ISSUE 76, THIRD QUARTER 2011

Xcell journal

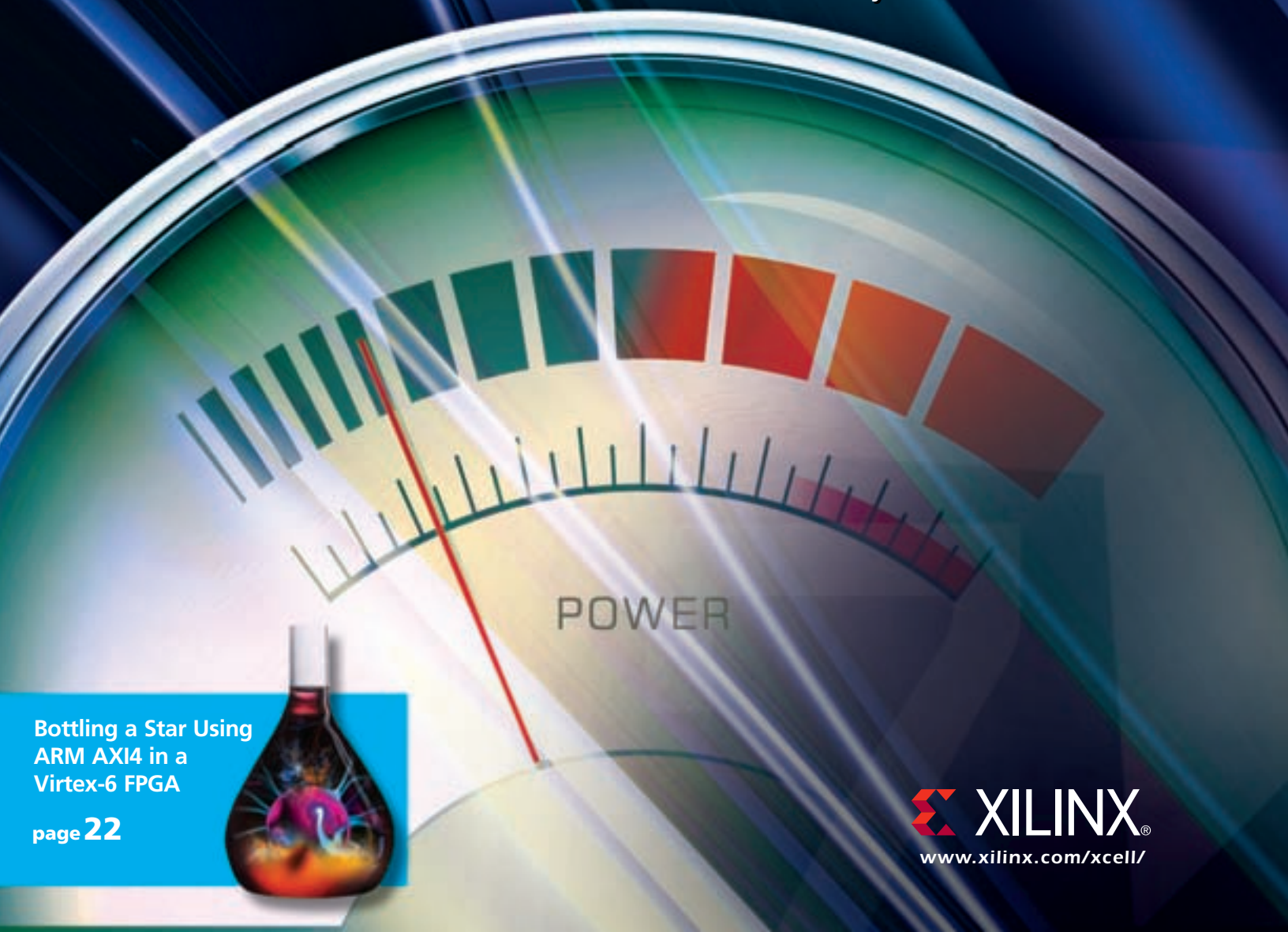
SOLUTIONS FOR A PROGRAMMABLE WORLD

How Xilinx Halved Power Draw in 7 Series FPGAs

FPGAs Enable Real-Time Optical Biopsies

Archiving FPGA Designs for Easy Updates

MIT Taps ESL Tools, FPGAs for System Architecture Course



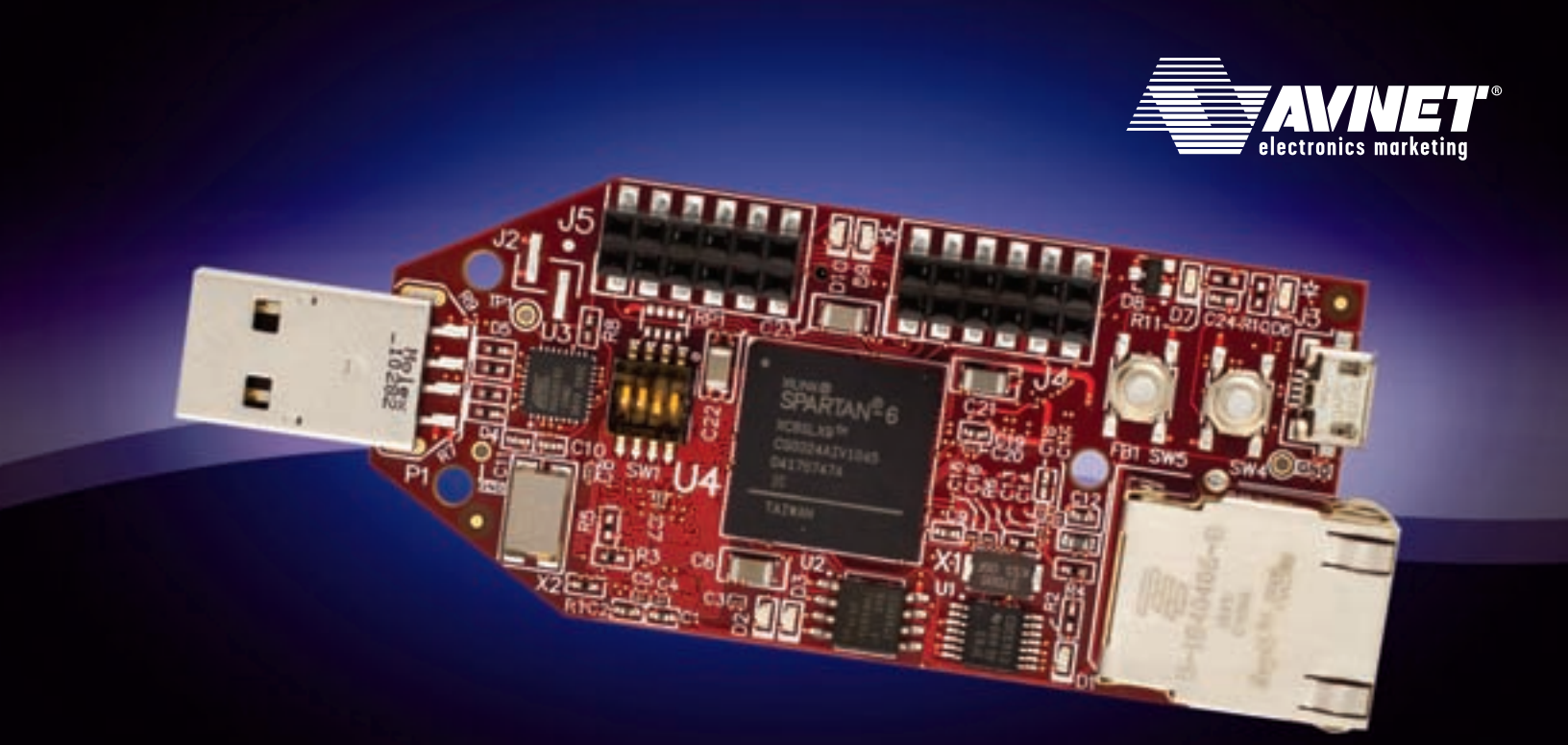
Bottling a Star Using ARM AXI4 in a Virtex-6 FPGA

page 22

XILINX®

www.xilinx.com/xcell/

AVNET electronics marketing



DESIGNED BY AVNET

SPEEDWAY DESIGN WORKSHOPS

New MicroBoard & Design Workshops Demonstrate the Versatility of Spartan®-6 FPGAs

Interested in exploring the MicroBlaze™ soft processor or Spartan®-6 FPGAs? Check out the low-cost Xilinx® Spartan-6 FPGA LX9 MicroBoard. Featuring pre-built MicroBlaze "systems," this kit enables software development similar to that of any standard off-the-shelf microprocessor. The included Software Development Kit (SDK) also provides a familiar Eclipse-based environment for writing and debugging code.

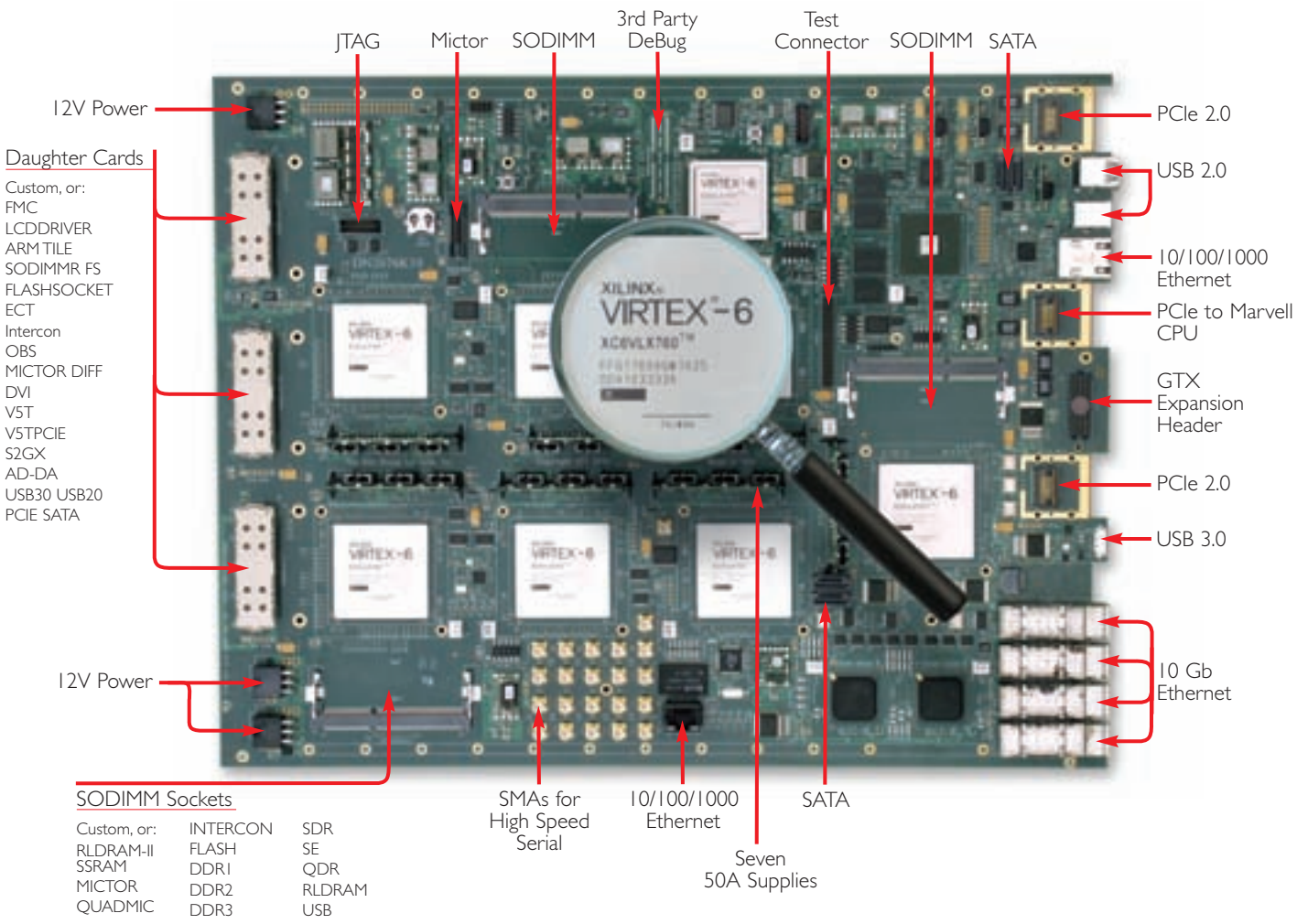
Want to take this kit for a test drive? Attend a SpeedWay Design Workshop™ hosted by Avnet and immerse yourself in a simplified Xilinx design experience.

Xilinx® Spartan®-6 FPGA LX9 MicroBoard Features

- Avnet Spartan-6 FPGA LX9 MicroBoard
- ISE® WebPACK® software with device locked SDK and ChipScope™ licenses
- Micro-USB and USB extension cables

To purchase this kit or register for a SpeedWay Design Workshop™, visit www.em.avnet.com/s6lx9speedway

Why build your own ASIC prototyping hardware?



DN2076K10 ASIC Prototyping Platform

All the gates and features you need are – off the shelf.

Time to market, engineering expense, complex fabrication, and board trouble-shooting all point to a 'buy' instead of 'build' decision. Proven FPGA boards from the Dini Group will provide you with a hardware solution that works — on time, and under budget. For eight generations of FPGAs we have created the biggest, fastest, and most versatile prototyping boards. You can see the benefits of this experience in our latest Virtex-6 Prototyping Platform.

We started with seven of the newest, most powerful FPGAs for 37 Million ASIC Gates on a single board. We hooked them up with FPGA to FPGA busses that run at 650 MHz (1.3 Gb/s in DDR mode) and made sure that 100% of the board resources are dedicated to your application. A Marvell MV78200 with Dual ARM CPUs provides any high speed interface you might want, and after FPGA configuration, these 1 GHz floating point processors are available for your use.

Stuffing options for this board are extensive. Useful configurations start below $25,000. You can spend six months plus building a board to your exact specifications, or start now with the board you need to get your design running at speed. Best of all, you can troubleshoot your design, not the board. Buy your prototyping hardware, and we will save you time and money.

DINI Group

www.dinigroup.com • 7469 Draper Avenue • La Jolla, CA 92037 • (858) 454-3419 • e-mail: sales@dinigroup.com

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

Xcell PUBLICATIONS
www.xilinx.com/xcell/

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

Xilinx Veteran Named ACM Fellow

"I became an engineer to build things, and that's what FPGAs allow you to do." So says Xilinx Fellow Steve Trimberger, the newly named Fellow of the Association for Computer Machinery, whom the ACM honored in June for his contributions to the design of programmable logic and reconfigurable architectures, and for the development of design automation tools that enable their use.

Xilinx's Steve Trimberger, at the ACM awards event in June.

Trimberger has been pivotal in the creation of many generations of FPGA architectures and EDA software since joining Xilinx in 1988. He currently holds more than 175 patents for Xilinx and has dozens more pending.

He began tinkering with EDA software as a student at the California Institute of Technology, from which he earned a BS degree in engineering and, in 1983, a PhD in computer science (gaining his MS from the University of California, Irvine, in between). "At Caltech, I wanted to design chips, but my chips didn't always work. So then I'd go create a tool so I'd be able to catch that problem if it ever happened again," said Trimberger. "After a few chips and a few debugging sessions, I started developing tools full time."

He did the same thing when he went to work at VLSI Technology Inc., a pioneering ASIC company. "Back then, every tool you developed was good because there weren't tools for pretty much anything," Trimberger said. "I guess everyone else's chips had problems, too."

Then, a few years later, Trimberger interviewed for a job at a pre-IPO chip company called Xilinx that had invented a new device called a field-programmable gate array.

"I heard about a company selling a logic chip that could be reprogrammed," he said. "I really wanted to use that chip, but I knew I had to move fast, because I didn't think they would be around that long." But it turned out that "FPGAs excelled at something ASICs could never do—give you unlimited do-overs." They had staying power.

At first, Trimberger mainly concentrated on software innovations to program FPGAs, but with the mentoring of then-VP of engineering Bill Carter, he soon began creating inventions for next-generation silicon architectures. "The tight linking of FPGAs and tools to program them has proved to be a key to success and is vitally important to the FPGA business," he said.

Trimberger's 175 patents span a wide range of design disciplines, from tools to circuits to full IC architectures. An early patent is for a "backwards simulator" that allowed engineers to essentially run a simulation in reverse to quickly pinpoint the origins of an error. Another puts the phenomenon of memory decay, in which a memory device loses its data when power is removed, to good use—Trimberger describes a way of using memory decay rates to tell how long a device has been powered off.

Trimberger has published three books: *Automated Performance Optimization of Custom Integrated Circuits*, *An Introduction to CAD for VLSI* and *Field Programmable Gate Array Technology*. He has also taught EDA software design to graduate students at Santa Clara University. In his spare time he coaches and supports high school teams in the FIRST Robotics competition (one of his teams won first place in 2011).

The ACM is the world's largest educational and scientific computing society. We join them in congratulating Steve Trimberger for a career spent advancing the state of the art in FPGAs, which you, our customers, in turn use to create remarkable innovations.

Mike Santarini
Publisher

Pass GO, Collect 3.6 Billion Samples per Second

X6 GSPS

Extremely Versatile

Complete chassis & embedded PC solutions available! Adapt to VPX, Cabled PCI-Express, PCI-Express, 64-bit PCI, and CompactPCI with our high-quality carriers.

Features

- Two 1.8 GSPS, 12-bit A/D Channels
- Single channel interleaved @ 3.6GHz
- +/-1V, AC-Coupled, 50 ohm, SMA Inputs
- Xilinx Virtex-6 SX315T/SX475T or LX240T
- 4 Banks of 1GB DRAM (4 GB total)
- Rugged levels available

VIRTEX 6 wireless ip cores

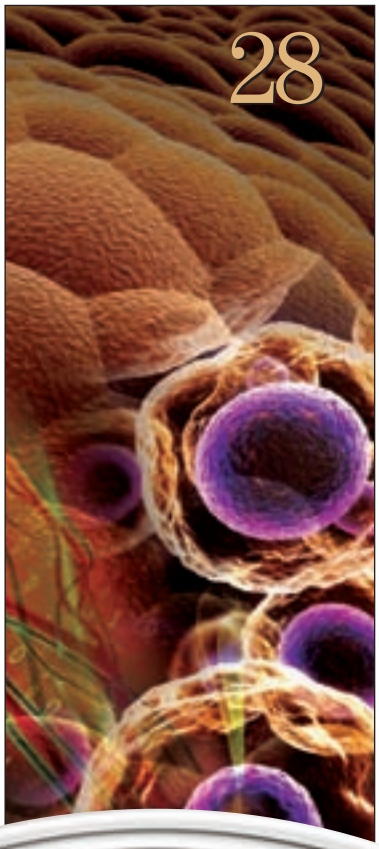
FrameWork Logic RoHS

805.578.4260 phone • www.innovative-dsp.com

Innovative Integration
a subsidiary of ISI
... real time solutions!

VIEWPOINTS

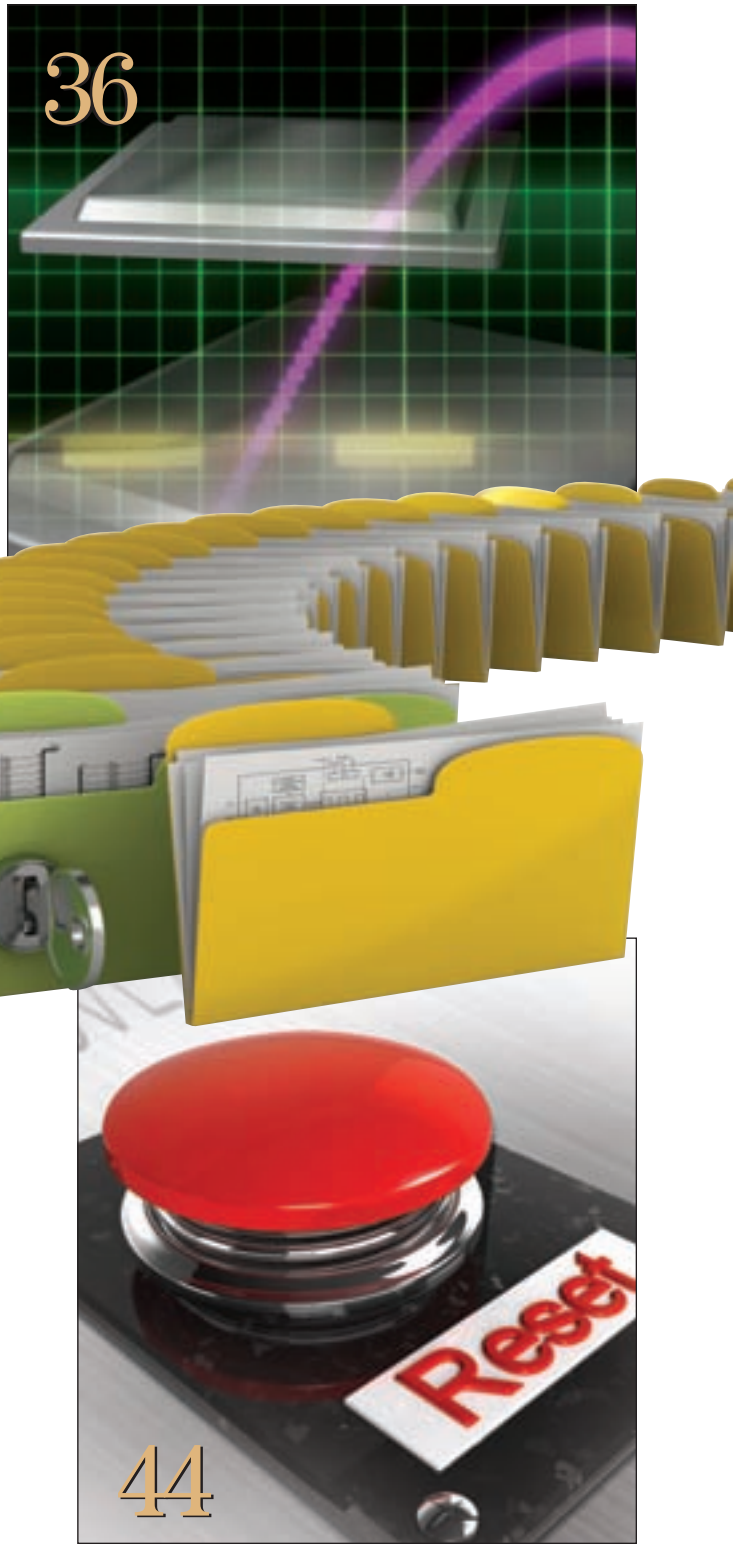
XCELLENCE BY DESIGN APPLICATION FEATURES



Cover Story

How Xilinx Halved Power Draw in 7 Series FPGAs

THE XILINX XPERIENCE FEATURES



XTRA READING

2010 APEX® AWARDS FOR PUBLICATION EXCELLENCE

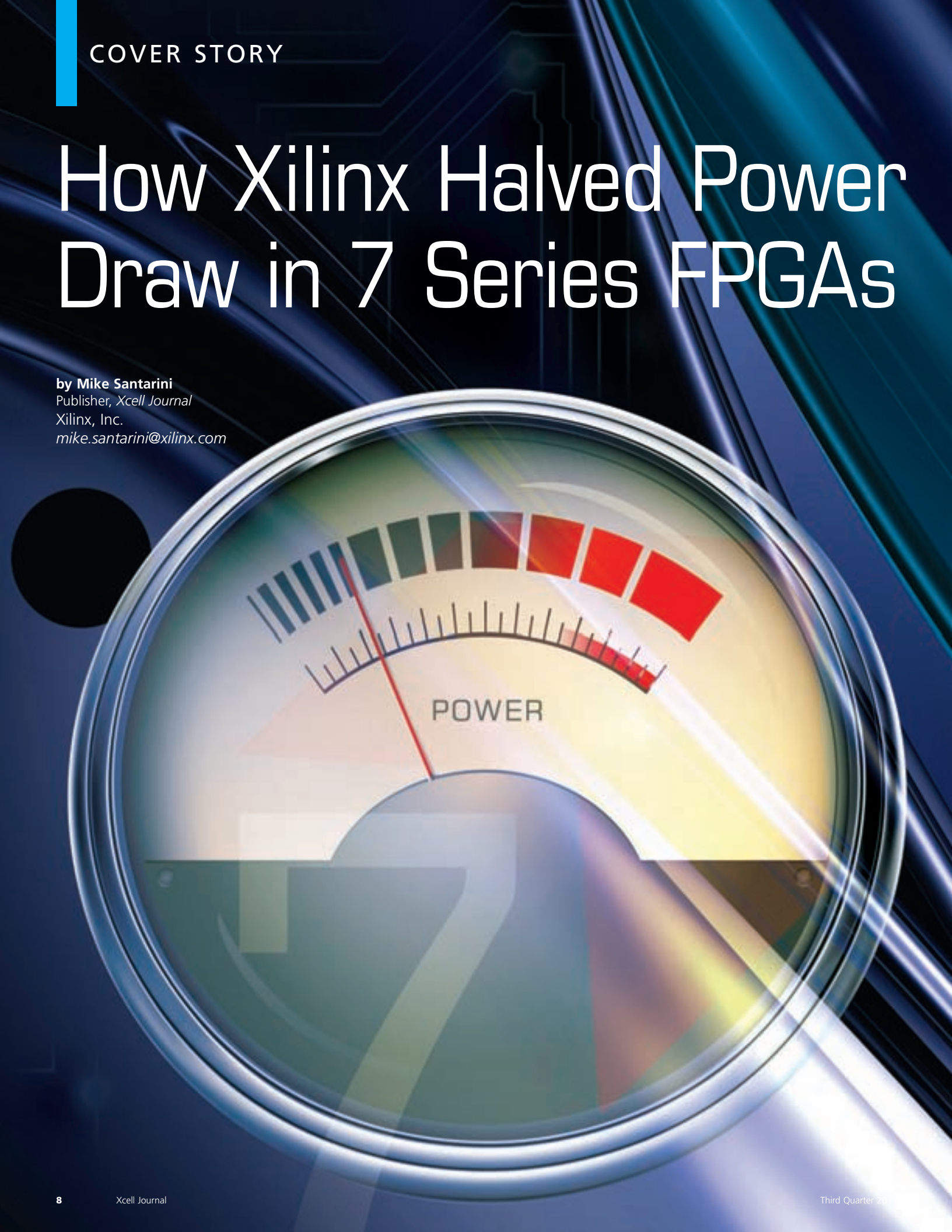
Excellence in Magazine & Journal Writing
2010, 2011

2011 APEX® AWARDS FOR PUBLICATION EXCELLENCE

Excellence in Magazine & Journal Design and Layout
2010, 2011

How Xilinx Halved Power Draw in 7 Series FPGAs

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



A silicon manufacturing process tailored for FPGAs and an innovative unified architecture allowed Xilinx to reduce power consumption by more than 50 percent over its previous-generation devices.

In interviewing several hundred customers in the process of defining the 7 series FPGA line, Xilinx chip architects found over and over again that one topic dominated the conversation: power. With such a clear customer mandate, Xilinx made power reduction and power management top priorities in its latest, 28-nanometer FPGAs, which began shipping to customers in March. In fact, 7 series FPGAs consume half the power of Xilinx's previous-generation devices, while still increasing logic performance, I/O performance and transceiver performance up to 28 Gbits/second and achieving record logic capacity (see Video).

A key enabler for this power reduction was Xilinx's choice to implement its series 7 FPGAs on TSMC's 28-nm HPL process, which Xilinx and TSMC developed specifically for FPGAs. Besides offering a slew of intrinsic advantages in terms of power, the process had ample headroom to enable power binning and voltage scaling, techniques that aren't available for FPGAs implemented in other processes. In addition to choosing a process ideally suited for FPGAs, Xilinx also refined device architectures to further cut power consumption.

This month, Xilinx will release revised power analysis tools to help designers evaluate the power profiles of Xilinx FPGAs.

TOP OF THE LIST

It's no secret why power management has moved to the top of the must-have list for most, if not all, FPGA users. The old rule of thumb held that if you were designing a system that your end customers would plug into a wall socket, you really didn't need to care too much about the power consumption of the FPGA you were using—you went for high performance and capacity when choosing one. How things have changed.

Over the last decade the industry has moved to new, faster semiconductor manufacturing processes that have a nasty side effect of leaky transistors. At the same time, system makers strive to differentiate their offerings by lowering the total cost of ownership or operating expenditures with lower-power products, and have created newer, innovative gadgets that require DC power (battery-based systems). Thus, lowering power and investing in system power management are something most customers have to do, even if they aren't targeting handheld devices. Whether you want it to or not, power demands your attention.

Video — 7 series devices consume half the power of previous-generation FPGAs. See *http://youtu.be/kVmHP-IFVaQ*.

POWER TO THE PEOPLE

At the 130-nm process node, transistors in ICs started to draw power even when a user placed the system in "standby" or "sleep" mode. This unintended current draw (often called static power or static leakage) got progressively worse with the introduction of 90-nm, 65-nm and 45-nm processes. At the 45-nm node, 30 to 60 percent of an average performing chip's power consumption was lost to static power under worst-case conditions. The remainder of the power budget went to dynamic power—the power the device consumes when running operations it was actually designed to handle. Higher-performing chips require even higher-performing transistors, which leak even more.

Wasting power is never a good thing, but static power loss has a much more serious consequence: It creates heat, and when added to the heat produced by dynamic power consumption, the transistors leak even more and in turn get hotter. That leads to more leakage and so on and so on. If left unchecked by proper cooling and power budgeting, this vicious cycle of leakage begetting heat and heat increasing leakage can shorten the lifespan of an IC or even lead to thermal runaway that abruptly causes catastrophic system failure. It has been widely reported that this was the phenomenon that felled the Nvidia ASIC at the heart of Microsoft's initial rev of its Xbox 360, resulting in a massive recall and redesign.

Many design groups have had to come up with tricks and techniques of their own to deal with the consequences of static power (see *EDN* article "Taking a Bite out of Power," *http://www.edn.com/article/460106-Taking_a_bite_out_of_power_techniques_for_low_power_ASIC_design.php*). Some designers employ schemes such as clock and power gating, or implement power islands in their designs. Many others deal with leakage by adding heat sinks, fans and even refrigeration and larger power circuitry to their systems for cooling. All these steps, however, add to the bill of materials and manpower costs of design projects.

Besides the general industrywide concerns about leakage, some companies have further incentives to lower power. Many companies today are jumping on the "green" bandwagon or are simply trying to differentiate their products by touting a lower cost of ownership or of operation, with systems that consume less power than competing systems and thus reduce electricity bills. This is especially true in networking and high-performance computing, where huge, hot systems must run reliably 24/7. The cost of powering these computing farms—and their cooling systems—can be enormous, so a savings of even a few watts per chip adds up. And of course, any system that is battery operated has power consumption as a top priority, as power directly affects the amount of runtime before the battery needs to be charged or replaced.

