ISSUE 83, SECOND QUARTER 2013

Xcell journal

SOLUTIONS FOR A PROGRAMMABLE WORLD

All Eyes on Zynq SoC for Smarter Vision

How OpenCV and Vivado HLS Accelerate Embedded Vision Apps

How to Configure Your Zynq SoC Bare-Metal Solution

Using Xilinx's Power Estimator and Power Analyzer Tools

Vivado Design Suite 2013.1 Available for Download

Inside Xilinx's High-Level Synthesis Tool page 32

XILINX®
www.xilinx.com/xcell/

AVNET electronics marketing

www.zedboard.org

ZedBoard™

Avnet Electronics Marketing presents a new series of video-based SpeedWay Design Workshops™, featuring the new Xilinx Zynq™-7000 All Programmable SoC Architecture. This workshop series includes three online courses progressing from introductory, through running a Linux operating system on the Zynq-7000 SoC, and finally to the integration of the high-speed analog signal chain with digital signal processing from RF to baseband for wireless communications.

SPEEDWAY DESIGN WORKSHOPS™

www.zedboard.org/trainings-and-videos

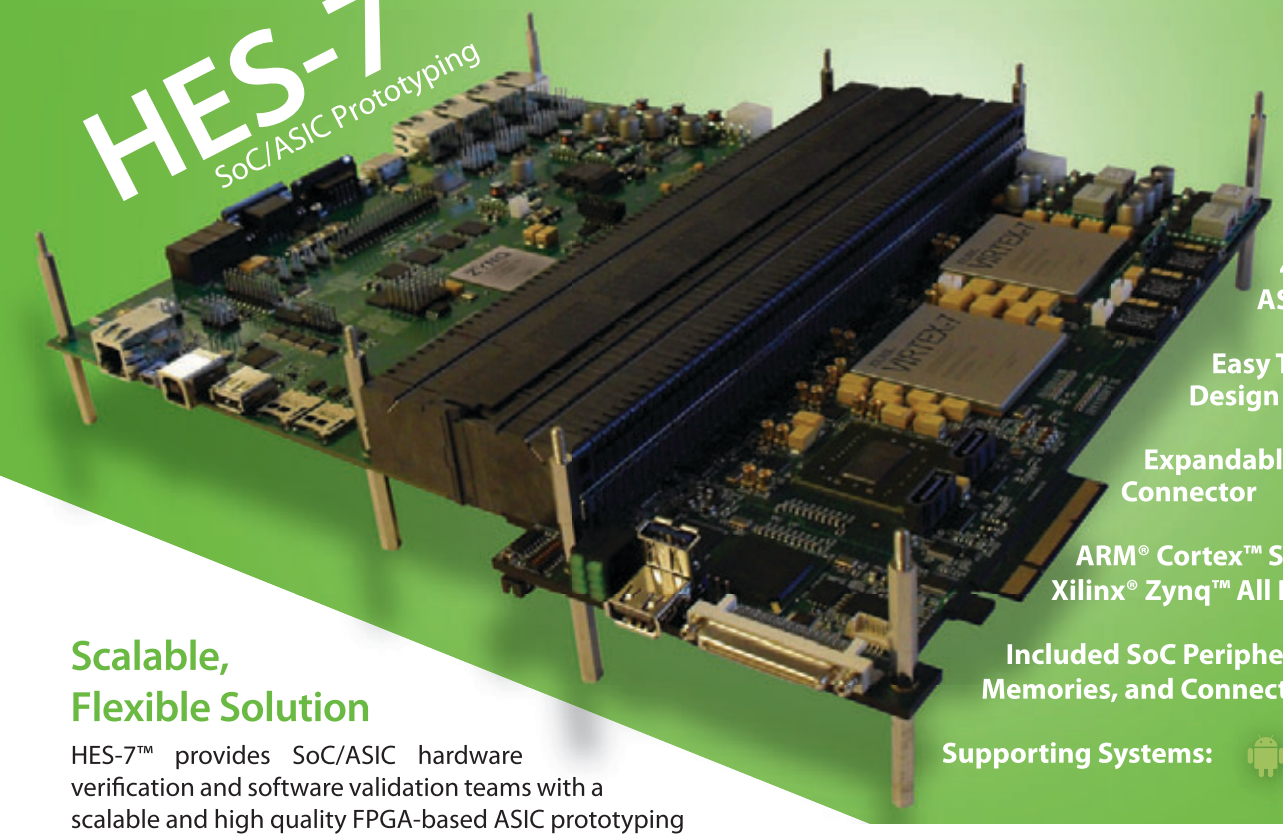
Accelerating Your Success™

MathWorks® ANALOG DEVICES XILINX ALL PROGRAMMABLE™

member XILINX ALLIANCE PROGRAM

HES-7™

SoC/ASIC Prototyping



4 to 96 million Scalable ASIC Gate Capacity

Easy To Use with Reduced Design Partitioning

Expandable via Non-Proprietary Connector

ARM® Cortex™ Support with Xilinx® Zynq™ All Programmable SoC

Included SoC Peripherals: Media Interfaces, Memories, and Connectors

Supporting Systems:

Scalable, Flexible Solution

HES-7™ provides SoC/ASIC hardware verification and software validation teams with a scalable and high quality FPGA-based ASIC prototyping solution backed with an industry leading 1-year limited warranty. Each HES-7 board with dual Xilinx® Virtex®-7 2000T has 4 million FPGA logic cells or up to 24 million ASIC gates of capacity, not including the DSP and memory resources.

The HES-7 prototyping solution was architected to allow for easy implementation and expansion. Using only one or two large FPGAs, rather than multiple low density FPGAs, HES-7 does not require as much labor-intensive partitioning or tool expense. Using a non-proprietary HES-7 backplane connector, HES-7 can easily expand prototype capacity up to 96 million ASIC gates and can include the expansion of daughter boards.

