Issue 60
Second Quarter 2007

Xcell journal

THE AUTHORITATIVE JOURNAL FOR PROGRAMMABLE LOGIC USERS

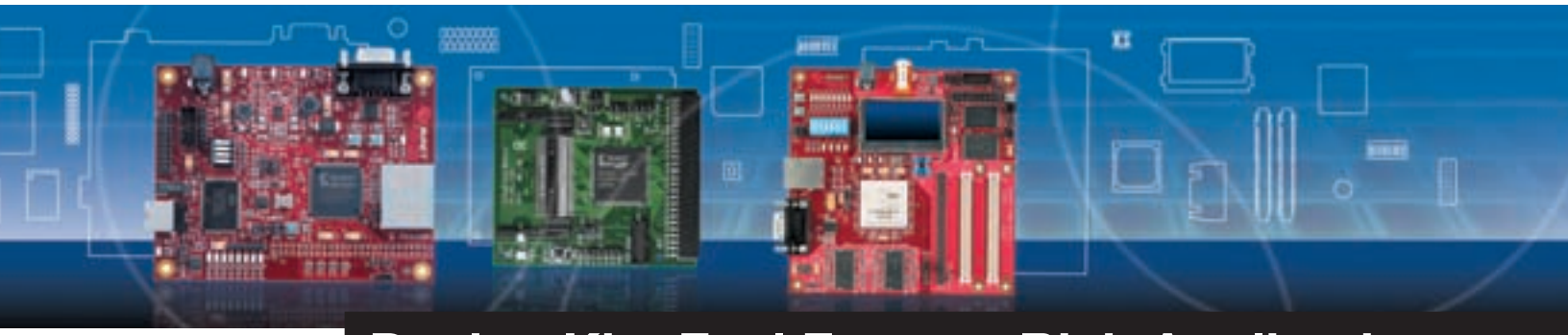
Targeting the Edge of the Network

INSIDE

XILINX®

www.xilinx.com/xcell/

Support Across The Board.™



Design Kits Fuel Feature-Rich Applications

Build your own system by mixing and matching:

- Processors
- FPGAs
- Memory
- Networking
- Audio
- Video
- Mass storage
- Bus interface
- High-speed serial interface

Available add-ons:

- Software
- Firmware
- Drivers
- Third-party development tools

Avnet Electronics Marketing designs, manufactures, sells and supports a wide variety of hardware evaluation, development and reference design kits for developers looking to get a quick start on a new project.

With a focus on embedded processing, communications and networking applications, this growing set of modular hardware kits allows users to evaluate, experiment, benchmark, prototype, test and even deploy complete designs for field trial.

By providing a stable hardware platform that enhances system development, design kits from Avnet Electronics Marketing help original equipment manufacturers (OEMs) bring differentiated products to market quickly and in the most cost-efficient way possible.

For a complete listing of available boards, visit **www.em.avnet.com/drc**

XILINX®

AVNET® electronics marketing

Avnet Green Initiative

Enabling success from the center of technology™

1 800 332 8638
em.avnet.com

IS YOUR CURRENT FPGA DESIGN SOLUTION HOLDING YOU BACK?

FPGA Design | Ever feel tied down because your tools didn't support the FPGAs you needed? Ever spend your weekend learning yet another design tool? Maybe it's time you switch to a truly vendor independent FPGA design flow. One that enables you to create the best designs in any FPGA. Mentor's full-featured solution combines design creation, verification, and synthesis into a vendor-neutral, front-to-back FPGA design environment. Only Mentor can offer a comprehensive flow that improves productivity, reduces cost and allows for complete flexibility, enabling you to always choose the right technology for your design. To learn more go to **mentor.com/techpapers** or call us at **800.547.3000.**

DESIGN FOR MANUFACTURING + INTEGRATED SYSTEM DESIGN
ELECTRONIC SYSTEM LEVEL DESIGN + FUNCTIONAL VERIFICATION

Mentor Graphics®

THE EDA TECHNOLOGY LEADER

Low-Power Transceivers

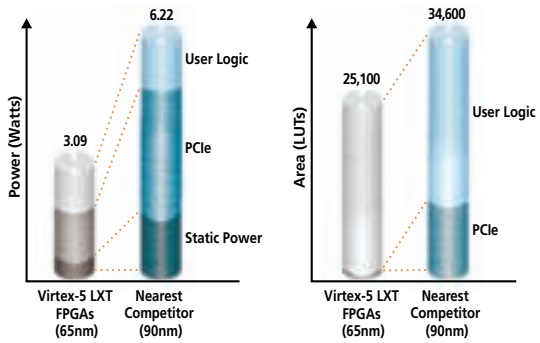
Ultimate Connectivity . . .

VIRTEX 5

Reduce serial I/O power, cost and complexity with the world's first 65nm FPGAs.

With a unique combination of up to 24 low-power transceivers, and built-in PCIe™ and Ethernet MAC blocks, Virtex-5 LXT FPGAs get your system running fast. Whether you are an expert or just starting out, only Xilinx delivers this complete solution to simplify high-speed serial design.

Power consumption and area required to implement a typical design including 8-lane PCIe endpoint



5VLX30T vs. 2SGX60D. Target Frequency = 200 MHz. Worst-case process. 25K LUTs, 17K Flip-Flops,1 Mbit On-Chip RAM, 64 DSP Blocks, 128 2.5V I/Os. Based on Xilinx tool v8.2 and competitor tool v6.0.1

Lowest-power, most area-efficient serial I/O solution

RocketIO™ GTP transceivers deliver up to 3.2 Gbps connectivity at less than 100 mW to help you beat your power budget. The embedded PCI Express® endpoint block ensures easy implementation and reduced development time. Embedded Ethernet MAC blocks enable a single-chip UNH-verified implementation. And the Xilinx solution is fully supported by development tools, design kits, IP, characterization reports, and more.

Visit our website today, view the Webcast, and order your free eval CD to give your next design the ultimate in connectivity.

XILINX®

www.xilinx.com/virtex5

The Ultimate System Integration Platform

Xcell journal

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

EDITOR Charmaine Cooper Hussain

ART DIRECTOR Scott Blair

DESIGN/PRODUCTION Teie, Gelwicks & Associates
1-800-493-5551

ADVERTISING SALES Dan Teie
1-800-493-5551

TECHNICAL COORDINATOR Kevin Kitagawa

INTERNATIONAL Dickson Seow, Asia Pacific
dickson.seow@xilinx.com

Andrea Barnard, Europe/
Middle East/Africa
andrea.barnard@xilinx.com

Yumi Homura, Japan
yumi.homura@xilinx.com

SUBSCRIPTIONS All Inquiries
www.xcellpublications.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/

Xcell Journal Wins Prestigious LACP 2006 Inspire Award

Nothing distinguishes your work and brings more integrity to your magazine than recognition from your peers.

Recently, the League of American Communications Professionals (LACP) honored issue 59 of *Xcell Journal* as the Bronze Winner in the 2006 Inspire Awards Newsletter & Magazine Competition, "Overall" category. The competition was judged by a field of communication professionals, a forum within the public relations industry that facilitates discussion of best-in-class practices within the profession.

"The *Xcell Journal* entry was remarkable in light of tremendous competition," said Christie Kennedy, LACP Managing Director. "We received more than 425 entries for the 2006 Inspire Awards, comprising newsletters and magazines from seven countries."

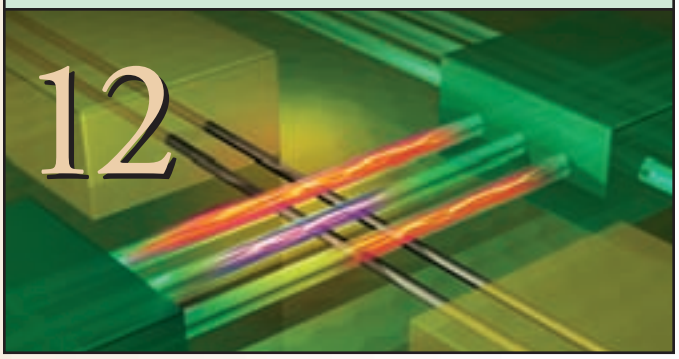
Xcell Journal was judged in several categories, including first impression, artwork, readability, creativity, message clarity, variety of features, audience focus, perceived relevance, and ease of navigation. In the end, the magazine received a score of 92 out of a possible 100 points, resulting in a "Superb – among the very best judged" rating.

"We're very proud that *Xcell Journal* has been selected as a winner of this year's LACP Inspire Awards," said Sandeep Vij, vice president of worldwide marketing at Xilinx. "*Xcell Journal* plays an important role in our overall marketing strategy. We work hand-in-hand with our staff and partners to develop editorial content that is relevant, compelling, and strategically focused so that the magazine continues to provide maximum value to our readers."

The global *Xcell* Publications team is setting the industry standard for innovative custom digital, online, and print publications. I'd like to personally thank all of the authors – from Xilinx, our partners, and our customers – who have contributed their time and effort over the years to make *Xcell Journal* an award-winning publication for programmable logic users around the world.

Forrest Couch
Publisher

POWER OPTIMIZATION



The Power of FPGA Architectures
The present and future of low-power FPGA design.

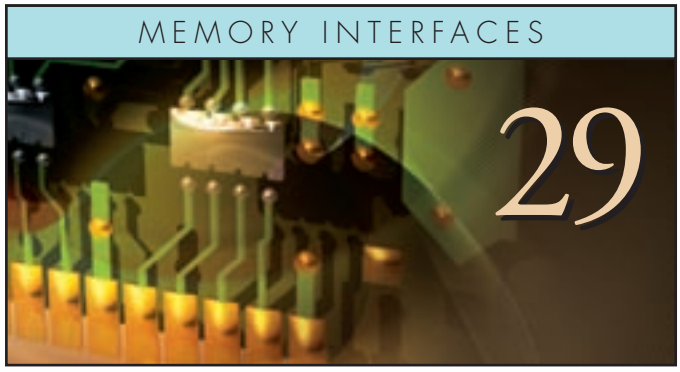
SECURITY

Low-Cost Security Solutions with Spartan-3A and Spartan-3AN Platforms
Could your low-cost FPGA design be more secure?

NON-VOLATILITY

Inkjet Power Unleashed
The Spartan-3AN FPGA activates digital ubiquitous marking and printing.

MEMORY INTERFACES



Implementing Low-Cost DDR2 Interfaces with Spartan-3A FPGAs
Xilinx provides complete memory interface solutions to help you get to market faster.

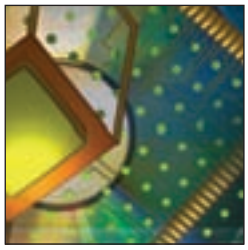
Cover Story

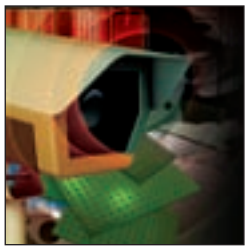
10

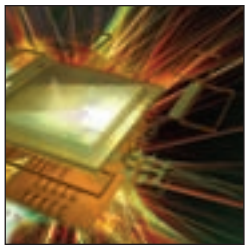
Programmable Logic in High-Volume Applications
Spartan FPGAs and CoolRunner-II CPLDs meet the security, differentiation, and flexibility needs of the high-volume market.

SECOND QUARTER 2007, ISSUE 60

Xcell journal







Xilinx PLDs: On the Go and On the Road

The broad adoption of high-volume solutions in CY06 results in record growth.

by Wim Roelandts
CEO and
Chairman of the Board
Xilinx, Inc.

In 2006 Xilinx experienced phenomenal success in the consumer and automotive segments, with a record 40% growth. Our success in these highly competitive markets can be attributed to broad customer adoption of our Spartan™ FPGAs, CoolRunner™ CPLDs, and Xilinx® Automotive (XA) product lines in high-volume applications.

We began our market diversification efforts in the late 1990s, when close to 80% of PLD revenues were generated from the communications sector. We set our sights on the consumer market with the introduction of Spartan Generation FPGAs (1998) and CoolRunner CPLDs (1999).

At the time, FPGAs in high-volume applications were a very rare commodity. This was a market typically served by standard, fixed-function semiconductor devices such as ASSPs and ASICs. FPGAs and CPLDs were considered too expensive in comparison to ASSPs and ASICs on a cost-per-unit basis to have an impact in the cost-sensitive consumer marketplace.

Xilinx believed otherwise. We exploited Moore's Law and increasingly finer semiconductor manufacturing process geometries to dramatically drive down our per-unit cost. At the same time, global competition quickened the pace at which new products hit the market, so the concept of time to market became increasingly important to consumer product manufacturers. As a result, product developers focused on a business dynamic that saw the majority of their profit and market share being made in a short period of time at and shortly after product launch. The inherent flexibility of our FPGAs and CPLDs provided an added time-to-market advantage over the long development time of ASICs and ASSPs.

As advanced process technology continues to drive down the cost of PLDs, we are becoming more and more competitive in high-volume consumer applications, especially in digital displays and consumer handsets. In the last quarter of CY 2006, we shipped more than one million devices in a single month to one of our top handset manufacturers.

In 2004, we continued our diversification efforts by pursuing the automotive market. We introduced the industry's first PLD line specifically developed to meet the stringent requirements of the automotive industry – the Xilinx Automotive (XA) family. Since that time, we have continued to make inroads in the automotive market. In 2006, a leading luxury vehicle debuted with 18 devices that enable infotainment and driver-assistance functions.

Today, Xilinx high-volume products represent 35% of the company's overall revenues, a compound annual growth rate of 38%.

The Year 2007 and Beyond

Xilinx pioneered the industry's transformation towards a focus on vertical markets and engaging with customers at the system architecture level. Today, this transformation helps us address our customers' complex design challenges by providing them with innovative, flexible, and compelling solutions that help them achieve their objectives of cost management, time to market, and leadership.

Industry analysts project further growth for the PLD industry in both the consumer electronics and automotive markets. New requirements across the consumer and automotive electronic industry are also forcing the need for flexible architectures that can cope with not only current applications but future and possible unknown features. According to iSuppli, the market for PLDs and ASICs in consumer electronics will grow to more than $5.3 billion by 2010.

To further accelerate the adoption of PLD designs in the consumer marketplace, we must now compete on more than just cost – our customers are demanding more application-specific solutions. In the face of rapidly changing consumer product requirements, we need to deliver a more complete solution. Xilinx has responded by offering domain-optimized PLDs along with high-value hard and soft IP cores and market-specific development platforms to bring a much more value-added and customer-specific set of solutions.

Xilinx continues to win designs in a myriad of emerging applications and markets. Our PLDs can be found in a variety of consumer electronic and automotive applications, including digital displays, mobile phones, PDAs, rear-seat entertainment, and satellite navigation systems. As it did in 2006, the consumer and automotive segments promise further growth for Xilinx in 2007 as these high-volume application design wins move into production.

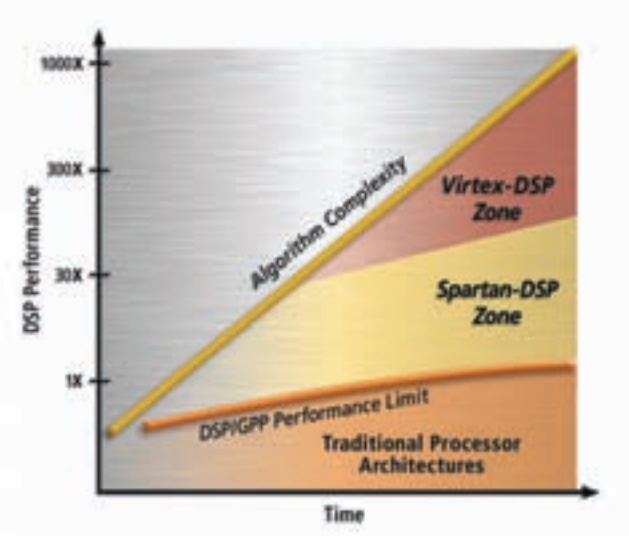
BREAK THROUGH

Now it's your turn to innovate...

XtremeDSP™

Leading the way in high-performance, configurable DSP

Outperform your competition with the high-performance signal processing delivered by XtremeDSP. With the new ***Spartan-DSP*** series, low-cost implementations are now possible without compromising performance. That opens up an exciting world of innovative applications in wireless, video, imaging and more.



Jump start your design with XtremeDSP solutions

- **XtremeDSP Device Portfolio:** *Configurable, scalable, high-performance devices: Spartan-DSP and Virtex-DSP series*
- **XtremeDSP Design Environment:** *Flexible, easy-to-use design flow with Xilinx Signal Processing Libraries, Xilinx System Generator™ for DSP, and AccelDSP™ high-level MATLAB® language-based tool*
- **XtremeDSP Development Tools:** *Application-specific reference designs, development boards, kits and IP blocks*
- **XtremeDSP Support:** *World-class support from ecosystem of partners and dedicated Xilinx teams*

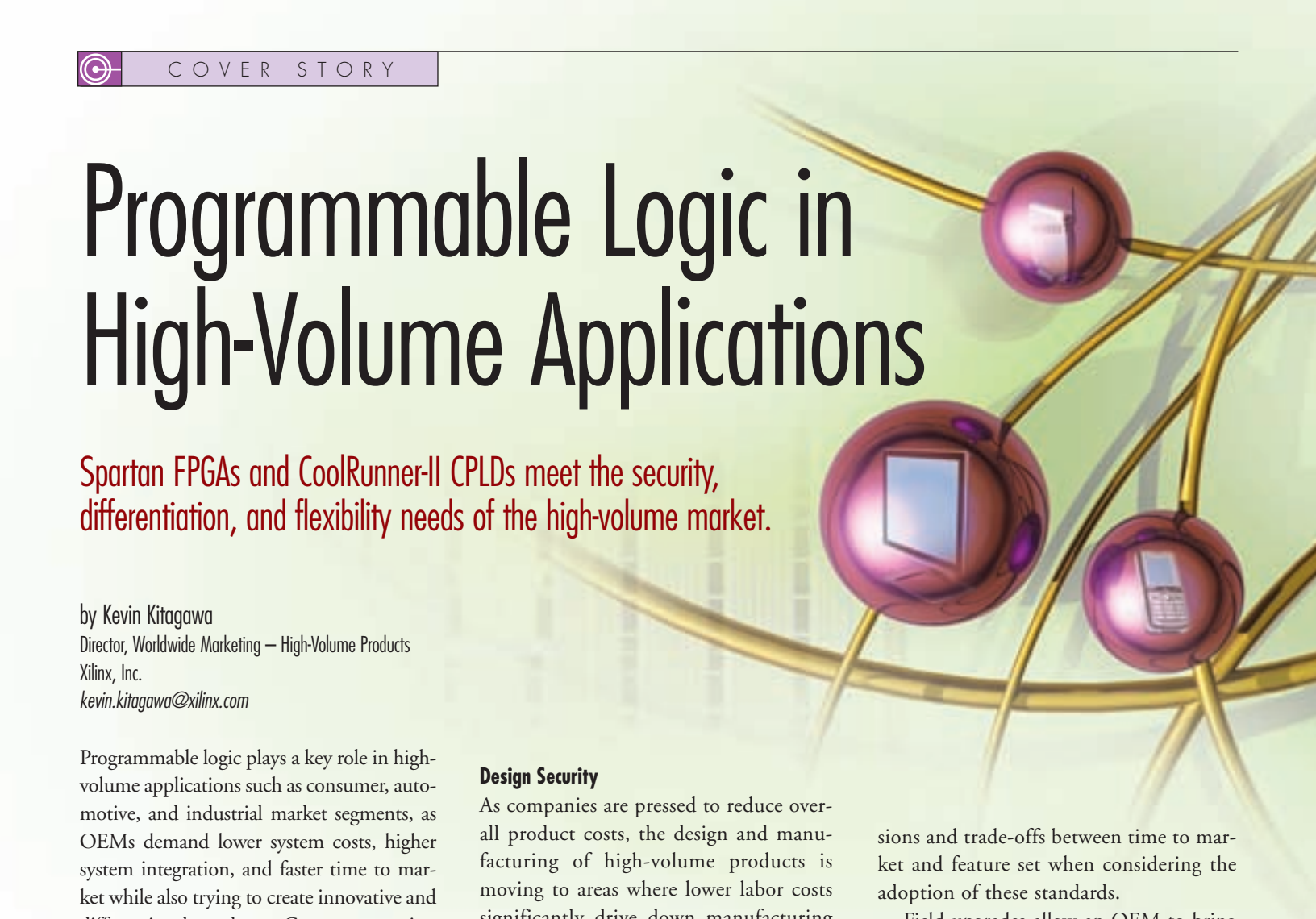
Visit ***www.xilinx.com/dsp*** today, order your FREE evaluation software, and take your next DSP design to the Xtreme.

XILINX®

www.xilinx.com/dsp

Innovation to the Xtreme

Programmable Logic in High-Volume Applications



Spartan FPGAs and CoolRunner-II CPLDs meet the security, differentiation, and flexibility needs of the high-volume market.

by Kevin Kitagawa
Director, Worldwide Marketing – High-Volume Products
Xilinx, Inc.
kevin.kitagawa@xilinx.com

Programmable logic plays a key role in high-volume applications such as consumer, automotive, and industrial market segments, as OEMs demand lower system costs, higher system integration, and faster time to market while also trying to create innovative and differentiated products. Customer requirements also vary widely as OEMs balance cost, performance, and product features to target different end customers. Although ASICs and ASSPs have traditionally played in these markets, the high initial costs and risks associated with ASICs and the inflexibility of ASSPs are not conducive to timely and cost-effective products.

Xilinx has addressed OEM needs by providing domain-optimized platforms such as Xilinx® Spartan™ series FPGAs and CoolRunner™-II CPLDs. The product families are tailored toward very different customer requirements in a wide variety of high-volume market segments. For example, CoolRunner-II products are optimized for battery-operated portable applications that value power consumption, while the non-volatile Spartan-3AN platform is optimized for applications that require high system integration or design security. The flexibility and quick time-to-market advantages of programmable logic devices minimize the risk and maximize the probability of timely and successful products.

Design Security

As companies are pressed to reduce overall product costs, the design and manufacturing of high-volume products is moving to areas where lower labor costs significantly drive down manufacturing costs. Although OEMs have enjoyed the benefits of lower manufacturing costs, they have also experienced new problems, as these same contract manufacturers sought to clone the OEM's designs or overbuild additional units. These unauthorized manufactured devices are lost revenue when sold in the marketplace and can also create additional costs if defective units are returned to the company for service or replacement.

As a result, design security has become essential. Xilinx products offer a wide range of security methods in products uniquely suited for high-volume applications.

Product Differentiation and Flexibility

OEMs are constantly adapting to standards that have evolved for better performance, more flexibility, or to address existing limitations. In addition, the ratification of a standard can take a long time, especially as more and more stakeholders participate in the approval process. OEMs must make hard decisions and trade-offs between time to market and feature set when considering the adoption of these standards.

Field upgrades allow an OEM to bring devices to market with the flexibility to update them after being released to mass production. This extends the device lifecycle tremendously, and without a lot of investment from the OEM. An OEM can now release devices without fear that a newly endorsed standard will render their device obsolete in the marketplace.

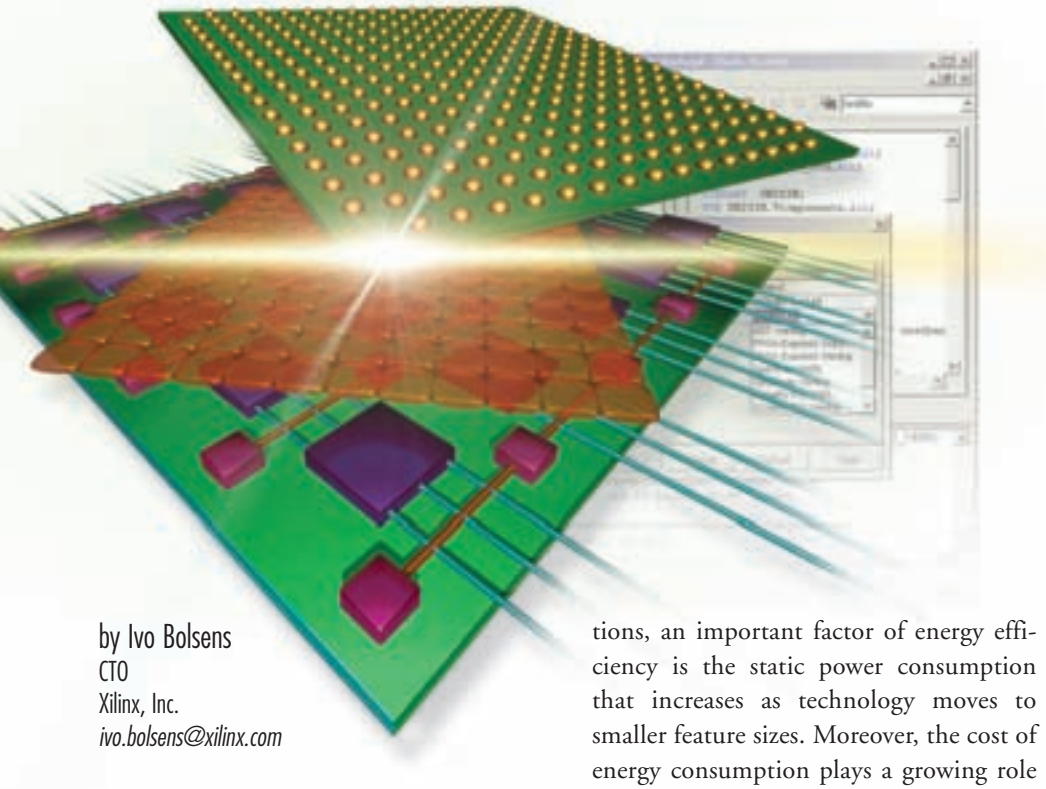
Spartan FPGAs and CoolRunner-II CPLDs allow OEMs to quickly adapt to changes. Decisions such as protocols, buses, and number of interfaces can all be changed to quickly address new market requirements.

Conclusion

The dynamic nature of high-volume applications is causing unprecedented challenges for design engineers. OEMs must deliver products that are not only feature-rich and power-efficient, but cost-efficient and quick to market as well. The comprehensive portfolio of domain-optimized Spartan series FPGA and CoolRunner-II CPLD products gives designers the best choice for their designs.

Energy-Efficient System Design

Power optimization techniques in silicon and software have become mainstream considerations in FPGA system design.



by Ivo Bolsens
CTO
Xilinx, Inc.
ivo.bolsens@xilinx.com

In recent years, power consumption as a figure of merit for FPGAs has grown in importance. Because of the increasing use of FPGAs in high-volume and embedded applications, power dissipation and energy efficiency are at the forefront of the system designer's concerns. Lower power consumption simplifies thermal management and results in smaller (or no) heat sinks, cheaper packages, and lower airflow requirements, which in turn lead to lower cost and more compact system solutions. Lower current requirements simplify power supply design, resulting in fewer system components and less PCB area.

As high-volume applications are becoming more battery operated, the importance of energy efficiency (power consumed over time) is growing to allow for longer battery life. In many applications, an important factor of energy efficiency is the static power consumption that increases as technology moves to smaller feature sizes. Moreover, the cost of energy consumption plays a growing role in the total cost and OPEX (operating expenditure) of end-customer solutions.

It is well known that, in terms of giga floating-point operations per second (GFLOPS)/Watt, the FPGA programmable architecture is more power-efficient than a processor architecture. A high-end Xilinx® Virtex™-5 component (such as an XC5VLX330) can deliver 3 to 12 times more GFLOPS (GFLOPS being 64-bit multiply, multiply/add, or add) and consumes 2 to 3 times less power than a leading-edge processor architecture (such as today's dual-core x86 architectures). Therefore, depending on the application, the high-end Virtex-5 family will deliver 3 to 30 times better GFLOPS/Watt than leading-edge processors. Comparing 32-bit floating-point, fixed-point, or integer computations per Watt would even further benefit the FPGA.

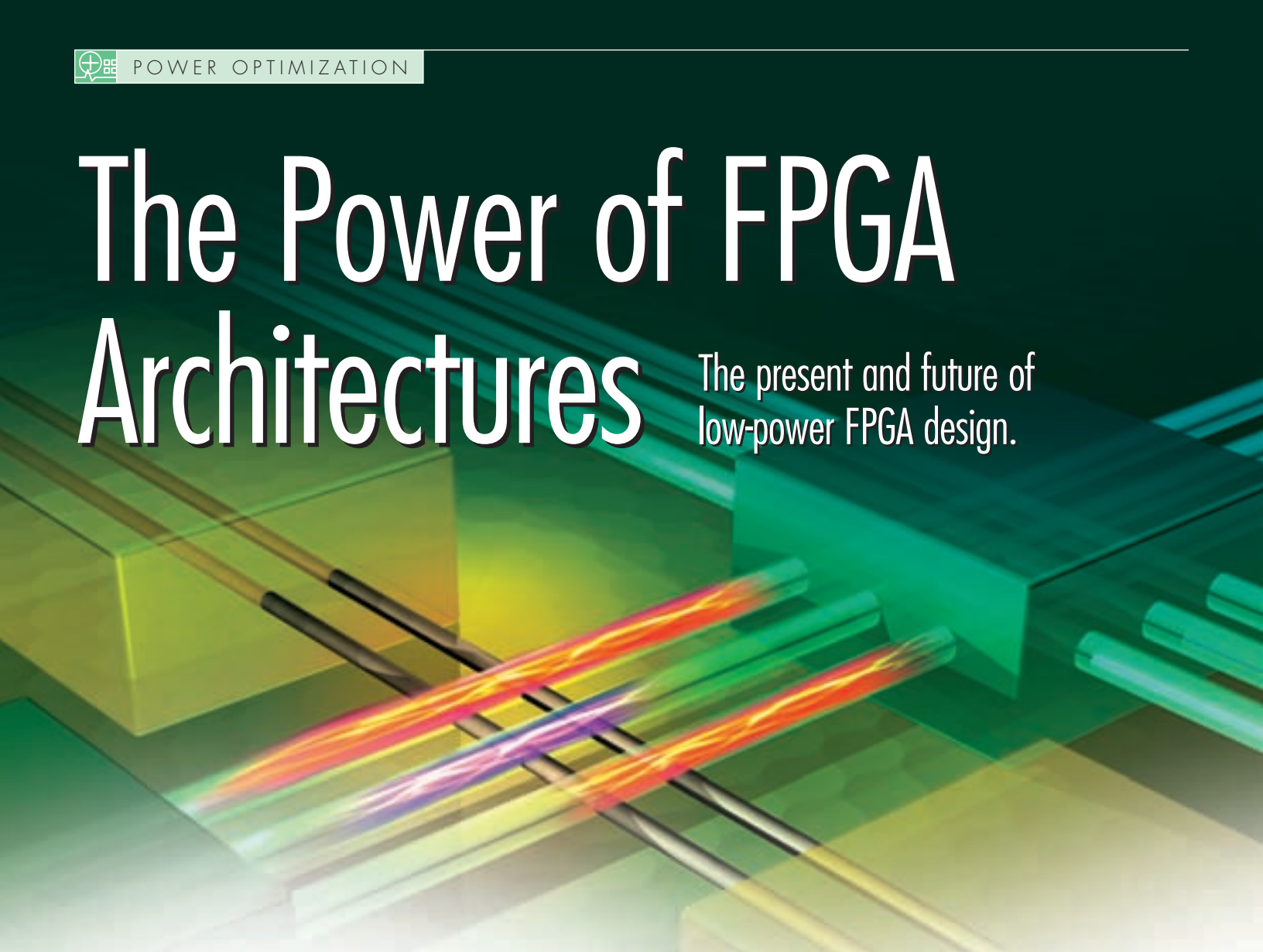
For a given system application, the relentless focus on power optimization has resulted in Virtex-5 devices consuming between 30% to 80% less power than Virtex-4 devices (depending on which resources are being used). In the low-cost Spartan-3 family, the focus has been further on ease of use of the power management and on improving energy efficiency by providing different power-saving modes during periods of inactivity.