While FPGAs have a way to go before they will be used broadly in commercial mobile phones (one of the few markets where products sell in quantities that justify the design of an ASIC), the number of low-power applications using FPGAs is growing immensely. Among them are automotive infotainment and driver assistance systems; on-soldier battlefield secure communications electronics; handheld, mobile medical equipment; 3D TV and movie cameras; aircraft and space exploration vehicles.

THE HPL PROCESS—TAILOR-MADE FOR FPGAS

In creating the 7 series FPGAs, introduced last year (see cover story, *Xcell Journal* Issue 72, *http://www.xilinx.com/publications/archives/xcell/Xcell72.pdf*), Xilinx evaluated multiple 28-nm foundry processes and ultimately worked with TSMC to develop one specifically suited to FPGAs. Called High Performance, Low power (HPL), this new process employs high-k metal gate technology, which dramatically lowers leakage in transistors and affords the optimal mix of power and performance. Prior to HPL's advent, Xilinx and other FPGA companies had to choose between a given foundry's low-power (LP) process and its high-performance (HP) process, said Dave Myron, Xilinx's director of product management. The LP process had lower-performance mobile applications in its sight, while the HP was specifically crafted for beefy graphics chips and MPUs.

"Neither type of process has been the optimal fit for FPGAs," said Myron. "If you went with an LP process, we were leaving performance on the table, and if we went with an HP process, our devices consumed more power than we would have liked. We had a little wiggle room to tweak the process, but not as much as we would have liked."

FPGAs find their way into a broad number of applications, Myron went on, "but they don't quite have the performance requirements of a graphics chip or the extreme low-power requirements of ASICs in commercial mobile phones." Myron said that in working together on an FPGA formula, TSMC and Xilinx found the optimal mix of transistors—both high speed and low leakage. "With HPL we were able to tailor the process to be centered right in the sweet spot of the power-and-performance requirements of FPGA applications," said Myron (see Figure 1). "Because our devices are centered correctly, it means that customers don't have to make radical power vs. performance trade-offs to get the most out of their designs."

One key advantage of the HPL process, Myron said, is that it has a larger voltage headroom than 28-nm HP processes. That gives users a choice of operating the device's V_{CC} at a wider range of values, enabling a flexible power/performance strategy—which is not possible with a 28-nm HP process. As Figure 2 shows, in High-Performance Mode (V_{CC} = 1 volt), 28-nm HPL offers better performance than 28-nm HP at half the static power in the range of performance targets for FPGAs. In Low-Power Mode (V_{CC} = 0.9V), it offers 70 percent lower static power than 28-nm HP. The headroom in HPL delivers a larger number of dice on the distribution that have good performance, even at V_{CC} = 0.9V. Dynamic power also drops roughly 20 percent at this lower voltage.

Another mode available in the 7 series FPGAs is called Voltage ID (VID). Here, customers have the ability to reduce power through control of V_{CC} and take advantage of extra performance in some devices. Each device stores a voltage ID. The readable VID identifies the minimum voltage at which the part can operate to still meet performance specifications.

What's exciting about this extra headroom is the choices it opens for designers, Myron said. "Customers can choose to implement their current designs in a series 7 device and essentially halve the power consumption of that current design—or they can keep their original higher power budget and add more system functionality to the design until they fill the headroom," he said. "That saves overall system power and board space, and improves performance while cutting overall system costs dramatically."

Xilinx uses the HPL FPGA-optimized process for all three FPGA families in its 7 series as well as the new Zynq™-7000 Extensible Processing Platform. Xilinx treated all of the FPGA fabric in these devices the same way—a unified ASMBL™ architecture based on small, power-efficient blocks. That allows customers to migrate designs more easily across all of these device families: the Artix™-7 low-cost and lowest-power FPGAs; Kintex™-7 FPGAs offering the best price/performance; the Virtex®-7 family, boasting the best performance and capacity; and the Zynq-7000 Extensible Processing Platform, which packs an embedded ARM dual-core Cortex™-A9 processor and is primed for embedded applications (see cover story, *Xcell Journal* Issue 75, *http://www.xilinx.com/publications/archives/xcell/Xcell75.pdf*).

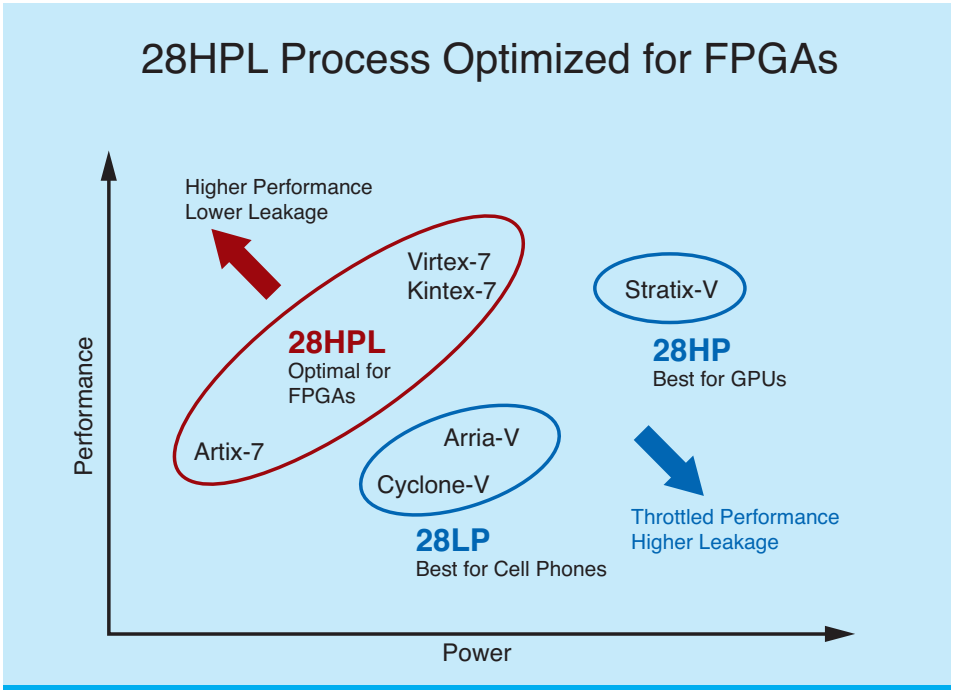


Figure 1 – HPL sits at the sweet spot in terms of the power/performance requirements of FPGA applications.

While FPGA competitors continue to implement variations of a single architecture in HP and LP processes, Xilinx firmly believes its unified silicon architecture implemented on an FPGA-tailored process will speed the maturation of FPGA technology as a programmable platform, in which the silicon serves as the foundation but not the entirety of a system solution. In Xilinx's programmable-platform strategy, introduced with the Virtex-6 and Spartan®-6 FPGA generation (see

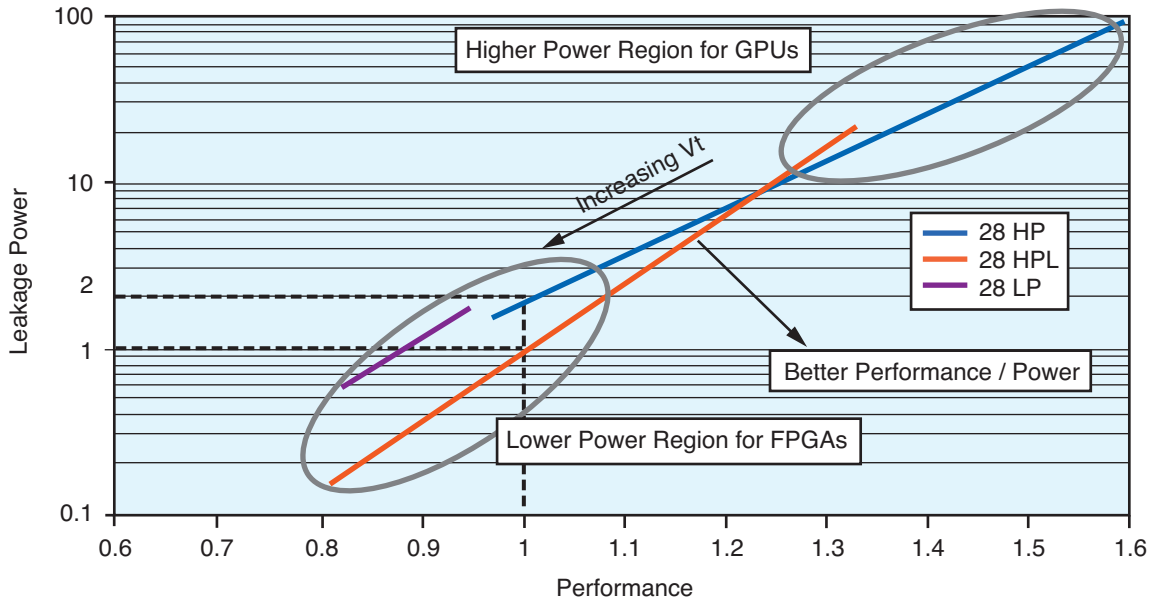


Figure 2 – Performance vs. leakage in 28-nm HPL, 28-nm high-performance (HP) and 28-nm low-power (LP) processes

cover story, *Xcell Journal* Issue 68, *http://www.xilinx.com/publications/archives/xcell/Xcell68.pdf*), Xilinx not only provides cutting-edge silicon but also market- and application-specific development boards, IP, tools and documentation to help customers create innovations quickly.

Myron notes that the move to an optimized process and unified architecture isn't unprecedented or radical in the semiconductor industry—it adapts to FPGAs a strategy Intel Corp. pioneered and has successfully implemented over the last five years (see the white paper "Inside Intel Core Microarchitecture," *http://software.intel.com/file/18374/*).

"In 2006, Intel stopped using multiple microprocessor architectures and implemented its single strongest hardware architecture—the Intel Core microarchitecture—on a single silicon process and across multiple product lines, from high-computing server products down to the mobile notebook products," said Myron, who formerly worked at Intel. "Why does a company like Intel, with seemingly unlimited resources at its disposal, focus on just one hardware architecture? The answer is, they want to do one thing and do it very well by focusing their resources behind a single architecture that can be scaled to fit different application needs and to make it easier for their customers to utilize this same architecture in multiple applications with a minimum of redesign."

The same is certainly true for Xilinx, said Myron. "By selecting a common architecture for the 7 series FPGAs, we can focus our software-engineering teams on optimizing quality of results without dividing their efforts over multiple product families. In a similar fashion, our customers have expressed a strong desire to optimize IP reuse," he said. "Maintaining a single architecture facilitates IP reuse with a minimum of engineering effort compared to doing so across multiple, different architectures."

The high-k metal gate HPL process was just the first stage in reducing static power. Xilinx took the effort a step further by making adjustments to the 7 series device architecture. In prior Xilinx FPGA releases, Xilinx added the ability for users to employ power gating to shut off unused transceivers, phase-locked loops (PLLs), digital clock managers and I/O. In series 7, designers can now do the same with unused block RAM. Since block RAM could account for as much as 30 percent of a given device's leakage current, power gating can make a huge difference.

TOTAL SYSTEM POWER REDUCTION

Myron said that while the choice of the high-k metal gate HPL silicon process reduced static power and dynamic power usage greatly, Xilinx took additional steps to reduce the total system power of the 7 series devices. That is, if one were to look at total system power, it consists of static leakage, dynamic power, I/O power and transceiver power (Figure 3).

Dynamic power reduction

Xilinx power guru Matt Klein, a Distinguished Engineer who has been involved in driving power reduction into Xilinx FPGAs, said dynamic power in the FPGA's logic is based on the standard "C V squared f" equation:

$$Dynamic\ power = \mu\ x f_{clk}\ x\ C_L\ x\ V_{DD}^2$$

For the C, or capacitance, term of the equation, many blocks in the Xilinx FPGA are already architected for low dynamic power thanks to optimal design, which significantly reduces capacitance. Xilinx has rearchitected some of these blocks further to be more compact and even lower in capacitance. "Some of the blocks in Xilinx FPGAs—including the DSP48s—consume less dynamic power than those in other 28-nm FPGAs, even though the nominal voltage is 1V vs. 0.85V," said Klein. "Xilinx offers voltage-scaling options to further reduce dynamic power." Also, he said, the fclk, or frequency, affects dynamic power "in a linear fashion."

Klein said that users can also augment the "alpha," or activity, factor of their designs to perform intelligent clock gating to reduce dynamic power. With this method, designers control the activity of given blocks. This technique, however, takes quite a bit of time to implement, especially with a large FPGA design, so most FPGA users often don't do it.

But Klein said there are alternatives. He said that all 7 series FPGAs have a clock hierarchy, which allows designers to program a given design to enable only the clock resources that are needed. This greatly drops clock load power. Additionally, designers can gate clocks at three levels: a global clock gating, a regional clock gating and a clock gating via a clock enable (CE) at local resources such as flip-flops.

"In the Xilinx FPGAs there are fundamentally eight flip-flops per slice. They share a common clock enable, but unlike previous architectures, the clock enable locally gates the clock and also stops the flip-flop toggling," said Klein. "Now, with this hardware, the ISE® design tools can automatically suppress unnecessary switching by looking for cases where flip-flop outputs will not be used by a downstream target. This is done by a logical examination and post-synthesis. The tool then generates local clock enables. Users can activate these features in the map phase via the use of the -power high or -power XE options."

Klein said that this automatic intelligent clock gating can save as much as 30 percent of the dynamic power of logic, and an average of 18 percent. "The additional logic to generate the intelligent clock gates is less than 1 percent, so this is a great benefit in reducing dynamic power," said Klein.

Users can also employ intelligent clock gating on block RAM. Most designers and synthesis tools will leave the enables of the block RAM at a static 1. Klein suggests considering a block RAM with address inputs and data outputs. The data out may be used downstream, but is sometimes selected via a multiplexor control signal, called sel. First, it is unnecessary to enable the block RAM when no write is occurring or if the read address has not changed from the last cycle. Second, if the system is not using the output of the block RAM on a given cycle it is unnecessary to enable it for reading.

Employing similar methods to those used to generate clock enables for flip-flops, ISE automatically generates CE

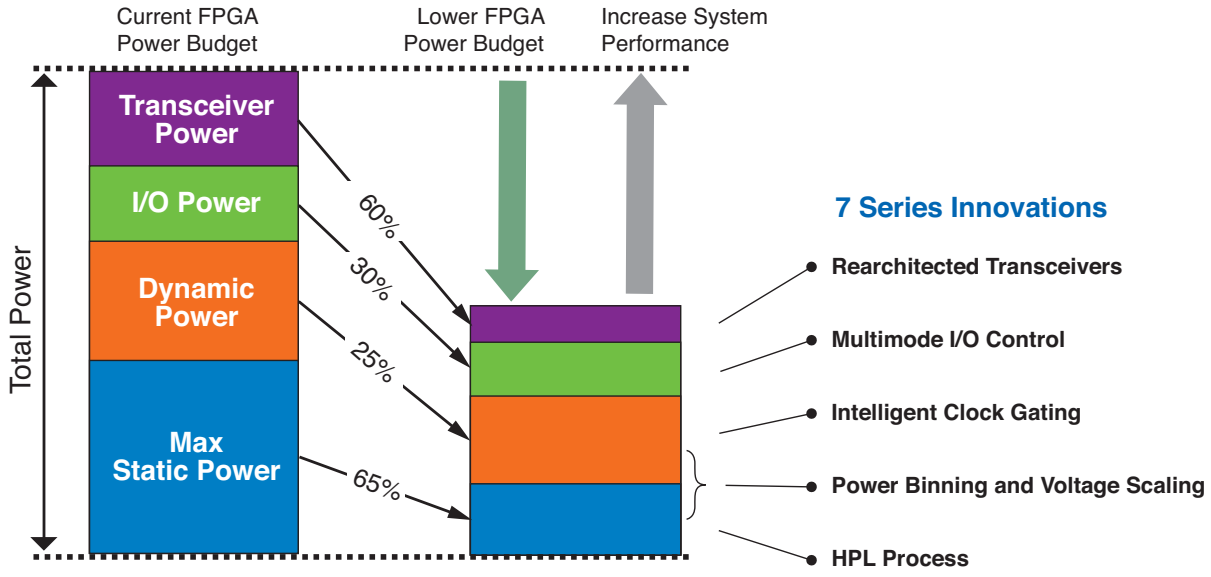


Figure 3 – Xilinx optimized all aspects of power in series 7 devices to achieve greater than 50 percent reduction in total power power consumption over its previous-generation devices.

(or clock enable) signals on a cycle-by-cycle basis. "For block RAM the savings are even greater and we find up to 70 percent lower block RAM power with an average of 30 percent reduction, again with very little logic overhead," said Klein. "Xilinx also offers both CORE Generator™ and XST options to construct power-efficient block RAM arrays, which can save up to 75 percent of the dynamic power of those block RAMs in an array."

I/O power reduction

Of course, there's more to the power picture than static and dynamic power. Total system power consists of two other types as well: I/O power and transceiver power.

To lower the power consumed by high-speed I/O, Xilinx added multimode I/O control and rearchitected its transceivers. Klein said that multimode I/O control delivers significant I/O power savings, particularly for memory interfaces: up to 50 percent for memory writes and as much as 75 percent power savings for memory idle state.

The first of these new power reduction features is valuable during a memory write: The I/O hardware automatically disables the IBUF (input buffer) during a write to external memory devices such as DDR2 and DDR3. "Since the input buffer is a referenced receiver, it burns DC power, independent of toggle rate, so now during a memory write this DC power is removed and the saving is proportional to the percent of write," said Klein. "During a memory write this feature saves an additional 50 percent of the total power compared with only disabling the termination."

The second power reduction feature added to the I/O of all 7 series FPGAs is the ability for the user to disable IBUF and the termination during times when the memory bus is idle. "Normally, during a bus idle period you need to get off the bus, but this would look like a memory read and without this [disabling] feature, both the termination and IBUF will burn power," said Klein. "By disabling them, the 7 series I/O will consume 75 percent less power compared to leaving the termination and input receiver on."

Xilinx also lowered the V_{CCAUX} voltage from 2.5V to 1.8V. This saves roughly 30 percent on power consumption for all blocks powered by V_{CCAUX}, including the PLL, IDELAY, input and output buffers, and configuration logic.

These new features are big benefits vs. Virtex-6 and other FPGAs for high-performance memory interfaces.

Transceiver power reduction

Transceiver power is another key contributor to a device's total power. Myron said Xilinx initially provided pretty conservative transceiver power figures when first announcing the 7 series power estimates in its XPower Estimator (XPE) tool. Xilinx has since further refined GTP and GTH transceiver power and correlated its tools to silicon results. The latest XPE release (version 13.2) reflects these new numbers more accurately.

"For the Artix-7 GTP, which offers up to 6.75-Gbps performance, the complete transceiver power is 60 percent lower compared to the Spartan-6 GTP at equivalent performance," said Myron. "We did this to satisfy the low-end market's need for absolute lowest power and cost. We also significantly lowered the Virtex-7 GTH power." That device, which can have as many as 96 transceivers, is used in high-bandwidth applications "where transceiver power can be a major contribution to total power," he said. "This put us on par in transceiver power with the competing 28-nm offering."

Power binning and voltage scaling

One of the most interesting power-savings innovations in series 7 is the ability for Xilinx to offer customers power-binning and voltage-scaling options of its parts that are capable of delivering even lower power but at the same performance as the standard versions. "This is made possible with the headroom available through the versatile 28-nm HPL process, and means no other 28-nm ven-

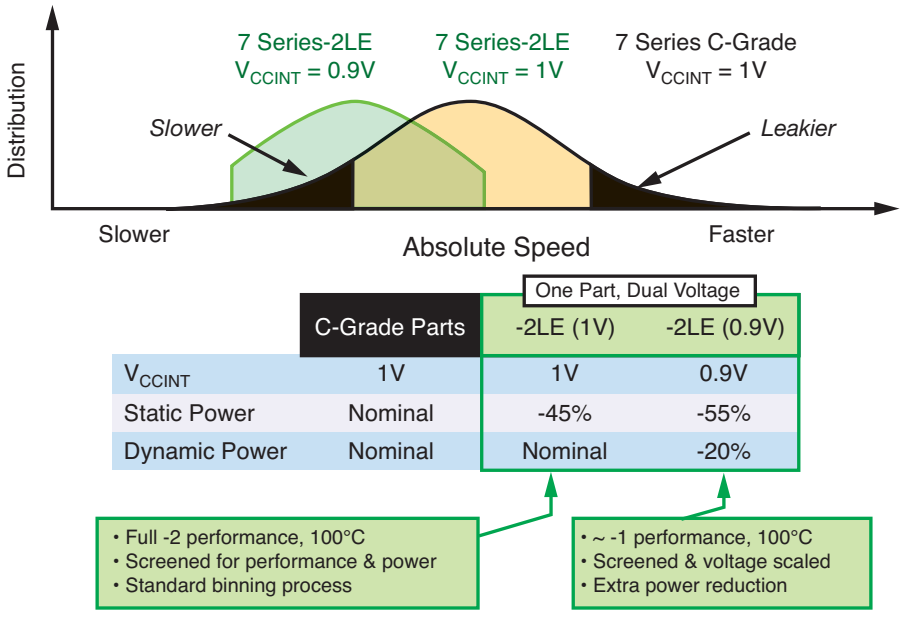


Figure 4 – Headroom in the 28-nm HPL process enables power binning and voltage scaling.

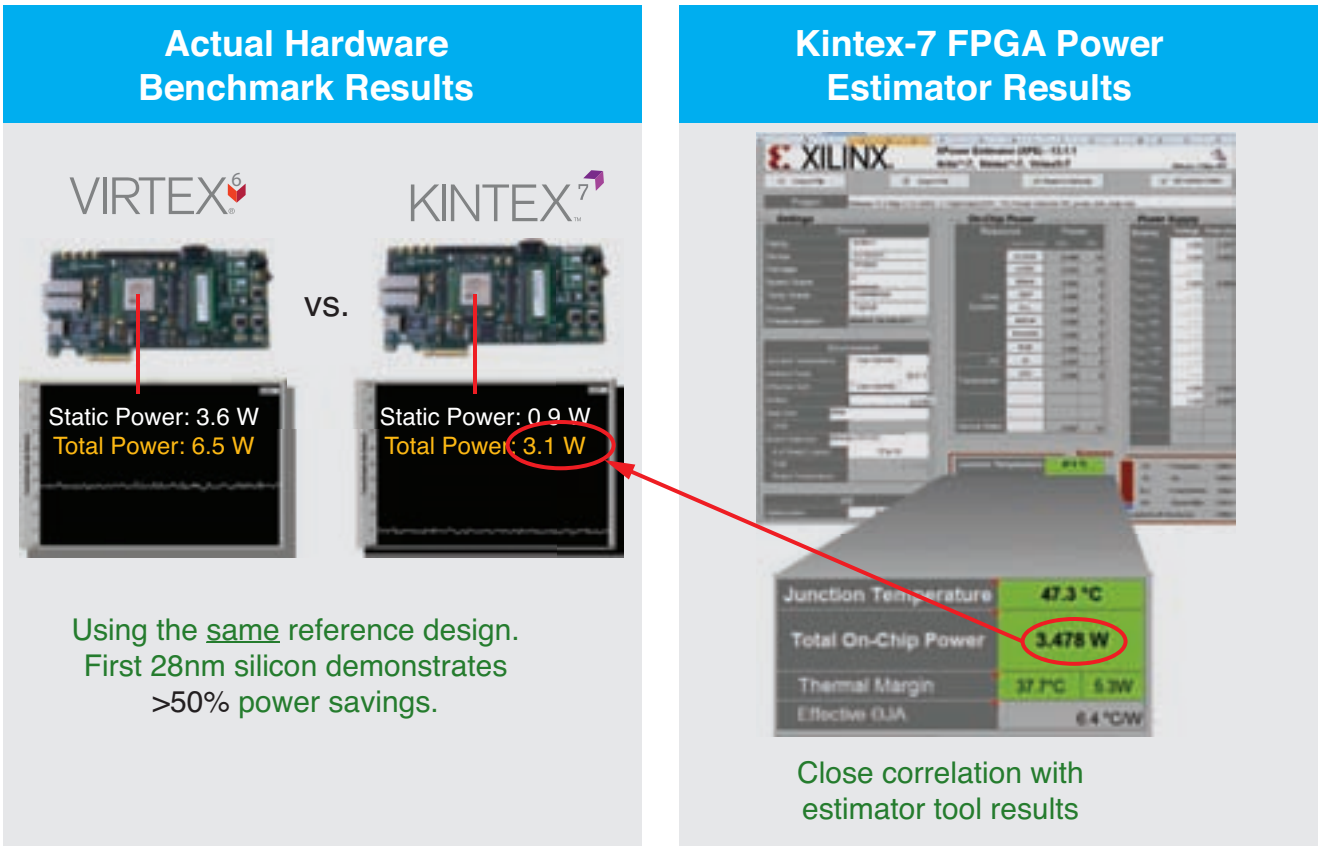


Figure 5 – The XPower Estimator (XPE) tool allows design teams to better evaluate the power profile of Xilinx FPGAs and compare them to competing offerings.

dor can provide you such a power-optimized option," said Myron. "How was Xilinx able to achieve this? If we first look at the standard part distribution [Figure 4], all parts in this curve run at 1V and have nominal static and dynamic power. We then remove the parts of distribution that are too slow or too leaky to give us the distribution for the -2L devices. The -2L devices operate at the same 1V core voltage, and therefore deliver the same -2 performance as the commercial or industrial counterparts, and can function at up to 100°C. The -2LE devices offer 45 percent static power saving, and are part of the standard binning process, and therefore there is no issue with availability of these devices. We then take the -2LE part and screen it to make sure that it's capable of running at 0.9V. By lowering core voltage to 0.9V, the power-optimized -2LE part can provide up to 55 percent static power reduction and 20 percent dynamic power reduction compared to the standard commercial devices."

AND THE BENCHMARK SAYS?

While competitors are apt to argue that Xilinx is fielding a one-size-fits-all approach at the 28-nm node, the company that invented the FPGA is quite confident that the 7 series is yet another innovation milestone. Xilinx has put together comprehensive benchmarks that show the 7 series is the optimal mix for the entire range of applications you target with FPGAs. Customers can view a number of benchmarks Xilinx has published at *http://www.xilinx.com/publications/technology/power-advantage/7-series-power-benchmark-summary.pdf* and view a TechOnline webinar at *http://seminar2.techonline.com/registration/wcIndex.cgi?sessionID=xilinx_jun1411*.

EMPOWER YOURSELF

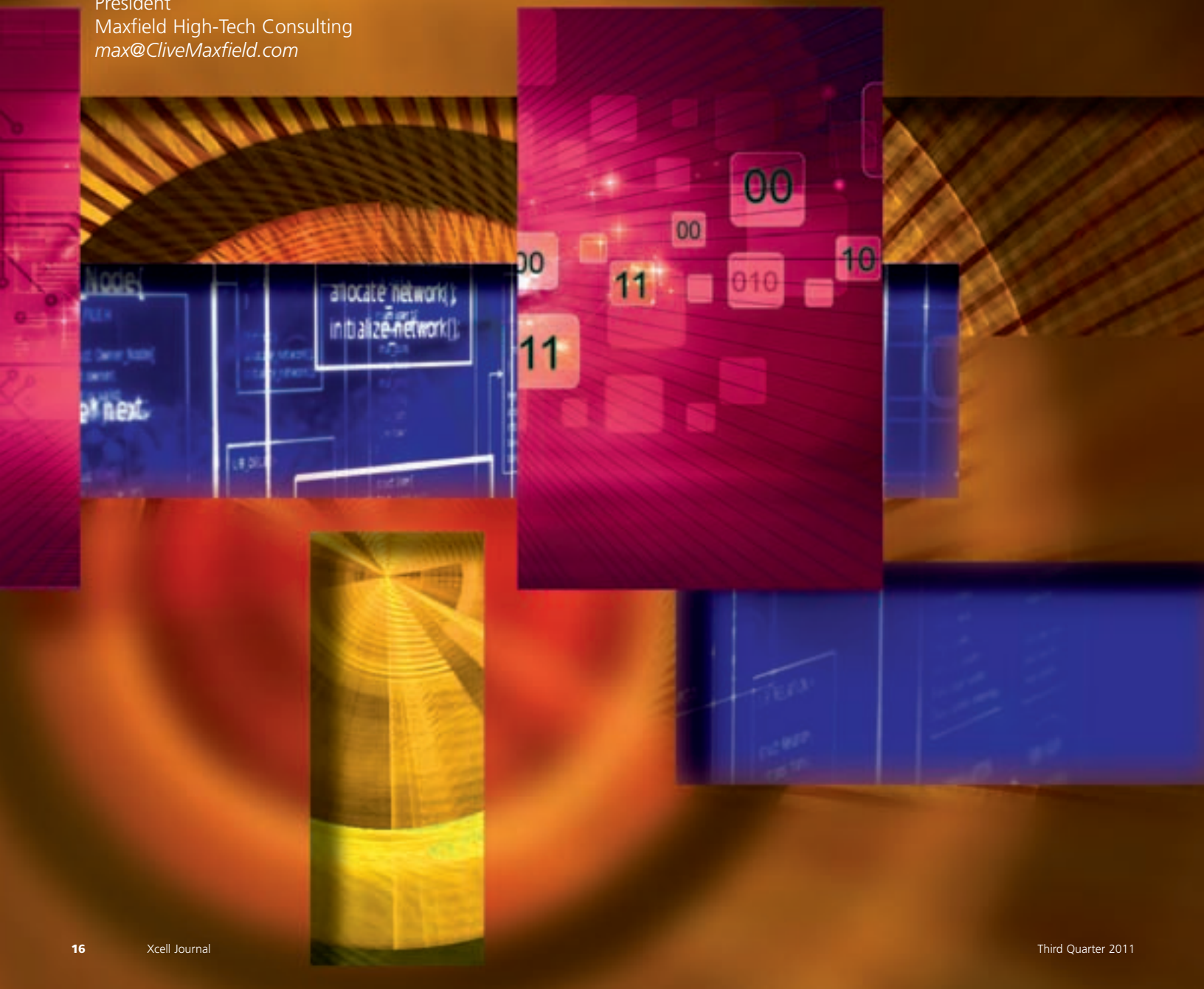
The latest version of the XPE power estimator tool, release version 13.2 (Figure 5), offers updated data on 7 series devices reflecting the recent product changes. It also provides the lower power data for the rearchitected GTP and GTH transceivers. By popular demand, this version of the tool also provides customers with necessary max-power data for their worst-case power supply and thermal planning.