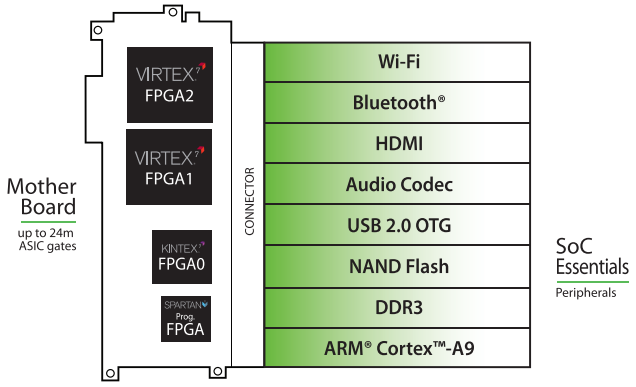
ARM Cortex Support

HES-7 supports ARM® dual-core Cortex™-A9 MPCore™ with Xilinx® Zynq™-7000 All Programmable SoC, allowing designers to leverage the serial processing capabilities of the Cortex-A9 processor for applications that require intensive computations and operating systems with the parallel processing capabilities of HES-7 ASIC prototyping platform to create applications across a diverse range of markets including: Video, Communications, Control Systems and Bridging.

The 4 to 96 million ASIC gate scalable capacity of HES-7, coupled with open-source Linux, Android, and FreeRTOS solutions available from Xilinx, delivers a powerful verification platform for HW/SW design teams.

HES-7™

SoC/ASIC Prototyping Platform



Essential SoC Peripherals

HES-7 provides SoC Peripherals, allowing designers the ability to interface real-world stimuli to the design-under-test (DUT). Users can utilize gigabit Ethernet transceivers to develop networking applications, WLAN 802.11 b/g/n and Bluetooth® v2.1 to develop wireless systems, or High Performance HDMI Transmitter to develop home entertainment products. Memories and connectors provide additional data storage for read/write capability of today's popular memory interfaces, and the ability to connect external hardware with the HES-7.

Aldec, Inc.
Corporate - **N. America**
Ph +1.702.990.4400
sales@aldec.com

www.aldec.com

Israel
Ph. +972.072.2288316
sales-il@aldec.com

China
Ph. +86.21.6875.2030
info@aldec.com.cn

Europe
Ph. +44.1295. 20.1240
sales-eu@aldec.com

India
Ph. +91.80.3255.1030
sales-in@aldec.com

Japan
Ph. +81.3.5312.1791
sales-jp@aldec.com

Taiwan
Ph. +886.3.6587712
sales-tw@aldec.com

ALDEC®
THE DESIGN VERIFICATION COMPANY

Xcell journal

PUBLISHER Mike Santarini
mike.santarini@xilinx.com
408-626-5981

EDITOR Jacqueline Damian

ART DIRECTOR Scott Blair

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

ADVERTISING SALES Dan Teie
1-800-493-5551
xcelladsales@aol.com

INTERNATIONAL Melissa Zhang, Asia Pacific
melissa.zhang@xilinx.com

Christelle Moraga, Europe/
Middle East/Africa
christelle.moraga@xilinx.com

Tomoko Suto, Japan
tomoko@xilinx.com

REPRINT ORDERS 1-800-493-5551

Xcell PUBLICATIONS
www.xilinx.com/xcell/

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

Work Smarter With Xilinx All Programmable Solutions

If you've checked out Xilinx's home page lately, you've probably noticed that we have plunged full force into a new campaign touting "Smarter" systems. Inspired by what our customers have already been able to create with Xilinx devices, we are delivering tools today to help you create the Smarter technologies that will shape tomorrow.

In the last two months, Xilinx has rolled out our Smarter Networks initiative, and now, as reflected in the cover story of this issue, our Smarter Vision program. In both cases, Xilinx not only offers devices that allow customers to create smarter systems, but the IP, tools and design environments to support those chips.

In this issue, DSP specialist Chris Dick writes about how the wireless carriers are rapidly getting smarter in finding ways to increase the profitability of their next-generation networks. Gone are the days when carriers would just install a greater number of faster and more powerful basestations to increase bandwidth. Today they are adding intelligence and looking toward self-organizing network architectures to dynamically shift coverage to where it is needed most—working smarter, not harder, and raising profits while doing so.

On another front, my cover story discusses a subject I find very fascinating: Smarter Vision. As you'll read, the world of embedded vision has been growing by leaps and bounds, to the point where vision systems are becoming ubiquitous. Today you can find them in everything from gaming consoles to the car you drive to factories and hospitals. Even produce distributors are benefiting from embedded vision technology, which traditionally has paired FPGAs with standalone processors. But we at Xilinx think that with a silicon platform as powerful as the Zynq™-7000 All Programmable SoC, you'll be able to create much more capable embedded vision systems—Smarter Vision systems—using one chip instead of two or three.

To facilitate that effort, Xilinx is going the extra mile by offering the industry the broadest software development environment support and proliferating high-level synthesis technology through the Vivado™ HLS tool in our Vivado Design Suite. System architects can design vision algorithms in C/C++ and use Vivado HLS to create RTL versions. In doing so, they can see, for example, whether certain algorithms or parts of an algorithm would run better in the Zynq SoC's processor system or in its FPGA fabric.

In early April, Xilinx added even more Smarter Vision automation to the Vivado Design Suite in release 2013.1. Designers now can use a tool called IP Integrator to deliver the industry's fastest time to integration. With this release of Vivado, we also announced our OpenCV cores library to help vision architects design more productively. In the initial release, Xilinx has provided more than 30 commonly used algorithms from the OpenCV library and created RTL versions that you can add to your design to get a jump on developing Smarter Vision systems. You can even use them to evaluate hardware platforms by running the functions on the Zynq SoC vs. other ARM+DSP and ARM+GPU-based hardware platforms. Jose Alvarez and Fernando Vallina discuss the OpenCV advantage on page 24.

Finally, I'm happy to announce that an old friend from the trade press, Steve Leibson, the former editor-in-chief of the *Microprocessor Report* and *EDN*, recently joined Xilinx and has penned a comprehensive backgrounder on smarter wired and wireless networks and smarter data centers. Steve's backgrounder goes into great depth on the trends in these markets and how Xilinx is helping customers address them with our Smarter Networking technology. You can download a PDF from *http://www.xilinx.com/publications/prod_mktg/smarter-networks-backgrounder.pdf*.

I hope this issue and associated resources will inspire you to take the next step and try out the Zynq SoC, if you haven't already.

