Xcell journal

SOLUTIONS FOR A PROGRAMMABLE WORLD

FPGAs Power Net-Centric Battlefield on Many Fronts

INSIDE

Make MicroBlaze Processing Roar With Hardware Acceleration
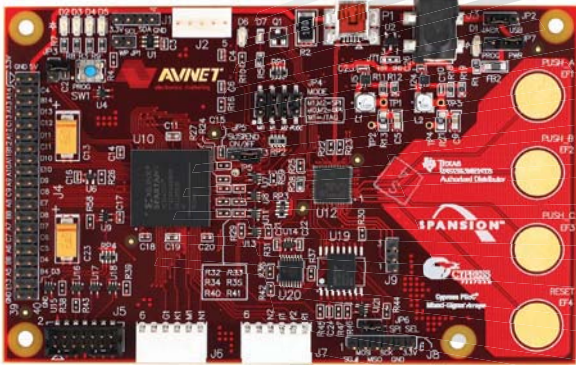
FPGAs Help CERN Track Particles Approaching Speed of Light

Hardware Trumps Software in Medical Device Design

Taming Power Draw in Consumer MPUs

XILINX®

www.xilinx.com/xcell/

Xilinx® Spartan®-3A
Evaluation Kit

DESIGNED BY AVNET

Target Applications

>> General FPGA prototyping

>> MicroBlaze™ systems

>> Configuration development

>> USB-powered controller

>> Cypress® PSoC® evaluation

Key Features

>> Xilinx XC3S400A-4FTG256C Spartan-3A FPGA

>> Four LEDs

>> Four CapSense switches

>> I²C temperature sensor

>> Two 6-pin expansion headers

>> 20 x 2, 0.1-inch user I/O header

>> 32 Mb Spansion® MirrorBit® NOR GL Parallel Flash

>> 128 Mb Spansion MirrorBit SPI FL Serial Flash

>> USB-UART bridge

>> I²C port

>> SPI and BPI configuration

>> Xilinx JTAG interface

>> FPGA configuration via PSoC®

The Xilinx® Spartan®-3A Evaluation Kit provides an easy-to-use, low-cost platform for experimenting and prototyping applications based on the Xilinx Spartan-3A FPGA family. Designed as an entry-level kit, first-time FPGA designers will find the board's functionality to be straightforward and practical, while advanced users will appreciate the board's unique features.

Get Behind the Wheel of the **Xilinx Spartan-3A Evaluation Kit** and take a quick video tour to see the kit in action *(Run time: 7 minutes)*.

BEHIND THE WHEEL

Ordering Information

Part Number	Hardware	Resale
AES-SP3A-EVAL400-G	Xilinx Spartan-3A Evaluation Kit	$39.00* USD (*Limit 5 per customer)

Take the quick video tour or purchase this kit at: www.em.avnet.com/spartan3a-evl

Kit Includes

>> Xilinx Spartan-3A evaluation board

>> ISE® *WebPACK*™ 10.1 DVD

>> USB cable

>> Windows® programming application

>> Cypress MiniProg Programming Unit

>> Downloadable documentation and reference designs

XILINX.

AVNET® electronics marketing

Accelerating Your Success™

1 . 8 0 0 . 3 3 2 . 8 6 3 8
www.em.avnet.com

Quickly see inside your FPGA

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

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

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

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

Also works with all InfiniiVision and Infiniium MSO models.

See how you can save time by downloading our free application note.

www.agilent.com/find/fpga_app_note

u.s. **1-800-829-4444**   canada **1-877-894-4414**

Agilent Technologies

Customer Collaboration, Teamwork Key to Next-Gen FPGA Development

Years of hard work and coordinated effort underlie a new Xilinx architecture. Even before it's launched, plans are under way for the next one.

At the year-and-a-half mark, I've been here at Xilinx long enough to have witnessed first-hand the remarkable effort the company and its network of partners undertake to conceive and design a next-generation FPGA, and the invaluable role you, our customers, play in its creation. The many steps involved in developing one FPGA in pace with Moore's Law is remarkable. Doing two FPGAs at once, as Xilinx did this year, and then creating the Targeted Design Platform strategy to automate customer flows is extraordinary.

Many years before the latest FPGA ends up on your desk, a small army of Xilinx product development experts visit a selection of customers, potential customers and even former customers to understand the requirements of their next-generation systems, what functionality they plan to include and, in the case of the ex-customers, why they switched to an alternate approach or device. We then vet the findings against the ongoing feedback gathered by Xilinx FAEs and sales personnel.

Performing this due diligence with customers across multiple markets provides a basis for making informed decisions about what functions to hardwire into next-generation FPGAs, what soft IP to offer (or which IP partners to work with) and what design tool functionality designers will need. Of course, this undertaking is tempered by the fact that to be successful commercially, an FPGA architecture's feature set must serve the needs of either a broad set of customers or of those playing in high growth markets—or both.

While the customer interviews are under way, the financial and manufacturing teams are doing their own extensive analysis of independent vertical-market and economic data as well as foundry and process node capabilities. Collating all this research, the development experts and silicon architects then start to formulate an architecture and associated tool and IP requirement specifications. A big part of the architectural-specification process is evaluating which of the many new, patented technologies our engineers have concocted to incorporate. So rich are the choices that the biggest challenge by far is determining what not to include.

Indeed, often a given circuit or architecture could be really cool from a hardware-engineering or academic standpoint. However, if the software can't take advantage of it, or if the new circuitry is too complicated for a broad set of users and including it would severely jeopardize a release window, then from a business perspective, it's wise to omit it.

After several cycles of refinement, including reviews with Xilinx's Customer Advisory Board and Field Advisory Board, we finalize the overall architecture specification and the chip architects and designers kick off their intensive design work. Simultaneously, the tool development group begins to assess what tools will be needed to support the size of the new architecture and any new circuitry the FPGA incorporates; the IP group transitions legacy intellectual property and develops (or partners for) new IP; and the development board group starts formulating the base and market-specific boards. With the advent of the Targeted Design Platforms, these groups

must now work even more closely together to create domain- and market-specific platforms (daughter card development boards, IP, reference designs and other documementation) to automate the more mundane design blocks and tasks, allowing customers to focus on the parts of their design that add value.

Once the foundry delivers first silicon, another cadre of Xilinx engineers does extensive characterization and testing. Xilinx's large quality-assurance group works closely with our foundries and prides itself on being thorough and rigorous, to the point of frowning upon a failure rate of one part per million. At the same time, Xilinx trains its FAEs, sales staff and distributors on the feature set and benefits of our new devices, tools and IP.

Even now, the design team doesn't get to rest on its laurels. Besides ironing out any minor kinks in the silicon, its next job is to craft automotive- and A&D-grade versions of the product family, adding circuitry and in some cases new semiconductor materials to the parts—and, thus, further layers of reliability to the silicon. Eventually these devices will spark an even more extensive round of testing, such as temperature-range reliability for automotive and A&D parts along with exhaustive single-event upset testing, in which our SEU gurus fire charged particle beams (neutrons, et al.) at Xilinx devices to test their fault tolerance and the functionality of error-correction software.

Finally, as each product family rolls out of testing, our marketing organization—of which I am a part—pulls together the press materials, promotion collateral and support documentation, updates the Web site and launches the new offering—silicon, boards, tools and related IP—to the public. By the time we're doing that, the due-diligence and even specification creation processes for the next new generation are well under way.

Even as a longtime trade press editor and industry watcher, I didn't fully appreciate all the hard work and coordinated effort it takes to garner the customer input that really shapes a device's future and get a new product generation up and running. The tripling of effort behind the release of the Virtex®-6 and Spartan®-6 FPGA families and the Targeted Design Platforms methodology shows that the company that invented the FPGA is relentlessly striving to perfect the devices you use to create a plethora of remarkable products. These achievements would not have been possible without your invaluable feedback and instruction. So thanks and congratulations—I can't wait to see how you put the fruits of our immense collaboration to great use.

Mike Santarini
Publisher

CONTENTS

THE XILINX XPERIENCE FEATURES

XTRA READING

42

46

56

Xilinx FPGAs to Power Next-Generation Networked Battlefield

FPGAs will play several pivotal roles in the GIG 2.0 network, speeding threat detection and the response time of military forces worldwide.

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

Roughly a decade ago, the U.S. government and its allies established a high-speed worldwide communications network they called the Global Information Grid. In the years since, the GIG has helped the military and various intelligence agencies to quickly identify and assess threats around the world and coordinate appropriate responses. What's more, the GIG allows U.S. and allied forces to communicate with one another on the battlefield and thus perform operations more effectively. This speedy network has also enabled the Defense Department to deploy ever-more-advanced technologies, most notably unmanned spy and assault vehicles that operators can control—in near real-time—from continents away.

But for many threat detection and battlefield scenarios, near real-time simply isn't fast enough, and so today the U.S. Defense Department is in the early stages of giving the GIG a massive performance overhaul. A major part of this overhaul will be the adoption of newer network equipment, which will speed data rates from 4 Gbits/second today to 40 Gbps for wireless communications, bringing real-time communications closer to reality.

Amit Dhir, senior director of Xilinx's aerospace and defense business, said that this move to GiG 2.0 will be accompanied by a massive effort to add localized computing to the many systems and personnel—satellites, airplanes, tanks, ships, unmanned aerial vehicles (UAVs) and foot soldiers—at the edge of the network. The aim is to have UAV, satellite or observation equipment not only take video and photographs, but perform some of the filtering and data analysis locally, on site, before sending it back to analysts in the Pentagon or a local command center. This will make the analysts more effective and efficient in their jobs, paring down what's commonly called "sensor to shooter" time.

"Xilinx has a rich, 20-year history serving the A&D community, and we play a significant role in the GIG's current net-work infrastructure," said Dhir. FPGAs currently are used in the GIG's wired network equipment as well as the communications systems of multiple satellites and UAVs such as the Global Hawk. They are found in the communications systems on the Joint Strike Fighter plane and many other aircraft and land vehicles, as well as the guidance systems of numerous generations of missile platforms.

"All of these systems were developed 10 or more years ago, because the defense development cycle is relatively slow," said Dhir. Some of this technology, developed in-house by defense contractors, pairs ASICs with early FPGA technology.

"Today's FPGAs have advanced tremendously," said Dhir. "In fact, today FPGAs are systems-on-chips." Today's devices have multimillion-gate logic capacity and extreme logic performance, multigigabit I/O, DSP and embedded processing, massive on-board memory, scalable security (encryption and tampering) and reliability that users can further enhance to suit their application requirements. "Where 10 years ago, there was a viable reason to pair an FPGA with an ASIC for GIG-related applications, today FPGAs are so advanced on all fronts, there simply is no good reason to go to the trouble and expense of designing and manufacturing an ASIC," Dhir said. "FPGAs can do the job of ASICs and they can do more—they can be reprogrammed. FPGAs are going to be key to helping the Defense Department provide the right information at the right time to the edge of the network—the front-line soldier."

"The assumptions of the GIG 2.0 are all based on distributed intelligence and wireless real-time links," rather than optical backbones, said Loring Wirbel, longtime communications journalist, blogger and author of the book *Star Wars: U.S. Tools of Space Supremacy* (Pluto Press). "This coincides with the Pentagon's increased use of semi-autonomous robotic vehicles. Where five years ago, UAVs were piloted from centralized bases like Creech Air Force Base

'FPGAs will be key in helping the Defense Department provide the right information at the right time to the edge of the network—the front-line soldier.'

in Nevada and flown via joysticks at workstation consoles, today UAVs are flown from laptops with touchscreens, using special Google Earth-like applications. Within two years, those applications will be ported to devices the size of smart phones."

In light of those changes, said Wirbel, "the need for the increased performance of GIG 2.0 is obvious, and so is the need for distributed intelligence and more parallelism in ever-smaller platforms—areas where FPGAs certainly excel."

Networked-Battlefield Basic Training

Today, the networked battlefield comprises many elements, starting with a sophisticated fiber-optic network connecting various assets to the Pentagon. The network equipment securely relays data to and from satellites, which in turn send it to a series of localized mobile networks or communications bubbles around the world and, more importantly, to front-line military units—ships, submarines, aircraft, UAVs, tanks, trucks, artillery and even individual foot soldiers—enabling commanders and those they lead to communicate in near real-time and more effectively coordinate military operations (Figure 1).

Each element of the GIG has its own unique operating requirements and its individual ways to communicate with the rest of the network. And experts say that

each will greatly benefit from the use of modern FPGAs, which can function as the basis for platforms that users can retask to serve a number of GIG-related applications.

Upgrading Command and Control

The Pentagon's command-and-control network is the center of the United States' military and intelligence operations worldwide. As such, it is the GIG's central communications hub. This network uses high-speed fiber optics to communicate with assets in North America and a high-speed wireless network to issue commands to a series of satellites, which in turn relay data to and from military and intelligence assets and allies worldwide.

Like any modern commercial communications network, the system is composed of sophisticated high-speed compute farms, servers and other network equipment that must be highly reliable, secure and extremely fast. The network must be able to receive and transmit data via many different communications protocols. As such, the same Xilinx FPGAs that manufacturers of commercial comms equipment have been using for decades also play a key role in the Pentagon's network. The major difference is that the Pentagon's network requires additional measures for security and reliability (see sidebar, "FPGAs Make Networked Battlefield Safer, More Secure").

Delfin Rodillas, senior manager for the Xilinx A&D group's avionics, space and high-performance computing segments, said that as the GIG moves to 2.0, Xilinx FPGAs will be even more central.

"A key element of the networked battlefield is the ability to do packet processing at very high line rates," said Rodillas.

Missiles and Munitions
- Used in most of the world's leading missile designs
- Integrated PowerPC processors, DSP engines, logic, and high-speed links
- EO/IR, imaging, and radar real-time DSP processing
- Guidance, control, target tracking, and RF applications

Space
- Space-grade, radiation-tolerant reconfigurable FPGAs
- Leadership in radiation testing with industry-wide consortium
- Software tools and applications support for space and high-reliability design
- Future radiation-hardened-by-design technology

Signals Intelligence
- Supporting the needs of the US and international SIGINT and law enforcement personnel
- Robust signal processing capabilities for reliable detection of signals under extreme conditions
- High-speed transceivers for increased communications channel capacity
- Providing flexibility for systems that monitor and detect signals across a wide frequency range

Radar
- Extreme SWAP-C benefits in active array radars
- Advanced signal processing capabilities
- Ability to upgrade firmware after deployment

"Bringing packets into and out of your chip at high throughputs is critical, as is the ability to inspect the contents of these flows and do some processing on these packets, whether it is classification or traffic management. A lot of our traditional communications customers have chosen FPGAs over ASICs for many years now because of the flexibility and value FPGAs bring for packet throughput and processing."

Traditionally, in the GIG as well as the commercial sector, communications companies have paired communications processors, which direct incoming data streams, with various banks of FPGAs that translate protocols. But as Xilinx FPGAs get larger and transceiver speeds rise, vendors are looking to integrate more of the processing functions into the FPGAs themselves, to further trim system latency and reduce the overall bill of materials (see cover story on wired communications in *Xcell Journal* Issue 67). Similarly, in the GIG 2.0 buildout, the Defense Department can take advantage of these increasing serial line rates—3.125 Gbps in the Xilinx® Spartan®-6 FPGA to 11 Gbps and beyond in the Virtex®-6 HXT—not only to build optimized wired networks but also to speed up the radios and receivers of the many systems that will connect wirelessly to them.

Pioneering Software-Defined Radio

The ability to rapidly receive and translate various signals and protocols, and to do so reliably, is essential in the GIG, said Manuel Uhm, director of Xilinx's Wireless Communications Group and chair of the markets committee of industry group the SDR Forum. Uhm also sits on the forum's board of directors. A given intelligence or military action may involve communication and coordination with allies, who likely use their own secure communications protocols and wireless waveforms for their own networks.

"The GIG actually has to be a meshed network, where everyone is connected to everyone else in multiple ways," said Uhm. "For example, if a unit of allied paratroopers enters the battlefield, the other units on the battlefield and in the network need to identify the paratroopers as friendly. The paratroopers also need to be able to communicate with the rest of the troops and commanders in the action."

Uhm said it's important that the unit be linked to nearby troops in multiple ways—including via direct connection, by a mobile ad hoc (local) network, as well as through satellite connections. "If, for example, a unit is cut off from direct communications because it is in a canyon, the local ad hoc network or, failing that, the satellite network should still make sure they are on the GIG's communications grid," he said. "If they fall off the grid or if a given allied unit's system doesn't recognize the protocol, the allied unit can be mistakenly identified and targeted as an enemy."

To create this connected-to-everyone-in-multiple-ways network ability, the Defense Department has been an early adopter of what's commonly known as software-defined radio (SDR), in which a system's software reprograms the FPGA and other devices in a given system's radio on the fly to receive and send signals in various protocols (see cover story on next-generation wireless networks in *Xcell Journal* Issue 65).

MilSatCom
- Embedded PowerPC® blocks for enabling best integrated processing solution
- Multi-gigabit transceivers for high-rate data transfer
- Largest logic densities, highest-performance and lowest-power solution

Milcom
- Leaders in SWAP-C optimized solutions
- SDR development kits
- Partnerships with key government agencies and program offices

Secure Solutions
- World's first single-chip FPGA cryptographic solution for US DoD applications
- Supporting Cryptographic Modernization Initiative
- Silicon, tools, and IP available today

UAV
- Guidance and navigation control for currently deployed UAVs
- Embedded PowerPC blocks for control plane processssing
- Image processing IP for target recognition and engagement

Avionics
- Infotainment and display systems
- Exhaustive analysis of FPGAs for neutron single event upsets
- IP for scrubbing configuration memory cells

Electronic Warfare
- Chosen by the top-5 equipment manufacturers
- High-performance FPGAs for jamming, self-protection and electronic support measures
- Advanced signal processing and embedded processing

Figure 1 – The Global Information Grid allows commanders and those they lead to better assess threats and coordinate responses to them.

In fact, thanks in large part to Xilinx FPGAs, SDR, with its redundant reliable connections, is one area where the Defense Department has led the commercial communication sector. The GIG already has deployed FPGA-based SDR systems in multiple points on the networked battlefield. As the GIG 2.0 comes online, SDR will be mandatory for all points in the networked battlefield.

Ian Land, senior manager of A&D product planning at Xilinx, points out that because the company continues its relentless pace to offer the most advanced and highest-capacity commercial-, defense- and space-grade FPGAs in the industry, customers will be able to integrate multiple GIG-related functions onto a single FPGA. That will further cut BOM costs and battery/power requirements, while also reducing the size, weight, power and cooling (SWAP-C) of systems.

Xilinx has the industry's most advanced and broadest range of FPGAs, from the space-grade SEU/radiation-tolerant Virtex-4QV, to Virtex-5Q defense-grade FPGAs and commercial-grade 40-nanometer Virtex-6 devices, ranging all the way to the lower-cost, lower-power, yet very advanced 45-nm Spartan-6 family. The latter FPGAs boast a generous amount of 3.125-Gbps serial I/O. Xilinx also gave the Spartan-6 FPGAs AES encryption with Block RAM and eFUSE support, better suiting them to the small battery-operated, handheld devices that connect to the GIG.

In tandem with this year's introduction of its Virtex-6 and Spartan-6 FPGAs, Xilinx took its technology a step further by rolling out its Targeted Design Platform strategy (see cover story in *Xcell Journal* Issue 68). The company offers domain- and market-specific development boards, tools, IP, reference designs and related reference materials to automate the more-mundane design tasks, allowing customers to focus the majority of their efforts on the areas that will differentiate their designs and make them unique. The A&D group is offering a Single Chip Crypto market-specific Targeted Design Platform (see

FPGAs Make Networked Battlefield Safer, More Secure

Reliability and security are important for any application, but for the networked battlefield, they're absolutely critical.

Depending on where in the world they operate and what function they perform, some of the systems in the GIG may have much more stringent reliability requirements than others. Many devices, especially those that operate in space or in high altitudes, must be able to handle single-event upsets (SEUs), caused when charged particles collide with electronic circuitry.

The number of charged particles in the atmosphere (and thus the chance of encountering an SEU) increases with altitude (see figure). If a given part is not radiation tolerant or doesn't employ at least some error-correction mitigation scheme, a single charge can cause reliability problems ranging from delay and the introduction of errors or false signals into a bitstream to actually latching up a device and potentially freezing an entire system (see *http://www.xilinx.com/esp/aero_def/radiation_effects.htm*).

To help, Delfin Rodillas, senior manager for the Xilinx A&D group's avionics, space and high-performance computing segments, said Xilinx offers customers a range of SEU mitigation options, from what he refers to as radiation-hardened-by-test to radiation-hardened-by-design technologies. "We have done a lot of testing to characterize the FIT [failure in time] rate of our devices under neutron SEU effects," said Rodillas. "In addition, Xilinx has also developed reference designs and IP to help with SEU detection and correction. For example, we have a Virtex-5 SEU macro that leverages some of the built-in features of the silicon." They include cyclic redundancy checks and frame ECC. In addition, "we have a macro for error correction. This further increases the reliability of our devices if they encounter SEUs," he said.

Rodillas said Xilinx's Triple Module Redundancy (TMR) tool allows customers to design further layers of SEU protection and reliability into their products. Essentially, the triple redundancy becomes useful if a device were to be hit by a large or multiple charged particles at the same time, creating a multibit upset, said Rodillas. "If one circuit fails, the two other redundant circuits do a voting function to identify, and self-correct, the leg that has been corrupted. All of this is done without disrupting the device operation."

For space-borne and other extremely high-reliability applications subject to radiation, Xilinx offers IP that can constantly correct the configuration of the FPGA, again without disrupting device operation. Rodillas notes that while these SEU mitigation and redundancy technologies, IP and methodologies were originally developed for A&D applications, they are also starting to find use in commercial systems such as high-speed computing and communications, where SEU mitigation is becoming a growing requirement.

Xilinx is a leading researcher of SEEs (single-event effects) on semiconductors. Visit *http://www.xilinx.com/esp/aero_def/see.htm*.

http://www.xilinx.com/esp/aero_def/crypto.htm) and will release a D0-254 (avionics reliability standard) market-specific TDP (see *http://www.xilinx.com/esp/aero_def/do254.htm*) to help A&D customers get products to market faster.

FPGAs Deliver Localized Computing

In addition to upgrading the GIG network infrastructure, the military and its many contractors are just now starting to add localized computing to the many assets that connect to, and communicate by means of, the GIG.

Traditionally in the networked battlefield, said Land, a military or spy unit will send data in a live feed through the network to the Pentagon, where analysts download, decrypt and analyze it and then respond. "A lot of customers are telling us that a problem with sending a signal through the network is that along the way, it can pick up a lot of noise," said Land. "If the processing is done quickly in the battlefield unit and then sent back through the network in digitized form, it can save a lot of time [analysts would otherwise spend] cleaning up and analyzing the photos, video images or messages at the command-and-control center."

Further, a spy vehicle may take several thousands of photographs and several hours of video. Analysts have to wade through all this material to find those photos and video segments that are of interest. Adding more localized computing or on-board processing in the surveillance craft itself will allow the craft to narrow down which photos or video segments are potentially of interest, and send only these to the analysts. Analysts would be able to identify threats more quickly and the military potentially could respond more rapidly and effectively to threats as they arise. "FPGAs are great at doing that kind of signal processing and doing it in real-time," said Land.

Also, doing more of the processing locally reduces the overall amount of traffic on the network and improves the network's overall performance.

From Foxhole to Satellite

In addition to reliability, security is another essential requirement of the net-centric battlefield. Many assets connected to the network must be able to communicate securely.

For communications, from a warfighter in a foxhole to the big pipes of a satellite, the Defense Department has set encryption requirements at every point in the networked battlefield. In addition, communication and interoperability between allied forces can be challenging, as each country uses its own encryption schemes and each has its own requirements for a given operational environment.

Xilinx's A&D group pioneered a huge leap in FPGA-based encryption back in 2006, when it delivered the industry's first single-chip encryption technology, which allows the defense industry to use FPGAs in cryptographic systems in a much more effective and efficient manner than they have in the past.

Prior to the availability of the single-chip product, cryptographic systems would use multiple chips to add layers of redundancy and reliability. Those multichip solutions were not extensible, and they increased a system's size, weight, power and cooling (SWAP-C). Xilinx's revolutionary one-chip crypto solution allows customers to do all the cryptographic functions in a single Xilinx FPGA while still meeting the government reliability requirements (see *http://www.mil-embedded.com/pdfs/NSA.Mar07.pdf*).

Cryptographic applications are only a small part of the general security concerns within the GIG. Physical security, or anti-tamper technology, is another area contractors must address. Xilinx, the first company to introduce bitstream
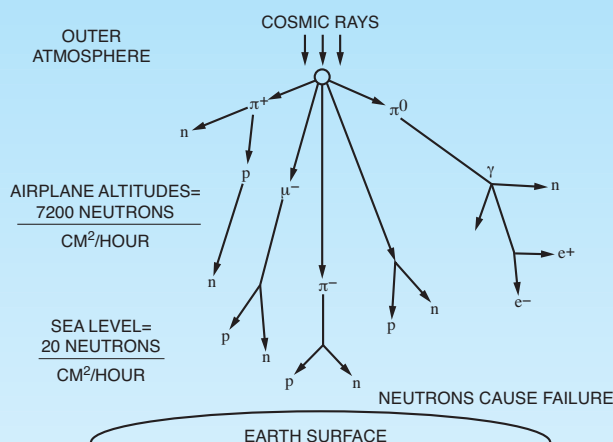
Neutron concentration and the likelihood of encountering a single-event effect is greatest at an altitude of 50,000 feet. At 30,000 feet, the typical cruising altitude for commercial aircraft, ICs are 300 times more likely to encounter an SEE than at sea level.

encryption, recently expanded its lead in FPGA security by introducing bitstream authentication as well as encryption in the Virtex-6. Besides benefiting defense applications, silicon enhancements to improve security, such as authentication, also address commercial concerns about protection of IP, piracy, cloning and the like. Xilinx has not only added enhancements within the silicon, it has also invested in the development of IP that further increases the security of the FPGA. — *Mike Santarini*

Figure 2 – The Global Hawk unmanned aerial vehicle typically flies reconnaissance missions at 60,000 feet and can refuel in midair to stay aloft many days at a time. It is one of a growing armada of unmanned vehicles remotely controlled via the GIG.

Expanding the Role of Unmanned Vehicles

Of the many assets that connect to the GIG, unmanned vehicles will likely see the greatest boost from an increase in network performance and from localized computing.

Ching Hu, senior engineer of strategic applications in Xilinx's A&D group, said that over the last decade, the military has placed greater emphasis on the development and deployment of unmanned vehicles. Today the military is deploying not only UAVs like the Global Hawk, Predator and Reaper, but also an array of unmanned ground vehicles (UGVs) and, in the case of the Navy, unmanned surface vehicles and unmanned undersea vehicles (respectively, USVs and UUVs).

"One of the mottoes the Air Force and the Navy use today is 'Persistence in the sky 24/7/365,'" said Hu. "What that means is that the military wants to be pervasive and persistent in the sky. For example, the U.S. Navy has a UAV program called BAMS [broad aerial maritime surveillance], a Global Hawk variant that provides coverage from multiple launching sites, where they can almost cover the entire globe. At all times they have a UAV flying at 60,000 feet and taking pictures continuously. The great thing about a UAV is that you don't have to worry about it getting tired—you just have to make sure it has fuel to keep going. It is one advantage we didn't have in the past."

Hu notes that unmanned vehicles can take on extremely dangerous and rigorous missions that in many cases would tax the endurance of human pilots and put their lives at risk. In some cases, UAVs perform missions that humans simply cannot physically do, such as rapid and violent threat evasions that require the craft to sustain a high G-force level, or simply fitting into tight spaces that cannot accommodate a piloted craft.

The Pentagon is also broadening the roles of the unmanned vehicles. Where the military initially deployed them mainly for surveillance, bomb detection and defusal, it is increasingly using the craft to deploy ordnance and actually attack targets.

Both armed and unarmed unmanned vehicles are remote controlled, "piloted" by personnel who in some cases may be very near the craft's area of operation and are running it via a direct signal—or who may reside on the other side of the globe, controlling the vehicle via the many connections of the networked battlefield. Because of the distance, there is a delay between what the craft sees and what its controller sees.

As a result, said Hu, operations can miss targets of opportunity. Also, most UAVs have to be landed by a separate operator close to the runway where they are to touch down, because the lag through the network could cause a distant operator to overshoot a landing strip and cause a crash. As a matter of government policy, UAVs cannot land at civilian airports because of the risk of crashing into commercial planes, airport structures or nearby homes. In fact, UAVs usually have their own designated military airfields, to keep soldiers and military equipment out of harm's way. A crash can be especially devastating if the vehicle is still carrying ordnance.

Because of this danger and the desire to shrink the sensor-to-shooter chain, the military is always looking for ways to speed up communications and to add more localized computing and more sophisticated sensor arrays to these UAVs. The technology is designed to land unmanned vehicles more safely and consistently, while enabling them to react more quickly to targets, sort data of interest and perform threat detection and evasion maneuvers in real-time.