For more details on power management of 7 series devices and benchmark information, read the 7 series white paper entitled "Lowering Power at 28 nm with Xilinx 7 Series FPGAs," which can be found at *www.xilinx.com/power*.

For further details on the 7 series power advantage, visit *http://www.xilinx.com/products/technology/power/index.htm*.

MIT Prof Uses ESL Tools, FPGAs to Teach System Architecture

By Clive (Max) Maxfield
President
Maxfield High-Tech Consulting
max@CliveMaxfield.com



A master's level course at MIT is changing the way educators teach digital design.

The last time I was on the receiving end of formal education was deep in the mists of time (circa the end of the 1970s). My final project for my control engineering degree was a digital controller that could display color text and "chunky graphics" on a cathode-ray tube. The entire design was implemented using cheap-and-cheerful 74-series TTL chips, each of which contained only a few simple logic gates or registers.

We didn't have computer-aided tools like schematic-capture systems or logic simulators (the programs I wrote for my computer class were entered on a teleprinter and stored on punched cards). So my design was captured as a gate-level schematic using pencil and paper; any proof-of-concept, testing and debug took place after I'd soldered everything together.

Not surprisingly, I didn't have the luxury of evaluating different architectural scenarios to see which would give me the best results. I just opted for an architecture I thought could "do the job" and I remember breathing a deep sigh of relief when my controller finally displayed a "Hello Max" message on the screen.

Today's chips, by contrast, offer designers mind-boggling logic capacities and resources to solve their problems. Along with design size, however, comes complexity, which is making it harder and harder to meet cost goals and performance, power and area specifications.

Decisions made early in the design cycle have the most impact with regard to all aspects of the final chip. For example, industry analyst Gary Smith estimates that 80 percent of a product's cost is determined during the first 20 percent of its development cycle. This means that it is absolutely imperative to select the optimum hardware architecture as early as possible in the development process.

But how can you teach this sort of thing to engineering students? With so much groundwork to be laid in the foundations of electrical engineering, and with limited time, universities historically haven't been able to focus on teaching architecture to the depth that is now required. A master's-level complex digital design course at MIT is trying to change all that. By leveraging the combination of FPGAs (through the Xilinx University Program) and real-world electronic system-level (ESL) design, which supports architectural exploration at higher levels of hardware abstraction, students are accomplishing in weeks what would have required an entire school year, or more, of study in the past.

WELCOME TO 6.375

I recently heard about a course called 6.375 at the Massachusetts Institute of Technology (MIT). It seems this course is changing the playing field when it comes to teaching digital design. In particular, a key focus of 6.375 is the use of architectural exploration to home in on optimal designs. The thing that really intrigued me is that the course is a mere 13 weeks long, of which the students have only six weeks to design, implement and verify their final projects. But these projects are of a complexity that would bring grizzled, practicing engineers to their knees, so how can this be possible?

First I bounced over to the MIT website (*http://csg.csail.mit.edu/6.375/6_375_2011_www/index.html*), where I read: "6.375 is a project-oriented subject teaching a new method for designing multimillion-gate hardware designs using high-level synthesis tools in conjunction with standard commercial EDA tools. The emphasis is on modular and robust designs; reusable modules; correctness by construction; architectural exploration; meeting area and timing constraints; and developing functional FPGA prototypes. This subject relies on high-level architectural knowledge and programming expertise rather than knowledge of low-level circuit design."

Well, this certainly sounds jolly interesting, but what does it mean in the real world? In order to learn more, I called Professor Arvind, the Johnson Professor of Computer Science and Engineering at MIT and a member of the Computer Science and Artificial Intelligence Laboratory. Arvind inaugurated 6.375 around seven years ago and has been evolving the course ever since. From what I hear, this has been quite an adventure.

Professor Arvind, the muscle behind 6.375

When 6.375 started, its focus was ASIC design. There were several problems with this, not the least that ASICs are so complex and there were too many tools involved in order to achieve anything realistic. Also, since the department didn't have the ability to fabricate the chips, everything was evaluated using software simulation, whose relatively slow speed limited the amount of testing that could be performed. And perhaps the most

The course, which draws a mix of computer science and electrical engineering majors, combines an ESL design and verification environment with a state-of-the-art FPGA development system from Xilinx that was designed with universities in mind.

important thing was that the lack of physical chips to play with meant that the class was not as stimulating for the students as Arvind had wished.

A few years into the course it was decided to switch to FPGAs (the curriculum largely ignores the special properties of FPGAs and concentrates on straightforward RTL design), in the belief that having physical realizations of their designs would be significantly more exciting for the students. Another big consideration was that software simulation takes so long and runs out of steam when it comes to the tremendous amount of vectors required to fully test today's complex projects. Many designs don't even start to exhibit interesting or corner-case behavior until a long sequence (perhaps tens or hundreds of millions) of test vectors has been processed.

Today, the course—which draws a mix of computer science and electrical engineering majors—features the combination of an ESL design and verification environment coupled with a state-of-the-art FPGA development system from Xilinx that was designed with universities in mind.

THE FIRST SIX WEEKS

The first half of the course uses examples of increasing complexity to introduce the Bluespec hardware description language (HDL) and associated design and verification environment. The second half is devoted to the students' projects. The students know hardly anything about Bluespec or hardware design before they start (this year, none of the students knew Bluespec and only three had rudimentary hardware design experience). Nevertheless, they tackle complex projects.

The first three weeks are devoted to teaching Bluespec SystemVerilog (BSV), an ESL hardware description language that is based on the concept of guarded atomic actions (a high-level abstraction for describing complex concurrent systems). BSV allows students to quickly and concisely capture their designs at a high level of abstraction (see Figure 1). These high-level, cycle-accurate representations can be simulated an order of magnitude faster than their standard Verilog equivalents. Of particular interest is the fact that the designs can be highly parameterized so as to facilitate architectural exploration.

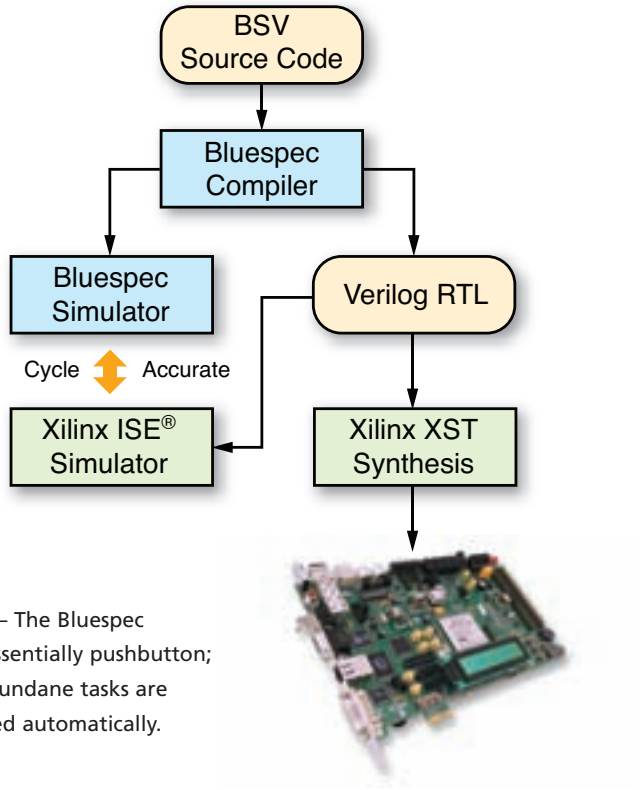


Figure 1 – The Bluespec flow is essentially pushbutton; all the mundane tasks are performed automatically.

In the fourth week the students learn how to use the FPGA development boards. These are Xilinx® XUPV5-LX110T development systems (Figure 2), a powerful and versatile platform packaged and priced for academia. The XUPV505-LX110T is a feature-rich, general-purpose evaluation and development platform with onboard memory and industry-standard connectivity interfaces. This provides a unified platform for teaching and research in disciplines such as digital design, embedded systems, digital signal processing, operating systems, networking, and video and image processing.

The students take the BSV test lab representations that they created in the first three weeks and synthesize them into corresponding Verilog RTL representations. (Arvind tells me that the students regard BSV vs. standard Verilog in the same way software developers regard C/C++ vs. assembly

Figure 2 – The powerful and versatile Xilinx XUPV5-LX110T development system is priced for academia.

language.) The Verilog is then synthesized into an equivalent gate-level representation that is loaded into the FPGA development board.

When FPGAs were first introduced to the course, things weren't quite as easy as they are now, and the students tended to spend too much time bringing up the FPGA infrastructure instead of working on their designs. Today, after much work by Bluespec, Xilinx and MIT students, the entire flow is essentially pushbutton, with all of the mundane tasks performed automatically behind the scenes. Now, the students no longer have to spend time worrying about getting the FPGAs to work—their focus is all about the architecture of their designs.

One of the things that helps keep things simple is that after the design has been synthesized and loaded into the FPGA development board, students continue to employ the original testbench they used to verify the high-level BSV representation by means of software simulation. They may create the testbench itself in BSV or C/C++. The interfacing between the testbench running on a PC and the FPGA development platform is achieved using the Standard Co-Emulation Modeling Interface (SCE-MI). Once again, all of this is largely transparent to the students.

The fifth and sixth weeks are devoted to labs on processor design—specifically, working with a pipelined processor core, bringing this core up on the FPGA development board and then writing C/C++ programs and executing them on the core running in the FPGA.

Working in teams of two or three, the students spend the sixth week (and spring break!) deciding on their projects, presenting these projects to the rest of the class in the seventh week and receiving a final approval from Arvind. They devote the next six weeks to designing, capturing, testing, debugging and verifying these projects. This is where the fun really starts.

ONLY SIX WEEKS TO DO WHAT?

Arvind's goal has always been for the students to work on sophisticated designs, but even he is surprised at the level of complexity that he's seeing. He notes that even for practicing engineers these are nontrivial projects to realize in only six weeks. He also says that when he describes the things his students are doing to people in the industry, their reaction is often "What? You must be joking!" The sidebar offers a peek at this year's projects.

One of the things Arvind is particularly interested in is the creation and use of intellectual property (IP). He encourages the students to use as much IP in their designs as they can find—both from previous years' projects and from the Internet. He also exhorts the students to produce their own IP blocks in a form that that will be of use to future classes. "The students very quickly learn that IP is not so easy to use unless it's been created and documented in an appropriate manner," he said. "This includes being designed in a highly parameterized way."

The ultimate goal of the class is not simply the creation of very complex designs in a very short period of time, but to also to evaluate different architectural scenarios so as to understand the effects alternative architectures have in terms of the area (resources), power consumption and performance/throughput of their corresponding implementations.

"I believe that the students' ability to perform architectural exploration is absolutely essential," said Arvind. "The combination of BSV at the front end with the ability to run millions of vectors on the FPGA boards for power/performance profiling at the back end allows the students to evaluate the effects of different architectures in a way that simply wouldn't have been possible just a few years ago. Today's ultramodern tools and techniques offer fantastic educational possibilities—it's incredible what clever people can do given the right tools."

WHO 'NOSE' WHAT THE FUTURE HOLDS?

I heard an interesting factoid the other day that struck me as being strangely pertinent to these discussions. El Capitan is a 3,000-foot vertical rock formation in Yosemite National Park. This granite monolith is one of the world's favorite challenges for rock climbers.

Once considered impossible to climb, El Capitan is now the standard for big-wall climbing. Today there are numerous established routes on

El Capitan in Yosemite National Park

both faces, the Southwest and the Southeast, but the most popular and historically famous route is the Nose, which follows the prow.

Believe it or not, the first ascent of the Nose, which occurred in 1958 by a team led by Warren Harding, took 45 days using fixed ropes. Seventeen years later, in 1975, Jim Bridwell, John Long and Billy Westbay made the first one-day ascent. In November 2010, Dean Potter and Sean Leary set a new speed record for the Nose, climbing the entire route in just two hours, 36 minutes and 45 seconds.

How is it possible to go from 45 days to only a couple of hours? Well, today's climbers operate under completely dif-

Sophisticated projects from student engineers

Seasoned engineers might balk at tackling some of the designs the MIT students created.

Project 1: Optical Flow Algorithm; Adam Wahab, Jud Porter and Mike Thomson, mentored by Abhinav Agarwal. Optical flow algorithms are used to detect the relative direction and magnitude of environmental motion observed in reference to an "observer." Optical flow has a wide range of applications, especially in robotics. The goal of this project was to develop an implementation of the Lucas-Kanade algorithm that could be incorporated into the Harvard RoboBee project, which aims to build micromechanical, autonomous, biologically inspired robots able to flap their wings (*http://robobees.seas.harvard.edu*). "It was amazing to me that these guys managed to create an architecture that could sustain 205 frames per second for 64 x 64 frames," Professor Arvind said. Initial ASIC synthesis in 130-nanometer process technology shows that this design would consume 42 microjoules/frame, compared with 1,960 µJ/frame running in a software version on an embedded PC.

Project 2: Rateless Wireless Networking with Spinal Codes; Edison Achelengwa, Minjie Chen and Mikhail Volkov, mentored by Kermin Elliott Fleming and Alfred Man Cheuk Ng. The aim was to provide an implementation for a novel rateless wireless networking scheme called Cortex. Arvind notes that this protocol was developed quite recently at MIT CSAIL by Professor Hari Balakrishnan and this is its first implementation in hardware. The paper provides analysis to show that implementing this design as an ASIC should achieve the desired data rates.

Project 3: Data Movement Control for the PowerPC® Architecture; Silas Boyd-Wickizer, mentored by Asif Khan. The goal was to explore whether extending an ISA with three instructions to move data between caches could help software make better use of distributed caches on multicore processors. The student modified an existing FPGA implementation of a multicore PowerPC done in BSV. This entailed many changes including in the cache-coherence protocols, and Boyd-Wickizer was able to run several benchmarks to show the advantage of his scheme.

Project 4: Viterbi Decoder; Omid Salehi-Abari, Arthur Chang and Sung Sik Woo, mentored by Myron King. Using a convolutional encoder at the transmitter associated with the Viterbi decoder at the receiver has been a predominant forward-error-correction (FEC) technique to increase the reliability of digital communication. However, a Viterbi decoder consumes large resources due to its complexity and ever-increasing data rates. The goal of this project was to boost the throughput of the Viterbi decoder by means of a novel parallel and pipelined architecture. The group has produced a Viterbi module that can be used by others and sustains 150 Mbits/second at 150 MHz on an FPGA. That's 400x faster than a MATLAB® implementation on a PC.

Project 5: H.265 Motion Estimation; Mehul Tikekar and Mahmut E. Sinangil, mentored by Alfred Man Cheuk Ng. Motion estimation is an essential component of any digital video encoding scheme. H.265, the next-generation standard in development to follow H.264, allows variable-size coding units to increase coding efficiency. The project goal was to implement a scheme that can sustain at least 30 frames per second (fps) for 1,280 x 720-frame resolution. The project produced a design that sustains 10 fps at 50 MHz on FPGA and 40 fps at 200 MHz when synthesized with a 65-nm cell library. The design is going to be submitted for fabrication in the next few months.

– Clive (Max) Maxfield

ferent assumptions to the early climbers and use a completely different approach. They carry no packs or shirts or food or water. All they take between them—in addition to minimalistic homemade climbing harnesses—is a single 200-foot length of 9mm rope, a few carabiners and a handful of spring-loaded camming devices.

If you start with the idea that you're going to have to camp out to climb the mountain, you are going to have to carry a lot more gear, which will slow you down and take longer. But what happens if you change your initial assumptions? If you plan to climb the mountain in less than a day you can cut down on the amount of gear you have to carry. If you plan on climbing it in a couple of hours you can also dispense with food and water.

In much the same way, chip design teams typically start with their own set of assumptions. They assume that learning a new approach comes at a cost. They assume that incremental change is all that's possible. They assume they have to painstakingly plan out the microarchitecture with (overly) detailed specifications. And they assume that they have only one shot at the architecture.

The experience of MIT's 6.375 Digital Design course is turning these assumptions on their head. With the right approach—using modern design tools and development platforms—it is possible for the students (and real-world designers) to quickly express and evaluate alternative architectures so as to come up with optimal implementations.

About the Author

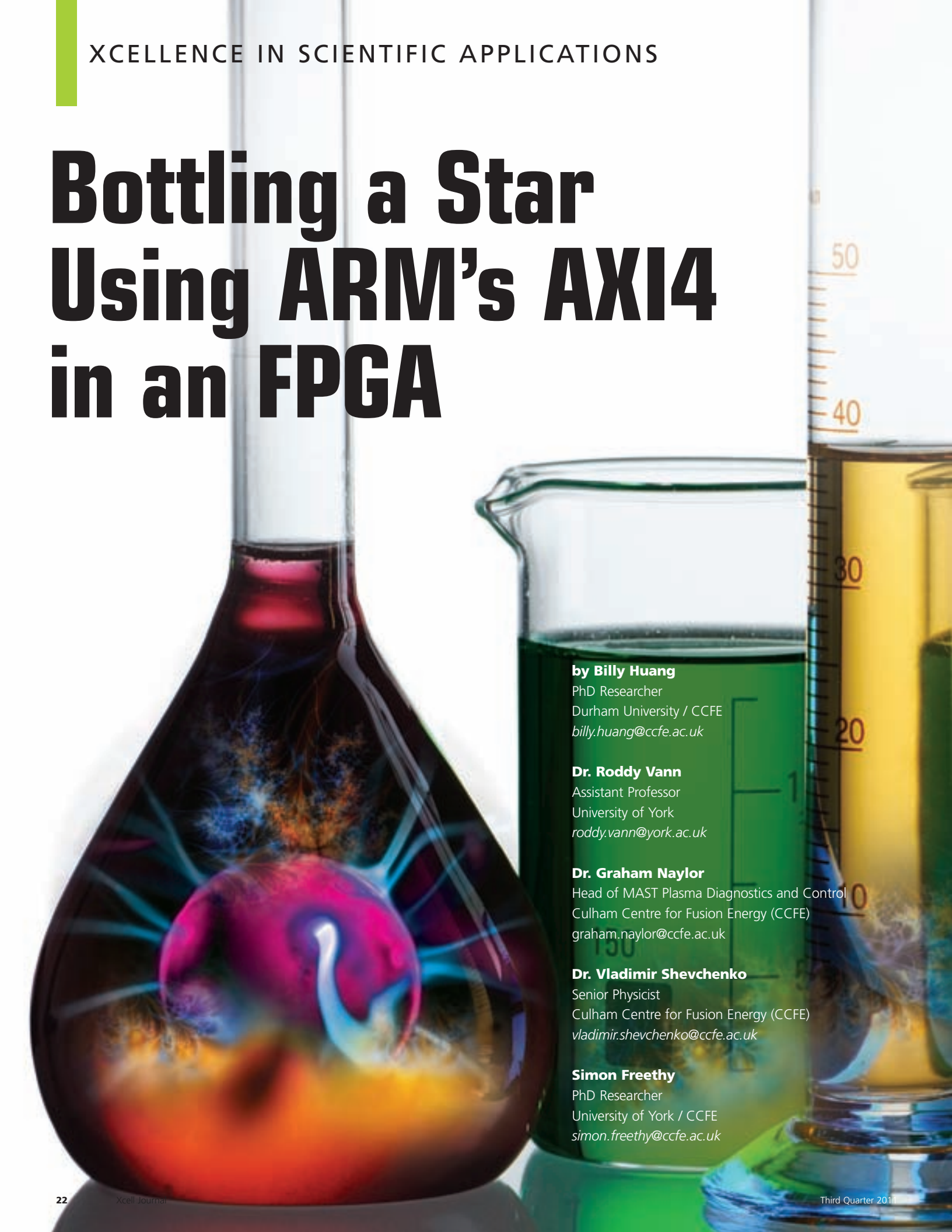
Clive "Max" Maxfield is president of Maxfield High-Tech Consulting and editor of the EE Times *Programmable Logic DesignLine.*

After receiving his BSc in control engineering in 1980 from Sheffield Hallam University, Sheffield, England, Max began his career as a designer of central processing units for mainframe computers.

Over the years, he has designed and built all sorts of interesting "stuff," from silicon chips to circuit boards and brainwave amplifiers to Steampunk "Display-O-Meters." Max has also been at the forefront of electronic design automation (EDA) for more than 20 years.

Max is the author and co-author of a number of books, including Bebop to the Boolean Boogie (An Unconventional Guide to Electronics), FPGAs: Instant Access *and* How Computers Do Math.

Bottling a Star Using ARM's AXI4 in an FPGA



by Billy Huang
PhD Researcher
Durham University / CCFE
billy.huang@ccfe.ac.uk

Dr. Roddy Vann
Assistant Professor
University of York
roddy.vann@york.ac.uk

Dr. Graham Naylor
Head of MAST Plasma Diagnostics and Control
Culham Centre for Fusion Energy (CCFE)
graham.naylor@ccfe.ac.uk

Dr. Vladimir Shevchenko
Senior Physicist
Culham Centre for Fusion Energy (CCFE)
vladimir.shevchenko@ccfe.ac.uk

Simon Freethy
PhD Researcher
University of York / CCFE
simon.freethy@ccfe.ac.uk

Fusion researchers in the U.K. demonstrate a data acquisition system for synthetic-aperture imaging using the latest ARM AXI4 interface on Xilinx technology.

Fusion energy is the combining of hydrogen atoms into larger atoms at extremely high temperatures. It is how all the stars, including the sun, create energy. To generate fusion energy on Earth, we heat ionized hydrogen gas (known as "plasma") to over 100 million degrees kelvin in a magnetic bottle called a "tokamak" (see Figure 1).

The end goal of fusion scientists like our team at the Culham Centre for Fusion Energy (CCFE)—a world-leading institution for the development of fusion energy near Oxford, England—is to create a fusion energy power station using hydrogen fuel that is readily available on Earth. In fact, there is enough fuel on Earth for fusion to supply our energy needs for more than a million years. The catch is that fusion is extremely difficult, just what you would expect when trying to bottle a star. The international ITER initiative, at $20 billion the world's largest terrestrial scientific project, will demonstrate fusion power on an industrial scale for the first time. Currently under construction in the south of France, ITER—the name means "the way" in Latin—expects to be in operation for two decades (see *http://www.iter.org/*).

A key part of fusion research is the real-time measurement of the fusion plasma. Each diagnostic has its own requirements. At CCFE (*http://www.ccfe.ac.uk/*), we have developed a diagnostic that images microwaves emitted from the plasma in order to measure the electrical current within it. For that purpose, we set out to design a synthetic-aperture imaging system.

ASSESSING MICROWAVE PHASES

Synthetic-aperture imaging uses phased arrays of antennas (see Figure 2) in a configuration that works similarly to your ears. If you hear a noise to your right, the sound will reach your right ear sooner than your left. Another way of saying this is that the sound reaches your ears at a different phase. Your brain interprets the phase difference as a direction. In much the same way, by looking at the phase of microwaves an antenna array has detected, you can determine where they were emitted from. We recombine a picture of the plasma edge from a phased antenna array that uses this principle.

The radio frequency (RF) system (see Figure 3) downconverts the signal at each antenna in frequency from 6 to 40 GHz, to the 250-MHz bandwidth signal that the FPGA data acquisition box will process. This 250-MHz bandwidth defines the clock requirement for the analog-to-digital converters (ADCs). We use eight antennas, giving 16 channels that need to be digitized (the factor of two resulting from resolving real and "imaginary" components—mathematically, those for which the signal is phase-shifted by 90 degrees).

The system had to acquire data continuously for 0.5 seconds from 16 analog channels at 14 bits at 250 MHz. Bit packing the 14 bits to 2 bytes gives us a requirement of 32 bytes * 0.25 Gbytes/s = 8 Gbytes/s. We needed to acquire 4 Gbytes of data in half a second, and wanted FPGA boards with the FPGA Mezzanine Card (FMC) interface for flexibility in the choice of ADC manufacturers and portability in the future. We also wanted the option of using our in-house-developed FMC digital I/O board.

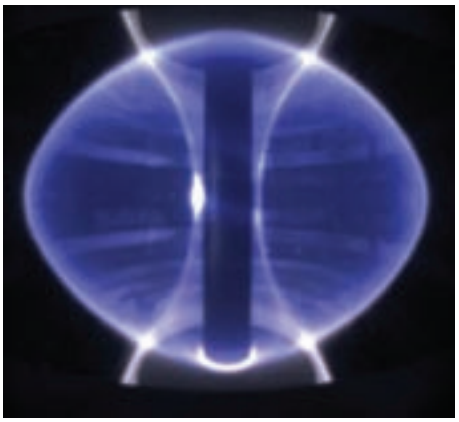


Figure 1 – The Mega Amp Spherical Tokamak (MAST) at CCFE is uniquely shaped, more similar to a cored apple than the traditional doughnut. The clear view inside the device shows the "bottled star" within.

We decided during the summer of 2010 that an ideal solution would use two Xilinx® Virtex®-6 LX240T ML605 boards combined with two FMC108 (eight-channel) ADC boards from 4DSP. At that time, 8 Gbytes/s was a gigantic data rate; in fact, it still is. We could have taken the approach of divide and conquer by using more FPGA boards and having fewer channels on each. However, that would have increased the cost and size of the system.

In fact, the technology to make this aspect of our design happen arrived around January 2011, when Xilinx released a revision of its ISE® design software supporting ARM's AMBA® AXI4 interface protocol. Before this the hardware existed, but not the means to exploit it to its full potential.

Figure 2 – The microwave-imaging phased antenna array utilizes novel PCB antennas.

LIFE BEFORE AXI4

For our system needs, the DDR3 SDRAM memory must be accessible to the MicroBlaze™ processor that resides on the Virtex-6, so that Linux can also have access to the real-time data we are capturing. This requirement constrains us to using a memory controller jointly accessible to the MicroBlaze bus and our real-time streaming IP. Initially we tried using the PLB bus, but found that limitations in the PLB-based memory controller meant we could not connect a 64-bit-wide interface at our required frequency. Instead, only 32 bits were available. We learned this the hard way, after writing a core that communicated via the low-level NPI protocol directly to the memory controller, but could achieve only 2 Gbytes/s. Even though this was still an impressive rate and smashed any speed records we could find, it still was not enough.

Thankfully, Xilinx then pushed out the AXI4 interconnect and memory controller, giving full access to the whole 64 bits at 400 MHz double data rate (800 million transactions per second). That effectively gave a throughput of 6.4 Gbytes/s—a truly blistering speed that exceeded our requirement of 4 Gbytes/s on each board. This was exactly what we needed.

We actually found two ways to achieve this speed: one is a modification of the axi_v6_ddrx memory controller (hidden under the AXI interconnect layer), and the other is an AXI Master PCore made in System Generator. The PCore can attach to the MicroBlaze system in Xilinx Platform Studio (XPS) as an AXI External Master.

Both solutions stream data into the DDR3 memory at 5 Gbytes/s. AXI is easy to program, and allows very high memory speeds with separate read and write channels. The XPS tool gives a lot of flexibility for AXI design. We used that flexibility to our advantage, such as choosing only a write channel if that was what we needed, thereby simplifying the logic design and freeing more resources.

A SOFT-PROCESSOR INTERFACE

A unique capability of the Xilinx tool set is the soft processor known as the MicroBlaze. It is "soft" in that it is enabled using FPGA logic. This processor is well-supported in the main branch of the Linux kernel thanks to the efforts of Xilinx and its partners. We are encouraged by these efforts and are working to extend the development in the Linux community.

This capability has meant that we can have a PC-like interface to the FPGA system. This is invaluable as it enables, for example, Web and SSH servers on the FPGA. We are able to mount the System ACE™ flash under Linux (when formatted as type msdos), which allows us to update the firmware remotely.

NETWORK STREAMING

Given that we could acquire 2 Gbytes on each FPGA board in half a second, the question we found ourselves facing was how to get this data off the board over a standard interface in a reasonable amount of time? Typical network speeds using the MicroBlaze processor over Gigabit Ethernet under Linux and a simple protocol such as UDP proved too slow, achieving only around 0.5 Mbyte/s. At that rate we would have to wait over an hour to download data that had taken only half a second to acquire!

Clearly, we needed to go to a lower level in the design. Our solution took the form of a homegrown protocol we have dubbed FireStark, a UDP-based protocol that sits inside the AXI Ethernet DMA driver. By modifying the MicroBlaze Linux kernel drivers and having the FPGAs on a dedicated private network, we are now able to download the entire 2 Gbytes in under 60 seconds, a factor-of-70 speed-up. Testing with jumbo frames of sizes up to 6 kbytes has doubled this speed—that is,

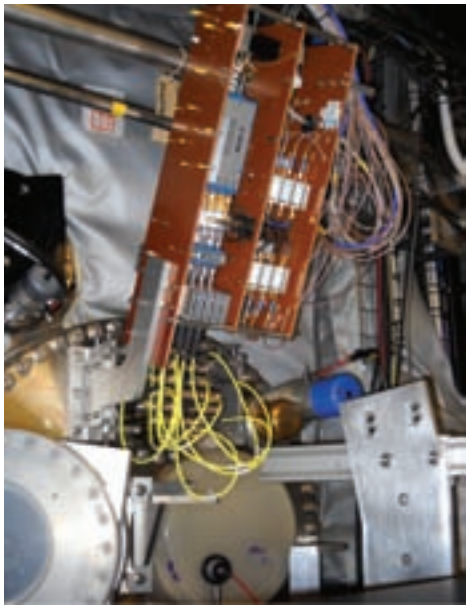


Figure 3 – The RF electronics connected to the MAST tokamak downconvert the incoming 6- to 40-GHz signal to the 250-MHz bandwidth signal the FPGA data acquisition box will process.

more than 70 Mbytes/s. Crucially, it shows that with DMA even the relatively slow MicroBlaze clock of 100 MHz is capable of high memory-to-network streaming throughput.

The latency measurement from the FPGA to the PC was 129 µs +/-13 µs. (The real latency is even lower, since this measurement includes the latency overhead of the packet traversing a switch, through the PC kernel, up the network stack and into user space.) We also plan to measure the FPGA-to-FPGA latency, which we expect will be lower.

CLOCK SYNCHRONIZATION

Our tokamak has numerous diagnostics and systems, all of which need to be synchronized to a 10-MHz global experimental clock. We derive our 250-MHz acquisition clock from this signal; this derived signal clocks the ADC boards. The onboard crystal clock drives the remaining FPGA logic.

Our system is unusual in that it does not send the experimental clock continuously, but only at trigger events for about 10 seconds. Outside these periods we need to switch back to an internally generated clock. Thus, we have essentially two clocks that we need to switch between, an external clock and an internal clock.

The key requirement for both FPGA boards is that they must be precisely synchronous. Ideally, since we are sampling at 4 ns, we can expect a readable input sine wave on our ADC at our highest expected frequency to have a period of 8 ns, equivalent to 360 degrees. If we would like 5-degree phase accuracy we need a maximum skew requirement of 8 * (5/360) = 111 ps. This degree of accuracy is very challenging. Even light travels only 3.3 cm in this amount of time.