In contrast to other figures of merit, such as cost and performance, the efficiency of power consumption cannot be captured by one single comparative number. Different application characteristics and use models can result in different benefits from all of the possible power optimizations. "The Power of FPGA Architectures" in this issue of the *Xcell Journal* demonstrates the effectiveness of design techniques such as sleep modes, power and clock gating, voltage scaling, and how specialized hardware depends heavily on the system application at hand. "Optimizing FPGA Power with ISE Design Tools," also in this issue, highlights the importance of intelligent design tools and sophisticated place and route algorithms that can exploit the characteristics of an application to optimize power consumption of the final implementation on an FPGA.

Designing power-efficient systems into an FPGA encompasses a holistic optimization process that includes an understanding of all system constraints, the selection of the optimal FPGA architecture, and the optimization of the design during all steps in the design flow, from architecture to final place and route. Aggressive scaling and cost reduction will continue to move FPGAs further into higher volume and lower power products. This will result in a constant improvement of FPGA architectures and design tools to meet the requirements of lower power and energy efficiency.

The Power of FPGA Architectures

The present and future of low-power FPGA design.



by Tim Tuan
Staff Research Engineer
Xilinx, Inc.
tim.tuan@xilinx.com

Steve Trimberger
Distinguished Engineer
Xilinx, Inc.
steve.trimberger@xilinx.com

Reducing power consumption in FPGAs delivers numerous benefits such as better reliability, lower cooling cost, simpler power supply and delivery, and longer battery life in portable systems. Designing for low power consumption without compromising performance requires a power-efficient FPGA architecture and good design practices to leverage the architectural features.

In this article, we'll present a brief overview of FPGA power consumption, current low-power features, user choices that impact power, and look at recent low-power research for insight into future trends on power-efficient FPGAs.

Power Components

FPGA power consumption has two components: dynamic power and static power. Dynamic power is dissipated when signals charge capacitive nodes. These capacitive nodes may be inside logic blocks, routing wires in the interconnect fabric, external package pins, or board-level traces driven by chip outputs. Total FPGA dynamic power is the combined power from charging all capacitive nodes.

Static power, on the other hand, has nothing to do with the activity of the circuit. It is dissipated either as transistor leakage current or as bias current. Total static power is the combined total of each transistor's leakage power and all bias currents in the FPGA. As transistor dimensions shrink, dynamic power improves because it depends on the side of driven capacitances. However, static power increases because smaller transistors leak more. As a result, static power is becoming an increasingly large fraction of the overall power consumption of integrated circuits.

Power consumption is highly dependent on supply voltage and temperature, as shown in Figure 1. Reducing the FPGA supply voltage results in a quadratic reduction in dynamic power and an exponential reduction in leakage power. Increasing temperature results in an exponential increase in leakage power. For example, an increase from 85 °C to 100 °C increases leakage power by 25%.

Power Breakdown

Let's examine the breakdown of total FPGA power to understand where most of the power is consumed. FPGA power is design-dependent; namely, it depends on the part family, clock frequency, toggle rate, and resource utilization. In this analysis, we'll use the Xilinx® Spartan™-3

XC3S1000 FPGA as an example and assume that clock frequency is 100 MHz, the toggle rate is 12.5%, and resource utilization is typical as determined by benchmarking many real designs.

Figure 2 shows the breakdown of the XC3S1000 FPGA in terms of active power and standby power. Active power is reported as the power of an active design at high temperature, which comprises both dynamic and static power. Standby power is the power of an idle design, comprising static power at nominal temperatures. Not surprisingly, CLBs make up the largest component in both active power and standby power, but other blocks contribute significant power as well. I/Os and clock circuitries make up a third of total active power, which would be even higher if you used high-powered I/O standards.

Configuration and clock circuitries consume nearly half of total standby power, largely because of bias currents. Therefore, total chip power reduction must come from multiple solutions that address all major power-consuming parts.

Designing for Low Power

Many power-driven design techniques are employed in the design of an FPGA. Because configuration memory cells can make up a third of the transistors in an FPGA, Xilinx uses a low-leakage "midox" transistor in Virtex™ families to reduce leakage in memory cells. Transistors with longer channel length and higher thresholds are also used throughout to reduce static power. Dynamic power consumption is addressed with low capacitance circuits and custom blocks. A multiplier in a DSP block consumes less than 20% of the power of a multiplier built from FPGA fabric. As manufacturing variations can lead to large spreads in leakage current, Spartan-3L FPGAs screen for low-leakage parts to effectively provide a part with 60% less core leakage power.

Beyond what is designed into the FPGA, many design choices also affect FPGA power consumption. Let's examine some of these choices.

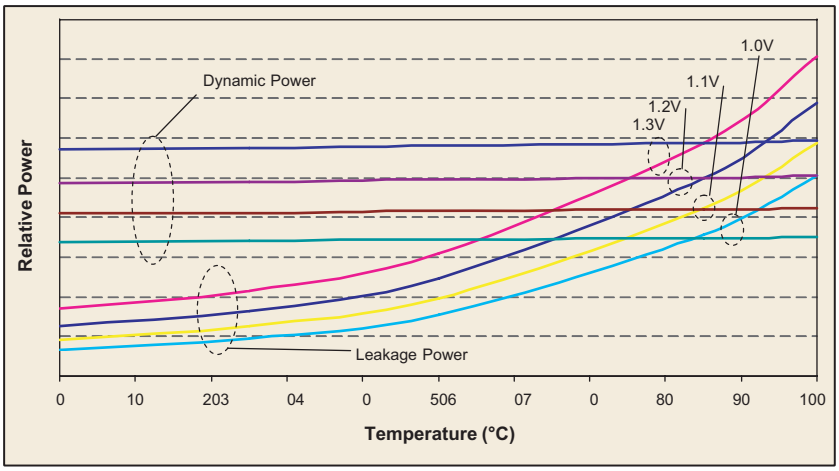


Figure 1 – Power dependency on voltage and temperature

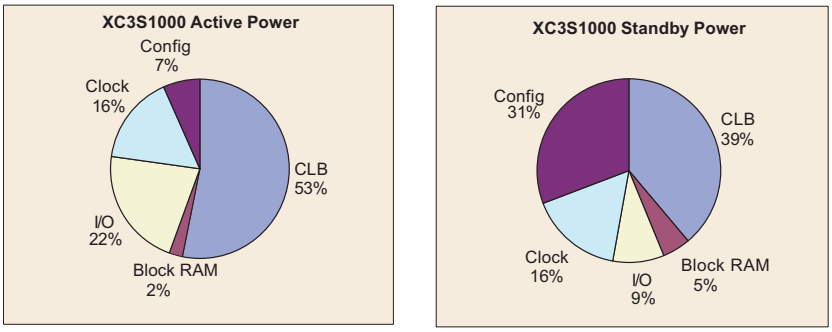


Figure 2 – Breakdown of a Spartan-3 XC3S1000 FPGA's typical power consumption

Power Estimation

A key step in low power design is power estimation. Although the most accurate way to determine FPGA power consumption is through hardware measurements, estimation can help identify high-power modules and is useful for power budgeting early in the design cycle.

Several tools are available to aid power estimation: Xilinx Power Estimator (XPE) and Xilinx Power Analyzer (XPA). XPE gives quick power estimates through a simple user interface. Its estimation is based on high-level statistics such as the number of logic cells, number of block RAMs, and average switching activity. XPA gives detailed power estimates based on simulated switching activity and exact utilization statistics generated post-place and route. Choosing the right tool depends on what design information is available and what level of accuracy is needed.

As shown in Figure 1, some external factors have exponential effects on power consumption; slight changes in environment can cause large changes in estimated power. Exact accuracy is difficult to achieve through power estimation tools, but they nevertheless provide excellent guidance for power optimization by identifying high-power blocks.

Voltage and Temperature Control

As shown in Figure 1, both lower voltage and lower temperature can lower leakage current significantly. Just a 5% lower supply voltage can lead to 10% lower power. Supply voltages can be easily adjusted by changing the power supply configuration. Current FPGAs do not support wide voltage adjustment; the recommended voltage range is typically +/- 5%. Junction temper-

ature can be reduced by using cooling solutions such as heat sinks and airflow. A 20 °C reduction in temperature can lead to a more than 25% reduction in leakage power. Lower temperature also exponentially improves chip reliability. Studies have shown that a 20 °C reduction can lead to a 10x improvement in overall chip life.

Suspend and Hibernate Modes

Spartan-3A FPGAs provide two low-power idle states. In suspend mode, the circuitry on the VCCAUX power supply is disabled to reduce leakage power and eliminate bias currents, reducing static power consumption by more than 40%. Chip configuration and circuit state are retained. Exiting from suspend mode is triggered by asserting an awake pin. The process takes less than 1 ms.

The hibernate mode in Spartan-3A devices allows all power regulators to be switched off to achieve zero power. To restart, the part must be re-powered and reconfigured, which takes tens of milliseconds. While powered off, all I/Os are in a high impedance state. If any I/Os need to be actively driven during hibernate mode, the corresponding I/O bank must remain powered, consuming a small amount of standby power.

I/O Standard Choices

Different I/O standards have considerably different levels of power consumption. You can achieve significant reduction by choosing a lower power I/O standard at the expense of speed or logic utilization. For example, LVDS is a high power consumer, with 3 mA per input pair and 9 mA per output pair. Thus, from a power perspective, LVDS should only be used when it is required by system specifications or when the highest performance is needed.

A lower power, higher performance alternative to LVDS is HSTL or SSTL, but these still consume 3 mA per input. When possible, we recommend LVCMOS inputs instead. Lastly, DCI standards are high power consumers. When connecting to memory devices such as RLDRAM, consider using ODT on the memory and LVDCI on the FPGA to save power.

Embedded Blocks

Using embedded blocks instead of programmable fabric can save a substantial amount of power. Xilinx FPGAs have a number of embedded blocks such as the PowerPC™ hard-core processor, DSP slices, ChipSync™ technology, embedded Ethernet MAC(s), FIFOs, and SRL16. Embedded blocks are custom-designed; hence they are smaller and have less switching capacitance than programmable logic. These blocks have between 5x and 12x lower power than equivalent programmable logic implementations. Using embedded blocks can lead to static power reduction if the design becomes smaller and fits into a smaller part. A potential pitfall is that very simple functions may not be more efficiently implemented using large embedded blocks. This can be easily avoided by checking both implementations using XPE.

Clock Generators

Power considerations in clock generation can save power. Digital clock managers are widely used to generate clocks with different frequencies or phases. However, DCMs consume a non-trivial amount of power off VCCAUX; therefore, their use should be limited when possible. A single DCM can often generate multiple clocks by using multiple outputs such as CLK2X, CLKDV, and CLKFX. This is a lower power solution than using multiple DCMs for the same function.

Block RAM Construction

Multiple block RAMs are often combined to create a single large RAM. How this is done can have significant power implications. The timing-driven method is to access all RAMs in parallel. For example, a 2k x 36 RAM would be constructed out of four 2k x 9 RAMs. The access time of this larger RAM is the same as that of a single block RAM; however, the power consumption per access is that of four block RAMs.

A low-power solution is to construct the same 2k x 36b RAM out of four 512 x 36b RAMs. Each access would be pre-decoded to select one of the four block RAMs to access. Although the access time is increased because of pre-decoding, the power-per-access of the larger RAM is approximately the same as that of a single block RAM.

Low-Power Research

In recent years, Xilinx Research Labs has studied various techniques for low-power FPGA design. These research items are not currently implemented in commercial FPGAs, but they do offer an insight into what FPGAs might look like in the future.

Voltage Scaling

As mentioned earlier, reducing supply voltage is one of the most effective ways to reduce power, and the associated performance degradation is acceptable to many designs that do not need peak performance. However, current FPGAs operate in a small range, and one of the limitations is found in some voltage-sensitive circuits.

At Xilinx Research Labs, we redesigned the CLB circuits to operate at a much lower voltage, enabling a substantial trade-off of performance headroom for lower power. For example, for a 90-nm process, a 200-mV reduction could bring a 40% power reduction at the expense of 25% peak performance; a 400-mV reduction could bring a 70% power reduction at the expense of 55% peak performance.

Fine-Grain Power Switching

One of the unique overheads of programmable logic design is that not all on-chip resources are used in any given design. However, they remain powered and contribute to total power in the form of leakage power. Block-level power switching can shut off power to individual unused blocks. Each block is coupled to the power supply through a power switch. When the switch is closed, the block is functional. When the switch is open, the block is effectively disconnected from the power supply, and so consumes 50-100x less leakage power. The granularity of the power switches may be as small as individual CLBs and block RAMs. In our design, these power switches can be programmable through configuration bitstreams or controlled by the user directly or through an access port. Benchmarking real

designs shows that fine-grain power switching can reduce leakage power by 30%.

Deep-Sleep Mode

One of the key requirements in portable electronics is to consume little or no power when the device is idle. In Spartan-3A FPGAs, you can achieve this by entering hibernate mode, which requires external control, is slow to wake up, and cannot restore the FPGA state. In our design, we dynamically control the fine-grain power switches described previously to power down all internal blocks while leaving the configuration and circuit state storage elements powered. The resulting state is a deep-sleep mode where leakage power is 1%-2% of nominal, the FPGA state is saved, and exiting this mode takes microseconds.

Heterogeneous Fabric

The maximum clock frequency of a circuit is determined by the delays of its timing-critical paths. Non-critical paths can be slower without affecting total chip performance. In large systems, a few blocks may be speed-critical, such as the data path in a processor, while other blocks may be non-critical, such as caches.

Today, FPGAs are homogeneous in terms of power and speed; every CLB has the same power and speed characteristics. Power can be saved in a heterogeneous architecture, where some blocks are low power (and slower), by implementing non-critical blocks in the low power blocks. Doing so does not impact overall chip performance, as the timing-critical blocks are not compromised.

One method of creating a heterogeneous architecture is to distribute two core power rails, a high-voltage rail (VDDH) and a low-voltage rail (VDDL). Each part of the FPGA selects from one or the other using embedded power switches and correspondingly takes on high-speed or low-power characteristics. Voltage selection is complete once you have a detailed timing of the design, so only the non-critical blocks should operate from VDDL.

Another method of creating a heterogeneous architecture is to divide the FPGA into different regions that are pre-fabricated to be high speed and low power. Regions can be implemented with different supply voltages, different thresholds, or through a number of other design trade-offs. To avoid performance degradation, design tools must map timing-critical parts of the design into the high-speed regions and non-critical parts into the low power regions.

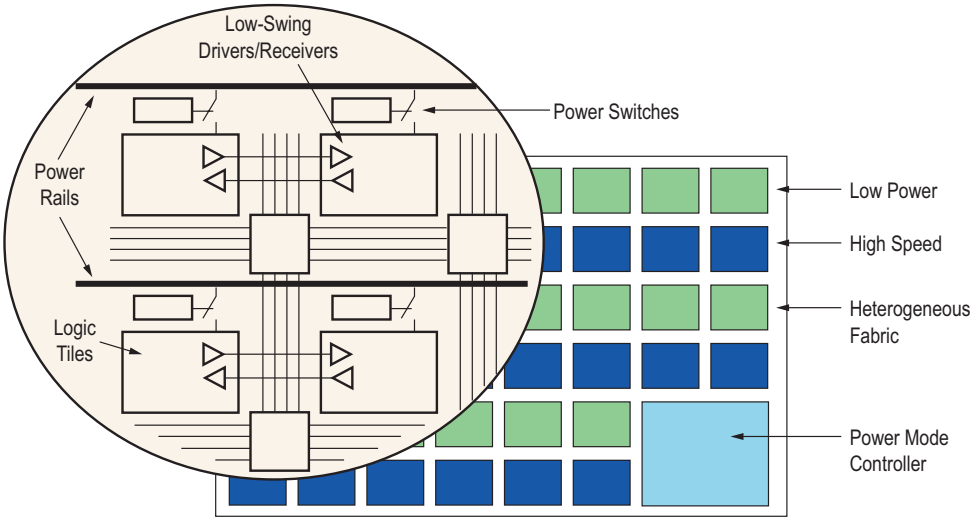


Figure 3 – Concept architecture with various solutions for power reduction

Low-Swing Signaling

As FPGAs increase in capacity, more and more power is consumed in on-chip programmable interconnect. An effective way to reduce this communication power is to use low-swing signaling, where the voltage swing on the wire is much lower than the supply voltage. Low-swing signaling is commonly found today in scenarios where communication takes place over high-capacitance wires such as buses or off-chip links. Low-swing drivers and receivers are more complex than CMOS buffers, so they consume more silicon area. However, as on-chip interconnect grows to be a larger portion of overall power, the power benefits of low-swing signaling will justify the added design complexity. Of course, FPGA users never see the different voltages of their internal signals.

Figure 3 depicts an FPGA architecture with some of the preceding concepts. The programmable fabric is heterogeneous, comprising both high-speed and low-power regions. An on-chip power-mode controller manages various power-down modes, be it deep sleep, suspend, or hibernate. Within the fabric, each logic tile can be powered off using dedicated power switches. Communication through the routing fabric goes through low-swing drivers and receivers to reduce interconnect power.

Conclusion

Beyond the power optimizations currently used in the design of modern FPGAs, a number of user design decisions can yield significant power benefits. Looking forward, more aggressive architectural solutions are available to contain power consumption in future technology generations and enable new FPGA applications.

In addition to the solutions described in this paper, Xilinx is also engaged in various software power optimization research and development work. These efforts are described in the article, "Optimizing FPGA Power with ISE Design Tools," also in this issue of the *Xcell Journal*.

Optimizing FPGA Power with ISE Design Tools

You can reduce the power dissipated in Virtex-4, Virtex-5, and Spartan-3 designs through new power-driven back-end flows.

by Subodh Gupta, Ph.D.
Staff Software Engineer, Design Software Division
Xilinx, Inc.
subodhg@xilinx.com

Jason Anderson, Ph.D.
Principal Engineer, Design Software Division
Xilinx, Inc.
janders@xilinx.com

In the more than 20 years since Xilinx invented the FPGA, research and development has produced dramatic improvements in FPGA speed and area efficiency, narrowing the gap between FPGAs and ASICs and making FPGAs the platform of choice for implementing digital circuits. Today, power consumption is a rising concern for FPGA vendors and their customers. Reducing the power of FPGAs is key to lowering packaging and cooling costs, improving device reliability, and opening the door to new markets such as mobile electronics.

Xilinx is taking the lead in offering low-power FPGA solutions. In this article, we'll illustrate the application of computer-aided design (CAD) techniques, such as those incorporated into Xilinx® ISE™ version 9.2i software, for effective power reduction.

Power dissipation in CMOS circuits comprises both static (leakage) power and dynamic power. Dynamic power is caused by transitions on the signals of a circuit and is governed by the equation:

$$P_{dynamic} = \frac{1}{2} \sum_{i \in signals} C_i \times f_i \times V^2$$

where C_i represents the capacitance of a signal, i; f_i is referred to as "switching activity" and represents the rate of transitions on signal i; and V is the supply voltage.

Static power, on the other hand, is consumed when a circuit is in a quiescent, idle state. Static power results from leakage currents in off transistors, primarily sub-threshold and gate-oxide leakage. An off-MOS transistor is an imperfect insulator allowing sub-threshold leakage current to flow between its drain and source terminals. Gate-oxide leakage is caused by tunneling current through the gate terminal of a transistor to its body, drain, and source.

Technology scaling, such as the recent move to 65 nm, means lower supply voltages and smaller transistor dimensions, leading to shorter wire lengths, less capacitance, and an overall reduction in dynamic power. Smaller process geometries also mean shorter transistor channel lengths and thinner gate oxides, producing an increase in static power as technology scales.

Power Dissipation in FPGAs

The programmability and flexibility of FPGAs make them less power-efficient than custom ASICs for implementing a given logic circuit. The FPGA configuration circuitry and configuration memory consume silicon area, producing longer wire lengths and higher interconnect capacitance. Programmable routing switches attach to the pre-fabricated metal wire segments in the FPGA interconnect and add to the capacitive load incurred by signals.

Most dynamic power in an FPGA is consumed in the programmable routing fabric. Likewise, static power is proportional to total transistor width. The interconnect comprises a considerable fraction of an FPGA's transistors and therefore dominates leakage. Consequently, the interconnect fabric should be a prime target for FPGA power optimization.

Certainly, power can be addressed through process technology, hardware architecture, or circuit-level changes. Virtex™-5 FPGAs, for example, contain "diagonal" interconnect resources, allowing connections to be made with fewer routing conductors and reduced interconnect capacitance. At the transistor level, Virtex-4 and Virtex-5 FPGAs employ triple-oxide process technology for leakage mitigation. Three possible oxide thicknesses are available for each transistor, depending on its speed, power, and reliability requirements. The proliferation and increased use of hard IP blocks such as DSPs and processors can also lower power consumption versus implementing such functionality in the standard FPGA fabric.

It is also possible to reduce power without incurring any costly hardware changes. You can tackle power issues through new power-driven CAD algorithms and design flows, such as those in ISE 9.2i software.

Power Optimization in ISE 9.2i Design Tools

ISE software version 9.2i incorporates power optimization in placement, routing, and through a post-routing technique that reduces power within logic blocks.

Placement

The core algorithm in the Xilinx placer uses analytical (mathematical) techniques. It begins with an initial overlapped placement of a design and then uses a force abstraction to move logic blocks away from highly congested regions, ultimately producing a feasible, overlap-free placement. Once analytical placement is finished, swap-based local optimization is run on the placed design to further refine the placement. The traditional cost function used in our placer considers wire length and timing in this equation:

$$Cost = a \times W + b \times T$$

where W and T are the wire length and timing costs, respectively, and a and b are scalar weights. The values of a and b can be set according to the relative priority of timing versus wire length. The placer costing scheme is illustrated in Figure 1.

As actual routes are not available during placement, the wire length cost depends on wire length estimation. Likewise, the timing cost depends on the user-supplied constraints and estimates of connection delays. To optimize power, we have extended the analytical placement and local optimizations by adding a power component to the cost function, as shown on the right-hand side of Figure 1. The modified cost function is as follows:

$$Cost = a \times W + b \times T + c \times P_{dynamic}$$

where $P_{dynamic}$ is the estimated dynamic power (as defined previously) and c is a scalar weight. You can extract signal switching activity data from simulation and supply

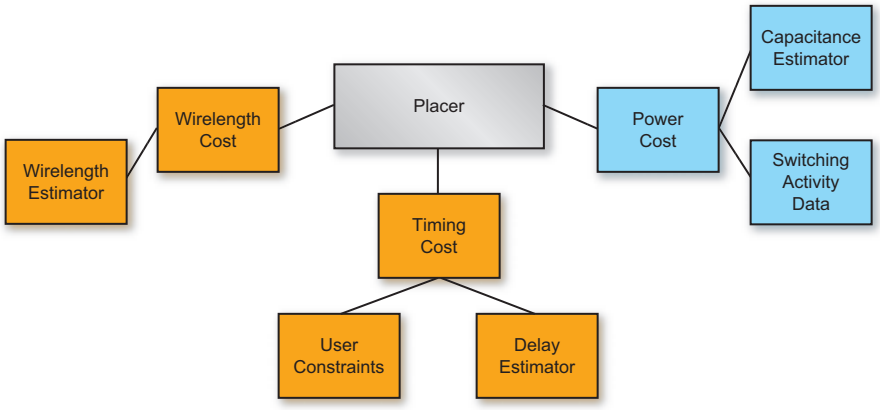


Figure 1 – Placer costing scheme incorporating power

The place and route optimizations discussed in this article aim to lower power in the interconnect fabric.

it to the tool. Alternately, if you do not provide any switching activity data, then the tool assumes a default switching activity for the primary inputs and sequential outputs and propagates activity to the rest of the signals, considering the logic functionality. For best results, user-supplied switching activity data is required.

During placement, the capacitance of a signal is not known and must therefore be estimated. We have constructed an empirically derived capacitance estimation model based on signal parameters available during placement:

$$C_i = f(FO_i, XS_i, YS_i)$$

where f represents a generic mathematical function, C_i is the capacitance of signal i, FO_i is the number of fanouts of signal i, and XS_i is the X-span and YS_i is the Y-span of signal i in the placement, respectively. These parameters are architecture-independent and are readily available during placement.

To build the model, we extracted the capacitance, number of fanouts, X-span, and Y-span for each signal in a suite of designs collected from Xilinx customers. We then used least-squares regression analysis to fit the capacitance to a quadratic function of the model parameters. On average, the analytical equation gives a 30% error for a wide range of designs.

Routing

Once the logic blocks are assigned to physical locations on the FPGA, we must route the connections between the blocks. The router uses a negotiated congestion-routing algorithm, where during initial iterations shorts between signals are permitted. In later iterations, the penalty for creating shorts is increased, until no routing conductor is used by more than one signal. Timing-critical connections are routed to minimize their delay, which involves computationally intensive RC delay calculations. Most connections, however, are not timing-critical.

In the power-aware router, we choose to optimize the capacitance of such non-critical connections. To achieve this, we changed the router's cost function for non-timing critical connections to consider capacitance, as opposed to the prior approach of basing cost on other factors such as estimated delay or scarcity. To illustrate the algorithm, consider the routing graph of Figure 2.

Each node in the graph represents a routing conductor or logic block pin and each edge represents a programmable routing switch. The router must select a path between the source and target pins. The original cost and capacitance cost of each node is shown internal to each node in the figure. To minimize the original costs, the routing between the source and target pin would have taken the path shown in blue. However, in the power-aware flow, the router will use the path shown in green, which has lower overall capacitance.

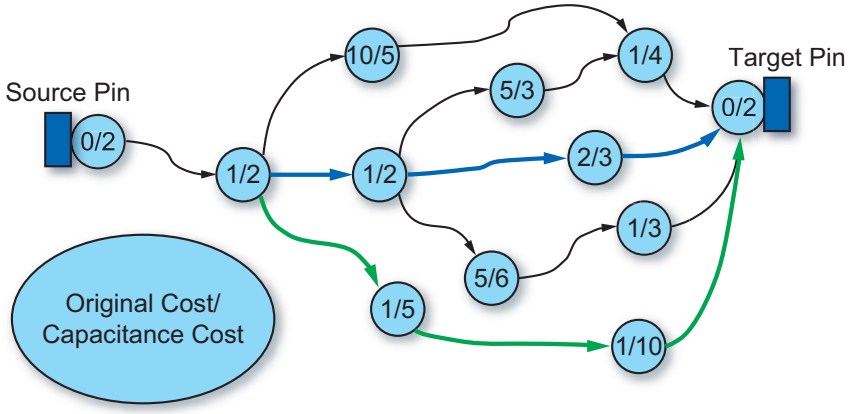


Figure 2 – Routing graph with capacitance costs on nodes

Results from Power-Aware Placement and Routing

A set of industrial designs were placed and routed using both the traditional place and route flow, as well as the power flow described earlier. The designs were augmented with built-in automatic input vector generation by attaching a linear feedback shift register-based (LFSR-based) pseudo-random vector generator to the primary inputs. This permitted us to perform board-level measurements of dynamic power without requiring a large number of externally applied waveforms.

The industrial designs were mapped into Spartan-3, Virtex-4, and Virtex-5 devices. Results showed that dynamic power was reduced as much as 14% for Spartan-3 FPGAs, 11% for Virtex-4 FPGAs, and 12% for Virtex-5 FPGAs. On average, across all designs, dynamic power was reduced by 12% for Spartan-3 FPGAs, 5% for Virtex-4 FPGAs, and 7% for Virtex-5 FPGAs. For all families, on average, the speed performance hit was between 3% and 4%, which is small and we believe acceptable in power-conscious designs. Considering that these are initial results coming from software changes alone, we consider the power benefits quite encouraging.

Reducing Power within Logic Blocks

The place and route optimizations discussed in this article aim to lower power in the interconnect fabric. We also devised an approach to reduce power within logic

blocks, specifically in look-up-tables (LUTs) when not all LUT inputs are used (Figure 3). A K-input LUT is a small memory capable of implementing any logic function with no more than K inputs. Figure 3 shows how a hypothetical three-input LUT (having inputs A1, A2, and A3) implements a two-input logical-AND function. The truth table for the logical-AND is shown in the LUT SRAM contents on the left side of the multiplexer tree.

Note that input A3 is unused in Figure 3. Normally, unused inputs are treated as "don't cares" and assumed to be either 0 or 1. Consequently, to account for this in the case of Figure 3, the Xilinx software "replicates" the logic function in both the top and bottom half of the LUT SRAM memory contents. Unused LUT inputs occur frequently in customer designs, especially in Virtex-5 designs, where LUTs have six inputs.

In the Virtex-5 hardware, unused LUT inputs are pulled high to logic-1; this property is the root of our optimization. If A3 is pulled to logic-1, the bottom input to the deepest two-input multiplexer in the tree is never selected. However, because the logic function is replicated in both the top and bottom half of the LUT memory contents, toggling can occur on internal multiplexer nodes n1 and n2 based on changes in signal inputs A1 and A2. This toggling consumes dynamic power needlessly, as transitions on n1 and n2 are never propagated to the LUT output.

This optimization entails detecting unused LUT inputs at the post-routing stage and setting LUT memory contents to be logic-0 so as to eliminate unnecessary internal toggling and without disrupting the logic functionality. Returning to the example of Figure 3, the bottom half of the LUT memory contents would be set to logic-0. No toggling would occur on internal nodes n1 and n2, and therefore no dynamic power would be consumed by charging or discharging n1 and n2.

We conducted board-level power measurements to evaluate this optimization on industrial designs and observed that it provides several percentage points of dynamic power savings. These results are especially promising, as this optimization is "no cost" in the sense that it can be carried out post-routing and causes no area or performance penalty.