Hu notes that customers designing unmanned vehicles could greatly benefit from Xilinx's many years of deployment in advanced sensor systems. In UAVs, the use of multifunction sensors teamed with localized computing could not only help shorten the sensor-to-shooter chain but also assist the craft in assessing and evading threats in real-time, and help it land more precisely, perhaps without the need of a local operator.

"These same FPGA technologies can usher in a new era for UAVs and the rest of the systems connected to the networked battlefield," said Hu.

Moving forward, the ever-increasing sophistication of FPGA technologies and the inherent flexibility they provide will give the U.S. military and its many contractors the opportunity to more rapidly mature and improve all points of the networked battlefield, while streamlining communications. It's not hard to foresee a day when, thanks to FPGA localized computing, the military will be able to perform threat detection and identification and mobilize the appropriate response (diplomatically or, if need be, militarily) so rapidly that it will defuse threats before they escalate. That will bring U.S. forces closer to the ideal of military preparedness defined by George Washington, namely, that "There is nothing so likely to produce peace as to be well prepared to meet the enemy."

For further information on Xilinx's A&D offerings, visit *http://www.xilinx.com/esp/aerospace.htm.*

Why Software Programmability of Electronics is a Game Changer

The architecture of the future has an embedded microprocessor at its heart and a series of hardware accelerators to handle particular tasks.

by Jacques Benkoski
USVP Venture Partner;
Executive Chairman,
Synfora
jacques.benkoski@synfora.com
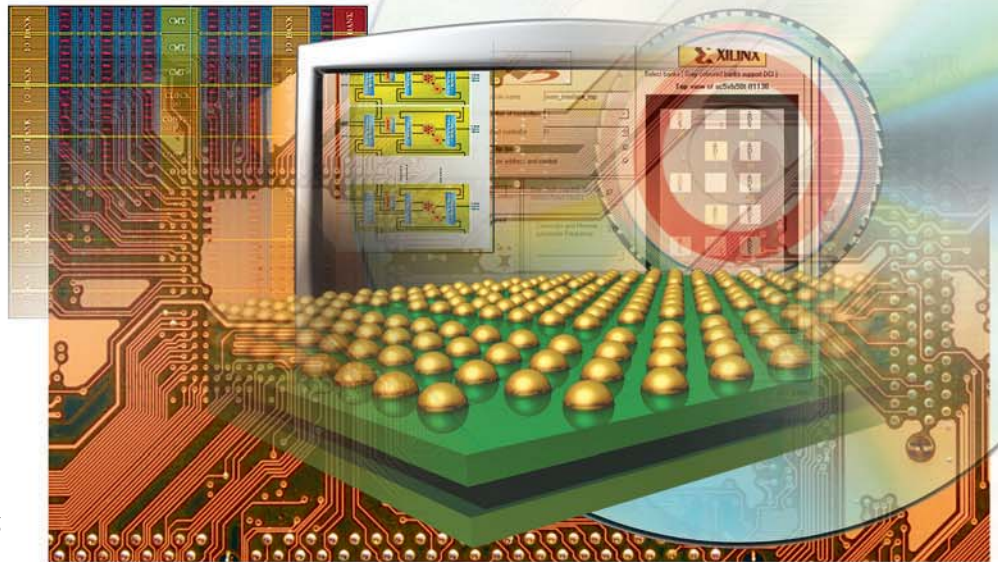
The common wisdom says that the semiconductor industry is in serious trouble, as increased design cost, growth in the amount of intellectual property (IP) necessary to put a product together and soaring mask costs combine to make any but the highest-volume chips economically unfeasible. This view foresees a world populated by microprocessors and a few consumer systems-on-chip (SoCs), with the FPGA vendors focused on serving the low-volume price- and power-insensitive applications.

In fact, the drive to enable software programmability of integrated circuits will turn that conventional view upside-down. Entering the scene are tools coming from the compilation world that are able to map the software description onto automatically created hardware extensions. The electronics are, in fact, software programmed and automatically compiled, without going through the equivalent of assembly language that RTL represents.

The resulting architecture is clean, with an embedded microprocessor at the heart of the IC and a series of hardware accelerators neatly brought to bear on the appropriate tasks. This is true of a dedicated ASSP or ASIC, but is equally applicable to a modern FPGA with embedded cores.

The Problem with Multicore

Executing a given task in software on a generic processor architecture, as opposed to dedicated hardware, has been proven to be two orders of magnitude off in cost, power and performance. The pressure to reach low-power solutions is now pervasive beyond mobile applications. With all due respect to the embedded cores, they would have to run at impossible speed to swallow a 5-Gbit/second stream, perform live video transcoding or aim a beam-shaping antenna array.

Although some argue that multicore is the solution, with dedicated cores earmarked for specific applications, the multicore programming problem remains stubbornly elusive beyond the simple multithreading on an identical architecture. Nobody seems able to master the unbounded complexity of heterogeneous processing engines with different characteristics communicating over ill-characterized buses and networks-on-chips. At the same time, the number of software engineers continues to grow and the number of hardware engineers continues to shrink—at least on a relative basis. Yet in most cases, companies give away the software as a necessary component of a platform rather than as the valuable differentiator it is made to be.

FPGA as Distributed Compute Engine

The FPGA world has been undergoing its own revolution. No longer simply seen as gobs of glue logic, FPGAs have now emerged as an attractive alternative implementation for many applications focused on power, price and performance. This alternative is now making its way into consumer and even mobile applications. Yet FPGAs also emerge as a fascinating distributed compute fabric with a regular architecture of computational elements and memories. They suddenly represent a quasi-systolic-array alternative to the Von Neumann digi-

tal processors, with a much more attractive performance, cost and power trade-off.

This contrast between microprocessor- or DSP processor-based designs and FPGA as a distributed compute engine is hard to understand. To get a grasp on it, one should try to imagine the amount of hardware resources that are being used to fetch data from memory, bring it to a datapath inside the processor and then put the data back in memory. To achieve the best performance, this Von Neumann processor architecture implies further hardware resources, such as branch prediction and speculative execution, that are wasteful of silicon real estate as well as being power hogs.

Now imagine that these repetitive loops are unrolled from the time domain and mapped to a two-dimensional array of distributed compute and memory resources available in an FPGA. The data flows from one side of the array to the other and is processed efficiently—this is the basic reason why software running on a processor is always inherently much less efficient that its equivalent mapped onto hardware. However, historically the FPGA hardware implementation always had to go through the equivalent of assembly coding—that is, RTL-level design—whereas the processors had compilers and high-level languages.

These trends all converge on the compelling need to enable software programmability of ICs. Not the kind of programmability that the original silicon compiler pioneers had dreamed of, but one that has evolved from the world of very long instruction word (VLIW) compilation and the world of hardware accelerators.

Microprocessor designers have long known that it pays to generate additional instructions to cope in hardware with either common or costly software routines. Back in the 1980s, we had Intel's 8087 math coprocessor, which later evolved into the more elegant multimedia extension (MMX) to the famed Intel Architecture. More recently, as ARM cores became ubiquitous, designers have been quick to recognize that they could offload in a similar way the more tedious tasks that the embedded micro-processor could not cope with. The path

from there to the most recent developments—those that I believe are game changing—was natural, at least in hindsight.

Built around embedded microprocessors and accompanying hardware accelerators, the new architecture radically changes the conundrum that traditional ICs—both SoCs and FPGAs—were facing by enabling a way to capture all the software horsepower and differentiation, and to map it onto hardware. This architecture also tames the heterogeneous multicore programming problem, as each hardware accelerator is

yet offering a similar clean programming model. However, there is no reason to erect an unnecessary wall between the two worlds, as many applications will be ideally served with an off-the-shelf SoC and a companion FPGA as a hardware accelerator for specific tasks.

One can now see a continuum of solutions, from software running on processors to processors with hardware accelerators on SoCs; from SoC/FPGA coprocessing solutions to FPGAs with cores and hardware accelerators; all the way to pure FPGA-as-

> ## FPGAs can now be a formidable competitor to DSPs and other dedicated machines, with a better power, cost and performance trade-off, yet offering a similar clean programming model.

seen as an accelerated subroutine. That means you can map all the software differentiation into hardware, minus the headache. The work of the system software engineers, which contains so much of the differentiated and unique IP of semiconductor and system companies, can be captured elegantly. And since it is now delivered in a differentiated, low-power, high-performance hardware package, it allows the manufacturers to actually get paid for their IP.

The outcome is a new wave of rapid product introductions of complex and targeted solutions for ever-more-powerful consumer electronics, myriads of mobile Internet devices, gaming-platform breakthroughs and automotive infotainment opportunities, to name only the most obvious.

Tear Down the Wall

These compiler tools have an even more dramatic impact on the FPGA world, since the software maps cleanly using the same paradigm on these devices' prearchitected distributed compute platform. FPGAs can now be a formidable competitor to DSPs and other dedicated machines, with a better power, cost and performance trade-off,

compute machines. The availability of a programming paradigm is the unifying factor underlying them all.

Already today, the DSP processing capability embedded into SoCs has surpassed that of the discrete DSP, as noted in the "DSP Silicon Strategies" report issued earlier this year. Meanwhile, the ITRS has predicted an explosion in the number of embedded accelerators. It is not clear if they are to be called "software-programmable integrated circuits" or if they will ever have a name. But I think if one steps back and looks at the overall landscape, that trend is obvious, inevitable and game changing.

Jacques Benkoski is working with several U.S. Venture Partners companies and currently serves as the executive chairman of Synfora. Before joining USVP, he was CEO of Monterey Design Systems. Prior to that, he held technical and management positions at Epic Design Technology, Synopsys, STMicroelectronics, IMEC and IBM. He holds a BSc in computer engineering from Technion Israel Institute of Technology, and an MS and PhD in computer engineering from Carnegie Mellon University.

How to Tame the Power Beast in Consumer Handheld MPUs

Using an FPGA is a popular way to expand the capabilities of an embedded processor. Designers can reduce power consumption at the same time.

by Rahul V. Shah
Director of Customer Solutions
eInfochips
RahulV.Shah@eInfochips.com

Vishesh Agrawal
ASIC Design and Verification Engineer
eInfochips
Vinesh.Agrawal@eInfochips.com

The consumer handheld market is growing by leaps and bounds. With more processing power and increased support for more applications, portable products are cross-pollinating with traditional computing systems even as the product life cycle has decreased considerably in this market segment. As a result, especially in this era of economic slowdown, it is imperative that new products meet the time-to-market window to gain maximum acceptance. A decrease in product life cycles requires a reduced development cycle and an increased emphasis on reusability and reprogrammability.

The emerging handheld market is also seeing interesting trends in which each individual device in a family has lower volumes but there is more customization across the series of devices, effectively upping the total unit volumes. The key challenge then becomes how to develop a system that is widely reusable and also customizable.

These requirements have led designers increasingly to turn to the FPGA for hand-held-product development. The FPGA has become more powerful and feature-rich, while gate counts, area and frequency have increased. FPGA development and turn-around cycles are considerably shorter than those of custom ASICs, and the added advantage of reprogrammability can make the FPGA a more compelling solution for handheld embedded systems.

In an ASIC- or an FPGA-based design, designers must take certain performance criteria into account. The challenges can be stated in terms of area, speed and power.

As with the ASIC, vendors are taking care of the area and speed challenges in FPGA design. With increased gate counts, the FPGA has more area and size to accommo-date larger applications, and tools include better algorithms to utilize the area optimal-ly. For example, technology advancements with newer standard-cell libraries have led to FPGAs achieving higher frequencies.

The newer and better FPGA technology brings with it a whole new set of challenges for the designer. Power utilization is one issue that moves to the forefront when designing an FPGA-based embedded sys-tem for a handheld or portable device.

FPGA in an Embedded System

A typical embedded system consists of a processor, memory, standard interfaces like USB, SPI, I²C and so on, along with periph-erals such as liquid-crystal display, audio-out and the like. The heart of the application still resides in the processor and the proces-sor interfaces, with various peripherals using onboard connections. The performance of the system depends on the performance of the processor, which has a very standard architecture and is not easy to customize.

The processor may also end up utilizing its activity time in processing information from a slower peripheral (for example, read-ing data from a slower ADC audio on an I2S interface). Although the processor usage may reach 100 percent in this case, the device is not doing microprocessor-centric activities but is working at a significantly lower performance level. Irrespective of its core frequency, the microprocessor must wait for the data from a slower clock. This also results in higher power consumption, as the processor is showing full utilization. The result is lower battery life along with larger
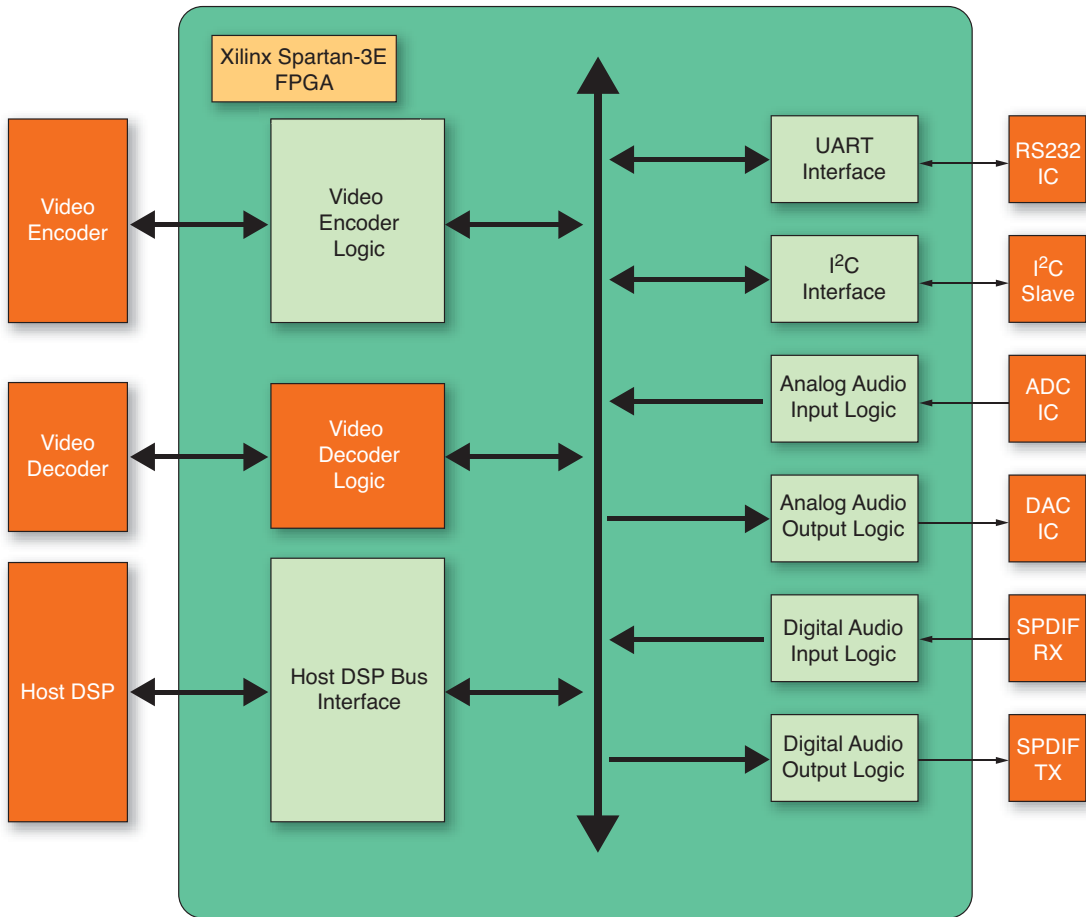

Figure 1 – FPGA architecture for audio/video distribution system

One technique to minimize static power consumption is to perform power estimations early in the design cycle and, if feasible, change the topology or use a different IP block. Xilinx's xPower Estimator tool signals very early in the design process whether you can meet the power budget.

heat sinks or fans for cooling, eventually affecting the reliability of the entire system.

FPGAs have started to play an important role on this front, since they can offload the processor of many of its peripheral interaction duties. For example, the embedded distribution system of an uncompressed audio/video stream through a standard Gigabit TCP/IP network, shown in Figure 1, has a dedicated DSP processor interfacing with a Xilinx® FPGA over a standard bus interface. The FPGA is connected to various slower peripheral devices.

For starters, it is interfacing with a 12-bit PCM audio input and a 12-bit PCM audio output, both over I2S. It also interfaces with a video encoder and decoder and communicates with I2C slaves and RS232 devices. There are few general-purpose I/Os connected to the FPGA. The standard bus operates at a high speed of 66 MHz, while the audio peripherals work at a low speed of 1.182 MHz. Serial interfaces like UART and I2C operate at 56.6 kHz and 100 kHz respectively. The data transfer takes place over multiple clock domains with only the processor configuring the data flow.

In this case, instead of the processor interacting with the slower peripheral device, the FPGA can read data from the slower PCM ADC audio and store it in its internal buffer. Either the processor can read that buffer periodically or the FPGA can send an interrupt to the processor when it has sufficient data. The processor now has more idle time in which to do processor-centric work if required; otherwise it can go into sleep mode during idle time.

Power Consumption

In a battery-operated embedded system, power saving is of the utmost importance.

Power consumption can be categorized into three areas: startup power, static power and dynamic power.

The designer cannot control startup power consumption, which plays an important role in deciding on the power supply selection. Most of the maximum current level drawn is achieved in this phase.

But static and dynamic power consumption are two areas where, with proper planning and by following correct guidelines, the embedded designer working with FPGAs can make a marked improvement in power optimization.

Static power is the flow of current through a component even when there is no activity in the system, generally due to device bias and leakage. Static power also depends upon operating voltage. Reducing the operating voltage will reduce the static power. But this decision is not always in the hands of the designer.

What the designer can do is to define the architecture in such a way that requires the least amount of resources, to use resource sharing whenever possible and to use the FPGA blocks in the most efficient manner.

Another technique to minimize static power consumption is to perform power estimations early in the design cycle and—if required and feasible—change the topology or use a different IP block. Xilinx's xPower Estimator tool is useful in knowing very early in the design whether you can meet the power budget. Early-stage power estimation may not be totally accurate, but it does serve as a guiding tool.

Dynamic Power Consumption

Dynamic power consumption is the result of some activity on the FPGA gate—namely, signal switching—when both gates are switched on briefly, causing current flow and capacitance. The speed of the signal

switching will determine how much power is consumed. Another factor that determines dynamic power consumption is the inherent capacitance created within the geometry of the circuitry.

Dynamic power is a function of the clock rate, the number of gates that are switching and the switching rate of these gates. The capacitive load on the gate fan-out and wire add to the dynamic power consumption, which is proportional to the product of capacitance, voltage and square of frequency.

The designer has maximum control over this type of power consumption, with access to many techniques that will reap the maximum benefit when it comes to dynamic power.

Reducing the signal switching frequency slashes power consumption exponentially. As in the reference design shown in Figure 1, the control logic for the UART, parity check or frame overrun error occurs in the slower clock domain. Even though there is no reduction in the gate count, the power consumption falls. Designers can also reduce dynamic power consumption by lowering the overall operating frequency, if feasible. After doing the feasibility and performance analysis, for example, the designers decided that instead of working at 133 MHz, the above design also works at 66 MHz. The DSP supported both those speeds, and reducing the voltage also helped in dissipating less power.

Another technique is to reduce the number of active gates in an operating mode. Sometimes a part of the logic, though switched on and configured at power-up, is not required to actually do anything. If, for example, the analog audio capture unit is active, the application is not performing any activity on digital SPDIF audio capture. In this case, the typical digital SPDIF audio capture still performs data

sampling, biphase decoding and so forth, which is an unnecessary power drain. If the entire digital SPDIF audio capture circuit is disabled so that no signal switching takes place in the circuit at all, the result will be a big reduction in dynamic power.

You can achieve this by disabling the clock that propagates to that portion of the design. A simple way of doing so is to AND the clock signal with an enable signal, shown in Figure 2. If the enable signal is low, the output of the AND gate will stay low. If the enable is high, the output of the AND gate follows the clock.

Other methods may also be applicable. If possible and supported by topology, multiplexing address and data lines can reduce the number of signaling lines. In our example, the output to the video encoder was 16-bit data, which we multi-

can adopt, with manifold benefits. First, it reduces the challenge of customizing the processor, as discussed earlier. Second, the interaction between the peripherals and the processor resides inside the FPGA and reduces the I/O count. Since I/Os use considerable power, this also results in some amount of power saving. Xilinx's Virtex®-5 edition supports PowerPC® 440 processors, hard processors and MicroBlaze™ soft processors, all of which designers can leverage to make any system with high-end or low-end applications in mind.

Xilinx has been at the forefront in this field and its latest FPGAs offer many power optimization features. The company's power analysis and power estimation tools, xPower Estimator and xPower Analyzer, have many features that help the designer in crafting a low-power FPGA system.
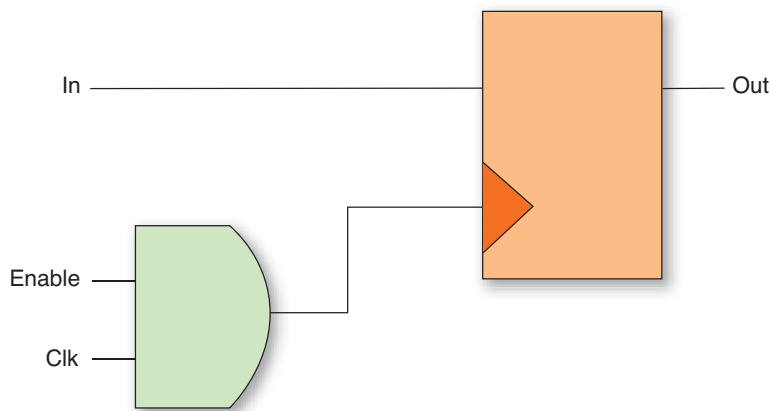
Figure 2 — A simple clock-gating mechanism

plexed over 8-bit and sent over both the edges of the clock. This also resulted in saving dynamic power. Choosing a serial rather than a parallel interface will also lower the dynamic power. Using LVTTL or LVCMOS I/O with a lower capacitive load helps too, although the designer may not always be the one to decide on the I/O.

Embedded Processors

Embedding the processor inside the FPGA is another strategy that handheld designers

With the advent of 90- and 65-nanometer semiconductor technologies, the gates are shrinking in size, resulting in manifestation of static power consumption—a challenging phenomenon given the growing sensitivity to power metrics. There is a lot of excitement in this field as power issues gain prominence with various FPGA vendors. Low-power design will determine how much more we can pack into one system. There is a dire need to standardize a design methodology that focuses on power consumption.

FPGAs Help Measure Trajectory of Particles in CERN's Proton Synchrotron

System processes 15 billion analog samples per second to track the path of particles traveling close to the speed of light.
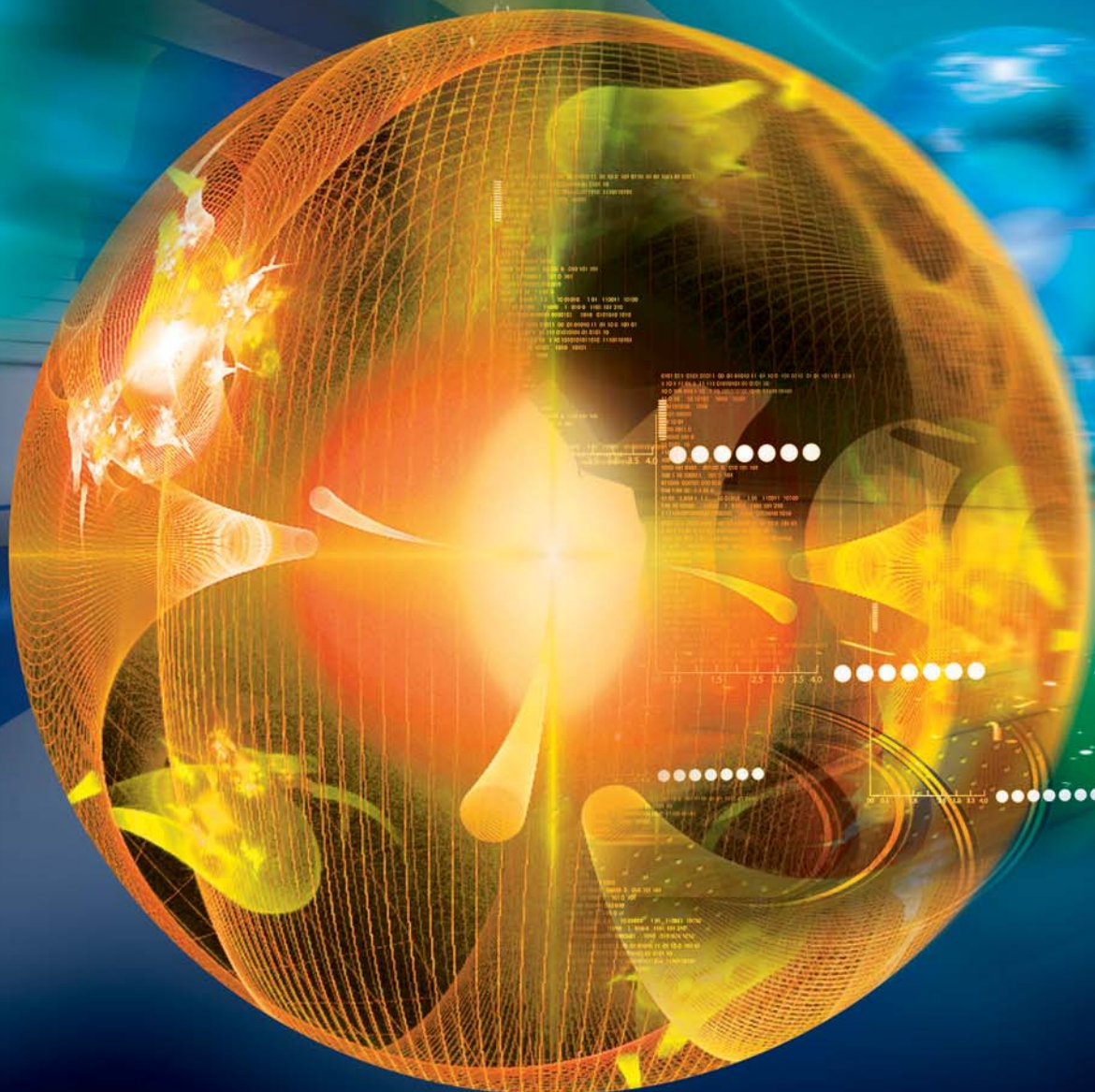
by Terry Barnaby
System Engineer
Beam Ltd.
terry@beam.ltd.uk

A particle has a hard life at CERN. A proton starts its journey as a hydrogen atom in a small, common-looking red bottle the size of a soda siphon at the head of an imposing 80-meter linear accelerator. With its electron stripped, electric fields accelerate it to about one third of the speed of light. It soon enters the 200-meter-diameter Proton Synchrotron (PS) machine, where further acceleration and conditioning take place, with other protons, as a beam. After that, the particle is sent to a bigger machine such as the Large Hadron Collider or directly used in an experiment. Normally its journey will end in a near-speed-of-light collision with some other unsuspecting particle. At every stage along the way, various sensors and high-speed data-processing engines are spying on the proton, not least in the PS machine.

A versatile juggler of particles, the Proton Synchrotron (Figure 1) is a key component in the accelerator complex at CERN, the European Organization for Nuclear Physics in Geneva, Switzerland. The PS machine accelerates and manipulates protons or heavy ions for various experiments. Apart from system downtimes, the PS runs continuously, operating on a different particle beam, for the various ongoing experiments, every 1.2 seconds or so. During operation it all looks quiet around the PS ring's complex. All that an observer can notice of the particle motorway traffic within the evacuated tubes is a loud and heavy surging buzz from the multiple large power transformers outside every 1.2 seconds as megawatts of power surge to supply the system.

One of the many important instruments required for this machine is the Trajectory Measurement System. The purpose of the TMS is to measure the track of the particles as they race around the ring at nearly the speed of light. There are 40 sets of metal plate detectors spaced around the PS ring, providing 120 analog signal channels. Each of these signals, three per detector pickup,

needs to be digitized at 125 megasamples per second. That is an overall data rate of 15 billion samples, or 30 Gbytes of data, per second. Due to this high data rate, there is a need to process the data in real time, to reduce the amount of data and pick out the information of interest before storing it.

CERN's scientists and engineers had devised the basic processing algorithms suitable for FPGA implementation. For engineers at Alpha Data and at Beam Ltd., our job was to develop the idea and pro-

generally employ a generic Linux-based computer as the main control and data-access module, with a number of FPGA-based modules at the front end to do the fast data capture and real-time processing work. The TMS is based on that system model. It comprises a Linux-based system controller that provides control, data postprocessing and external data access, and four, eight-slot, CompactPCI rack modules, each with a Concurrent Technologies Intel dual-core processing

Figure 1 – CERN scientists accelerate and condition particles in the ring-shaped Proton Synchrotron.

duce a complete working system in a reasonable time frame and at reasonable cost to do the job.