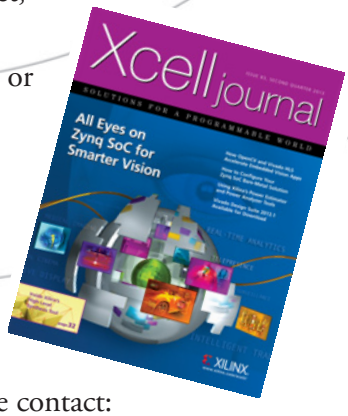
Mike Santarini
Publisher

Get Published

Would you like to write for Xcell Publications? It's easier than you think.

Interested in adding "published author" to your resume and achieving a greater level of credibility and recognition in your peer community? Consider submitting an article for global publication in the highly respected, award-winning *Xcell Journal*.

The *Xcell* team regularly guides new and experienced authors from idea development to published article with an editorial process that includes planning, copy editing, graphics development, and page layout. Our author guidelines and article template provide ample direction to keep you on track and focused on how best to present your chosen topic, be it a new product, research breakthrough, or inventive solution to a common design challenge.



Our design staff can even help you turn artistic concepts into effective graphics or redraw graphics that need a professional polish.

We ensure the highest standards of technical accuracy by communicating with you throughout the editorial process and allowing you to review your article to make any necessary tweaks.

Submit final draft articles for publication in our Web-based *Xcell* Online or our digital and print *Xcell Journal*. We will assign an editor and a graphics artist to work with you to make your papers clear, professional, and effective.

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

Mike Santarini
Publisher, Xcell Publications
xcell@xilinx.com

Xcell Publications

www.xilinx.com/xcell/

XILINX®

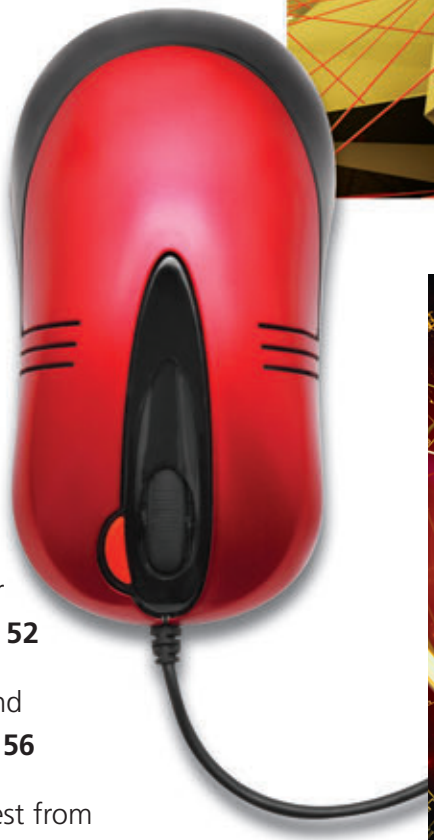
VIEWPOINTS

XCELLENCE BY DESIGN APPLICATION FEATURES

Cover Story

All Eyes on Zynq SoC for Smarter Vision

THE XILINX XPERIENCE FEATURES



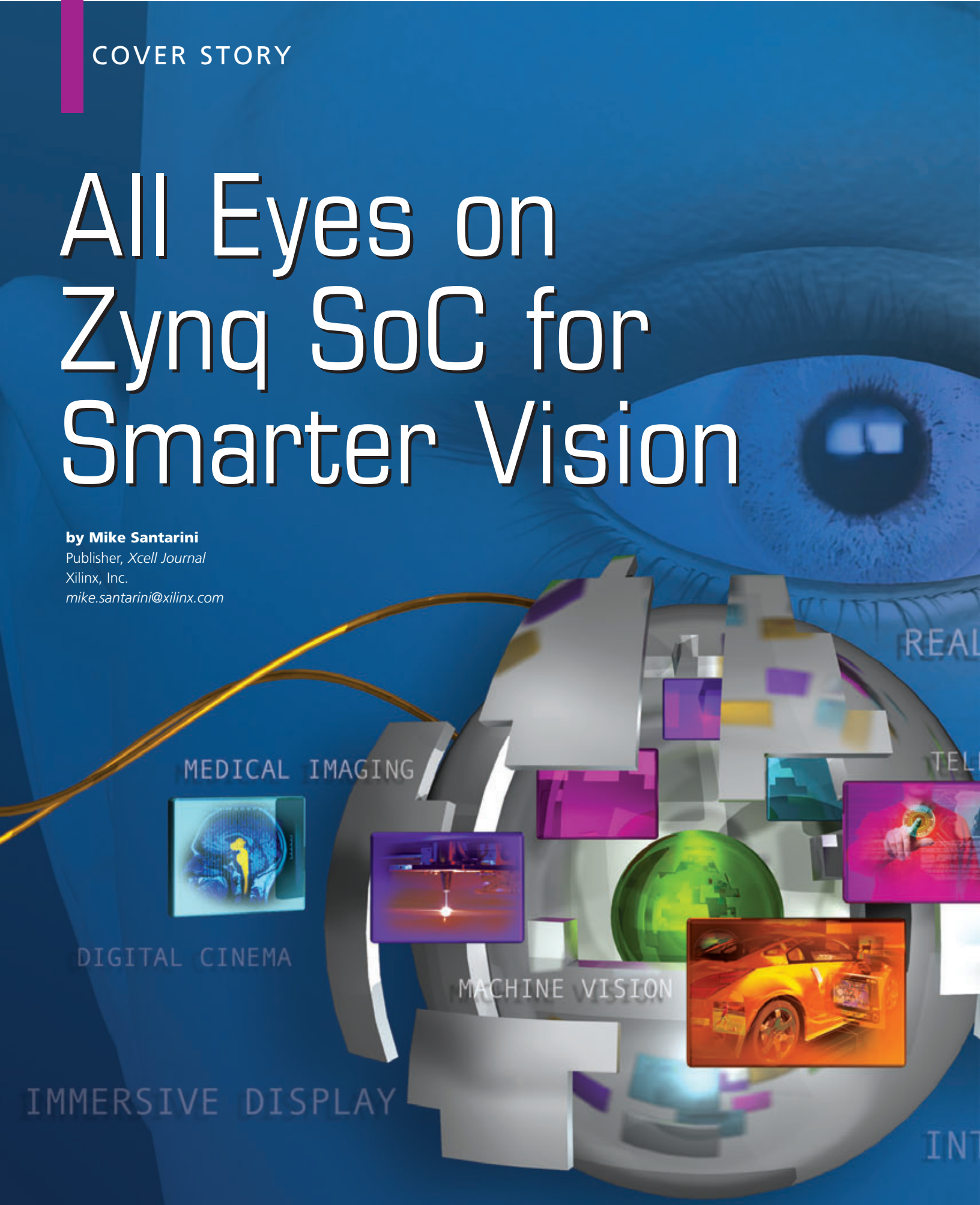
XTRA READING

APEX® AWARDS FOR PUBLICATION EXCELLENCE
Excellence in Magazine & Journal Writing
2010, 2011

APEX® AWARDS FOR PUBLICATION EXCELLENCE
Excellence in Magazine & Journal Design and Layout
2010, 2011, 2012

All Eyes on Zynq SoC for Smarter Vision

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



The Zynq All Programmable SoC, in tandem with new Xilinx tools and IP, forms the foundation for the next generation of embedded vision products.