We have designed the firmware such that it is identical on both boards. We use a DIP switch to enable or disable different functions required of each board. This dramatically reduces the development time, as we only need to synthesize the firmware once.

The clock, which is generated on one of the boards, travels out over two closely placed SMA ports and then feeds back in (using cables of equal length) to the ADC board that is connected to each FPGA board's FMC port. This is to ensure that each board is running on precisely the same clock, with the only phase difference being that equal to the difference between the two SMA ports on leaving the FPGA board. Figure 4 more clearly illustrates this arrangement.

In a similar way to how the external 10 MHz arrives and gets sent out, coming back in on both ADCs, the external triggering uses the same method to ensure that both boards are triggered synchronously.

BENEFITING FROM UNIQUE FEATURES

The Xilinx FPGA architecture offers a number of novel features that we have put to good use in our design. For example, we use the IODELAY primitive to fine-tune path delays at the pins. This allows us to compensate for differences in track length. It was vital to have this capability, since the data path lengths on the ADC attached to the FMC are not equal. Unless we compensated for the path delays, the data from the ADC would have been garbage. The data was coming off the ADC at double data rate with a 250-MHz clock, so the time between each valid piece of data was merely 2 ns. IODELAY allowed us to align the data paths very precisely, in steps of 125 ps.

Equally important are the Mixed Mode Clock Managers (MMCMs),

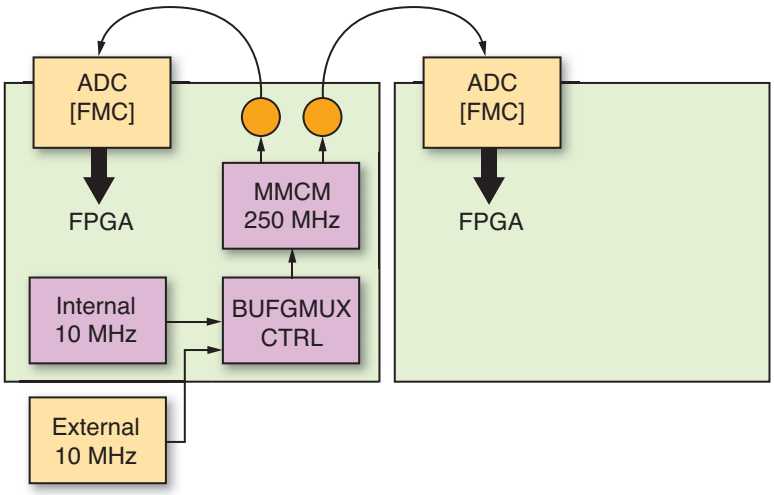


Figure 4 – The two FPGA boards must be precisely synchronous. The clocking scheme shown here ensures that they are.

Figure 5 – The FPGA data acquisition box comprises Xilinx's ML605 evaluation board, 4DSP's FMC108 ADC board and our in-house FMC/PMOD header board. We wired the ADC SSMC connectors internally to front-panel SMA bulkheads to extend the life of the ADC analog connections.

which perform clock management tasks such as multiplication and phase shifting. In cascaded mode whereby one MMCM connects to another, we were able to generate a wide range of clocks from the original 10 MHz. This includes the 250-MHz ADC sampling clock, as well as additional clocks that we used for other purposes.

We likewise made good use of the BUFGMUX_CTRL and IDDR primitives. Since our system switched between internal and external 10-MHz clocks, it is crucial that switching between the two be glitch-free. The BUFGMUX_CTRL primitive allowed us to make sure it was. You can also use this primitive for standard logic such as triggers (not only for clocking); however, you need to ensure that the attributes IGNORE0, IGNORE1 are set to 1 to bypass the deglitching circuitry, which would otherwise not allow the logic to pass through.

The ADC, meanwhile, provides data in a DDR format; that is, the data is valid on both the rising and falling clock edges. To recover this data into single data rate (SDR), we use the IDDR primitive, which is hardwired on the I/O pads. This has a single data pin input, and two data pin outputs. We used the SAME_EDGE_PIPELINED attribute, which ensured the data was valid on both pins at the same time, thus reducing other logic. This does come at the cost of an additional cycle of latency, but for us the latency did not matter.

Another feature of the Xilinx architecture that helped in our design was the FPGA Mezzanine Connector (FMC). Strictly speaking this is not a unique feature of an FPGA, but of an FPGA board. Even so, it has proven very useful and has worked well with the Virtex-6. FMC connectors include high-frequency clock pins, which are wired to clock-capable pins on the Virtex-6 on the ML605 board. As such, it is possible to send a clock via the FMC and into the FPGA. This is advantageous since it means that we need only one entry point for the clock.

USING THE XILINX TOOL SUITE

Xilinx provides a number of tools to aid in the development of an FPGA system. We used a good number of them.

We used Project Navigator for manually coding VHDL and Verilog code. Additionally there is a graphical interface whereby you can make a "schematic" that allows the creation of logic visually. However, we found Project Navigator to be a low-level tool in that while we could operate easily on flip-flops (single bits), expanding to operations on larger bit numbers was more complicated. We found Project Navigator most useful for low-level clock design. It enabled us to have precise control over which clock was driving specific logic.

For high-level logic design, we turned to System Generator. It is particularly suited to designs where logic is driven by a single clock frequency (although isn't restricted to this case). System Generator is simple to use and has access to large range of IP, such as FFTs, divide generators and filters, to name a few. Additionally, you can tie logic easily into the MicroBlaze processor as read/write registers and shared memory. The tool automatically creates a peripheral core (PCore) and adds it into your XPS project.

We used CORE Generator™ for fine-tuning the parameters of the ADC FIFO. This FIFO had to be 256 bits wide with a write clock of 125 MHz and a read clock of 200 MHz. We imported the resulting generated NGC file into XPS as a PCore. We did this manually by creating the necessary .mpd, .pao and .bbd files.

The Impact tool helped us to program the FPGA, and also to generate the SystemACE™ file for permanently placing the firmware onto the CompactFlash. The CompactFlash worked very reliably, however it should be noted that this added an extra requirement (see under SDK, below) to our system.

Since we wanted our system to include the MicroBlaze processor, we needed the tool that creates the processor system: Xilinx Platform Studio. XPS is a comprehensive tool suite that allows you to build processor-centric systems. It allows you to set up the required links through a wizard. You can also include IP from CORE Generator by using the Create IP Wizard. It now includes the high-performance AXI4 on-chip interconnect.

Finally, we used the Xilinx Software Development Kit (SDK) to develop programs that run on the processor. In fact, we have only one program to run initially, and that is the SREC bootloader. Due to the CompactFlash having a FAT file system, the libraries required to access the SREC program (also on the flash) inflated the size of the resulting executable. We reduced its size by turning off debugging, turning on optimization and including "mb-strip –g <elf_file_name>" as the post-compilation command. Even after all these steps, the result was a large, 91-kbyte executable. Therefore, we had to increase the internal BRAM so that we could initialize the bitstream with this size of executable.

One problem we faced was the large compilation time with the Virtex-6. The Xilinx software PlanAhead™ can significantly help with this challenge. We intend to utilize PlanAhead to its full potential to reduce the compilation time.

We are excited by the possibilities that the new Zynq™-7000 extensible processing platform will provide (see cover story, *Xcell Journal* Issue 75). However, it remains to be seen whether Zynq will make the MicroBlaze obsolete, or if the MicroBlaze will hold its own thanks to its soft nature and the 10-plus years of development effort behind it. Could a future cache-coherent multiprocessor MicroBlaze system outperform the ARM® dual-core Cortex™-A9 MPCore™? Could the Physical Address Extension in the Zynq or MicroBlaze lead to more powerful systems that provide more than 32 bits of address space, thus allowing more than 4 Gbytes of RAM? It will be interesting to watch and see how time answers those questions.

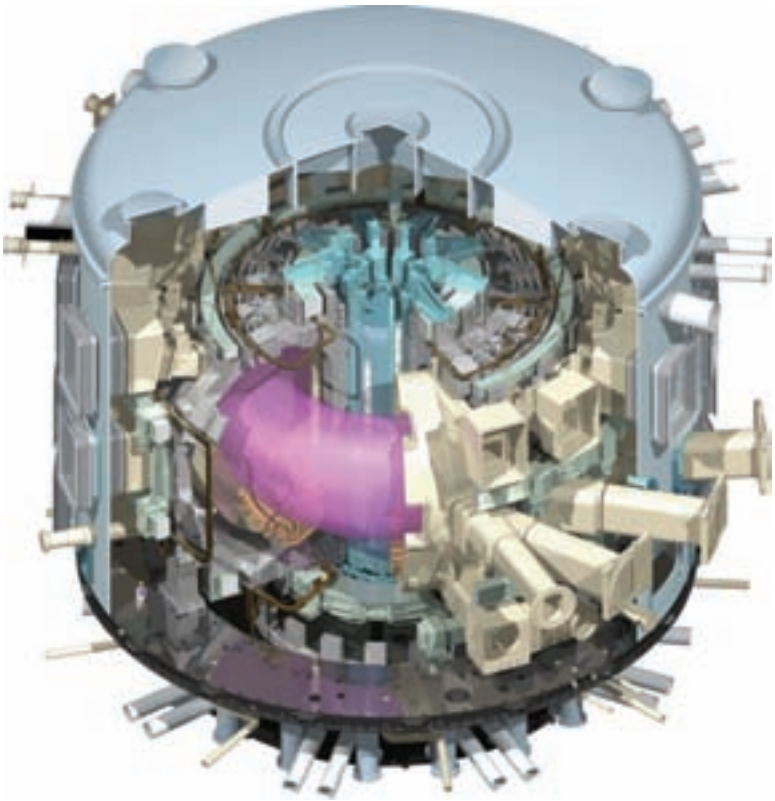


Figure 6 – The ITER tokamak, currently being built in the south of France, will produce 500 megawatts of fusion energy, paving the way for a demonstration fusion power plant.

A CUTTING-EDGE SYSTEM

Ultimately, we developed a fully functional data acquisition system (see Figure 5) that is cutting edge in the world of FPGAs, making use of the latest Xilinx technology. It is capable of real-time acquisition at 10 Gbytes/s (or 80 Gbits/s). The end cost was less than $15,000. We have demonstrated technology that we hope will find its way onto the largest fusion experiments in the world, such as the ITER project (Figure 6).

Fusion energy is one of the biggest technological challenges ever attempted. FPGAs are helping us to crack this tough nut in different ways by leveraging their unique advantages. Our fusion research device, which incorporates Virtex-6 FPGAs using the latest AXI4 interconnect and the Xilinx tool flow, achieves extremely high data rates on a small, compact system.

A new website (*http://fusion.phys.tue.nl/fpga/doku.php*) promises to be a meeting place for people to communicate ideas and material for developing FPGA technology on fusion devices.

Acknowledgements

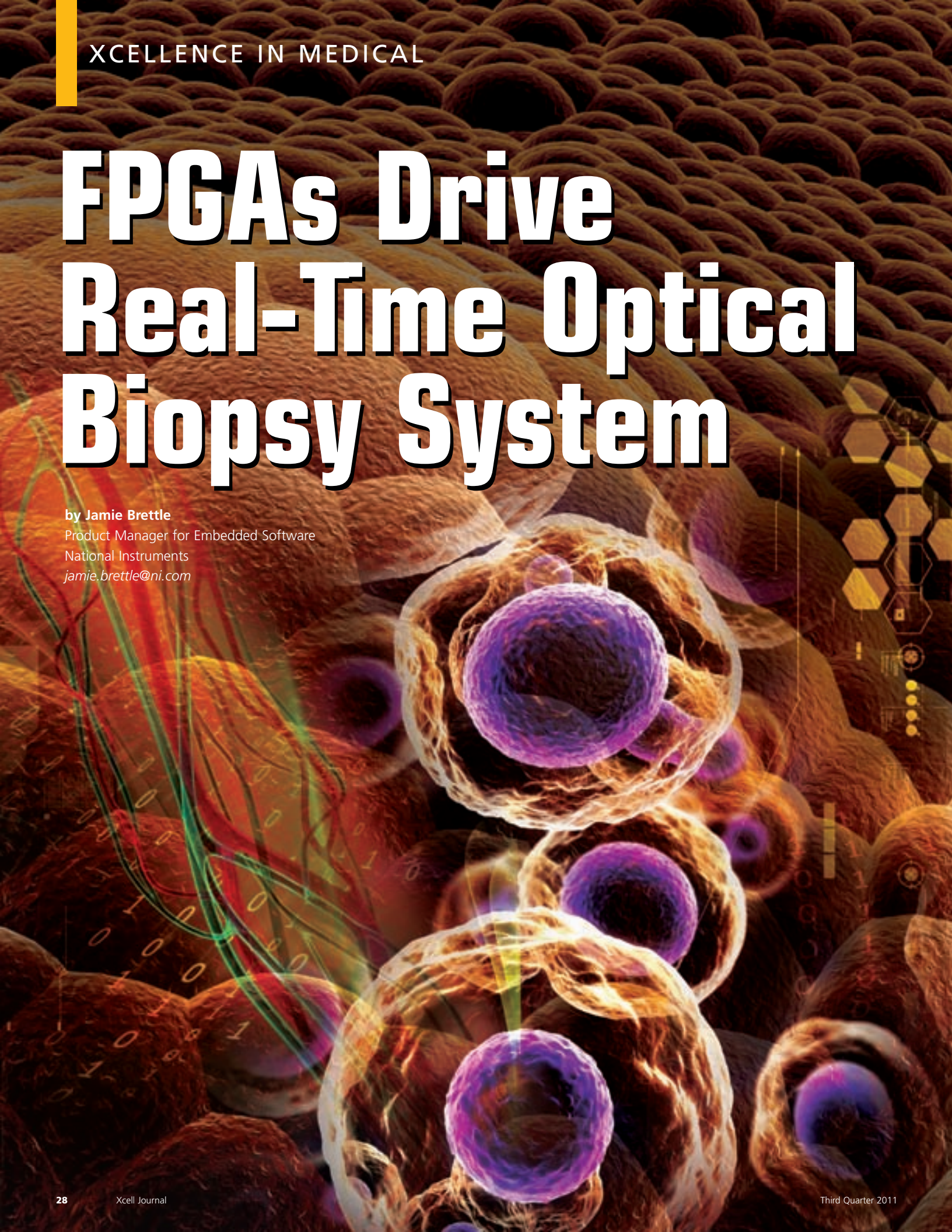
CCFE is associated with Durham University's Centre for Advanced Instrumentation and the University of York's Plasma Institute.

This work was funded partly by EPSRC under grant EP/H016732, by the University of York, by the RCUK Energy Programme under grant EP/I501045 and by the European Communities under a contract of association between EURATOM and CCFE.

The authors wish to thank John Linn, open-source Linux engineer at Xilinx, and the other Xilinx employees and Xilinx partners who have contributed to Linux support for the MicroBlaze processor.

FPGAs Drive Real-Time Optical Biopsy System

by Jamie Brettle
Product Manager for Embedded Software
National Instruments
jamie.brettle@ni.com



Researchers in Japan are using the Virtex-5 and National Instruments LabVIEW to develop next-generation 3D OCT imaging.

Thanks to the availability of ever-increasing processing power, scientists in the field of medical-device research are rapidly finding innovative ways to more effectively treat patients suffering from a variety of ailments. Commercial off-the-shelf (COTS) hardware coupled with FPGA technologies and flexible integration platforms help these researchers to develop prototype imaging systems more quickly, while continuing to deliver new products to the market.

Among them are researchers at Kitasato University in Japan, who recently set out to create an instrument that can detect cancer during medical screenings using a methodology that would not require the patient to undergo the stress of a biopsy. The data acquisition system they developed utilizes optical coherence tomography (OCT) to create real-time 3D imaging.

OCT is a noninvasive imaging technique that provides subsurface, cross-sectional images. The medical community has a keen interest in OCT because it provides much greater resolution than other imaging technologies such as magnetic resonance imaging (MRI) or positron emission tomography (PET). Additionally, the method does not require much preparation and is extremely safe for the patient, because it uses low laser outputs and does not require ionizing radiation.

STATEMENT OF THE PROBLEM

OCT uses a low-power light source and the corresponding light reflections to create images—a method similar to ultrasound, but it measures light instead of sound. When the light beam is projected into a sample, much of the light scatters, but a small amount reflects as a collimated beam, which the system can detect and use to create an image.

This methodology presents a promising diagnostic tool for optical biopsies, useful in many medical examinations and of particular interest in oncology. In most OCT applications, high-speed imaging is crucial for rapid inspection and for image quality without motion artifacts. For example, to inspect the human eye, which can be held relatively motionless using a chin rest, an A-scan ultrasound is necessary to eliminate all motion artifacts. However, when inspecting the digestive or respiratory systems, the tissue being imaged cannot be fixed in place, making an ultrahigh-speed OCT methodology necessary to eliminate motion artifacts of the tissue.

When performing a noninvasive, real-time optical biopsy, the imaging speed must be fast enough to display the 3D image for immediate diagnosis, just like a conventional endoscope. A few previous methods have been proposed for ultrahigh-speed OCT, but none have succeeded in the real-time display of 3D OCT movies.

INITIAL PROTOTYPE

Initially, the Kitasato University team developed a first-generation system around 32 National Instruments (NI) high-density eight-channel digitizers. The key element of the OCT method was using optical demultiplexers to separate 256 narrow spectral bands from a broadband incident-light source. This allowed simultaneous, parallel detection of an interference fringe signal (a critical requirement for OCT imaging) at all wavelengths in the measured spectrum. The system captured all 256 spectral bands simultaneously at 60 Msamples/second, using the digitizer's onboard memory, and then transferred the data to a PC for processing and visualization.

While it was possible to capture volumetric OCT videos with the system, the massive amount of data acquired by all channels simultaneously made the onboard memory of the digitizers the limiting factor. Overall, the designers restricted the duration of an OCT video to about 2.5 seconds. After transferring the data to the PC, they still required about three hours to fully process and render the 3D video data.

Performing a real-time optical biopsy (a primary goal for endoscopic OCT) was not possible with this system. The team vowed to create a new prototype with an upgraded data acquisition subsystem and enough performance to make real-time processing possible.

HIGH-LEVEL ARCHITECTURE OF THE SYSTEM DESCRIPTION

The new system uses a broadband superluminescent diode as the light source, combined with a filter that selects the wavelength range to match the optical demultiplexers. A semiconductor optical amplifier amplifies the output light from the diode, while a collimator lens and an objective lens direct the light. A resonant scanner and a galvano mirror scan the light beam. The system collects back-scat-

The use of Xilinx Virtex-5 FPGAs made it possible to implement processing algorithms in hardware, significantly increasing processing performance by moving portions of the code from the PC to the FPGA.

tered or back-reflected light from the target of the light beam using light-illuminating optics and directs it to another optical amplifier with an optical circulator. Figure 1 illustrates the block diagram of the system.

Optical demultiplexers divide the light into 320 wavelengths and direct it to differential photoreceivers. The data outputs of the photoreceiver system then go to the data acquisition system.

out an understanding of VHDL coding. NI FlexRIO combines interchangeable, customizable, high-speed analog and digital I/O adapter modules with a user-programmable FPGA. The use of Xilinx® Virtex®-5 FPGAs made it possible to implement processing algorithms in hardware, significantly increasing processing performance by moving portions of the code from the PC to the FPGA.

Module, they can translate graphical code to HDL and then target it toward Xilinx FPGAs, which makes it possible for developers to quickly prototype algorithms and generate and test FPGA code.

Figure 2 shows a diagram of the data acquisition system. For high-speed acquisition, the Kitasato University team used an NI adapter module with a 50-Msample/s sample rate on 16 simul-

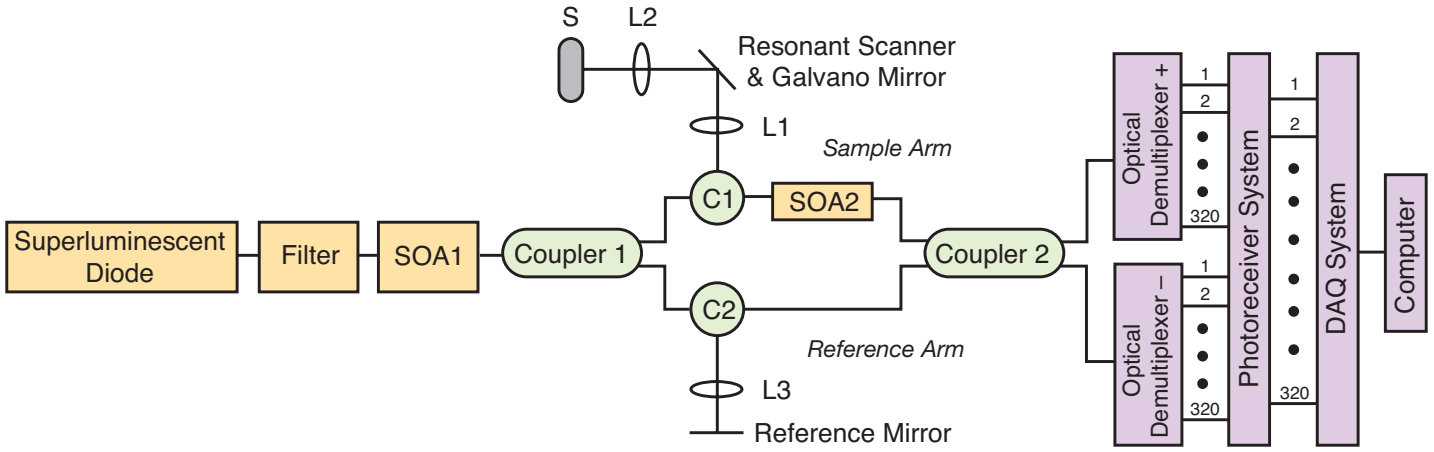


Figure 1 – Experimental setup of the ultrahigh-speed OCT system (SOA stands for semiconductor optical amplifier; C1 and C2 are the collimator lenses)

DATA ACQUISITION AND REAL-TIME PROCESSING

The 320-channel data acquisition system captures data from the photoreceiver using the NI FlexRIO modular FPGA hardware, which is programmed with the NI LabVIEW FPGA Module. This module is a graphical design language that designers can use to craft the FPGA circuitry with-

The LabVIEW graphical programming environment provides high-level abstraction of hardware and software algorithms by means of an interface for programming using intuitive graphical icons and wires that resemble a flowchart. Designers can use LabVIEW to target a number of machine architectures and operating systems. Using the LabVIEW FPGA

taneous channels at 14-bit resolution. The adapter module interfaces to an NI FlexRIO FPGA module, featuring a Xilinx Virtex-5 SX50T, to perform the first stage of processing: subtraction of the sample-cut noise and multiplication of a window function. In total, the group used 20 modules spread across two PXI Express chassis, requiring timing and synchronization modules to dis-

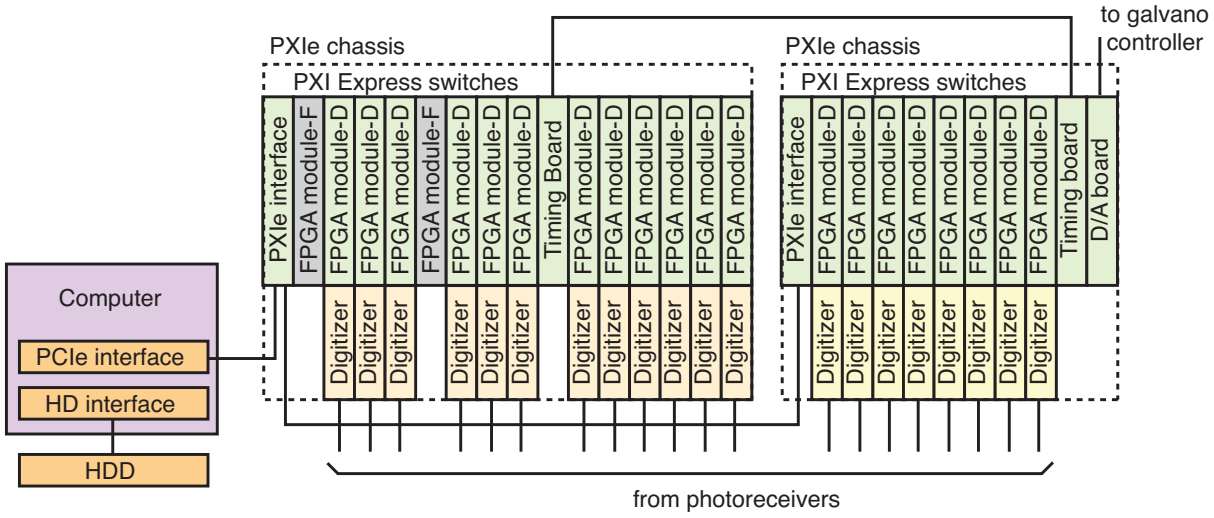


Figure 2 – The 320-channel data acquisition and processing system

tribute clocks for the system and assure precise phase synchronization across all the channels in the system.

Building the medical imaging system around the NI PXI platform was critical in achieving the necessary performance. As a result, the application required the high data throughput of PCI Express, the standard upon which PXI builds with additions for instrumentation. Using the x4 PCI Express interface of the PXI Express modules delivered sustained data throughput of more than 700 Mbytes/s.

In addition, the team also used the PCI Express architecture to achieve point-to-point communication between different instruments connected to the bus over direct DMA, without the need to send data through the host processor or memory. Specifically, the system streams data between NI FlexRIO FPGA modules without going through the host PC, which was traditionally a requirement for achieving real-time imaging. While this is normally a complex development task, the Kitasato team used the NI peer-to-peer (P2P) streaming API to connect multiple FPGAs in the system, eliminating concerns around the low-level implementation. That freed them to focus their expertise on the FPGA algorithms that determine the imaging performance of the system.

Using P2P streaming, the designers streamed the preprocessed data of the first stage to two additional NI FPGA modules built around the high-performance Virtex-5 SX95 FPGA. These FPGAs are extremely efficient at signal processing thanks to a large number of onboard digital signal-processing (DSP) slices. The team used these modules to perform the required fast Fourier transform (FFT) processing. To achieve 3D imaging capabilities, the two FPGAs in the system computed more than 700,000 512-point FFTs every second. While the designers used LabVIEW FPGA IP for most of the algorithm development, they also integrated Xilinx CORE Generator™ VHDL IP into their LabVIEW FPGA application to achieve the complex FFT processing performance required.

Using LabVIEW to integrate and control the different parts of the system, they transferred data over a high-speed MXI-Express fiber-optic interface from the PXI system to a quad-core Dell Precision T7500 PC equipped with an Nvidia Quadro FX 3800 graphics processing unit (GPU) to perform real-time 3D rendering and display. Because their goal was to achieve a 12 volume/s rate, they required an overall data rate back to the PC of slightly more than 500 Mbytes/s. (Volume represents the depth of the surface being scanned.)

Figure 3 – The data acquisition system interfaces to the Dell PC for real-time OCT image scanning.

The system includes three real-time display modes: continuous display of rendered 3D images; continuous 2D cross-sectional frame scanning in a 3D cube along each of the axes; and continuous display of all acquired B-scan images.

While the architecture does not limit the image acquisition time, it's possible to log up to 100 minutes on the prototype system by storing data to the NI HDD-8264 RAID system, which provides 3 Tbytes of hard-drive space.

In the photograph of the system (Figure 3), the left side of the rack contains the bank of photoreceivers and the right side includes the data acquisition system. The two PXI chassis are located in the center of the rack, while signal-breakout boxes are positioned above and below.

REAL-TIME RESULTS

The system includes three real-time display modes: continuous display of rendered 3D images; continuous 2D cross-sectional frame scanning in a 3D cube along each of the axes; and continuous display of all acquired B-scan images.

Figure 4 shows an example of a continuously rendered 3D display—in this case, the skin of a human finger. The rendered images are continuously refreshed and can be changed in real time. Figure 4a shows the fingerprint pattern clearly while Figure 4b shows the sweat glands, making it possible for doctors to observe sweat glands in real time.

Although data at 12 volumes/s is continuously transferred from the data acquisition system to the PC, volumetric data arrays must be reformatted for the GPU rendering process, which causes a data-processing bottleneck. Currently, the prototype system refreshes the rendered images twice per second, but the GPU board was benchmarked to perform volume-rendering processing at four times per second. Thus, there's room to improve the refresh rate of the system by further optimizing the algorithms.

Notable for their potential in the diagnosis of disease are the images in Figure 5, derived from a real-time video displaying rendered OCT views of the three-layer structure in an

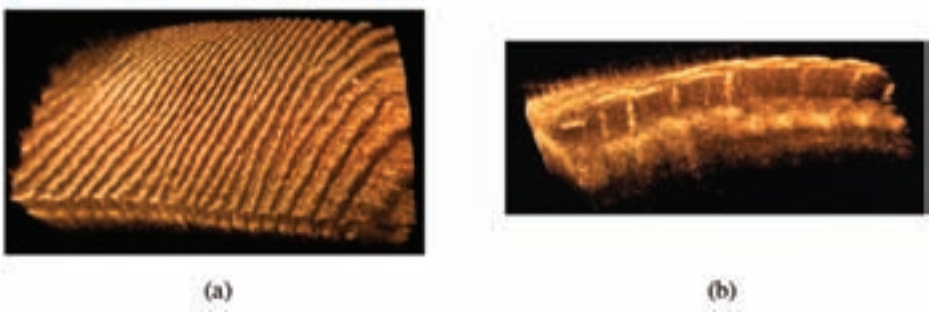


Figure 4 – Real-time rendered 3D image of the skin of a finger; (a) shows the fingerprint clearly, while (b) offers a look at the sweat glands.

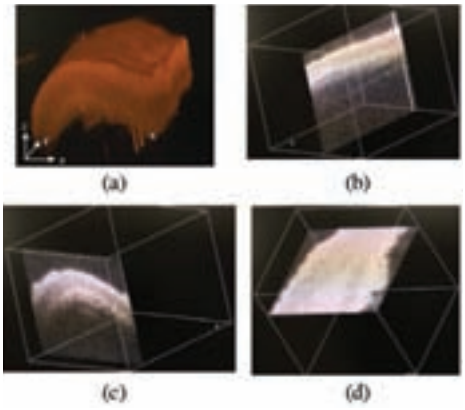


Figure 5 – Real-time OCT images of the extracted esophagus of a pig (a); the 2D cross-sectional image slices in b, c and d are scanned along the x, y and z axes, respectively.