Designing the TMS
From the early stages of design, we envisaged a very modular system that would provide fault tolerance and ensure easy live maintenance. It also had to be flexible and expandable. We took the concept of modularity down even into the FPGA fabric, where each FPGA implements three separate pickup-detector channel blocks.

Alpha Data and Beam have worked jointly on a number of real-time, FPGA-based instrumentation systems. These

card running Linux. Within the rack modules are specially designed, FPGA-based, Pickup Processing Engine, or PUPE, cards (Figure 2).

When developing such real-time systems, especially with FPGAs, it is important to decide what work the various system modules will handle. FPGA hardware is excellent for doing simple, real-time, repetitive things in parallel and at relatively high speed. Software running on a conventional processor is good for control and data access as well as flexible data postprocessing. Getting the partitioning of this work right, together with the protocols used among the system's various hardware
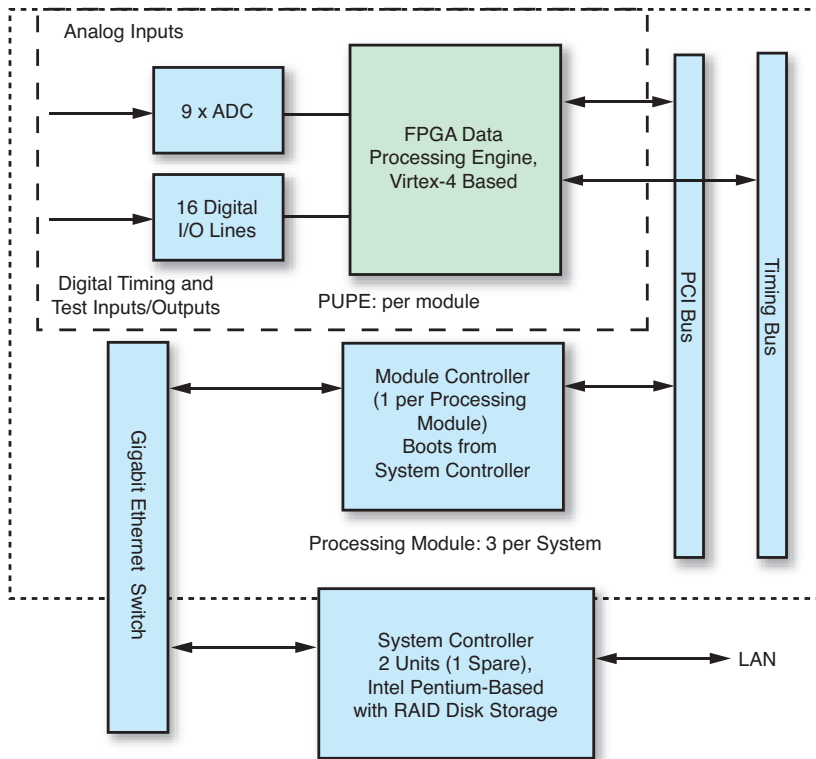
Figure 2 – The Trajectory Measurement System

and software modules, is essential in projects of this kind.

Communication between the TMS modules uses Gigabit Ethernet for both external links and internal communications between the system controller and module controllers. The Beam Object Access Protocol (BOAP) is a simple and efficient protocol for this work. The BOAP system employs quality-of-service protocols to tighten message delivery times. We implemented communications with the PUPE FPGA cards as a register and shared memory interface over a CompactPCI bus.

Along with the 120 analog channels, the CERN PS machine produces a number of digital system-synchronization signals. The main one is a systemwide 10-MHz clock that is used to synchronize the complete system. A digital timing bus links these signals to each of the PUPE boards.

PUPE FPGA Board Design

To reduce project costs and development time, we decided to use commercial off-

the-shelf components wherever possible. For the Pickup Processing Engine, we initially considered using off-the-shelf FPGA modules and A/D converter modules that Alpha Data produces. However, in this case, we decided instead to design and produce a custom CompactPCI board based on an existing PMC design. The size and performance of FPGAs had reached the

stage where one could easily support nine ADC channels with the appropriate data-processing algorithms. Sharing the FPGA and associated communications hardware among nine ADC channels allowed us to reduce the system cost considerably, trim the system's physical size, more tightly couple the ADC clocking system and support extra features. Alpha Data's experience in producing Xilinx FPGA boards allowed us to produce the new PUPE board within seven months, from conception to first working boards.

These PUPE CompactPCI cards (designated ACP-FX-N2/125) are the heart of the Trajectory Measurement System (Figure 3). They utilize a Xilinx® Virtex®-4 FX100 FPGA, 1 Gbyte of DDR2 SDRAM and nine 125-MHz, 14-bit ADCs. The board's core design is based on Alpha Data's ADM-XRC/FX100-10/1G FPGA PMC module, providing a high degree of FPGA firmware compatibility with this hardware. The design employs a low-jitter, phase-locked loop (PLL) synchronized clock source for the ADCs. One of the design issues was to distribute this clock to all nine ADCs without increasing the clock jitter appreciably. The board's PCB layout was also crucial to achieving higher performance from the A/D converters as well as low on-board noise from the board's components.

The board employs a second Xilinx Virtex-4 LX25 device for CompactPCI interface duties. This uses the PCI bus'
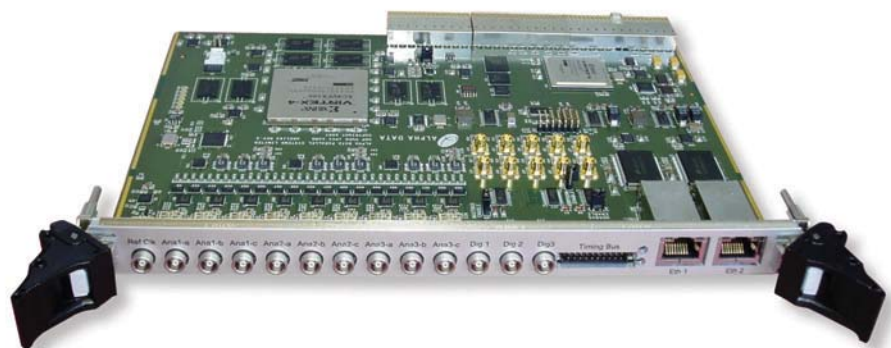

Figure 3 – Alpha Data's Pickup Processing Engine (PUPE) board
is based on Xilinx FPGAs.

The core module is the particle beam synchronized phase-locked loop. The FPGA allowed us to implement this real-time structure together with the DSP processing algorithms side-by-side in the same chip rather than in dedicated hardware.

FPGA firmware as developed by Alpha Data for its existing PMC boards. The PUPE also has two Gigabit Ethernet PHYs with the associated RJ45 connectors on the front panel, connected directly to the FPGA. The large number of I/O pins available on the FX100 allowed us to connect all of the ADCs and all other on-board components directly to the FPGA with minimal glue logic. We use quick-switch devices for interfacing to external 5-volt digital signals.

CERN is a primarily a research institution. So throughout the TMS design process, we tried to add features that could be useful in the future as needs changed. The FX100, for example, could be swapped for an FX140 if a CERN project required more hardware resources. Since five PUPE boards share the CompactPCI bus, there is a communications limit of about 100 Mbytes/s (20 Mbytes/s per board). The Gigabit Ethernet channels can provide up to 500 Mbytes/s, per five PUPE modules, for future needs. The PUPE is thus capable of employing the internal PowerPC processors with the Gigabit Ethernet interfaces to run a local Linux control process if required.

Design and development of the PUPE progressed relatively smoothly, with only a few minor issues to sort out before the first production board run. The on-board JTAG chain and ChipScope™ interface helped enormously with initial board debugging, allowing us to configure the FPGA fabric for test purposes (Figure 4).

PUPE Firmware

The PUPE FPGA firmware is written in VHDL. We used the ISE® 9.2 tool set for development purposes. It was helpful using a readily available development package; CERN was using this same tool set, so we

could exchange FPGA firmware designs relatively easily without the necessary changes for different vendors' tools. We also used Alpha Data's standard SDK to simplify development and deployment.

The design makes use of a number of key core modules that we had developed for other projects. The use of these standard, proven cores helped us to produce the FPGA firmware within the project time

scale with relative ease. The most difficult and complex block is the SDRAM access block. This has a time-shared, dual-port access scheme so that the host computer can read the acquired data while it is being captured in real time. This block caused us

the most issues during development, mainly due to the tight timing requirements, and thus increased code compilation times. Thankfully, our modular approach helped out here, allowing us to develop and test the firmware using just one of the three pickup channels. This reduced compilation times significantly, with the lower gate count used, and thus improved the rate of firmware development.

Figure 4 – The PUPE Board Structure
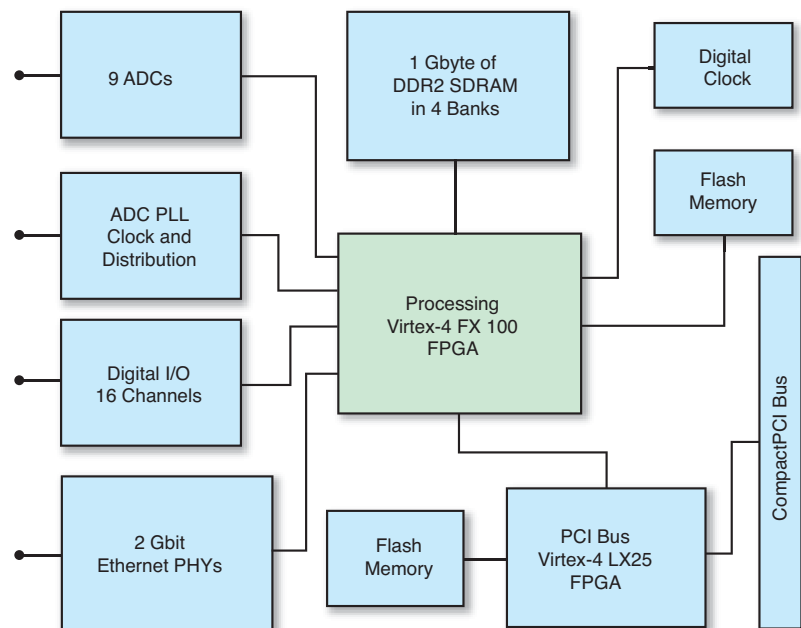
We implemented the interface to the host computer as a register plus shared memory interface. The firmware implements a set of state machines that are software programmable to carry out most of the data-capture and data-processing func-

tions with minimal host software interaction. The FPGA hardware can thus be programmed to implement the appropriate real-time data capture and data processing for the PS machine's cycle.

The core module is the particle beam synchronized phase-locked loop. This structure allows the processing algorithms to synchronize to the incoming analog beam signals. Implementing this real-time structure would have probably required dedicated hardware. The FPGA allowed us to implement it together with the DSP processing algorithms side-by-side in the same chip.

The firmware design called for a number of clock domains. The ADC clock, the PCI bus clock, the external system clock as well as the lower-level clocks were all needed. The Virtex-4's DLLs and clock structure handle the requirements admirably.

We developed the firmware in parallel with the hardware and system software, using Alpha Data's FPGA boards as a test and development platform prior to having the real hardware available. This helped to tighten up the project development cycle and ensured that any prospective issues emerged at an early stage.

The FPGA utilization was about 75 percent. During development we were conscious of and targeted the power levels the system would use. One thing we noticed, during development, was the need to turn off the "keep alive" for the Multi-Gigabit Transceivers. This saved 7.5 watts of power usage by each FPGA.

TMS System and Software

We used a reduced but standard Fedora Core 6 Linux distribution for the system controller and our own simplified Busybox-based Linux system for each of the three module controllers. The open-source nature of Linux, as well as its stability, is a real boon when used in scientific research. The real-time software we developed provides control, data postprocessing and access, system monitoring and test.

The FPGA's firmware bit file resides on the system controller and is downloaded to the FPGAs on initialization. This method

makes it easy to test out and use different processing algorithms within the FPGAs.

We built test blocks into the FPGA firmware and system software. This allowed testing of the system during development when there were no ADCs available, and later when system testing. (It is difficult to find an old Proton Synchrotron lying around for test purposes!) Again, the FPGA helped enormously here. We could easily add test hooks and module emulation hardware within the FPGA that could be switched in and out under system control or just included during development. In fact, the PUPE firmware has a complete data and logic analyzer built in to aid with diagnostics in the running system.

FPGAs in Use

In scientific research projects like this one, FPGAs really do excel. The nature of scientific work entails a degree of algorithm development as ideas and experiments change. The use of an FPGA makes it easy to tightly integrate parallel DSP functions together with digital PLL, logic state machines and other structures into one chip. The programmable nature of the FPGA allows the engineers to modify the fast real-time data-processing algorithms as required to meet the needs of the experiments. In our case, it also made the overall TMS and PUPE board itself quite flexible and suitable for many other application areas.

The resulting power usage of the complete system is around 700 W in full use (one system controller, three Linux module controllers and 15 PUPE boards). Each PUPE board takes about 35 W—a very reasonable figure considering the processing work involved.

The TMS is now in use at CERN. It is providing more detailed and complete information on the trajectory of the particle beams, giving CERN scientists more valuable data in their never-ending search to discover the workings of the universe we live in.

More information on this system is available at *http://portal.beam.ltd.uk/ support/cern.* Alpha Data's FPGA product range is detailed at *http://www. alpha-data.com/product_comp.php.*

Hardware-Centric Approach Builds More Reliable Medical Devices

Improved design methods combined with Xilinx FPGAs reduce time-to-market and enhance quality.

by Chuck Russo
General Manager
HEI Inc.
chuck.russo@heii.com

P.J. Tanzillo
Medical Product Marketing Manager
National Instruments
pjtanzillo@ni.com

Greg Crouch
Embedded Business Development Manager
National Instruments
greg.crouch@ni.com

Just about everyone who owns a mobile phone has experienced a dropped call at one time or another. While system failures or glitches in these and other consumer products are inconvenient, they don't have catastrophic consequences. But a single system failure or glitch in medical electronics can literally prove fatal—one reason why medical equipment, the devices contained in these systems and the software running on those devices must pass rigorous testing and conform to stringent requirements imposed by the Food and Drug Administration (FDA). While the task of bringing these sophisticated devices to market may seem daunting, the rewards of creating a new system that will improve the quality and even length of life for some patients are what motivate our team at HEI Inc.
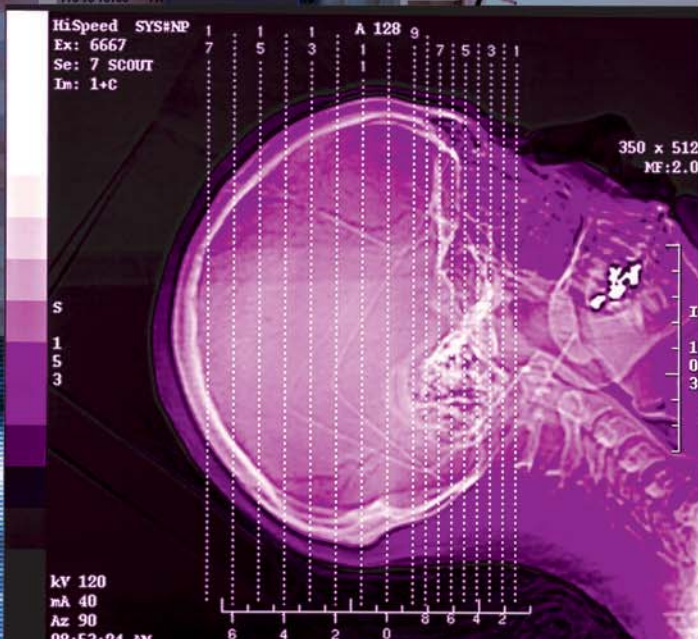
To ensure that our new designs function optimally and reliably, and stream through the FDA's approval process with ease, we employ a highly structured design methodology we call the "scrum/sprint development process." We also reduce any possibilities of software errors by simply lowering the amount of functionality we implement in software. Instead, we implement functionality in Xilinx® FPGAs.

As a bit of background, HEI is a contract medical-device design company that supports the entire life cycle of our customers' products, from early business development support through design and development, new-product introduction and prototyping, initial production ramp, volume manufacturing and end-of-life manufacturing. To better understand our methodology, let's first examine the medical-device design process.

Three-Phase Life Cycle

The FDA has some strict regulations, requirements and guidelines for medical electronics, with the stated purpose of ensuring the safety of the general public. Within these regulations, the FDA has stringent requirements for the life cycle of a medical device (Figure 1). Generally, electronics companies must apply these regulations to any component, part or accessory of a medical device; any software used in the production of a device; and any software used in the implementation of the device manufacturer's quality system, such as a program that records and maintains the device history record.

One can divide the medical-device life cycle into three main phases. The first is the early product life cycle (see Figure 2). The
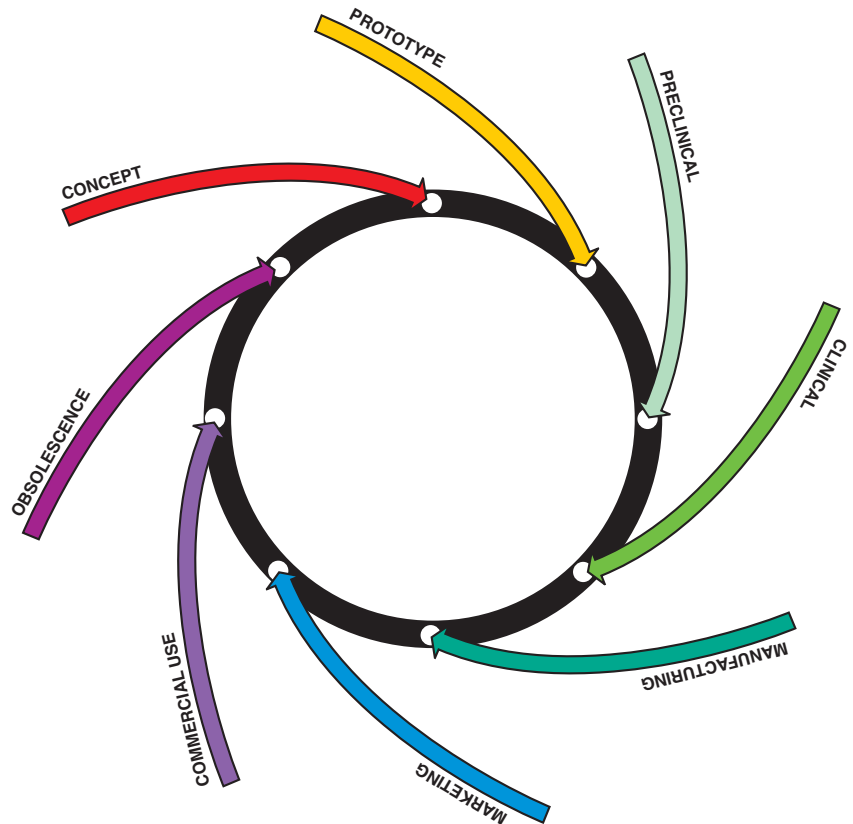
Figure 1 – Diagram of the medical-device design life cycle as defined by the FDA

least regimented of the phases, this is where companies primarily focus on the research and development of theories and ideas. This phase can last from a few weeks to many years, depending on the complexity of the system the company is trying to develop.

A fundamental part of the early product life cycle phase is data collection and analysis. Researchers and product design specification teams typically use many tools to help streamline this process (see sidebar). At this stage, HEI will often use National Instruments' LabVIEW product

to sort out FPGA I/O. Once we understand the problem, we can design a solution. For device development and prototyping, we reuse the math and signal-processing capabilities along with intuitive graphical programming to develop new algorithms. Then, using commercial off-the-shelf hardware, we verify the algorithm's performance against real-world data. In many cases we use NI's Xilinx FPGA-based prototyping platforms for experimental prototypes of the final device. Specifically, we use the LabVIEW Real-
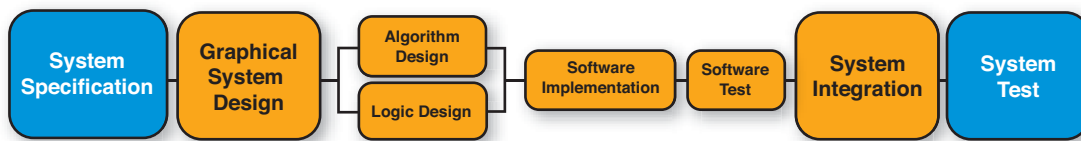
Figure 2 – Applying virtual instrumentation and the HEI design method early in the product life cycle gives a clear understanding of the problem to be solved. Graphical system design helps us develop functional prototypes in record time. The prototyping platform incorporates the same components as the final deployed system, reducing overall development time.

Time and FPGA modules, along with the NI CompactRIO to quickly iterate between the algorithm design and the device prototype stage. Using off-the-shelf hardware for prototyping shrinks the time-consuming step of hardware development and integration, and gives us more time to focus on delivering a bulletproof software design.

The second phase of the medical-device life cycle can be called the middle product life cycle (see Figure 3), addressing the productization, verification, validation and manufacturing of the designed device. The focus in this phase is to develop well-defined specification documents that have

clear and measurable requirements. Once these specifications are defined, it is time to develop a clear mapping between requirement documentation and actual implementation code.

In today's complex medical-device market, customers must get to market as quickly as possible. Many companies try to do this using the traditional "waterfall" development methodology, in which design groups attempt to complete each stage of the design process before moving on to the next step (Figure 4). The waterfall methodology is highly dependent on having complete and accurate specifications at the

beginning of the project. However, in the medical-device market, more times than not needs evolve with the development of the product. What's required is a process that takes this evolution into account. HEI's scrum/sprint development process is the answer to this problem.

In the scrum/sprint process, we require only a high-level system architecture and specifications to start a project. We divide the project into "sprints" of four to six weeks in duration. Within each sprint, we identify all tasks we think the process will require and place them on a "burn-down" list. At the beginning of each sprint, we allocate tasks to the team members in a planning session. The team meets daily for a brief stand-up meeting called a "scrum," in which each team member answers the following three questions: "what did I complete yesterday," "what will I complete today" and "what obstacles are in my way?" The project manager, or "scrum master," manages the burn-down list to track progress on a daily basis.

Figures 5 and 6 show schematics of the process. HEI's companywide use of the scrum/sprint development process has reduced our development times by 30 percent, allowing us to implement new products months ahead of time.

Of course, medical-device product development is nothing if it does not comply with FDA and other regulatory requirements. At HEI, we have a track record of regulatory compliance going back 25 years. In fact, the FDA has audited our scrum/sprint product development process and found it to comply with all elements of the agency's Quality System Regulation (QSR). Table 1 summarizes the waterfall and scrum/sprint development approaches.

The third and final phase in the development of a medical device is the late product life cycle (Figure 7). Very little engineering work is necessary in this phase, but customer feedback and market successes help drive concept development of the next-generation product, when the cycle starts anew.

HEI is able to quickly iterate on future product derivatives, thanks to the scrum/sprint product development process
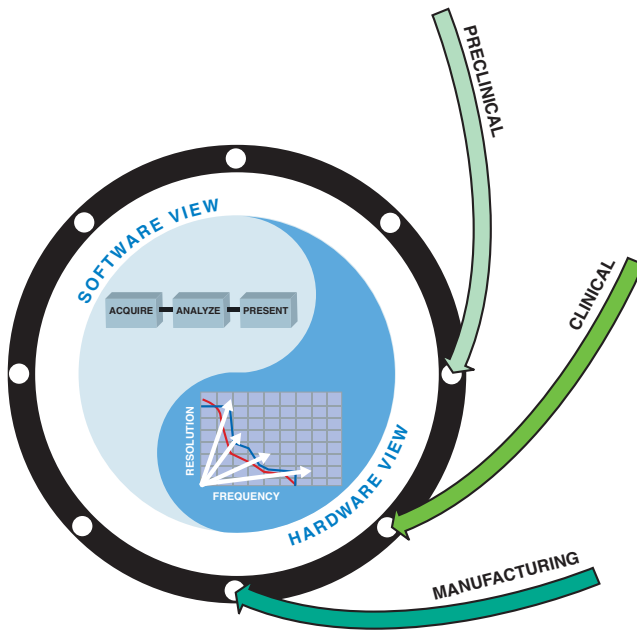
Figure 3 – Software designs are typically validated and verified during the middle of the product life cycle.
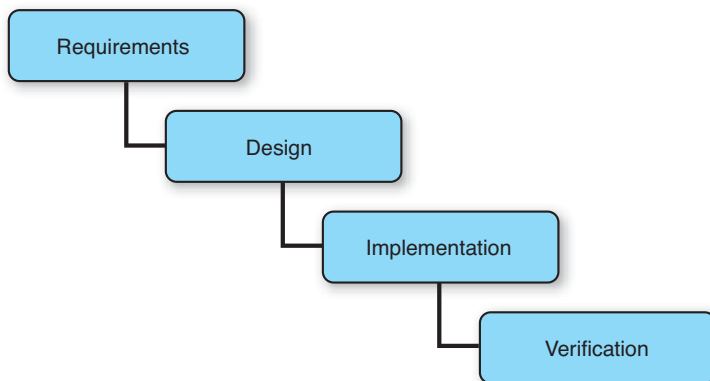
Figure 4 – Traditional "waterfall" development process attempts to complete each stage before moving on to the next.

	Scrum/Sprint	Waterfall
QSR compliant	Yes	Yes
Speed in execution	Yes	No
Reduces program risk	Yes	No
Early verification of requirements	Yes	No
Easy to implement change	Yes	No
Able to proceed without complete requirements	Yes	No
Short development cycles	Yes	No
Overall time-to-market	**Much reduced**	**Typical**

Table 1 – The scrum/sprint methodology outperforms the waterfall approach to development.

and our use of off-the-shelf FPGA-based hardware as well as high-level FPGA software design tools that scale from research to manufacturing. In many cases, we find that we can use the generic core architectures we develop in multiple products. For example, the same architecture for a pump controller that regulates an IV and medication flow pump can be used in another design project that controls a motor for administering blood transfusions.

Why Hardware Trumps Software

To employ this methodology effectively and further speed our design process, we had to change the way we thought about designs, moving from a software-centric to a more hardware-centric approach. As many are aware, medical-device recalls reached an all-time high in 2008—up 43 percent from 2007. Key causes, according to the experts at the FDA, involve two primary issues: defects in manufactured raw materials and poorly developed software. Companies can monitor the quality of their raw materials fairly easily, but it can be very tricky to address software quality. As the lines of code in devices continue to mushroom, the problem will only worsen—a particular concern since the FDA's consumer safety branch says much of the burden of safety is now the responsibility of the medical-device designer.

At HEI, we feel there is a potential solution to this problem, but it's not in more testing, code reviews and process. Instead, we simply try to write less software and

push more of the logic into hardware elements like Xilinx FPGAs. Let's look at some of the common causes of software failure and how we are addressing these issues with FPGAs.

Eliminating Deadlock

Most modern devices need to be able to handle multiple tasks simultaneously, yet most embedded processors are still limited to one processing core. This means the processor can execute only one instruction at a time. Meanwhile, parallel processes

aren't much better, as they must still share the main CPU. In addition, other shared resources such as communication drivers, hard disk and user interface elements present opportunities for deadlock—the condition that occurs when two or more processes are waiting for one another to release a resource.

Deadlock can be very difficult to reproduce and debug, because the situation often relies on multiple processes and usually requires a specific and synchronized sequence of events to occur. Unit testing alone will not catch most deadlock issues; they are usually uncovered by code reviews, adept system testers or simply by luck.

With FPGAs, by contrast, "processes" that are independent have their own physical circuitry and therefore, there are no shared resources. On each clock tick, combinatorial logic latches in each circuit and stores values in separate registers. No deadlocking can occur because neither process relies on the other's resources. This allows you to put much more faith in the results of simulation and unit testing, since other unknowns like resource contention are no longer an issue.
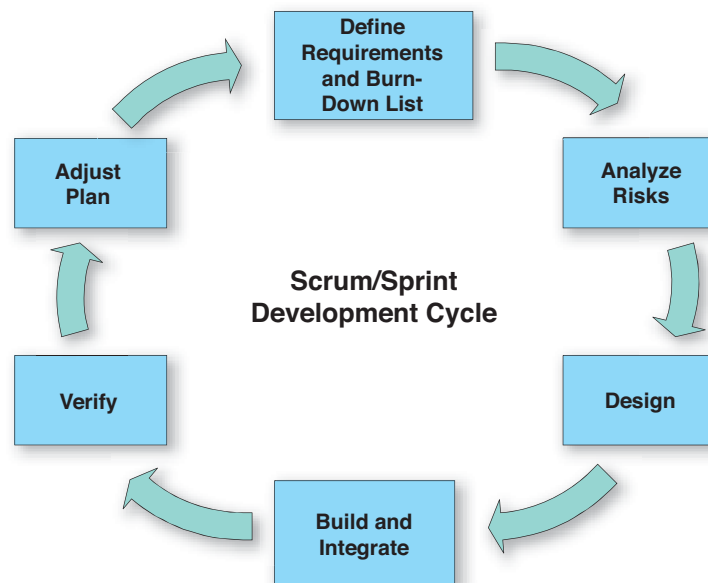
Figure 5 – In the "scrum/sprint" process, only high-level system architecture and specifications are required to start the project. Development is divided into "sprints" four to six weeks in duration. Each sprint follows a six-step development cycle, as shown.