If you have seen a demonstration of Audi's Automated Parking technology in which the car autonomously finds a parking spot and parks itself without a driver—or if you have played an Xbox 360 game with its Kinect controller or even just bitten into a flawless piece of fruit from your local grocery store—then you can count yourself as an eyewitness to the dawning of the era of smarter vision systems. All manner of products, from the most sophisticated electronic systems down to the humble apple, are affected by smarter vision technologies. And while today's systems are impressive enough, some experts predict that in 10 years' time, a vast majority of electronics systems—from automotive to factory automation, medical, as well as surveillance, consumer, aerospace and defense—will include smarter vision technologies with even more remarkable capabilities.

As smarter vision systems increase in complexity, we'll very likely become passengers in autonomous automobiles flowing in networked highways. Medical equipment such as Intuitive Surgical's amazing robotic-assisted surgical system will advance even further and may enable surgeons to perform procedures from remote locations. Television and telepresence will reach new levels of immersion and interactivity, while the content on screens in theaters, homes and stores will cater to each individual consumer's interests, even our moods.

Xilinx® All Programmable solutions for Smarter Vision are at the forefront of this revolution. With the Zynq™-7000 All Programmable SoC—the first device to marry an ARM® dual-core Cortex™-A9 MPCore™, programmable logic and key peripherals on a single chip—as the foundation, Xilinx has fielded a supporting infrastructure of tools and IP that will play a pivotal role in enabling the development and faster delivery of these innovations in vision. The supporting infrastructure includes Vivado™ HLS (high-level synthesis), the new IP Integrator tools, OpenCV (computer vision) libraries, SmartCORE™ IP and specialized development kits.

"Through Xilinx's All Programmable Smarter Vision technologies, we are enabling our customers to pioneer the next generation of smarter vision systems," said Steve Glaser, senior vice president of corporate strategy and marketing at Xilinx. "Over the last decade, customers have leveraged our FPGAs to speed up functions that wouldn't run fast enough in the processors they were using in their systems. With the Zynq-7000 All Programmable SoC, the processor and FPGA logic are on the same chip, which means developers now have a silicon platform ideally suited for smarter vision applications."

In support of the device, said Glaser, "We've complemented the Zynq-7000 All Programmable SoC with a robust development environment consisting of Vivado HLS, new IP Integrator tools, OpenCV libraries, SmartCORE IP and development kits. With these Smarter Vision technologies, our customers will get a jump on their next design and be able to achieve new levels of efficiency, lower system power, increase system performance and drastically reduce the bill of materials—enriching and even saving lives while increasing profitability as these innovations roll out at an ever faster pace."

Cutting-edge vision systems enhance and analyze images, but also trigger actions based on those analyses. As such, the need for compute power has spiked dramatically.

FROM DUMB CAMERAS TO SMARTER VISION

At the root of Smarter Vision systems is embedded vision. As defined by the rapidly growing industry group the Embedded Vision Alliance (*www.embedded-vision.com*), embedded vision is the merging of two technologies: embedded systems (any electronic system other than a computer that uses a processor) and computer vision (also sometimes referred to as machine vision).

Jeff Bier, founder of the Embedded Vision Alliance and CEO of consulting firm BDTI, said embedded vision technology has had a tremendous impact on several industries as the discipline has evolved beyond motorized pan-tilt-zoom analog camera-based systems. "We have all been living in the digital age for some time now, and we have seen embedded vision rapidly evolve from early digital systems that excelled in compressing, storing or enhancing the appearance of what cameras are looking at into today's smarter embedded vision systems that are now able to know what they are looking at," said Bier.

Cutting-edge embedded vision systems not only enhance and analyze images, but also trigger actions based on those analyses. As such, the amount of processing and compute power, and the sophistication of the algorithms, have spiked dramatically. A case in point is the rapidly advancing market of surveillance.

Twenty years ago, surveillance systems vendors were in a race to provide the best lenses enhanced by mechanical systems that performed autofocus and tilting for a clearer and wider field of view. These systems were essentially analog video cameras connected via coaxial cables to analog monitors, coupled with video-recording devices monitored by security guards. The clarity, reliability and thus effectiveness of these systems were only as good as the quality of the optics and lenses, and the diligence of the security guards in monitoring what the cameras displayed.

With embedded vision technology, surveillance equipment companies began to use lower-cost cameras based on digital technology. This digital processing gave their systems extraordinary features that outclassed and underpriced analog and lens-based security systems. Fisheye lenses and embedded processing systems with various vision-centric algorithms dramatically enhanced the image the camera was producing. Techniques that correct for lighting conditions, improve focus, enhance color and digitally zoom in on areas of interest also eliminated the need for mechanical motor control to perform pan, tilt and zoom, improving system reliability. Digital signal processing has enabled video resolution of 1080p and higher.

But a clearer image that can be manipulated through digital signal processing was just the beginning. With considerably more advanced pixel processing, surveillance system manufacturers began to create more sophisticated embedded vision systems that performed analytics in real time on the high-quality images their digital systems were capturing. The earliest of these embedded vision systems had the capacity to detect particular colors, shapes and movement. This capability rapidly advanced to algorithms that detect whether something has crossed a

virtual fence in a camera's field of view; determine if the object in the image is in fact a human; and, through links to databases, even identify individuals.