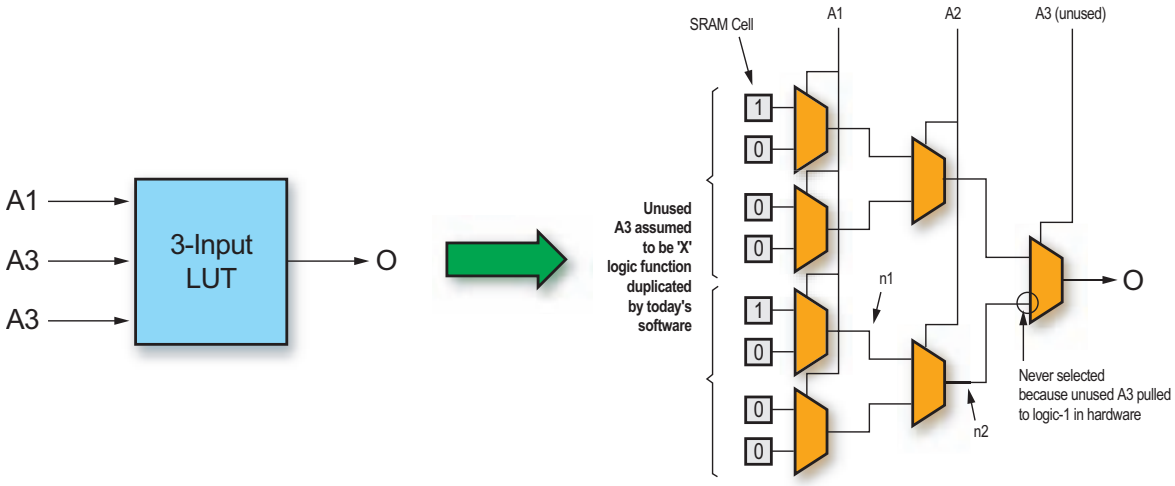


Figure 3 – Optimizing power within a LUT

Conclusion

Many future directions exist for further reducing power in design tools. In front-end HDL synthesis, proven optimizations from the ASIC domain can be adapted for FPGAs, such as clock gating and operand isolation. As well, FPGA-specific power optimizations can be applied, such as mapping logic into available block RAMs (which can be used as large ROMs) instead of using LUTs and the general fabric.

Power-aware logic synthesis and activity-driven technology mapping to LUTs are well-studied in the literature and will yield considerable power reductions for Xilinx FPGAs. In placement, improved capacitance estimation accuracy will yield even higher power reductions.

Two areas that we feel are particularly fertile are glitch and leakage optimization. Glitches are spurious transitions that occur on signals caused by unequal path delays in circuits. Such transitions are unnecessary, yet they play a significant role in dynamic power. CAD techniques to mitigate glitches include equalizing path delays or inserting registers along the most "glitchy" paths. Leakage in a digital CMOS circuit depends strongly on the circuit's applied input state. Therefore, one approach to leakage reduction in CAD is to automatically alter the circuit so that its signal values spend more time in low-leakage states.

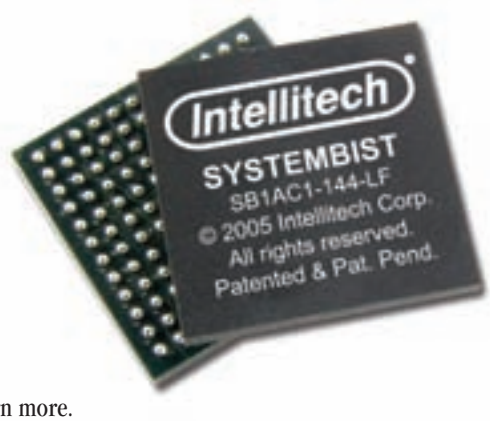
The results show that significant strides have already been made in reducing power through ISE design tools. We are optimistic that the future is bright for achieving additional power reductions in software. Power-conscious solutions comprising power-aware CAD algorithms and power-optimized devices such as Virtex-5 FPGAs form a compelling story. Continued advances in low-power software and hardware will open the door for Xilinx FPGAs entering new power-sensitive markets.

Great Test, Less Filling

SystemBIST™ enables FPGA Configuration that is less filling for your PCB area, less filling for your BOM budget and less filling for your prototype schedule. All the things you want less of and more of the things you do want for your PCB – like embedded JTAG tests and CPLD reconfiguration.

Typical FPGA configuration devices blindly "throw bits" at your FPGAs at power-up. SystemBIST is different – so different it has three US patents granted and more pending. SystemBIST's associated software tools enable you to develop a complex power-up FPGA strategy and validate it. Using an interactive GUI, you determine what SystemBIST does in the event of a failure, what to program into the FPGA when that daughterboard is missing, or which FPGA bitstreams should be locked from further updates. You can easily add PCB 1149.1/JTAG tests to lower your down-stream production costs and enable in-the-field self-test. Some capabilities:

- User defined FPGA configuration/CPLD re-configuration
- Run Anytime-Anywhere embedded JTAG tests
- Add new FPGA designs to your products in the field
- "Failsafe" configuration – in the field FPGA updates without risk
- Small memory footprint offers lowest cost per bit FPGA configuration
- Smaller PCB real-estate, lower parts cost compared to other methods
- Industry proven software tools enable you to get-it-right before you embed
- FLASH memory locking and fast re-programming
- New: At-speed DDR and RocketIO™ MGT tests for V4/V2

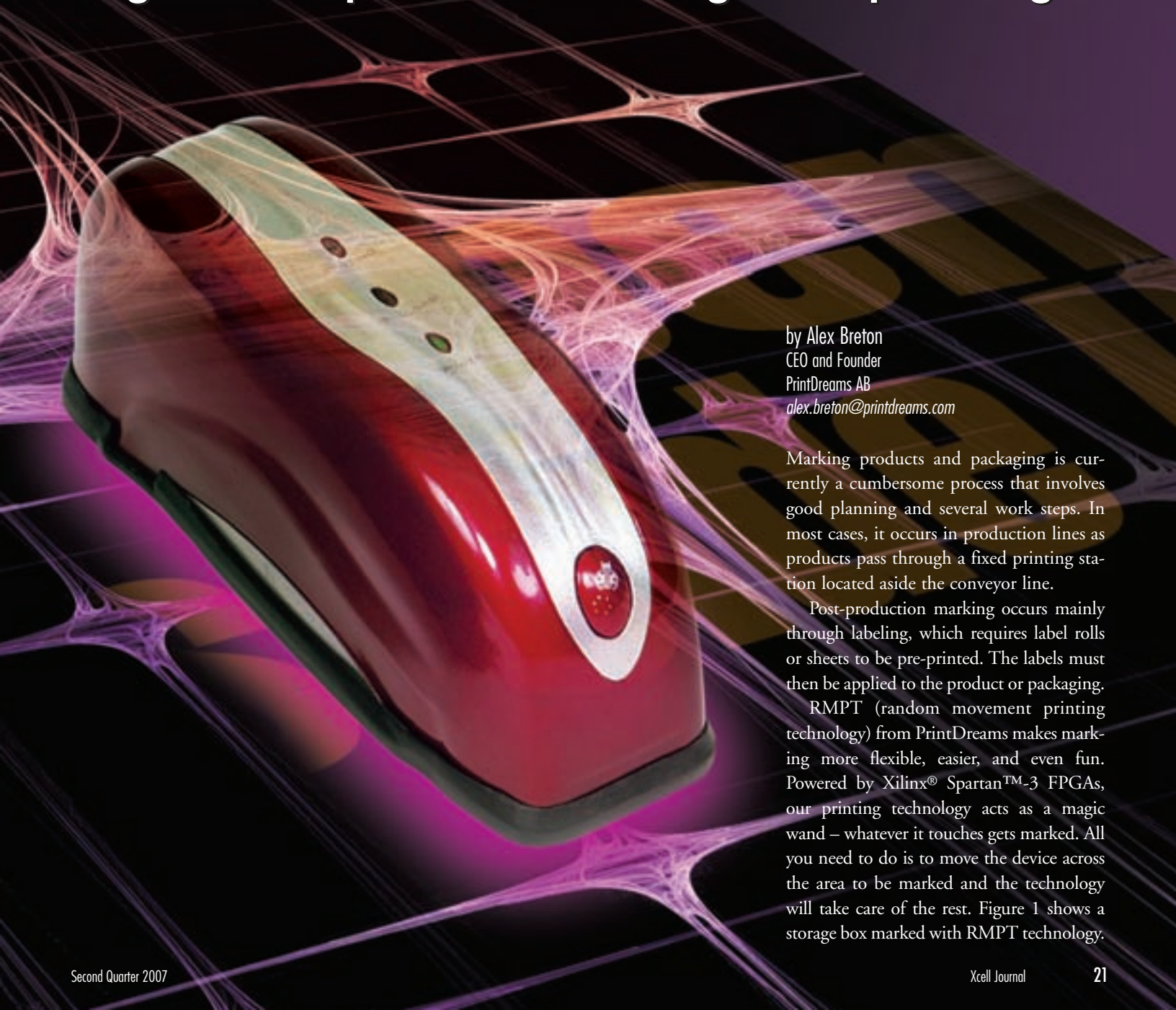


If your design team is using PROMS, CPLD & FLASH or CPU and in-house software to configure FPGAs please visit our website at http://www.intellitech.com/xcell.asp to learn more.

Intellitech®

Inkjet Power Unleashed

The Spartan-3AN FPGA activates digital ubiquitous marking and printing.



by Alex Breton
CEO and Founder
PrintDreams AB
alex.breton@printdreams.com

Marking products and packaging is currently a cumbersome process that involves good planning and several work steps. In most cases, it occurs in production lines as products pass through a fixed printing station located aside the conveyor line.

Post-production marking occurs mainly through labeling, which requires label rolls or sheets to be pre-printed. The labels must then be applied to the product or packaging.

RMPT (random movement printing technology) from PrintDreams makes marking more flexible, easier, and even fun. Powered by Xilinx® Spartan™-3 FPGAs, our printing technology acts as a magic wand – whatever it touches gets marked. All you need to do is to move the device across the area to be marked and the technology will take care of the rest. Figure 1 shows a storage box marked with RMPT technology.

The extremely compact size surprises users, who immediately wonder how they can insert a letter-size sheet of paper through such a small device. Instead, now it is the printer that goes to the paper (or anything else, for that matter).

The secret resides in advanced sensor technology in combination with effective control algorithms to determine what, when, and where to print. We can control all inkjet nozzles as a column. We can also control every nozzle independently if movement with three degrees of freedom is required. By using the Spartan-3AN FPGA platform, we can configure new parameters very quickly. The FPGA accesses the content to be printed directly from its internal block RAM if content size is limited.

RMPT printers are hand-held devices that are portable and battery operated. The unique dual-mode power management (with static power reductions of as much as 99%) combined with great performance is something that our engineers and customers highly appreciate.

Unlimited Sizes and Formats

There are basically no size or format limits for what you can achieve with RMPT technology. The great reconfigurable nature of Xilinx FPGAs makes it possible to move into new formats very fast. We can get them to work as separate controllers, adding new entities to achieve larger print swaths.

In this sense, another useful feature in the Spartan-3AN platform is the built-in multi-boot feature, which allows us to use the same hardware design to run different configurations in the geometry of print-head nozzles and print-head blocks. This saves us money and new design efforts.

When we invented RMPT, we broke one of the holiest unspoken rules in the printer industry: "The width of the printer device by default always supersedes the width of the media to be printed." Suddenly, the physical limit went all the way down to the print-head cartridge size, a battery, and an electronic board. With our new approach, the PCB footprint starts to become the bottleneck of further size reduction, while in the old technological approach there was always plenty of space in the large housings. The on-chip flash of the Spartan-3AN platform is therefore a feature we welcome very much, as it helps us to additionally reduce footprint size and external components.

Figure 1 – PrintDreams RMPT technology deployed on a storage box

The extremely compact size surprises users, who immediately wonder how they can insert a letter-size sheet of paper through such a small device. Instead, now it is the printer that goes to the paper (or anything else, for that matter). One of our initial customers came up with a slogan that summarizes the technology quite well: "Take the printer to the project."

Affordable and Environmentally Friendly Technology

Our technology is very competitive even when compared to current solutions, both because of the affordability of Spartan-3 FPGAs and the fact that RMPT is basically solid state, without complex mechanical setups.

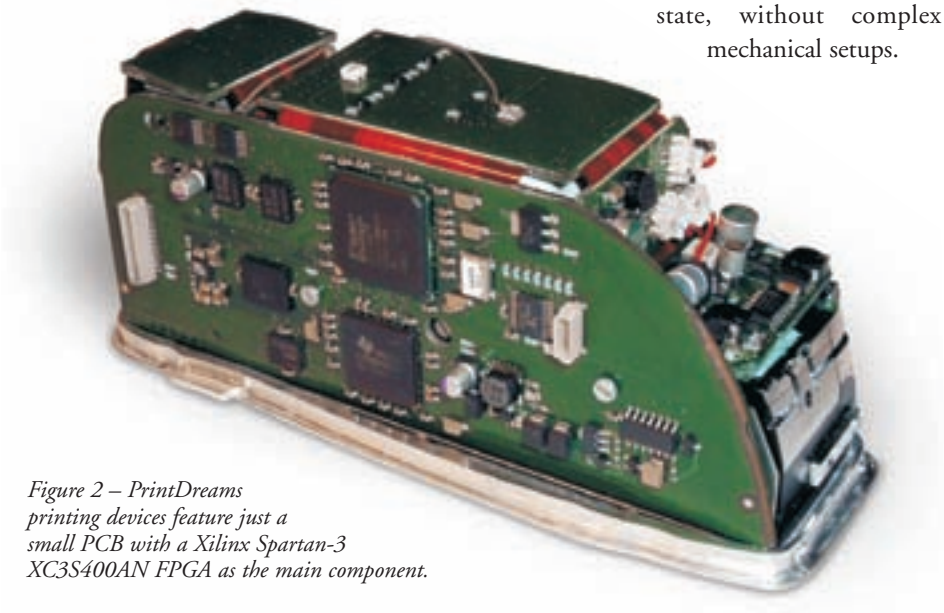


Figure 2 – PrintDreams printing devices feature just a small PCB with a Xilinx Spartan-3 XC3S400AN FPGA as the main component.

If you take apart any desktop or portable printer device, you will notice how many mechanical parts are involved – sometimes more than 200 pieces. It makes the design and manufacture of printer devices a complex, energy-consuming, and costly process to set up. Every part requires special tooling, plus a step in a long assembly line. The industry eats up hundreds of tons of steel and plastic every year for the manufacture of printing devices.

In our case, all we basically need is one small PCB and a print head. Our devices can weigh about as much as a cell phone and are extremely easy to recycle. They eliminate the need for labels and are an excellent choice for marking, coding, and printing specific tasks. The only area where RMPT printing is not competitive is for printing larger print jobs (more than 5-10 pages at a time). We do not aim to compete with very high-quality printing either. Instead, we want to compete with our compact size, affordability, and flexibility.

Too Much Flexibility?

Our technology is not dependent on certain types of substrates. Provided that a suitable ink is available, there are no limits for the types of substrates on which RMPT can print. Cardboard, wood, fabrics, and plastics are the most common materials, but we even have solutions to print onto metal and glass if required.

One of the first segments where RMPT was introduced was the crafts industry, through a product called the Xyron Design Runner. It is amazing to see the amount and variety of materials that scrapbookers use for their projects. We have been surprised to see some projects done in cork and others in very rough fabrics.

For some of our upcoming products – this time in the toy industry – the great flexibility of our technology becomes even a concern. We would not want children printing "digital graffiti" on floors and walls. The most hair-raising scenario (mainly for parents than for anyone else) would be children painting each other using our devices, because skin is just another one of the various substrates where our technology has no problem printing.

In other areas, this flexibility is welcome. Because we can control not only common inkjet print heads but also other types of equipment such as laser heads, we can easily arrange more advanced and permanent marking of metal parts or other hard substrates. Again, the flexibility of Xilinx FPGAs is indispensable to us.

Pushing Technology to its Limits

The most critical issue during technical development of the advanced version of RMPT has been the real-time aspects. In many industries, such as telecom and networking, the delays that define real time are measured in milliseconds. We measure real time on a microsecond level.

Our system must be able to handle as many as 2.5 million decisions every second, deciding whether to print from one of each of the print-head nozzles. In a typical 600-dpi inkjet print head, this corresponds to more than 300 nozzles. That exerts an immense pressure on both the hardware and software components.

It has been a bottleneck elimination process all along. Whereas we once used a DSP to serially run more than 300 calculations, we have now enhanced our system to operate at high capacity and with great parallelization, without losing the ability to reconfigure new types of print heads. An ASIC would have locked us into probably one single type of print-head architecture.

Being the first with a technology like this, we are of course very keen on protecting our IP. Therefore, the new Spartan-3AN platform security features that protect against cloning and reverse engineering eliminated our outstanding concerns during the selection of computing technology for our devices.

Conclusion

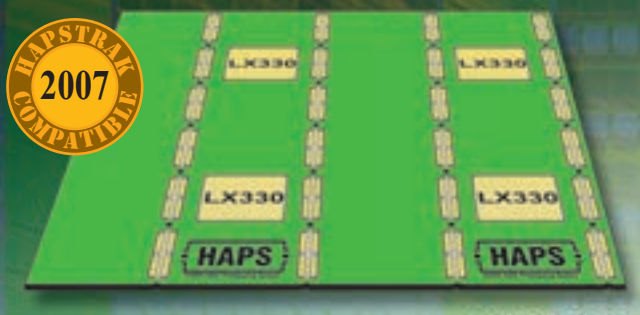
High flexibility and performance with good security at an affordable price are the features that led us to choose the Xilinx Spartan-3AN platform. Even better power management and a smaller footprint make the platform simply irresistible.

For more information about PrintDreams and its RMPT technology, visit *www.printdreams.com.*

ASIC PROTOTYPING

HAPS – HARDI ASIC Prototyping System
a high performance, high capacity FPGA platform for ASIC prototyping and emulation composed of multi-FPGA boards and standard or custom-made daughter boards

HapsTrak
a set of rules for pinout and mechanical characteristics, which guarantees compatibility with previous and future generation HAPS motherboards and daughter boards



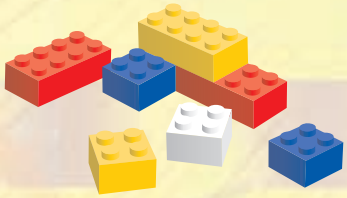
HAPS-50

HAPS-30

HAPS-20

HAPS-10

HARDI Electronics



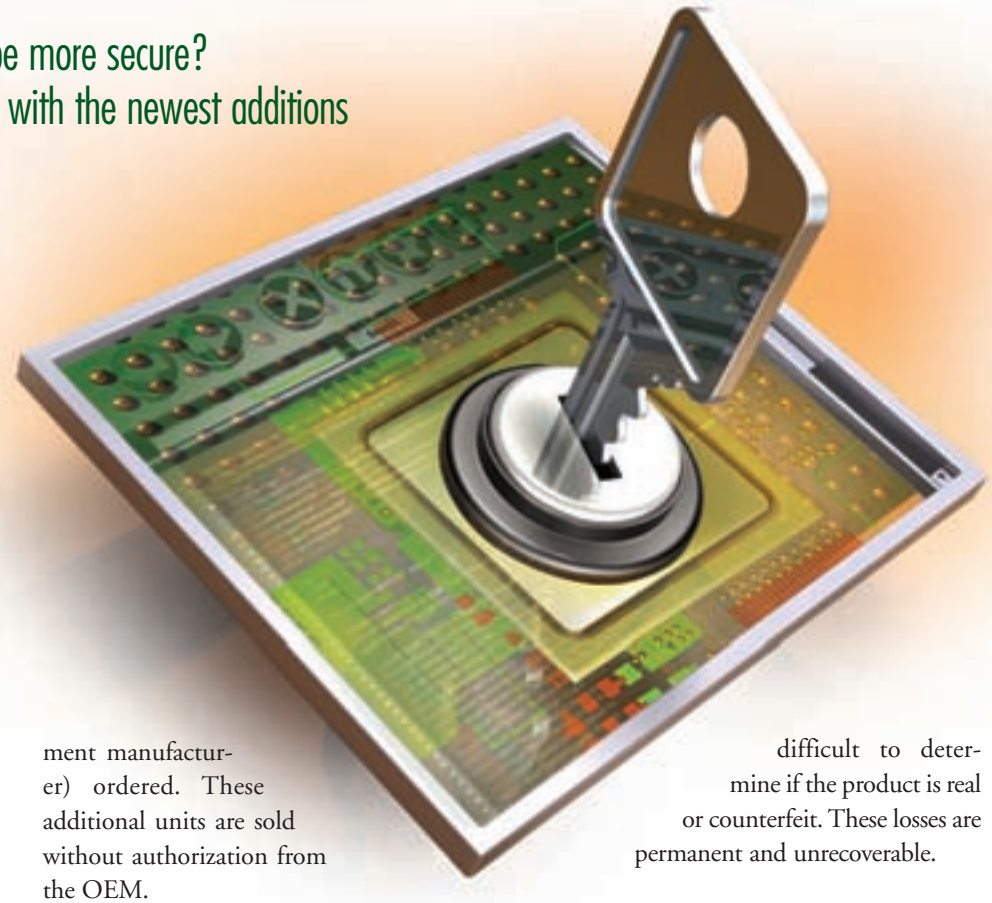
www.hardi.com, haps@hardi.com

HARDI Electronics Inc., 26831 Magdalena Lane, Mission Viejo, CA 92691, (949) 202-5572

Low-Cost Security Solutions with Spartan-3A and Spartan-3AN Platforms

Could your low-cost FPGA design be more secure? Check out what Xilinx has to offer with the newest additions to the Spartan-3 generation.

by Maureen Smerdon
Strategic Marketing Manager
Xilinx, Inc.
maureen.smerdon@xilinx.com



Security has become a hot topic: whether boarding a plane, closing the front door, or beginning a next-generation circuit design, security is a significant issue. For designers, the greatest threat comes from the alarming amount of counterfeited products on the market as a result of design thefts. According to the Anti-Counterfeiting Coalition, the estimated dollar exchange associated with counterfeiting throughout the United States in 2003 was $287 billion, or 63% of the total $456 billion of counterfeit products sold annually worldwide.

In this article, I'll describe Xilinx® security measures that can protect your low-cost FPGA designs.

The Top Three Security Threats

The most common security breach in electronics design is reverse engineering. This occurs when a thief attempts to recreate or rebuild a product with the intent of selling it on the open market at a lower cost. Through reverse engineering, thieves can build designs much faster without incurring the expense of R&D, quite probably at a lower cost.

Today, as companies have moved to outsource manufacturing, they are subject to new security breaches called overbuilding and cloning. In overbuilding, the outsourced manufacturer simply manufactures more units than the OEM (original equipment manufacturer) ordered. These additional units are sold without authorization from the OEM.

Cloning is when a thief creates a duplicate of your design, IP, or product under the same (or another) label. Again, cloners do not incur any R&D costs. Both overbuilt and cloned products have a drastically reduced time to market.

What remains unknown are the intangibles associated with such security breaches. Whether a product is reverse-engineered, overbuilt, or cloned, it means a substantial revenue loss for the OEM. In addition to the loss of revenue, there is also a cost associated with quality in the form of returns. This could affect the brand image and potentially add to the OEM's bottom line due to increased RMAs (return materials authorizations) or technical support to determine what the issue is and how to resolve the end customer's problem. Ultimately, it may be difficult to determine if the product is real or counterfeit. These losses are permanent and unrecoverable.

Security Using DeviceDNA

Traditionally, FPGAs use some kind of bitstream encryption to guard against reverse engineering and cloning. In yesteryear's model, this worked well. It will not, however, protect you in today's world against overbuilding.

As a designer, how can you protect your design against all three security thefts? Xilinx has a few solutions, recently introducing the Spartan™-3A and Spartan-3AN device families with DeviceDNA to help you fend off cloners, overbuilders, and reverse engineers.

The DeviceDNA design-level security protects the design, the IP, and the embedded code. DeviceDNA is a specific 57-bit ID that is unique to each device. This 57-bit ID is contained in an area on the FPGA and is fixed or set at the Xilinx factory; it

cannot be altered. Both Spartan-3A and Spartan-3AN FPGAs contain a unique ID in each device shipped.

This ID is then combined with a designer's personalized algorithm and stored on the FPGA. The algorithm is basically an arithmetic equation that defines how to take the DeviceDNA and create a result. The result can then be stored anywhere, such as in the external memory or in flash. The algorithm is the secret to the security because only the designer knows it. Although it is stored on the FPGA, to an onlooker it just looks like part of the bitstream.

Spartan-3A Security

For Spartan-3A devices, the algorithm compares the result using DeviceDNA to the result stored in the flash after the device has been configured. If they match, the design is authorized. If they do not match, then the design can be set up to behave in a variety of ways, from slight to severe functional impairment.

Let's look at an example of authentication in our daily life. Say you stop at a local fast food restaurant for a snack. You are out of cash and are forced to use your ATM card (DeviceDNA), which only you are authorized to use. You place your order and then swipe the card. The machine asks for your PIN number (personalized algorithm). The system then compares the PIN number you entered with the number stored at the bank. If it matches, you get your snack. If not, you will go hungry.

The potential weakness is if someone has both your ATM card and your PIN number. The PIN authorization algorithm number, once learned, is easily cloned. This is why the authorization algorithm is incorporated into the design itself. The algorithm is placed in the most secret location inside of programmable logic, with millions of configuration options.

Spartan-3AN Security

For the Spartan-3AN platform, our new non-volatile FPGA, the process is almost the same – with a few enhancements. The first security enhancement is that the bitstream is hidden inside the FPGA. This makes it more difficult for someone to monitor.

The second security enhancement that the Spartan-3AN FPGA has are two unique serial numbers, the DeviceDNA and a factory flash ID, found in flash memory. The two unique IDs give more than 70 bytes of serial numbers, resulting in a large number of algorithmic possibilities and therefore increasing the amount of time it would take to breach the authentication algorithm. Now the design is specifically tied to both the FPGA and the flash IDs.

Applying the earlier analogy, having two unique IDs is like requiring two different ATM cards to get a snack.

The third improvement is in the stored authorization code. On the Spartan-3AN platform, the authorization code can be stored on-chip in a special one-time programmable 64-byte register called the Flash User Field. This allows the complete security system to be self-contained. With no need for external interfaces or storage, overall security increases and reverse engineering is more difficult.

The authentication algorithm is user-defined, allowing you to implement the right level of security within your design budget. The authentication algorithm is also the primary secret in the security system. Something in the authentication process must be a secret so that security is not breached. Because the algorithm is unknown, it is the key to the design-level security. The algorithm is implemented in the fabric of the FPGA; therefore, it becomes a handful of bits within the millions of configuration bits in the FPGA. Unless you know how the bits fit together or what the algorithm is, it looks like just a mass of numbers. Figure 1 outlines a possible flow using Spartan-3AN devices.

The Spartan-3AN design-level security shown in Figure 2 is a completely self-contained security solution. The flash contains both the FPGA configuration bitstream and a previously generated

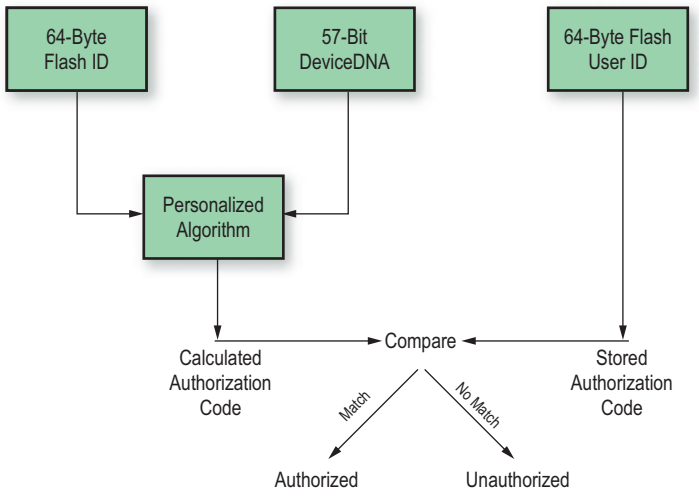


Figure 1 – Possible security setup with Spartan-3AN FPGAs

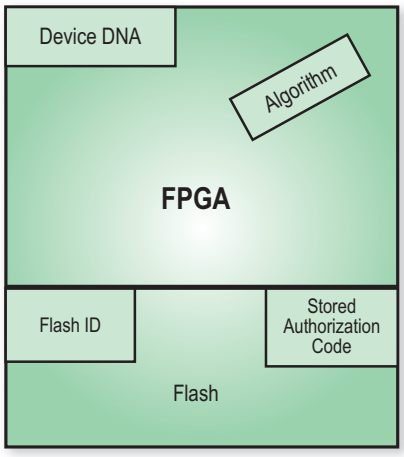


Figure 2 – Spartan-3AN device with security

authorization code. This code is stored in the one-time programmable Flash User Field by a trusted/secured manufacturer or registration process.

At power up, the FPGA configures normally. Once configured, the FPGA application includes circuitry that validates the design authorized to operate on the associated Spartan-3AN FPGA. The DeviceDNA and factory flash ID will be read by the authentication algorithm, which in turn generates an active authorization code that is compared to the previously generated authorization code stored in the Flash User Field. If both codes are equal, the device is authenticated. Otherwise, the device is illegitimate and unauthorized.

Access Denied

The handling of failed authentication is another one of the strengths of the DeviceDNA design-level approach. Authentication can be completely integrated into the design. Thus, multiple responses can result from an unauthorized design, such as:

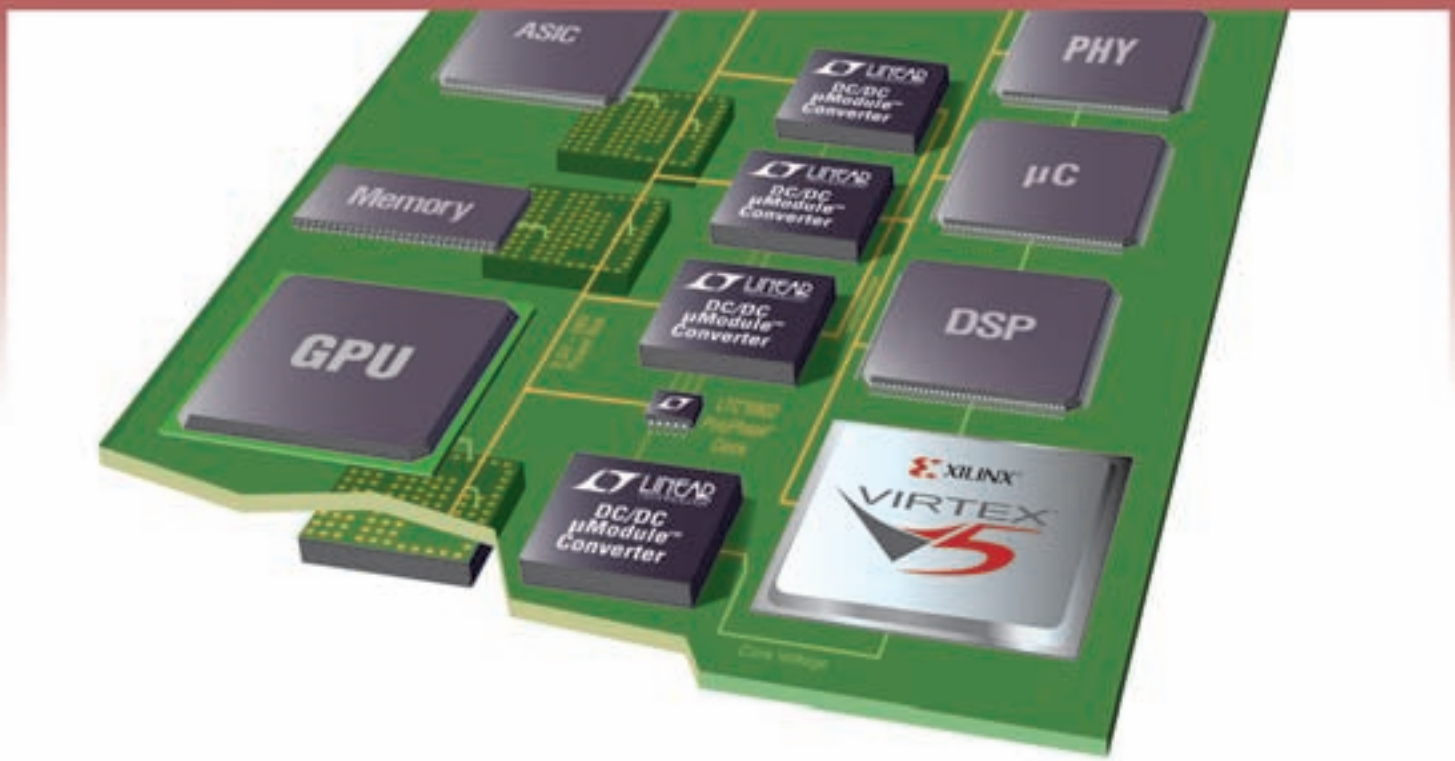
- No functionality – the design completely stops functioning
- Limited functionality – primary or key circuits are disabled or bypassed
- Time bomb – full functionality for only a limited period of time
- Active defense – the system monitors activities and defends against attack
- Permanent self-destruction – erases all flash content and permanently lock-downs flash to all zeros

The design-level security described here is the basic level of security that can be achieved within the Spartan-3A and Spartan-3AN platforms.