Source: Adapted from *Agile Software Development with Scrum* by Ken Schwaber and Mike Beedle
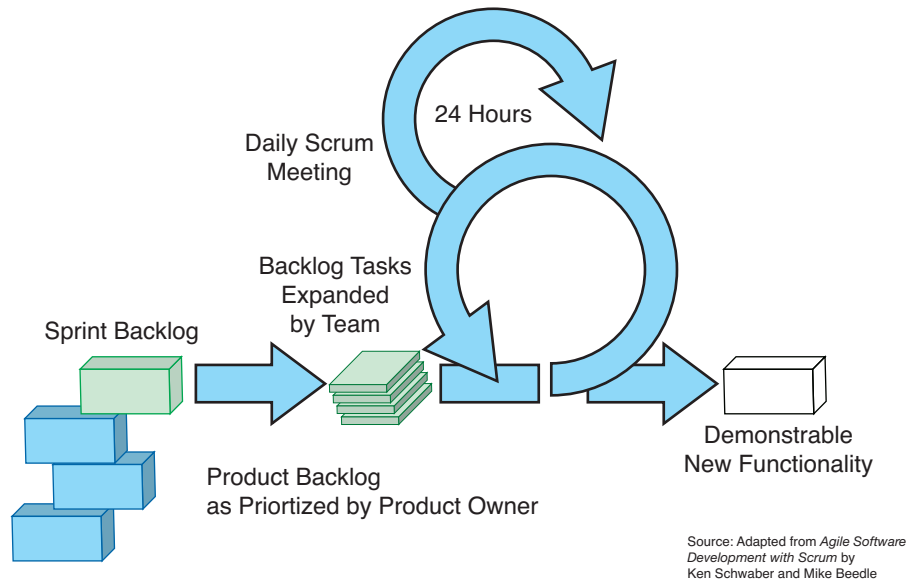
Figure 6 – The team identifies a "sprint backlog" of work tasks for each sprint in the cycle, expands and assigns them, and then reviews progress and removes roadblocks during a daily "scrum" meeting. The team delivers product functionality to the customer at the end of each sprint.

Incompatible Middleware

When developing embedded software, you almost never implement every line of code from scratch. Instead, various tools are available to make the firmware designer more productive. These range from simple drivers to network stacks to operating systems and even code-generation tools. Though these systems are generally well tested individually, no software is bug-free. With so many possible combinations of tools and libraries, the likelihood of using components together in a novel way is relatively high.

For this reason, the FDA mandates that for all off-the-shelf software used in medical devices, companies must validate that the software stack works for each of their particular design's specific use case. What does that mean? If, for example, we are using a signal-processing library that contains a fixed-point fast Fourier transform and we are detecting the presence of a certain frequency component, we do not need to validate that the FFT returns the correct answer for all possible inputs. Rather, we need to validate that it returns what we expect for all valid inputs according to specifications. For example, if we are detecting only tones audible to humans, there is no reason to test that the function returns correct values for inputs over 20 kHz.

Unfortunately, software components that seem independent are not necessarily so. Therefore, if we are using that software stack with an SPI driver coupled with a real-time operating system (RTOS), we need to validate all of these components together to really have confidence in the FFT. If the FFT passes a valid output to the SPI driver, but the SPI driver crashes, then clearly there is a problem. If we then decide to modify the SPI driver, we need to validate the entire software stack again. This

Design Tools Must be Reliable, Too

An important part of HEI's design methodology is our tool set. Because the FDA regulations are so stringent for medical devices, companies that develop medical products are wise to use tools that are mature (not buggy themselves) and have a reputation for reliability. For example, among the many reliable tools we use in our flow is National Instruments' LabVIEW Real-Time and LabVIEW FPGA. We use this high-level programming tool to efficiently define a solution to a given problem. Specifically, we apply LabVIEW FPGA to our design flow process to develop IP on accelerated time lines, with fewer development translation errors and no software rework. LabVIEW's multicore characteristic helps in deploying code to multiple processors.

We also leverage proven hardware building blocks such as NI's Single-Board RIO hardware, which shares both a real-time processor and a Xilinx FPGA. Because so many companies have used the RIO hardware and the underlying middleware drivers in a broad range of applications, we consider them very reliable. This gives us peace of mind and allows us to focus on moving as much software as possible out of the processor and into the FPGA. It also allows us to concentrate on designing the custom circuitry that a particular product requires.

Other off-the-shelf tools our design team relies on include IBM Requisite Pro for requirements management, the Tigris SubVersion configuration-management software for documents and source code tracking, the Seapine Test Track Pro for defect tracking, SolidWorks for mechanical modeling and SolidWorks PDMWorks for mechanical-file management. For PCB electronics design, we use the Mentor PADS Suite. – *Chuck Russo*

Figure 7 – The late product life cycle takes the product to market, where feedback helps determine the features for the next generation of the device. This completes the cycle and brings it back around to the concept phase.

can become very cumbersome, and one faulty component can delay the entire system development. For this reason, at HEI we reuse as much known-good and proven middleware and RTOS driver combinations as possible. We use, for example, the middleware drivers that are a part of NI's Single-Board RIO platforms.

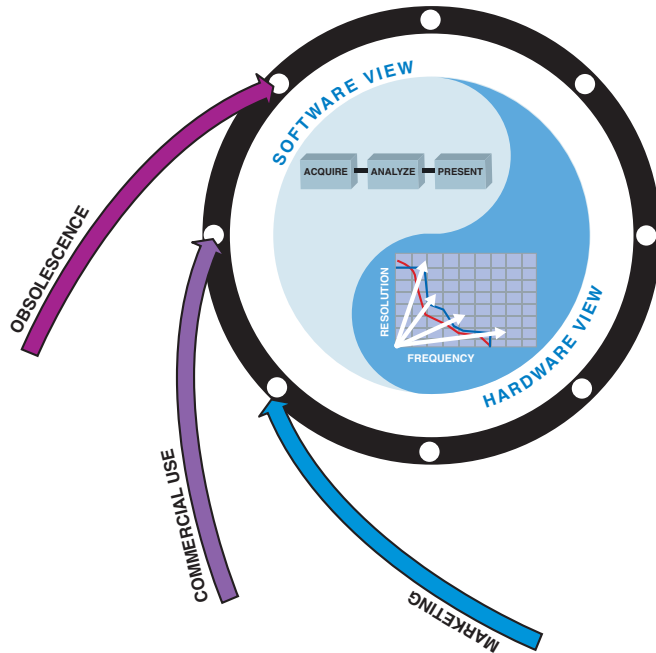In addition to validating software in each use case, we also need to validate all the third-party intellectual property (IP) we are using in our FPGA-based design. However, once we have validated all of our use cases, we have confidence that the IP will behave as expected when integrated with other components. Let's look at our FFT example again. If we use an FPGA, we can acquire or generate an FFT IP core and validate its numerical correctness for each use case—the same as with the software. However, the risk of intermittent failure decreases drastically because there is no need for all the processor middleware we would have in a software-centric design. As such, there is no longer an RTOS, and the SPI driver is its own IP core—its operation does not directly affect the FFT. Furthermore, if we modify the SPI driver

implementation, we don't need to revalidate the unaffected areas of the system.

Managing Buffer Overflow

Another example of how we use FPGAs to mitigate errors that typically occur in software-centric systems is in buffer overflow management. Buffer overflow occurs when a program tries to store data past the end of the memory that you've allocated for that storage, and it ends up overwriting some adjacent data that it shouldn't. This can be a really nasty bug to diagnose, since the memory that was overwritten could be accessed at any time in the future, and it may or may not cause obvious errors. One of the more common buffer overflows in embedded design is a result of high-speed communications of some sort, perhaps from a network, disk or A/D converter. When communications are interrupted for too long, buffers can overflow, so we need to account for this to avoid crashes.

We use FPGAs to manage buffer overflow in two ways. In one example, we use the FPGA to manage a circular or double-buffered transfer, where it can offload the burden from a real-time processor. In this

case, the FPGA serves as a coprocessor that reduces the interrupt load on the main processor. This is a common configuration, especially in high-speed A/D converters.

In a second example, we use the FPGA as a safety layer of protection, routing all of the patient-facing I/O through the FPGA before it gets to the processor. In this case, you can add extra safety logic to the FPGA so that you can place all your outputs in a known and safe state in the event of a software crash on the processor. The FPGA serves as a watchdog and its logic ensures that risk to the patient is lowered in the event of a software failure. By making the architectural decision to use an FPGA in your device's primary signal chain, you can use one or both of these two methods to guard against buffer overflow and to better handle the situation if and when it does occur.

In fact, the more overall system functionality—software as well as hardware—we move into the FPGA, the faster we can expedite our design process and ultimately validate our design running in the customer's end product. The sooner we can validate the reliability of our design running in the overall system, the sooner our customer can validate the entire end product and get it to the FDA for approvals. This of course means our clients can then get their products to the market faster, and in doing so improve quality of life or even save lives.

If we were to implement a design using an ASIC process, we would have to wait many months for a foundry to create the hardware. The extra steps of having to verify the ASIC's physical design, create masks and then manufacture the design create more room to introduce errors and defects. If a design does pick up an error in any of these steps, the result can be significant delays in getting products to market in a timely manner. Because FPGAs are already fabricated and thoroughly tested, we only have to worry about our design, our software and ensuring that our design runs to the customer's specifications. With our scrum-and-sprint methodology, our hardware-centric mind-set, use of reliable tools and choice of FPGAs over ASICs, we can make a difference for our customers and, in turn, their customers—the patients.

Implementing Wireless LAN Interface in an FPGA

Complex, high-speed IP such as WLAN can run at full speed and even pass Wi-Fi certification when built on the platform of a Xilinx device.

by Agnès Faïn
Senior Design Engineer
Wipro-NewLogic
agnes.fain@wipro.com

Wolfgang Meryk
Senior Product Marketing Manager
Wipro-NewLogic
wolfgang.meryk@wipro.com

Every company designing a complex and high-speed intellectual-property (IP) core like an IEEE 802.11 wireless LAN controller faces the challenge of how to prove the design is working and is of high quality. A test chip is important, since only it can show that the IP is silicon-proven. But a test chip can capture the hardware design state at only one point in time. However, IP needs to be flexible to support a variety of configurations and applications. WLAN brings an additional requirement for flexibility and upgradability, since there are still new additions to market-relevant specifications (Wi-Fi, ETSI/FCC) coming, and some of them require or at least benefit from hardware changes.

At Wipro-NewLogic, we decided to investigate an FPGA-based platform to solve this problem. We called the resulting product the WiLDSYS board, to reflect the name of our WLAN IP: WiLD a/b/g.

We formulated several key requirements for this design. First, it had to support running the WLAN IP at full speed (54-Mbit/second air data rate for 802.11a and 802.11g). Next, we needed a single database for targeting both FPGA and ASIC. Third, we wanted a very high level of confidence that test results and bug fixes from the FPGA target could be applied to the ASIC target. Finally, we needed to implement the external interfaces to the host and to the radio all on the FPGA, so as to minimize the number of external components and to leverage the programmability of the FPGA to allow alternative interfaces in the future. During the course of the project we found that the FPGA platform not only met all these goals, but even allowed us to take the WiLDSYS board to a full Wi-Fi certification.

But let's start at the beginning, with an overview of the WiLDSYS design and the selection of the FPGA device.

WiLDSYS Overview

Figure 1 shows the block diagram of the WiLDSYS platform as we implemented it in the original design. The WiLD Core to the left comprises the 802.11 a/b/g media-access controller (MAC) and modem hardware. The MAC functions are split over the WiLD Burst Processor, a buffer management unit/DMA engine and the WiLD Stream Processor, which takes care of the RC4 and AES encryption. Both blocks serve as master on the central Advanced High-Performance Bus (AHB) and are not timing-critical. The WiLD modem contains the CCK/DSSS and OFDM signal processing for the 802.11 a/b/g standard. Signal processing designed for an ASIC is not easy to map to an FPGA, and we will describe the challenges and solutions in more detail later.

There is also a complete ARM7-based processor platform that runs the Layer 2 MAC software. We support both access point and station mode in the software.
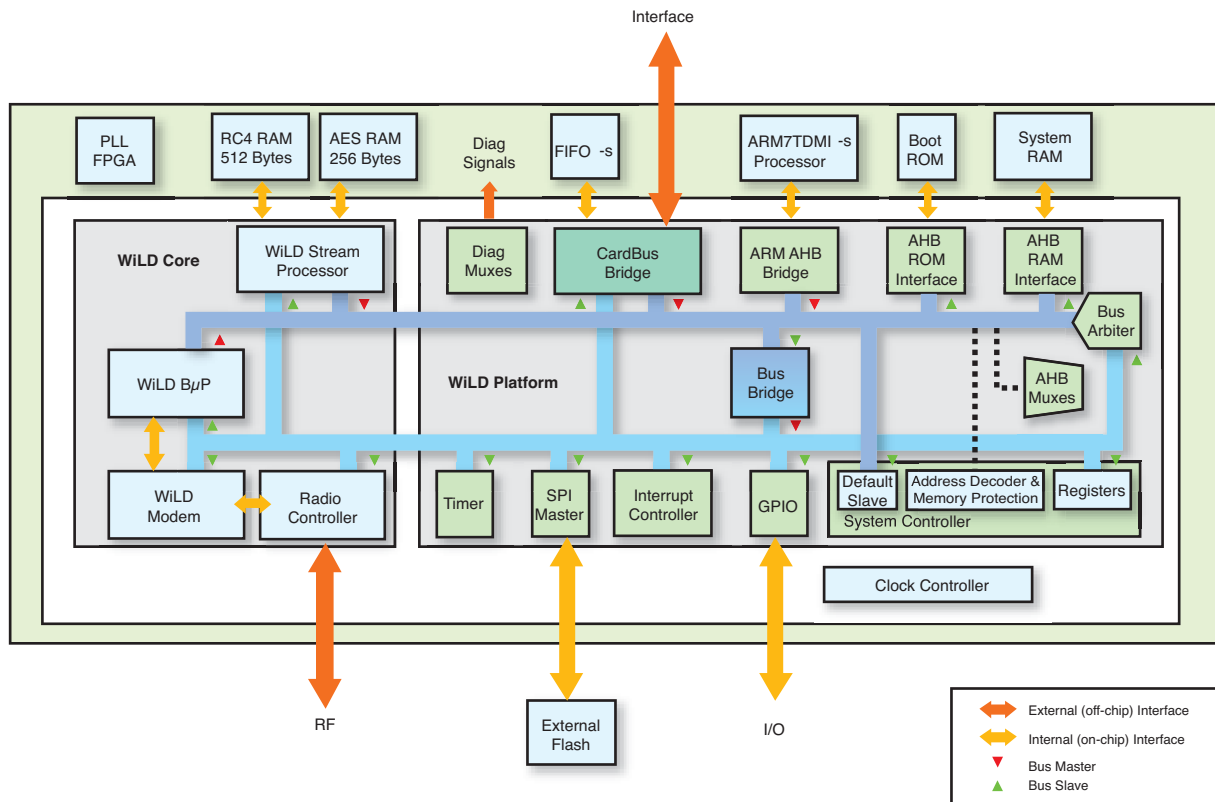
Figure 1 –Block diagram of the WiLDSYS design

Placing an external ARM7 chip on the FPGA board is a help for customers who want to use the board but are not ARM licensees. We also added external SRAM for applications that exceed the amount of memory we have inside the FPGA, as well as a flash device for booting. We deployed an ample number of header connectors to access the FPGA I/O pins of all external interfaces and for debug. For future upgradability, we added a footprint for a second FPGA, though we have not had cause to use it so far. Figure 2 is a photo of the WiLDSYS FPGA board.

FPGA Choice

For this design, we used a Xilinx® Virtex®-4 FPGA. We selected this device family for its high speed and many specialized resources, including DSP blocks, internal memories, flexible I/O pads and high-performance clocking support. We went for the Virtex-4 LX200 FPGA for its size, since the WiLD IP including the platform was already in the area of 700k gates (equivalent to NAND2). Table 1 shows a utilization report of our design.

The Virtex-4 FPGA supports a clock speed of up to 500 MHz. We needed that speed to implement our custom High-Speed Serial Interface toward the Wipro-Newlogic WLAN radio chip, which runs at 240 MHz.

Since one of our goals was to use a single database for both the ASIC and FPGA target, we had to map the ASIC code to the FPGA without optimizing it for the best utilization of FPGA resources. Typically, ASIC code uses longer combinational paths, and we needed margin between the FPGA's maximum speed and our design's highest clock frequency. Xilinx has greatly improved the signal-processing blocks of the Virtex-4 FPGA, especially the multipliers. Dedicated blocks called DSP48 are optimized for multiplication and addition. We took advantage of the DSP48 resources to run the 802.11 a/g modem at 80 MHz in the FPGA.

We were also motivated by the availability of global clock lines. The Virtex-4 LX200 FPGA features 32 global clock buffers, with eight of them not constrained to a specific FPGA region. Since it's possible to place flip-flops linked to one of these eight clock

buffers anywhere in the FPGA, we obtained more options for mapping, placing and routing our design.

Challenges of a Single Database

We knew that a validation done on FPGA would be meaningful only if the results were representative to the ASIC design. The code mapped to the FPGA had to follow any bug fix or new feature added to the ASIC code. In the other direction, any correction done during FPGA validation had to be reported back into the ASIC design and fully simulated using the ASIC's verification environment. It was clear to us from the beginning that we could handle all this in an efficient way only if we maintained a single design database for targeting both FPGA and ASIC. We quickly ran into several challenges in the implementation.

In some cases, the only solution was to modify the ASIC code. We had logic paths in our OFDM modem that did not meet timing at 80 MHz on the Virtex-4 device. So we split the combinational paths by inserting additional flip-flops. This increased the overall latency of the modem,

RAM/ROM Utilization		
Dual-port RAMs (RAM16x1D)	140	
64x1 ROMs (ROM64X1)	12	
256x1 ROMs (ROM256x1)	138	
Number of block RAMs	211 of 336	62%
Logic Utilization		
Number of slice flip-flops	52,943 out of 178,176	29%
Number of four-input LUTs	133,498 out of 178,176	74%
Logic Distribution		
Number of occupied slices	83,217 out of 89,088	93%
Total number of four-input LUTs	137,384 out of 178,176	77%
Number of bonded IOBs	361 out of 960	37%
Number of BUFG/BUFGCTRLs	8 out of 32	25%
Number of FIFO16/RAMB16s	214 out of 336	63%
Number of DSP48s	17 out of 96	17%
Number of DCM_ADVs	1 out of 12	8%

Table 1 – WiLDSYS Virtex-4 LX 200 utilization report

but that was OK, since we still had sufficient margin. We reported the modifications back to the ASIC database after we met timing on the FPGA and ran a full regression test on the ASIC target. After the ASIC synthesis, there was even an area benefit for the ASIC implementation of the modified design, since the relaxed timing allowed the ASIC synthesis tool to use a lower number of buffers and smaller combinational cells.

The clock controller is one of the most critical ASIC blocks, and also one of the most difficult to map on an FPGA. Ultimately we had to create a unique block just for the FPGA implementation. We set up the synthesis scripts for ASIC and for FPGA to select the right block for each target. We used the Virtex-4 digital clock manager (DCM) resources to replace the phase-locked loop that would normally be used in a WLAN ASIC design. The DCM created the required 240-MHz frequency from a 40-MHz or 50-MHz oscillator input, exactly as in the ASIC.

We were targeting the WiLD IP at low-power applications and therefore added support for clock gating to the design,

along with separate power domains. We knew we could not fully implement the two features in the FPGA, but we wanted to include at least the control and switching of the clock and power domains. With the eight-region free global clock buffers of the Virtex-4 LX200 FPGA, we didn't have sufficient resources to cover all the clock frequencies and clock-gating functionalities of the ASIC. We chose to implement in the FPGA just the clock gating required for correct functionality of the system, and not the clock gating aiming only at saving power.

To ease the clock tree, we used the clock-gating conversion feature available with the Synplify Pro FPGA synthesis tool. The synthesizer removes the gating condition from the clock line and places it on the flip-flop enable input or on the data input. We used BUFG and BUFGCE buffers for clock gating so that the output of this logic was still routed using the high-speed, low-skew global clock lines of the Virtex-4 FPGA. A BUFGMUX resource provided glitch-free multiplexing between active mode and low-power clocks.

We implemented several voltage domains in the WiLD design that could be switched on and off at different times. We validated this complex and critical functionality with the Virtex-4, even though the FPGA contains only one power domain. A domain that got powered off must be reset at power-up. On the FPGA, we emulated the power-down state with a reset of the domain. Although this method does not completely validate the power domain connections and must be associated with other ASIC checks and verifications, we were able to spot functional errors in domain interconnections and reset generations, which we might not have detected otherwise with the time-limited ASIC simulations.

The WiLD platform follows the AHB standard, with four bus masters sharing access to peripherals such as memories and the Advanced Peripheral Bus (APB) subsystem. We discovered that the critical part for timing closure was the connection of the address and data lines. We overcame this problem by mapping the CPU and memories inside the FPGA, instead of using the external devices. For the latter we took advantage of the large built-in memory blocks of the Virtex-4 device, drastically reducing our routing delays. The Synplify Pro tool also helped, since it was able to extract timing data from the memory netlist files (EDN), rather than seeing them only as black boxes.

Signal processing designed for an ASIC is not easy to map to an FPGA, because it is optimized for area, with reuse of hardware modules on each clock cycle when the data rate is slower than the clock. FPGA synthesis goes around this constraint using retiming and logic duplication. We used the Xilinx PlanAhead™ floor-planning tool to constrain critical blocks of the OFDM modem to a given FPGA zone in order to reduce the routing delays inside and between these blocks.

External Interfaces

The WiLDSYS FPGA platform is flexible and easy to adapt to changes in the external interfaces, thanks to the Virtex-4's ability to support different interface drive characteristics through its adapted pads.

Figure 2 – A Virtex-4 FPGA sits at the heart of the WiLDSYS board.

We generated all interface signals with the correct electrical characteristics on the FPGA, without use of a single external component. We routed the interface signals to header connectors so that all we needed to do was to design the appropriate mechanical interface adapter.

In the original design we used CardBus as the host interface. We defined the FPGA pads as "PCI 3.3V" and then linked them one-to-one to a CardBus connector. When customers asked us to change the host interface to MII and SDIO, we leveraged the Virtex-4's adapted pads to also drive these interfaces directly. We didn't have to change anything on the WiLDSYS board itself. We simply designed new mechanical connector adapters for MII and for SDIO, and the WiLDSYS board was ready to use in our customers' projects.

The same applied to the RF interface. We first designed the WiLDSYS board to connect to the Wipro-NewLogic radio through our custom High-Speed Serial Interface. It uses differential LVDS drivers and the maximum clock speed is 240 MHz. We were able to implement this very demanding connection completely on the FPGA thanks to the Virtex-4's built-in LVDS pads. We packed a flip-flop inside those pads to match the tight 240-MHz timing constraint on this data path. When we wanted to port our IP to an analog I/Q RF connection, all we had to do was to map the digital I and Q signals to standard pads and to another header connector, where we then plugged in a daughterboard with ADCs and DACs.

FPGA and ASIC Co-Verification

With the single database for both ASIC and FPGA, we were ready to start our FPGA-ASIC co-verification. Having just one hardware database meant we also had to maintain just one version of our WLAN MAC software for both targets. We could set up the FPGA board to either station or access point mode and have it interact with other WLAN nodes that were set up using commercially available equipment. We started the FPGA-based verification by running the complete ASIC system test plan, which covered all functional features, including many of the test cases from the Wi-Fi Alliance. These included tests for bandwidth sharing, encryption and quality of service.

Whenever we detected a problem on the FPGA platform, we first investigated it using FPGA-specific means. The WiLDSYS board's logic analyzer connectors offer a much broader debug interface than the ASIC, which is often limited in pin count. We utilized GPIO signals to map diagnostic signals to these connectors and probed internal FPGA signals with the help of the Xilinx FPGA Editor. The tool opens a graphical view of the routed FPGA design; designers can pick any internal wire and route it to an unused pad. FPGA Editor directly modifies the FPGA bitmap, avoiding the time-consuming steps of mapping and routing the design again.

By probing relevant signals, we were able to pinpoint the source of the problem with enough precision to reproduce it in the ASIC RTL simulation testbench and then correct it. All we had to do now was to run a new FPGA synthesis and create a new bitmap file.

We ran a final verification of the fix on the FPGA board by repeating the test case specific to the bug as well as performing a selected number of regression tests.

We then took advantage of having the WLAN IP run at full air data rate and connecting through a real radio to another WLAN node to add test cases for stress testing, including overnight testing. We acquired the equipment from the Wi-Fi Alliance test bed to cover all the interoperability testing. We achieved significantly higher test coverage and hence robustness of both the WLAN hardware and the software. The highlight of all the testing was when we submitted the FPGA board to an official Wi-Fi lab for certification, and we passed not only the baseline test plan for 11 a/b/g, but also the optional testing for quality of service (WMM), 802.11d and 802.11h.

We received the final proof that our FGPA-ASIC co-verification is yielding a very high confidence for a successful ASIC development when we started to use the WiLDSYS board in customer projects. We ran a complete FPGA-based verification that included extensive system-level and Wi-Fi testing before tapeout and achieved several first-silicon successes.

Ultimately, we had to overcome several design challenges to achieve our goals of running the WLAN IP at full air data rate out of an FPGA and to rely on the FPGA platform for most of the system-level verification. The Xilinx Virtex-4 FPGA and the Xilinx tools made all this possible. In the end we gained with the WiLDSYS board an extremely valuable tool, not only for our internal development program, but also for customer projects.

This positive experience has led us to use the same Virtex-4 board with different daughterboards to develop our Bluetooth 2.1EDR IP. And since there is still demand for 802.11 a/b/g, we are now looking into fitting the IP into a Xilinx Spartan-6, since this new family is offering the needed resources at a much lower cost than the Virtex-4. This opens a new market for us, since running wireless LAN out of an FPGA will now become commercially feasible.

Improving DDR SDRAM Efficiency with a Reordering Controller

Virtex-6 memory controller achieves excellent sustained transfer rates for real-world workloads.
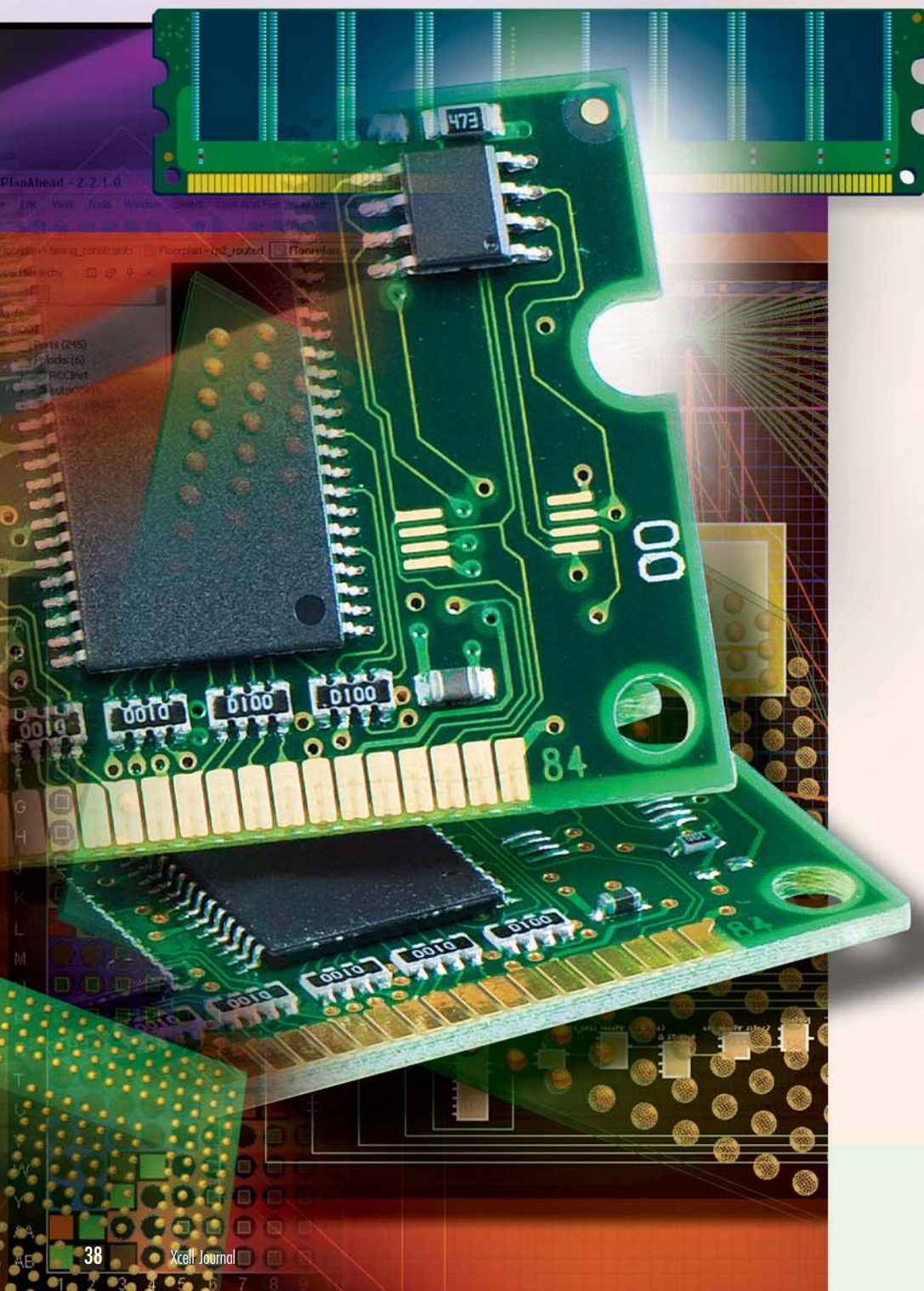
by Leith Johnson
Managing Member
Nokhu Systems LLC
leith@nokhu.com

Vendors of high-speed double-data-rate (DDR) SDRAMs commonly specify their devices by their peak data transfer rate. For example, if a given vendor dubs a device a DDR3-1600, it means the vendor has specified the SDRAM component will transfer data at a peak rate of 1,600 mega-transfers per second.