SUSPICIOUS BEHAVIOR

The most advanced surveillance systems include analytics that track individuals of interest as they move through the field of view of the security network, even as they leave the field of view of one camera, move into a blind spot and then enter into the field of view of another camera in the surveillance network. Vision designers have programmed some of these systems to even detect unusual or suspicious movements. "Analytics is the biggest trend in the surveillance market today," said Mark Timmons, system architect in Xilinx's Industrial, Scientific and Medical (ISM) group. "It can account for human error and even take away the need for diligent human viewing and decision making. As you can imagine, surveillance in crowded environments such as train stations and sporting events can become extremely difficult, so having analytics that can spot dangerous overcrowding conditions or track individuals displaying suspicious behavior, perhaps radical movements, is very advantageous."

To further enhance this analysis and increase the effectiveness of these systems, surveillance and many other markets leveraging smarter vision are increasingly using "fusion" architectures that combine cameras with other sensing technologies such as thermal vision, radar, sonar and LIDAR (Light/Laser Detection and Ranging). In this way, the systems can enable night vision; detect thermal/heat signatures; or pick up objects not captured by or visible to the camera alone. This capability drastically reduces false detections and in turn allows for much more precise analytics. Needless to say, the added complexity of fusing the technologies and then analyzing that data requires ever more analytic-processing horsepower.

Timmons said that another megatrend in this market is products that perform all these forms of complex analysis "at the edge" of a surveillance system network—that is, within each camera—rather than having each camera transmit its data to a central mainframe system, which then performs a more refined analysis from these multiple feeds. Localized analytics adds resilience to the overall security system, makes each point in the system much faster and more accurate in detection, and thus can warn security operators sooner if indeed a camera spots a valid threat.

Localized analytics means that each unit not only requires greater processing horsepower to enhance and analyze what it is seeing, but must also be compact and yet incorporate highly integrated electronics. And because each unit must be able to communicate reliably with the rest of the network, it must also integrate electronic communication capabilities, adding further compute complexity. Increasingly, these surveillance units are connected via a wireless network as part of a larger surveillance system. And increasingly, these surveillance systems are becoming part of larger enterprise networks or even larger, global networks, like the U.S. military's Global Information Grid (see cover story, *Xcell Journal* issue 69, *www.xilinx.com/publications/ archives/xcell/Xcell69.pdf*).

This high degree of sophistication is being employed in the military-and-defense market in everything from foot soldier helmets to defense satellites networked to central command centers. What's perhaps more remarkable is how fast smarter vision technology is moving into other markets to enhance quality of life and safety.

SMARTER VISION FOR THE PERFECT APPLE

Take, for example, an apple. Ever wonder how an apple makes it to your grocery store in such good condition? Giulio Corradi, an architect in Xilinx's ISM group, said that food companies are using ever-smarter vision systems in food inspection lines to, for example, sort the bad apples from the good ones. Corradi said first-generation embedded vision systems deployed on high-speed food inspection lines typically used a camera or perhaps several cameras to spot surface defects in apples or other produce. If the embedded vision system spotted an unusual color, the apple would be marked/sorted for further inspection or thrown away.

BENEATH THE SKIN

But what happens if, at some point before that, the fruit was dropped but the damage wasn't visible? "In some cases, damage that resulted from a drop may not be easily spotted by a camera, let alone by the human eye," said Corradi. "The damage may actually be in the flesh of the apple. So some smarter vision systems fuse an infrared sensor with the cameras to detect the damage beneath the surface of the apple's skin. Finding a bruised fruit triggers a mechanical sorter to pull the apple off the line before it gets packed for the grocery store." If the damaged apple had passed by without the smarter fusion vision system, the damage would likely become apparent by the time it was displayed on the grocery store shelves; the fruit would probably have to be thrown away. One rotten apple can, of course, spoil the bunch.

Analytics can also help a food company determine if the bruised apple is in good enough condition to divert to a new line, in which another smarter vision system can tell if it is suitable for some other purpose—to make applesauce, dried fruit or, if it is too far gone, best suited for composting.

Factory floors are another site for smarter vision, Corradi said. A growing number use robotic-assisted technologies or completely automated robotic lines that manufacturers can retool for different tasks. The traditional safety cages around the robots are too restrictive (or too small) to accommodate the range of movement required to manufacture changing product lines.

Virtual barriers can reduce factory accidents, but 'why not also have virtual barriers in amusement parks, or at our homes, around swimming pools or on cars?'

So to protect workers while not restricting the range of motion of automated factory lines, companies are employing smarter vision to create safety systems. Cameras and lasers erect "virtual fences or barriers" that audibly warn workers (and safety monitor personnel) if someone is getting too close to the factory line given the product being manufactured. Some installations include a multiphase virtual barrier system that will send an audible warning as someone crosses an outer barrier, and shut down the entire line automatically if the individual crosses a second barrier that is closer to the robot, preventing injury. Bier of the Embedded Vision Alliance notes that this type of virtual barrier technology has wide applicability. "It can have a tremendous impact in reducing the number of accidents in factories, but why not also have virtual barriers in amusement parks, or at our homes around swimming pools or on cars?" said Bier. "I think we'll see a lot more virtual barrier systems in our daily lives very soon."

SMARTER VISION FOR BETTER DRIVING

Automotive is another market that is fully embracing smarter vision to create a less stressful and safer driving experience. Paul Zoratti, a systems architect within Xilinx Automotive, said that advanced driver assistance systems (ADAS) are all about using remote sensing technologies, including smarter vision, to assist drivers (see cover story, *Xcell Journal* issue 66, *www.xilinx.com/publications/archives/xcell/Xcell66. pdf*).

Each year over the past decade, automakers have unveiled ever-more impressive DA features in their luxury lines, while also making a growing number of driver assistance features available in their sport and standard product lines. Many of these features—such as blind-spot detection, lane change assist, pedestrian and sign detection—send drivers a warning if they sense a potentially dangerous situation. Recent offerings from auto manufacturers include even more advanced systems such as automatic emergency braking and lane keeping, which not only monitor the vehicle environment for potential problems but assist the driver in taking corrective actions to avoid accidents or decrease their severity.

Zoratti said that some new-model cars today are outfitted with four cameras—located on the sides, front, and rear of the vehicle—that offer a continuous 360-degree view of the vehicle's surroundings. While the first-generation surround-view systems are using those cameras to provide an image to the driver, future systems will bundle additional DA features. Using the same four cameras and image-processing analytics, these next-generation systems will simultaneously generate a bird's eye view of the vehicle and also warn of potential danger such as the presence of a pedestrian. Furthermore, while the vehicle is traveling at higher speeds, the automobile will use the cameras on the side and rear of the vehicle for blind-spot detection, lane change assistance and lane departure warning. Adding another, forward-looking camera behind the windshield will support traffic sign recognition and forward-collision warning features. Finally, when the driver reaches his or her destination and activates automated parking, the system will employ those same cameras along with other sensors to help the car semi-autonomously maneuver into a parking spot.