Conclusion

The security measures in Spartan-3A and Spartan-3AN platforms provide many ways to protect from reverse engineering, overbuilding, and cloning. To learn more about securing your low-cost FPGA designs, see the Spartan Generation Configuration User Guide at *www.xilinx.com/bvdocs/userguides/ug333.pdf*.

Compact Power Supplies for Xilinx-Based Systems



Tiny µModule™ Power Supplies Fit on Both Sides of PC Board

Our µModule DC/DC point-of-load power supply family is complete with built-in inductor, MOSFETs, bypass capacitors and compensation circuitry. At only 2.8mm height, these tiny, lightweight (1.7g) point-of-load regulators fit the tightest spaces on top and bottom of your board. Small size and impressive low thermal impedance allow high power conversion from a wide range of voltages. Our µModule DC/DC converters simplify the design of your Xilinx-based system and are backed by rigorous testing and high product reliability.

µModule DC/DC Converters for Core, I/O, Clock & System Power

V_{IN}: 4.5V-28V; V_{OUT}: 0.6V-5V						LGA Package (15°C/W)	
Part No.	I_{OUT} (A)	Current Share	PLL	Track, Margin	Remote Sense	Height (mm)	Area (mm)
LTM®4602	6	Combine two for 12A to 24A or 4x LTM4601 for ≤48A				2.8	15x15
LTM4603	6		✔	✔	✔		
LTM4603-1	6		✔	✔			
LTM4600	10						
LTM4601	12		✔	✔	✔		
LTM4601-1	12		✔	✔			
V_{IN}: 2.25V-5.5V; V_{OUT}: 0.8V-3.3V							
LTM4604*	4	2x for 8A	✔			2.3	9x15

*Future Product

Info & Online Store

Nu Horizons Electronics Corp.
Tel: 1-888-747-NUHO
www.nuhorizons.com/linear

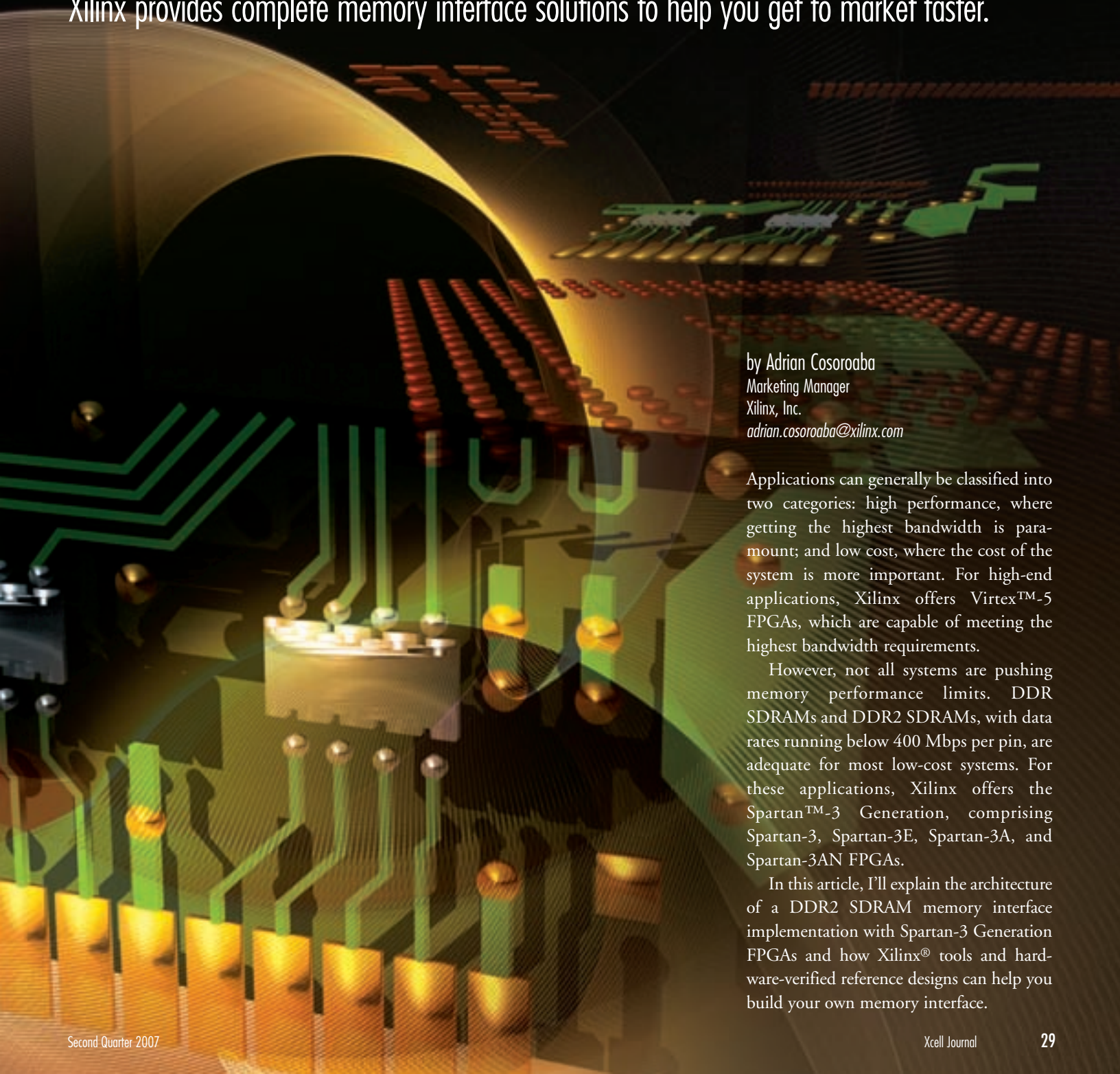
FREE
LTM4600
µModule Board
to qualified customers.

NU HORIZONS ELECTRONICS

LINEAR TECHNOLOGY

Implementing Low-Cost DDR2 Interfaces with Spartan-3A FPGAs

Xilinx provides complete memory interface solutions to help you get to market faster.



by Adrian Cosoroaba
Marketing Manager
Xilinx, Inc.
adrian.cosoroaba@xilinx.com

Applications can generally be classified into two categories: high performance, where getting the highest bandwidth is paramount; and low cost, where the cost of the system is more important. For high-end applications, Xilinx offers Virtex™-5 FPGAs, which are capable of meeting the highest bandwidth requirements.

However, not all systems are pushing memory performance limits. DDR SDRAMs and DDR2 SDRAMs, with data rates running below 400 Mbps per pin, are adequate for most low-cost systems. For these applications, Xilinx offers the Spartan™-3 Generation, comprising Spartan-3, Spartan-3E, Spartan-3A, and Spartan-3AN FPGAs.

In this article, I'll explain the architecture of a DDR2 SDRAM memory interface implementation with Spartan-3 Generation FPGAs and how Xilinx® tools and hardware-verified reference designs can help you build your own memory interface.

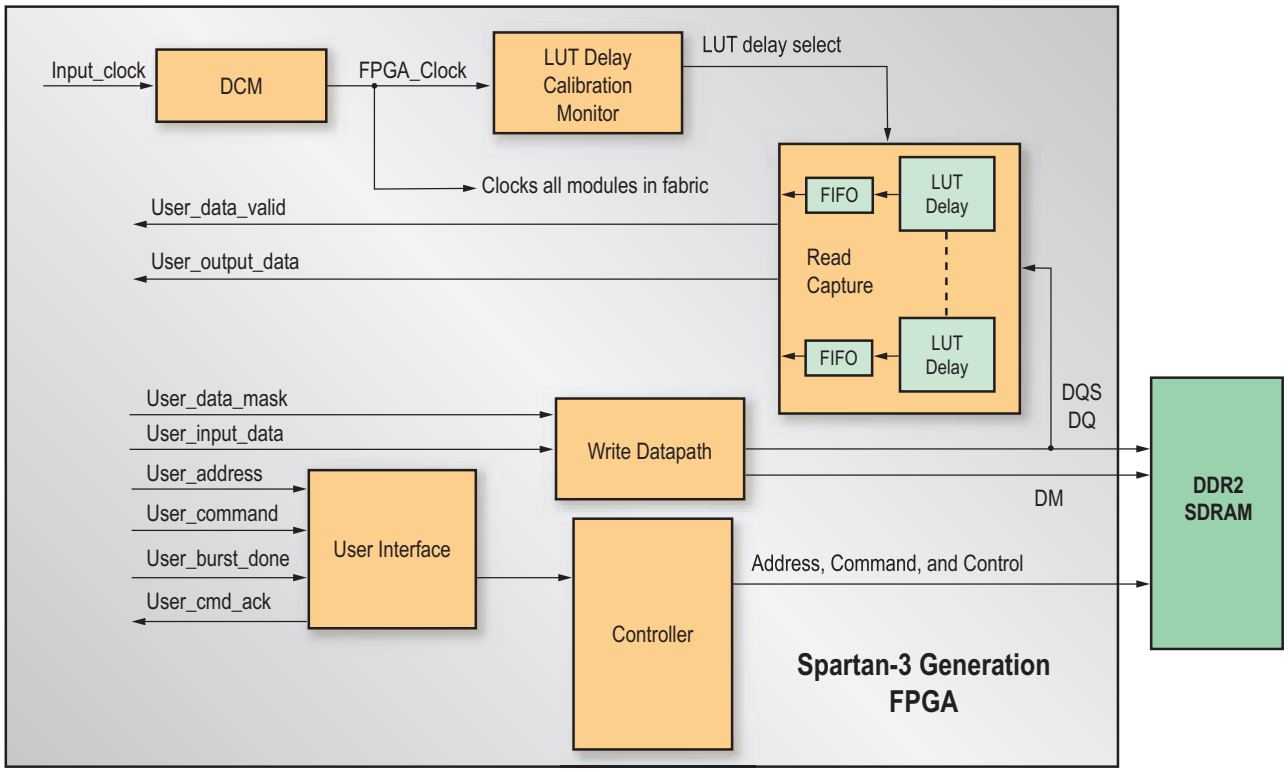


Figure 1 – DDR 2 SDRAM interface implementation for Spartan-3 Generation FPGAs

Memory Controller and DDR2 Interface

Three fundamental building blocks comprise a controller and memory interface for an FPGA-based design: the read and write data interface; the memory controller state machine; and the user interface, which bridges the memory interface design to the rest of the FPGA design (Figure 1).

In Spartan-3 Generation FPGAs, the user interface is a handshaking-type interface. When you send a command (read or write) along with the address and data for write, the user-interface logic responds with the "user_cmd-ack" signal when the next command can follow.

The functional blocks implemented in the fabric are clocked by the output of the digital clock manager (DCM), which also drives the LUT (look-up table) delay calibration monitor. This calibration circuit is used to select the number of LUT-based elements required to delay the read data strobe (DQS) with respect to read data (DQ), in order to properly align it for capture inside the FPGA.

During a read transaction, the DDR2 SDRAM device sends the DQS and associated data to the FPGA edge-aligned with the DQ. Capturing the DQ is a challenging task, because data changes at every edge of the non-free-running DQS strobe.

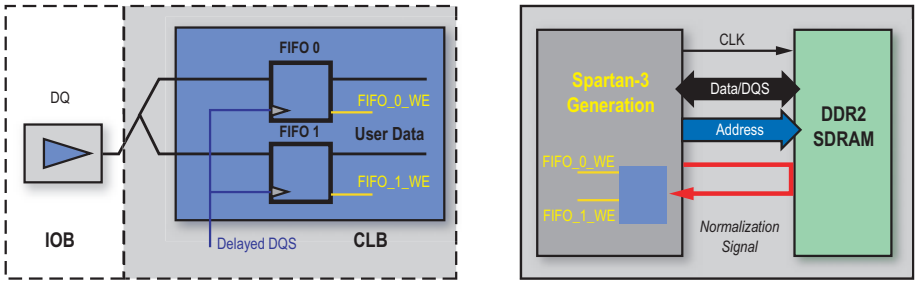


Figure 2 – DQ capture FIFO implementation for Spartan-3 Generation FPGAs

DQ capture is implemented using the LUTs in the configurable logic blocks (CLBs). The DQ capture implementation uses a tap-delay mechanism based on LUTs. The DQS clocking signal is delayed to provide enough of a timing margin. The capture of DQ is implemented in dual-port LUT-based RAM (Figure 2). The LUT RAM is configured as a pair of FIFOs; each data bit is input into the rising-edge (FIFO_0) and falling-edge (FIFO_1) FIFOs. These 16-entry-deep FIFOs are asynchronous, with independent read and write ports.

The transfer of DQ from the DQS clock domain to the memory controller clock domain occurs through these asynchronous FIFOs. Data can be simultaneously read out of both FIFO_0 and FIFO_1 in the memory controller clock domain. FIFO read pointers are generated in the FPGA internal clock domain. The generation of write enable signals (FIFO_0 WE and FIFO1_WE) occurs using the DQS and an external loop-back or normalization signal.

The external normalization signal is driven to an input/output block (IOB) as an output; it is then taken as an input through the input buffer. This technique compensates for the IOB, device, and trace delays between the FPGA and the memory device.

The write data interface generates and controls write data commands and timings. The write data interface uses IOB flip-flops and the DCM's 90-, 180-, and 270-degree outputs to transmit a DQS strobe properly aligned to the command and data bits, per DDR and DDR2 SDRAM timing requirements.

There are other aspects of the design, including the overall controller state machine logic generation and user interface. To make the complete solution easier for designers, Xilinx developed the Memory Interface Generator (MIG) tool.

Controller Design and Integration with the MIG Software Tool

Integrating all of your building blocks including the memory controller state machine is essential for the completeness of your design. Controller state machines vary with memory architecture and system parameters. State machine code can also be complicated – and a function of many variables – such as:

- Architecture (DDR, DDR2)
- Number of banks (external or internal to the memory device)
- Data bus width
- Memory device width and depth
- Bank and row access algorithms

Finally, parameters like data-to-strobe ratios (DQ/DQS) can add further complexity to the design. The controller state machine must issue the commands in the correct order while considering the timing requirements of the memory device.

The complete design can be generated with the MIG software tool, freely available from Xilinx as part of the ISE™ software CORE Generator™ suite of reference designs and IP. The MIG design flow is very similar to the traditional FPGA design flow. The benefit for designers is that with this software tool, there is no need to generate RTL code from scratch for the physical layer interface or the memory controller.

To set system and memory parameters, you use the MIG's graphical user interface (GUI). For example, after selecting the FPGA device, package, and speed grade, you can select the memory architecture and even pick the actual memory device or DIMM (dual inline memory module). This same GUI provides a selection of bus widths and clock frequencies. Other options provide control of the CAS (column access strobe) latency, burst length, and pin assignments.

In less than a minute, the MIG tool generates RTL and UCF files (the HDL code and constraints files, respectively). These files are generated using a library of hardware-verified reference designs, with modifications based on your inputs. The output files are categorized in modules that apply to different building blocks of the design, such as user interface, physical layer, and controller state machine.

You also have complete flexibility to further modify the RTL code. Unlike other solutions that offer "black-box" implementations, the code is not encrypted, providing complete flexibility to change and further customize the design. After any optional code changes, you can perform additional simulations to verify the functionality of your overall design.

The MIG tool also generates a synthesizable test bench with memory checker capability. The test bench is a design example used in the functional simulation and hardware verification of the Xilinx reference design. The test bench issues a series of writes and read-backs to the memory controller. You can also use it as a template to generate your own custom test bench.

The final stage of the design is to import the MIG files in the ISE project, merge them with rest of your FPGA design files (followed by synthesis and place and route), run additional timing simulations if necessary, and perform hardware verification. The MIG software tool generates a batch file with the appropriate synthesis, map, and place and route options to help you optimally generate the final bit file.

Hardware Verification and Development Boards

Hardware verification of reference designs is an important final step to ensure a robust and reliable solution.

Xilinx has fully verified the implementation of a DDR2 SDRAM memory interface for Spartan-3A FPGAs in hardware. We implemented a DDR2 SDRAM design using a low-cost Spartan-3A starter kit board. Our design uses the on-board 16-bit-wide DDR2 SDRAM memory device and the XC3S700A-FG484 FPGA. The reference design uses only a small portion of the Spartan-3A device's available resources: 13% of IOBs, 9% of logic slices, 16% of BUFG MUXs, and one of the eight DCMs. This implementation leaves plenty of resources for other functions.

Xilinx FPGA	Spartan-3 FPGAs	Spartan-3E FPGAs	Spartan-3A FPGAs
Development Board	SL361	Starter Kit 3E	Starter Kit 3A
Memory Interfaces Supported	DDR	DDR	DDR2

Table 1 – Low-cost development boards for memory interfaces

Xilinx has verified memory interface designs for various Spartan-3 Generation FPGAs. Table 1 shows hardware-verified memory interfaces for each of the Spartan-3 Generation FPGA development boards.

Conclusion

You can speed up your memory interface and controller designs with low-cost Spartan-3 Generation FPGAs, the Memory Interface Generator (MIG) tool, and Xilinx development boards.

For more information and details on memory interface solutions, visit *www.xilinx.com/memory.*

FREE on-line tutorials with Demos On Demand

A series of compelling, highly technical product demonstrations, presented by Xilinx experts, is now available on-line. These comprehensive videos provide excellent, step-by-step tutorials and quick refreshers on a wide array of key topics. The videos are segmented into short chapters to respect your time and make for easy viewing.

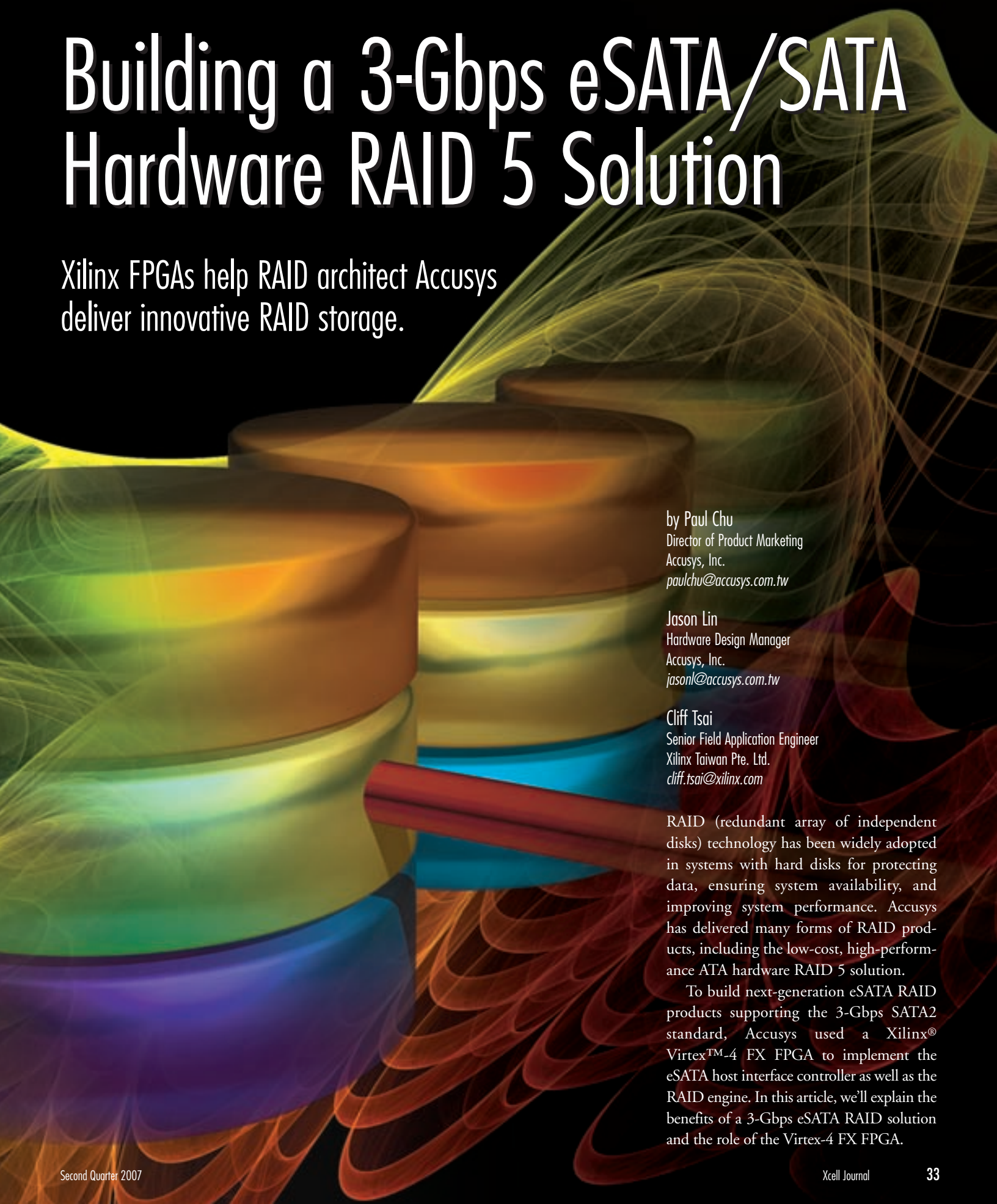
Ready for viewing, anytime you are

Offering live demonstrations of powerful tools, the videos enable you to achieve complex design requirements and save time. A complete on-line archive is easily accessible at your fingertips. Also, a free DVD containing all the video demos is available at **www.xilinx.com/dod**. Order yours today!

XILINX®

Building a 3-Gbps eSATA/SATA Hardware RAID 5 Solution

Xilinx FPGAs help RAID architect Accusys deliver innovative RAID storage.



by Paul Chu
Director of Product Marketing
Accusys, Inc.
paulchu@accusys.com.tw

Jason Lin
Hardware Design Manager
Accusys, Inc.
jasonl@accusys.com.tw

Cliff Tsai
Senior Field Application Engineer
Xilinx Taiwan Pte. Ltd.
cliff.tsai@xilinx.com

RAID (redundant array of independent disks) technology has been widely adopted in systems with hard disks for protecting data, ensuring system availability, and improving system performance. Accusys has delivered many forms of RAID products, including the low-cost, high-performance ATA hardware RAID 5 solution.

To build next-generation eSATA RAID products supporting the 3-Gbps SATA2 standard, Accusys used a Xilinx® Virtex™-4 FX FPGA to implement the eSATA host interface controller as well as the RAID engine. In this article, we'll explain the benefits of a 3-Gbps eSATA RAID solution and the role of the Virtex-4 FX FPGA.

Introduction to RAID

Different types of RAID algorithms offer different performance, reliability, and capacity. RAID 0 can improve I/O performance by concurrently distributing data over multiple disk drives, which also offers bigger space than the capacity of a single disk drive. However, if any of the disk drives in the RAID fails, the data will be lost. To protect the data, extra information associated with that data must be stored in case the original data has to be regenerated.

RAID 1 stores data to two disk drives simultaneously, so data is retained when one disk drive fails. But it is very costly because double storage capacity is required. RAID 5 offers the most cost-effective solution because only the capacity of one disk drive is needed for protecting data on any number of other disk drives. It also delivers superior performance because data is distributed to multiple disk drives, like RAID 0. As a result, RAID 5 is the most popular RAID level adopted in most of the storage systems deployed.

Advantages of Hardware RAID

RAID processing is performed either by software running on the host computer (software RAID) or by a dedicated RAID processor (hardware RAID). However, for most mission-critical systems or heavy-duty applications, a hardware RAID solution is preferred because it can deliver better performance with optimized hardware than software RAID, which consumes considerable resources on the host computer. Hardware RAID also provides superior reliability because it can regenerate the data of a disk drive in a shorter time; thus, the RAID can be recovered faster to a normal state. Therefore, mainstream RAID products are all implemented based on hardware RAID.

High-Speed and Low-Cost eSATA

One of the key features of a RAID solution is its host interface, by which the host computer accesses the storage presented by the RAID solution. There are two commonly used host interfaces: PCI-X/PCIe for RAID cards and SCSI family connections like Fibre Channel, or serial attached SCSI for RAID systems. Despite its high performance, using RAID solutions with these two interfaces is too troublesome for users without IT expertise because of the complicated installation procedure. On the other hand, the eSATA interface, which makes it easy to attach external storage devices, is gaining popularity in desktop computers.

The eSATA features a high-speed serial connection running at 3 Gbps and supports native command queuing (NCQ) for optimizing multi-tasking applications or multi-stream video processing. In addition, there is no need to install extra drivers or add-on cards for accessing eSATA devices, so its cost is lower and it is easier to use. As a result, RAID storage devices using eSATA as the host interface are good solutions in terms of performance and ease of use.

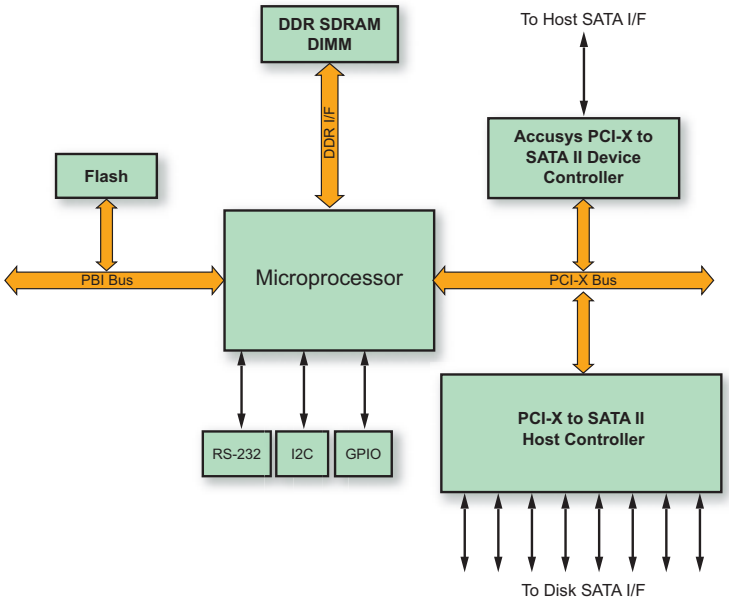


Figure 1 – Block diagram of SATA2-SATA2 RAID controller

Prior Art

Most existing eSATA RAID products on the market do not support hardware RAID 5, which largely restricts their application fields. The RAID implementation of these products are derived from SATA port multipliers, offering only simple packet dispatching and supporting only basic RAID levels like RAID 0 or RAID 1. RAID 0 can serve high-performance applications like video editing, but users are bothered by the routine backup and unexpected data losses caused by bad sectors and drive crashes. RAID 1 provides data protection, but the capacity wasted is too great and there is no performance benefit.

Unlike other eSATA RAID solutions on the market, the Accusys ACS-76000 RAID system on chip offers hardware RAID 5. However, its performance is limited by the 1.5-Gbps SATA1 speed. As high-definition media content and other applications become popular, a higher performance solution will be necessary.

Implementation Overview

To achieve higher performance, the first requirement is to upgrade the storage interface from SATA1 to SATA2, so you can achieve 3 Gbps and support NCQ. The second requirement is to have a high-performance and standard bus for connecting to a processor. We chose the PCI-X bus because it is commonly used in high-performance embedded systems. Finally, to support hardware RAID 5 processing, an exclusive OR (XOR) engine is needed to calculate the parity and regenerate data. Figure 1 is a block diagram of the SATA2-SATA2 RAID controller board.

We selected the Xilinx Virtex-4 FX FPGA because it provides the multi-gigabit serial I/O for implementing the 3-Gbps SATA host interface and can support the SATA device-mode controller logic for connecting to the digital parallel bus interface of the MGT hard core. We also imple-

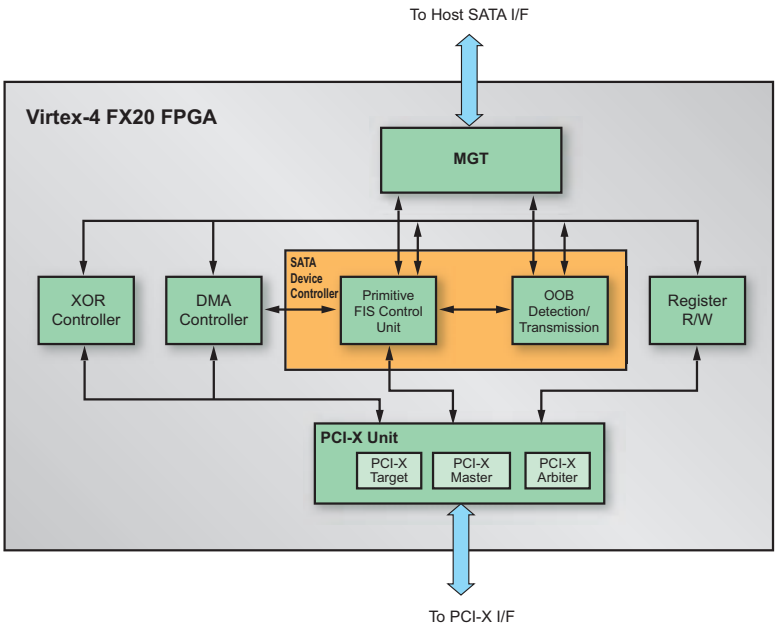


Figure 2 – Block diagram of 3-Gbps eSATA RAID engine

mented a 64-bit/133-MHz PCI-X controller and an XOR controller on the FPGA. A PCI-X-based external processor can parse commands from the SATA host and return status. In addition, the XOR controller can directly read the source memory data on the PCI-X bus and calculate the XOR data for the RAID 5 encoding. All of these tasks are done without the processor's intervention. Please refer to Figure 2 for a block diagram of the FPGA.