While the devices can indeed reach their specified transfer rate, in practice they can't sustain it for real-world workloads. That's because row-address conflicts, data-bus turnaround penalties and write recovery all can degrade the device's peak transfer rate.

To make matters worse, the negative impact from these assorted degradations has increased in lockstep with each new generation of faster SDRAM. For this reason, memory controllers employing simple in-order scheduling algorithms will likely achieve sustained throughput that's substantially below the specified peak. However, more-advanced reordering scheduling techniques can overcome these degrading factors, ensuring that your memory will deliver excellent sustained transfer rates for real-world applications.

With the Virtex®-6, Xilinx® has introduced the first reordering DDR SDRAM memory controller optimized for FPGAs. The new controller enables Virtex-6 users to capitalize on the improved capacity, performance and power efficiency of the latest-generation DDR SDRAM technology.

On the surface, a DDR SDRAM component is simply a read-write memory. In reality, however, modern DDR SDRAMs are complex devices. DDR SDRAM controllers must generate very precise sequences of addresses, commands and data while observing a myriad of timing requirements. High performance requires command pipelining at the minimum allowed timings.

Ins and Outs of DDR SDRAM

What is the nature of the degradations that affect memory performance, and why does increasing the peak transfer rate exacerbate their impact?

As illustrated in Figure 1, a basic DDR SDRAM access has the memory controller sending a row address with the activate command to the memory, waiting for the RAS-to-CAS delay interval (defined as the number of clock cycles between the row and column address strobes) and then sending the column address and either a read or a write command. Completing the
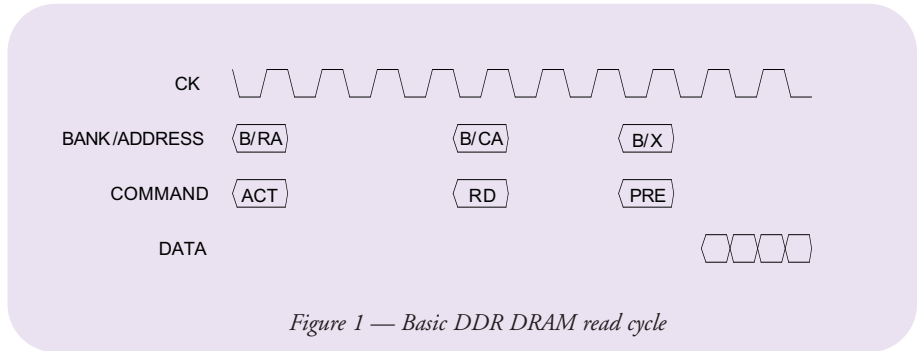
Figure 1 — Basic DDR DRAM read cycle

request requires waiting for the CAS-latency interval and then sampling data for reads or supplying data for writes. When the data transfer is complete, the controller issues a precharge command to close the active row. Following the precharge interval, the controller may issue another activate command.

After that, it may issue multiple read or write commands without a precharge-activate sequence. This is commonly referred to as fast-page-mode access.

Fast page mode is very efficient, since it avoids the time-consuming and power-hungry activate-precharge sequence, but the accesses must be to the same row address. If the workload contains a pattern of accesses to different row addresses, then the activate-precharge sequence must take place between each access. In this case, sustained

throughput will be relatively low, very significantly less than the peak transfer rate.

DDR SDRAMs are "banked," or broken into some number of equal-size, quasi-independent sections. DDR3 DRAMs have eight banks. Accesses to different banks may be overlapped. For example, if the workload contains a read to bank 1 followed immediately by a read to bank 2, the memory controller is allowed to send the row address and activate to bank 1, then the row address and activate to bank 2. After the RAS-to-CAS delay interval, the controller can then send the column address and command for the first read followed by the column address and command for the second read, wait for the CAS latency and then transfer two seamless data bursts. Given a workload that sequentially steps through the banks, this
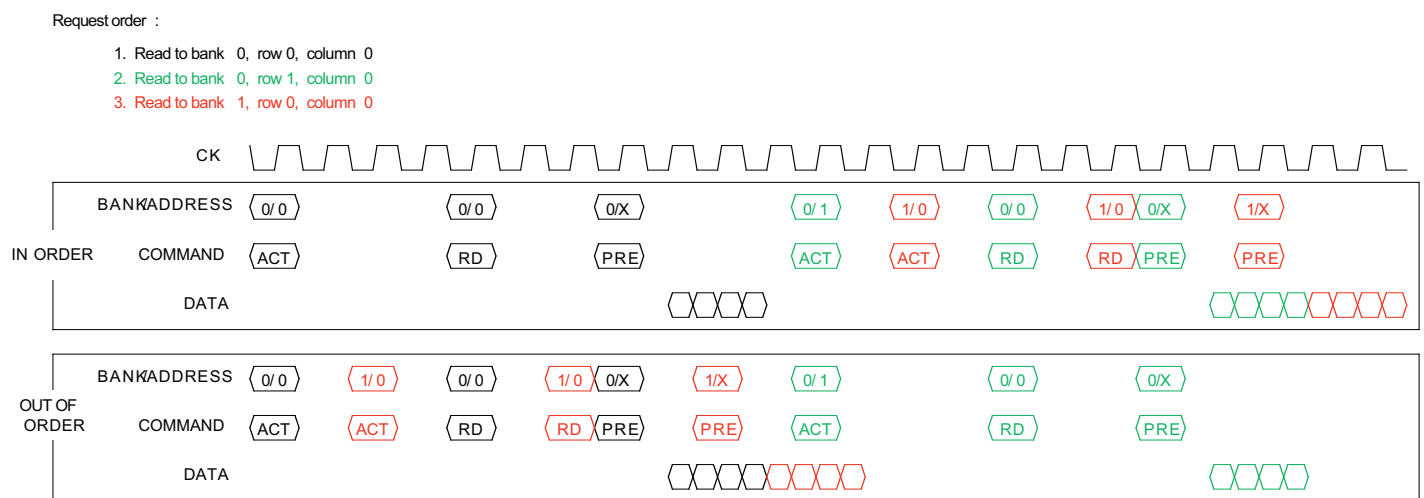
Figure 2 – Reordering to avoid page conflict

In addition to overcoming much of the degradation associated with row-address conflicts, a reordering memory controller can also deal with degradation due to write recovery and bus turnaround.

process may be extended in such a way as to sustain the peak transfer rate.

The best way to achieve good DRAM performance is to manipulate the workload such that it falls into these fast-page or rolling-bank modes. Often, however, this is not possible. Processors typically generate quasi-random workloads that are not easily manipulated in this way.

If the workload contains adjacent accesses to different row addresses on the same bank (row-address conflict), followed by an access to a different bank, a simple in-order memory controller will serialize all three accesses, incurring a substantial efficiency penalty due to the precharge-activate for the second access. In contrast, a reordering memory controller is able to send the activate for the first access and then the activate for the third access, overlapping these accesses and thereby improving efficiency.

Figure 2 illustrates this concept. The pattern is shown executed in order and

then with the second request moved in front of the third. As you can see, the reordered sequence completes before the in-order sequence.

In addition to overcoming much of the degradation associated with row-address conflicts, a reordering memory controller can also address degradation due to write recovery and bus turnaround. At the end of a write cycle, internally the DRAM is busy actually writing the data into the array. While adjacent write accesses can proceed at peak rate, a write access followed by a read access must wait for the write to complete in the DRAM array. This gives rise to the write-recovery specification.

DDR SDRAMs utilize a bidirectional shared-bus structure. This bus has the controller on one end and typically two to four dual in-line memory modules (DIMMs) on the other end. The electrical length of this bus is about 5 inches. This translates to a bus propagation time of approximately 1 nanosecond. Whenever a

bus driver stops driving, it propagates a glitch on the bus. Nominally, two bus propagation times are required for this glitch to settle out.

Built into the DDR SDRAM protocol is a data preamble consisting of one DDR SDRAM clock or two unit intervals. Data is not transferred during the preamble. The preamble can be suppressed if the same driver drives the data for two adjacent accesses. In fact, the preamble must be suppressed if the peak transfer rate is to be achieved.

The performance impact of these write-recovery and data-bus degradations can be significant. A simple in-order memory controller is at the mercy of the workload. Workloads with alternating read-write patterns will exhibit very substantially less than peak transfer rates.

But a reordering memory controller is able to group accesses based on the bus driver. In a simple single-rank configuration, it groups reads and writes together and issues them in bursts. In a multirank

Figure 3 – Reordering to avoid write-recovery and bus-turnaround penalty

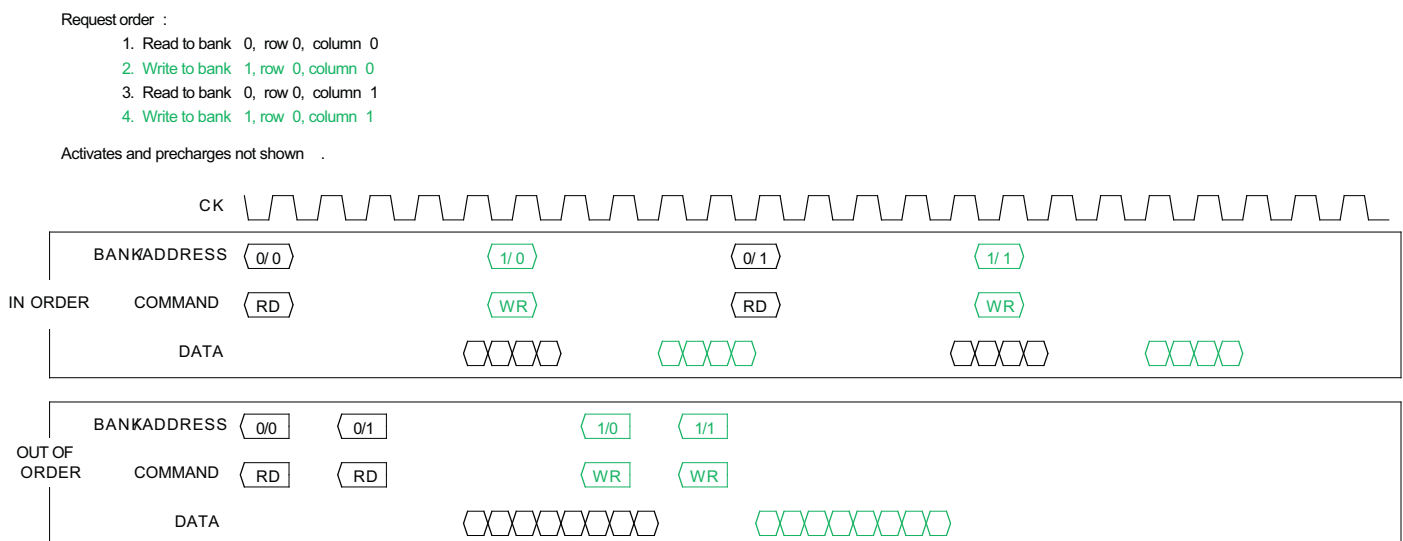|	DDR1-6		DDR2-25		DDR3-15E	
Peak transfer rate (Megatransfers/s)	333		667		1333	
Unit intervals (nanoseconds)	3		1.5		0.75	
	ns	UIs	ns	UIs	ns	UIs
RAS cycle	60	20	55	37	49.5	66
Write recovery	15	5	15	10	15	20
2x tBUS_PROP	2.0	0.67	2.0	1.3	2.0	2.6

Table 1 – DDR SDRAM transfer rates are increasing geometrically; core device characteristics are not.

	In-order	Reordered
CPU-DMA streams	36%	76%
Alternating reads/writes	25%	63%

Table 2 – In two example workloads, memory controller efficiency improves when reordering is turned on.

configuration, it may organize accesses such that reads for each rank are grouped and issued in a burst.

Figure 3 illustrates the impact of request reordering to avoid bus-turn-around and write-recovery penalties. In this example, sequential reads to bank 0 are interleaved with a sequential stream of writes to bank 1. Two DMA machines reading and writing to memory could very easily generate this workload.

The in-order case loses considerable bus bandwidth with each changeover from read to write and write to read. The reordered case can avoid these penalties to attain significantly higher throughput.

Transaction reordering always raises the issues of data corruption and starvation. Promoting a write in front of a read to the same address corrupts the contents of a memory and must be prevented. Endlessly deferring a read access leads to starvation and must also be avoided.

Reordered data returning from the memory controller may be an issue for some applications. It's easy to rectify this problem by fitting the memory controller with a reorder-back-to-issue order buffer.

For some corner cases, reordering may increase latency for some read accesses. However, since reordering improves efficiency and therefore reduces memory controller occupancy, the result is to improve average read latency.

Table 1 illustrates the progression of DDR SDRAM technology. Some interesting specifications are provided for DDR1, DDR2 and DDR3. The speed grades chosen are intended to represent midlife, high-volume production.

The transfer rates are increasing geometrically, while the core device characteristics such as RAS cycle are improving at a much slower rate or, in the case of write recovery, not at all. The relative penalty normalized to unit interval increases dramatically with each generation.

Other DDR SDRAM parameters have similar scaling issues. The above three were chosen for illustrative purposes.

In-Order and Reordering Modes

Let's look at two example workloads that will demonstrate the capability of the new Virtex-6 DDR SDRAM memory controller. Because this device is able to operate in both in-order and reordering modes, the impact of the reordering is readily visible.

The first workload models two masters. One is a CPU executing a code sequence that is stepping through a matrix. The "stride" of this stepping through the matrix generates a memory request pattern that targets a single bank, but increments the row address with each request. This so-called "unfortunate stride" creates a stream of read requests to a single DRAM bank, but different row addresses. The other master is a DMA engine generating a simple sequential stream of read accesses to a different DRAM bank.

The in-order and reordering cases will actually execute a different overall stream of requests. The row cycle time of the DRAM limits the CPU stream completion rate. Meanwhile, the DMA workload can proceed independently. For the in-order case, the DMA stream becomes serialized with the CPU stream. For the reordering case, the DMA stream fills in the otherwise unused DQ bus cycles with useful work. Since the throughput of the CPU stream is limited by the DRAM row cycle time, it won't improve much. However, the DMA stream will execute at a higher rate when reordering is enabled.

The second workload models two DMA engines, one generating a sequential stream of reads, the other a sequential stream of writes. It is assumed that the memory controller receives the read and write requests on an alternating basis. To avoid other conflicts, all reads target bank 0, row 0, and all writes target bank 1, row 0.

We ran each workload with reordering turned off, then reran them with reordering turned on. We used default memory controller parameters, with the exception of turning off refresh, ZQ and other DRAM maintenance functions.

Table 2 summarizes the efficiency, measured over a period of steady-state operation, for the two example workloads. We computed efficiency as the number of unit intervals containing payload divided by the total number of unit intervals in the period. When reordering is turned on, the memory controller's efficiency improves significantly.

The trend of nonscaling DDR SDRAM parameters is expected to intensify as peak data rates continue to increase. Future memory controllers will need to implement continuously improving scheduling algorithms to extract the performance potential of future DDR SDRAM generations.

Splayed Design Makes Pin Fin Heat Sinks Cooler

New configuration brings in more air to keep up with FPGA cooling demands.

by Barry Dagan, P.E.
Chief Technology Officer
Cool Innovations Inc.
barryd@coolinnovations.com

In recent years, the functionality of cutting-edge FPGAs has skyrocketed to levels never thought possible. Unfortunately, the flip side of this progress is an increase in heat dissipation. As a result, designers need more-effective heat sinks to provide sufficient cooling for these integrated circuits.

In response to this need, thermal-management suppliers have introduced a variety of high-performance heat sink designs that provide greater cooling in a given volume. One of the more prominent technologies to debut in recent years is the splayed-pin fin heat sink. Originally designed to cool FPGAs, these devices have several characteristics that make them especially well-suited for use in common FPGA environments.

Better Cooling and Airflow Management

Splayed-pin fin heat sinks consist of an array of round pins embedded in a square or rectangular base. The pins, which serve as the heat sink's fins, are slanted outward, as shown in Figure 1. Due to their unique physical structure, splayed-pin fin heat sinks produce unprecedented cooling in the moderate- and low-airspeed environments for which they were optimized. Copper and aluminum versions have footprints ranging from 0.54 x 0.54 inch to 2.05 x 2.05 inches and may stand less than a half inch to just over an inch high. These dimensions accommodate FPGAs of all sizes.

Splayed-pin fin heat sinks are a derivative of traditional pin fin heat sinks, which feature vertical rather than slanted pins (Figure 2). To grasp the cooling characteristics of the splayed-pin variant, it's useful to start by considering the cooling attributes of traditional pin fin heat sinks. The latter provide outstanding cooling performance, as indicated by their low thermal resistance. The thermal-resistance value, stated in degrees Centigrade per watt (°C/W), quantifies the temperature rise in °C (above ambient) for each watt the device dissipates.

Several characteristics are responsible for the low thermal resistance of traditional pin fin heat sinks. The round shape of the pins, the omnidirectional structure of the pin array and its large surface area, and the high thermal conductivity of the base and pins all contribute to the heat sink's performance. Round pins offer less resistance to incoming airstreams than square or rectangular fins. That, combined with the omnidirectional structure of the pin array, enables surrounding airstreams to easily enter and exit the pin array.

The turbulence created when the airstream strikes the round pins further increases the airflow. As a result of the low resistance to airflow and turbulence within the pin array, the large surface area of the heat sink is exposed to a significant volume of airflow.

The highly conductive alloys that are used to fabricate pin fin heat sinks also contribute to performance. Both tradition-
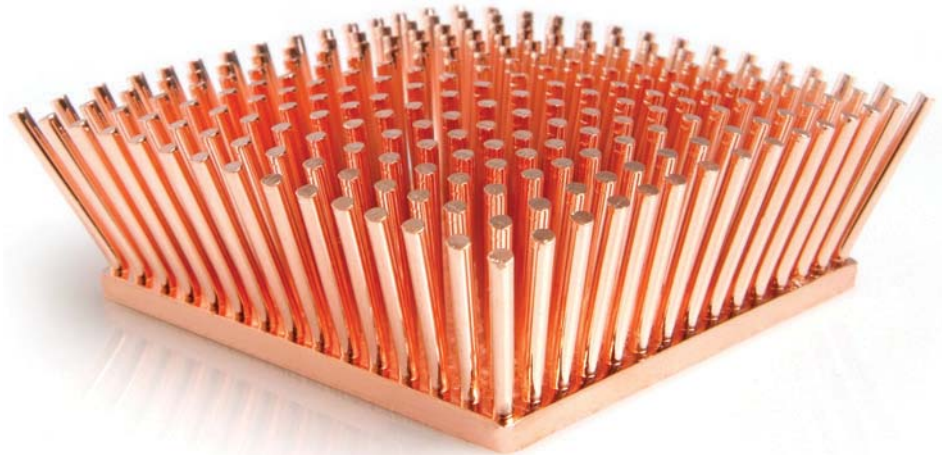
Figure 1 – Splayed copper pin fin heat sink

Figure 2 – Traditional aluminum pin fin heat sink

al and splayed-pin fins are manufactured in a forging process that permits use of the AL1100 and CDA110 alloys, which feature higher thermal conductivities than alloys used in other styles of heat sinks.

Splayed pins take the cooling performance of traditional pin fins to an even higher level by increasing the spacing between the pins. To understand the impact of the increased spacing on heat sink performance, we have to consider the conflict between surface area and

pin/fin density that is inherent in heat sink design.

For a heat sink to provide substantial cooling, it must possess sufficient surface area. Otherwise, with very limited surface area, the heat sink will not be able to dissipate much heat. At the same time, the more surface area the heat sink features (that is, the more pins or fins it contains), the harder it will be for surrounding airflows to enter the pin array. And unfortunately, without the exposure to surrounding air-

Widening the gap between pins allows air to pass through more easily. So the speed of the air exiting the heat sink is closer to the speed of the air entering.

flows, a heat sink will not be effective, regardless of its total surface area.

So, increasing surface area by packing fins closer together aids a heat sink's cooling performance. Paradoxically, however, it also detracts from its performance by reducing the airflow. This is the inherent conflict suppliers face in creating any heat sink with vertically oriented fins.

But splayed-pin fins overcome this surface area-vs.-density conflict by bending the pins outward. This technique increases the spacing between the pins substantially for a given footprint without giving up surface area. As a result, surrounding airstreams can enter and exit the pin array much more easily. The surfaces of the heat sink are exposed to faster airspeeds and the amount of cooling the heat sink provides is much greater. This improvement is most beneficial at lower airspeeds, because the lower the airspeed, the more difficult it is for the surrounding air to enter a heat sink pin array. So the performance premium afforded by splayed-pin fin heat sinks will be greatest in low-airspeed environments.

An additional advantage of the splayed-pin design is its low pressure drop. Widening the gap between pins allows air to pass through the heat sink more easily. So the speed of the air exiting the heat sink is closer to the speed of the air entering, when compared with heat sinks on which pins are spaced more closely together. The impact of pressure drop is especially important in boards that contain a large number of heat sinks and other components—a low pressure drop means more airflow is available to cool devices downstream from the fan.

We conducted two experiments to illustrate the advantage designers can gain by switching to the splayed-pin fin design. Both experiments compare splayed and traditional pin fin heat sinks that feature the same footprint, height, pin count, surface area and metallurgy. When looking at the results, it's important to note that traditional pin fins are highly powerful by

nature and, on their own, are considered one of the most efficient heat sink technologies available today.

Experiment 1: Cooling a Single Advanced FPGA
The power or heat dissipated by a given FPGA is positively correlated with the number of gates it has. Naturally, the more gates in the FPGA, the more heat it will dissipate. For our first experiment, we chose an advanced FPGA device that dissipates 30 W in a 42.5-mm² package. First, we powered up this FPGA with a traditional pin fin heat sink and took

temperature measurements to determine the heat sink's thermal resistance. Then we changed to a splayed-pin fin heat sink, repeated the measurements and recalculated thermal resistance. Both heat sinks featured a footprint of 2.05 x 2.05 inches, a height of 0.6 inch and were com-

posed of the highly conductive AL1100 aluminum alloy.

We ran this experiment three times, each time at a different airspeed. In the first test, the approaching airspeed was a moderate 400 linear feet per minute (LFM); in the second test, we reduced airspeed to 200 LFM (a low airspeed); and in the third test, we dropped it to 100 LFM (a very low airspeed).

The results, shown in Table 1, demonstrate outstanding cooling performance for both the splayed and traditional pin fins. Nevertheless, the switch to the splayed design produced a substantial premium in cooling

Heat sink style	Thermal resistance (°C/W)		
	@100 LFM	@200 LFM	@400 LFM
Traditional pin fin heat sink (model #3-202006U)	1.60	1.08	0.71
Splayed-pin fin heat sink (model #3-202006P)	1.29	0.95	0.68
Reduction in thermal resistance with splayed-pin fin	24%	14%	4%

Table 1 – Heat sink performance when cooling a single FPGA dissipating 30 W at different airspeeds

Heat sink style	Thermal resistance (°C/W)		
	Heat sink 1	Heat sink 2	Heat sink 3
Traditional pin fin heat sink (model #3-202011U)	0.47	0.67	0.73
Splayed-pin fin heat sink (model #3-202011P)	0.44	0.52	0.58
Reduction in thermal resistance with splayed-pin fin	6.5%	29%	26%

Table 2 – Heat sink performance when cooling three devices aligned with a fan producing 400 LFM of airflow

performance in the lower-airspeed environments. While at 400 LFM the splayed heat sink was only slightly better than the traditional model, at 200 LFM it was already 14 percent more effective. And at 100 LFM, the improvement was a dramatic 24 percent reduction in thermal resistance. These results

demonstrate the increasing value of the splayed-pin fin design at lower airspeeds.

Experiment 2: Cooling Multiple FPGAs

Boards that contain a large number of devices present much more complex cooling challenges than those equipped with a single FPGA. That's because the multidevice situations require that surrounding airflows be shared among the devices on the board. When cooling multiple "hot" FPGAs, design engineers must consider not only the thermal resistance of the heat sink but also the pressure drop of each heat sink. The lower the pressure drop, the more air will be available for cooling devices that are positioned downstream from the air source.

Splayed-pin fins feature a lower pressure drop than vertically constructed heat sinks, as there is more space for the air to enter and exit the pin array. To illustrate the dual advantage (lower pressure drop and added cooling) of splayed-pin fins in multi-FPGA environments, we constructed a simple experiment that involved putting three heat sinks on identical FPGAs placed in a row behind a fan. The fan pushed a given airflow and then we took the temperature measurements needed to determine heat sink thermal resistance. Each FPGA dissipated 30 W. We ran the experiment twice, once with three traditional pin fins and then again with three splayed-pin fins. The heat sinks we used measured 2.05 x 2.05 inches x 1.1 inch tall, and the fan was blowing 400 LFM in a free-air environment.

The results of this experiment demonstrate that the switch to splayed-pin fins on a board with multiple heat sinks pays a huge dividend. The experiment showed a 26 to 29 percent reduction in thermal resistance for the second and third devices with the splayed heat sinks. This performance premium is a product of the lower thermal resistance of the heat sink and its lower pressure drop.

Looking Forward

As heat loads dissipated by cutting-edge FPGAs continue to escalate, designers will require even greater cooling performance from their heat sinks. In some cases, a passive heat sink by itself will be insufficient and designers will be forced to adopt active heat sink solutions such as fan sinks, which mount a fan directly on the heat sink. In time, we can expect thermal-management vendors to offer more and more fan sink solutions.

One example of a new, high-performance fan sink solution is an integrated model that embeds a fan within a very efficient pin fin heat sink (Figure 3). Taking advantage of the added turbulence caused by the round pins and the large surface area achieved by the pin array, this integrated fan sink attains outstanding cooling performance in a very low-profile package that can fit into ATCA and PCI Express applications.

Until the fan sink comes into widespread commercial use, however, designers will gain an edge by using the splayed-pin fin heat sink in their FPGA-based designs.

Figure 3 – Integrated copper fan sink, the next step in cooling
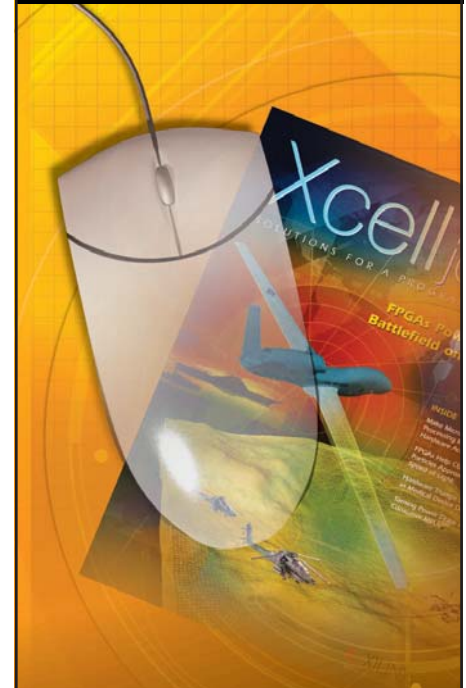
Supercharge MicroBlaze Processing with Hardware Acceleration

Deft design around MicroBlaze core creates powerful systems that can outperform standard controllers or DSPs.

by Karsten Trott
Field Application Engineer
Xilinx GmbH (Munich, Germany)
karsten.trott@xilinx.com

There are many algorithms in the world that can be changed into pure hardware to accelerate your processor. An average standard-deviation algorithm, creation of minimum or maximum values in a given time range, filters and FFTs are typical algorithms that can be moved into hardware without a big effort. But even more-unusual algorithms such as bit reversing can make the transition with the right hardware accelerator.