Handling all these tasks in real time requires a tremendous amount of processing power that is well-suited for parallel hardware computation, Zoratti said. That's why many early DA systems paired standalone microprocessors with FPGAs, with the FPGA handling most of the parallel computations and microprocessors performing the serial decision making.

The cost pressures in automotive are driving the analytics to be performed in a central compute hub, rather than at each camera as is the case in other markets, such as surveillance. In this way, carmakers can minimize the cost of each camera sensor and, ultimately, the cost of the entire system. That means the processing platform in the central unit needs to deliver very high performance and bandwidth to support the simultaneous processing of four, five or even six real-time video inputs.

A SMARTER VISION FOR LONGER LIVES

Another area where smarter vision is making a dramatic difference is in the medical electronics industry, which uses smarter vision technology in a wide range of medical imaging systems, from endoscopes and imaging scanners (CT, MRI, etc.) to robotic-surgical systems such as Intuitive Surgical's Da Vinci, detailed in *Xcell Journal* issue 77 (see *www.xilinx.com/publications/archives/xcell/Xcell77.pdf*).

Da Vinci's sophisticated 3D vision system allows surgeons to guide robotic surgical instruments with extreme precision, fluidity and tactile sensitivity to perform a number of delicate, intri-

cate surgical procedures. With every generation of system, surgeons are able to perform a greater number and variety of surgeries, helping to ensure better patient outcomes and faster recovery times. The degree of technological sophistication to control and coordinate these procedures is remarkable and is heavily reliant on the combined horsepower of processing and logic. Each generation of newer technology will thus benefit from greater integration between processor and logic.

Ben Runyan, director of the broadcast and consumer segment marketing at Xilinx, said that companies developing telepresence technologies are seeking ways to create a more immersive experience for users. "The goal is to make users feel like they are in the same room when in fact they could be on the other side of the globe," said Runyan. "To do this requires state-of-the-art cameras and display technologies, which require advanced image processing. As these technologies become more advanced and deliver a more immersive experience, it will make collaboration easier and make companies more productive while cutting down the need, and thus expense, to travel."

decade. Today, after five years in development, Xilinx is delivering a holistic solution that will help developers of smarter vision applications to quickly deliver the next generation of innovations.

For more than a decade, embedded vision designers have leveraged the programmability, parallel computing and fast I/O capabilities of Xilinx FPGAs in a vast number of embedded vision systems. Traditionally, designers have used FPGAs to speed up functions that were slowing down the main processor in their systems, or were using the FPGA to run parallel computing tasks that processors simply could not perform. Now, with the Zynq-7000 All Programmable SoC, embedded vision developers have a fully programmable device that is ideally suited for developing the next generation of smarter vision applications.

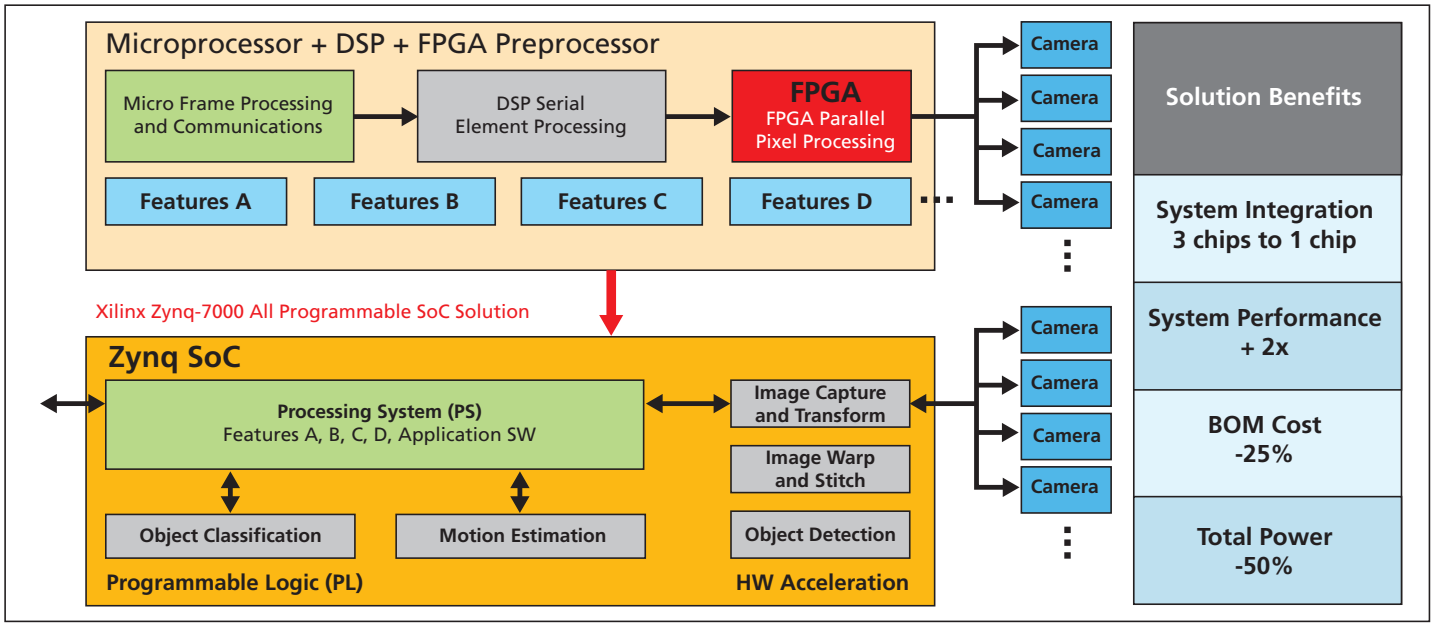


Figure 1 – Zynq All Programmable SoC vs. multichip, multiple-camera systems in driver assistance applications

A SMARTER VISION FOR AN IMMERSIVE EXPERIENCE

Smarter vision is also making great strides in keeping us connected. If you work in a modern office building, chances are your company has at least one conference room with an advanced telepresence conferencing system that not only allows you to talk to others around the world, but also lets you see them as though they were there in person. These videoconferencing systems are increasing in sophistication to the point that they can sense who at a table or conference is speaking, and automatically turn and zoom in on that person, displaying him or her in ever-higher-quality immersive video.