Implementing the 3-Gbps eSATA

Three different layers are defined in the SATA controller: the transport layer, link layer, and physical layer (PHY):

1. Transport layer. The SATA command execution and data transfer occurs by exchanging frame information structures (FISs), which contain ATA registers or disk data. The transport layer constructs FISs from the PCI-X bus for transmission, decomposes received FISs, and passes them to the PCI-X bus. In short, the transport layer is the interface between the link layer and the higher application layer, which in our design is the PCI-X bus.
2. Link layer. The link layer is responsible for packet framing, 8b/10b encoding and decoding, and generating and checking the CRC codes. The link layer also handles flow control, buffering data and exchanging primitives as needed to accommodate burst transfers. The Virtex-4 FX device's multi-gigabit transceiver (MGT) embeds the 8b/10b encoder and decoder, as well as a CRC-32 generation circuit, which provides tremendous convenience for building the SATA link layer.
3. Physical layer. The Virtex-4 FX FPGA's 6.5-Gbps RocketIO™ transceiver provides the basic building block of the SATA physical layer, responsible for de-serialization of received data from the host PHY and for serialization of outbound 10b encoded data from the link layer. It also supports out-of-band (OOB) signals for initializing the connection between the SATA host and device. To support external 3-Gbps SATA2, Gen2m electrical specifications are required in the implementation of the SATA physical layer.

Implementing the PCI-X Bus

The PCI-X controller includes both the master and target function of a 64-bit/133-MHz PCI-X device. It is equipped with a buffer RAM for storing data from the SATA transport layer and from the PCI-X bus for processing the split transaction flow. In addition, an embedded PCI-X arbiter is implemented to provide more flexible control for the PCI-X master. The DMA controller plays an important role in this design because it can provide an efficient bi-directional data transfer between FISs (from the SATA transport layer) and the PCI-X bus. As a result, the performance of bulky data transfer significantly improves.

Implementing the XOR Controller

For applications like video editing or recording, efficient XOR calculation is mandatory to meet the demanding requirements for low I/O latency and high throughput. The XOR controller is designed with as many as 256 data sources and 128 chained commands. Each of these can calculate 64-KB parity data. With the XOR controller implemented on the FPGA, the processor and the memory bandwidth are freed from performing the lengthy and routine XOR calculation on 256 sources with 128 chained commands at 64 KB per entry.

Conclusion

The 3-Gbps eSATA hardware RAID offers high performance, low cost, and ease of use. As more computers and embedded systems are shipped with eSATA ports, the eSATA RAID product represents an important external RAID solution for customers requiring both high performance and high reliability. Leveraging the advanced features of Virtex-4 FX FPGAs such as its digital clock managers and easy-to-use building blocks like dual-port block RAM, we developed the key component for the 3-Gbps eSATA RAID system quickly and efficiently.

With our success delivering products based on Xilinx Virtex-4 FX FPGAs, Accusys is positioned to deliver eSATA RAID products for high-end applications like video editing and workgroup-shared storage, as well as entry-level applications like DVR backup or home storage. For more information about RAID and storage technologies, visit the Accusys website at *www.accusys.com.tw*.

Implementing a Real-Time Beamformer on an FPGA Platform

We designed a flexible QRD-based beamforming engine using Xilinx System Generator.

by Chris Dick
Xilinx Chief DSP Architect
Xilinx, Inc.
chris.dick@xilinx.com

Fred Harris
Professor
San Diego State University
fred.harris@sdsu.edu

Miroslav Pajic
Engineer
Signum Concepts
miroslav.pajic@signumconcepts.com

Dragan Vuletic
Engineer
Signum Concepts
dragan.vuletic@signumconcepts.com

Most real-world communication systems have a mix of processing elements. For example, application programs, human/machine interface management, and higher networking protocol stack processing are best implemented on a general-purpose processor.

But for high-rate, algorithmically complex data processing – often with hard real-time deadlines – hardware resources like FPGAs are a better match. The interface between the two depends on the circumstance; the FPGA can be a pre-processor, coprocessor, post-processor, or some combination thereof. The trick is to get these heterogeneous systems to interoperate in an elegant fashion.

In this article, we'll describe the development of a flexible, optimized, adaptive beamforming engine that you can easily control through software. The DSP-intensive tasks run on the FPGA, while the command and control run on an external processor. The beamforming engine is a compact QR decomposition (QRD-based circuit) with a novel construction. The interface between the engine and the host processor is implemented by the shared memory abstraction in the Xilinx® System Generator design flow.

MVDR Beamformer

Adaptive beamforming is the application of adaptive filters to spatial signal processing. Time series collected from uniformly spaced array elements are weighted and summed to form a signal component from a selected direction of arrival while suppressing signal components from other directions of arrival (Figure 1). When the directions of arrival of the undesired signal components are unknown or vary with time, the filter weights must be adaptively adjusted to steer nulls to their directions. The adaptation process is performed subject to a constraint that the steering vector has unity gain in the signal direction. The steady state weights of such a beamformer form the minimum variance distortionless response (MVDR) from the array elements.

For reasons of numerical robustness and computational complexity, a common method for computing the required weight vector without directly inverting the correlation matrix is based on QR decomposition; this is the approach adopted here. For details of the procedure, consult "Adaptive Filter Theory" by Simon Haykin.

The QRD Matrix Inversion Process

The QRD process is formed by a sequence of two operators: the unitary rotations that convert complex input data to real data and associated angle-and-element combiners that individually annihilate the selected elements of the input data set. The QRD process is most compactly represented in Figure 2's signal flow diagram. This representation is the systolic array realization of the QRD least-squares solution processor.

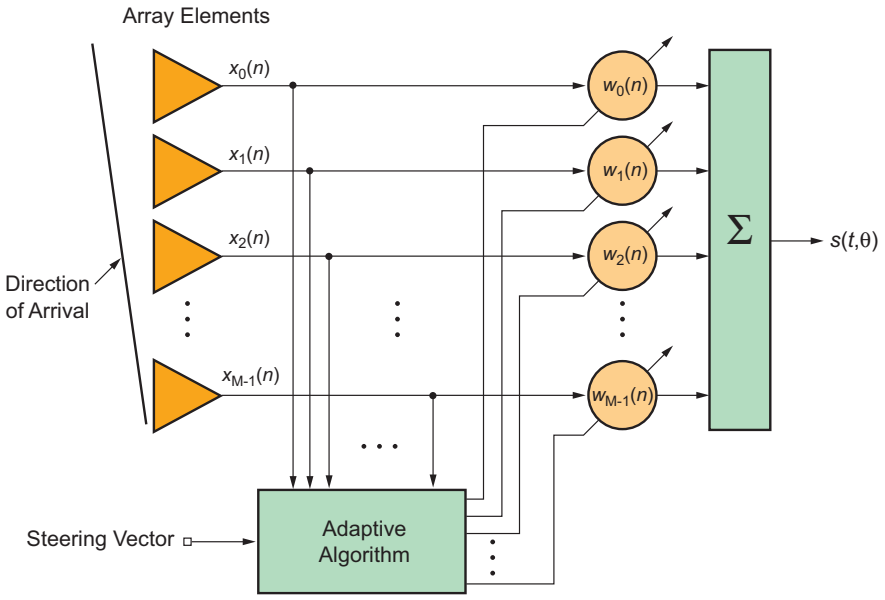


Figure 1 – Adaptive beamformer structure

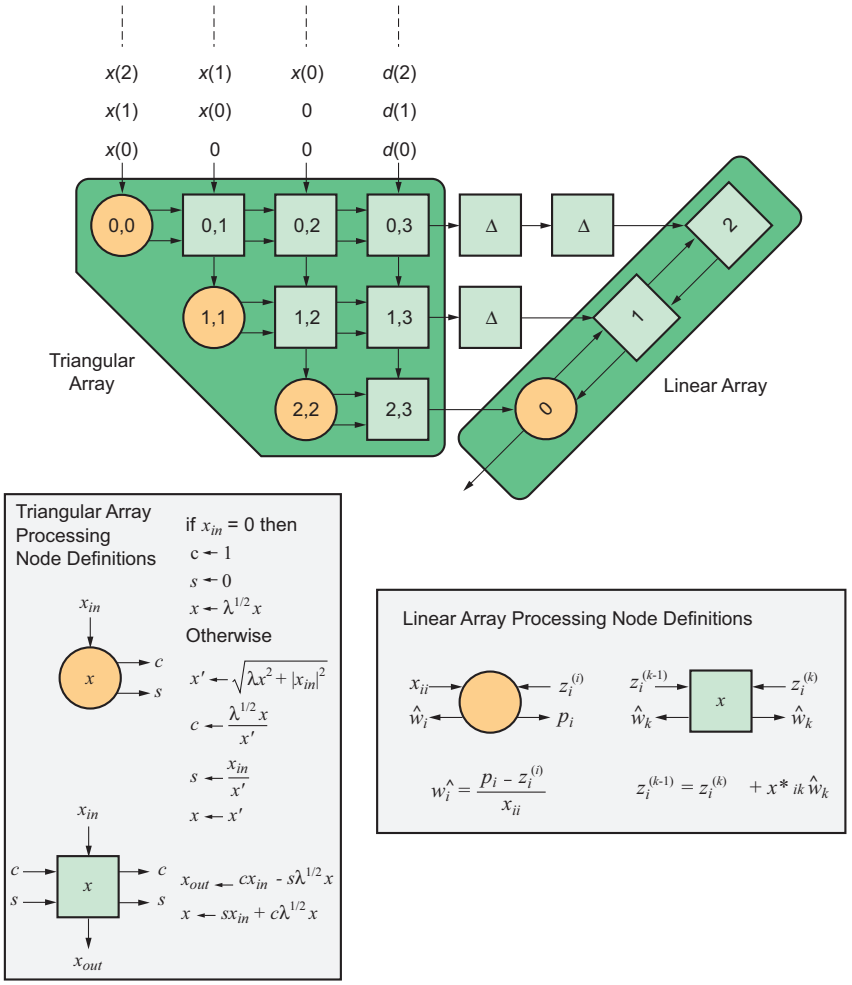


Figure 2 – Systolic array implementation of QRD matrix inversion for a 3 x 3 array

The array contains three types of processing cells: boundary cells, internal cells, and output cells. The boundary cells perform the "vectoring" operation on complex input samples to nullify their imaginary parts and form rotation angles used by internal cells. The internal cells perform Givens rotations of the input values by the angles passed from the boundary cells to annihilate the non-upper-triangular entries of the transformed data matrix. The output cells in the linear array process the elements of the upper triangular array to perform the required back substitution that produces the beamformer weights.

FPGA Implementation of QRD

Our goal was to produce a compact QRD FPGA implementation. The design comprises a single boundary, internal and back substitution cells. The systolic array in Figure 2 is folded onto this set of processing resources. The boundary cell is required to compute two angles. The first angle

$$\Phi = \arctan(\Im(x_{in})/\Re(x_{in}))$$

transforms the complex input samples presented to the boundary-cell input port into real-valued data. The transformation that forces the imaginary component of x_{in} to 0 must be applied to all elements in the same row associated with the boundary cell; this operation is one of the tasks performed by the internal cells.

Now that the data in the leading position of two adjacent rows are real-valued, a second angle is formed as

$$\Theta = \arctan(x_{in}e^{-j\Phi}/x)$$

which is used to annihilate a term of the input data set in an ordered manner that eventually produces the upper-right triangular matrix R. The arithmetic employed in the boundary cells could be realized in hardware by literally implementing the equations indicated in Figure 2. This would require hardware support for performing square roots and divisions. Although these circuits are commonly implemented in FPGA hardware, we sought alternative methods for computing the required angles that had a lower resource cost over direct and obvious implementations.

One well-known and relatively simple method for computing angles is the vectoring mode of the Coordinate Rotation Digital Computer (CORDIC) algorithm. The CORDIC algorithm is an iterative procedure capable of computing a rich set of mathematical functions. The elemental operations required in the CORDIC algorithm are addition, subtraction, bit-shift, and table lookup. All of these functions are efficiently supported by FPGA architectures such as the Virtex™ series of devices from Xilinx, so the vectoring mode of the algorithm is a good candidate for the foundation of QRD processor boundary cells. As shown in Figure 3, two CORDIC engines are in use in the boundary cell: one for computing ϕ and the other for computing θ.

The CORDIC algorithm is iterative in nature, with each iteration refining the angle estimate by approximately 1 bit of precision. For a process employing an N-iteration CORDIC process, a new output is generated every N clock cycles and a new set of operands presented every N clock cycles. To increase the throughput of the boundary cell, we employed a fully parallel, or unrolled, architecture for the CORDIC (not shown here). After the initial start-up latency of the circuit is absorbed, the initiation

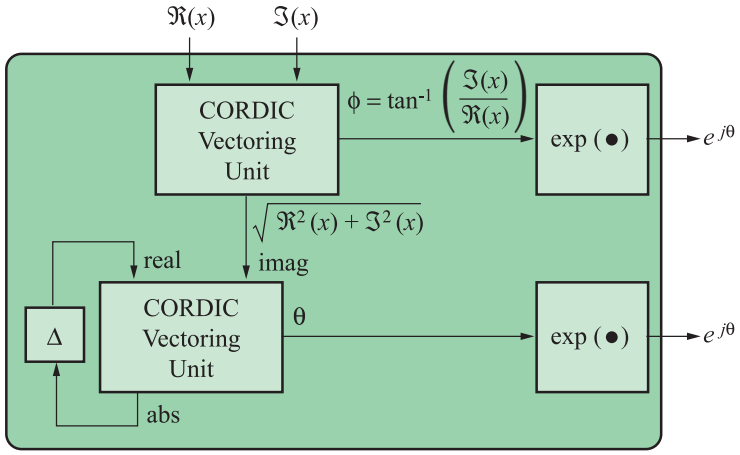


Figure 3 – Boundary-cell architecture based on two vector-mode CORDIC processing engines

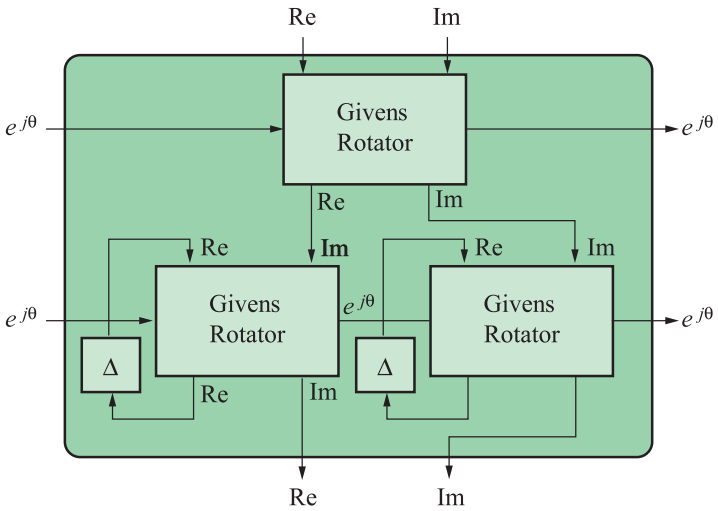


Figure 4 – Systolic array internal cell architecture employing three MAC-based Givens rotation engines

and completion rate of the cell is one new input/output per clock cycle.

Each data element x_{in} entering an internal cell (Figure 4) in row m must be rotated by the angle ϕ computed by the boundary cell for the m^{th} row:

$$\begin{bmatrix}\Re(\upsilon)\\ \Im(\upsilon)\end{bmatrix} = \begin{bmatrix}\cos(\phi) & -\sin(\theta)\\ \cos(\phi) & \sin(\theta)\end{bmatrix}\begin{bmatrix}\Re(x_{in})\\ \Im(x_{in})\end{bmatrix}$$

One option that has been commonly used for the rotation task in QRD processors is the rotation mode of the CORDIC algorithm. An alternative is to simply implement the rotation in the obvious manner using multiply accumulate (MAC) functional units; this is the approach adopted in our implementation. The target FPGA technology for the design is the Virtex-4 FPGA. These devices have a vast array of embedded MAC units referred to as DSP48 slices.

The DSP48 slice supports a rich set of opcodes – capable of being updated on a per-clock-cycle basis – that define the arithmetic operation computed by the tile during a given clock cycle. The four multiplications implied in the preceding equation are folded onto a pair of DSP48 slices, with each DSP48 slice computing one of the two output terms $\Re(\upsilon)$ and $\Im(\upsilon)$. Two clock cycles are required to compute the two output terms. Each DSP is supplied with a unique opcode for each clock period. Consider computing the term $\Re(\upsilon)$. During the first clock period, the product $\cos(\Phi)$ x $\Re(x_{in})$ is computed and stored in the DSP48 product or p register.

During the second clock cycle, $\sin(\Phi)$ x $\Im(x_{in})$ is formed and subtracted from the value in the p register to generate the final output term. A similar sequence of computations is performed to produce $\Im(\upsilon)$. Using the DSP48 embedded blocks rather than a CORDIC-based approach for the internal cell reduces the latency of this phase of the computation and also minimizes the amount of FPGA logic fabric (look-up tables [LUTs] and registers) required for the implementation. Table 1 provides a breakdown of the area for the major functional units in the QRD implementation, along with the total area of the design.

The $\cos(\Phi)$, $\sin(\Phi)$, $\cos(\Theta)$, and $\sin(\Theta)$ terms required by the internal cells are computed using a simple LUT that maps the angles Φ and Θ computed by the vectoring units in the boundary cell to their corresponding sine and cosine. Linear interpolation is applied to the output samples of the LUT to increase the accuracy of the mapping from angle to amplitude, while keeping the LUT itself constrained to a single block RAM.

The row and column dimensions of the input array for the QRD processor can be dynamically adjusted at runtime by writing the new dimensions to control registers that are part of the FPGA control plane.

Table 2 provides timing information for several configurations of the input data set.

Functional Unit	LUTs	FFs	DSP48 Slices	Block RAM	Slices
Boundary Cell	2,145	2,057	3	1	1,266
Inner Cell	216	329	6	0	176
Back Substitution	2,862	3,286	4	1	1,932
QRD Total	5,411	5,916	13	6	3,530

Table 1 – FPGA resource utilization for folded QRD and back substitution array

M	N	Cycles for Triangularization	Cycles for Back Substitution	Total Cycles	Time (μs) for 250-MHz Clock
3	3	792	147	939	3.76
8	3	2,112	147	2,259	9.04
5	5	2,540	255	2,795	11.18
9	5	4,572	255	4,827	19.31
7	7	5,656	371	6,027	24.11
10	7	8,080	371	8,451	33.80
9	9	10,476	495	10,971	43.88
11	9	12,804	495	13,299	53.20
10	10	13,630	560	14,190	56.76

Table 2 – Execution time for the triangularization and back substitution phases of the FPGA QRD implementation for an M x N matrix.

Design Flow

Our QRD implementation uses the Xilinx System Generator for DSP model-based design flow. In addition to providing a natural development environment for developing FPGA signal-processing implementations, System Generator has a rich set of features that support the development of heterogeneous applications comprising not just the FPGA element but a processor. The processor could be the embedded PowerPC™ 405 hard IP block, the MicroBlaze™ soft-processor core, or a processor external to the FPGA.

The beamformer developed for this project was partitioned between the host PC and the FPGA platform. In our implementation the host application running on the PC might be considered more of an element of the beamformer verification process (test bench), but the host applica-

tion could be as arbitrary and complex as required by the task at hand.

Our beamformer host application is a MATLAB script (m-code) that simulates the sensor array for the beamforming network. The script simulates a dynamic target and generates the samples of the far-field radiation pattern for the moving target. The samples of the electric field at each sensor are generated in MATLAB and forwarded to the FPGA QRD processor. A new estimate of the beamformer weight vector is produced and returned to the MATLAB environment for further processing.

In this case, the additional processing involves plotting the polar radiation pattern for the updated complex valued weight vector. Note that the host application does not necessarily have to be associated with the MATLAB environment; the application could be a program written in C, for example.

An interesting element of the beamformer application is the management of the interface between the host application, running on a PC in this case, and the QRD process executing on the FPGA platform. System Generator provides a suite of shared memory library objects (ROM, RAM, FIFO) that abstracts virtually all of the details of the processor/FPGA interface and enables the host software and FPGA hardware to be somewhat insulated from each other (Figure 5).

Each new update of the beamformer is essentially a three-step procedure:

1. New input samples from each antenna element, as generated by the MATLAB host application, are forwarded to the QRD engine in situ on the FPGA.
2. The QRD process is triggered.
3. The new weight vector is returned from the FPGA to the host.

The shared memory library modules and associated application programmer interface transform the transfer of data between the FPGA and the host PC into simple assignment statements based on name/space references in MATLAB (or C). For example, the new weight vector w, resident in the MATLAB workspace, is updated with the new beamformer coefficients, FPGAWeights, as computed by the FPGA QRD process using the simple assignment w = FPGAWeights. (FPGAWeights is the name assigned to a shared-memory buffer in the System Generator description of the QRD engine.)

The management of this type of host processor/FPGA interaction by the System Generator framework makes the development of heterogeneous applications straightforward, rapid, less error-prone, and enables an FPGA accelerator engine (like the QRD module in this case) to be easily ported between different hardware platforms without needing to modify the FPGA source code – the System Generator model itself.

The interface abstraction supports transactions between the host application and the System Generator source model, as well as the host application and the final design running on the FPGA platform. This latter element significantly contributes to the validation process of the software and hardware (FPGA) dimensions of the system, as both components can be brought online rapidly using the shared memory abstraction.

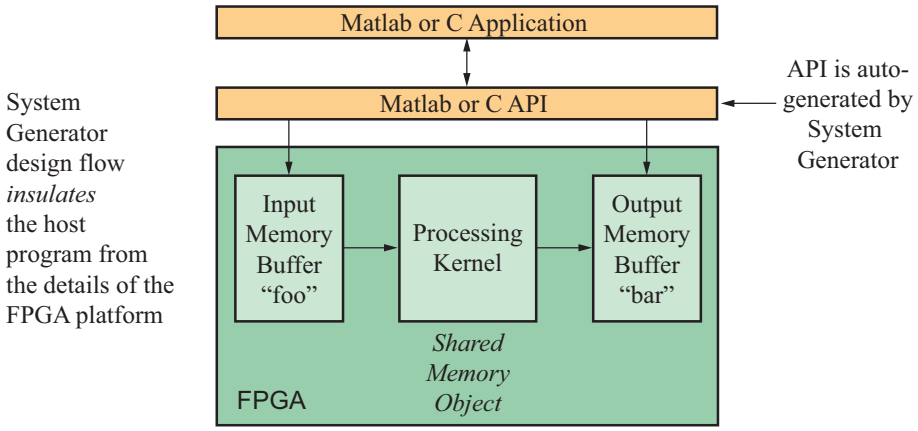


Figure 5 – Hardware/software abstraction enabled by the System Generator shared memory library elements. The host application performs transactions with the FPGA storage elements using simple name/space references – in this case to the memories named "foo" and "bar."

Conclusion

In this article, we've described the FPGA implementation of a flexible QRD processor that enables the run-time definition of the input matrix dimensions. The design employs a mixture of CORDIC-based processing (array boundary cell) and MAC-based (array internal cell) arithmetic that is well matched to the computational resources of an FPGA like the Xilinx Virtex-4 family.

All of the boundary- and internal-cell processing were projected onto a single boundary-cell functional unit and internal-cell functional unit; however, it should be noted that the abundant resources of FPGA platforms support the realization of a fully parallel systolic array, should the throughput requirements of the target application demand extremely high performance.

The System Generator programming environment enables the rapid development of heterogeneous systems (processors and FPGAs) while insulating programmers from the frequently complex and error-prone programming associated with hardware/software partitions.

This work was performed by the Xilinx Advanced Systems Technology Group (ASTG), the R&D organization within the Xilinx DSP Division, together with our partner organizations Signum Concepts and San Diego State University.

The Ultimate Embedded Solution

for Xilinx MicroBlaze™

With a rich debugging environment and a small, auto-configurable RTOS, Accelerated Technology provides the ultimate embedded solution for MicroBlaze development. Our scalable **Nucleus** RTOS and the Eclipse-based **EDGE** development tools are optimized for MicroBlaze and PowerPC processors, so multi-core development is easier and faster than ever before. Combined with world-renowned customer support, our comprehensive product line is exactly what you need to develop your FPGA-based device and get it to market before the competition!

For more information on the Nucleus complete solution, go to
www.acceleratedtechnology.com/xilinx

Accelerated Technology, A Mentor Graphics Division
info@AcceleratedTechnology.com • www.AcceleratedTechnology.com

Accelerated Technology®
A Mentor Graphics Division

Using CRC Hard Blocks in Virtex-5 FPGAs

CRC blocks provided as hard macros speed up the process of error detection.

by Sunita Jain
Associate Design Engineer
Xilinx India Technology Services Pvt. Ltd.
(Xilinx Hyderabad, XHD)
sunita.jain@xilinx.com

Guru Prasanna
Technical Lead
Xilinx India Technology Services Pvt. Ltd.
(Xilinx Hyderabad, XHD)
guru.prasanna@xilinx.com

Data corruption is the principal problem associated with data transmission and storage. Whenever data is transmitted over channels, there is always a finite probability that some errors will occur.

It is essential that the receiving module can differentiate between an error-free message and an erroneous one. There are various methods to detect errors; most error-checking methods do so by introducing redundant bits exclusively for this purpose. Commonly used methods for error detection in data communication include parity codes, Hamming codes, and cyclic redundancy check (CRC); of these, CRC is the most widely used.

CRC is computed on a given set of data bits and appended at the end of the data frame before transmission or storage. When the frame is received or retrieved, its validity is verified by recalculating the CRC for the contents of the frame to ensure that the data is error-free.

In this article, we'll take a quick look at the theory behind CRC calculation and its hardware implementation using linear feedback shift registers. Then we'll direct our attention to the CRC hard block present in Xilinx® Virtex™-5 LXT/SXT devices.

CRC is based on polynomial codes defined over a field with two elements. Polynomial codes treat bit streams as polynomial representations with coefficient values of either 0 or 1.

Theory

Addition and subtraction operations are performed using modulo 2 arithmetic; that is, they are the same as the exclusive OR (XOR) operation. Adding two numbers in polynomial arithmetic is the same as adding numbers in ordinary binary arithmetic, except there is no carry.

For example, the binary message stream 11001011 is represented as $x^7+x^6+x^3+x+1$. The transmission and reception points agree on a fixed polynomial called a generator polynomial; this is the key parameter of a CRC calculation.

The data is interpreted as the coefficients of a polynomial, which are divided by a given generator polynomial. The remainder of this division is the CRC. Given an m-bit message sequence and a generator polynomial of degree r, the transmitter creates an n-bit (n = m+r) sequence called the frame check sequence (FCS) so that the resultant (m+r)-bit frame is divisible by a predetermined sequence.

The transmitter appends r 0-bits to the m-bit message and divides the resulting polynomial of degree m+r-1 by the generator polynomial. This produces a remainder polynomial of degree (r-1) or less. The remainder polynomial has r coefficients, which form the checksum. The quotient is discarded. The data transmitted is the original m-bit message followed by the r-bit checksum.

At the receiver, you can follow one of two standard approaches to assess the validity of the received data:

- Compute the checksum again for the first m-bits received and compare against the received checksum (the last r-bits received)
- Calculate the checksum for all of the (m+r) received bits and compare against a remainder of 0

To see how the second method results in a remainder of 0, let's use the following convention:

M = polynomial representation of the message

R = polynomial representation of the remainder computed at the transmitter

G = generator polynomial

Q = quotient obtained by dividing M by G

The transmitted data corresponds to the polynomial $Mx^r - R$. The variable x^r signifies an r-bit shift of the message to accommodate the checksum.

We know that:

$$Mx^r = QG + R$$

Appending the checksum R to the message at the transmitter is equivalent to subtracting the remainder from the message. The transmitted data then becomes $Mx^r - R = QG$, which is clearly a multiple of G. This is how we obtain a remainder of 0 in the second case.

However, this procedure is insensitive to the number of leading and trailing 0-bits in the data transmitted. In other words, if a message has trailing 0-bits inserted or deleted, the remainder will still remain 0 and the error will go undetected, which manifests that the same bit sequence will not be reproduced back. Let's explain a tactical method to overcome this drawback.

Residue Approach

In practice, the checksum is complemented before appending. This makes the remainder calculated (over m+r bits) at the receiver non-0. The remainder, which is obtained at the receiver in such cases, is a fixed value known as the residue value of a polynomial.

A little mathematics helps illustrate this idea more clearly.

Assume that the % symbol denotes a modulo operation in the following representation:

For the case where checksum is appended without inversion:

$$(Mx^r - R)\ x^r \% G = 0$$

where the receiver again performs the same operation of shifting as the transmitter.

Now consider the case where the checksum is inverted and appended to the message stream at the transmitter:

$$(Mx^r - R^c)\ x^r \% G$$

where R^c denotes complemented checksum.

This can also be written as:

$$(Mx^r - R + (x^{r-1} + \ldots + x + 1))\ x^r \% G$$

The complement of a bit is the same as XOR with 1. Here, the + sign represents addition in modulo 2 arithmetic (also note that addition and subtraction are the same in modulo 2 arithmetic).

In which case, the remainder is the same as:

$$(x^{r-1} + \ldots + x + 1)\ x^r \% G$$

which works out to be a constant for a given generator polynomial.

The most commonly used CRC-32 generator polynomial is

$$G(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$$

which is 04C11DB7 in hexadecimal.

The constant residue value corresponding to CRC-32 is C704DD7B in hexadecimal. For a given generator polynomial G, the residue value remains a constant for any data pattern provided at the input.

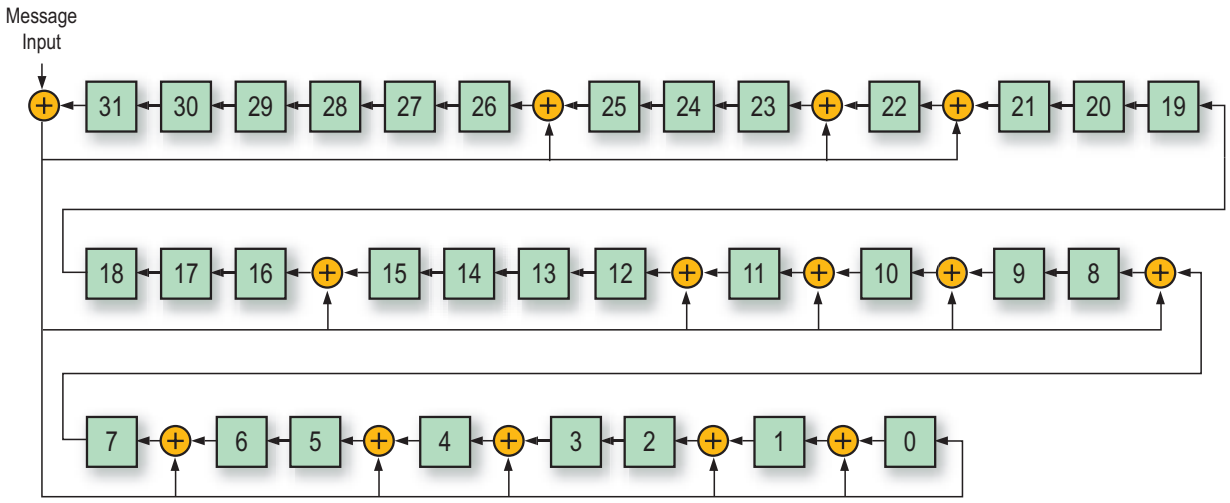


Figure 1 – LFSR implementation for the CRC32 polynomial

Hardware Implementation

The computation of a CRC checksum is a polynomial division process. Implementing it in hardware makes use of a shift register (also called the CRC register). The length of the shift register is the same as the degree of the generator polynomial.