Xilinx's flexible embedded systems make it easy to attach hardware accelerators to an FPGA-based solution. Given the high computational power of FPGAs, such a system can easily outperform any standard processor, controller or DSP.
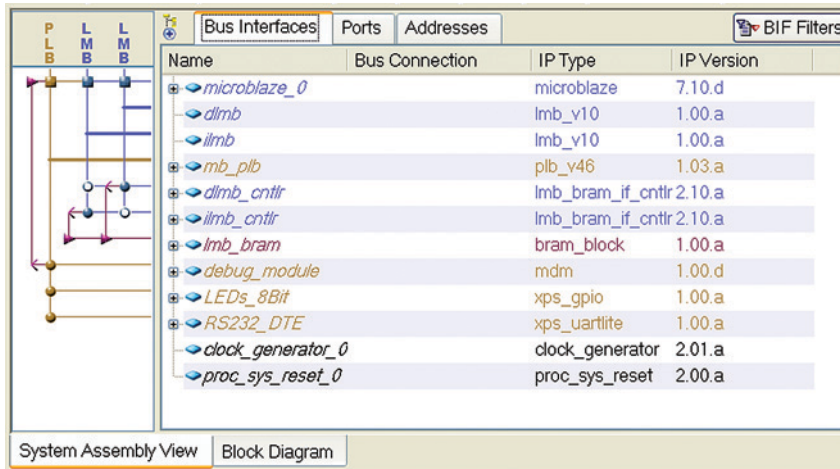
Figure 1 – Typical MicroBlaze design using only embedded memory

The MicroBlaze™ processor, one of the two 32-bit cores Xilinx provides in the Embedded Development Kit (EDK), is an apt vehicle for implementing hardware acceleration. A typical MicroBlaze design looks like the one in Figure 1—the core itself contains a 32-bit multiplier, but no floating-point unit (FPU), barrel shifter or special hardware accelerators. For Spartan® FPGA devices, the default system is set to contain an area-optimized version of the MicroBlaze (using a three-stage pipeline), but most customers will often start with the speed-optimized version (with a five-stage pipeline) to do performance evaluations. It is small and simple, but can easily be extended.

A couple of real-world examples that our customers were requesting on that processor design will show the power of the MicroBlaze route to hardware acceleration. We will focus on Spartan devices to demonstrate that we can achieve price/performance even if we compare FPGA solutions to standard controller cores. But the same techniques apply to Virtex® FPGAs as well.

Implementing a Bit-Reverse Algorithm

For the first example application, let's assume that the MicroBlaze processor is running at only 50 MHz. This is quite easy to achieve in any Spartan-3 or Spartan-6 device. All internal buses, such as local memory buses (instruction and data LMB) as well as the processor local

bus (PLB), are likewise running at 50 MHz. For simplicity's sake, let's assume no external DDR memory is attached.

Now imagine a customer who wants to implement a bit-reverse algorithm on that CPU. The MicroBlaze itself does not provide this functionality directly by hardware. Let's assume further that the bit reversing has to be done 20,000 times per second.

To solve this problem, most customers will start first with a pure software implementation, since that's the easiest way to achieve the desired functionality. And if the performance is sufficient, then there is no need to change anything.

To that end, let's start with a small and simple solution implemented as a simple software algorithm. It turns out to be small, nice and easy to understand—but quite inefficient:

```
unsigned int v=value;
unsigned int r = v;
int s = sizeof(v) * CHAR_BIT - 1;
for (v >>= 1; v; v >>= 1)
{
  r <<= 1;
  r |= v & 1;
  s--;
}
r <<= s;
return r;
```

This implementation worked quite well, but it took 220 cycles to run the

algorithm for a single 32-bit word on a MicroBlaze that was optimized for speed (using the five-stage pipeline). To execute the 20,000 required operations took approximately 88 milliseconds on our 50-MHz MicroBlaze.

The customer now tried to optimize the algorithm by using a slightly different approach, still implemented as a pure software solution:

```
unsigned int v=value;
unsigned int r=0;
if (v & 0x00000001) r |= 0x80000000;
if (v & 0x00000002) r |= 0x40000000;
if (v & 0x00000004) r |= 0x20000000;
if (v & 0x00000008) r |= 0x10000000;

if (v & 0x00000010) r |= 0x08000000;
if (v & 0x00000020) r |= 0x04000000;
if (v & 0x00000040) r |= 0x02000000;
if (v & 0x00000080) r |= 0x01000000;

if (v & 0x00000100) r |= 0x00800000;
if (v & 0x00000200) r |= 0x00400000;
if (v & 0x00000400) r |= 0x00200000;
if (v & 0x00000800) r |= 0x00100000;

if (v & 0x00001000) r |= 0x00080000;
if (v & 0x00002000) r |= 0x00040000;
if (v & 0x00004000) r |= 0x00020000;
if (v & 0x00008000) r |= 0x00010000;

if (v & 0x00010000) r |= 0x00008000;
if (v & 0x00020000) r |= 0x00004000;
if (v & 0x00040000) r |= 0x00002000;
if (v & 0x00080000) r |= 0x00001000;

if (v & 0x00100000) r |= 0x00000800;
if (v & 0x00200000) r |= 0x00000400;
if (v & 0x00400000) r |= 0x00000200;
if (v & 0x00800000) r |= 0x00000100;

if (v & 0x01000000) r |= 0x00000080;
if (v & 0x02000000) r |= 0x00000040;
if (v & 0x04000000) r |= 0x00000020;
if (v & 0x08000000) r |= 0x00000010;

if (v & 0x10000000) r |= 0x00000008;
if (v & 0x20000000) r |= 0x00000004;
if (v & 0x40000000) r |= 0x00000002;
if (v & 0x80000000) r |= 0x00000001;
return r;
```
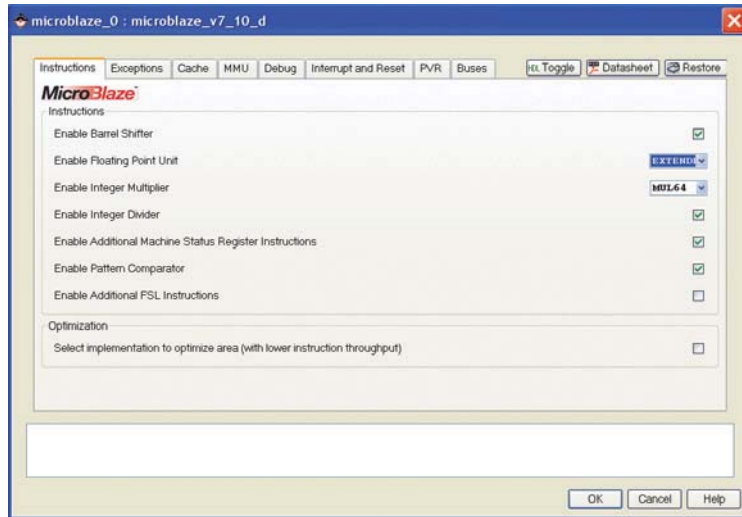
Figure 2 – Activation of the barrel shifter in the MicroBlaze properties

That code is much longer, but it is more efficient. Code execution on the standard MicroBlaze took 130 cycles for a single 32-bit word. This is quite an improvement, but still too long. The whole execution time for the 20,000 operations is now roughly 52 ms.

Then the customer did some research on the Internet to find a better algorithm and came upon this one:

```
unsigned x = value;
unsigned r;
x = (((x & 0xaaaaaaaa) >> 1) | ((x
& 0x55555555) << 1));
x = (((x & 0xcccccccc) >> 2) | ((x
& 0x33333333) << 2));
x = (((x & 0xf0f0f0f0) >> 4) | ((x
& 0x0f0f0f0f) << 4));
x = (((x & 0xff00ff00) >> 8) | ((x
& 0x00ff00ff) << 8));
r = ((x >> 16) | (x << 16));
return r;
```

This code looks quite efficient and even small. And it does not need any branching that might break the pipelining. It runs on the core system with only 29 cycles.

However, the algorithm makes use of shift operations of 1, 2, 4, 8 and 16 bits. Let's activate the barrel shifter in the property window of the MicroBlaze, as seen in Figure 2. The barrel shifter allows us to execute shift instructions in a single cycle,

independent of the length of the shift operation. It makes it possible to run the pure software algorithm slightly faster on our MicroBlaze.

Activating the barrel shifter in the MicroBlaze hardware reduces the number of cycles it takes to process the algorithm to 22. This is quite an improvement from the first version of the software algorithm. It now takes the algorithm only about 8.8 ms to run the whole 20,000 times, a tenfold improvement—but still not enough for the customer.

This is where the pure software solution, even if supported by parameter changes of the MicroBlaze core, runs out of steam. It's the point when a pure hardware solution needs to kick in, via a new approach that makes full use of hardware accelerators.

To accelerate such a basic operation, only a very simple core need be attached to a MicroBlaze Fast Simplex Link (FSL). The standard FSL implementation uses an FSL bus, including synchronous or asynchronous FIFOs, to transmit data from the MicroBlaze core to the FSL hardware accelerator intellectual property (IP). The FSL bus with the FIFOs decouples the accesses from both sides.

If the standard implementation with an FSL bus including FIFOs is used, then the typical execution time is four cycles: one cycle to write the data from the MicroBlaze to the FSL bus into the FIFO, one cycle to transfer the data from that FIFO to the FSL IP, one cycle to transfer the result back from the FSL IP to the FSL bus FIFO and one cycle to read the result from the FSL bus into the MicroBlaze.

The connection from the MicroBlaze to the FSL bus and from the FSL bus to the FSL IP is easy to create in the graphical view of the EDK, as seen in Figure 3.

There is still room for improvement, however. The latency in the algorithm is crit-
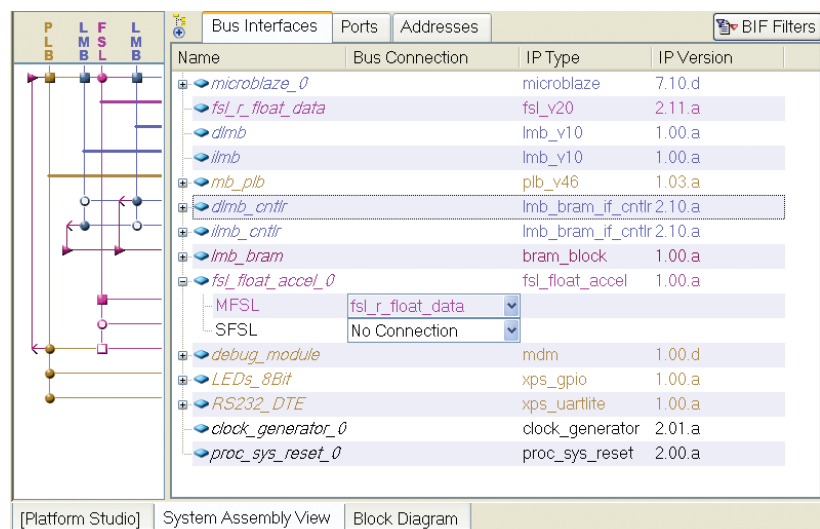

Figure 3 – Connecting FSL hardware using the FSL bus

The hardware core will execute the bit-reverse operation in only two cycles—one cycle to write data to the IP, one cycle to read it back. The processing of all 20,000 bit-reverse operations now takes only 0.8 ms.

ical and should be as small as possible. But our implementation with the two FSL buses is still requiring four clock cycles. We can halve the latency, to only two clock cycles, by changing this connection to a direct link between the MicroBlaze and the hardware accelerator. Now it takes just one cycle to write the data to the FSL hardware accelerator IP and one cycle to read the result back.

There are a few things to take care of when using direct links. First, the coprocessor IP should store the inputs and provide the results in a registered way. Note that there is no FSL bus with FIFO to handle that operation.

Moreover, running the MicroBlaze and the FSL IP at different clock speeds is quite tricky in this case. To avoid conflicts, designers would be best served by running the two at the same speed.

But how to connect the MicroBlaze and the FSL IP directly without an FSL bus in between? That's quite easy to do—just connect the data lines of the MicroBlaze and the hardware accelerator. Then attach the handshaking signals when required.

For the example with the bit-reversing IP, only a write signal is required—the IP is always fast enough to react to any requests from the MicroBlaze.

The IP itself is quite simple. Here is an extract of the VHDL code:

```
architecture behavioral of
    fsl_bitrev is
    -- data value sent by microblaze:
    signal data_value  :
std_logic_vector(0 to 31) := (oth-
ers=>'0');

begin
    -- bitreversed value to write
back:
```

```
FSL_M_Data <= data_value;

process(FSL_Clk)
begin
    if rising_edge(FSL_CLK) then
    if (FSL_S_Exists = '1') then
      -- create the bitreversed data:
    data_value(0) <= FSL_S_Data(31);
    data_value(1) <= FSL_S_Data(30);
    data_value(2) <= FSL_S_Data(29);
        ...
    data_value(30) <= FSL_S_Data(1);
    data_value(31) <= FSL_S_Data(0);
      end if;
    end if;
  end process;
end architecture behavioral;
```

To attach that IP without any FSL bus in between, you must make the following changes to your project's MHS file:

```
BEGIN microblaze
  ...
  PARAMETER C_FSL_LINKS = 1
  ...
  PORT FSL0_S_EXISTS = net_vcc
  PORT FSL0_S_DATA = FSL0_S_DATA
  PORT FSL0_M_DATA = FSL0_M_DATA
  PORT FSL0_M_WRITE = FSL0_M_EXISTS
  PORT FSL0_M_Full = net_gnd
END

BEGIN fsl_bitrev
  PARAMETER INSTANCE = fsl_bitrev_0
  PARAMETER HW_VER = 1.00.a
  PORT FSL_S_DATA = FSL0_M_DATA
  PORT FSL_S_EXISTS = FSL0_M_EXISTS
  PORT FSL_M_Data = FSL0_S_DATA
  PORT FSL_M_Full = net_gnd
  PORT FSL_Clk = clk_50_0000MHz
END
```

Now the game has changed dramatically. The hardware core will execute the bit-reverse operation in only two cycles—one cycle to write data to the IP, one cycle to read it back. The processing of all 20,000 bit-reverse operations now takes only 0.8 ms.

This is an improvement over the very first algorithm of 110 times. To put it another way, it would require a processor running at about 5.5 GHz to run the application in the same amount of time using the first algorithm—quite impressive by any measure.

Comparing the approach with the last, most effective software algorithm, the system is still 11 times better. Otherwise, it would require a CPU running at 550 MHz to execute the same algorithm in pure software (assuming you have a barrel shifter in your CPU).

This example, of course, is valid only if your CPU does not provide bit-reverse addressing at all. Most DSPs have that functionality, but virtually no microcontrollers do. And having the capability to add such functionality can dramatically speed up the processing of such an algorithm.

The change was small, but quite efficient. We even reduced the code size to only two words. Of course, some additional slices are required for the hardware now. But this trade-off was worth the speedup in a solution that outperforms any standard microcontroller.

Speedy Floating-Point Performance

In another example of the power of MicroBlaze acceleration for algorithms, a customer claimed that his floating-point processing was running extremely slowly on the MicroBlaze system. The algorithm had a simple loop to create a few results at once.

```
for (i=0;i<512;i++) {
  f_sum           += farr[i];
```

Sometimes small changes can have a dramatic effect in the processing of your algorithm. And implementing those small changes can help your 50-MHz MicroBlaze system compete against a high-performance DSP.

```
f_sum_prod += farr[i] * farr[i];
f_sum_tprod        += farr[i] *
farr[i] * farr[i];
f_sqrt            +            =
sqrt(farr[i]);
    if (min_f > farr[i]) { min_f =
farr[i]; }
    if (max_f < farr[i]) { max_f =
farr[i]; }
}
```

All values were single-precision floating-point values. The first idea that came to mind was the most basic question of all: Is the floating-point unit activated? Examining the project settings, we found out that the FPU was still disabled. That's why it took forever to calculate just these few numbers. Activation of the FPU takes places inside the MicroBlaze property settings (see Figure 4).

There are two versions of the FPU support. We chose Extended FPU to support the sqrt() operation too. The whole loop for 512 values took now 1,108,685 cycles on our 50-MHz MicroBlaze. Just a quick look into the generated assembler code showed that the math-lib functionality was still being used to create the square root. That is defined in the math functionality as:

```
double sqrt(double);
```

But the customer used the square-root function only to process floating-point values. For that reason the MicroBlaze FPU defined a new function to work around that function, resolving the issue:

```
float sqrtf(float);
```

Changing the line *f_sqrt* += *sqrt(farr[i])*; to *f_sqrt* += *sqrtf(farr[i])*;

caused the usage of the FPU internal square-root functionality inside the MicroBlaze. Now the code ran in only 35,336 cycles. Once again, we achieved

this improvement of 31 times by means of only small changes, especially compared with the first implementation, which did not use the FPU at all. You would need a
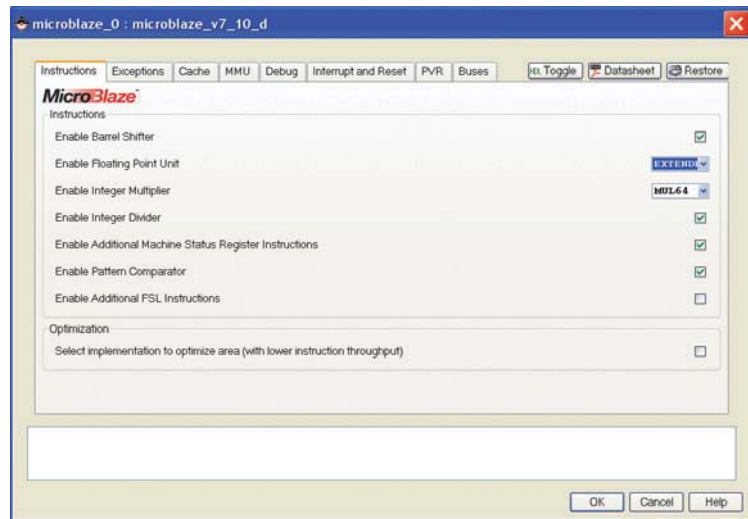

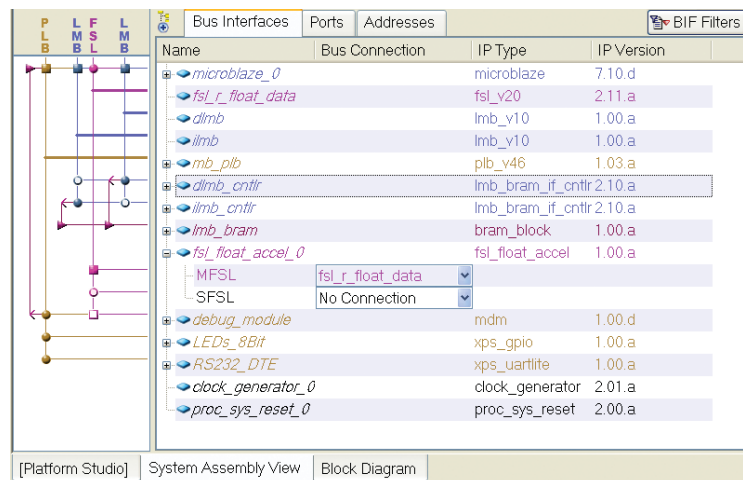*Figure 4 – Activation of extended FPU support*

Figure 5 – System using a parallel floating-point accelerator

CPU running at roughly 1.5 GHz to create those results in the same execution time.

But the customer was still not satisfied; he was looking for even more speed. In this case, changing the algorithm from floating- to fixed-point arithmetic was not a suitable option. So instead, we created a new, special hardware accelerator (new FSL IP) to speed up the processing of the loop.

The new FSL IP utilizes the CORE Generator™ module floating-point_v4_0 to create nine instances for the operations 4x ADD, 2x MUL, 1x GREATER, 1x LESS and 1x SQRT. All of them are instantiated and working fully in parallel on the same input data (Figure 5).

The instances in the FSL IP are created with some internal latency, but a throughput of 1. This required some additional slices for the controller hardware inside the accelerator, but makes it possible to feed the coprocessor with new data every clock cycle.

Only at the end of the processing loop are some additional cycles required before the results can be retrieved.

We connected the MicroBlaze to the FSL IP using a direct link—FIFO was not required. All data that is transferred will be buffered inside the IP and processed immediately afterward.

The link from the FSL IP back to the MicroBlaze was created using an FSL bus. This was easier to achieve, because we have to send back some results—and that's simpler to do inside the IP. Some of the CoreGen modules have some latency that is being added to the execution time and fully covered by the getfsl() call. The MicroBlaze just waits until all results are stored into the FSL bus FIFOs. But this is no problem as long as the data rate is 1 to achieve the required throughput.

The additional delay of the FSL bus does not cost too much (a few cycles only). The C code to use the FSL IP looks like this:

```
for (i=0;i<512;i++) {
  putfsl(farr[i],fsl0_id);
}
```

```
// get the min,max values:
getfsl(min_f,fsl0_id);
getfsl(max_f,fsl0_id);
// get the sum and products:
getfsl(f_sum,fsl0_id);
getfsl(f_sum_prod,fsl0_id);
getfsl(f_sum_tprod,fsl0_id);
getfsl(f_sqrt,fsl0_id);
```

The final implementation of that algorithm required only about 4,630 cycles. And it is still a full floating-point implementation.

The hardware calculates all results in parallel at the cost of additional slices that are used to implement the hardware accelerator. But in the end, we gain an improvement of factor of about 7.6 compared with the extended FPU implementation. Otherwise, replacing this 50-MHz processor with a standard processor would require a CPU running at roughly 380 MHz (assuming it had a floating-point sqrt function in hardware).

Even more dramatic is the comparison to the original version utilizing the FPU, but with the sqrt() function used: an overall improvement of approximately 239 times. This would require a floating-point processor running at roughly 12 GHz.

As these examples show, sometimes small changes can have a dramatic effect in the processing of your algorithm. And implementing them can help your 50-MHz MicroBlaze system compete against a high-performance DSP.

First, identify the core algorithm that takes too long to execute, then try to speed it up—either by simple changes in software, by the use of hardware or by more complex changes using hardware accelerators. And your processor system will always outperform any standard controller.

Karsten Trott is a Xilinx FAE in Munich, Germany. He holds a PhD in analog chip design and has a strong background in chip design and synthesis. You can reach him at Karsten.Trott@xilinx.com.
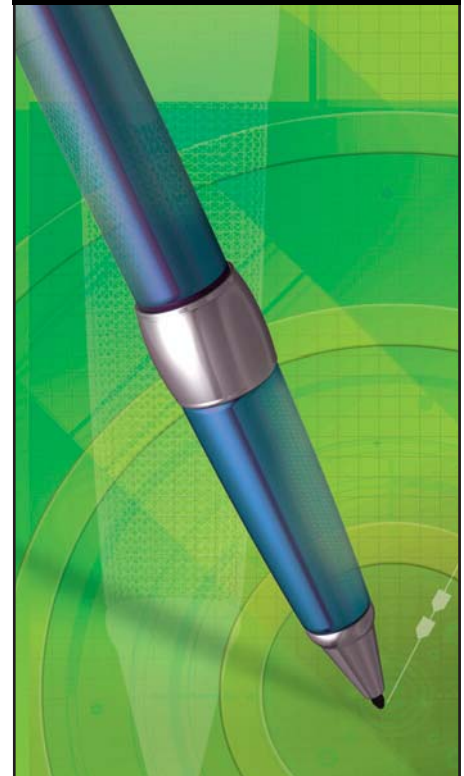
Application Notes

If you want to do a bit more reading about how our FPGAs lend themselves to a broad number of applications, we recommend these notes.
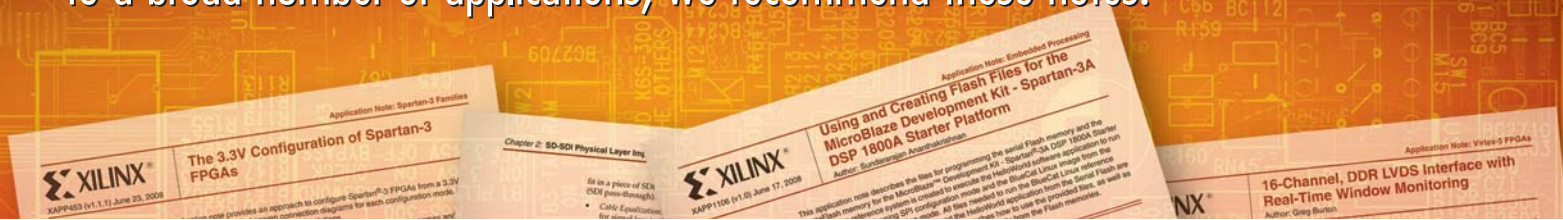
XAPP864: SEU Strategies for Virtex-5 Devices

http://www.xilinx.com/support/documentation/application_notes/ xapp864.pdf

Single-event upsets have the potential to affect most digital electronic circuits. Xilinx takes the issue of SEUs seriously, and designs its devices to have an inherently low susceptibility to these radiation-caused events. Because Xilinx also recognizes that SEUs are unavoidable within commercial and practical constraints, the company provides built-in SEU detection in the Virtex®-5 and Extended Spartan®-3A families to simplify and improve the system design.

An application note by Ken Chapman and Les Jones discusses strategies and representative calculations for handling SEUs with an emphasis on reliability when addressing these low-probability events. An accompanying reference design is optimized for use with the Virtex-5 FPGA ML505 evaluation platform, but can port to other hardware. You can use its SEU controller macro in any Virtex-5 FPGA design to implement an SEU detection and correction scheme.

Due to the infrequent and unpredictable nature of real SEUs, small-scale testing of their effects is impractical, as is system verification. For this reason, the SEU controller macro and reference design can emulate an SEU by deliberately injecting an error into the FPGA configuration so that its subsequent detection and correction can be confirmed. You can also use the technique of injecting errors to assess SEU mitigation circuits implemented in a design.

XAPP1137: Linux Operating System Software Debugging Techniques with Xilinx Embedded Development Platforms

http://www.xilinx.com/support/documentation/application_notes/ xapp1137.pdf

In this application note involving Linux issues, author Brian Hill discusses debugging techniques for the Linux operating system, including debugging boot issues, kernel panics, software and hardware debuggers, driver-application interaction and various other tools. This application note, which includes a reference system built for the Xilinx ML507 Rev A board, is best suited to users who are comfortable configuring, building and booting Linux on a Xilinx embedded platform.

XAPP1140: Embedded Platform Software and Hardware In-the-Field Upgrade Using Linux

http://www.xilinx.com/support/documentation/application_notes/ xapp1140.pdf

New features and bug fixes often necessitate upgrading flash images to replace the existing FPGA bitstream, boot loader, Linux kernel or file system. It's a challenge to provide a convenient mechanism with which end users can perform this task. This application note by Brian Hill discusses an in-the-field upgrade of the Virtex-5 FXT bitstream, Linux kernel and loader flash images, using the presently running Linux kernel. Upgrade files come from either a USB mass-storage device, using the XPS USB host core, or over the network, from an FTP server. The running Linux image performs the flash upgrade. The note includes a reference design and an example methodology.

XAPP1141: The Simple MicroBlaze Microcontroller Concept

http://www.xilinx.com/support/documentation/application_notes/ xapp1141.pdf

A small microcontroller programmed in C or C++ can be more efficient than doing the same function in HDL. An application note by Christophe Charpentier provides a new way to easily add a simple MicroBlaze™ microcontroller to an FPGA design without having to learn about new tools. This small-form-factor 32-bit microcontroller based on the MicroBlaze processor is instantiated directly into the HDL. You can use it immediately in a standard FPGA design flow without special scripts or complicated steps. You need only three files to get started.

XAPP1136: Integrating a Video Frame Buffer Controller in System Generator

http://www.xilinx.com/support/documentation/application_notes/ xapp1136.pdf

This application note by Douang Phanthavong and Jingzhao Ou describes how to integrate an embedded processor system with the Xilinx Multi-Port Memory Controller (MPMC) and Video Frame

Buffer Controller (VFBC) IP cores in System Generator for DSP. You can then develop custom logic inside System Generator, attach it to the imported processor system and, as the authors show, build a powerful validation platform through System Generator's hardware co-simulation capability. Combined with the MPMC, the VFBC is an ideal solution for applications such as motion estimation, video scaling, on-screen displays and video capture used in video surveillance, videoconferencing and video broadcast products.

Integrating an off-the-shelf double-data-rate or DDR2 SDRAM memory controller into a system is a time-consuming task. Verifying that the design runs correctly is also not trivial. The MPMC is a parameterizable memory controller that supports SDRAM, DDR and DDR2 memory access. It provides access to various types of external memory by using one to eight ports, each of which can be configured from a set of Personality Interface Modules. The VFBC is one of these PIMs. It is unique in that user IP or other interface circuits can read and write data in two-dimensional sets regardless of the size and organization of the external memory. This allows you to access the external memory without having to know all the details about the complicated external memory-access protocols.