XILINX: ALL-PROGRAMMABLE FOR SMARTER VISION

To enable smarter vision to progress on all fronts rapidly and to reach new markets requires an extremely flexible processing platform, a rich set of resources and a viable ecosystem dedicated to smarter vision. Xilinx devices have played a key role in helping companies innovate these vision systems over the last

"Smarter vision can be implemented in separate processors and FPGAs communicating on the same board, but what the Zynq SoC delivers is a level of integration that the electronics industry didn't have before," said Jose Alvarez, engineering director of video technology at Xilinx. "Now, instead of

interchanging information between the main intelligent processor and FPGA logic at board speeds, we can do it at silicon speeds through 3,000 high-performance connections between the processor and logic on the same chip."

Figure 1 reveals the benefits of the Zynq SoC over a traditional multicamera, multichip architecture in the creation of a multifeature automotive driver assistance system. Using one set of cameras connected to one Zynq SoC, the Xilinx architecture (bottom left in the graphic) can enable feature bundles such as blind-spot detection, 360-degree surround view, lane departure warning and pedestrian detection. In comparison, existing multifeature DA systems require multiple chips and multiple cameras, which complicates integration, adversely affects performance and system power consumption, and leads to higher BOM costs.

A few silicon vendors offer ASSPs that pair ARM processors with DSPs or with GPUs, but those devices tend to be too rigid or provide insufficient compute performance for many of today's smarter vision applications. Often, solutions based on these devices require the addition of standalone FPGAs to fix these deficiencies.

PROGRAMMABILITY AND PERFORMANCE

The Zynq SoC's programmability and performance provide key advantages over GPU- and DSP-centric SoCs. The ARM processing system is software programmable; the FPGA logic is programmable via HDLs or C++; and even the I/O is fully programmable. As a result, customers can create extremely high-performance smarter vision systems suited to their specific applications, and differentiate their systems from those offered by competitors.

Figure 2 details a generic signal flow of a smarter vision system and shows how the Zynq SoC stacks up against ARM-plus-DSP and ARM-plus-GPU-based ASSPs.

The first signal-processing block in the flow (shown in green) is the input that connects the device to a camera sensor. In the Zynq SoC, developers can accommodate a wide range of I/O signals to conform to whatever camera connectivity their customers require. The next signal-processing block performs pixel-level processing or video processing (depending on whether the application is for image processing or display). The next block performs analytics on the image, a compute-intensive process that often requires parallel computing best implemented in FPGA logic. The subsequent three blocks (in red) are where the processing system derives metadata results from the analytics, creates a graphic representation of the result (the graphics step) and encodes results for transmission.

In the Zynq SoC, the processing subsystem and FPGA logic work together. When compression is required, the appropriate codec can be readily implemented in FPGA logic. Then, in the final signal-processing block (labeled "Output"), the Zynq SoC's programmable I/O allows developers to target a vast number of communication protocols and video transport standards, whether they be proprietary, market specific or industry-standard IP protocols. In comparison, in both DSP- and GPU-centric SoCs, developers run the risk of developing algorithms that require performance not achievable with the DSP or GPU sections of these ASSPs. They will often have to make up for this deficiency by adding a standalone FPGA to their systems.

While the Zynq SoC is clearly the best silicon choice for smarter vision

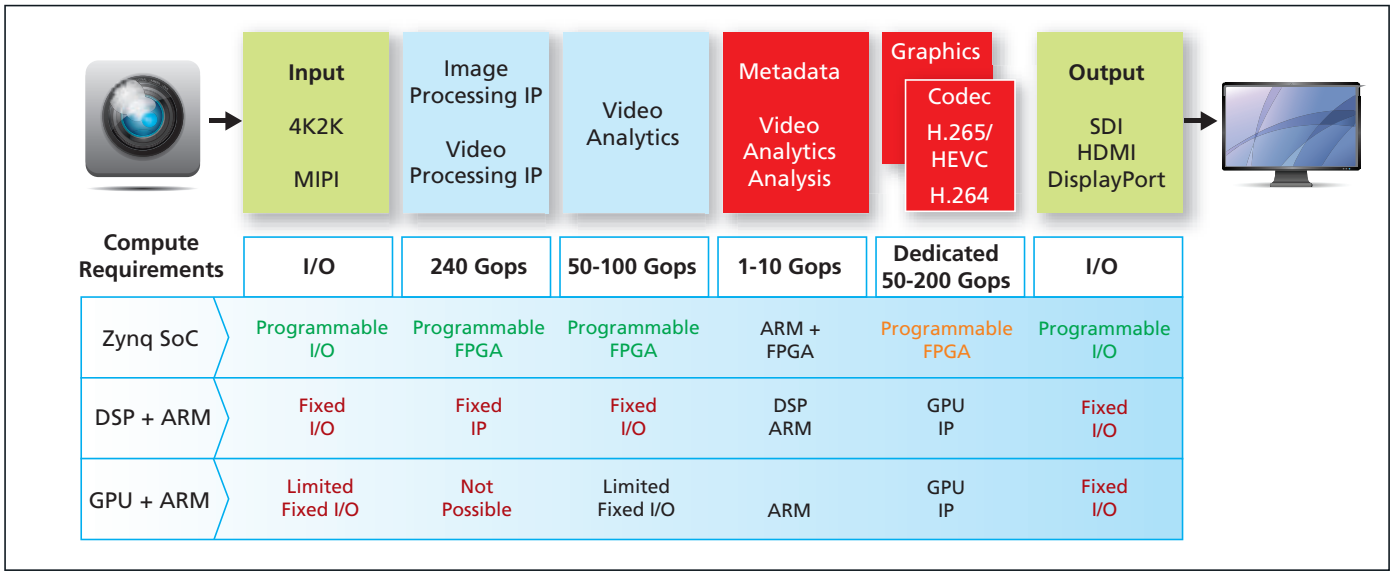


Figure 2 – Generic video- and image-processing system flow

systems, Xilinx realized early in the device's development that programming needed to be streamlined, especially for designers who are more accustomed to C and C++ based development of vision algorithms. To this end, in June 2012 Xilinx delivered to customers a state-of-the-art software environment called the Vivado Design Suite that includes, among other technologies, best-in-class high-level synthesis technology that the company gained in its January 2011 acquisition of AutoESL. Vivado HLS is particularly well-suited to embedded vision applications. If, for example, vision developers using the Zynq SoC have created an algorithm in C or C++ that doesn't run fast enough or is overburdening the processing system, they can send their C algorithms to Vivado HLS and synthesize the algorithms into Verilog or VHDL to run in the FPGA logic on the device. This frees up the processing subsystem on the Zynq SoC to handle tasks it is better suited to run, and thus speeds up the overall system performance.