The procedure for CRC calculation is to:

1. Initialize the CRC register.
2. Get the message bit and continue as long as message bits exist. If the higher order bit in the CRC register is 1, shift left one position and XOR the result with G. Otherwise, shift left one position.

When all of these steps are completed for a given message, the CRC register holds the remainder.

These steps can be implemented with a circuit known as the linear feedback shift register (LFSR). Figure 1 shows an LFSR implementation for calculating CRC using the CRC32 polynomial. Note that the placement of XOR gates depends on the coefficient of the corresponding terms being 1 in the generator polynomial. Each of the numbered blocks in the figure represents a memory element (flip-flop).

CRC Block

The hardware implementation of CRC makes use of a simple LFSR. Although such a circuit is simple to implement, it takes n-clock cycles to calculate CRC values for an n-bit data stream. This latency is intolerable in high-speed data networking applications where data frames must be processed at higher speeds. The implementation of CRC generation and checking on a parallel stream of data become desirable in such high-speed networking applications.

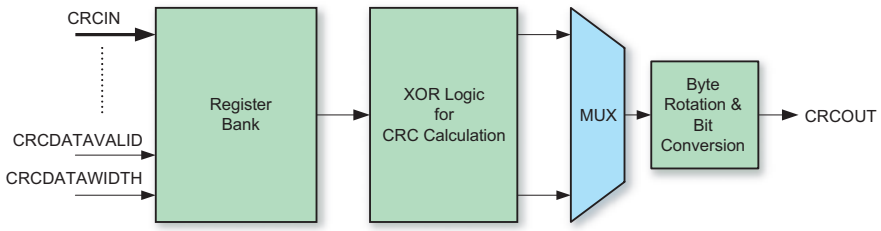


Figure 2 – CRC hard block implementation

The CRC block implemented in Virtex-5 LXT/SXT devices helps designers by speeding up checksum calculations.

The CRC hard block in Virtex-5 LXT/SXT devices is based on the CRC32 polynomial. Virtex-5 FPGAs contain CRC32 and CRC64 hard blocks, which provide a latency of one clock cycle for CRC generation for 4- and 8-byte data input. The interface is easy and simple to use. The hard block functions as a CRC calculator on a given message stream, along with some CRC-specific parameters as input. The functionality for CRC comparison is beyond the scope of the hard block and should be built into the FPGA fabric.

Each CRC hard block in the FPGA computes a 32-bit checksum asynchronously. Figure 2 is a block-level diagram depicting the hard block's architecture. The CRC hard block provides an output, which is bit-inverted and byte-reversed.

Figure 3 provides an application overview of the CRC hard block. CRC is calculated and appended at the end for a given data packet at the transmitter. At the receiver, CRC is re-calculated over the entire received packet along with the appended CRC from the transmitter.

The validity of the received packet is established by the residue method. In this case, for the CRC32 polynomial, the

residue works out to be 1CDF4421 in hexadecimal, as it is a bit-inverted and byte-reversed value for C704DDB7. The concept of byte rotation and bit inversion is shown in Figure 4.

Figure 5 illustrates a waveform for normal CRC operation.

We have also come up with a LogiCORE™ CRC wizard, which provides a LocalLink wrapper for the CRC hard macro present in Virtex-5 devices. The core also provides an example design demonstrating the use of the CRC hard block. Additionally, the core provides various options such as pipelining, complementing, and transposing.

For more information, visit *www.xilinx.com/crcwizard*.

Conclusion

The presence of CRC blocks in Xilinx FPGAs makes the inclusion of error-detection mechanisms in various designs easier and effortless for designers. You can use the CRC Wizard IP, available in CORE Generator™ software, to incorporate error-detection in various protocols like Aurora and PCI Express.

For more information on the embedded CRC blocks in Virtex-5 FPGAs, see the Virtex-5 RocketIO™ GTP Transceiver User Guide (UG196) at *www.xilinx.com/bvdocs/userguides/ug196.pdf*.

The theory on CRC presented in this article is a summary of the CRC theory published in various technical journals and textbooks. Please contact the authors for applicable references and URLs.

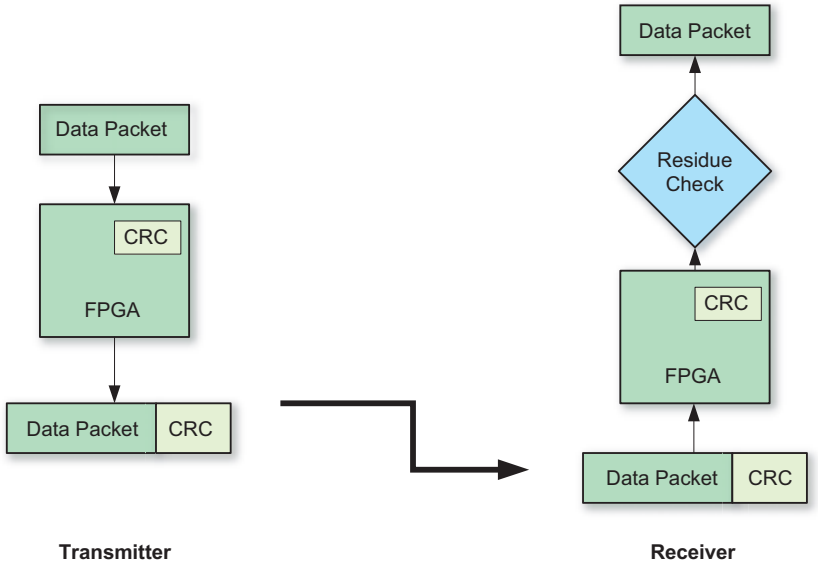


Figure 3 – Application overview of CRC block

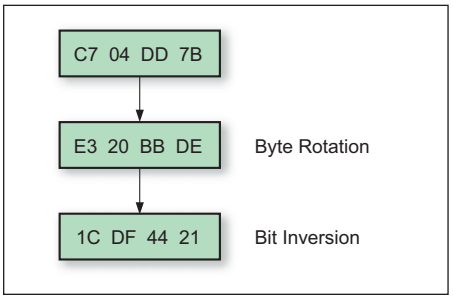


Figure 4 – Byte rotation and bit inversion depiction

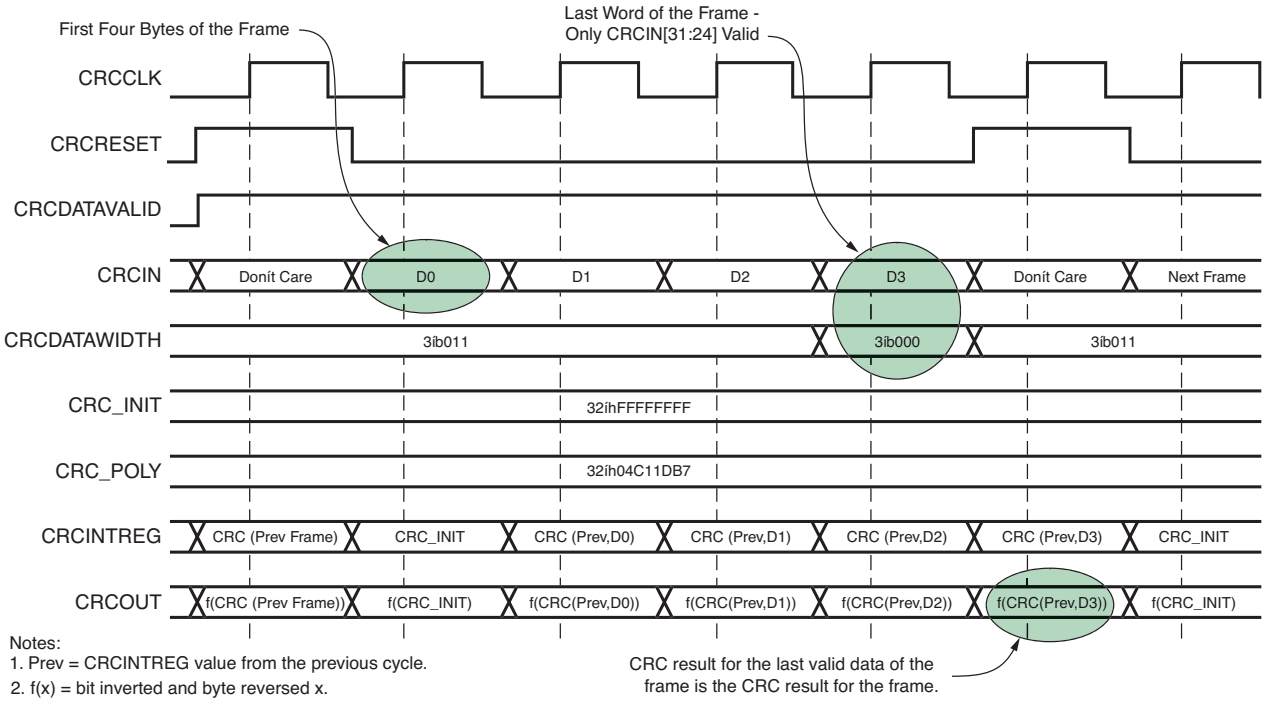
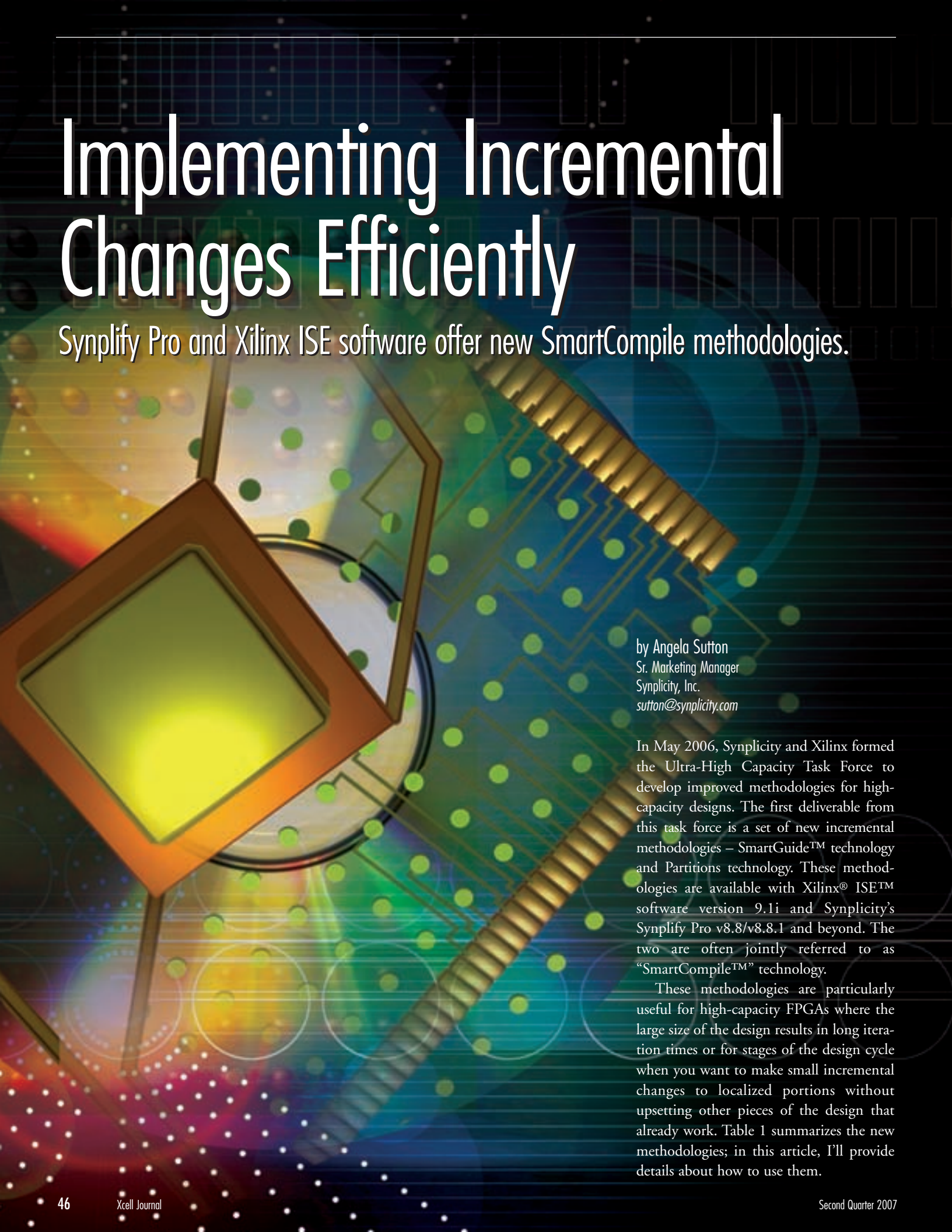


Figure 5 – Waveform for CRC operation

Implementing Incremental Changes Efficiently

Synplify Pro and Xilinx ISE software offer new SmartCompile methodologies.



by Angela Sutton
Sr. Marketing Manager
Synplicity, Inc.
sutton@synplicity.com

In May 2006, Synplicity and Xilinx formed the Ultra-High Capacity Task Force to develop improved methodologies for high-capacity designs. The first deliverable from this task force is a set of new incremental methodologies – SmartGuide™ technology and Partitions technology. These methodologies are available with Xilinx® ISE™ software version 9.1i and Synplicity's Synplify Pro v8.8/v8.8.1 and beyond. The two are often jointly referred to as "SmartCompile™" technology.

These methodologies are particularly useful for high-capacity FPGAs where the large size of the design results in long iteration times or for stages of the design cycle when you want to make small incremental changes to localized portions without upsetting other pieces of the design that already work. Table 1 summarizes the new methodologies; in this article, I'll provide details about how to use them.

Incremental Methodology	When to Use	How it Works	Advantage
SmartGuide	**Small incremental RTL changes to non-critical paths**	When you make a small change to your RTL and re-run synthesis. ISE software detects netlist changes and runs incremental PAR based on those netlist changes alone and timing goals.	Design preservation and consistency from one run to the next Saves PAR runtime (up to 70%) Easy to use – no need to create partitions or block-based constraints
Partition	Useful for **block-based or team-designed methodologies.** Allows you to get more stable results from one run to the next by locking down individual blocks.	Fully integrated block-based design methodology, synthesis through place and route. Blocks ("partitions") defined at RTL level in Synplify Pro. Block definitions and subsequent block changes automatically communicated between Synplify and ISE software.	Teams can work on and tune specific RTL blocks in parallel Blocks that already work can be preserved

Table 1 – New Synplicity/Xilinx incremental methodologies

SmartGuide

SmartGuide minimizes changes in a design's place and route (PAR) implementation compared with previous implementations used as a reference. Typically, you would synthesize the design to get an output netlist, which ISE software places and routes. You would then make changes to the RTL, rerun synthesis, and rerun ISE incremental place and route with SmartGuide enabled.

If you elect to use SmartGuide, ISE software place and route attempts to do the minimal changes based on what changed in the netlist, while still maintaining a legal netlist and meeting timing. This saves significant PAR runtime. In this methodology, there is no need to manually create partitions, nor are constraint or synthesis changes involved when using this capability (Figure 1).

How Does SmartGuide Work?

ISE software runs fully automatic incremental place and route on an entire design based on a netlist "name match" comparison (the current netlist output by Synplicity's Synplify Pro tool versus the initial netlist). The Synplify Pro software's ability to generate consistent and repeatable netlists has been key for this to work well. Synplify Pro software v8.8 employs many new techniques to maximize netlist consistency. These include:

- Repeatability of instance names from one synthesis run to the next
- Output netlist changes that are localized to the very specific paths where the RTL changed from one synthesis run to the next

Xilinx results with Synplify Pro v8.8 software and ISE software version 9.1i showed that, for small RTL changes where the change was not on the critical path, place and route runtimes were halved and in many cases as much as 70% shorter. More than half of the test designs preserved 97% of placement and routing, leading to very consistent designs.

SmartGuide is best used when you need to make small changes to your RTL and those changes are not on a critical path. In such cases, SmartGuide can cut PAR time in half on average compared with running place and route again on the entire design.

Examples of a small changes include changing things that result in less than 10% of the netlist changing – a change in the value of an RTL constant, changing a logical OR to an AND, changing the logic condition on a "if" statement, or minor changes to a state machine. Also, this flow is recommended for roughly 10-15 consecutive incremental runs, after which you may want to rerun place and route on the entire design.

Partition

Partition is an incremental block-based design methodology – it allows place and route to run incrementally on blocks that are being modified or tuned while preserving other blocks that have not changed. This preservation can occur all the way down to the routing level. This approach results in shorter runtimes.

Blocks, also referred to as "partitions," are defined at the RTL level before running synthesis. The block hierarchy is specified

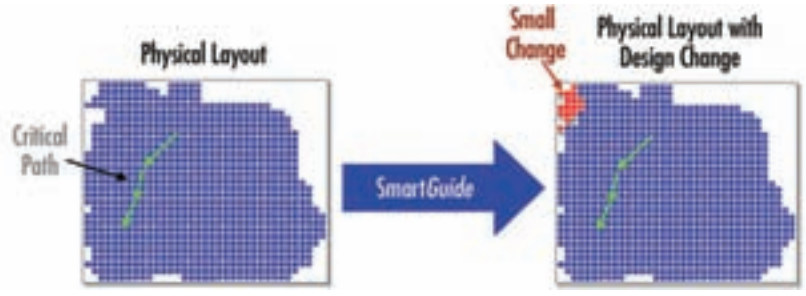


Figure 1 – SmartGuide can halve your place and route runtime. Small changes to RTL/constraints result in small changes to the physical layout, especially when the changes were not on the critical path.

ISE software and Synplify Pro have been integrated so that ISE software automatically determines which blocks/subblocks changed using partition date-stamp information.

as Synplify Pro compile points. Each time a block is re-synthesized it is date-stamped and updated by Synplify Pro in the output EDIF netlist and communicated to ISE software. ISE software automatically detects which blocks have changed by reading the EDIF time stamp.

To run the partition methodology (illustrated in Figure 2):

1. Set compile points. Define blocks/subblocks (partitions) as compile points in Synplify Pro. For example:

```
#
# Define Compile points in Synthesis .sdc file.
# You need to separately define time budgets
#
define_compile_point {v:work.or1200_cpu}
-type {locked, partition} -cpfile {}
define_compile_point{v:work.or1200_dc_top}
-type {locked, partition} -cpfile {}
```

   You can specify your compile points in the GUI using the SCOPE editor or the Synplify Pro command line.

2. Implement the design (original PAR layout) and run synthesis. At the end of synthesis, Synplify Pro writes an extra property into the EDIF that includes the time stamp for when the compile point (block/partition) was last modified. For example:

```
(instance or1200_dc_top (viewRef verilog
(cellRef or1200_dc_top))
(property PARTITION (string "1169730682"))
)
```

   In subsequent synthesis runs, the old time stamp will be inserted in the EDIF for blocks/partitions that did not change. A new time stamp is inserted in blocks/partitions that did change.

   You can then run ISE place and route on the output from Synplify Pro software. ISE software automatically reads the Synplify Pro compile points as partitions. ISE software also notes and stores the EDIF date stamp for each block/partition.

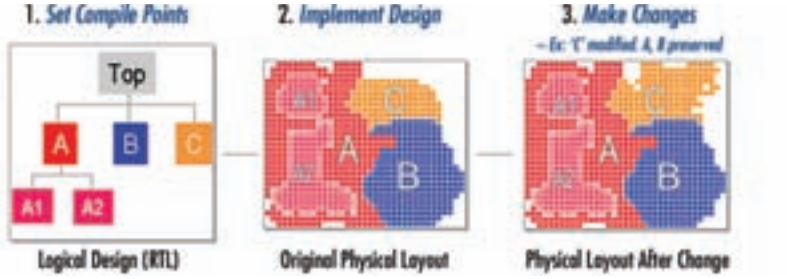


Figure 2 – The Synplify Pro/ISE Partition is a fully integrated (RTL through physical layout) block-based design methodology.

3. Make changes. Tune individual blocks (for example, RTL or constraints) in Synplify Pro and resynthesize, then rerun ISE place and route. ISE software and Synplify Pro have been integrated so that ISE software automatically determines which blocks/subblocks changed using partition date-stamp information. ISE runs place and route on the changed partitions and leaves other placed/routed blocks untouched (as long as timing can still be met).

   You can control whether ISE software leaves placement alone on the unchanged blocks or both placement and routing using the partition property setting. Each partition has a customizable level of preservation with four values: "synthesis" (preserve the netlist), "placement" (preserve the netlist and placement), "routing" (preserve netlist, placement, and routing), and "inherit" (use in sub-blocks only when the preservation level of the sub-block must be the same as that of its parent block). The default setting is "routing," for which the synthesis netlist, placement, and routing will be exactly preserved from one run to the next (provided that the RTL within the compile point block is unchanged).

In Synplify Pro software, individual block results can be timing-driven. To get the best quality of results within a block, it is usually better to create timing constraints for each compile point individually.

If you change a top-level constraint in Synplify Pro software, only the impacted partitions are marked as changed (and therefore resynthesized).

When Synplify Pro's partition methodology is used, you can create multiple instances of a module.

Conclusion

The latest generation of 65-nm FPGAs has enabled complex system-on-chip designs to be implemented on multimillion-gate-capacity FPGAs. As FPGA designs grow in capacity, it becomes increasingly important to deliver flows that are fast and flows that converge. Users are looking for better "divide-and-conquer" approaches to quickly tune their designs and assimilate small design changes.

Synplicity and Xilinx formed a joint Ultra-High Capacity Task Force to tackle this very challenge. SmartGuide and Partition are the first of several new solutions that Synplicity and Xilinx will be delivering to provide users with more efficient design methodologies.

Logic analyzers up to 50% off

A special limited-time offer from Agilent.

Agilent portable and modular logic analyzers

- *Increased visibility with FPGA dynamic probe*
- *Customized protocol analysis with Agilent's exclusive packet viewer software*
- *Low-cost embedded PCI Express packet analysis*
- *Pricing starts at $9,450*

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

The FPGA dynamic probe, when combined with an Agilent Windows®-based logic analyzer, allows you to access different groups of signals inside your FPGA for debug—without requiring design changes. You'll increase visibility into internal FPGA activity by gaining access up to 128 internal signals with each debug pin.

Our 16800 Series logic analyzers offer you unprecedented price-performance in a portable family, with up to 204 channels, 32 M memory depth and a pattern generator available.

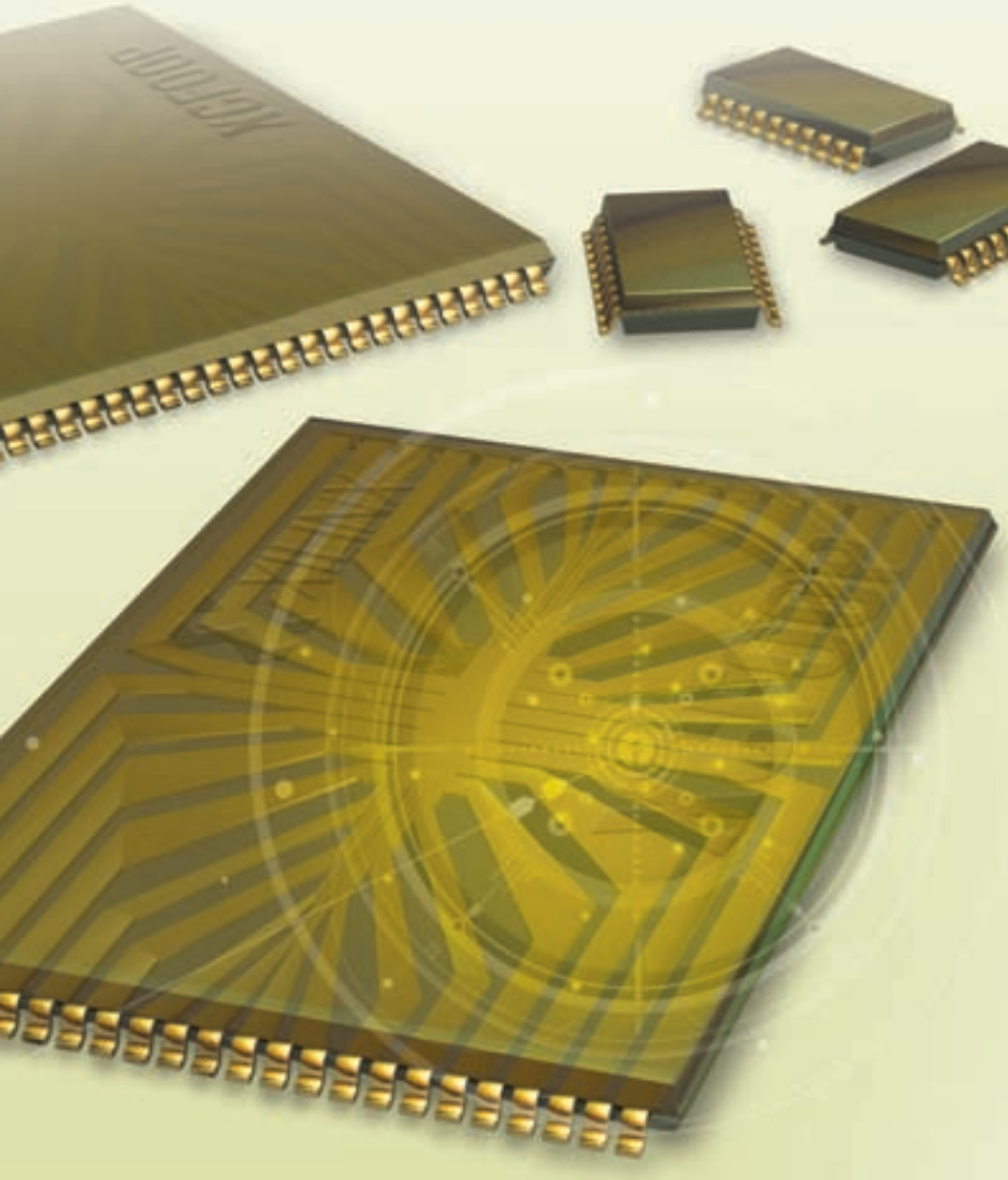
And now for a limited time, you can receive up to 50% off our newest 16901A modular logic analyzer mainframe when you purchase eligible measurement modules. Offer valid February 1, 2007 through August 15, 2007. Reference promotion 5.564.

www.agilent.com/find/logic-offer

Agilent Technologies

Picking the Right PROM for Your System

Platform Flash PROMs deliver at every stage in the product lifecycle.



by Jameel Hussein
Applications Engineer
Xilinx, Inc.
jameel.hussein@xilinx.com

Frank Toth
Senior Marketing Manager, EasyPath
Xilinx, Inc.
frank.toth@xilinx.com

The flexible and fully integrated FPGA configurations from Xilinx take advantage of FPGA capabilities like remote updating and reconfiguration. Available configuration silicon options (see Figure 1) include Platform Flash, byte peripheral interface (BPI) supporting parallel NOR flash, and serial peripheral interface (SPI flash).

Each of these configuration options have different speed and complexity tradeoffs and can offer you the flexibility of sharing configuration memory with other system-level tasks, thus greatly reducing total cost. For example, some applications use a microprocessor that interfaces with BPI flash on board. In these cases, you can configure the FPGA from the same BPI flash that the microprocessor uses to save cost and board space.

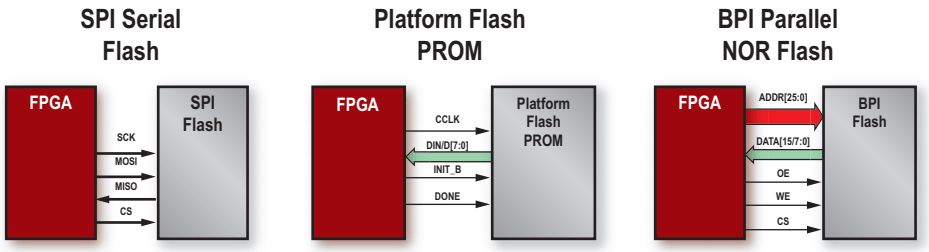


Figure 1 – Available configuration silicon options in Spartan-3E/3A/3AN and Virtex-5 devices

Other applications are more cost-sensitive; you may have to make certain sacrifices to keep the project within your budget. If you can afford a slower configuration time, SPI PROMs can provide the lowest cost solution in many cases.

Because Xilinx gives you the flexibility to use multiple memory sources for FPGA configuration, you will need to consider several factors at the start of the design: the configuration speed required to meet system specifications (for example, PCI Express); ease of use and design (such as a simple power-up configuration); and memory, with value-added features like multiple design revisioning (for product customization) and safe-update capabilities (for flawless remote configuration). Other features such as compression can also save costs.

Before the introduction of the Xilinx® Spartan™-3E, Spartan-3A, and Virtex™-5 families, designers who configured Xilinx FPGAs with commodity flash would use a three-chip configuration solution: a commodity flash PROM, a CPLD, and of course the FPGA. The newer FPGAs eliminate the need for a CPLD (or other controller) by providing a direct interface for leading commodity flash devices, thus reducing the chip count to one memory device.

Safe-Update Solutions with Platform Flash PROMs

In-system safe-update solutions, also referred to as remote-update solutions, are becoming more and more popular; in many cases they are essential to a product's success. There are several variations of a safe-update solution, but the basic idea is to update the FPGA bitstream in-system, which involves updating the image inside the PROM and reconfiguring the FPGA from that updated image. The "safe" part of a safe-update solution comes from the ability to recover from a failed attempt at updating the FPGA, which is only possible if the FPGA always has a known-good or golden bitstream stored in the PROM, from which it can always configure in case of an error. With competing FPGAs, these solutions required a third device, typically a CPLD, to control the update process and help the FPGA recover in the event of an error.

For example, where there is a need to reconfigure multiple devices (see Figure 2) from a base station, it is important that the update be flawless. If there is any corruption of the new configuration data, there is a fail-safe fallback position.