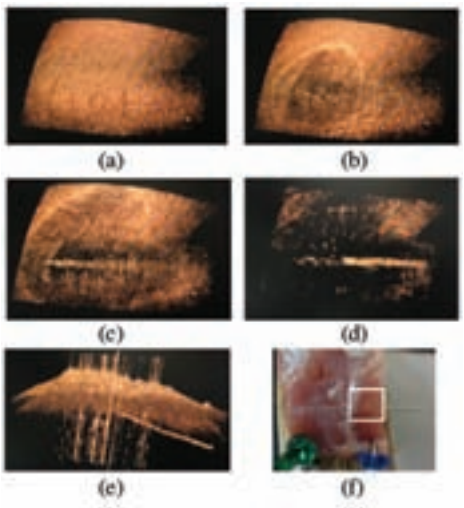


Figure 6 – Virtually cutting a sample—in this case, a piece of chicken—in real time reveals an object inside it. Here, it's a sewing needle that the researchers inserted, but the same technique could assess the depth and spread of a cancer in real time.

extracted pig esophagus (a). The real-time display of 2D cross-sectional slice scanning along an axis designated as x, y or z is also possible, as shown in Figures 5b, 5c and 5d, with a depth range of 4 mm. This image-penetration depth is sufficient to detect early-stage cancer.

While displaying 3D rendered images, we can also virtually cut part of the tissue away to reveal the inside structure in real time. Figure 6a is a rendered image of a piece of chicken meat viewed from above. Figure 6b shows a virtual cut of the thin surface layer of the tissue. As we increase the thickness of the layer to cut, as in Figures 6c and 6d, we can see a rodlike object embedded within the chicken meat—the reflected light from a steel sewing needle that the scientists inserted into the sample. The virtual cutting process can be done reversibly in real time. As the rendered image is rotated, the rod is directly visible without a virtual cut as shown in Figure 6e. Virtually cutting a rendered image is useful for estimating depth and the spread of cancer in real time.

SOLUTION ANALYSIS

Overall, the Japanese scientists leveraged the flexibility and scalability of the PXI platform and NI FlexRIO to develop the world's first real-time 3D OCT imaging system. They used LabVIEW to program, integrate and control the different parts of the system, combining high-channel-count acquisition with FPGA and GPU processing for real-time computation, rendering and display.

Using FPGA-based processing enabled by NI FlexRIO, they computed more than 700,000 512-point FFTs every second to achieve 3D imaging, while maintaining high channel density for the 320-channel system. They used high-throughput data transfers over PCI Express, accurate timing and synchronization of multiple modules and peer-to-peer data streams to transfer data directly between FPGA modules without going to the host. Additionally, they were able to maintain a high-throughput connection between the I/O and the GPU processing in the host PC and integrate RAID hardware for extended image data logging. The combination of COTS hardware, featuring an FPGA, and high-level design tools like LabVIEW gave the researchers the ability to develop their prototype imaging system more quickly than conventional methodologies would have allowed.

With this processing system, Kitasato University demonstrated continuous real-time display of 3D OCT images with the ability to rotate the rendered 3D image in any direction, also in real time. Observation of tissues, such as the trachea or esophagus, with good image penetration depth demonstrates the applicability of high-speed OCT imaging to optical biopsy. Further, revealing the inside of a structure by virtually cutting the tissue surface in real time will be very useful for cancer diagnosis in addition to observing dynamic tissue changes. Surgeons could use this system to observe blood flow and tissue changes during surgery.

The Kitasato University scientists hope to keep moving their OCT technology toward commercialization, but this is the first-generation system that boasts these capabilities. So, a lot of testing needs to be done. In addition, the team would also like to reduce the overall system cost before such a product could move into the real-world setting of hospitals.

As a result, the system is still some time out from real-world deployment. Nevertheless, it represents an important milestone in demonstrating the possibilities of this technology.

SIMULATION

Active-HDL™

FPGA Design and Simulation Made Easy

Active-HDL™ is a Windows® based, integrated FPGA Design and Simulation solution with design entry and documentation tools and a high-performance mixed-language simulator. Active-HDL features an easy-to-use, FPGA Flow manager that controls VHDL, Verilog, SystemVerilog and SystemC simulation, synthesis (XST™, Synplify™ and Precision™) and implementation (Xilinx™ Place and Route) from one common environment.

Top Features

Text & Graphical Design Entry
High Performance Mixed Language RTL Simulator

IEEE VHDL, Verilog®, SystemVerilog (Design), SystemC

Advanced Debugging & Code Coverage

Secure IP and IP Encryption based on IEEE Standards

DSP Co-simulation with MATLAB® and Simulink®

PCB Design Interface

HTML and PDF Design Documentation

Pre-Compiled Xilinx® Libraries

Design Rule Checker

Windows® 7/Vista/XP/2003
32/64 bit support

Aldec, Inc.
Corporate Office, USA
Phone: 702.990.4400
E-mail: sales@aldec.com
www.aldec.com

VERIFICATION

Riviera-PRO™

Advanced Verification Platform

Riviera-PRO™ is a complete verification platform that enables Design and Simulation of VHDL, Verilog, SystemVerilog, and SystemC designs. Riviera-PRO is optimized for use with all configurations of Xilinx™ ISE Design Suite products. Advanced debugging tools are available on all major platforms (32/64 Linux® and Windows®) and support for the latest UVM library make Riviera-PRO and ideal verification platform for designs targeting the latest Xilinx devices.

Top Features

Advanced Verification Platform (UVM/OVM and VMM)

High-Performance Simulator for Mixed Language Designs

IEEE VHDL, Verilog®, SystemVerilog (Design and Verification), SystemC/C/C++

Different Levels of Abstraction (ESL/TLM, RTL, Gate-Level)

Transaction-Level Debugging Environment

Assertion-Based Verification (SVA, PSL and OVA)

Code and Functional Coverage

DSP Co-Simulation with MATLAB® and Simulink®

Linux® and Windows® 7/Vista/XP/2003
32/64 bit support

Europe Phone: +33.6.8032.6056 Email: sales-eu@aldec.com
Israel Phone: +972.3.2257.3422 E-mail: sales-il@aldec.com
Japan Phone: +81.3.5312.1791 Email: sales-jp@aldec.com
China Phone: +86.21.6875.2030 Email: info@aldec.com.cn
India Phone: +91.80.4150.6434 Email: sales-sa@aldec.com
Taiwan Phone: +886.2.2659.9119 E-mail: sales-tw@aldec.com

ALDEC

inrevium

A Revolutionary Innovation Platform

TOKYO ELECTRON DEVICE LIMITED

Headquarter : Yokohama , Japan
E-mail:psd-sales@teldevice.co.jp
http://www.inrevium.jp/eng/x-fpga-board/index.html

US Offices : Fremont , CA
E-mail:psd-nasales@teldevice.co.jp

China Offices : Shanghai
E-mail:sales@teldevice.cn

XILINX® ALLIANCE PROGRAM PREMIER MEMBER

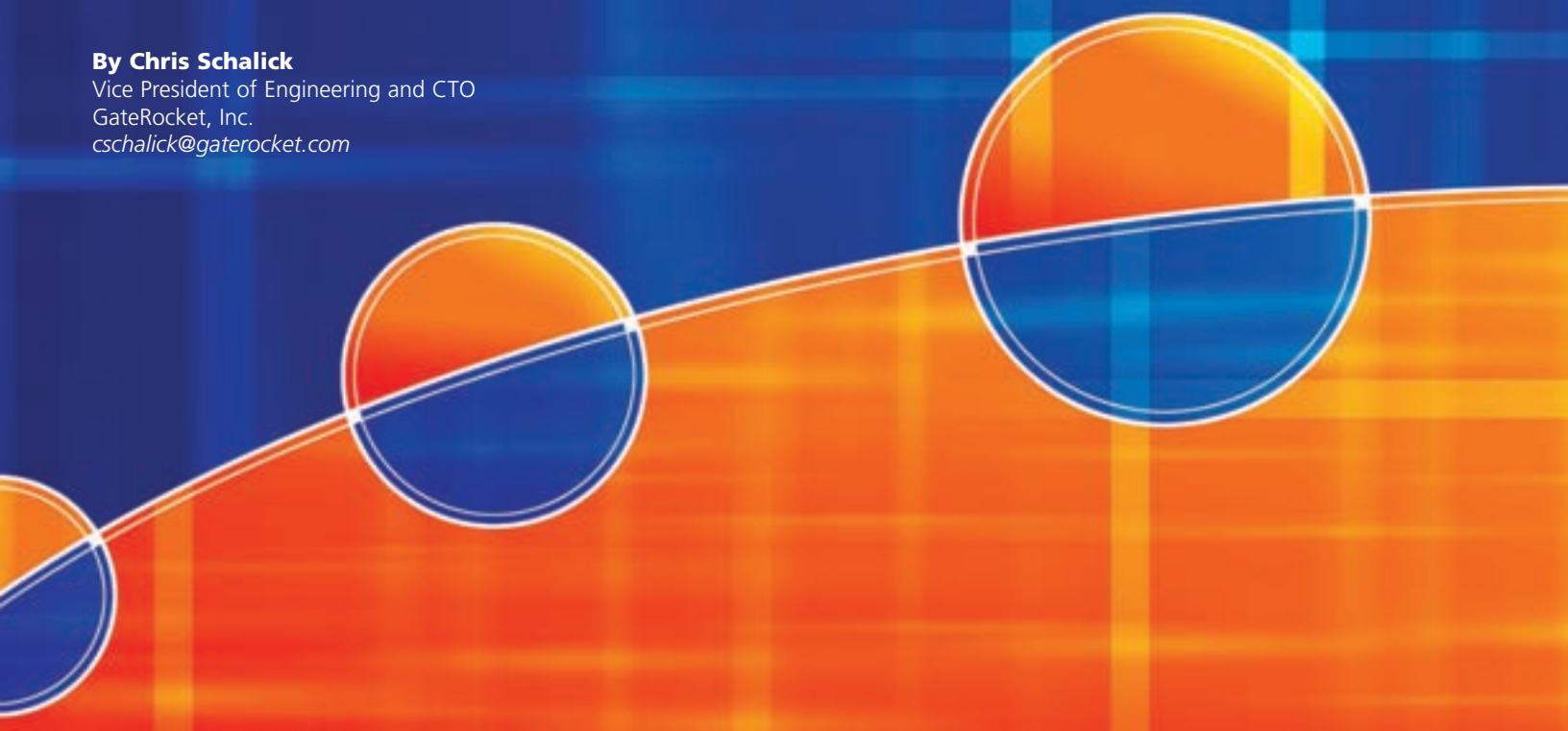
Spartan-6 FPGA Consumer Video Kit 2.0
Part Number : TB-6S-CVK2-PRO / TB-6S-CVK2-FND

Spartan-6 FPGA Broadcast Connectivity Kit
Part Number : TB-6S-BCK-FND

More Than One Way to Verify a Serdes Design in FPGA

The choice of strategy will depend on the complexity of your application and trade-offs in development time, simulation time and accuracy.

By Chris Schalick
Vice President of Engineering and CTO
GateRocket, Inc.
cschalick@gaterocket.com



As FPGAs increase in performance and capacity, developers are using them more widely for connectivity in a broad range of media, signal-processing and communications applications. At the same time, developers have turned to higher-speed serial connections for on-chip and chip-to-chip communications, replacing parallel buses to achieve significantly higher data rates. Serdes (serializer-deserializer) technology is the key enabler for this type of interface, as protocols based on a serdes approach allow higher data rates with fewer device pins.

The multigigahertz line rates that serdes enable introduce new design challenges in FPGAs, notably signal-integrity issues. Equally as demanding, if not more so, is the functional-verification challenge associated with this complex technology. FPGA designers find that logic simulation of serdes-based designs can bog down in long serial test sequences that can extend simulation times by one or two orders of magnitude. In addition, serdes technology employs complex, hierarchical protocols, making it harder to thoroughly exercise internal logic. And because serdes are often incorporated in a design by means of unfamiliar third-party IP blocks, debugging the resulting system is problematic.

Designers may use a variety of functional-verification strategies to address the serdes simulation bottleneck. Each method will affect verification performance, accuracy and engineering productivity, so the choice will involve trade-offs between development time, simulation time and simulation accuracy. Among the approaches to consider are:

- Removing the serdes from the simulations and verifying the rest of the chip using parallel communications;
- Placing a second serdes in the testbench and connecting the two back-to-back;
- Verifying the serdes portion of the design on the board in the lab, in-system;
- Executing the entire device in native FPGA hardware using an emulation-like approach;
- Writing custom behavioral models of serdes.

SIMULATION COMPLEXITY

Modern FPGA devices use configurable, high-performance serdes elements to provide access to serdes technology for a broad range of applications, ranging from simple pin-reducing chip-to-chip data transfer protocols to standards-based high-performance buses that connect to modern computer motherboards. These serdes elements are generally delivered to end users as hard IP blocks. Commonly available FPGA serdes technology has advanced to bit rates beyond 10 Gbits/second.

The Xilinx® Virtex®-5 GTP_DUAL cell is representative of modern serdes. It has eight serial I/O signals that operate from 100 Mbits/s to 3.75 Gbits/s, along with 342 core-side signals, some of which are optionally active. Completing the feature set are 184 configurable parameters, along with nine input clocks and five output clocks.

Without any further information, designing or verifying a design with those characteristics suggests a substantial endeavor. The transceiver documentation [1] lists 17 communications standards that the serdes module supports. These standards use 15 different reference clock frequencies and a unique selection of the configured parameters.

Homegrown protocols can make use of any FPGA serdes settings and clock frequencies, whether overlapping standards-based protocols or not. Of the configurable parameters, 68 have two possible values, 70 are numerical with a total of 730 variable bits and eight are non-numerical, multivalue. The span of available configurations by parameter settings alone for this simulation model is greater than 2^{730}, an astonishingly large number. The serdes transceiver can clearly operate in a wide variety of modes and support an additionally wide variety of user designs.

To accurately model the behavior of this serdes design element entails a very complex simulation model. The apparent load on a logic simulator of designs that use the simulation model is an indicator of this complexity. Indeed, FPGA serdes models can dominate simulation time for any designs that use them.

TROUBLE WITH TRANSACTORS

In any design, the standard verification practice to test an interface is to use transactors, or models that couple to pins of the interface and deliver and consume the data protocol of the interfaces. These transactors typically abstract cycle-accurate, pin-level functions to less granular, more easily understood and manipulated functions. Developing complete transactors to model FPGA serdes system connections requires a great deal of flexibility and functionality.

Modeling external interfaces for FPGA serdes means trading off development time, simulation time and accuracy. The simplest implementation uses another serdes simulation model in the transactor. This approach requires little development time, but doubles the effective load on the logic simulator of the serdes simulation model. Alternatively, you can write a “quick and dirty” behav-

ioral model with fast execution times—but at the expense of functional completeness and accuracy. An option between the two extremes is to model only the functions the FPGA design actually uses, leaving other serdes functions untested.

VERIFYING SYSTEM FUNCTIONS

Before discussing techniques for verifying serdes-based designs, let's look at some sources of verification escapes (bugs undiscovered during functional simulation) and challenges of identifying them. In other words, why is functional verification of FPGA serdes necessary?

Verification of an FPGA-based system is similar in scope to verifying the ASIC-based systems of three years past. Readily available FPGA devices are capable of implementing logic designs with 500,000 flip-flops, multiple megabytes of onboard RAM, hard and soft microprocessor cores, and a host of purpose-built communications, data-processing and bus-interface IP. Verifying a system incorporating these devices requires discipline. You must validate assumptions of interface behavior, subsystem interaction, and logical and implementation correctness. Use of FPGA serdes technology complicates each of these areas.

INTERFACE BEHAVIOR

One obvious source of functional escapes is in the immediate connections to the serdes themselves, either on the serial or parallel sides. The serial-side data is typically an encoded form of user data, encapsulated by a stack of data manipulations between the user's core-side logic and electronics outside the device under design. The stack of conversions on user data may be shallow or deep.

An example of a simple, shallow stack is shown in the design in Figure 1. It is a simple conversion of 8-bit parallel data to 10-bit serial data using built-in 8b10b encoding in the FPGA serdes device. Functional patterns to validate the user path of this example are straightforward; an incrementing pattern for 256 core-side cycles will verify that all possible data words can traverse the interface. Though architecturally a simple conversion, in practice there are hundreds of signals to connect and hundreds more parameters to configure to make use of such a conversion with the native FPGA serdes devices. It's all too easy to make mistakes of misconnection, misconfiguration or misunderstanding of device specifications, even with this uncomplicated example. Thus, even a simple use of serdes can result in functional escapes without proper verification.

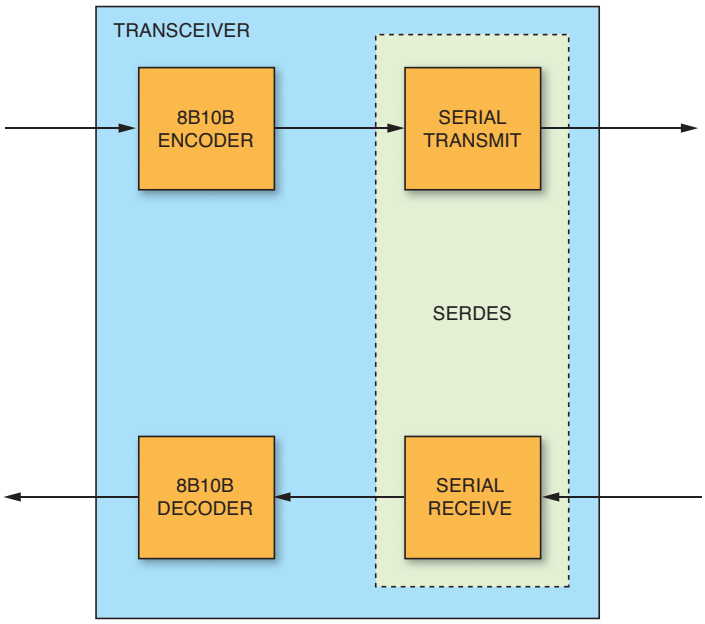


Figure 1 – Simple 8b10b FPGA block diagram

More complex examples include packet-based bus protocols like Xaui, PCI Express® or RapidIO. These interfaces are commonly crafted using a combination of hardened serdes IP and soft (programmable-logic-only) IP inside the FPGA devices. The combined FPGA IP is configured to meet system requirements. The external serial interfaces in these examples form connections to standard buses. Use of prevalidated IP to implement the bus interface helps prevent basic functional errors on the bus, but shifts the interface verification upstream to the parallel core-side data interfaces of the soft IP. The control and data operations on these interfaces differ from vendor to vendor, leaving room for misinterpretation of specifications and creating vendor-specific design and verification tasks for a standard bus interface.

Because the core-side interfaces are proprietary and nonstandard, the logic on the core side of the design can only be exercised by generating vendor-specific unique activity on the standard bus that results in desired events. This testing then is not portable from IP vendor to IP vendor. Long simulation sequences are sometimes required to activate interface signals on the core side of the IP, extending the simulation time to validate interconnecting user logic.

For the Xaui design in Figure 2, of the 59 seconds of simulation time, it takes 55 seconds to initialize the Xaui link and just 4 seconds to test data transfer on the link. This initialization sequence must be repeated for any other tests. Verifying interface behavior

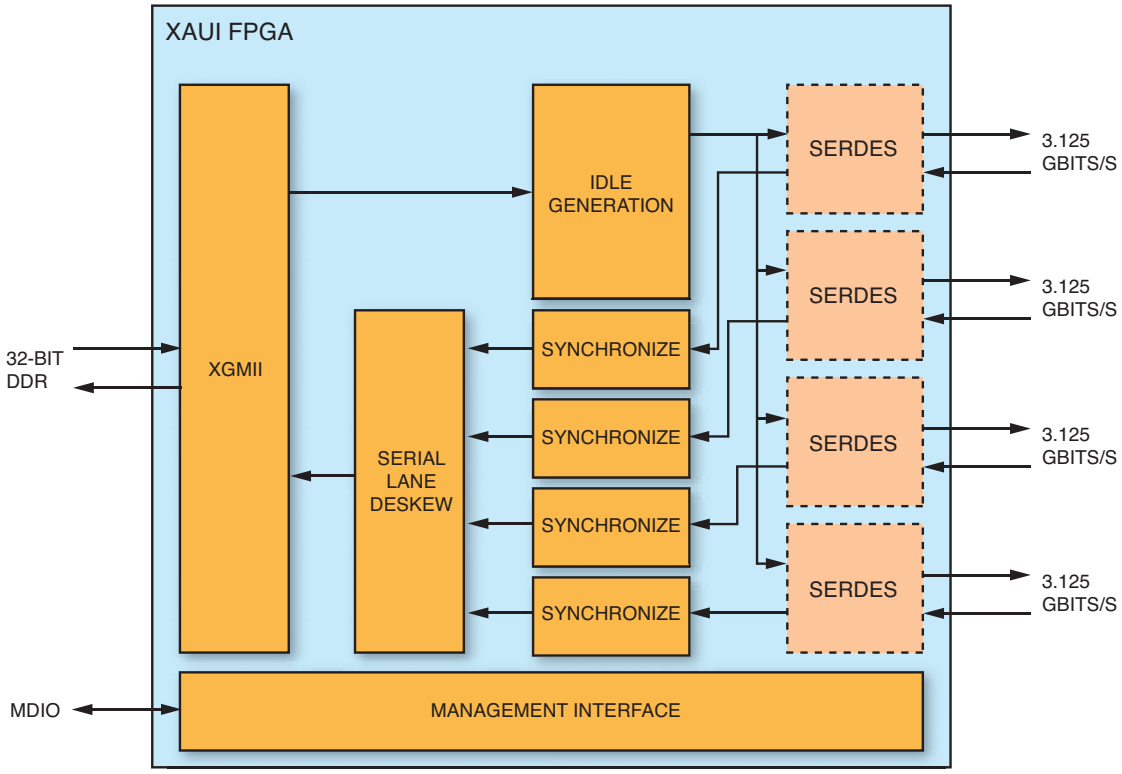


Figure 2 – Xaui FPGA block diagram

is complicated by the simulation time that FPGA serdes simulation models introduce as well as the uniqueness of protocol IP core-side interfaces.

SUBSYSTEM INTERACTION

Another area of functional escapes common in serdes-based systems is interaction between subsystems on opposite ends of a serial link. System designs employing multiple FPGAs that use serdes for chip-to-chip communication must operate to specifications both independently and in tandem to deliver contemplated system function. Design assumptions for serdes components contribute to the subsystem verification effort.

For example, the round trip of data from one FPGA device to another and back, across a serdes-based link, involves two serialization and two deserialization times. Given that these times can vary based on the temperature and voltage of the device, systems built around them must be tolerant to these variances. Designers must selectively vary models to best- or worst-case conditions to verify that designs will behave correctly under those conditions.

Consider, for instance, the circuit in Figure 3, which issues a request each cycle to a remote device attached via serial link. Each request is stored in a FIFO until the remote device acknowledges receipt. Variability in serialization and deserialization times will cause changes in the consumed depth of this FIFO. Verifying that the maximum depth occurs prior to FIFO overflow requires accurate modeling of FPGA serdes latencies.

As noted previously, accurate simulation models come at the cost of added simulation time. Using inaccurate models increases simulation speed but introduces the opportunity for escapes like the FIFO overflow. Such escapes cannot then be detected until designs reach the lab or customer site.

IMPLEMENTATION AND TOOL FLOW

It is common knowledge that gate simulations show behavior that RTL simulations cannot. Test initialization sequences and device simulation models frequently ignore or take for granted the initial state of the design in logic simulation. Further, logic simulators can demonstrate behavior with valid HDL code that, in gates, performs differently when fed with high-impedance or unknown inputs. These conditions can manifest themselves as functional escapes from RTL simulation, discoverable only in the lab. For these reasons, designers do gate simulations to “sanity check” initialization and gate behavior of implemented FPGA designs. Serdes-based FPGA designs are no exception.

Working with gates in logic simulation takes time. Typically there will be an order-of-magnitude more events for the simulator to process with a gate-level design, relative to the original

The most common functional-error escapes from implementation and tool flow are initialization errors. Undefined values presented to serdes models may go undetected in RTL.

RTL design. Coupling gate-level simulations with the performance impact of serdes simulation models can result in excessively time-consuming work. Depending on modeling accuracy, this may not even capture tolerance issues. As an example, the RTL portions of the example design shown in Figure 3 operate 30 times more slowly in gate simulations than in RTL.

The most common functional-error escapes from implementation and tool flow are initialization errors. Undefined values presented to serdes models may go undetected in RTL, where gate simulations may expose defects clearly visible in a lab environment.

APPROACHES TO VERIFICATION

Verifying FPGA serdes-based designs puts the user in the position of managing the impact of the flexibility and complexity of the serdes simulation models in the context of common escapes discussed above. Simulation time balloons cause prolonged initialization sequences before any useful testing can occur. But there are ways of handling these prolonged sequences.

For the following verification approaches, consider the two designs shown in Figure 1 and Figure 2—respectively, a simple bidirectional 8b10b serial link and a 10-Gbit/s Xaui interface. The example Xaui design, which converts and transfers data between a core-side XGMII parallel interface and the industry-standard serial Xaui interface, was generated by Xilinx's CORE Generator™ using version 8.1 of the core. The Xaui core documentation is available from Xilinx. [2]

Now let's compare the various functional-verification approaches using these two designs. In our

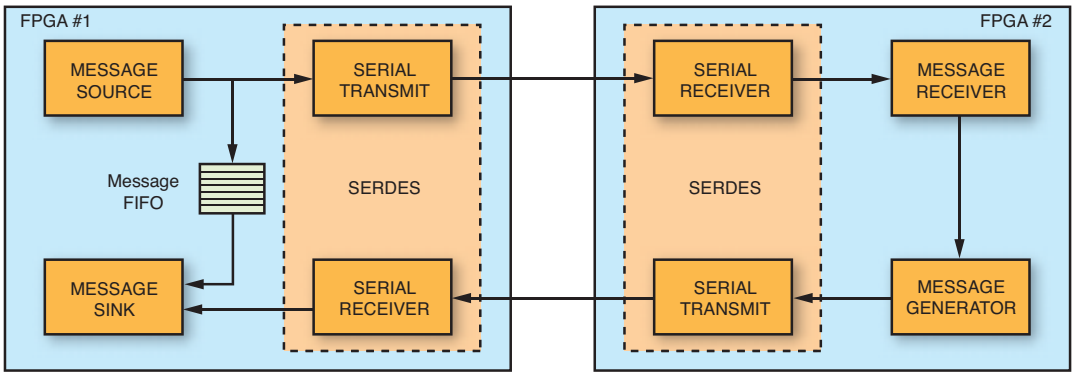


Figure 3 – Design susceptible to latency modeling inaccuracies

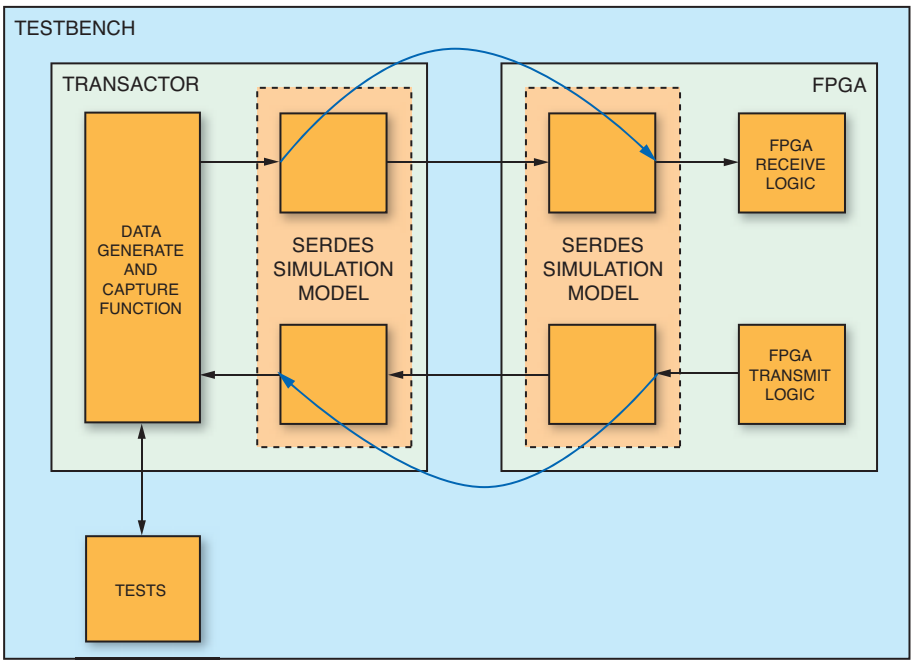


Figure 4 – Scoping around serdes in design

experiments at GateRocket, we performed all simulation runs presented on the same machine. [3]

OPTION 1: REMOVING THE SERDES MODELS

One common approach to simulating designs with FPGA serdes models is to eliminate the models altogether and replace them with shells that connect the parallel core-side data directly from source to destination. This is typically done by "scoping in" to the simulation model to drive the serdes model's parallel outputs directly and monitor its parallel core-side inputs directly from transactors in the testbench. This approach is diagrammed in Figure 4.

This approach has the advantage of eliminating the need to develop complex serial transactors, at the expense of accuracy of dynamics and function of the serial link. For example, any impact on the core-side logic of serialization or deserialization time will not be properly modeled; nor will errors occurring during deserialization. For the simple 8b10b example in Figure 1, this is a straightforward solution, because the serdes elements are used solely to transport parallel data from end to end. No control information passes through the serial link.

By contrast, using this approach with the Xaui design in Figure 2 requires substantial knowledge of the configuration and I/O of the core side of the serdes. Simply looping transmit data and the "comma character" control signals to the receiver is not sufficient, as the Xaui core logic expects transitions through synchronization states on the control outputs of the GTP_DUAL serdes elements.

This highlights the need for accurate serdes models. It's easy to build core-side circuitry that will succeed with simplified models but fail when connected to the accurate behavior of real physical devices. Creating a model that passes logic simulations while scoping around the minimum required set of core-side signals will not guarantee a functional device once the FPGA is built and operating in the lab.

Verifying the interface operation has two sides: verifying the core-side logic and interface, and verifying the serdes operation itself. The scoping method does nothing to verify the configuration or implementation of the serdes itself.

Using this approach with the design in Figure 1 results in very fast simulations; testing completes in less than 1 second. The same approach applied to the Xaui design in Figure 2 delivers 18-second simulations. The modeling effort is trivial (minutes to hours) for the 8b10b design and more substantial (days to weeks) for the Xaui design.

OPTION 2: USING A SECOND SERDES

Another approach to functionally verifying FPGA serdes designs is to use an FPGA serdes simulation model in the testbench as a transactor. This has the benefit of rapid development time, at the expense of the time it takes to run each simulation, since the load on the simulator from the serdes models doubles.

