In addition, the company has released the latest version of its FM series of multifunction high-performance digital signal processor boards, the FM680. With the addition of the Virtex-6 FPGA, the XMC and PMC industry-standard VITA 42.3-compliant module allows users to implement ever-more-complex algorithms in their systems.

"It can be used for quite a few applications," said Vulliez. "As with our previous-generation products, customers can use the new card for software-defined radio, real-time processing, radar and sonar applications as well as video processing and compression."

Customers can configure the FM680 from a variety of memory options such as QDRII SRAM, DDR2 SDRAM and DDR3 SDRAM. Optionally, customers can also populate the card's unique user-configurable BLAST mounting sites with JPEG2000 codecs, solid-state flash drives or even specific logic devices or circuit designs.

For more information on the FMC cards, visit http://www.4dsp.com/fmc_list.php. For more information on the FM680, go to <http://www.4dsp.com/FM680.php>.

Xpress Yourself in Our Caption Contest

GETTY IMAGES



If you have a yen to Xercise your funny bone, here's your opportunity. We invite readers to step up to our verbal challenge and submit an engineering- or technology-related caption for this photograph showing a fraught moment during robot games. The image might inspire a caption like "The robots had succeeded in bringing their creator to their level. Now it was time. So much for artificial intelligence...."

Send your entries to xcell@xilinx.com. Include your name, job title, company affiliation and location, and acknowledge that you have read the contest rules (www.xilinx.com/xcellcontest). After due deliberation, we will print the submissions we like the best in the next issue of *Xcell Journal* and award the winner the new Xilinx® SP601 Evaluation Kit, our entry-level development environment for evaluating the Spartan®-6 family of FPGAs (approximate retail value, \$295; see <http://www.xilinx.com/sp601>). Runners-up will gain notoriety, fame and a cool, Xilinx-branded gift from our SWAG closet.

The deadline for submitting entries is 5 pm Pacific Time (PT) on June 18, 2010. So, get writing!

NO PURCHASE NECESSARY. You must be 18 or older and a resident of the fifty United States, the District of Columbia or Canada (excluding Quebec) to enter. Entries must be entirely original and must be received by 5:00 pm Pacific Time (PT) on June 18, 2010. Official rules available online at www.xilinx.com/xcellcontest. Sponsored by Xilinx, Inc. 2100 Logic Drive, San Jose, CA 95124.

MANOJ VISWAMBHARAN,

technical manager, Acceleration Technologies, at Credit Suisse won an SP601 Evaluation Kit with this caption for the photograph of the oversize Pucca and her weary friend that appeared in Issue 70 of *Xcell Journal*.



GETTY IMAGES

"This is the last time I try Internet dating."

Congratulations as well to our two runners-up:

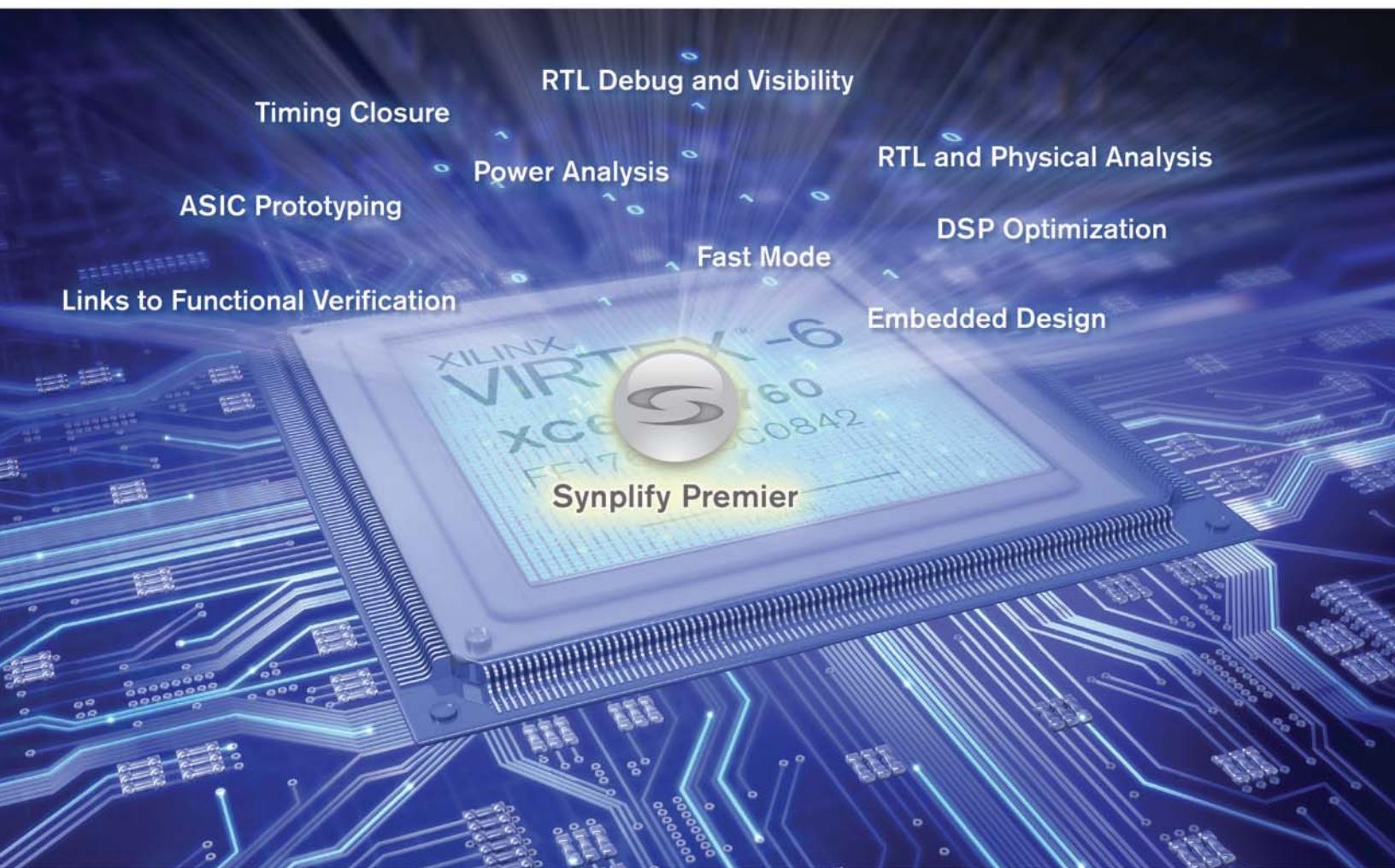
"Pucca and the engineer each regretted their choice of partner for the blind date."

— Paul Lindemann,
consultant, Montage Marketing

"Anime dating driven by Virtex®-6 FPGA power enables a little too much reality..."

— Keith Felton,
*group director of product management,
Cadence Design Systems Inc.*

Accelerate FPGA Design



Synplify Premier, the Ultimate in FPGA Implementation

The Synplify® Premier software from Synopsys® is the preferred implementation and debug environment for today's FPGAs. It provides a comprehensive suite of tools and technologies for advanced FPGA design as well as ASIC prototypers. The Synplify Premier solution addresses the toughest FPGA challenges including long turnaround times, timing-closure, power analysis, IP support, DSP implementation, debug, and ASIC compatibility including DesignWare® support. Synplify Premier is tightly integrated with Xilinx® back-end tools and provides highly accurate correlation to final timing resulting in shorter development schedules and improved quality of results.

For more information on how the Synplify Premier software
can help you achieve your design goals visit
<http://www.synopsys.com/fpga>



ACCELERATE CHANGE. ADVANCE INNOVATION.

The power to innovate is in your hands. Focus your energy on innovation and add differentiating value to your electronic systems. Xilinx Targeted Design Platforms integrate advanced FPGA silicon and design tools, IP cores, development boards, and reference designs—giving you the freedom to innovate without risk. Find out more at www.xilinx.com

© 2010 Xilinx, Inc. All rights reserved. Xilinx and the Xilinx logo are registered trademarks of Xilinx in the United States and other countries. All other trademarks are property of their respective holders.