Whereas competing solutions required a third device, Virtex-5 FPGAs have "fallback" logic already built into the device; only the FPGA and PROM are necessary. This fallback logic is among the key parts to a safe-update solution, which comprises:

- A file-generation flow for updated image: iMPACT + XAPP972
- An embedded in-system programmer: XAPP058 or XAPP424
- Non-volatile configuration memory: a Platform Flash XCF00P PROM
- An FPGA with a fail-safe configuration feature: Virtex-5 fallback feature

The first step is creating a file that contains the information to update the PROM with the new bitstream for the FPGA; you can do this with iMPACT software tools. The file created will contain the new bitstream as well as the instructions to interface with the PROM. For more details, see Xilinx application note XAPP972, "Using Serial Vector Format (SVF) Files to Update a Platform Flash PROM Design Revision In-System," at *www.xilinx.com/bvdocs/appnotes/xapp972.pdf.*

The next step is to run or play this file (Figure 3); you can do this with an embedded in-system programmer available in Xilinx application note XAPP424,

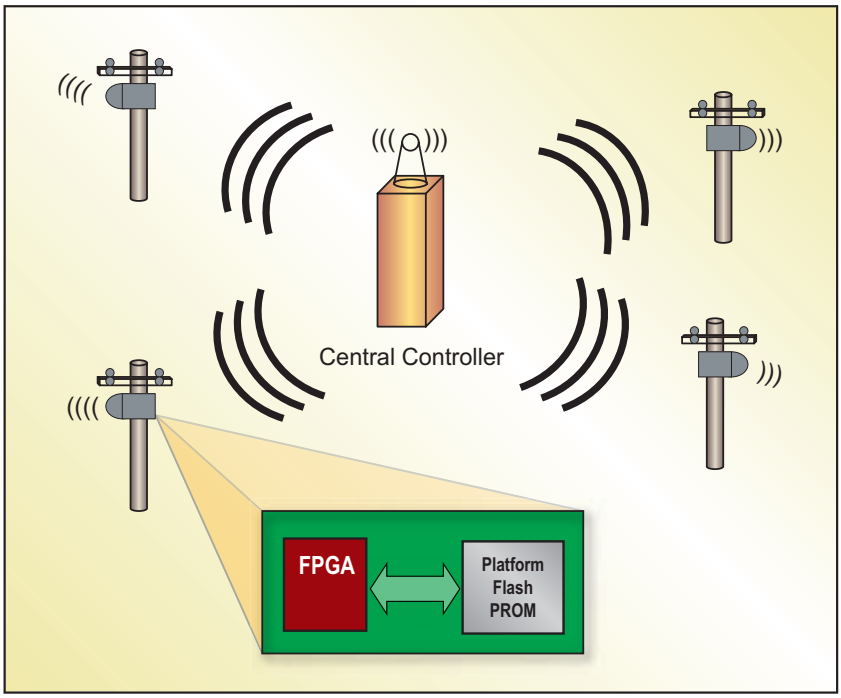


Figure 2 – Reconfiguring multiple systems from a remote base station

"Embedded JTAG ACE Player," (*www.xilinx.com/bvdocs/appnotes/xapp424.pdf*) or XAPP058, "Xilinx In-System Programming Using an Embedded Microcontroller," (*www.xilinx.com/bvdocs/appnotes/xapp058.pdf*). XAPP424 is the latest in-system programmer available from Xilinx and is capable of significant performance gains over previous solutions in many cases. The programmer uses JTAG to interface with the Platform Flash PROM, which is the next key step in the process.

The Platform Flash XCF00P PROM (Figure 3) is capable of storing as many as four different designs or revisions, but many safe-update solutions only use two: the known-good or golden bitstream and the updated or enhanced bitstream. The key to maintaining safety is to only erase and reprogram the enhanced bitstream and leave the golden bitstream completely unmodified from original programming. XAPP972 describes a method to ensure safety, which is made possible with Platform Flash PROMs.

The last step is the fallback logic inside Virtex-5 FPGAs. Once the Platform Flash PROM has been reprogrammed with the updated bitstream, the FPGA must be reconfigured. The danger is that if there was an error reprogramming the PROM, the FPGA will not be able to configure from the updated bitstream. Fallback logic helps to eliminate this danger. If the FPGA is triggered to reconfigure from the updated bitstream and configuration fails, the FPGA will automatically attempt to reconfigure from the golden bitstream. The golden bitstream is the known-good bitstream that was unmodified during the update process; the FPGA should regain functionality with the golden design. The functionality of the golden bitstream is at your discretion.

This safe-update solution is a two-chip solution made possible with Platform Flash PROMs and Virtex-5 FPGAs. The resources to build such a solution are available on *www.xilinx.com* and can significantly reduce time to market, as well as add value and flexibility to the project.

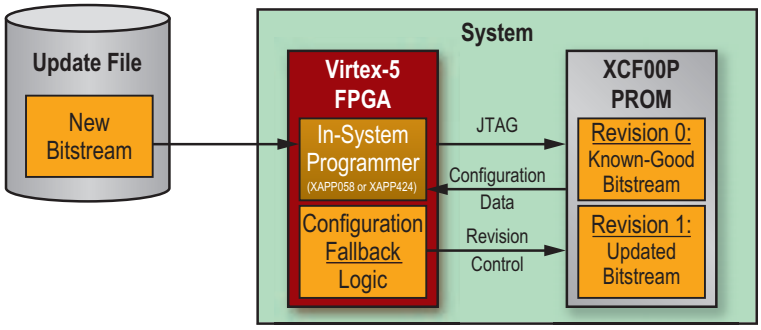


Figure 3 – Safe-update system overview with Virtex-5 FPGAs and Platform Flash XCF00P PROM

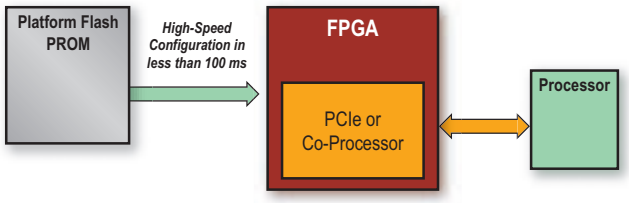


Figure 4 – High-speed configuration with Platform Flash PROMs in less than 100 ms for all Spartan-3E/3A/3AN and Virtex-5 devices up to LX85 and LX85T

Configuration Speed

Another advantage of using Platform Flash PROMs is the configuration speed. For most applications, configuration speed is not an issue; however, for several projects, the configuration speed is the difference between success and failure.

For example, applications using PCI Express require the FPGA to configure in less than 100 ms (Figure 4). For some SPI and BPI NOR flash devices, it can be difficult to configure larger FPGAs in that amount of time. The FPGA provides the configuration clock (CCLK) when configuring from a third-party flash device. The CCLK is generated by internal circuitry and on some FPGAs can have 50% variation, which means that if the CCLK is set to 60 MHz, the output clock could range from 30 to 90 MHz. You must set the CCLK to ensure that the maximum clock rate on the PROM is not exceeded.

Finding the correct CCLK speed is easy with a few simple formulas. The FPGA CCLK + 50% should be less than the maximum clock rating on the PROM. The same FPGA CCLK – 50% is the worst-case FPGA CCLK. The worst-case FPGA CCLK is critical when determining the maximum configuration time, which is roughly calculated by dividing the number of configuration bits for a device by the number of bits per second being delivered to the FPGA.

Platform Flash has the ability to run from an external stable clock source. When the FPGA is in slave mode, it too can run from a high-speed stable clock source to deliver the fastest and most consistent configuration times. In most cases configuration times are less than 100 ms, even for mid-sized Virtex-5 devices. (Numbers are approximate; actual numbers may vary across different devices. Refer to the appropriate data sheets for more information at *www.xilinx.com/xlnx/xweb/xil_publications_index.jsp*.)

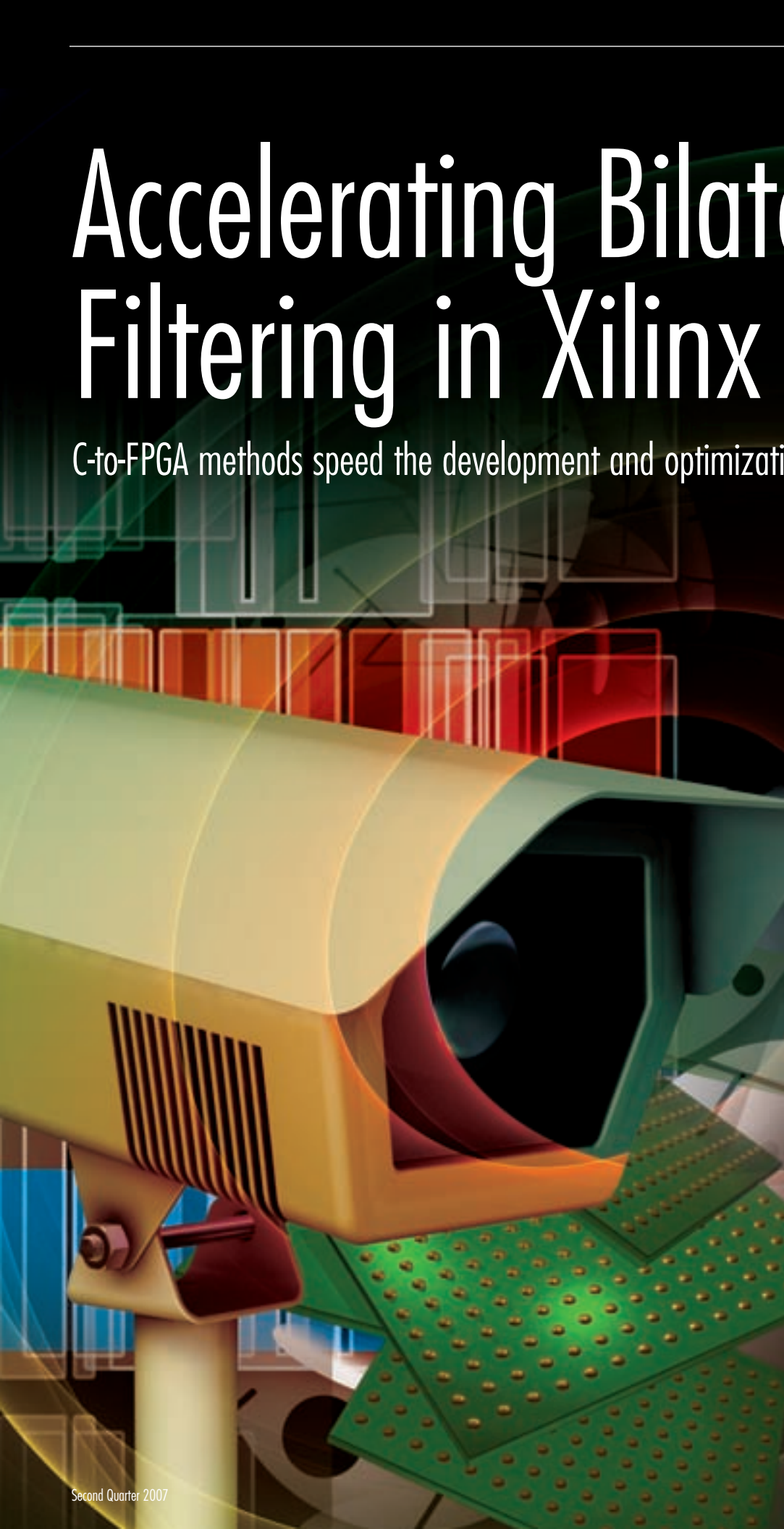
Conclusion

Designers must make trade-offs in speed, complexity, functionality, and cost during configuration. Existing on-board BPI and SPI devices can be put to dual-mode use for both configuration and system-level memory. Platform Flash PROMs provide unique money-saving features, including the ability to store and select multiple bitstreams. Virtex-5 and Spartan-3E/3A devices provide you with integral support for SPI and BPI flash, in addition to Platform Flash.

For more information, please visit *www.xilinx.com/products/silicon_solutions/proms/pfp/index.htm.*

Accelerating Bilateral Filtering in Xilinx FPGAs

C-to-FPGA methods speed the development and optimization of complex filters.



by Ed Trexel
Senior Applications Engineer
Impulse Accelerated Technologies, Inc.
edward.trexel@impulsec.com

Dov Stamler
Hardware Engineer
Vigilant Technology Ltd.
dovs@vigilanttechnology.com

Bilateral filtering is used in many video applications to smooth images while preserving edge detail. One such application uses a bilateral filter to reduce random noise during video pre-processing and to improve video compression efficiency in a real-time video environment.

Vigilant Technology provides intelligent IP surveillance and security solutions. Vigilant currently supports tens of thousands of cameras in airports, government offices, financial institutions, correctional facilities, casinos, and town centers.

In this article, we'll describe how one portion of an advanced closed-circuit television (CCTV) system – the bilateral filter – has progressed from an initial software model to being implemented in actual hardware using C-to-FPGA tools. Starting with a verified C-language model, we developed a hardware implementation through the use of C-to-hardware tools and iterative design methods. To allow for quick prototyping and algorithm experimentation, we used common and familiar C-language tools for debugging and software/hardware design analysis.

Iterative hardware-level testing was an important part of this conversion. We accomplished this testing with an FPGA-embedded software test bench. For this purpose, the embedded processor features of the Xilinx® Virtex™-4 FX device were employed for direct, hardware-in-the-loop embedded testing of the image filtering algorithm.

Bilateral Filtering

The bilateral filter is representative of many other critical image algorithms. It works like this: for each pixel, a new value is calculated using those pixels immediately around the pixel of interest, using the equation:

$$\text{New pixel} = \frac{\sum_n f_n \times coeff_n}{\sum_n coeff_n}$$

where each coefficient is calculated by:

New coefficient

$coeff_n = ORIGCOEFF \times KERNEL$

The pixel being recalculated and those around it form a 3 x 3 matrix referred to as the "input vector." The input vector is presented to the filter with all of the pixels in parallel forming a 9-byte-wide data path. From each input vector, the filter calculates the new value for the center pixel, referred to as the "output vector," which is 1 byte wide. Calculation of the output pixel is defined by a proprietary kernel algorithm developed at Vigilant Technology, and is the heart of the bilateral filter.

Figure 1 summarizes the input and output vectors.

Starting from a C-Language Model

Vigilant Technology wrote the original implementation of the bilateral noise filter in C, using floating-point math. As a first-pass optimization, we converted all mathematical operations to unsigned fixed point. We accomplished this conversion in part by using C-language macros included with the Impulse C tools.

As a second level of optimization, we reorganized the C code to minimize the number of loops and simplify data handling, using buffered streams for all filter I/O. We defined this I/O using streaming-related interface functions provided within Impulse C.

In certain cases, we replaced array references in the C code with individual variables in order to avoid data-timing dependencies and produce parallel calculations. C-code dependencies included summations that took a linear approach of summing as coefficients were calculated, as well as a multi-tiered approach. During this manual optimization, we used macros to keep the C code relatively clean and portable.

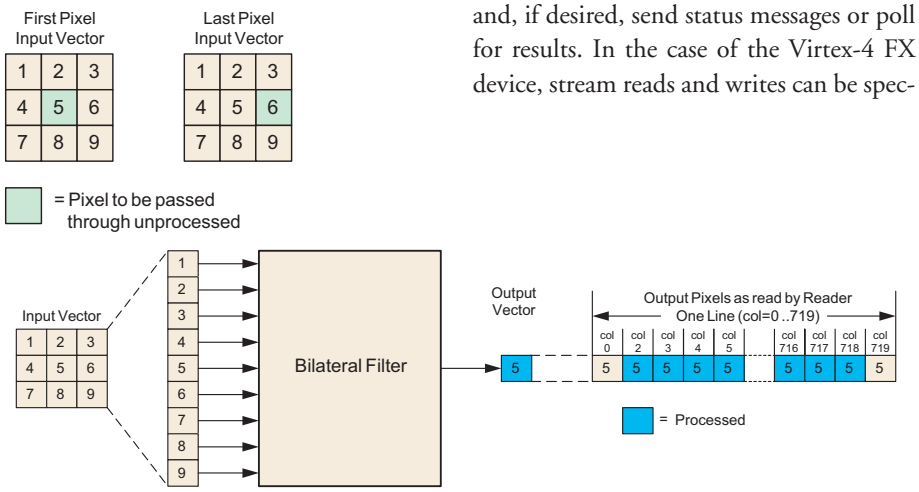


Figure 1 – Bilateral filter processing flow

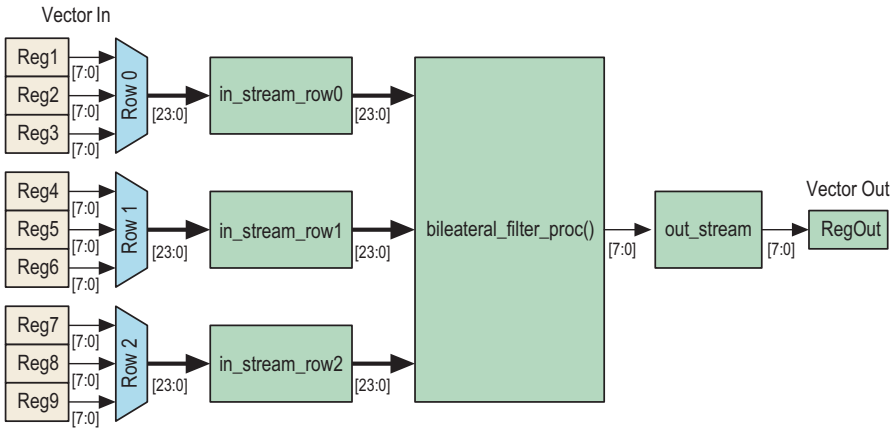


Figure 2 – Block diagram showing streams and processes

As the code was optimized and modified for hardware generation, we used a standard C debugger (part of the Borland C++ Builder tools) to step through both the original and modified C code, allowing for validation of the math and the overall functionality of the application. Throughout development, we performed real-world validation using actual video clips representing various CCTV scenarios.

Figure 2 illustrates how we implemented and tested the bilateral noise reduction filter using the Impulse C streaming interfaces and a Xilinx ML410 development board.

On the software side of the application (in this case on the PowerPC™ 405 processor used for hardware-level testing), Impulse C functions open and close data streams, read or write data on the streams, and, if desired, send status messages or poll for results. In the case of the Virtex-4 FX device, stream reads and writes can be specified as operations that take advantage of the auxiliary peripheral unit (APU) interface, which provides a high-performance software-to-hardware serial interface capable of transferring as many as 128 bits of data in a single transaction.

The embedded PowerPC allowed us to exercise the filter at-speed, using the processor as a test generator.

Generating and Testing the FPGA Hardware

When generating hardware, the Impulse C compiler extracts low-level parallelism from otherwise sequential C-language statements. Structures such as loops are evaluated at a high level and either unrolled or pipelined to increase throughput. These compiler optimizations save application developers a great deal of time.

The Impulse C compiler generates hardware in either VHDL or Verilog. This hardware is synthesized using Xilinx ISE™ software. On the processor side, the compiler generates run-time libraries ready for use on the embedded PowerPC or MicroBlaze™ processor.

Iterative, hardware-level unit testing was an important part of this project. The embedded PowerPC allowed us to exercise the filter at speed, using the processor as a test generator. Using this approach allowed us to validate the correctness of the data, as well as to directly measure the filter performance and throughput.

In fact, an entirely C-language testing method using the embedded PowerPC proved to be far more practical than HDL simulation given the much faster, hardware-speed operation of tests involving video data. Xilinx Platform Studio (XPS) tools proved invaluable during this process. XPS includes a board support package for the Xilinx ML410 board, allowing the embedded PowerPC to be set up as a test generator and communicate with the hardware filter through the Virtex-4 FX APU interface. The Impulse tools include XPS hardware and software export features that greatly simplified the creation of a custom, APU-connected peripheral from the generated bilateral filter hardware.

ISE tools provided the primary measurements of size and clock speed performance, while the ML410 board provided the final determination of whether the generated filter worked properly in hardware. HDL simulation was used at various points in the development process to validate the functionality of the generated HDL code, but more importantly to observe cycle-level operation and find opportunities for C code optimization.

HDL simulation was of limited value for heavily pipelined sections of the generated hardware, but proved quite useful when analyzing sequential blocks and stages of generated HDL code. This HDL-level analysis could be related directly back to the C source code through the use of the Impulse interactive optimizer, which provides visualizations of C-code parallelism and helps to match the generated hardware code to corresponding sections of the original C-language source.

Iterative Optimization

After we developed an initial, non-optimal hardware prototype, we began a process of iterative optimization to meet system timing and size constraints. Based on previous experience with hand-optimized HDL implementations of similar filters, available resources, and real-time video requirements, Vigilant specified a budget of 1,000 FPGA slices, along with a target video data rate of 60 frames per second.

An initial restructuring of the algorithm placed the critical loops in two parallel individual processes. This method of optimization at first appeared promising, with a relatively high clock rate of 143 MHz, but was limited by a noncontiguous pipeline. This characteristic caused the system performance to be limited by the latency of the slowest pipelined process. Even with the two processes running in parallel in a system-level pipeline, the best rate was substantially slower than the required pixel rate.

After additional analysis, we made various modifications to the C code, including the use of a single pipelined process for the main filtering loop. The result of this series of optimizations was a filter that met the slice-count budget and achieved a single-frame rate of just over 15 ms (66 frames per second) using an FPGA clock rate of just 27 MHz, which is the input clock rate of a D1 video stream received in BT.601 format. These results are summarized in Table 1.

Filter Version	Occupied Slices	Frame Rate
Parallel Multi-Process	3,592	12.11 Hz
Pipelined Multi-Process	1,570	10.10 Hz
Pipelined Single Process	930	65.31 Hz

Table 1 – Iterative C-language optimizations resulted in incremental improvements

The widely differing performance results seen in this example highlight the importance of using appropriate C programming methods for FPGAs. Although C-to-FPGA tools can dramatically reduce the amount of development time required to convert complex algorithms to hardware, these tools must be applied carefully. You may need to use iterative optimization methods to achieve results comparable to hand-crafted HDL. Fortunately, these techniques are well within the experience and skills of embedded software developers.

Conclusion

Software-to-hardware design tools can play an important role in the fast prototyping and development of complex FPGA algorithms. However, some level of hand-optimization will be required to obtain performance comparable to hand-crafted HDL. Iterative design methods combined with interactive design and optimization tools are an efficient way to manage the complexity of large embedded and high-performance computing applications.

For additional information about C-to-FPGA programming and optimization, visit *www.ImpulseC.com/xilinx*.

Ultra-High-Speed Spectral Analysis in Xilinx FPGAs

A 32k-point FFT with 2-GSPS data throughput in a single FPGA establishes a new level of performance.

by Bruno Stuber
Professor
University of Applied Sciences Northwestern Switzerland,
School of Engineering, Institute of Automation
bruno.stuber@fhnw.ch

Dino Zardet
Development Engineer
University of Applied Sciences Northwestern Switzerland,
School of Engineering, Institute of Microelectronics
dino.zardet@fhnw.ch

A fast Fourier transform (FFT) is the ultimate way to accomplish spectral analysis in digital signal processing. Many real-time applications require seamless operation with continuous data throughput, a high-speed data rate, excellent spectral resolution, and a large dynamic range. Today, many standard solutions are offered to perform real-time FFTs, either with DSPs or FPGAs. However, to go beyond established limits, a thorough analysis of the system architecture is necessary, as is a design approach that focuses on timing.

FPGAs offer the potential for a huge amount of parallelism. For an FFT application, the number of multipliers and RAM blocks determines the overall performance, expressed in terms of data throughput and the number of spectral lines or bins.

In this article, we will discuss the implementation of a 32k-point FFT (32,768 points) with power spectrum accumulation. The 8-bit input data flows continuously at a rate of 2 gigasamples per second (GSPS), therefore providing an analog bandwidth of 1 GHz. The spectrometer is implemented in a single Xilinx® Virtex™-II Pro XC2VP70 FPGA operating at an overall clock speed of 125 MHz.

We developed the unit for spectroscopy applications in millimeter and terahertz radio astronomy. However, many other applications are possible, including low-frequency radio astronomy, atmospheric physics, physical chemistry, radio surveillance, and spectral measurements.

High-Speed FFT Architecture in an FPGA

To achieve the enormous throughput of 2 GSPS, a strictly synchronous and pipelined architecture is necessary along the signal processing path. A "pipeline FFT" is best suited for a high-speed parallel implementation.

In typical FFT signal flow graphs, a number of r input data, where r is a power of two, are processed within the first butterfly stage. The butterfly outputs r intermediate results; r is the radix of the FFT implementation. The butterfly involves some additions and (r-1) complex multiplications with twiddle factors. This process is visualized in Figure 1.

M stages are needed for an N-point FFT, M = $\log_r(N)$. All stages (with the exception of the last one) perform twiddle multiplication and data shuffling, which are necessary to implement the well-known "butterfly" signal flow graph of the FFT algorithm.

Therefore, a radix-r FFT pipeline forwards r data streams with the pipeline's clock frequency, f_c. The data throughput is thus r x f_c.

The hardware resources involved in the FFT processing are adders, multipliers, memory for the twiddle factors and data shuffling, and a number of multiplexers. We recommend using dedicated multipliers, preferably in conjunction with the assigned internal registers. Twiddles are stored in Xilinx block RAM instantiated as ROM. You can also use block RAM for most of the data shuffling stages, employing the dual-port capability, whereas some smaller memory blocks are more efficiently realized with distributed RAM.

A closer analysis of the pipeline FFT shows that there are (N-r) complex twiddle factors needed and a total number of N complex data memory for the data shuffling. The number of multipliers is 4 x (M-1) x (r-1) for a radix-2 or radix-4 FFT pipeline.

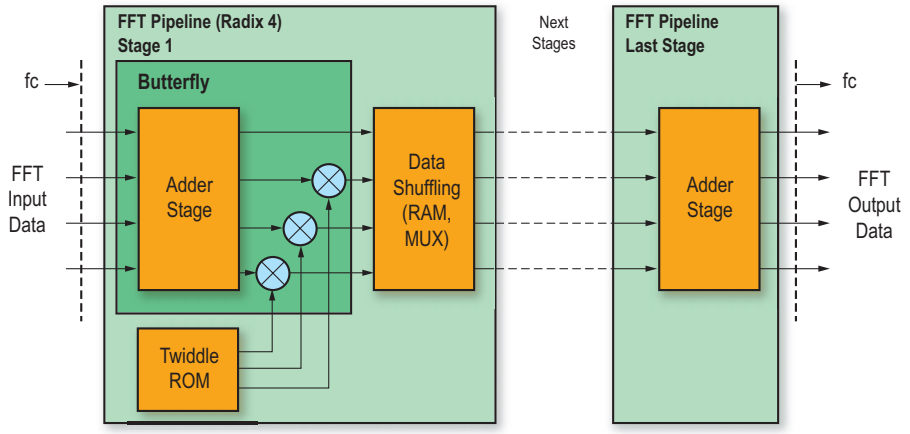


Figure 1 – Pipeline FFT structure

A crucial factor with respect to the throughput is f_c. The dedicated signal processing elements, multipliers, and RAM blocks in Virtex-II FPGAs have a propagation delay of as much as 5 ns. Together with the signal path delays – assuming that such a complex circuit is spread over a large area – an overall delay of approximately 8 ns is feasible, resulting in an overall system clock of 125 MHz.

Several pipelines may operate in parallel with the aid of an extended input, buffering and scheduling to further enhance overall data throughput.

Applying the FFT in Radio Astronomy

Spectrometers in millimeter astronomy are used to identify and explore atomic and molecular lines of cosmic sources. Because the signals are very weak, the signal-to-noise ratio may be less than -60 dB; the power spectra will be accumulated over a period of milliseconds up to hours and weeks. To extract the chosen signal, the accumulation process is carried out "on source," with the antenna directed to the object, and "off source," with the antenna deflected. The resulting spectra are then subtracted.

The frequency range observed is spread from tens of kilohertz to approximately 2 terahertz. The bandwidths of the spectrometers should therefore be as large as possible while the frequency resolution is reasonably in the range of 100 kHz. Currently, broadband spectrometers are mainly based on acousto-optical devices with bandwidths as high as 3 GHz. They have, however, a resolution of about 1 MHz and pose some problems in dynamic range, stability, and cost.

New generations of FPGAs and high-speed digital-to-analog converters enable FFT-based solutions that are competitive with or even superior to acousto-optical devices, with the advantages of an all-digital, reconfigurable, and programmable system.

Radio spectroscopy based on FFT typically comprises three processes:

- Digitizing the receiver's intermediate signal
- FFT computation
- Accumulation (averaging) of the power spectra; that is, the squared magnitude of complex FFT values

We attempted to establish a new level of performance with an FFT-based spectrometer – one with an analog bandwidth of 1 GHz subdivided in 16,384 channels.

For more information, measurement results, and the features of the spectrometer system, visit *www.astro.phys.ethz.ch/instrument/argos/argos_nf.html*.

Designing the Spectrometer Unit

A detailed block diagram of our spectrometer unit is shown in Figure 2. The analog input signal is fed to a pre-amplifier and analog-to-digital (A/D) converter, which digitizes the signal at a rate of 2 GSPS. The data are transferred to the FPGA through a 32 x 8-bit-wide bus. Next, the FFT frames (32k data samples) are multiplied with a programmable window function and buffered. The complex spectral values are then computed with the aid of two parallel, radix-4 32k-point FFT pipelines. The data are transported continuously in 2 x 4 streams of complex numbers with the system clock f_c = 125 MHz. A 32k FFT frame is processed every 16.384 µs.

At the input, the streams have a width of 2 x 9 bits. They are expanded to 2 x 18 bits later within the pipelines, thus maintaining an adequate level of precision and matching the FPGA's RAM and multiplier structures.

After computing the power spectrum, the resulting values are accumulated in 36-bit-wide registers. The number of accumulation cycles is programmable. Hence, a single shot is possible as well.

When an accumulation cycle is completed, the data are flushed to an output buffer. The data is transferred through interrupt or polling over the PCI interface to the host computer.

A number of control and status registers are accessible through PCI, allowing for flexible programming of the whole unit.

FPGA Utilization

The XC2VP70 comprises 328 block RAM and as many multipliers. In the first design step, we analyzed the need for processing resources. With the guidelines presented here, you can easily determine the number of multipliers.

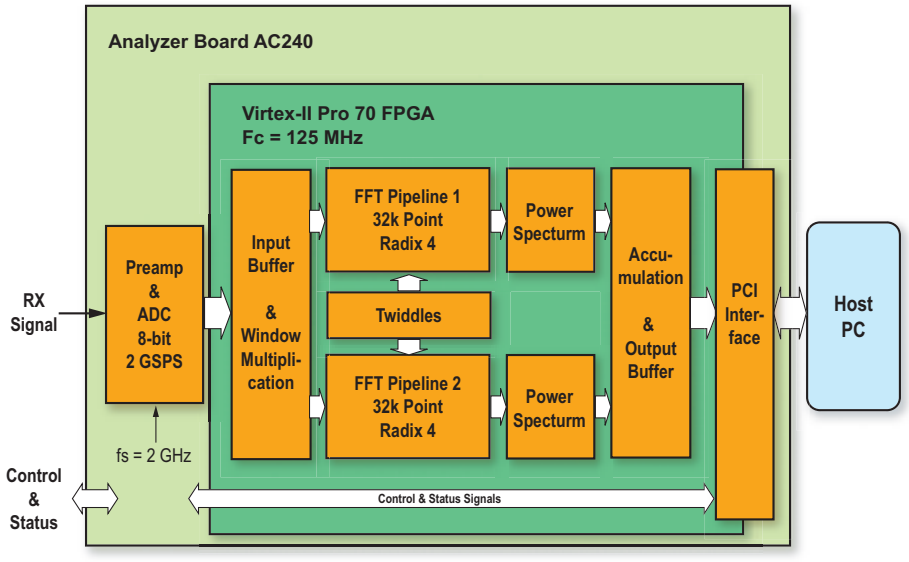


Figure 2 – Spectrometer unit block diagram

A rough estimate of memory is also possible, taking into account all buffer stages at the pipeline's input and output, the data-shuffling stages within the pipelines, and the twiddle and window ROM. However, determining the exact number of RAM blocks requires a detailed stage-by-stage analysis, taking into account the word widths involved. A crucial factor is the granularity of RAM blocks with respect to memory size, number of ports, and configuration possibilities.

Detailed analysis shows that for our spectrometer application, RAM blocks are the most limiting resource. Table 1 summarizes device utilization. All functional units, as shown in Figure 2, are included.