The basic approach (diagrammed in Figure 5) is to instantiate the serdes

Entity	Percent of CPU Load	Simulation Time in Seconds
GTP_DUALs	71%	41.8
Xaui core	11%	6.5
Transactors	10.6%	6.3
Testbench	7.4%	4.4

Table 1 – Simulation profile of a Xaui core

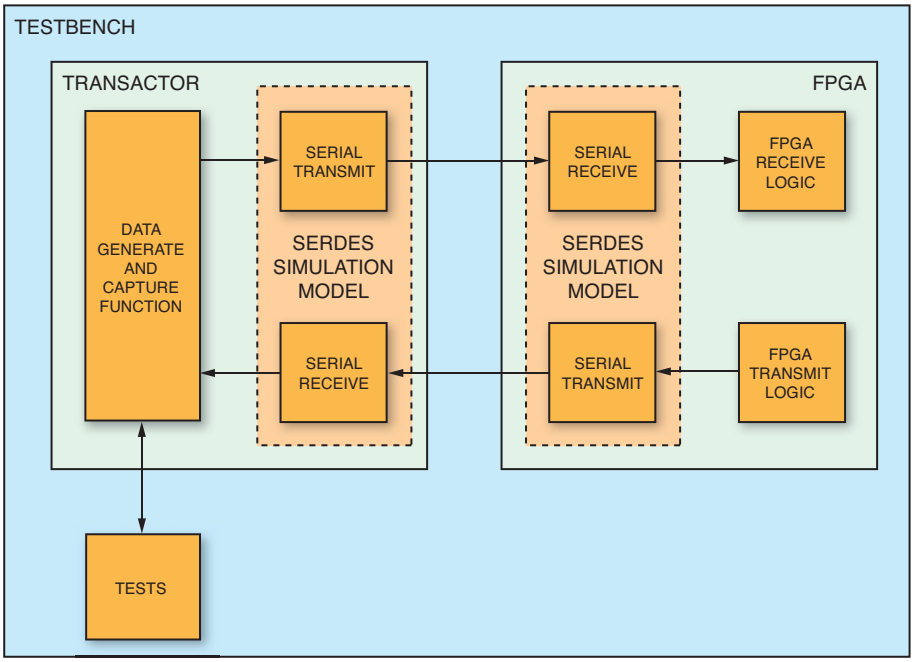


Figure 5 – Connection of testbench serdes

Approach	Development Time	Accuracy	Simulation Performance
Remove serdes	Trivial	Low	1 second
Use serdes in testbench	Short	Medium	64 seconds
Verify in the lab	Medium	High	n/a
Native hardware emulation	Long	High	1 second
Write custom behavioral models	Medium	Low	1 second

Table 2 – Comparison of approaches for 8b10b

Approach	Development Time	Accuracy	Simulation Performance
Remove serdes	Medium	Low	18 seconds
Use serdes in testbench	Short	Medium	101 seconds
Verify in the lab	Long	High	n/a
Native hardware emulation	Long	High	18 seconds
Write custom behavioral models	Medium	Low	59 seconds

Table 3 – Comparison of Approaches for Xaui

model in the transactor exactly as it is instantiated in the design being tested. This preserves all configuration options. The serial-transmit outputs of the transactor are wired to the serial-receive inputs of the design being tested—and vice versa—in the testbench.

Table 1 shows the simulator profile of the design in Figure 2 running in simulation. This testbench uses behavioral transactors for serial-side stimulus and captured response. The serdes in the Xaui design dominates the 59-second simulation. Converting the testbench to use serdes simulation models instead of behavioral transactors in the testbench doubles the load on the simulator. In this case, it nearly doubles the entire simulation time for each simulation run; runtimes increase from 59 to 101 seconds. For the 8b10b design in Figure 1, the profile is more dramatic; the simulator reports 100 percent of the time in the serdes simulation models. Using a second serdes in the testbench doubles simulation time from 32 to 64 seconds.

OPTION 3: VERIFYING SERDES IN THE LAB

One solution to verifying FPGA serdes designs is to skip verification of the serdes' serial and core-side connections in logic simulation and advance directly to the lab for validation of those parts of the design.

One benefit of this approach is the volume of data that you can transfer through the serdes interfaces in a real-time system. Simulation clock frequencies are commonly five to seven orders of magnitude slower than free-running physical FPGA devices. Verifying with silicon in the lab thus provides many-orders-of-magnitude more test cycles than software simulation. It's much harder to debug in the lab environment, however, due to the lack of visibility of the high-frequency serial signals and the opaque contents of the core-side logic.

OPTION 4: VERIFYING WITH NATIVE HARDWARE

A different solution to the verification challenge is to build an emulation platform that incorporates the FPGA design and surrounding electronics to stimulate and respond to it. You can connect a purpose-built hardware platform containing the FPGA being tested to a logic simulator or software to deliver patterns and check responses of the FPGA design.

With this technique, you must either define and produce a seamless interface to the simulator for existing RTL tests, or discard existing RTL testbenches in lieu of decoupling the FPGA design from the tests and transactors.

Circuits operating within a software simulator maintain a strict notion of synchronicity. Rigorously defined rules of hardware description language (HDL) behavior and scheduling make software-only design and simulation predictable and productive. Coupling a logic simulator with a hardware platform and maintaining setup-and-hold relationships between hardware and simulated HDL are not trivial tasks.

Designing for verification of a decoupled design is likewise a complex task. Typical FPGA verification flows build up from simple block-level functions to test full-chip functionality. Decoupling the design from the testbench is a step beyond common FPGA verification environments.

In either case, you must construct a custom hardware platform to emulate the behavior of the serdes design. Any design changes quickly make such a platform obsolete, because a different

device, device vendor or changes to the pinout will require rewiring of the printed-circuit board. A hardware platform built for the design shown in Figure 1 would not be usable for the design shown in Figure 2, since many of the serdes serial connections in one do not exist in the other.

Running the Xaui design in Figure 2 in an FPGA emulation system results in silicon-accurate serdes behavior and reduces the 59-second simulation to 18 seconds. However, crafting a flexible, purpose-built hardware emulation solution that is tightly coupled with a logic simulator is a substantial undertaking for an FPGA design-specific project. The commercial EDA community has responded, however, and GateRocket now offers a well-proven and design-independent solution called a RocketDrive, which provides that capability. The data in this paper is derived from this solution.

OPTION 5: CUSTOM BEHAVIORAL MODELS

As we saw in Option 1, replacing a serdes simulation model with a simplified model can result in incorrect behavior. It is common practice to write behavioral models for a specific mode used in a design. Rather than implementing all possible functions of a serdes in a simulation model, you can implement only the selected parameter settings and port connections. This reduces the overall scope of model development.

You can reuse models produced in this way when your new design uses the same parameters and port connections. For example, you could model the simple 8b10b design in Figure 1 with a simple 8b10b encoder/decoder. Any use of a serdes with this simple operation can then reuse this simulation model.

The usefulness of these models is limited to designs of the specified function. When new designs or serdes functions are desired, you must write new models.

A good example of this approach is seen in the testbench the Xilinx CORE Generator tool wrote for the Xaui design in Figure 2. The testbench implements behavioral serial transactors specific to the Xaui protocol. However, other designs—including the very simple 8b10b design shown in Figure 1—cannot reuse them.

This simulation runs in 59 seconds with accurate, vendor-provided models for serdes in the design and purpose-built behavioral serdes models in the transactors in the testbench.

COMPARING THE APPROACHES

Tables 2 and 3 compare the presented approaches for both the 8b10b and Xaui designs shown in Figures 1 and 2. The measurements suggest some observations and recommendations.

For example, using a second serdes is appropriate only if the serdes models represent a small portion of the overall simulation time or if there are very few tests. If the simulation profile shows a large percentage of time spent in the serdes models, the resulting impact on simulation time is dramatic.

The presented verification approaches are sometimes used in conjunction. For example, design teams might remove serdes models in simulation and also verify the serdes function in the lab.

The native hardware emulation approach gives the performance of custom behavioral models with the accuracy of the silicon behavior.

Accuracy of software simulation models generally comes at the expense of simulation performance. Hardware solutions provide higher performance and accuracy at the expense of development time.

Extending simulation times creates pressure to reduce the number of tests in verification cycles, which naturally leads to functional-defect escapes.

Using evaluation boards or target systems to verify serdes designs can put a schedule at risk while waiting for the final article.

RECOMMENDATIONS

Verifying serdes-based FPGAs is a complex process, as our examples show. Among the many challenges are debugging the core logic in the design and ensuring that the serdes are properly configured, deciding which set of the 2^{730} configurations implement your desired functionality and addressing the slow simulation time of serdes models or the gate-level simulation to accurately verify the design.

GateRocket recommends thoughtful planning and rigorous execution of verification projects, but if the tools you have at your disposal result in very long simulations or cause you to skip tests to meet project schedules, then the project risks increase significantly. We believe there is no such thing as "too much verification," but there certainly can be too little. When planning verification projects, ask yourself if you have the time to do it right—or to do it twice. Having the right tools to verify your serdes-based design is important. Just as the ASIC design team employs hardware-assisted verification to ensure the design is done right the first time, so too can a hardware emulation verification method benefit advanced serdes-based FPGA designs.

Visit *http://www.GateRocket.com* to learn more about our implementation of this technology.

REFERENCES

[1] Xilinx Inc., Virtex-5 FPGA RocketIO™ GTP Transceiver User Guide, December 2008. *www.xilinx.com/support/documentation/user_guides/ug196.pdf*

[2] Xilinx Inc., Xaui User Guide, April 2010. *www.xilinx.com/support/documentation/ip_documentation/xaui_ug150.pdf*

[3] System specifications, Intel Xeon CPU X3363 @ 2.83GHz, 6-MB cache, 16-GB DDR3 main memory, PC2-5300 / 667 MHz

How Do I Reset My FPGA?

Devising the best reset structure can improve the density, performance and power of your design.

by E. Srikanth
Solutions Development Engineer
Xilinx, Inc.
serusal@xilinx.com

In an FPGA design, a reset acts as a synchronization signal that sets all the storage elements to a known state. In a digital design, designers normally implement a global reset as an external pin to initialize the design on power-up. The global reset pin is similar to any other input pin and is often applied asynchronously to the FPGA. Designers can then choose to use this signal to reset their design asynchronously or synchronously inside the FPGA.

But with the help of a few hints and tips, designers will find ways to choose a more suitable reset structure. An optimal reset structure will enhance device utilization, timing and power consumption in an FPGA.

UNDERSTANDING THE FLIP-FLOP RESET BEHAVIOR

Before we delve into reset techniques, it is important to understand the behavior of flip-flops inside an FPGA slice. Devices in the Xilinx® 7 series architecture contain eight registers per slice, and all these registers are D-type flip-flops. All of these flip-flops share a common control set.

The control set of a flip-flop is the clock input (CLK), the active-high chip enable (CE) and the active-high SR port. The SR port in a flip-flop can serve as a synchronous set/reset or an asynchronous preset/clear port (see Figure 1).

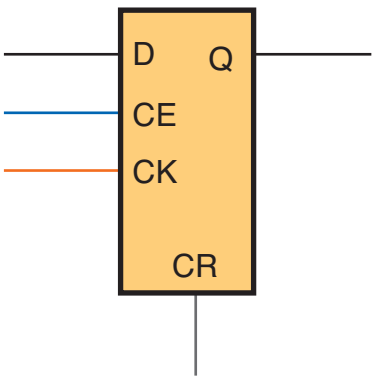


Figure 1 – Slice flip-flop control signals

The RTL code that infers the flip-flop also infers the type of reset a flip-flop will use. The code will infer an asynchronous reset when the reset signal is present in the sensitivity list of an RTL process (as shown in Figure 2a). The synthesis tool will infer a flip-flop with an SR port configured as a preset or clear port (represented by the FDCE or FDPE flip-flop primitive). When the SR port is asserted, the flip-flop output is immediately forced to the SRVAL attribute of the flip-flop.

In the case of synchronous resets, the synthesis tool will infer a flip-flop whose SR port is configured as a set or reset port (represented by an FDSE or FDRE flip-flop primitive). When the SR port is asserted, the flip-flop output is forced to the SRVAL attribute of the flip-flop on the next rising edge of the clock.

In addition, you can initialize the flip-flop output to the value the INIT attribute specifies. The INIT value is loaded into the flip-flop during configuration and when the global set reset (GSR) signal is asserted.

The flip-flops in Xilinx FPGAs can support both asynchronous and synchronous reset and set controls. However, the underlying flip-flop can natively implement only one set/reset/preset/clear at a time. Coding for more than one set/reset/preset/clear condition in the RTL code will result in the implementation of one condition using the SR port of the flip-flop and the other conditions in fabric logic, thus using more FPGA resources.

If one of the conditions is synchronous and the other is asynchronous, the asynchronous condition will be implemented using the SR port and the synchronous condition in fabric logic. In general, it's best to avoid more than one set/reset/preset/clear condition. Furthermore, only one attribute for each group of four flip-flops in a slice determines if the SR ports of flip-flops are synchronous or asynchronous.

```
signal Q:std_logic:='1';
....
async: process (CLK, RST)
begin
  if  (RST='1')then
    Q <= '0';
  else
    if (CLK'event and CLK='1') then
      Q <= D;
    end if;
  end if;
end process async;
```

(a)

INIT

SRVAL

```
signal Q:std_logic:='1';
....
sync: process (CLK)
begin
  if (CLK'event and CLK='1') then
    if (RST='1') then
      Q <= '0';
    else
      Q <= D;
    end if;
  end if;
end process sync;
```

(b)

Figure 2 – SRVAL and INIT attributes define flip-flop reset and initialization: here, VHDL code to infer asynchronous (a) and synchronous (b) reset.

RESET METHODOLOGY

Regardless of the reset type used (synchronous or asynchronous), you will generally need to synchronize the reset with the clock. As long as the duration of the global reset pulse is long enough, all the device flip-flops will enter the reset state. However, the deassertion of the reset signal must satisfy the timing requirements of the flip-flops to ensure that the flip-flops transition cleanly from their reset state to their normal state. Failure to meet this requirement can result in flip-flops entering a metastable state.

Furthermore, for correct operation of some subsystems, like state

machines and counters, all flip-flops must come out of reset on the same clock edge. If different bits of the same state machine come out of reset on different clocks, the state machine may transition into an illegal state. This reinforces the need to make the deassertion of reset synchronous to the clock.

For designs that use a synchronous reset methodology for a given clock domain, it is sufficient to use a standard metastability resolution circuit (two back-to-back flip-flops) to synchronize the global reset pin onto a particular clock domain. This synchronized reset signal can then initialize all storage elements in the clock domain by using the synchronous SR port on the flip-flops. Because both the synchronizer and the flip-flops to be reset are on the same clock domain, the standard PERIOD constraint of the clock covers the timing of the paths between them. Each clock domain in the device needs to use a separate synchronizer to generate a synchronized version of the global reset for that clock domain.

Now let's get down to brass tacks. Here are some specific hints and tips that will help you arrive at the best reset strategy for your design.

Tip 1: When driving the synchronous SR port of flip-flops, every clock domain requires its own localized version of the global reset, synchronized to that domain.

Sometimes a portion of a design is not guaranteed to have a valid clock. This can occur in systems that use recovered clocks or clocks that are sourced by a hot-pluggable module. In such cases, the storage elements in the design may need to be initialized with an asynchronous reset using the asynchronous SR port on the flip-flops. Even though the storage elements use an asynchronous SR port, the deasserting edge of the reset must still be synchronous to the clock. This requirement is characterized by the reset-recovery timing arc of the flip-flops, which is similar to a setup requirement of the deasserting edge of an asynchronous SR to the rising edge of the clock. Failure to meet this timing arc can cause flip-flops to enter a metastable state and synchronous subsystems to enter unwanted states.

The reset bridge circuit shown in Figure 3 provides a mechanism to assert reset asynchronously (and hence take effect even in the absence of a valid clock) and deassert reset synchronously. In this circuit, it is assumed that the SR ports of the two flip-flops have an asynchronous preset functionality (SRVAL=1).

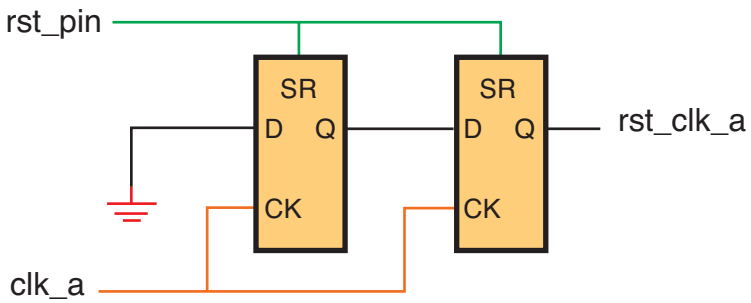


Figure 3 – Reset bridge circuit asserts asynchronously and deasserts synchronously.

You can use the output of such a reset bridge to drive the asynchronous reset for a given clock domain. This synchronized reset can initialize all storage elements in the clock domain by using the asynchronous SR port on the flip-flops. Again, each clock domain in the device needs a separate, synchronized version of the global reset generated by a separate reset bridge.

Tip 2: A reset bridge circuit provides a safe mechanism to deassert an asynchronous reset synchronously. Every clock domain requires its own localized version of the global reset with the use of a reset bridge circuit.

The circuit in Figure 3 assumes that the clock (clk_a) for clocking the reset bridge and the associated logic is stable and error free. In an FPGA, clocks can come directly from an off-chip clock source (ideally via a clock-capable pin), or can be generated internally using an MMCM or phase-locked loop (PLL). Any MMCM or PLL that you've used to generate a clock requires calibration after it is reset. Hence, you may have to insert additional logic in the global reset path to stabilize that clock.

Tip 3: Ensure that the clock the MMCM or PLL has generated is stable and locked before deasserting the global reset to the FPGA.

Figure 4 illustrates a typical reset implementation in an FPGA.

The SR control port on Xilinx registers is active high. If the RTL code describes active-low set/reset/preset/clear functionality, the synthesis tool will infer an inverter before it can directly drive the control port of a register. You must accomplish this inversion with a lookup table, thus taking up a LUT input. The additional logic that active-low control signals infers may

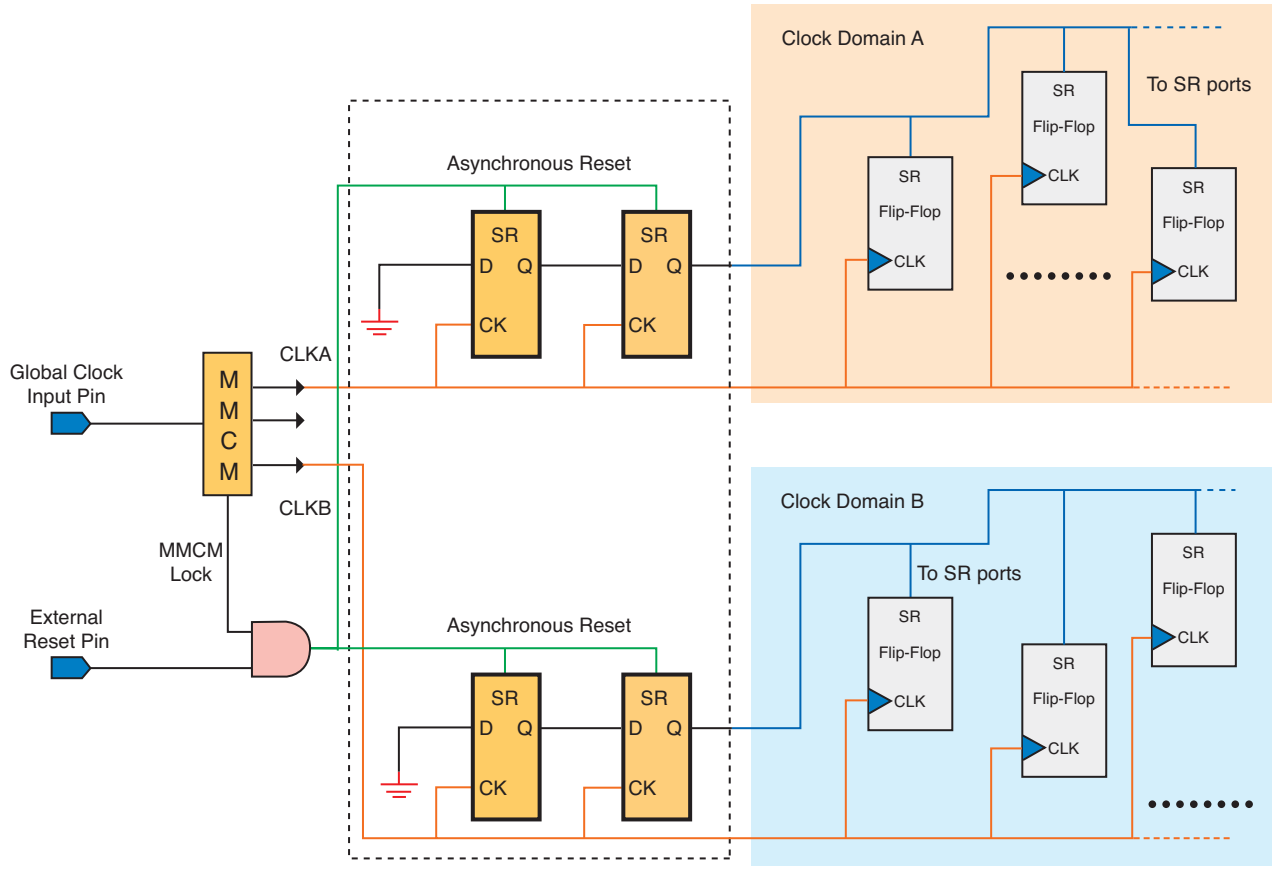


Figure 4 – Typical reset implementation in FPGAs

lead to longer runtimes and result in poorer device utilization. It will also affect timing and power.

The bottom line? Use active-high control signals wherever possible in the HDL code or instantiated components. When you cannot control the polarity of a control signal within the design, you need to invert the signal in the top-level hierarchy of the code. When described in this manner, the inferred inverter can be absorbed into the I/O logic without using any additional FPGA logic or routing.

Tip 4: Active-high resets enable better device utilization and improve performance.

It's important to note that FPGAs do not necessarily require a global reset. Global resets compete for the same routing resources as other nets in a design. A global reset would typically have high fanout because it needs to be propagated to every flip-flop in the design. This can consume a significant amount of routing resources and can have a negative impact on device utilization and timing performance. As a result, it is worth exploring other reset mechanisms that do not rely on a complete global reset.

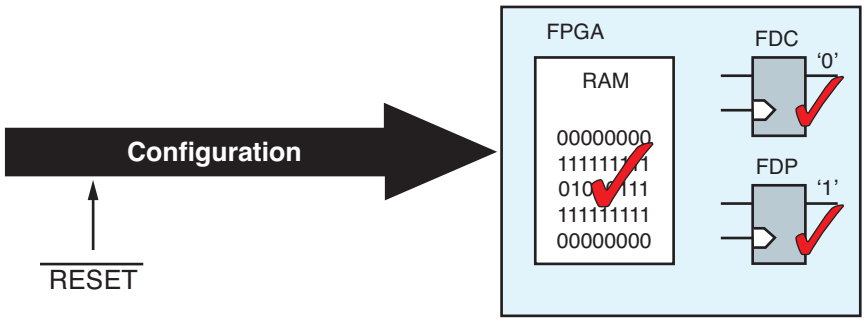


Figure 5 – FPGA initialization after configuration

When a Xilinx FPGA is configured or reconfigured, every cell (including flip-flops and block RAMs) is initialized as shown in Figure 5. Hence, FPGA configuration has the same effect as a global reset in that it sets the initial state of every storage element in the FPGA to a known state.

```
signal reg: std_logic_vector (7 downto 0) := (others <= '0');
....
process (clk) begin
   if (clk'event and clk= '1') then
     if (rst= '1') then
        reg <= '0';
     else
        reg <= D;
     end if;
   end if;
end process;
```

Figure 6 – Signal initialization in RTL code (VHDL)

You can infer flip-flop initialization values from RTL code. The example shown in Figure 6 demonstrates how to code initialization of a register in RTL. FPGA tools can synthesize initialization of the signals even though it is a common misconception that this is not possible. The initialization value of the underlying VHDL signal or Verilog reg becomes the INIT value for the inferred flip-flop, which is the value loaded into the flip-flop during configuration.

As with registers, you can also initialize block RAMs during configuration. With the increase in embedded RAMs in processor-based systems, BRAM initialization has become a useful feature. This is because a predefined RAM facilitates easier simulation setup and eliminates the requirement to have boot-up sequences to clear memory for embedded designs.

The global set reset (GSR) signal is a special prerouted reset signal that holds the design in the initial state while the FPGA is being configured. After the configuration is complete, the GSR is released and all of the flip-flops and other resources now possess the INIT value. In addition to operating it during the configuration process, a user design can access the GSR net by instantiating the STARTUPE2 module and connecting to the GSR port. Using this port, the design can reassert the GSR net, which will return all storage elements in the FPGA to the state specified by their INIT property.

The deassertion of GSR is asynchronous and can take several clocks to affect all flip-flops in the design. State machines, counters or any other logic that can change state autonomously will require an explicit reset that deasserts synchronously to the user clock. As a result, using GSR as the sole reset mechanism can result in an unreliable system.

Hence, you are better served by adopting a mixed approach to manage the startup effectively.

Tip 5: A hybrid approach that relies on the built-in initialization the GSR provides, along with explicit resets for portions of the design that can start autonomously, will result in better utilization and performance.

After using the GSR to set the initial state of the entire design, use explicit resets for logic elements, like state machines, that require a synchronous reset. Generate the synchronized version of the explicit reset using either a standard metastability resolution circuit or a reset bridge.

USE APPROPRIATE RESETS TO MAXIMIZE UTILIZATION

The style of reset used in RTL code can have a significant impact on the ability of the tools to map a design to the underlying FPGA resources. When writing RTL code, it is important that designers tailor the reset style of their subdesign to enable the tools to map to these resources.

Other than using the GSR mechanism for initialization, you cannot reset the contents of SRLs, LUTRAMs and block RAMs using an explicit reset. Thus, when writing code that is expected to map to these resources, it is important to code

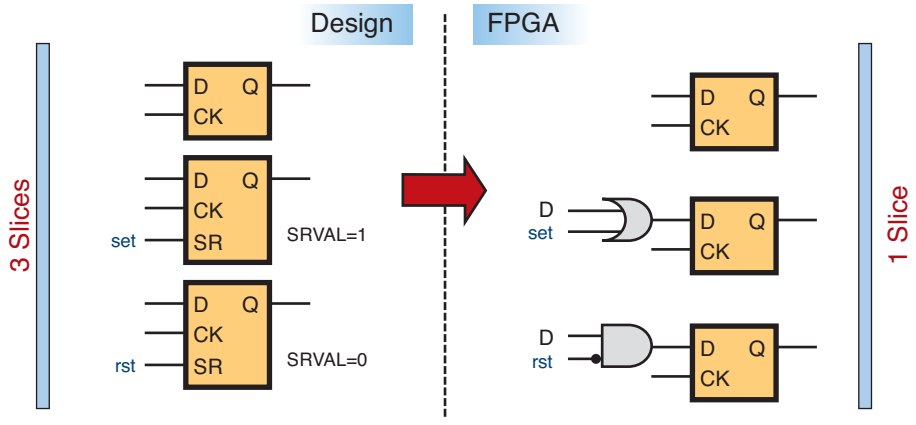


Figure 7 – Control set reduction on SR

specifically without reset. For example, if RTL code describes a 32-bit shift register with an explicit reset for the 32 stages in the shift register, the synthesis tool would not be able to map this RTL code directly to an SRL32E because it cannot meet the requirements of the coded reset using this resource. Instead, it would either infer 32 flip-flops or infer some additional circuitry around an SRL32E in order to implement the required reset functionality. Both of these solutions would require more resources than if you had coded the RTL without reset.

Tip 6: When mapping to SRLs, LUTRAMs or block RAMs, do not code for a reset of the SRL or RAM array.

In 7 series devices, you cannot pack flip-flops with different control signals into the same slice. For low-fanout resets, this can have a negative impact on overall slice utilization. With synchronous resets, the synthesis tool can implement the reset functionality using LUTs (as shown in Figure 7) rather than control ports of flip-flops, thereby removing the reset as a control port. This allows you to pack the resulting LUT/flip-flop pair with other flip-flops that do not use their SR ports. This may result in higher LUT utilization but improved slice utilization.

Tip 7: Synchronous resets enhance FPGA utilization. Use them in your designs rather than asynchronous resets.

Some of the larger dedicated resources (namely block RAMs and DSP48E1 cells) contain registers that can be inferred as part of the dedicated resource functionality. Block RAMs have optional output registers that you can use to improve clock frequency by means of an additional clock of latency. DSP48E1 cells have many registers that you can use both for pipelining, to increase maximum clock speed, as well as for cycle delays (Z-1). However, these registers only have synchronous set/reset capabilities.

Tip 8: Using synchronous resets allows the synthesis tool to use the registers inside dedicated resources like DSP48E1 slices or block RAMs. This improves overall device utilization and performance for that portion of the design, and also reduces overall power consumption.

If the RTL code describes asynchronous set/reset, then the synthesis tool will not be able to use these internal registers. Instead, it will use slice flip-flops since they can implement the requested asynchronous set/reset functionality. This will not only result in poor device utilization but will also have a negative impact on performance and power.

MANY OPTIONS

Various reset options are available for FPGAs, each with its own advantages and disadvantages. The recommendations outlined here will help designers choose a suitable reset structure for their design. An optimal reset structure will enhance the device utilization, timing and power consumption of an FPGA.

Many of the tips explained in this article are described in the Designing with the 7 Series Families training course. More information on Xilinx courses is available at *www.xilinx.com/training*.