The note emphasizes tool and design flows rather than the technology details of the IP cores. The intent is to show you how to connect the MPMC and the VFBC inside System Generator using the Xilinx ML506 hardware platform, although you could apply the same design flows and methodologies to other boards. An example video application—a simplified version of the reference design currently available in the Xilinx Video Starter Kit—illustrates how to exercise and validate the VFBC using existing System Generator and Platform Studio integration.

It is crucial to provide seamless, transparent and easy-to-use tool flows when designing a complex system around these popular IP cores. Fortunately, the well-integrated flows between System Generator and Platform Studio allow you to quickly put together a system in a matter of hours or days instead of weeks or months.

XAPP458: Implementing DDR2-400 Memory Interfaces in Spartan-3A FPGAs

http://www.xilinx.com/support/documentation/application_notes/xapp458.pdf

With their requirement for low-cost, high-bandwidth memory, high-end consumer products demand high-performance DDR2 memory interfaces. Xilinx integrates a Memory Interface Generator (MIG) in the CORE Generator™ software for ultimate design flexibility and ease of use. This free, user-friendly tool is designed to create memory interfaces in unencrypted RTL. The MIG supports multiple memory architectures across a variety of FPGA selections, providing the flexibility you need to easily customize your designs.

In this application note, author Eric Crabill discusses the DDR2-400 (200-MHz) memory interface that Xilinx has validated in Spartan-3A FPGAs with the higher speed grade. The validation results also apply to Spartan-3AN and Spartan-3A DSP FPGAs with the same speed grade. This DDR2-400 interface is derived from the MIG's default output.

The design is fully verified in hardware using Spartan-3A FPGAs assembled on Spartan-3A Starter Kits. The validation effort includes characterization at different process corners, as well as temperature and voltage variations that meet commercial-grade requirements.

XAPP940 (Updated): Using Xilinx CPLDs as Motor Controllers

http://www.xilinx.com/support/documentation/application_notes/xapp940.pdf

Different electronic systems use many types of motors, each with its own advantages. Standard DC motors are often found in automotive as well as consumer applications and even toys. Brushless DC motors are commonplace in power tools and, along with standard DC motors, can operate in the medium to high speed range. AC induction motors are often found in white goods and transportation applications, and these can operate at very high speeds. On the lower end of the operating-speed range are stepper motors. These models have a huge variety of uses in the consumer and industrial markets, including a variety of positioning systems along with printers, scanners, plotters, disk drives, fax machines and medical equipment.

This application note shows how to use a Xilinx CPLD as a simple stepper motor controller. The Xilinx CoolRunner™-II CPLD is the perfect device with which to control stepper motors, with the added benefit that it can be reprogrammed to accommodate a different motor if the system specifications change.

XAPP1121: Reference System – Optimizing Performance in PowerPC 440 Processor Systems

http://www.xilinx.com/support/documentation/application_notes/xapp1121.pdf

The PowerPC® 440 processor block has many performance advantages compared with previous embedded solutions. This application note by James Lucero discusses how to measure and optimize performance in PPC 440 systems, an important factor in ensuring the system is optimally built.

The author describes a reference design that can improve system performance in the PowerPC 440 Processor Block on the Virtex-5 FXT FPGA while optimizing performance for embedded DMA solutions. In the system, he shows how to connect the XPS Central DMA master interface to the Processor Local Bus v4.6 on either PLB Slave 0 or PLB Slave 1. He then modifies the XPS Central DMA parameters for the DMA engine to allow for parallel reads and writes through the crossbar. For HDMA, Lucero discusses setting threshold values for interrupts and changing addresses in main memory for buffer descriptors and transmit/receive buffers. In this, he connects a simple loopback core to the LocalLink interface on one HDMA. In addition, he includes performance cores for PLB v.4.6 master interfaces and HDMA to the design to measure system performance before and after system optimizations.

The note includes two standalone software applications to demonstrate DMA transactions for XPS Central DMA and HDMA to DDR2.

Xilinx Tool & IP Updates

Xilinx is continually improving its products, IP and design tools as it strives to help designers work more effectively. Here we report on the most current updates to the flagship FPGA development environment, the ISE® Design Suite, as well as to Xilinx IP, as of September 2009. Also, look for new tools, IP and development boards from Xilinx partners in the Tools of Xcellence section of this issue.

Product updates offer significant enhancements and new features to three versions of the ISE Design Suite: the Logic, Embedded and DSP editions. Keeping your installation of ISE up to date is an easy way to ensure the best results for your design. Updates to the ISE Design Suite are available from the Xilinx Download Center at www.xilinx.com/download. For more information or to download a free 30-day evaluation of ISE, visit www.xilinx.com/ise.

ISE Design Suite: Logic Edition

Ultimate productivity for FPGA logic design
Latest version number: 11.3
Date of latest release: September 2009
Previous release: 11.2
URL to download the latest patch:
www.xilinx.com/download

Revision highlights:
This latest release of the ISE Design Suite: Logic Edition supports the new Virtex®-6 HXT FPGA platform. The Virtex-6 family of devices delivers the industry's highest-bandwidth FPGA with up to 72 serial transceivers for applications such as bridging, switching and aggregation in wired telecommunications and data communications systems.

ChipScope™ Pro: The Integrated Bit Error Ratio Tester (IBERT) 2.0 now supports the Spartan®-6 LXT FPGA family.

iMPACT: Reading and programming of eFUSE registers is now supported for devices in the Spartan-6 family, and eFUSE support has been extended to the Linux operating system in addition to the 32-bit version of Microsoft Windows.

ISE Simulator (ISim): File names in the ISim console now link to the associated file. Also, ISim users now have the ability to clear the console for greater ease of use.

PlanAhead™ design analysis tool: PlanAhead now supports the creation of DCI Cascade groups and membership editing. A new SSN Predictor is available when targeting Virtex-6 FPGAs. In addition, new PlanAhead interface enhancements allow users to label pin rows in the package view when zooming. Additional DRCs are available for designs targeting Virtex-6 and Spartan-6 FPGAs.

Xilinx Power Analyzer (XPA): The Xilinx Power Analyzer provides greater ease of use with new capabilities to interrupt the power analysis process, support for bus reconstruction in the I/O view and the ability to select and edit multiple cells within the interface.

ISE Design Suite: Embedded Edition

An integrated software solution for designing embedded processing systems
Latest version number: 11.3
Date of latest release: September 2009
Previous release: 11.2
URL to download the latest patch:
www.xilinx.com/download

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

Xilinx Platform Studio (XPS) and the Embedded Development Kit (EDK): For command line users, Platform Flash XL is now supported, as is contract-based IP licensing. XPS has added Clock Generator support for Virtex-6 and Spartan-6 FPGA families as well as Clock Wizard support for the Multi-port Memory Controller (MPMC) for the Virtex-6 and Spartan-6.

ISE Design Suite: DSP Edition

Flows and IP tailored to the needs of algorithm, system and hardware developers
Latest version number: 11.3
Date of latest release: September 2009
Previous release: 11.2
URL to download the latest patch:
www.xilinx.com/download

Revision highlights:
All ISE Design Suite editions include the enhancements listed above for the Logic Edition. The following enhancements are specific to the DSP Edition.

Support for the Virtex-6 HXT FPGA platform: This latest release of the ISE Design Suite supports the new Virtex-6 HXT FPGA platform, delivering the industry's highest-bandwidth FPGA with up to 72 serial transceivers for applications such as bridging, switching and aggregation in wired telecommunications and data communications systems.

System Generator for DSP: This tool now supports the following new devices: Virtex-6 FPGA Lower Power (Virtex-6-1L), Virtex-6 HXT FPGA and Virtex-5Q FPGA. System Generator for DSP also provides support for JTAG hardware co-simulation for the Spartan-6 FPGA SP605 Development Platform.

New operating system support: System Generator for DSP has expanded its OS support to include Microsoft Windows Vista Business 32-bit (English), Red Hat

Enterprise Desktop Linux 5.2 (32- and 64-bit) and SUSE Linux Enterprise 10 (32- and 64-bit).

Blockset enhancements:

• **DDS Compiler 4.0** is now available in System Generator for DSP with a new option that makes it possible to use the block as a phase generator or SIN/COS lookup table only. This capability allows you to customize the direct digital synthesizer to fit your individual application needs. The spurious-free dynamic range (SFDR) has been increased from 120 dB to 150 dB. Also, new options include an ability to configure DDS using system-level parameters (SFDR, frequency resolution) or hardware parameters (phase and output width); and to trade off XtremeDSP™ slice usage for maximum performance. Another option allows designers to configure phase increment and phase offset as constant, programmable or dynamic for modulation.

(Note: This block supersedes the DDS Compiler 3.0 block. DDS Compiler version 4.0 is not bit-accurate with respect to earlier versions. Also, latency of phase offset effects has been balanced with the latency of phase increments for ease of use in the streaming modes. This change also applies to existing programmable and fixed modes.)

• **CIC Compiler 1.3** is now available in System Generator for DSP, supporting Virtex-6 and Spartan-6 FPGA platforms. An input and output streaming interface has been added for multiple-channel implementations. Also new is the capability to specify hardware oversampling specification as a sample period and to leverage the oversampling factor to optimize resource utilization.

(Note: This block supersedes the CIC Compiler 1.2 block. The RATE_WE signal no longer acts as a reset to the core; the core will update to the new rate on the next input sample, for a single-channel implementation, or the next input to the first channel, for multiple-channel implementations.)

• **Upsample Block** has added a new latency parameter.

• **Additional IP:** Xilinx has upgraded a number of blocks to support the latest version of the LogiCORE™ (with no change in block functionality). The Multiplier LogiCORE v11.2 leverages speed and area optimization for LUT implementation. Also, Block Memory Generator v3.3, Distributed Memory Generator v4.2 and FIFO Generator v5.3 have been upgraded to support Virtex-6 Lower Power (Virtex-6-1L), Virtex-6 HXT and Virtex-5Q FPGA.

Xilinx IP Updates

Name of IP: ISE IP Update 11.3
Type of IP: All

Targeted application: Xilinx develops IP cores and partners with third-party IP providers to decrease customer time-to-market. The powerful combination of Xilinx FPGAs with IP cores provides functionality and performance similar to ASSPs, but with flexibility not possible with ASSPs.
Latest version number: 11.3
Date of latest release: September 2009
URL to access the latest version:
www.xilinx.com/download

Informational URL: *www.xilinx.com/ipcenter/coregen/updates_11_3.htm*

Release notes: *www.xilinx.com/support/documentation/user_guides/xtp025.pdf*

Installation instructions:
www.xilinx.com/ipcenter/coregen/ip_update_install_instructions.htm

Listing of all IP in this release:
www.xilinx.com/ipcenter/coregen/11_3_datasheets.htm

Revision highlights: Starting with 11.1, all ISE CORE Generator™ IP Updates are bundled with quarterly ISE software updates. The latest versions of IP products have been tested and are delivered with the current IP release. Support for Virtex-6 HXT, Virtex-6 Lower Power and Virtex-5Q have been added to selected cores in this release. There are a number of new cores in this release, as described below.

FPGA features and design:

• **Spartan-6 SelectIO™ Interface Wizard v1.1** – Generates an HDL file that contains I/O logic such as IOSERDES and IODE-LAY blocks customized to the user's interface requirements.

• **Virtex-6 FPGA GTH Transceiver Wizard v1.1** – Generates a custom wrapper that configures one or more Virtex-6 FPGA GTH transceivers according to user requirements. In addition, it produces an example design, testbench and scripts to allow you to observe the transceivers operating under simulation and in hardware.

Video and image processing:

• **Video On Screen Display v1.0** – A sophisticated module that provides three hardware-accelerated functions including multiple alpha-blending layer compositors, simplified graphics processing unit (boxes) and simplified text processing unit for video systems.

• **Video Direct Memory Access v1.0** – Allows video cores to access external memory via the Video Frame Buffer Controller within the Multiport Memory Controller under control of the host processor.

Communications and networking:

• **RXAUI v1.1** – The Xilinx Reduced Pin 10 Gigabit Attachment Unit Interface (RXAUI) LogiCORE IP provides a two-lane high-speed serial interface, delivering total throughput of up to 10 Gbits/second. Operating at an internal clock speed of 156.25 MHz, the core includes the Dune Networks RXAUI implementation, the XGMII Extender Sublayers (DTE and PHY XGXS) and the 10GBASE-X sublayer, as described in clauses 47 and 48 of IEEE 802.3-2005. The core supports an optional serial MDIO management interface for accessing the IEEE 802.3-2005 clause 45 management registers.

Enhancements to the CORE Generator:

• The CORE Generator now checks for IP license availability before proceeding through the process of core generation.

• Automated core upgrade to latest-version capability has been added for the following IP cores: CIC Compiler v1.3, DDS Compiler v4.0, Distributed Memory v4.2 and Multiplier Generator v11.2.

Tools of Xcellence

News and the latest products from Xilinx partners

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

AVnu Audio/Video Network Alliance Promises Interoperability

Whether you are creating products that employ advanced audio and video for consumer systems, the automotive market, professional recording studios or even next-gen AV systems for stadiums, the newly formed AVnu Alliance wants to help you link all these products together and run them efficiently and harmoniously on the widely deployed IEEE 802.1 Layer 2 networks—without having to pay proprietary network royalties.

AVnu Alliance's president and chairman, Rick Kreifeldt, said advanced audio and video are found in an ever-growing number of products in an expanding number of markets, driving the need for these devices to be able to share bandwidth efficiently in a single network.

"According to a recent consumer electronics study, there are 27 or more devices in the home that could be networked for some kind of audio and video," said Kreifeldt, who is the vice president of Harman International. Because all these devices have to be synchronized, and must share data and bandwidth, they are susceptible to signal slowdowns, pops, hisses and even freezes as more products in the home or even neighborhood try to access data and stream video on residential networks. Professional audio and video recording and stadium-size AV systems are especially vulnerable to these problems, Kreifeldt said, because in many cases they transfer and share huge amounts of compressed and uncompressed data.

To get around these issues, many vendors face a build-or-buy decision. Vendors can license (for a fee) proprietary networks such as CobraNet from Cirrus Logic, or they can build their own customized backplane/network and develop their own protocols to interconnect components. A third way would involve establishing a base set of protocols running on an already established network standard that all AV-related manufacturers can support—without paying royalties. The approach could greatly facilitate new-product introduction and send those products into new markets.

That was the idea that impelled the formation, in late August, of the AVnu Alliance. Its goal is to promote an open network standard for time-synchronized, low-latency streaming services through 802.1 networks. Founding members include Avid Technology Inc., Broadcom Corp., Cisco Systems Inc., Harman International, Intel Corp., Marvell, Meyer Sound Laboratories, Samsung Electronics Co. and Xilinx Inc.

The AVnu Alliance members will support and promote the adoption of the open IEEE 802.1 Audio/Video Bridging (AVB) standard, along with the related IEEE 1722 and IEEE 1733 networking standards, combining them as a broader networking standard for nearly all products that receive, play, record and transfer audio/video data.

To further refine the specifications, the AVnu Alliance is collaborating with various IEEE working groups to form foundational standards. As of September, all were in draft form, with expected completion in 2010-11. In particular, the IEEE 802.1 AS specification defines a master clock that all 802.1 AVB-compliant components can reference to synchronize their playback and recording.

The IEEE 802.1 QAT task group defines a simple reservation protocol. This protocol allows applications on endpoint devices to notify the various network elements and other connected devices to reserve a particular amount of bandwidth for the stream.

Similarly, the IEEE 802.1 QAV task group is defining queuing and forwarding rules to ensure the stream can pass through the network efficiently and within the latency specified when creating bandwidth reservation.

Finally, the IEEE 802.1BA standard specifies inter-relations between the protocols and conditions for AVB systems.

"We expect the AVB protocols to be fully ratified in 2010 and 2011," said Kreifeldt. "It is our goal to have compliance and interoperability ready upon ratification."

Amy Chang, IP marketing manager at Xilinx, said Xilinx became an active member of the AVnu Alliance and its board of directors so as to provide customer solutions that open up new opportunities in segment markets from automotive to professional to consumer. Xilinx silicon solutions bring the customization and flexibility needed to address the customer requirements, she said.

In fact, Xilinx was the first company to release an FPGA-based Ethernet AVB Endpoint last year. The reprogrammabilty of the FPGA allows customers to keep up with the latest changes in the specification and enables customization as the IEEE ratifies each phase of the standard. The core features time synchronization and the low-latency queuing and forwarding implementations per the 802.1 AS and 802.1 QAV specifications. It is available now. More information can be found at *http://www.xilinx.com/products/ipcenter/DO-DI-EAVB-EPT.htm*.

Chang notes that Xilinx and its IP partner network have served the wired and wireless network infrastructure, automotive and professional broadcast markets for many years. They offer a broad set of IP cores supporting multiple open and proprietary networks and protocols serving a broad range of vertical applications. "Xilinx is not going to dictate which protocol customers must use," said Chang. "Ultimately, our customers decide which networks and protocols are best. We support those customer preferences."

To learn more about the AVnu Alliance and how it relates to various vertical markets using AV equipment, read the following white papers at *www.AVnu.org*: "AVnu for Home/CE," "AVnu for Professional A/V" and "AVnu for Automotive."

Samplify Tips New Version of Prism Compression/Decompression Technology

Mixed-signal semiconductor and IP signal-compression vendor Samplify Systems Inc., a Xilinx Alliance partner, has released version 3.0 of its Prism decompression algorithm for implementation in FPGAs.

The Santa Clara, Calif., company announced itself to the world in 2008 and proceeded to garner a slew of trade publication awards and customer accolades for user-modifiable compression/decompression technology targeting a range of markets, including ISM and communications, that require the transfer of large amounts of data via high-speed I/O.

Samplify's technology is largely composed of two elements: a 16-channel A/D data converter compression IC that runs a 65-megasample decompression algorithm containing 16 channels in 12 bits. Users typically implement this algorithm in an FPGA.

"Users can set the technology to get anywhere from four-to-one [in lossless mode] or two-to-one [in near-lossless mode] compression, or even run it without compression and still see best-in-class results because it is so low power," said Allen Evans, vice president of marketing at Samplify. "The closer compression is to the analog domain, and the closer decompression is to the software domain, the better."
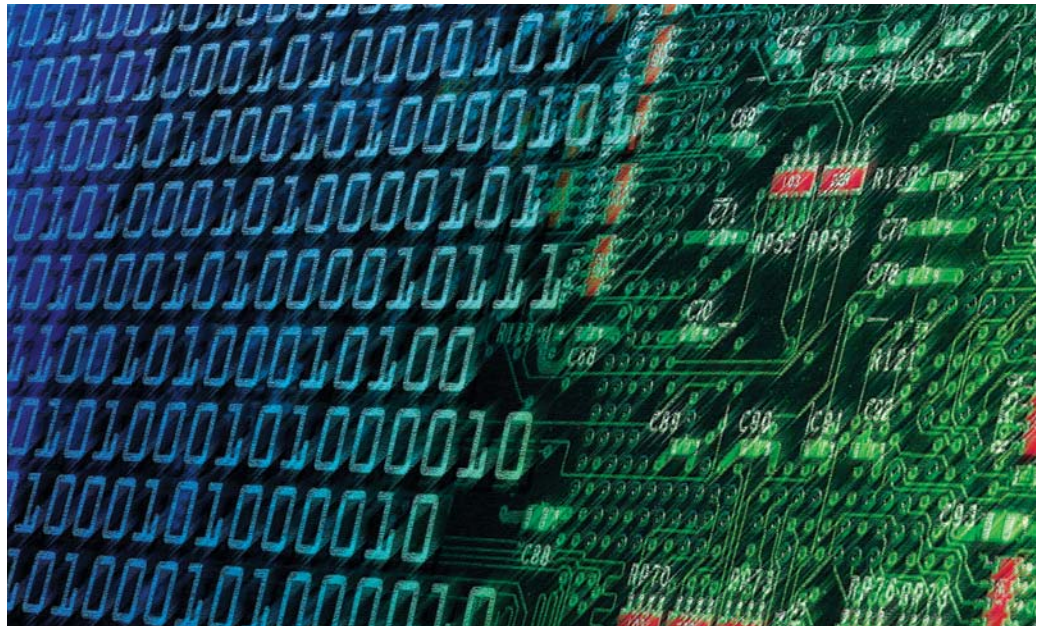
Evans says Samplify's technology is ideally suited for medical ultrasound equipment and 4G communications. "As the number of smart-antenna systems increases in 4G systems, so do the number of data converter channels," said Evans. "What's unique about this product is that it has signal compression built in on the back end. With each of these data converters now generating nearly 1 Gbit of data apiece, the compression solves not only the amount of data that needs to get out of the chip but the number of LVDS pairs to get it off the chip. At four-to-one compression, we need four LVDS pairs instead of 16."

To further refine the overall offering, Samplify has now released version 3.0 of the Prism algorithm. Where earlier versions were symmetrical, sharing the compression and decompression burden evenly between the IC (running compression) and the FPGA (handling decompression), Evans said that with version 3.0, the company has also optimized the algorithm to make it asymmetrical. That puts less of the decompression burden on the FPGA. As a result, he said, the algorithm still runs optimally in terms of performance and signal quality, but uses nearly half the FPGA logic resources required by the previous, symmetric version.

At the same time, the company also optimized the encoding format for software decoding, resulting in what it says is an improvement of 200 percent in decompression performance on x86-based CPUs or graphics processing units.

"I can say that the vast majority of our customers are implementing the decompression on FPGAs," said Evans.

With Prism 3.0, the company is also introducing a new compression mode called SignalTrak, which helps maintain signal quality for pulsed signals common in ultrasound, sonar and radar applications. In these cases, a signal pulse or burst contains both strong and weak reflections over very short intervals, on the order of 100 microns. Evans said SignalTrak is a better alternative to traditional feedback gain control techniques, which often adapt to signal levels too slowly to reduce the pulsed signal's dynamic range without losing signal quality (for weak signals) or overloading (for strong signals). To get around this problem, SignalTrak's redundancy remover automatically adapts to the signal, which means users no longer have to program any signal characteristics into Prism.

"It adapts the signal, tries to widen it and then uses a data encoder and various feedback loops to control how the compression operates," said Evans. "It has a fixed-rate mode called RateTrack that monitors packet size and adjusts the parameters up front to try to maintain that packet size." SignalTrak also has a fixed-quality mode, where it tries to maintain a specific quality range by adjusting the decompression ratio, he said.

Samplify Prism 3.0 is available now in netlist format for Xilinx and other FPGA devices. Prices start at $25,000 for a development license plus royalties, which are application dependent. For more information, visit *www.samplify.com*.

Opal Kelly Releases Virtex-5-Based USB Integration Module

USB-in-an-FPGA specialist Opal Kelly (Portland, Ore.) has released a USB Integration Module based on the Xilinx Virtex®-5 FPGA. The XEM5010 boasts more memory and I/O than previous offerings, along with an API for a PC interface.

Since its founding in 2004, Opal Kelly has specialized in fielding Xilinx-FPGA-based USB modules to help design teams develop USB-based systems, prototype IP and ASICs that use USB or simply integrate the modules into a growing number of applications, including retail and in-house test equipment, consumer systems, medical devices, communications gear, R&D and education.

According to founder and president Jake Janovetz, Opal Kelly has seen great success with these modules, especially its Spartan®-3 FPGA-based XEM3010 product.

"A lot of our customers are interfacing to LVDS signals, high-speed cameras and ultrasound equipment as a low-cost emulation environment for IP or ASIC development," said Janovetz. "Now they are running the FPGAs faster and harder, and need a bit more memory bandwidth, so we're offering a Virtex-5-based system."

The XEM5010 runs a Xilinx Virtex-5 XC5VLX50FPGA. With a footprint of 3.35 x 2.4 x 0.69 inches, the system features 256-mebibyte DDR2 SDRAM using two completely independent 128-Mbyte DDR2 SDRAMs (with a total memory bandwidth of 2.128 Gbytes/second or 17.024 Gbits/s), 32 Mbits of serial flash and a high-speed USB 2.0 interface for downloading and control.

Janovetz said the "secret sauce" for Opal Kelly customers is the company's FrontPanel software interface, which is an API for communication, configuration and interfacing to the PC. The FrontPanel API works with Microsoft Windows XP and Vista, Mac OS X and Linux, and with a variety of languages including C, C++, C#, Ruby, Python and Java. Opal Kelly includes the API software free with its modules.

Because the XEM5010 is based on the Virtex-5 device, designers can also program the system with the Xilinx WebPack tool suite, available at no charge from Xilinx. To facilitate overall system verification, Opal Kelly also offers the simulation models for the HDL in the XEM5010, along with DLL support for a variety of third-party tools, including The Mathworks' MATLAB® and National Instruments' LabVIEW.

Among the early customers of the XEM5010 is Samplify Systems. Its engineers and sales team use the module for a variety of tasks, including IP development, regression testing and sales demonstrations of Samplify's Prism compression/decompression technology (see related article, previous page).

Janovetz said that among the many end uses for the modules, more and more customers are tapping them for hardware-assisted verification. "We didn't set out to do emulation, but some of our customers were doing smaller design projects that couldn't justify the purchase of big, expensive emulation systems," he said. "And so they use our modules for regression testing and even simulation runs." As a result, Janovetz said, "we are looking into supporting it as a hardware-assisted verification tool."

The XEM5010 modules are available immediately, off the shelf, from the Opal Kelly Online Store. Prices start at $1,500, with volume discounts available. For more information on the XEM5010 and Opal Kelly's other offerings, visit *www.opalkelly.com*.

Opal Kelly's latest USB Integration Module, the memory-rich Virtex-5-based XEM5010, comes with an API for a PC interface.
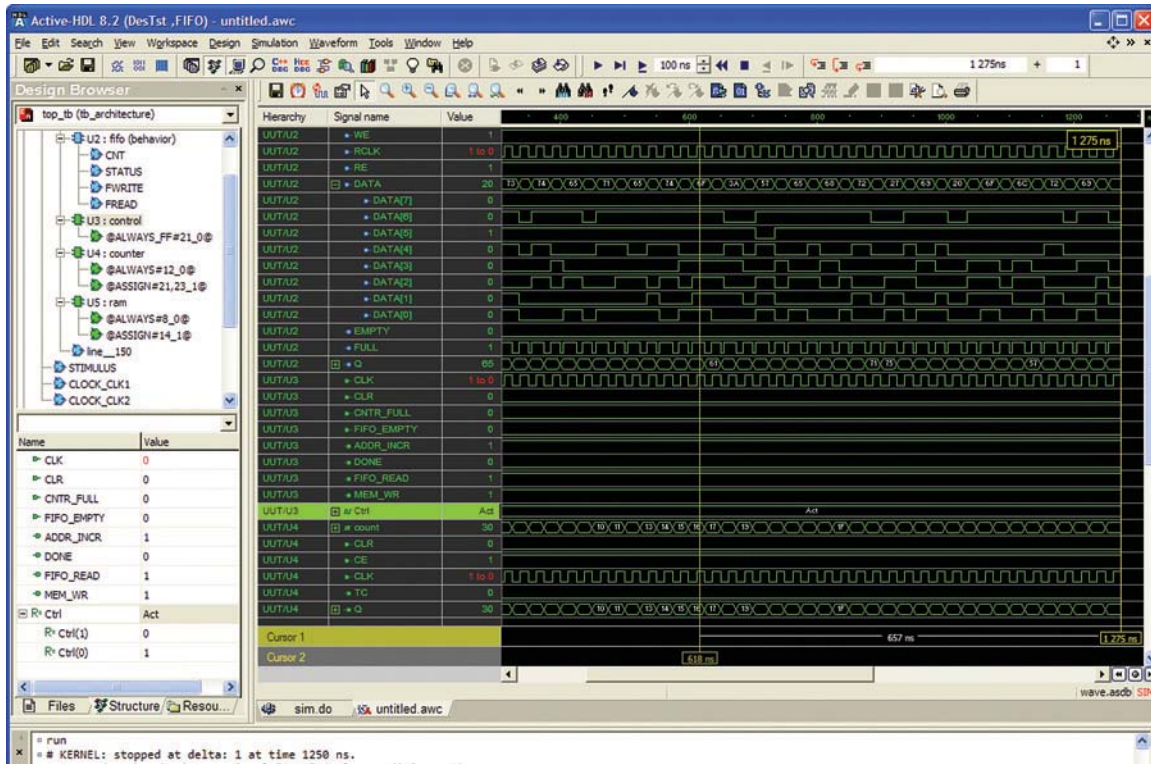
*Aldec's Active-HDL Designer Edition targets a niche between single-language vendor offerings
and more-expensive multilanguage simulators from third-party tool suppliers.*

Aldec's Multilanguage Simulator Takes Aim at Mainstream Users

A new version of the Active-HDL multilanguage simulator from Aldec Inc., priced at $1,995, claims to run simulation twice as fast as FPGA vendor-supplied RTL simulators.

According to Dave Rinehart, vice president of marketing at the Henderson, Nev., company, the Active-HDL Designer Edition targets a price point between FPGA vendor, single-language simulation offerings that typically sell for less than $1,000 and multilanguage simulators from third-party simulation tool vendors, which generally start at $6,000 for a single-node license.