Number of Occupied Slices	32,379 out of 33,088	98%
Number of Slice Flip-Flops	41,549 out of 66,176	63%
Number of Block RAMs	309 out of 328	94%
Number of MULT18X18s	196 out of 328	59%

Table 1 – FPGA device utilization

Managing Timing

Such a complex and time-critical application requires a detailed inspection of every signal path with respect to propagation delay. Throughout the design process, we allocated multipliers and RAM blocks directly in VHDL. All arithmetic elements, RAM blocks, and logic stages are registered consecutively.

We developed the FFT unit in VHDL using the Mentor Graphics tool chain and Xilinx ISE™ software for place and route. Taking precautions very early in the design stage, we did not apply any location constraints for synthesis and place and route.

Using an XC2VP70 Virtex-II Pro device with speed grade -6, we obtained a minimum clock period of 7.495 ns. Thus, a clock speed of even 133 MHz is feasible.

Flexible Hardware Platform

Figure 3 shows the hardware platform, an Acqiris AC240 2-GSPS analyzer CompactPCI card. Two interleaved 8-bit A/D converters, with a pre-amplifier stage, provide a configurable sampling rate as high as 2 GSPS. The signal processing core comprises a Virtex-II Pro FPGA. Communication to a host computer is established through the CompactPCI bus.

A firmware development kit (FDK) is included with the AC240. With this concept, you have all of the interfaces avail-

able in a uniform description at the VHDL level. Data input streams, I/O streams through PCI, and front-panel control and status signals are accessible in the VHDL environment. A number of registers are accessible through PCI with driver software. A number of trigger and interrupt mechanisms are also available.

To implement our application, we had to connect the FFT unit to the FDK's interfaces. We found embedding applications in the FDK environment both practical and convenient.

For more information about the hardware, visit *www.acqiris.com*.

Conclusion

With our implementation, we established a new level of performance in the field of FFT-based spectrometers. This is possible through an architecture that is highly adapted to the high-speed requirements and signal processing resources available in FPGAs. Block RAM and multipliers are the key elements.

The spectrometer unit has been tested with complete success in the laboratory and in radio astronomy applications. Taking into account the latest developments in FPGA technology in conjunction with faster A/D converters (10 bits at 4 GHz), we may be able to further increase performance.

Figure 3 – AC240 analyzer board with FPGA

Spartan-DSP Takes Aim at Affordable DSP Performance

The latest addition to the XtremeDSP platform portfolio takes price/performance to a new level.

by Greg Brown
Sr. Manager, IC Marketing, Processing Solutions Group
Xilinx, Inc.
greg.brown@xilinx.com

The search for the best DSP solution for a given application usually leads designers into a maze of price, performance, and power consumption trade-offs, often requiring a compromise of one or more to accommodate the others. With the introduction of Spartan™-3A DSP, the latest addition to the Xilinx® XtremeDSP™ portfolio and the first Spartan FPGA to be DSP optimized, Xilinx has broken through the maze to deliver the most efficient combination of these three critical values for a host of applications, including wireless base stations, mobile defense communications systems, surveillance, automotive, video, and medical imaging technologies.

Spartan-3A DSP delivers 32 or more GMAC/s (32 billion multiply accumulate operations per second), up to 2,200 Mbps memory bandwidth, and size-reduced packaging. This represents a compelling price/performance breakthrough that hits the mark for applications such as digital front-end (DFE) and baseband solutions in a single-channel picocell wireless base station; military mobile software-defined radios (SDRs); ultrasound systems; driver assistance/media systems; high-definition video; and Smart IP cameras.

Moreover, with as much as 53,712 logic cells, 2,268 Kb of block RAM, 373 Kb of distributed RAM, 519 I/O pins, DeviceDNA for security, and newly developed hibernate/suspend power-management features, Spartan-3A DSP offers enough integration capacity to drive price/performance/power ratios even lower. Add to this the inherent benefits afforded by FPGA-based DSP solutions – lower risk through design flexibility and faster time to market – and the value of the Spartan-DSP platform becomes increasingly apparent (see Table 1).

	Spartan-DSP		Virtex-DSP					
	Spartan-3A DSP		Virtex-4 SX FPGA			Virtex-5 SXT FPGA		
	XC3SD1800A	XC3SD3400A	XC4VSX25	XC4VSX35	XC4VSX55	XC5VSX35T	XC5VSX50T	XC5VSX95T
Max DSP Performance (GMAC/s)	21	32	64	96	256	106	158	352
Max Memory Bandwidth (Mbps)	1,736	2,196	4,608	6,912	11,520	3,326	5,227	9,662
Max DSP Frequency (MHz)	250	250	500	500	500	550	550	550
XtremeDSP DSP48 Slices	84	126	128	192	512	192	288	640
Min Footprint (mm)	19 x 19	19 x 19	27 x 27	27 x 27	27 x 27	27 x 27	27 x 27	27 x 27
Distributed RAM (Kb)	260	373	160	240	384	520	780	1,520
Block RAM (Kb)	1,512	2,268	2,304	3,456	5,760	3,024	4,752	8,784
Logic Cells	37,440	53, 712	23,040	34,560	55,296	34,816	52,224	94,208

Table 1 – The Spartan-DSP platform fills an important performance slot – the <40 GMAC/s range – in the XtremeDSP platform portfolio.

DSP-Optimized Spartan FPGAs

At the heart of the Spartan-3A DSP is a modified version of the XtremeDSP DSP48 slice – the DSP48A. Initially introduced with the release of Virtex™-4 FPGAs, the DSP48 slice has an Application Specific Modular Block (ASMBL™) architecture that powers the DSP functions in Virtex DSP devices. These XtremeDSP slices enable designers to implement solutions to complex challenges, such as hundreds of IF-to-baseband down-conversion channels, 128x chip-rate processing for 3G spread spectrum systems, and high-definition H.264 and MPEG-4 encode/decode algorithms in a cost- and power-efficient manner.

The DSP48 slices support many independent functions, including multiplier, multiplier accumulator (MAC), multiplier followed by an adder, three-input adder, barrel shifter, wide bus multiplexers, magnitude comparator, or wide counter. The architecture also supports connecting multiple DSP48 slices to form wide math functions, DSP filters, and complex arithmetic functions without the use of general FPGA fabric, thereby reducing power consumption

while delivering very high performance and efficient silicon utilization.

The Spartan-DSP DSP48A slice is a simplified and cost-reduced version of the Virtex-4 DSP48 slice. To reduce cost, the rounding modes, 17-bit shifters, and 3-input adder modes were removed from the DSP48A slice; you can implement these functions in the FPGA fabric if necessary. Two additional enhancements in the DSP48A slice are a fly-independent C-port and a pre-adder. The independent C-port provides increased flexibility in implementing DSP algorithms. The pre-adder increases density for common DSP filters and FFTs. Specifically, the pre-adder can be used to reduce the number of DSP48A slices that are required by 50% for symmetric FIR filters and by 25% for FFT algorithms, respectively. In the Spartan-3A DSP platform, the optimized DSP48A slice achieves 250 MHz operation in the slowest speed grade.

Application Impact

Typical of the price/performance/power efficiency made possible by Spartan-3A DSP devices, a single XC3SD1800A device can replace two $25 DSP processors in a Smart IP camera application, absorbing the entire video pipeline section in the process. Besides the immediate $25 cost reduction, this enables you to place the remaining control functions in a smaller, less expensive DSP processor, thereby dropping the cost of materials by another $10.

For high-volume consumer applications such as the Smart IP camera or high-definition video, this represents tremendous value to the manufacturer that goes directly to the bottom line. Add to this the savings in power, footprint, and bill of materials (BOM), and Spartan-DSP can have a direct positive impact on profitability, reliability, and product migration.

Similar studies in multi-stream video servers have shown that a design employing six $25 DSP processors can be reduced to three $25 Spartan-3A DSP devices, literally cutting device costs in half. Here again, the positive impact on power, footprint, and BOM is highly appealing.

In some cases, such as SDR for mobile defense communications, the Spartan-3A DSP can serve as a reconfigurable co-processor to a discrete DSP, providing both the aforementioned price/performance/power economies as well as eliminating the need for duplicate circuitry to support multiple waveforms.

Clearly, in every application for which Spartan-DSP is an appropriate platform, the result is greater efficiency in performance, power, and price.

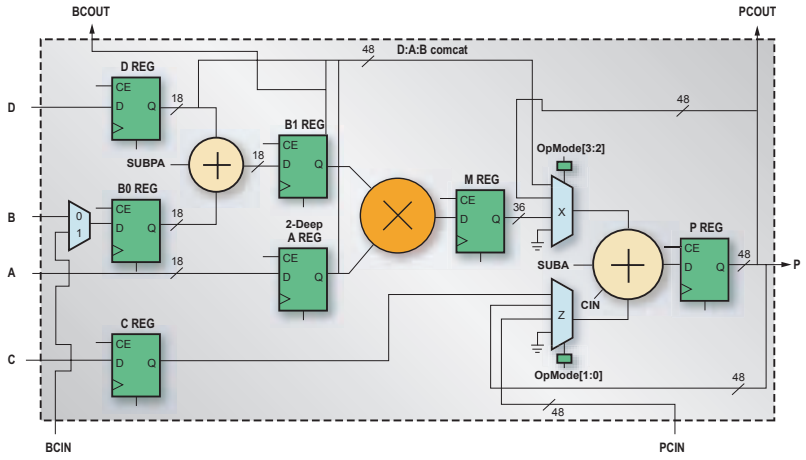


Figure 1 – The pre-adder saves nine slices for every Spartan-DSP DSP48A slice used in a FIR filter design.

XtremeDSP Solutions

The XtremeDSP Initiative, launched in November 2000, gave light to a corporate commitment shared by Xilinx and its partners to make all of the available FPGA-based DSP horsepower and flexibility accessible to three distinct designer profiles: the system designer, the DSP engineer, and the FPGA/hardware engineer. Each profile represents a unique set of responsibilities (and preferences) that dictate the requirements for their particular design environment.

System designers must quickly determine how best to partition the various functions of a system-level design across the available processing resources. Their concerns focus on the selection of processing resources that meet product performance and throughput requirements while meeting size, cost, and power budgets.

DSP engineers are more focused on the creation and refinement of DSP algorithms. Typically unfamiliar with detailed hardware design, they rely on tools to abstract away the details of hardware design so that they can focus on higher level design exploration and validation.

Hardware engineers usually work in VHDL or Verilog to extract the maximum performance from their designs. They often require the capability to work simultaneously with higher level functional blocks and their own RTL-level design from within the same design environment, running test benches to validate function and performance.

Thus, an essential factor in the success of the XtremeDSP Initiative is the degree to which the design tools accommodate all three profiles. Since the launch of the initiative, XtremeDSP tools such as SystemGenerator and AccelDSP have evolved to provide system modeling, algorithmic development and exploration, automatic generation of test benches, design verification and debugging, and HDL generation and simulation. Whether you prefer to work with VHDL, Verilog, C/C++, MATLAB, Simulink, HDL, or any combination of these, XtremeDSP tools provide fast and efficient access to the full power of the FPGA.

Conclusion

In the DSP marketplace, it is not always the fastest, the least expensive, or the most power-efficient processor that wins: it is the platform that provides the best fit in every category. For a large and growing number of applications, Spartan-3A DSP provides the most efficient combination of price, performance, and power consumption, backed by a robust set of tools, IP, and support infrastructure. If your next DSP design needs extreme efficiency, you may need the latest XtremeDSP solution from Xilinx.

Performance. Delivered.

X5 400m

PCI Express XMC Module

Download Data Sheets NOW!

Features

- Two 400 MSPS, 14-bit A/D Channels
- Two 400 MSPS, 14-bit D/A Channels
- ±2V, 50 Ohm, SMA Inputs and Outputs
- Xilinx Virtex5, LX110 FPGA (SX95 coming)
- 1GB DDR2 DRAM, 4MB QDR SRAM
- 8 Rocket IO Private Links, 2.5 Gbps each
- >1 GB/s, 8-lane PCI Express Host Interface
- Power Management Features

Perfect for

- Wireless Receiver and Transmitter
- WLAN, WCDMA, WiMAX front end
- RADAR
- Electronic Warfare
- High-Speed Data Recording and Playback
- High-Speed Servo Controls
- IP Development

FrameWorkLogic

www.innovative-dsp.com • sales@innovative-dsp.com

Innovative Integration

... real time solutions!

High-Volume Design Application Notes

These FPGA and CPLD application notes can help you design for high-volume markets.

Spartan-3 Generation FPGAs	
Audio, Video, and Image Processing	
XAPP485 – 1:7 Deserialization in Spartan-3E FPGAs at Speeds Up to 666 Mbps	*www.xilinx.com/bvdocs/appnotes/xapp485.pdf*
XAPP486 – 7:1 Serialization in Spartan-3E FPGAs at Speeds Up to 666 Mbps	*www.xilinx.com/bvdocs/appnotes/xapp486.pdf*
XAPP491 – Inverting LVDS Signals for Efficient PCB Layout in Spartan-3 Generation FPGAs	*www.xilinx.com/bvdocs/appnotes/xapp491.pdf*
XAPP930 – Color-Space Converter: RGB to YCrCb	*www.xilinx.com/bvdocs/appnotes/xapp930.pdf*
Digital Signal Processing	
XAPP634 – Analog Devices TigerSHARC Link	*www.xilinx.com/bvdocs/appnotes/xapp634.pdf*
XAPP753 – Interfacing Xilinx FPGAs to TI DSP Platforms Using the EMIF	*www.xilinx.com/bvdocs/appnotes/xapp753.pdf*
Communication and Networking	
XAPP551 – Viterbi Decoder Block Decoding – Trellis Termination and Tail Biting	*www.xilinx.com/bvdocs/appnotes/xapp551.pdf*
XAPP569 – Digital Up and Down Converters for the CDMA2000 and UMTS Base Stations	*www.xilinx.com/bvdocs/appnotes/xapp569.pdf*
XAPP942 – Reference System: OPB Ethernet MAC	*www.xilinx.com/bvdocs/appnotes/xapp942.pdf*
Memory Interface and Storage Elements	
XAPP229 – Wider Block Memories	*www.xilinx.com/bvdocs/appnotes/xapp229.pdf*
XAPP291 – Self-Addressing FIFO	*www.xilinx.com/bvdocs/appnotes/xapp291.pdf*
XAPP482 – MicroBlaze Platform Flash/PROM Boot Loader and User Data Storage	*www.xilinx.com/bvdocs/appnotes/xapp482.pdf*
CoolRunner-II CPLDs	
Handheld Consumer Electronics	
XAPP394 – Interfacing to Mobile SDRAM with CoolRunner-II CPLDs	*www.xilinx.com/bvdocs/appnotes/xapp394.pdf*
XAPP905 – Using CoolRunner-II with OMAP, Xscale, i.MX & Other Chipsets	*www.xilinx.com/bvdocs/appnotes/xapp905.pdf*
XAPP914 – Connecting Intel PXA27x Processors to Hard-Disk Drives with a CoolRunner-II CPLD	*www.xilinx.com/bvdocs/appnotes/xapp914.pdf*
Industrial, Scientific, and Medical	
XAPP384 – Interfacing to DDR SDRAM with CoolRunner-II CPLDs	*www.xilinx.com/bvdocs/appnotes/xapp384.pdf*
XAPP385 – CoolRunner-II CPLD I^2C Bus Controller Implementation	*www.xilinx.com/bvdocs/appnotes/xapp385.pdf*
XAPP386 – CoolRunner-II Serial Peripheral Interface Master	*www.xilinx.com/bvdocs/appnotes/xapp386.pdf*
XAPP387 – PicoBlaze 8-bit Microcontroller for CPLD Devices	*www.xilinx.com/bvdocs/appnotes/xapp387.pdf*
XAPP940 – Using Xilinx CPLDs as Motor Controllers	*www.xilinx.com/bvdocs/appnotes/xapp940.pdf*

A Quicker Quick Start

Spartan-3 kits now come with an Out-of-the-Box Guarantee.

by Wil Florentino
Starter Kits Marketing Manager
Xilinx, Inc.
wil.florentino@xilinx.com

Many development systems claim to offer a "quick start" toward feature evaluation and a simple out-of-the box experience. Getting the device set up sometimes takes more energy than running your first program. Unfortunately, getting up and running could mean loading CD files, setting jumpers, downloading code, or hooking up banks of headers. You may need third-party tools or external reference designs. You may even have to go to your local electronics store to get cables and a power supply – all to get to "step one."

To keep things a lot simpler, Xilinx has introduced an "Out-of-the-Box Guarantee" starting with the Spartan™ family of starter kits. When you open the box, you will have all of the key hardware components that will quickly highlight some features of the FPGA device as soon as you turn it on. For example, the newly released Spartan-3A Starter Kit contains a development board pre-programmed with demo software, a global power supply, a download cable, and a resource CD that contains example designs as well as documentation.

When you open the box, all you need to do is plug in the power supply, switch it on, and connect the board to a display; a program already built into the board will run automatically and allow you to quickly sample the device features. For example, it shows how the multi-boot feature works by seamlessly toggling between configuration samples, or the fractal display generator's complex polynomial mathematical capabilities. There is also a demo that reads the chip's DeviceDNA – a unique production-stamped identifier that gives you an additional level of intellectual property security.

In addition, these demonstrations do not just run out of the box. They also are available in source code so that you are free to use and reconfigure the samples as you wish.

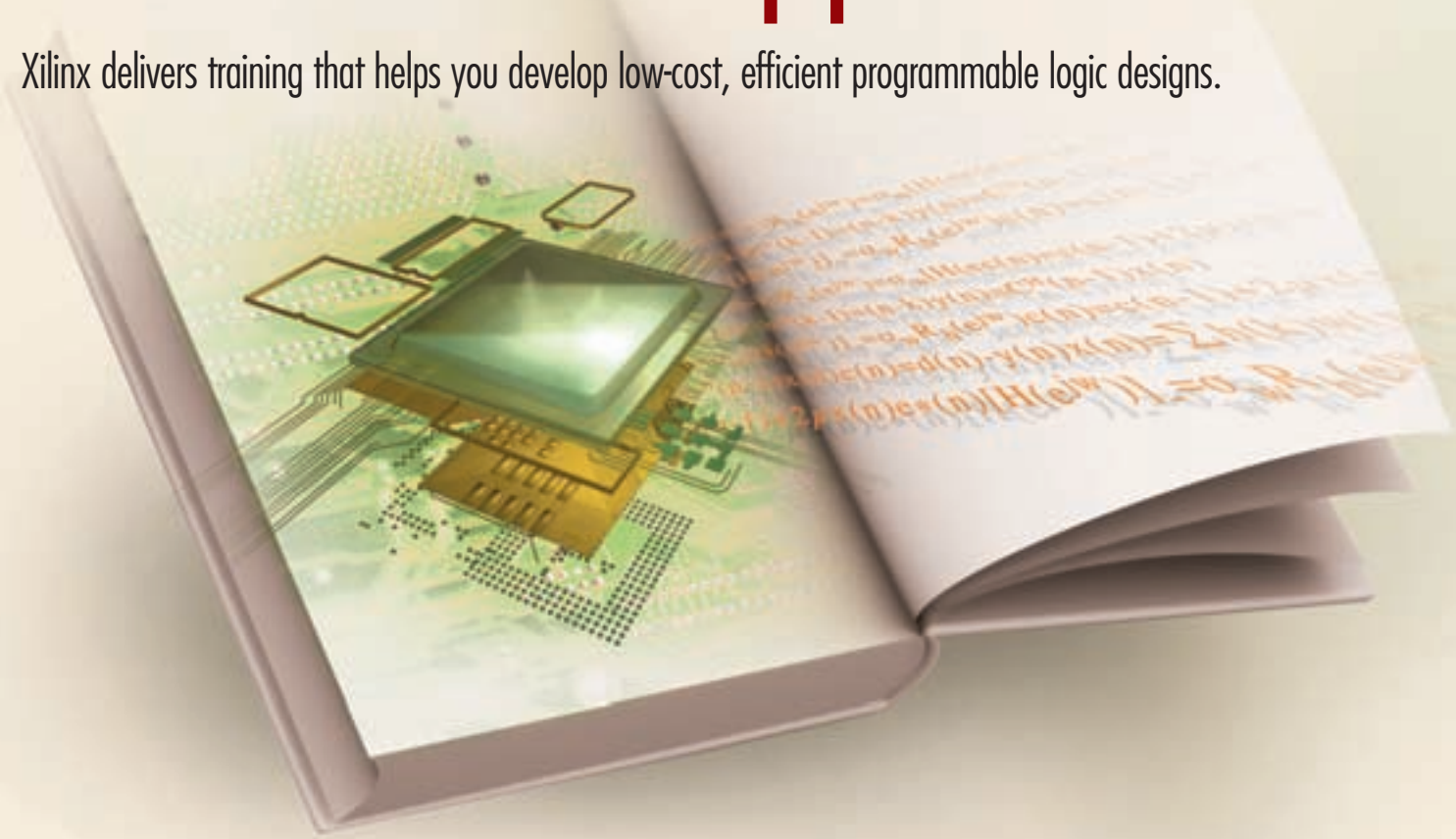
As customers link their technology expertise to FPGA devices, it is best to equip them with hardware and software tools for easy development and provide them a glimpse into device functionality when they start up a system. The Spartan-3A Starter Kit is just one example of how Xilinx provides customers a true "quick start" by providing predictable and complete components for you to see the device's capabilities out-of-the-box.

Spartan-3A Highlights:

- Power management modes
- Enhanced multiboot configuration loads multiple designs from a single PROM
- DeviceDNA identifier is unique to each FPGA
- Spartan-3A Starter Kit speeds development

Educational Opportunities

Xilinx delivers training that helps you develop low-cost, efficient programmable logic designs.



Choose from a variety of flexible training options, including instructor-led classes (both in person and online) and recorded e-learning that allows you to learn at your own pace.

For tuition and registration information as well as class schedules, please contact an authorized training provider at *www.xilinx.com/support/training/atp.htm*. For a complete listing of all Xilinx® training opportunities, visit *www.xilinx.com/support/education-home.htm*.

FPGA Design Courses

- **Fundamentals of FPGA Design** – This course covers ISE™ software 9.1i features, such as the architecture wizard and floorplan editor. Other topics include design planning, implementation options, and global timing constraints. For more emphasis on improving overall design performance, take the follow-up course, Designing for Performance, which builds on the basic principles covered in this course.

On the Web at *www.xilinx.com/support/training/abstracts/fundamentals.htm*

- **Design Techniques for Lower Cost** – This course will appeal to engineers who have an interest in developing low-cost products, particularly in high-volume markets. The course and exercises cover several different design techniques, which will be exciting and challenging for any digital designer regardless of the final application.

On the Web at *www.xilinx.com/support/training/abstracts/low-cost.htm*

CPLD Design Courses

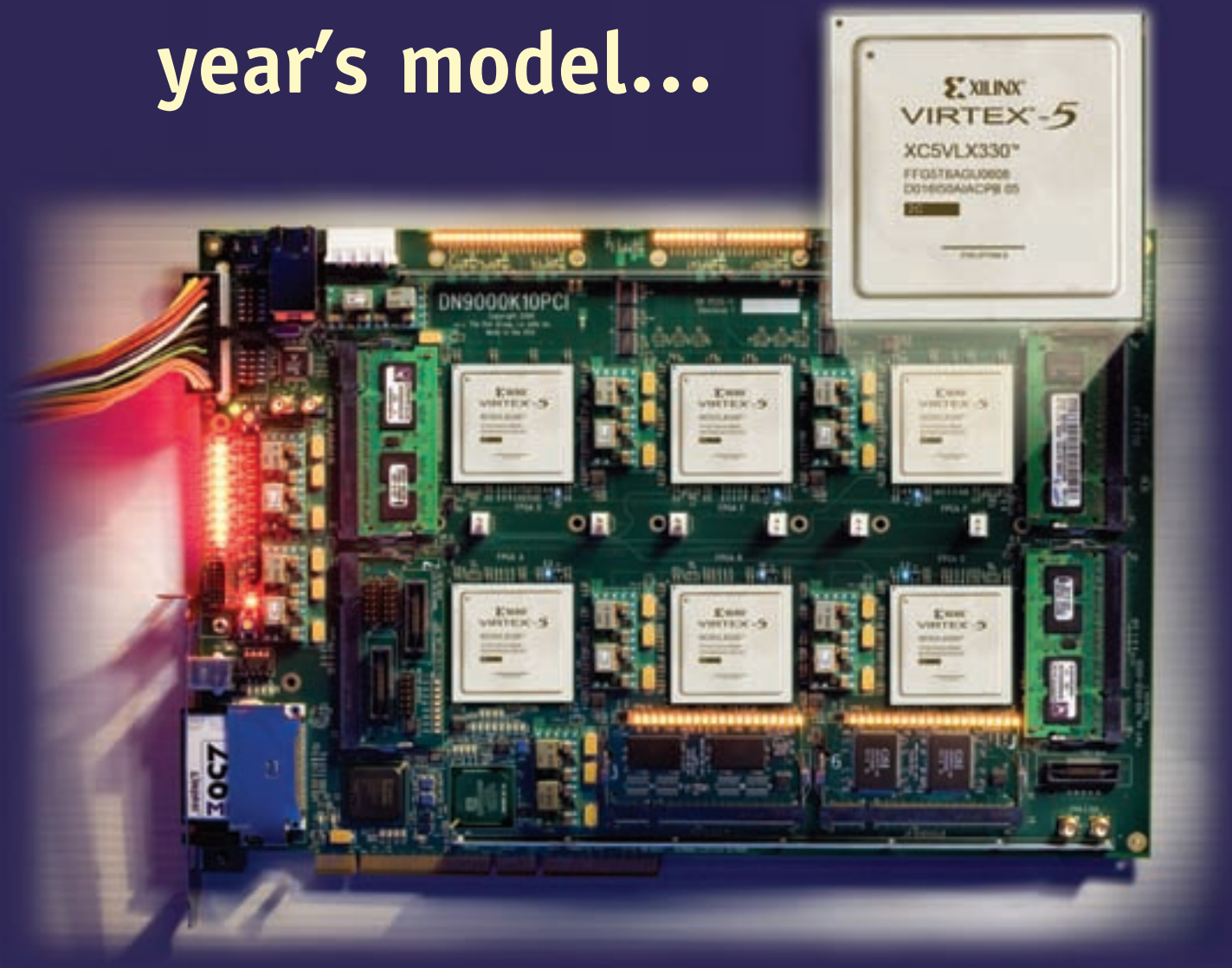
- **Fundamentals of CPLD Design** – This comprehensive course provides you with an introduction to designing with Xilinx CPLDs by using ISE software tools. You will learn the basics of ISE software flow and how to interpret CPLD reports for optimum performance designs.

On the Web at *www.xilinx.com/support/training/abstracts/cpld.htm*

- **Designing for Performance for CPLDs** – This intermediate-level course provides a comprehensive overview on CPLD software flow. By applying the techniques presented in this course, you will be able to enhance the performance of your designs and make the best possible use of Xilinx CPLD architectures.

On the Web at *www.xilinx.com/support/training/abstracts/dfp_cpld.htm*

30% faster than last year's model...



Here are 6 of the new, faster, bigger, Virtex-5 FPGAs on a 12 Million ASIC Gate Board that offers unmatched performance to ASIC Prototypers, IP Designers, and FPGA Developers. The V5 65nm process, with 6 input LUT and advanced interconnect, enables 30% faster clock speeds in your application. The Dini DN9000k10PCI captures this performance on an easy to use board with these handy features:

- 33/66 MHz PCI bus or stand-alone operation
- 6 - DDR2 SODIMM Sockets
- 7 - Global Clock Networks
- 3 - 400pin FCI-MEG Connectors for daughter cards
- Easy configuration via CompactFlash, USB or PCI

All necessary operating software, including reference designs and Synplicity Certify™ models to simplify partitioning, is supplied with the board. If your need is speed — visit The Dini Group web site www.dinigroup.com for complete details on the fastest FPGA board ever.

The DiNI Group

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

Push the Performance of Your Complex Designs with Virtex-5 and Synplify Pro

Synplicity and Xilinx Team Up and Deliver

In May 2006, Xilinx announced Virtex-5 - the industry's first 65nm FPGA. Long before this announcement was made, however, Xilinx knew that an innovative kind of Synthesis tool would be required to enable users to take advantage of the performance and logic density of these revolutionary devices. That's why, once again, Xilinx teamed up with Synplicity.

Engineers at Xilinx and Synplicity have worked closely over the past year to ensure that Synplicity's market-leading synthesis software would provide optimal support for the new Virtex-5 devices. The resulting changes made to the synthesis algorithms allow users to take maximum advantage of these high-capacity Virtex-5 devices.

The enhanced optimizations built into the Synplify Pro software, along with its timing-driven approach to synthesis, allow designers to push the performance of their complex designs while remaining comfortably within their time-to-market goals.

Looking Beyond Today

Future generations of devices will contain even greater density points and capabilities than the current devices, further expanding the reach of advanced FPGA architectures across a wide range of application domains. That's why Synplicity and Xilinx have formed an Ultra-high Capacity Timing Closure Task Force. The purpose of this task force is to enable engineers from both companies to collaborate to define and implement new design flows that maximize the quality of results of design productivity of ultra-high density designs with next-generation 65-nanometer FPGAs.

To learn more about how Synplicity's tools can help you gain maximum performance from your complex designs, contact Synplicity at info@synplicity.com or visit our website at www.synplicity.com.

Synplicity®

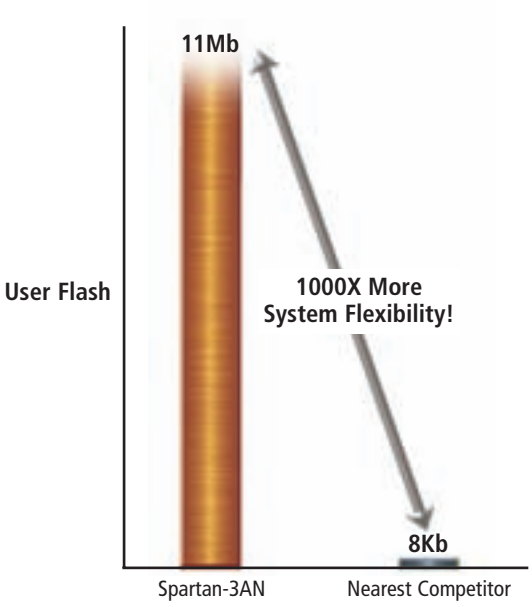
Simply Better Results

VIRTEX™ 5

INTEGRATION

For security and performance, get the best of both worlds.

Introducing Spartan-3AN, a non-volatile FPGA platform for highest system integration.



With all the features and performance of SRAM-based FPGAs, the new Spartan-3AN devices also provide the board space savings and worry-free configurability of non-volatile FPGAs. Each FPGA has up to 11Mb user-Flash, a Device DNA serial number and a Factory Flash ID, enabling customizable security solutions to deter reverse engineering, cloning and overbuilding.

Driving the next wave of high-volume applications

The Spartan-3 Generation of FPGAs gives you all the choice you need to solve any design challenge:

- **Spartan-3AN platform** – *Non-volatile*
- **Spartan-3A platform** – *I/O optimized*
- **Spartan-3E platform** – *Logic optimized*
- **Spartan-3 platform** – *Optimized for high density and pin count*

Visit **www.xilinx.com/spartan** today, and find out how the Spartan-3 Generation of FPGAs gives you the best of all worlds.

XILINX®

www.xilinx.com/spartan

The Ultimate Low-Cost Applications Platform