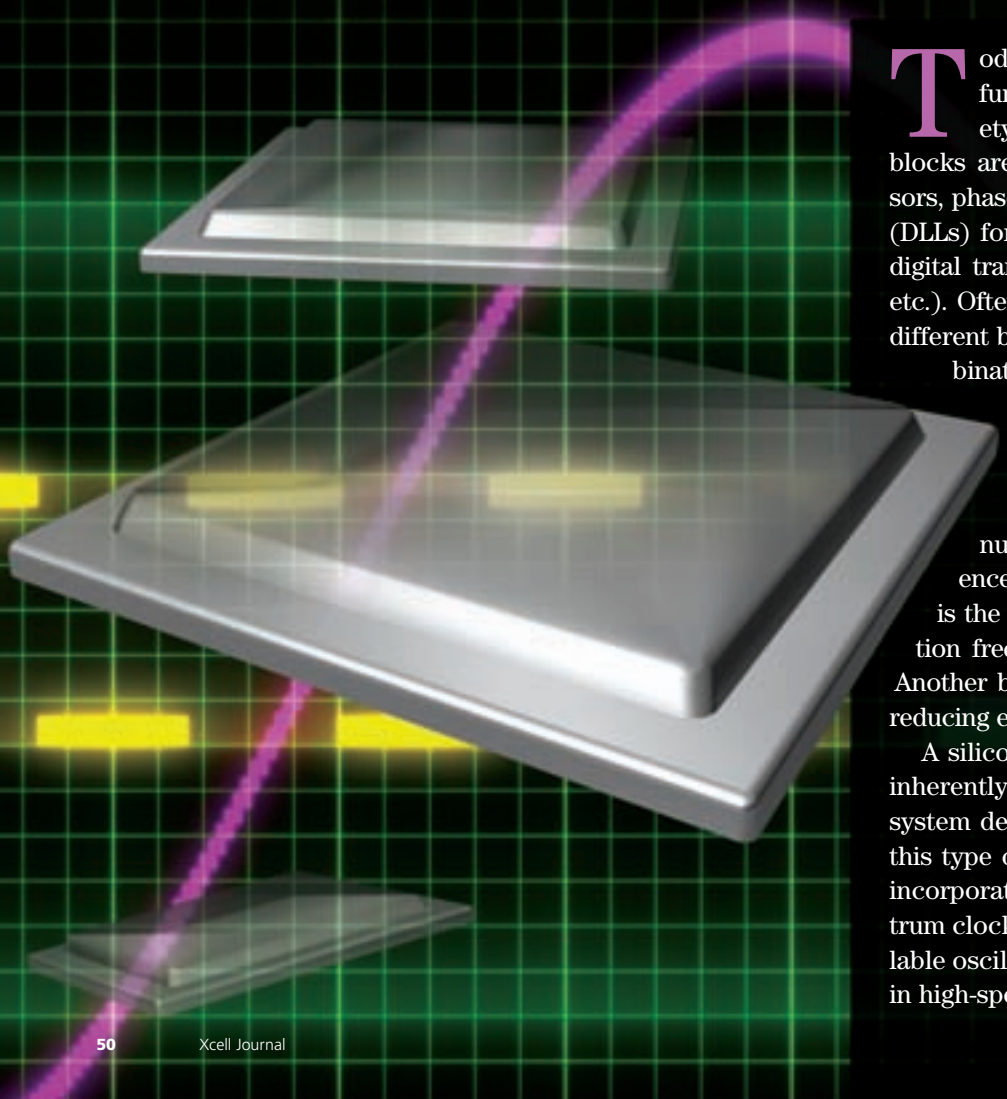
About the Author

Srikanth Erusalagandi is currently working as a solutions development engineer on Xilinx's Global Training Solutions team. Srikanth develops content for the Xilinx training courses. His areas of expertise are FPGA design and connectivity. Prior to joining Xilinx in January 2010, he spent close to six years at MosChip Semiconductors as an applications engineer.

Programmable Oscillators Enhance FPGA Applications

Clock oscillators offer unique advantages in highly customizable FPGA-based systems, including flexibility and EMI reduction.

by Sassan Tabatabaei
Director, Strategic Applications
SiTime Corp.
stabatabaei@sitime.com



Today's complex FPGAs contain large arrays of functional blocks for implementing a wide variety of circuits and systems. Examples of such blocks are logic arrays, memory, DSP blocks, processors, phase-locked loops (PLLs) and delay-locked loops (DLLs) for timing generation, standard I/O, high-speed digital transceivers and parallel interfaces (PCI, DDR, etc.). Often, these FPGAs use multiple clocks to drive different blocks, typically generating them using a combination of external oscillators and internal PLLs and DLLs. The system designer has to decide how to combine external and internal resources for optimal clock tree design.

Programmable clock oscillators offer a number of advantages when used as timing references for FPGA-based systems. Chief among them is the design flexibility that arises from high-resolution frequency selection for clock tree optimization. Another big benefit is spread-spectrum modulation for reducing electromagnetic interference (EMI).

A silicon MEMS clock oscillator architecture that is inherently programmable solves many problems for system designers who use FPGAs. The architecture of this type of microelectromechanical system can easily incorporate additional features such as spread-spectrum clocking for EMI reduction and a digitally controllable oscillator for jitter cleaning and fail-safe functions in high-speed applications.

FREQUENCY SELECTION

A typical system needs a number of clock frequencies. Some are standard, either because they are mandated by an industry specification—for example, 100 MHz for PCI Express®—or by virtue of being used widely, such as 75 MHz for SATA or 33.333 MHz for PCI™. Such frequencies are associated with I/O interfaces to ensure interoperability, because the two sides of the interface may not belong to the same system. In contrast, the user may select the clock frequency for driving processors, DSP and state-machine engines to optimize for speed, power or resource usage.

The PLL feedback loop forms a band-limited control system. The output period jitter depends mostly on the reference clock phase noise (PN_{in}) and the internal VCO phase noise (PN_{VCO}), as formulated here.

$$PN_{out} \approx H_{in}.PN_{in} + H_{VCO}.PN_{VCO}$$

As related to the output phase noise, the input reference and VCO phase noise contributions go through low-pass and high-pass filter responses, H_{in} and H_{VCO}

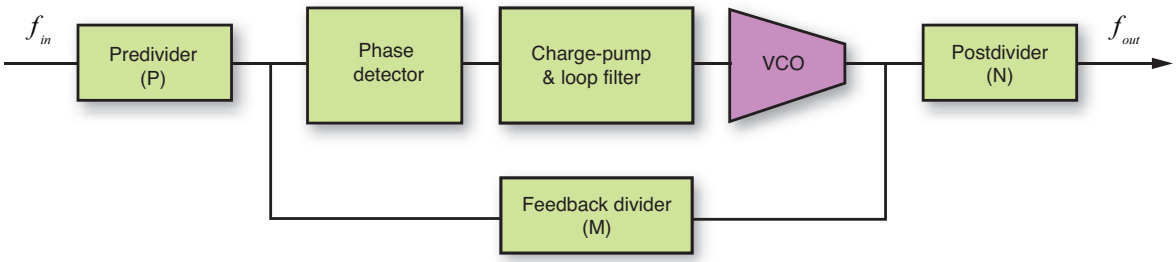


Figure 1 – Block diagram of a typical integer PLL

When optimizing for speed, you should clock the processing engines with the highest clock frequency to maximize the number of operations per second. However, the clock period jitter must also be low to ensure the minimum clock period is greater than the critical timing path in the design; otherwise logical errors may occur. A common approach for frequency selection is to use internal FPGA PLLs to synthesize a higher-frequency clock from a standard external reference oscillator. This approach is effective if the internal PLL has high-frequency resolution and low jitter.

Some FPGAs incorporate internal, low-noise fractional PLLs that meet all of these requirements. In this case, you can use a simple external oscillator reference. However, in many cases, FPGAs use PLLs with a ring VCO and integer feedback dividers to synthesize different frequencies. Such PLLs are small and flexible, relatively easy to design and control, and consume very little power. But when using these internal PLLs, it is difficult to achieve high resolution and low jitter simultaneously.

The general architecture of an integer PLL is shown in Figure 1. The PLL output frequency is programmed with a combination of predivider (P), feedback divider (M) and postdivider (N), as in the equation below:

$$f_{out} = \left(\frac{f_{in}}{P}\right).\left(\frac{M}{N}\right)$$

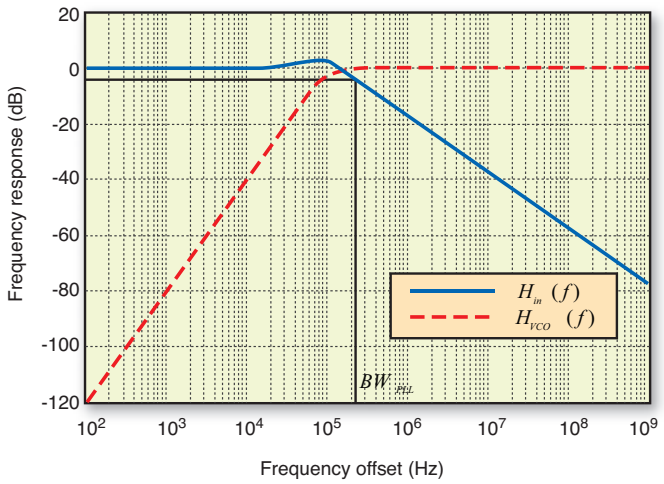


Figure 2 – Phase noise transfer function examples for input and VCO, using a second-order PLL

respectively. The cutoff frequencies of H_{VCO} and H_{in} are directly related. Figure 2 illustrates how H_{in} and H_{VCO} relate to each other in an exemplary second-order PLL. The maximum PLL bandwidth depends on the phase detector update rate. In most practical PLLs, the maximum practical limit is as shown below:

$$BW_{PLL} < (f_{in} / P) / 10$$

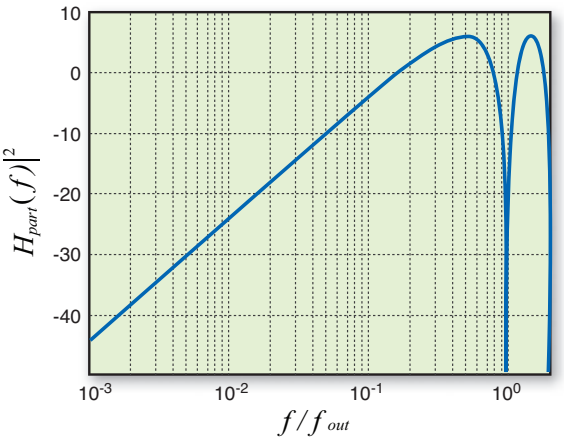


Figure 4 – The filter response spectrum relating phase noise to period jitter

For example, if the PLL input frequency is 40 MHz and P=40, the maximum practical PLL bandwidth will be 100 kHz.

The period jitter is related to phase noise by a sine filter response, as shown in Figure 4. [1] As you can see, period jitter is more sensitive to overall PLL output phase noise at frequency offsets closer to $f_{out}/2$. Since the PLL bandwidth is significantly smaller than $f_{out}/2$, the reference clock typically makes a small contribution to the period jitter while the internal VCO phase noise contributes more.

The higher bandwidth of the PLL reduces the contribution of the internal VCO to output period jitter and yields lower overall period jitter. In most cases, it is desirable to set the bandwidth higher to reduce the internal VCO noise and improve jitter. On the other hand, achieving high-frequency resolution requires larger values of divider P, which limits the maximum PLL bandwidth. This conflict imposes a trade-off between high resolution and low jitter. Use of an external high-resolution oscillator alleviates this trade-off by moving the burden of the high resolution to the external reference.

High-performance programmable oscillators, such as those available from SiTime, are an example of such devices. With these types of oscillators, the internal PLL only needs to support very limited frequency synthesis functionality, which allows higher bandwidth and less jitter.

Another advantage of programmable external reference oscillators is the ability to select a higher-frequency reference. This allows higher-bandwidth internal PLLs, which yield lower jitter. For example, an application may require a 56-MHz clock with 10-picosecond RMS period jitter to meet timing requirements.

Figure 5 shows two ways of obtaining a 56-MHz clock. The first uses a standard 25-MHz reference and the second one uses a nonstandard 28-MHz reference. The first method requires a large predivide ratio to achieve the required resolution, but leads to higher output jitter. The second method minimizes the P value and allows higher PLL bandwidth, leading to lower output period jitter.

Most programmable oscillators use a resonator element and one or more PLLs to synthesize different frequencies. Traditionally, quartz crystals have been the choice for stable resonators. However, packaging challenges have limited the availability of such programmable oscillators. More recently, silicon MEMS oscillators have arrived on the market, offering a cost-effective combination of stable resonators and high-performance PLLs in a number of industry-standard small packages. These oscillators provide an attractive solution for FPGA clocking for optimizing the clock tree in FPGA systems. Such clocks also meet the requirements for more stringent jitter specifications of high-speed transceivers. [2]

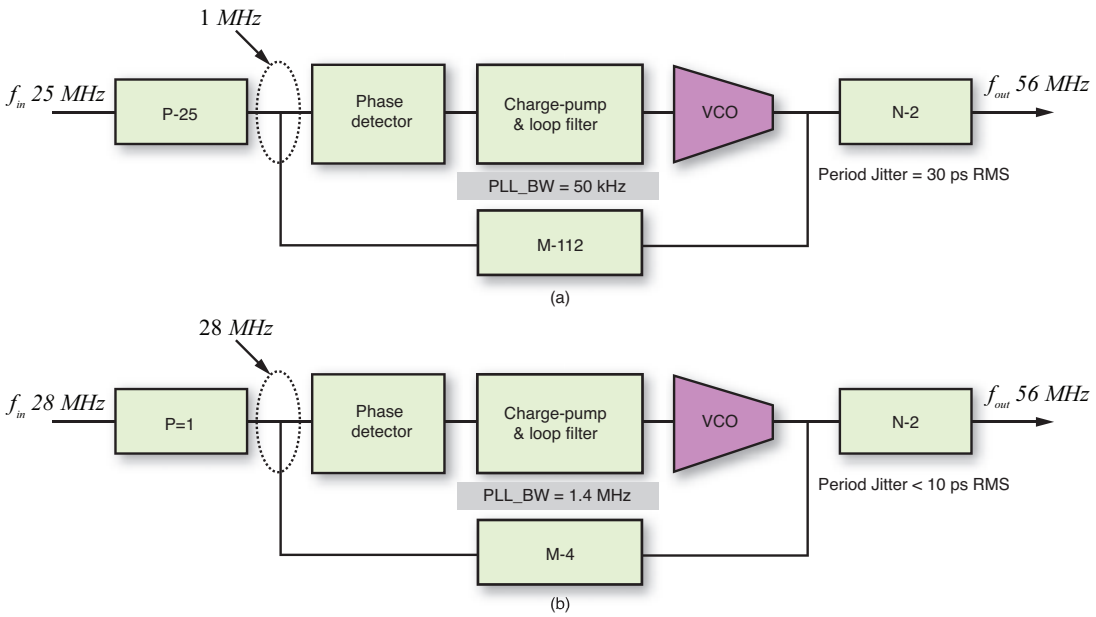


Figure 5 – (a) Lower PLL bandwidth, higher jitter architecture; (b) Higher PLL bandwidth, lower jitter architecture

Xilinx Part Number	Xilinx Evaluation Kit Description	SiTime Oscillator Part Number	SiTime Oscillator Description
ML605	Virtex®-6 FPGA Evaluation Kit	SiT9102AI-243N25E200.00000	200 MHz, Differential LVDS, 2.5V
		SiT8102AN-34-25E33.00000	33 MHz, LVCMOS, 2.5V
		SiT8102AN-44-25E47.00000	47 MHz, LVCMOS, 2.5V
SP605	Spartan®-6 FPGA Evaluation Kit	SiT9102AI-243N25E200.00000	200 MHz, Differential LVDS, 2.5V
		SiT8102AN-34-25E33.00000	33 MHz, LVCMOS, 2.5V
SP601	Spartan-6 FPGA Evaluation Kit	SiT9102AI-243N25E200.00000	200 MHz, Differential LVDS, 2.5V

Table 1 – SiTime programmable devices on Xilinx FPGA demo boards

EMI REDUCTION

Once a stable resonator pairs up with a high-performance synthesizer in a programmable oscillator, many other useful clock features become easily accessible. One example is spread-spectrum clocking (SSC) for EMI reduction.

The SSC oscillator is a clock whose frequency is modulated to ensure the energy of the clock signal is spread over a larger frequency range, hence reducing overall peak electromagnetic radiation within a given frequency range. SSC is especially useful in FPGA-based systems because it reduces EMI from all circuits and I/Os that share the same clock source. By contrast, trace filtering and rise/fall control methods tend to decrease EMI in certain sections of the system. Figure 6 shows how SSC reduces peak EMI radiation.

The important parameters in SSC are modulation range and modulation method (center-spread or down-spread).

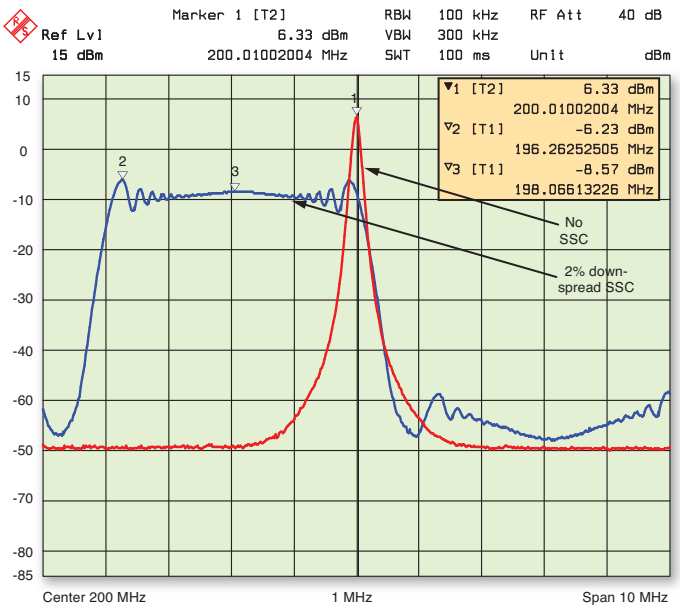


Figure 6 – Spread-spectrum clocking modulation to reduce peak EMI radiation

Some programmable oscillators, such as the SiT9001 from SiTime, provide a wide selection of SSC modulation range, from 0.5 percent to 2 percent in both down-spread and center-spread flavors. This menu election allows designers to optimize the SSC for the best system performance while minimizing EMI. [3]

Another example of a useful feature that fractional-N PLLs in programmable oscillators enable is digitally controllable oscillators. The DCO is a powerful feature for implementing low-bandwidth, fully digital PLLs with FPGAs for fail-safe, holdover or jitter cleaning in high-end telecom and networking systems.

THE SILICON MEMS ADVANTAGE

New silicon MEMS oscillators have expanded the portfolio of commercially available programmable oscillators significantly over the last several years. These oscillators enable the user to customize the reference frequency for optimal clock tree design by choosing the best combination of external reference and the FPGA's internal PLL parameters. Additionally, designers can easily select power supply voltage, package, temperature range, frequency stability and drive strength to match the application needs. The programmable aspects of these oscillators also reduce lead time and allow rapid prototyping and fast production schedules.

SiTime's single-ended and differential silicon MEMS oscillators already reside on some Xilinx demo boards (see Table 1). These parts can be ordered for different frequencies, voltages and packages for optimal system performance.

References

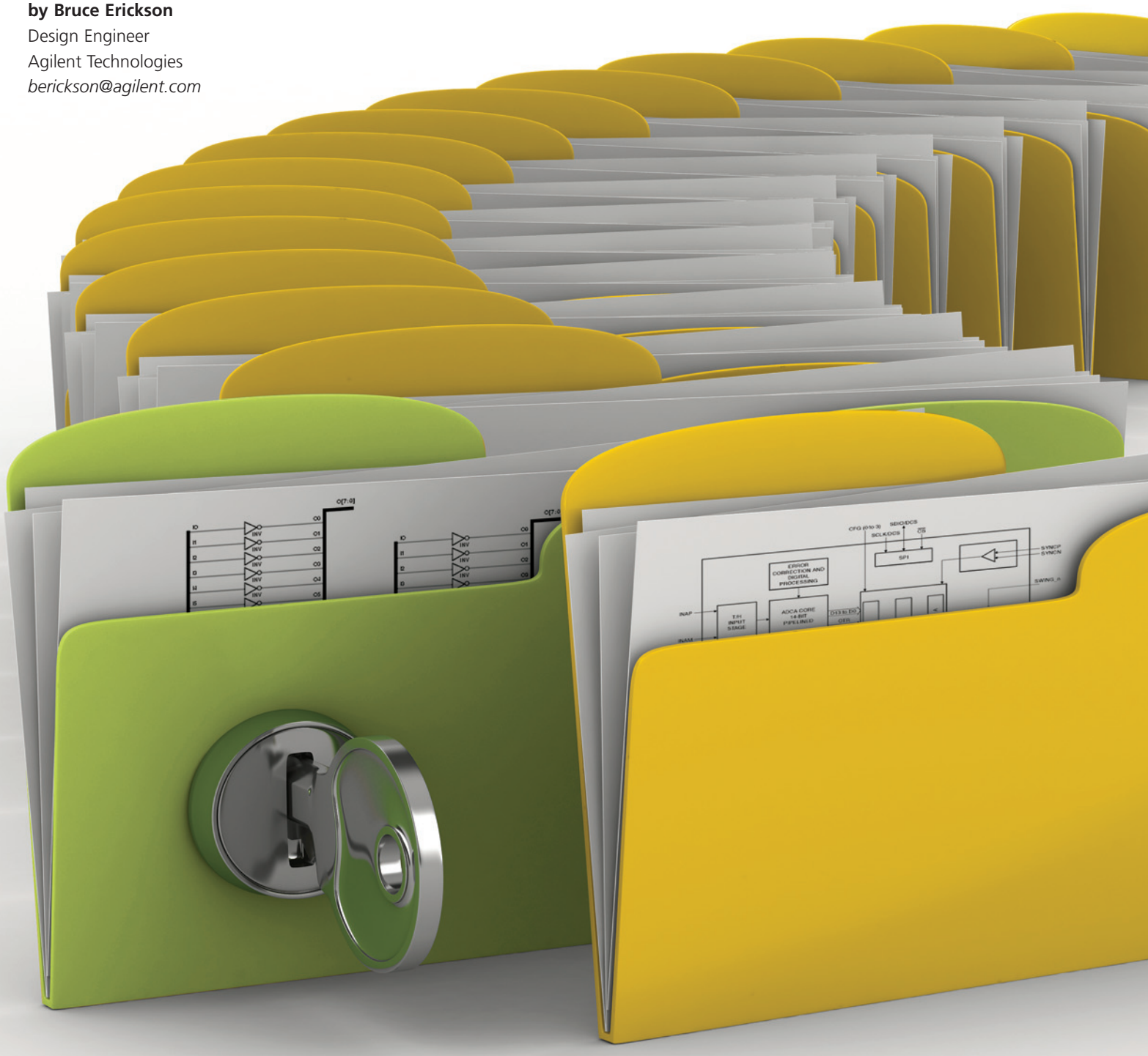
[1] Mike Li, *Jitter, Noise, and Signal Integrity at High-speed*, Prentice Hall, 2007

[2] SiTime, "Phase Noise and Jitter Requirements for Serial I/O Applications," application note AN 10012

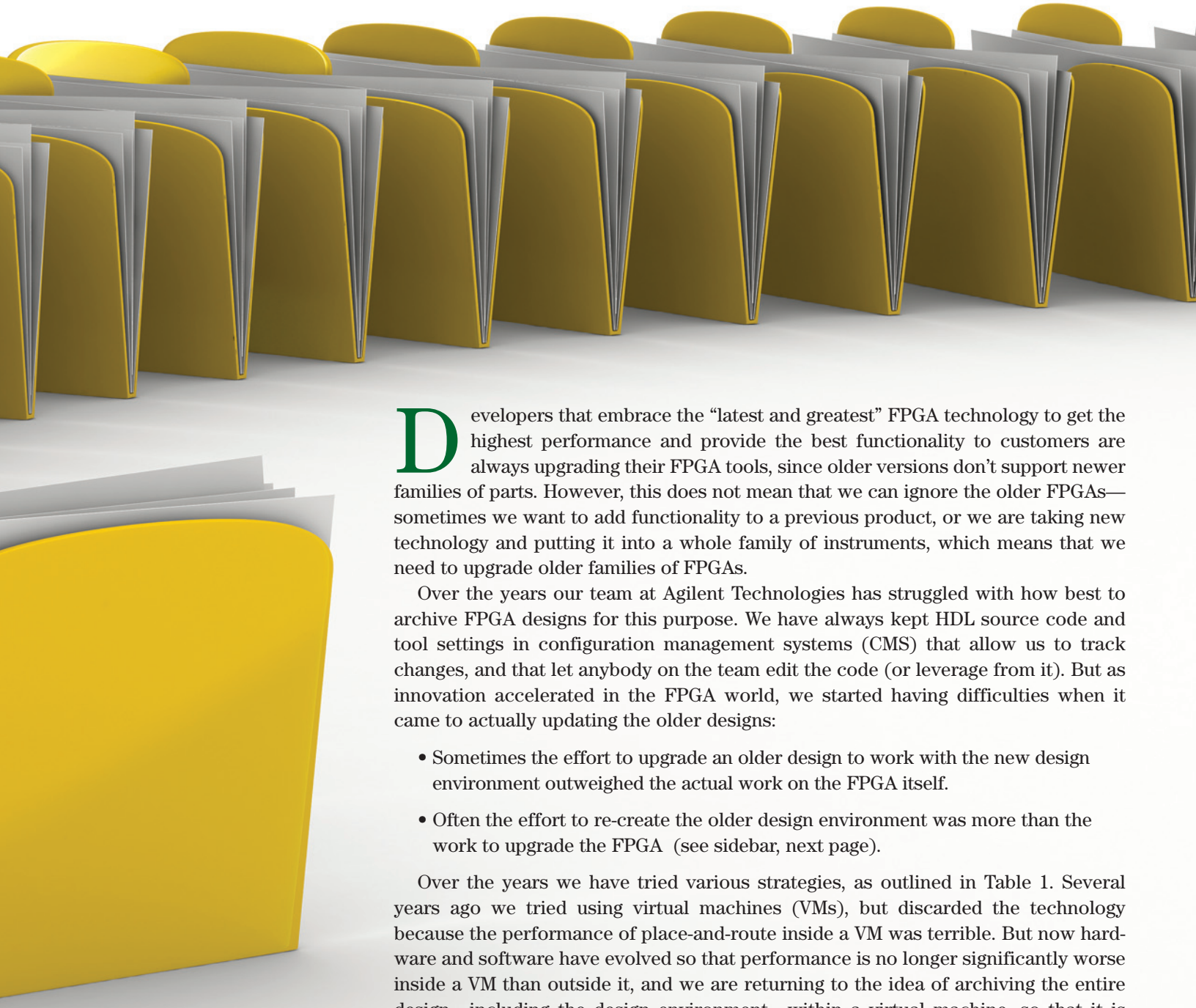
[3] Sassan Tabatabaei, *Clocking Strategies for EMI Reduction*, Interference Technology – EMC Test and Design Guide, November 2010

Archiving FPGA Designs for Easy Updates

by Bruce Erickson
Design Engineer
Agilent Technologies
berickson@agilent.com



Virtual machines can store your entire design, from design environment to FPGA code, making it easy to change an FPGA after you've completed development.



Developers that embrace the "latest and greatest" FPGA technology to get the highest performance and provide the best functionality to customers are always upgrading their FPGA tools, since older versions don't support newer families of parts. However, this does not mean that we can ignore the older FPGAs—sometimes we want to add functionality to a previous product, or we are taking new technology and putting it into a whole family of instruments, which means that we need to upgrade older families of FPGAs.

Over the years our team at Agilent Technologies has struggled with how best to archive FPGA designs for this purpose. We have always kept HDL source code and tool settings in configuration management systems (CMS) that allow us to track changes, and that let anybody on the team edit the code (or leverage from it). But as innovation accelerated in the FPGA world, we started having difficulties when it came to actually updating the older designs:

- Sometimes the effort to upgrade an older design to work with the new design environment outweighed the actual work on the FPGA itself.
- Often the effort to re-create the older design environment was more than the work to upgrade the FPGA (see sidebar, next page).

Over the years we have tried various strategies, as outlined in Table 1. Several years ago we tried using virtual machines (VMs), but discarded the technology because the performance of place-and-route inside a VM was terrible. But now hardware and software have evolved so that performance is no longer significantly worse inside a VM than outside it, and we are returning to the idea of archiving the entire design—including the design environment—within a virtual machine, so that it is easy to make changes to an FPGA after initial development is finished.

At the end of an FPGA project, we copy the design tools and design to a VM's virtual hard drive (VHD). Then we check the VHD into the CMS. Because the VHD is binary, there is no usable "change record" for the HDL (Verilog or VHDL) sources. So we

also check the HDL sources into a CMS to make it easy to track the logical changes to the FPGA design. When we need to make a quick change, we simply check out the VHD to our current machine and run the VM using it. The result is a window on our screen that looks like our old design environment—the old tools and even the operating system are all there. There is no need to upgrade the design to make a simple change.

WHAT ARE VIRTUAL MACHINES?

Virtual machines consist of a program, often called a "hypervisor," that intercepts I/O from programs running under it. The hypervisor provides emulated hard drives (also known as virtual hard drives, or VHDs), along with emulated or, more commonly, "pass-through" LANs, DVD drives, USB ports and even RS232 peripherals. A BIOS or hardware abstraction layer (HAL) is also part of the VM program. The combination of these functions is a virtual machine, in just the same way that the marriage of a CPU, peripherals and BIOS is commonly called a personal computer.

When the VM starts from power-up you see a BIOS boot display; then you can load an operating system from a CD or boot from an OS already loaded on the VHD. This means that you can, for example, run a full version of Linux under a VM running on a host Windows machine, a full version of Windows on a Linux machine—or even a full version of Windows on a Windows machine. Why do most people find this useful? Because they can change physical hardware without making any changes to an execution environment. Many companies run several VMs on a single computer, allowing them to consolidate several smaller servers without having to change anything about the servers as far as their software—and the software that uses the server—is concerned.

Around 2006 both Intel and AMD started making hardware support for virtual machines more widespread; basically, they added methods for swapping entire sets of registers and redirecting I/O on a per-process basis. This means that software in a VM runs at almost the same speed as it would run under the real hardware. In fact, compute-intensive programs (such as place-and-route) run at almost the same speed under a VM as they do on the main OS, because the VM doesn't need to get involved in a lot of I/O redirection.

VMware and Sun/Oracle make excellent VMs. VMware has, in my opinion, the widest and most stable VM environment (and the priciest). Sun's (now Oracle's) VM, known as VirtualBox, is also pretty good, and is licensed under GPL version 2; download it from *www.virtualbox.org*. Microsoft provides a VM for Windows XP known as Virtual PC 2007. Another VM, called Windows Virtual PC, is available with Windows 7 Enterprise and Professional editions; it's free, but you need to download it from Microsoft.

CREATING A VM WITH AN FPGA DESIGN

At this point, let's walk through the process of creating and using a VM. We'll focus on VirtualBox under a Windows 7 system, emulating a Windows XP system with Xilinx® ISE® 10.1. However, the general steps are applicable to other VMs as well.

The first step is to verify that hardware support for virtual machines is enabled. I don't know why, but the last four PCs (from various vendors) I have looked at do not have the CPU's virtualization technology enabled; I've had to go into the BIOS to enable it. It exists under various names, like "virtualization acceleration," "virtualization technology," "VMT" and, in one case, "VT." Microsoft provides a program to

Challenges in upgrading designs to new tools

- If the design is too old, it may not be possible to read the project-definition file into the new tools.
- If you can't read in the old project, it can be challenging to figure out what are the correct files (or versions of files), as well as language and processing settings.
- IP that you used in the design may not be available for the new tools—or may require new licensing fees for the upgrade.
- It could take a lot of effort to upgrade certain parts of the design, such as soft-microprocessor cores.