It's fairly common today for FPGA designers to integrate a mix of Verilog and VHDL IP blocks into their designs. Indeed, some designers—specially those who until recently have focused on ASIC work—may even develop or use blocks in more-advanced languages such as SystemVerilog or SystemC.

While there are a lot of translators out there, it still behooves designers to use simulators that can natively handle a mix of the most common languages. Aldec's Active-HDL Designer Edition includes mixed-language simulation support for VHDL, Verilog and SystemVerilog design files. On top of that, the tool also includes VHDL and Verilog encrypted IP and Xilinx SecureIP support, and puts no performance limitations on FPGA design size.

The product is a pared-down version of Aldec's Active-HDL line, which runs many times faster than Active-HDL Designer Edition and packs several more features, including support for SystemC, code coverage, design rule checks, DSP modeling and verification. Rinehart said the company put a performance governor on the Active-HDL Designer Edition, opting to omit those high-end features to target the needs of mainstream design and verification engineers. Winnowing out some of the more-exotic features also eases the time and cost burdens on Aldec's support staff.

Rinehart believes that many designers will be so impressed with what they see in Designer Edition, they'll upgrade to the full version of the tool and reap the rewards of its advanced feature set.

At a price point of $1,995 for a node-locked license and $2,495 for a floating license, the simulator is certainly worth a test drive. To do so, visit *www.aldec.com/DesignerEdition.*

Innovative Integration Launches X5 G12 XMC I/O Module Family

Innovative Integration Inc. has released the Virtex-5 version of its XMC I/O module family, the X5 G12 series, featuring dual channels performing 1-Gsample/second 12-bit digitizing, 512 Mbytes of DDR, additional SRAM memory and an eight-lane PCI Express host interface.

"Typically, our customers are systems companies that will take our cards and integrate them into their larger systems," said Dan McLane, co-owner and head designer of the 30-employee Simi Valley, Calif., company. "They'll use our card, like the X5 G12, as a front end to do digitizing for applications like radar or as 3G/4G wireless receivers."

For the X5 G12 module family, McLane said, Innovative Integration uses the Xilinx Virtex-5 SX95T or LX155T plus 512 Mbytes of off-chip DDR2 in addition to 4 Mbytes of QDR-II memory. The combination allowed his team to implement a very high-performance DSP core in the FPGA, giving the X5 G12 modules real-time signal-processing rates exceeding 300 GMACs/s. The architecture includes tight integration with an eight-lane PCI Express interface that provides sustained transfer rates to the host of better than 1 Gbyte/s. As such, the module family is well suited for a broad number of digitizing-centric applications.

Innovative also does some very creative things with Xilinx cores. For example, McLane's group layered a packet system on top of the Xilinx PCI Express core to allow its cards to sustain the 1-Gbit/s rates of a workstation, enabling users to do high-speed data logging with the system. The modification also allowed the company to field an 8-Tbyte data logger. "We spend a lot of time on the hardware so that the hardware is doing a lot of the heavy lifting, and we also spend a lot of time on system integration features," said McLane. "That way our customers, who may not be all that familiar with logic design, can focus most of their time on optimizing at the system level—software and algorithms—instead of logic."

He points out that the Virtex-5 devices make it possible for users to customize (or, for a fee, have Innovative's team customize) the board at a hardware level using VHDL as well as at an algorithm level using The Mathworks' MATLAB, or at a software level using their language of choice. Innovative also offers its own tool suite, called FrameWork Logic, which takes some of the bite out of mod-ifying the logic in the cards and provides an organized framework for modifying algorithms and software. To a certain degree, McLane said, FrameWork drives the Xilinx ISE® Design Suite tools. But even users unfamiliar with logic design should be able to implement the modifications they want, if they are persistent. If not, Innovative is willing to offer a helping hand.

Along with its modules, Innovative also offers users its own FPGA-targeted IP as well as access to third-party cores to help the more HDL-savvy customers further customize the modules to their requirements. For example, for the X5 G12, Innovative offers customers a software-defined radio core that provides from 16 to 4,096 display data channels, essentially allowing users to turn the X5 G12 modules into receivers.

Innovative also provides with its modules software tools for host development as well as libraries and drivers for Windows and Linux.
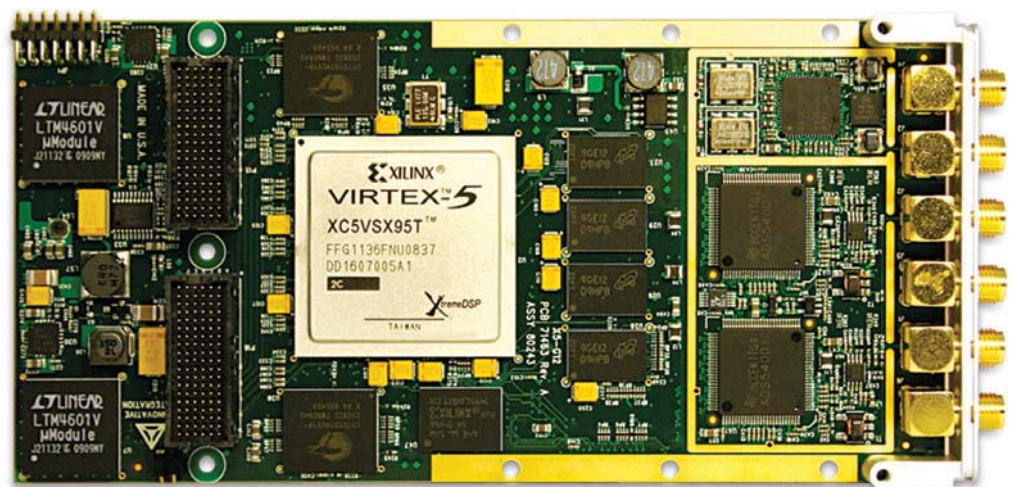
The Virtex-5 version of Innovative Integration's XMC I/O module family has two 1-Gsample/s channels and 12-bit digitizing, plenty of memory and an eight-lane PCI Express host interface.

App notes and example configurations of the module, including one for logging A/D samples to disk, are included as well.

And because Innovative offers a host of other cards developed since its founding in the mid-1980s, the company supplies carrier adapters to cool the card and the overall system it's going into. The X5 G12 card also plugs into the company's eInstrument Embedded PC, SBC-ComEx Single-Board Computer and Andale Data Loggers.

For more information on the X5 G12, visit *www.innovative-dsp.com/products.php?product=X5-G12.*

Xpress Yourself
in Our Caption Contest

If you have a yen to Xercise your funny bone, here's your opportunity. Starting with this issue, we invite readers to step up to a verbal challenge and submit an engineering- or technology-related caption for a strange, humorous or otherwise evocative illustration, cartoon or photograph. Here, for example, the Dali-like image of people and animals rubbernecking out of their castle windows might inspire a caption like "Maybe management took the advice to move engineers out of their tech silos a bit too literally."

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

The deadline for submitting entries is **December 11**. So, get writing!
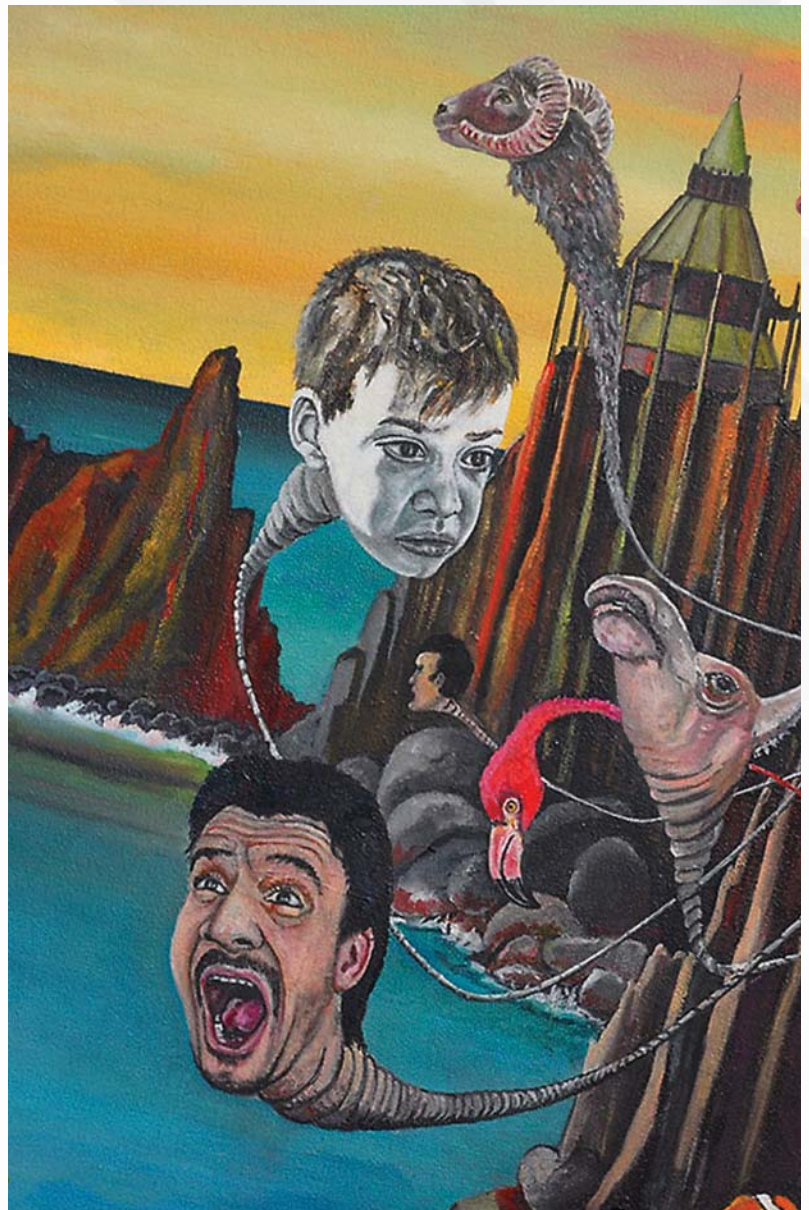
© Hemera

Here is a list of our A&D FPGAs from our product selection guide. To see our entire portfolio visit http://www.xilinx.com/publications/matrix/Product_Selection_Guide.pdf or to obtain a hard copy of the selection guide, contact your local Xilinx sales representatives.

Defense-Grade FPGAs

Virtex-5Q FPGAs

Part Number	XQ5VLX30T	XQ5VLX85	XQ5VLX110	XQ5VLX110T	XQ5VLX155T	XQ5VLX220T	XQ5VLX330T	XQ5VSX50T	XQ5VSX95T	XQ5VSX240T	XQ5VFX70T	XQ5VFX100T	XQ5VFX130T	XQ5VLX200T
Logic Resources — Slices[2]	4,800	12,960	17,280	17,280	24,320	34,560	51,840	8,160	14,720	37,440	11,200	16,000	20,480	30,720
Logic Cells[3]	30,720	82,944	110,592	110,592	155,648	221,184	331,776	52,224	94,208	239,616	71,680	102,400	131,072	196,608
CLB Flip-Flops	19,200	51,840	69,120	69,120	97,280	138,240	207,360	32,640	58,880	149,670	44,880	64,000	81,920	122,880
Memory Resources — Maximum Distributed RAM (Kbits)	320	840	1,120	1,120	1,640	2,280	3,420	780	1,520	4,200	820	1,240	1,580	2,280
Block RAM/FIFO w/ECC (36Kbits each)	36	96	128	148	212	212	324	132	244	516	148	228	298	456
Total Block RAM (Kbits)	1,296	3,456	4,608	5,328	7,632	7,632	11,664	4,752	8,784	18,576	5,328	8,208	10,728	16,416
Clock Resources — Digital Clock Manager (DCM)	4	12	12	12	12	12	12	12	12	12	12	12	12	12
Phase Locked Loop/PMCD	2	6	6	6	6	6	6	6	6	6	6	6	6	6
I/O Resources — Maximum Single-Ended Pins	360	560	800	680	680	680	960	480	640	960	640	680	840	960
Maximum Differential I/O Pairs	180	280	400	340	340	340	480	240	320	480	320	340	420	480
DSP48E Slices	32	48	64	64	128	128	192	288	640	1,056	128	256	320	384
Embedded Hard IP Resources — PowerPC® 440 Processor Blocks	–	–	–	–	–	–	–	–	–	–	1	2	2	2
PCI Express® Interface Blocks	1	–	–	1	1	1	1	1	1	1	3	3	3	4
10/100/1000 Ethernet MAC Blockset	4	–	–	4	4	4	4	4	4	4	4	4	6	8
RocketIO™ GTP Low-Power Transceivers	8	–	–	16	16	16	24	12	16	24	–	–	–	–
RocketIO™ GTX High-Speed Transceivers	–	–	–	–	–	–	–	–	–	–	16	16	20	24
Configuration — Configuration Memory (Mbits)	9.4	21.9	29.1	31.2	43.1	55.2	82.7	20	35.8	79.7	27.1	39.4	49.3	70.9
Miscellaneous — Speed Grades	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1, -2	-1	-1, -2	-1	-1	-1, -2	-1, -2	-1, -2	-1
Manufacturing Grades	–	–	–	–	–	–	–	–	–	–	–	–	–	–

Package availability (SelectIO pins (RocketIO transceivers))

Package[7]	Area	XQ5VLX30T	XQ5VLX85	XQ5VLX110	XQ5VLX110T	XQ5VLX155T	XQ5VLX220T	XQ5VLX330T	XQ5VSX50T	XQ5VSX95T	XQ5VSX240T	XQ5VFX70T	XQ5VFX100T	XQ5VFX130T	XQ5VLX200T
EF676	27 x 27 mm		440	440											
EF1153	35 x 35 mm			800											
FF323	19 x 19 mm	172 (4)													
EF665	27 x 27 mm								360 (8)			360 (8)			
EF1136	35 x 35 mm				640 (16)	640 (16)				640 (16)		640 (16)	640 (16)		
EF1738	42.5 x 42.5 mm						680 (16)	960 (24)					680 (16)	680 (16)	
FF1738	42.5 x 42.5 mm										960 (24)			840 (20)	960 (24)

Notes: 1. A single Virtex-5Q CLB comprises two slices, with each containing four 6-input LUTs and four Flip-Flops (twice the number found in a Virtex-4 slice), for a total of eight 6-LUTs and eight Flip-Flops per CLB; 2. Virtex-5 logic cell ratings reflect the increased logic capacity offered by the new 6-input LUT architecture; 3. Digitally Controlled Impedance (DCI) is available on I/Os of all devices; 4. I/O standards supported: HT, LVDS, LVDSEXT, RSDS, BLVDS, ULVDS, LVPECL, LVCMOS33, LVCMOS25, LVCMOS18, LVTTL, PCI33, PCI66, PCIX, GTL, GTL+, HSTL I (1.2V,1.5V,1.8V), HSTL II (1.2V,1.8V), HSTL III (1.5V,1.8V), HSTL IV (1.5V,1.8V), SSTL2 I, SSTL2 II, SSTL18 I, SSTL18 II; 5. One system monitor block included in all devices; 6. Available I/O for each device-package combination: number of SelectIO pins (number of RocketIO transceivers).

Manufacturing Grades
http://www.xilinx.com/products/milaero/rpt003.pdf

Grade	Description	Temperature
V	QPro Radiation Hardened QML Class V Military Ceramic	Tc = –55C to +125C
H	QPro Flip-Chip Radiation Tolerant Ceramic	Tj = –55C to +125C
B	SMD Radiation Tolerant and Non-RT SMD Military Ceramic	Tc = –55C to +125C
N	Military Plastic	Tj = –55C to +125C
M	Military Ceramic or Plastic	Tj = –55C to +125C (Plastic), Tj = –55C to +125C (Ceramic)
I	Industrial Plastic	Tj = –40C to +100C

XILINX AEROSPACE & DEFENSE SOLUTIONS

Defense-Grade FPGAs

Part Number	Virtex-4Q FPGAs								Virtex-II Pro XQ FPGAs		Virtex-II XQ FPGAs		
	XQ4VLX25	XQ4VLX40	XQ4VLX60	XQ4VLX100	XQ4VLX160	XQ4VSX55	XQ4VFX60	XQ4VFX100	XQ2VP40	XQ2VP70	XQ2V1000	XQ2V3000	XQ2V6000
Core Voltage	1.2V	1.2V	1.2V	1.2V	1.2V	1.2V	1.2V	1.2V	1.5V	1.5V	1.5V	1.5V	1.5V
Slices[1]	10,752	18,432	26,624	49,152	67,584	24,576	25,280	42,176	19,392	33,088	5,120	14,336	33,792
Logic Cells	24,192	41,472	59,904	110,592	152,064	55,296	56,880	94,896	44,632	74,448	11,520	32,256	76,032
CLB Flip-Flops	21,504	36,864	53,248	98,304	135,168	49,152	50,560	84,352	38,784	66,176	10,240	28,672	67,584
Maximum Distributed RAM (Kbits)	168	288	416	768	1,056	384	395	659	606	1,034	160	448	1,056
Block RAM/FIFO w/ECC (36Kbits each)	72	96	160	240	288	320	232	376	192	328	40	96	144
Total Block RAM (Kbits)	1,296	1,728	2,880	4,320	5,184	5,760	4,176	6,768	3,456	5,904	720	1,728	2,592
Digital Clock Manager (DCM)	8	8	8	12	12	8	12	12	8	8	8	12	12
Maximum Single-Ended I/Os	448	640	640	960	960	640	576	768	804	996	432	720	1,104
Maximum Differential I/O Pairs	224	320	320	480	480	320	228	384	396	492	216	360	552
Digitally Controlled Impedance	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES	YES
DSP Slices	48	64	64	96	96	512	128	160	–	–	–	–	–
18 x 18 Multipliers	–	–	–	–	–	–	–	–	192	328	40	96	144
RocketIO™ Transceivers	–	–	–	–	–	–	16	20	8 or 12	20	–	–	–
PowerPC® Processor Blocks	–	–	–	–	–	–	2	2	2	2	–	–	–
Speed Grades	-10	-10	-10	-10	-10	-10	-10	-10	-5	-5	-4	-4	-4
Configuration Memory (Mbits)	4.8	12.3	17.7	30.7	40.3	22.7	21.0	33.0	15.5	25.6	4.1	10.5	21.9
Manufacturing Grades	M	I, M	M	I	I	M	I, M	I	N	N	N	M, N, B	M
Packages	SF363, FF668	FF668	FF668, FF1148, EF668	FF1148	FF1148	FF1148	EF672, FFG1152*	FF1152	FF1152, FG676	FF1704	FG456, BG575	CG717, BG728	CF1144

Notes: 1. Each slice comprises two 4-input logic function generators (LUTs), two storage elements, wide-function multiplexers, and carry logic.

Manufacturing Grades
http://www.xilinx.com/products/milaero/rpt003.pdf

Grade	Description	Temperature
V	QPro Radiation Hardened QML Class V Military Ceramic	Tc = -55C to +125C
H	QPro Flip-Chip Radiation Tolerant Ceramic	Tj = -55C to +125C
B	SMD Radiation Tolerant and Non-RT SMD Military Ceramic	Tc = -55C to +125C
N	Military Plastic	Tj = -55C to +125C
M	Military Ceramic or Plastic	Tj = -55C to +125C (Plastic), Tj = -55C to +125C (Ceramic)
I	Industrial Plastic	Tj = -40C to +100C

Space-Grade QPro™ FPGAs

	Part Number	Virtex-4QV FPGAs				Virtex-II XQR FPGAs	Virtex XQR FPGAs	
		XQR4VLX200	XQR4VSX55	XQR4VFX60	XQR4VFX140	XQR2V3000	XQVR300	XQVR600
	Core Voltage	1.2V	1.2V	1.2V	1.2V	1.5V	2.5V	2.5V
Logic Resources	Slices[1]	89,088	24,576	25,280	63,168	14,336	3,072	6,912
	Logic Cells	200,448	55,296	56,880	142,128	32,256	6,912	15,552
	CLB Flip-Flops	178,176	49,152	50,560	126,336	28,672	6,144	13,824
Memory Resources	Maximum Distributed RAM (Kbits)	1,392	384	395	987	448	1,711	3,523
	Block RAM/FIFO w/ECC (18Kbits each)	336	320	232	552	96	–	–
	Total Block RAM (Kbits)	6,048	5,760	4,176	9,936	1,728	64	96
Clock Resources	Digital Clock Manager (DCM)	12	8	12	20	12	4	4
I/O Resources	Maximum Single-Ended I/Os	960	640	576	896	720	316	316
	Maximum Differential I/O Pairs	480	320	224	448	360	–	–
	Digitally Controlled Impedance	YES	YES	YES	YES	YES	–	–
	DSP Slices	96	512	128	192	–	–	–
Embedded Hard IP Resources	18 x 18 Multipliers	–	–	–	–	96	–	–
	10/100/100 Ethernet MAC Blockset	–	–	4	4	–	–	–
	PowerPC® Processor Blockss	–	–	2	2	–	–	–
Miscellaneous	Speed Grades	-10	-10	-10	-10	-4	-4	-4
	Configuration Memory (Mbits)	51.4	22.7	21.0	47.9	10.5	1.7	3.5
	Manufacturing Grades	V	V	V	V	M, V	M, V, B	M, V, B
	Total Ionizing Dose (krad)	300	300	300	300	200	100	100
	SEL Immunity (MeV-cm2/mg)	>125	>125	>125	>125	>160	>125	>125

Package[2]	Area	Available User I/Os						
		XQR4VLX200	XQR4VSX55	XQR4VFX60	XQR4VFX140	XQR2V3000	XQVR300	XQVR600
CGA Packages (CG): ceramic column grid array (1.27 mm ball spacing)								
CG717[3]	35 x 35 mm					516		
CFA Packages (CF): flip-chip ceramic column grid array (1.0 mm ball spacing)								
CF1144[4]	35 x 35 mm			576				
CF1140[5]	35 x 35 mm		640					
CF1509[6]	40 x 40 mm	960			768			
CQFP Packages (CB): ceramic brazed quad flat pack (0.025 inch lead spacing)								
CB228	1.55 x 1.55 in						162	162

Notes: 1. Each slice comprises two 4-input logic function generators (LUTs), two storage elements, wide-function multiplexers, and carry logic. 2. For information on DSCC SMD availability contact Xilinx. 3. The BG728 and CG717 packages are footprint / pin compatible. 4. The CF1144 and FF1152 packages are footprint / pin compatible. 5. The CF1140 and FF1148 packages are footprint / pin compatible. 6. For the XQR4VLX200, the CF1509 and FF1513 packages are footprint / pin compatible. For the XQR4VFX140, the CF1509 and the FF1517 are footprint / pin compatible.
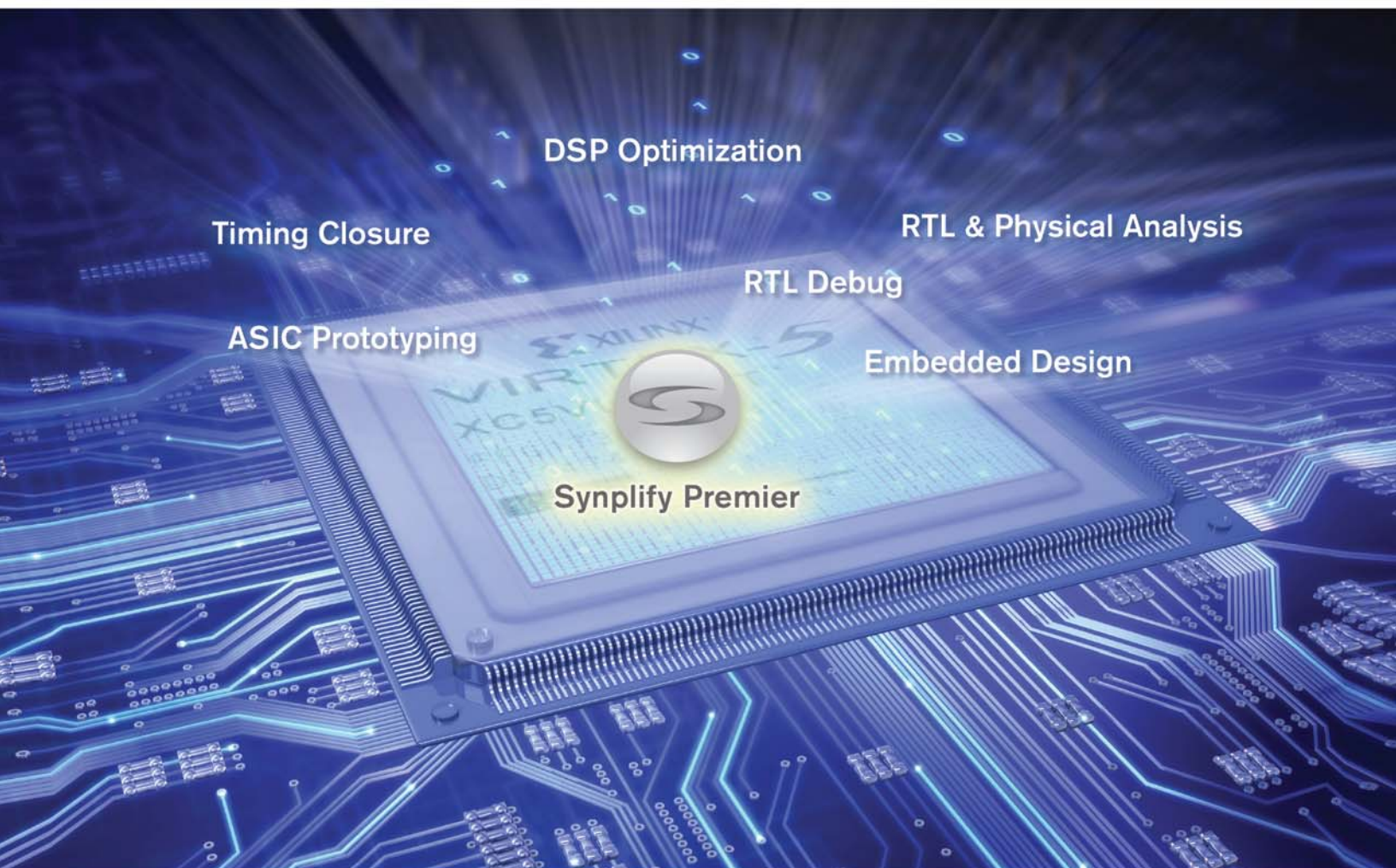
Manufacturing Grades
http://www.xilinx.com/products/milaero/rpt003.pdf

Grade	Description	Temperature
V	QPro Radiation Hardened QML Class V Military Ceramic	Tc = -55C to +125C
H	QPro Flip-Chip Radiation Tolerant Ceramic	Tj = -55C to +125C
B	SMD Radiation Tolerant and Non-RT SMD Military Ceramic	Tc = -55C to +125C
N	Military Plastic	Tj = -55C to +125C
M	Military Ceramic or Plastic	Tj = -55C to +125C (Plasttic), Tj = -55C to +125C (Ceramic)
I	Industrial Plastic	Tj = -40C to +100C

Part Number	Defense-Grade Configuration PROMs				Space-Grade QPro Radiation Tolerant Configuration PROMs	
	XQ1701L	XQ17V16	XQ18VQ04	XQF32P	XQR1701L	XQR17V16
Core Voltage[1]	3.3V	3.3V	3.3V	3.3V	3.3V	3.3V
Storage Bits	1M	16M	4M	32M	1M	16M
Manufacturing Grades	M, N	M, N	N	M	M, V	M, V
Total Ionizing Dose (krad)	–	–	–	–	50	50
Packages	CC44, VQ44	CC44, VQ44	VQ44	VQ48	CC44	CC44

Package[2]	Area
CC44	0.69 x 0.69 in
VQ44	12 x 12 mm
VQ48	20 x 20 mm

Notes:
1. Xilinx configuration PROMs have adjustable I/O voltages for compatibility with all Xilinx FPGAs.
2. The CC44 and PC44 packages are footprint/pin compatible. For information on DSCC qualification contact Xilinx.

Manufacturing Grades
http://www.xilinx.com/products/milaero/rpt003.pdf

Grade	Description	Temperature
V	QPro Radiation Hardened QML Class V Military Ceramic	Tc = -55C to +125C
H	QPro Flip-Chip Radiation Tolerant Ceramic	Tj = -55C to +125C
B	SMD Radiation Tolerant and Non-RT SMD Military Ceramic	Tc = -55C to +125C
N	Military Plastic	Tj = -55C to +125C
M	Military Ceramic or Plastic	Tj = -55C to +125C (Plastic), Tj = -55C to +125C (Ceramic)
I	Industrial Plastic	Tj = -40C to +100C

XILINX®

FASTER THAN THE SPEED OF CHANGE

The path to true innovation is never a straight line. Only Xilinx programmable silicon, software, IP and 3rd party support gives you the agility to stay ahead of the competition and adapt to changing market requirements, without slowing down. So you have the freedom to innovate without risk. Find out more about Xilinx Targeted Design Platforms at **www.xilinx.com**.

PN 2426