Challenges in restoring old design environments

- Old and new versions of tools may not coexist easily. If you have to remove your current tools, your current project is going to suffer.
- It may take a long time to find and install old versions of tools.
- It may not be clear what version of the tools were used with the project; you may end up trying several versions before you find one that works.
- Even if the version is documented, it might not be clear what patches are necessary.
- Settings defined by environment variables may not be documented, and finding the right values might take a long time.
- Old tools may not even run on newer operating systems.

	Keeping original PC w/o upgrades	Keeping old root hard drive	Keeping old project hard drive	Keeping VHD
Cost	Expensive: Reserving a PC without upgrading it or using new tools on it is costly.	Inexpensive. Hard drives are cheap!	Inexpensive.	Inexpensive: just some hard-drive space. VM may or may not cost you.
Setup complexity	Easy; turn off the system and lock it away. Probably best to put a note on it documenting what it is for.	Easy; just pull out the hard drive. Document the PC's hardware for later use. But you need to put a new one into the PC if you are reusing the PC, and install a new OS on it.	Easy; just make sure hard drive has tools installed on it, and that all System Registry entries related to the tools are archived somehow.	Moderate; must set up design tools in it, and verify that the build works.
Ease of restoring the design environment	Easy; turn on the old PC and (if it boots) you are ready to go.	A little harder: you must find a PC that has hardware compatible with the original system for Windows to boot easily. But after that, pretty easy.	Need some way to archive and restore the System Registry related to the tools.	Same VM software must be available; restore the VHD and you are ready to go.

Table 1 – Design environment archiving strategies

check to see if it is enabled; search *microsoft.com* for Hardware-Assisted Virtualization Detection Tool.

Next, install the hypervisor (the VM software). During the initialization, the VM software will install one or more drivers. One of them usually installs a "filter" on the LAN driver, so the LAN can become disconnected during installation. And as always, there is a danger that new drivers can cause problems. Therefore, your first step is to create a "restore point" so that you can recover from the potential installation of an incompatible driver. The second step is to make sure you are sitting at the computer as you install the VM; don't try to do it by means of Remote Desktop or any other remote control system that uses the LAN.

VirtualBox has several install options. There are two that are not necessary for our use:

- VirtualBox Python Support : This allows you to control the hypervisor by the Python programming language. However, we don't build VMs very often, so we always create and manage the VM manually.
- VirtualBox Host-Only Networking: In this mode, the emulated LAN that the guest OS sees does not connect to the physical LAN on the host OS. If you use this option, you can access only the host machine (and any other guest OSes that are running).

CREATING THE VM

Now it's time to build the virtual machine. First, start the "VirtualBox" program. You will get a dialog box titled "Oracle VM VirtualBox Manager." Note that this is not the VM itself; rather, it is a tool used to create or modify VM environments. Pressing the "new" button will start a wizard that will ask several questions.

- The "name" of the VM will be used to create the name of the VM's settings file.
- The OS should match with the operating system you will be installing later. This step does not install an OS; instead, it prepares the hypervisor to work with the OS better.
- The amount of "base memory" is the amount of RAM that you will dedicate to the VM's use (advertised to the guest OS). The more RAM you dedicate to the VM, the larger the FPGA design you can run at a good speed without having the synthesizer or place-and-route start swapping

	Microsoft Virtual PC	Sun VirtualBox	VMware server
Cost	Provided with Windows 7 Professional and Enterprise editions (but must be downloaded)	Free under GPL version 2. Integration and control extensions may be available for a fee.	One free license. Additional licenses and integration/control extensions available for a fee.
Host OS	Windows 7 and XP, if CPU supports it (may be a patch if HW does not support it)	Windows XP and newer; various flavors of Linux	Windows XP and newer; various flavors of Linux
VHD compatibility	Cannot use VirtualBox or VMware VHDs	Can use Microsoft Virtual PC or VMware VHDs	Can use Microsoft Virtual PC or VirtualBox VHDs
Can run across Remote Desktop (RDP)	There are problems with the mouse.	Nothing noted	Nothing noted

Table 2 – Characteristics of some popular VMs

(which slows the process down tremendously). XP has a limit of just under 4 Gbytes of RAM that it can use, so even if the host machine has 16 Gbytes, you should dedicate only about 3.5 Gbytes to the XP guest.

- The "Virtual Hard Disk" pages address the common case of the VM having a single emulated hard drive, from which it boots. Don't be afraid to make this disk large, because not only will you have to install the guest OS on it, but also all of your FPGA tools. I suggest you use the "dynamically expanding storage" feature to set a maximum size; just because you tell it you want a 200-Gbyte hard drive, the file it creates won't be a full 200 Gbytes until it needs all of that space.
- You are now presented with the "Settings" page. This is a dialog box that allows you to change various aspects of the VM. Note that you cannot make many changes unless the VM is in the "powered off" state (not a problem right now because you haven't started it yet after creating it).
- Verify that the boot order includes the CD drive, because the next step is to put an operating system on the VHD.

INSTALLING THE GUEST OS

After obtaining the appropriate XP install CD, slide it into the CD drive on the host PC and start the VM. A dialog box will tell you about how you interact with the VM's keyboard and mouse. The key is to remember that when the VM is running and has "captured" the mouse and keyboard, interaction is sent to the VM, not the host machine. To return control to the host machine, you must press a "keyboard uncapture" key (or multiple keys). The default for VirtualBox is the Control key to the right of the space bar. VirtualBox displays the "uncapture" key except when it is in full-screen mode. You might want to write the key down, as it is really frustrating to be "locked" into the VM's screen because you forgot how to get out.

Once you dismiss this dialog box, you will encounter the "first-run" wizard. Just click through, answering the questions (choosing what physical CD drive on your host machine will be the CD drive for the guest OS). After that, the steps for installing the guest OS are identical with what you do when installing on a hardware PC; just follow the prompts.

When you initially create the VM, you should probably install the same Windows service packs and patches that you have on the machine you want to archive.

Important note: Just as each PC must have a license from Microsoft to run XP or Windows 7 (for example), so each VM must have its own license. The license for the host machine has no relationship to the VM's guest OS. An exception is when the host OS is Windows 7 Enterprise or Professional and the VM is Microsoft's Windows Virtual PC. In this case, Microsoft provides a special version of XP for free.

INSTALLING AND LICENSING THE FPGA TOOLS

You can install the Xilinx ISE design suite across the Internet or from media (depending on what is available). If you are using node-locked licenses, be aware that each VM has a different MAC address (although you may change it using the advanced settings in the Network setup). Note, however, that nasty things can happen if two machines (even if one of them is virtual) are running at the same time with the same MAC on the same subnet.

If you use network licensing (lm_manager), you may need to contact Xilinx when using older tools.

Unless you do extra work, the guest OS is not "joined" to a domain. In an enterprise environment, that makes it a

bit harder to share files and folders. But it's not a lot more difficult; just map a network drive as a "different user," then transfer files with normal drag-and-drop or copy-and-paste methods.

Since the idea is to archive a completely working version of the design environment, it is always good practice to verify the archive is complete. If everything is in place, you should just be able to build your design in the VM as you did outside the VM. We usually find that we have forgotten some folders (like leveraged HDL), and thus iterate a few times trying to make the build work. But once it works, we know that the environment is complete and it will be possible to rebuild the design—with or without changes—in the future.

CHECKING IN THE FILES NECESSARY TO RESTORE THE VM

The exact steps necessary to archive the VHD and the VM environment vary with the different hypervisors. You may need to read documentation (or consult the Web) for your particular VM, but here are the steps we use with VirtualBox:

- Make sure to power off the guest OS. This minimizes the number of files you need to check into the CMS.
- In the VirtualBox Manager, use the menu command File -> Export Appliance.
- We choose the .ova extension, since that keeps everything together. If you choose the more versatile .ovf extension, you need to keep all three files (.mf, .ovf and .vmdk) together, or else it may be difficult to restore the VM.
- The .ova files are somewhat compressed, so we usually don't convert them to .zip files. Some people, however, have reported significant reduction in space by doing this, so you may want to try it.
- Now you can check the .ova file into a CMS. Because of its size, we typically do not "version" this file.

RESTORING THE VM

Again, the exact steps necessary to restore a VHD will vary with the VM, but here is the way we go about restoring a VirtualBox .ova file:

- In the VirtualBox Manager, use the menu command File -> Import Appliance.
- When it finishes, the list of VMs will show the one you just restored. Start it up and you are back with the environment you had when you first archived it.

After you have updated your HDL and ISE files, it is a good idea to check them into a CMS outside of the VHD. Then, power off the VM and create a new .ova file.

GOOD COMPROMISE

Although we do not currently start new designs in a VM, there is no reason that we couldn't (although editing files and building on the host machine is a little more convenient). If we did this, then when we came to the end of a project, it would be easy to create the .ova file and check it in. As a side benefit, if the host machine were to crash, it would be trivial to resume working on the project on a new machine (assuming we had backed up the VirtualBox folder).

For now, using the VM method to archive designs in preparation for making future changes easy seems to be a good compromise between efficiency in creating the design and preparing for the inevitable. I'm sure refinements will come in the future, but at the moment, this seems to be the best method available.

VM buzzwords

Hypervisor	The program that creates the environment for the virtual machine. It provides services like BIOS subroutines, I/O (either emulated or "pass through") and a display, among other things.
Host OS	The operating system that the hardware boots. The hypervisor is started from the host OS.
Guest OS	The operating system running under the hypervisor.
VHD	Virtual hard drive; a file on the host OS that looks like a disk drive to the guest OS.
BIOS	Basic I/O System. Many older programs make direct calls to the BIOS; most guest OSes will make calls to the BIOS.
HAL	Hardware abstraction layer; a replacement for a BIOS.
Settings	Information about the VM, such as the amount of RAM advertised to the guest OS, what emulated devices are available, what VHDs are mounted in the VM, etc.

FPGAs Made Accessible

JTAG-HS1™

Lowest-cost, highest-speed programming solution for Xilinx® devices

- Compatible w/ all Xilinx tools & devices
- Up to 30 Mbits/sec on JTAG bus
- Supports 6 & 14-pin programming headers
- Volume pricing available

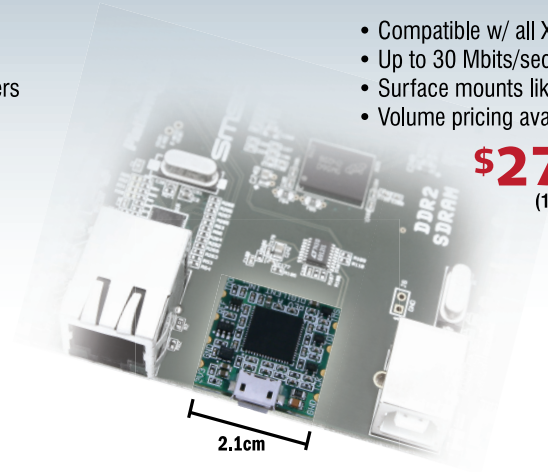
$49.99

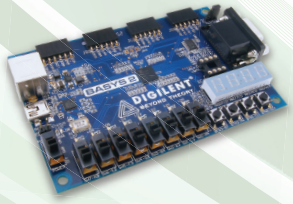
JTAG-SMT1™

Embedded programming for Xilinx® devices

- Compatible w/ all Xilinx tools & devices
- Up to 30 Mbits/sec on JTAG bus
- Surface mounts like any other component
- Volume pricing available

$27.99
(100+)



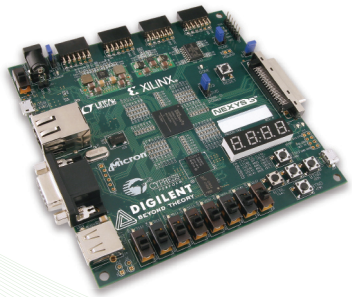


BASYS 2™

The Ultimate FPGA Starter

- Spartan® 3E
- USB-powered
- Ideal first time FPGA users

$49.00
(Student Price)

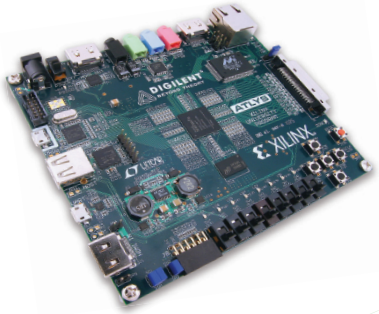


NEXYS 3™

Spartan®-6 power at an entry-level price

- Spartan®-6 LX16
- 32MB PCM & 16MB RAM
- 10 / 100 Ethernet, USB UART
- USB host for mouse & keyboard

$119.00
(Academic Price)



ATLYS™

Advanced video processing w/ Spartan-6 power

- Spartan®-6 LX45
- 128MB DDR2 & 16MB PCM
- Gigabit Ethernet, USB UART
- 4 HDMI ports, AC97 audio

$199.00
(Academic Price)

(509) 334-6306

DIGILENT®
BEYOND THEORY

www.digilentinc.com

Introducing ZYNQ, the new element in processing. Finally, the processor comes together with the FPGA in a fully extensible processing platform called Zynq.™ More intuitive to program in the way you already know. Fully customizable to your requirements. Faster to implement and get to market. As a software engineer, if you know ARM® Cortex,™ you already know Zynq. And if you know Xilinx, you already know this is innovation you can count on. Visit us at www.xilinx.com

XILINX
ZYNQ™

Xilinx Tool & IP Updates

Xilinx is continually improving its products, IP and design tools as it strives to help designers work more effectively. Here, we report on the most current updates to the flagship FPGA development environment, the ISE® Design Suite, as well as to Xilinx® IP, as of July 2011. Product updates offer significant enhancements and new features to three versions of the ISE Design Suite: the Logic, Embedded and DSP editions. Keeping your installation of ISE up to date is an easy way to ensure the best results for your design. Updates to the ISE Design Suite are available from the Xilinx Download Center at www.xilinx.com/download. *For more information or to download a free 30-day evaluation of ISE, visit* www.xilinx.com/ise.

NEW NAVIGATOR

A new application called **Documentation Navigator** allows users to view and manage Xilinx design documentation (software, hardware , IP and more) from one place, with easy-to-use download, search and notification features. To try out the new Xilinx **Documentation Navigator**, now in open beta release, go to the download link at *www.xilinx.com/support*.

ISE DESIGN SUITE: LOGIC EDITION

Front-to-Back FPGA Logic Design

Latest version number: 13.2
Date of latest release: July 2011
Previous release: 13.1
URL to download the latest patch: *www.xilinx.com/download*

Revision highlights:

A newly redesigned user interface for PlanAhead™ and the IP suite improves productivity across SoC design teams in a progression toward true plug-and-play IP that targets Spartan®-6, Virtex®-6 and 7 series FPGAs. In addition, the latest ISE Design Suite provides up to 25 percent better performance on the industry-leading 2 million-logic-cell Virtex-7 2000T device.

PlanAhead design analysis tool: Xilinx has further enhanced the graphical user interface (GUI) to provide an intuitive environment for both new and advanced users. A new clock domain interaction report analyzes timing paths between clock domains. PlanAhead 13.2 also offers the ability to invoke TRACE on post-implementations and localization of tool tips for Japanese and Chinese languages.

Team design: A team design methodology using PlanAhead addresses the challenge of multiple engineers working on a single project by providing a methodology for groups to work in parallel. Building on the design preservation capability made available in ISE Design Suite 12, the team design flow provides additional functionality and lets you lock down early implementation results on completed portions of the design without having to wait for the rest of the design team. This new capability facilitates faster timing closure and timing preservation for the remainder of the design, increasing overall productivity and reducing design iterations.

Xilinx Power Estimator (XPE) and Power Analyzer (XPA): These tools now offer improved power estimation, and XPA features vectorless algorithms for activity propagation.

ISE DESIGN SUITE: EMBEDDED EDITION

Integrated Embedded Design Solution

Latest version number: 13.2
Date of latest release: July 2011
Previous release: 13.1
URL to download the latest patch: *www.xilinx.com/download*

Revision highlights:

All ISE Design Suite editions include the enhancements listed above for the Logic Edition. The following enhancements are specific to the Embedded Edition.

Xilinx Platform Studio (XPS): This software boasts a number of enhancements, including support for the Kintex™ KC705 platform and for single or dual AXI4-based MicroBlaze™ designs. Base System Builder uses a new two-page setup for easier configuration. The Create/Import IP wizard now supports AXI4, AXI-Lite and AXI4-Stream IP.

SDK enhancements: Xilinx has updated the Software Development Kit to Eclipse 3.6.2 and CDT 7.0.2 releases to provide stability and enhancements in this open-source platform. MicroBlaze v8.20a support now features a 512-bit data width for AXI cache interconnects.

IP enhancements: The release includes new AXI PCIe™ and QuadSPI IP, along with improved AXI V6 DDRx read /write arbitration.

EDK overall enhancements: The Embedded Development Kit now offers consistent SDK workspace selection behavior across Project Navigator, Xilinx Platform Studio (XPS) and the SDK.

ISE DESIGN SUITE: DSP EDITION

For High-Performance DSP Systems

Latest version number: 13.2
Date of latest release: July 2011
Previous release: 13.1
URL to download the latest patch: *www.xilinx.com/download*

Revision highlights:

All ISE Design Suite editions include the enhancements listed above for the Logic Edition. Specific to the DSP Edition, version 13.2 offers hardware co-simulation support for the Kintex KC-705 platform.

In addition, the CIC Compiler offers an input width of 24 bits and the new Divider Generator features operand support to 64 bits.

XILINX IP UPDATES

Name of IP: ISE IP Update 13.2
Type of IP: All

Targeted application: Xilinx develops IP cores and partners with third-party IP providers to decrease customer time-to-market. The powerful combination of Xilinx FPGAs with IP cores provides functionality and performance similar to ASSPs, but with flexibility not possible with ASSPs.

Latest version number: 13.2
Date of latest release: July 2011
URL to access the latest version: *www.xilinx.com/download*

Informational URL:

www.xilinx.com/ipcenter/coregen/updates_13_2.htm

Installation instructions:

www.xilinx.com/ipcenter/coregen/ip_update_install_instructions.htm

Listing of all IP in this release:

www.xilinx.com/ipcenter/coregen/13_2_datasheets.htm

Revision highlights:

In general, any IP core for Virtex-7, Kintex-7, Virtex-6 and Spartan-6 device families will now support the AXI4 interface. Older production versions of IP will continue to support the legacy interface for the respective core on Virtex-6, Virtex-5 and Virtex-4, and Spartan-6 and Spartan-3, device families only. Starting with release 13.1, all ISE CORE Generator™ IP supports Kintex-7 and Virtex-7 devices. The 13.2 release adds the following new IP cores.

AXI infrastructure IP: Several new cores make it easy to create designs using AXI4, AXI4-Lite or AXI4-Stream interfaces.

- AXI Interconnect LogiCORE™ IP v1.03 connects one or more AXI4 memory-mapped master devices to one AXI4 slave device. The AXI interconnect supports address widths from 12 to 64 bits with interface data widths of 32, 64, 128, 256, 512 or 1,024 bits. Users can now implement a DDR2 or DDR3 SDRAM multiport memory controller using MIG and AXI interconnect IP from CORE Generator.

- AXI Bus Functional Model (BFM) v1.9, created for Xilinx by Cadence Design Systems, enables Xilinx customers to verify and simulate communication with AXI-based IP they are developing. The AXI BFM IP in CORE Generator delivers testbench and script examples that demonstrate the use of the BFM test-writing APIs for AXI3, AXI4, AXI4-Lite and AXI4-Stream masters and slaves.

- AXI Direct Memory Access (DMA) LogiCORE IP v4.00 provides a flexible interface for transferring packet data between system memory (AXI4) and AXI4-Stream target IP. The AXI DMA provides optional support of scatter/gather for offloading processor management of DMA transfers and descriptor queuing for prefetching transfer descriptors to enable uninterrupted transfer requests by the primary DMA controllers.

Audio, video and image processing IP: The Video Timing Controller v3.0 now supports the AXI4-Lite interface and Virtex-7 and Kintex-7 device families. The Triple-Rate SDI IP has added Spartan-6 support.

Additional IP supporting AXI4 interfaces: Xilinx has updated the latest versions of CORE Generator IP with production AXI4 interface support. For more detailed support information, see *www.xilinx.com/ipcenter/axi4_ip.htm*.

For general information on Xilinx AXI4 support see *www.xilinx.com/axi4.htm*.

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 application notes and a white paper.

FEATURED WHITE PAPER – WP392: XILINX AGILE MIXED-SIGNAL SOLUTIONS

http://www.xilinx.com/support/documentation/white_papers/wp392_Agile_Mixed_Signal.pdf

The industry-leading 28-nanometer 7 series of advanced FPGAs has greatly expanded the capabilities of the integrated analog subsystem over previous generations of FPGA families. The analog subsystem in the Xilinx® 7 series is called the XADC and includes dual, independent, 1-Megasample/second (MSPS), 12-bit analog-to-digital converters (ADCs), along with a 17-channel analog multiplexer front end. By closely integrating the XADC with FPGA logic, Xilinx has delivered the industry's most flexible analog subsystem. This novel combination of analog and programmable logic is called Agile Mixed Signal.

Pairing the XADC together with the programmable logic enables system designers to easily eliminate a wide range of mixed-signal devices from their products, including "housekeeping" analog functions such as power monitoring and management; supervisors, voltage monitors and sequencers; thermal management; system monitors and control; single and multichannel ADCs; and touch sensors. The savings in cost, board space and I/O pins can be significant—especially for designs with area and cost constraints or those that ship in high volumes. Additional benefits of the integrated solution include improvements in failure-in-time (FIT) rates, simplified inventory management and elimination of potential end-of-life issues for mature mixed-signal devices.

This white paper by Anthony Collins and Robert Bielby provides an introduction to the benefits and features of the XADC and Agile Mixed Signal solutions implemented with Artix™-7, Kintex™-7 and Virtex®-7 FPGAs, and the Zynq™-7000 Extensible Processing Platform (EPP).

XAPP875: DYNAMICALLY PROGRAMMABLE DRU FOR HIGH-SPEED SERIAL I/O

http://www.xilinx.com/support/documentation/application_notes/xapp875.pdf

Multiservice optical networks today require transceivers that can operate over a wide range of input data rates. High-speed serial I/O has a native lower limit for operating data rates, preventing easy interfacing to low-speed client signals. The noninteger data recovery unit (NI-DRU) that Paolo Novellini and Giovanni Guasti present in this application note is specifically designed for RocketIO™ GTP and GTX transceivers in Virtex-5 LXT, SXT, TXT and FXT platforms and consists of lookup tables (LUTs) and flip-flops. The NI-DRU extends the lower data rate limit to 0 Mbits/second and the upper limit to 1,250 Mbps, making embedded high-speed transceivers the ideal solution for true multirate serial interfaces.

The NI-DRU's operational settings (data rate, jitter bandwidth, input ppm range and jitter peaking) are dynamically programmable, thus avoiding the need for bitstream reload or partial reconfiguration. Operating on a synchronous external reference clock, the NI-DRU supports fractional oversampling ratios. Thus, only one BUFG is needed, independent of the number of channels being set up, even if all channels are operating at different data rates.

Given the absence of a relationship between the reference clock and incoming data rate, two optional barrel shifters ease the interfacing of the NI-DRU with an external FIFO or with any required decoder. The first barrel shifter has a 10-bit output that can be easily coupled to an 8b10b or 4b5b decoder (neither of which is included in the accompanying reference design). The second barrel shifter has a 16-bit output and is specifically designed for 8-bit protocols, such as Sonet/SDH. The user can design other barrel shifters.

XAPP459: ELIMINATING I/O COUPLING EFFECTS WHEN INTERFACING LARGE-SWING SINGLE-ENDED SIGNALS TO USER I/O PINS ON SPARTAN-3 FAMILIES

http://www.xilinx.com/support/documentation/application_notes/xapp459.pdf

The Spartan®-3 families, consisting of Spartan-3, Spartan-3E and Extended Spartan-3A devices, support an exceptionally robust and flexible I/O feature set, such that they can easily meet the signaling requirements of most applications. You can program user I/O pins of these families to handle many different single-ended signal standards.

The standard single-ended signaling voltage levels are 1.2V, 1.5V, 1.8V, 2.5V and 3.3V. But in a number of applications, it is desirable to receive signals with a greater voltage swing than user I/O pins ordinarily permit. The most common use case is receiving 5V signals on user I/O pins that are powered for use with one of the standard single-ended signaling levels. This type of large-swing signal might be received by design or might be applied to the user I/O unintentionally from severe positive or negative overshoot, which can occur regardless of the programmed "direction" of a user I/O pin.

This application note by Eric Crabill describes ways to receive large-swing signals by design. In one solution (and in the general case of severe positive or negative overshoot), parasitic leakage current between user I/O in differential pin pairs might occur, even though the user I/O pins are configured with single-ended I/O standards. The application note addresses the parasitic leakage current behavior that occurs outside the recommended operating conditions.

XAPP486: 7:1 SERIALIZATION IN SPARTAN-3E/3A FPGAS AT SPEEDS UP TO 666 MBPS

http://www.xilinx.com/support/documentation/application_notes/xapp486.pdf

Spartan-3E and Extended Spartan-3A devices are used in a wide variety of applications requiring 7:1 serialization at speeds up to 666 Mbits/second. This application note targets Spartan-3E/3A devices in applications that require 4-bit or 5-bit transmit data bus widths and operate at rates up to 666 Mbps per line with a forwarded clock at one seventh the bit rate. This type of interface is commonly used in flat-panel displays and automotive applications. (Associated receiver designs are discussed in XAPP485, "1:7 Deserialization in Spartan-3E/3A FPGAs at Speeds Up to 666 Mbps," *http://www.xilinx.com/support/documentation/application_notes/xapp485.pdf*.)

These designs are applicable to Spartan-3E/3A FPGAs and not to the original Spartan-3 device. The design files for this application note target the Spartan-3E family, but the Extended Spartan-3A family also supports the same design approach.

Two versions of the serializer design are available. In the Logic version, the lower-speed system clock and the higher-speed transmitter clock are phase-aligned. The FIFO version, for its part, uses a block RAM-based FIFO memory to ensure there is no phase relationship requirement between the two clocks. Both versions use a transmission clock that is 3.5 times the system clock along with double-data-rate (DDR) techniques to arrive at a serialization factor of seven. This is done both to keep the internal logic to a reasonable speed and to ensure that the clock generation falls into the range of the digital frequency synthesizer (DFS) block of the Spartan-3E FPGA.

The maximum data rate for the Spartan-3E FPGA is 622 Mbps for the -4 speed grade and 666 Mbps for the -5 speed grade. The maximum data rate for the Spartan-3A FPGA is 640 Mbps for the -4 speed grade and 700 Mbps for the -5 speed grade.

The limitation in both devices is the maximum speed of the DFS block in Stepping 1 silicon.

XAPP1026 (UPDATED FOR AXI4): LIGHTWEIGHT IP (LWIP) APPLICATION EXAMPLES

http://www.xilinx.com/support/documentation/application_notes/xapp1026.pdf

This application note explains how to use lightweight IP (lwIP), an open-source TCP/IP networking stack for embedded systems, to develop applications on Xilinx FPGAs. The Xilinx Software Development Kit (SDK) supplies lwIP software customized to run on Xilinx embedded systems containing either a PowerPC® or a MicroBlaze™ processor.

Focusing solely on the MicroBlaze, authors Stephen MacMahon, Nan Zang and Anirudha Sarangi describe how to utilize the lwIP library to add networking capability to an embedded system. In particular, they take you through the steps for developing four applications: echo server, Web server, TFTP server and receive-and-transmit throughput tests. The authors have updated this application note for the AX14 interface. The document includes PLB- and AXI4-based reference systems for the Xilinx ML605, SP605 and SP601 FPGA Starter Kit boards.

Xpress Yourself in Our Caption Contest

DANIEL GUIDERA

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 cartoon by Daniel Guidera, showing how easily a laboratory can morph into a beach. The image might inspire a caption like "One Friday afternoon, Fred made good on his idea for the ultimate staycation."

Send your entries to *xcell@xilinx.com*. Include your name, job title, company affiliation and location, and indicate that you have read the contest rules at *www.xilinx.com/xcellcontest*. After due deliberation, we will print the submissions we like the best in the next issue of *Xcell Journal*. The winner and two runners-up will each receive an Avnet Spartan®-6 LX9 MicroBoard, an entry-level development environment for evaluating the Xilinx® Spartan®-6 family of FPGAs (*http://www.xilinx.com/products/boards-and-kits/AES-S6MB-LX9.htm*).

The deadline for submitting entries is 5:00 pm Pacific Time (PT) on October 1, 2011. So, put down your sun block and get writing!

TIM O'CONNELL, an engineer at Quasonix, LLC, won a shiny new Xilinx SP601 evaluation board for this caption for the scene of the elephant in the boardroom from Issue 75 of *Xcell Journal*:

DANIEL GUIDERA

"He only got moved to management because his soldering is sloppy and he won't come near a mouse. It's kind of the 'you know what' in the room."

Congratulations as well to our two runners-up:

"I heard the new guy works for peanuts."

– Mike Hughes, product development engineer, Analog Devices, Inc.

"I don't want to say anything but...when they said we were going to use an FPGA, I didn't think they meant a Fancy Presentation Giant Animal."

– Richard Otte, programmer, West Side Road Inc.

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 October 1, 2011. Official rules available online at *www.xilinx.com/xcellcontest*. Sponsored by Xilinx, Inc. 2100 Logic Drive, San Jose, CA 95124.

XILINX

SYNOPSYS®
Predictable Success

Now Available!

The industry's first methodology manual for FPGA-based prototyping of SoC Designs

FPGA-Based Prototyping Methodology Manual:
Best Practices in Design-for-Prototyping

Go to:
http://www.synopsys.com/fpmm

- Purchase the book or download a free ebook version
- Become involved with the online community for prototypers

XILINX®

Twice the performance. Half the power.

Innovate without compromise with the Xilinx 7 series FPGAs. The new 7 series devices are built on the only unified architecture to give you a full range of options for making your concepts a reality. Meet your needs for higher performance and lower power. Speed up your development time with next-generation ISE® Design Suite. And innovate with the performance and flexibility you need to move the world forward.

www.xilinx.com/7

ARTIX 7™
LOWEST SYSTEM COST

KINTEX 7™
BEST PRICE / PERFORMANCE

VIRTEX 7®
HIGHEST SYSTEM PERFORMANCE

2011 WINNER EE Times ACE AWARDS

PN 2487