OPENCV LIBRARY

Xilinx has also rounded out its Smarter Vision technology offering by releasing its OpenCV (computer vision) library. OpenCV is an industry-standard, open-source library of algorithms from OpenCV.org that embedded vision developers use to quickly create vision systems. Embedded vision developers across the world actively contribute new algorithms to the library, which now contains more than 2,500 algorithms written in C, C++, Java and Python (see OpenCV story, page 24). Algorithms in the library range in complexity from simple functions such as image filters to more advanced functions for analytics such as motion detection.

Alvarez said that these OpenCV algorithms target implementation in just about any commercial microprocessor and DSP. Because the Zynq SoC uses an ARM processing system, users can implement these algorithms, written in C++, in its processor portion.

Thanks to Vivado HLS, said Alvarez, users can also take these algorithms written in C or C++, modify function calls from OpenCV to HLS and then, using Vivado HLS, synthesize or compile the algorithms into RTL code optimized for implementation in the logic portion of the Zynq-7000 SoC. Having OpenCV in the Vivado environment allows smarter vision architects to easily compare and contrast whether a given algorithm in their design will run most optimally in the processor or FPGA logic portion of the Zynq-7000 All Programmable SoC. With the release of Xilinx's Open Source library, Xilinx has essentially given customers a head start. Using Vivado HLS, Xilinx has already compiled more than 30 of the most used embedded vision algorithms from the OpenCV library. Customers can quickly make processor vs. logic trade-offs at the systems level and run them immediately in the Zynq-7000 All Programmable SoC to derive the optimal system for their given application.

Xilinx and its Alliance members will actively migrate more functions from the OpenCV library on an ongoing basis, making them available to Xilinx's user base quarterly. Because developers can run OpenCV libraries on just about any commercial processor, vision designers will be able to compare and even benchmark the performance of algorithms running on various silicon devices.

As part of its Smarter Vision initiative, Xilinx has also created an intellectual property (IP) suite called SmartCORE IP, addressing smarter vision requirements from across the many market segments that will design smarter vision into their next-generation products. Customers can implement cores from the SmartCORE IP suite and algorithms from the OpenCV library into their designs quickly using Xilinx's newly introduced IP Integrator tool. The new tool is a modern plug-and-play IP environment that allows users to work in schematics or, if they prefer, a command-line environment.

TARGETED PLATFORM AWARE

Alvarez said that since the Vivado Design Suite's inception, Xilinx architected the suite to be device aware, so as to take full advantage of each device's capabilities. Alvarez said that thanks to IP Integrator, the Vivado Design suite is not only device aware but now targeted platform aware as well—supporting all Zynq SoC and 7 series FPGA boards and kits. Being target platform aware means that the Vivado Design Suite will configure and apply board-specific design rule checks, which ensures rapid bring-up of working systems.

For example, when a designer selects the Xilinx Zynq-7000 SoC Video and Imaging Kit, and instantiates a Zynq SoC processing system within IP Integrator, Vivado Design Suite preconfigures the processing system with the correct peripherals, drivers and memory map to support the board. Embedded design teams can now more rapidly identify, reuse and integrate both software and hardware IP, targeting the dual-core ARM processing system and high-performance FPGA logic.

Users specify the interface between the processing system and their logic with a series of dialog boxes. IP Integrator then automatically generates RTL and optimizes it for performance or area. Then users can add their own custom logic or use the Vivado IP catalog to complete their designs.

It's remarkable to see what smarter vision systems Xilinx customers have created to date with Xilinx FPGAs. The advent of the Zynq-7000 All Programmable SoC and the powerful Smarter Vision environment guarantees that the next crop of products will be even more amazing.

Xilinx All Programmable Devices Enable Smarter Wireless Networks

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

With the meteoric growth of mobile IP, carriers need flexible silicon and design tools to build the high-performance heterogeneous network architectures of the future.

Wireless carriers are urgently looking for ways to capitalize on the explosive growth in mobile Internet usage. But instead of simply deploying more and faster equipment, the carriers are seeking smarter ways to use their networks. They are actively developing novel architectures such as self-organizing networks to deliver superior quality of service to customers and to maximize profitability.

Xilinx® is helping wireless-network companies pioneer these new architectures and deliver them to market with 28-nanometer All Programmable devices and Smarter Wireless Networking solutions and supporting infrastructure, including SmartCORE™ IP, the Vivado™ Design Suite and services expertise.

EXPLOSIVE GROWTH OF MOBILE IP

The last four years have seen explosive growth in mobile Internet Protocol usage. The International Telecommunications Union's ICT statistics show that in the period from 2000 to 2010, the number of mobile cellular subscriptions increased by eightfold, a rate that far exceeds the growth in the number of Internet users over the same period. The dominance of mobility stands in stark contrast to trends for fixed telephone line subscriptions, which continue to decline. As of 2010 there were four times more mobile subscriptions than fixed-line. The introduction of smartphone and tablet technology has fundamentally changed the way we do business, spend our leisure time and interact with our family and friends.

There is a symbiotic connection between the capabilities and the business models enabled by smartphones and tablets, and the cellular network that is the anchor point to the Internet: Each feeds off the other. As smartphones and tablets become more capable, people figure out new ways to exploit their capabilities, stressing network capacity. Marketplace dynamics engage, and network operators respond with network upgrades. In turn, mobile equipment companies introduce new devices that consume the new capacity, and so the cycle continues. Today, analysts paint a picture of a network in 2020 needing to support a capacity that is 1,000 times greater than today's.

In the four years since the smartphone moved into the mainstream, we have seen screen pixel density double from 163 pixels per square inch to 326 PPI today. At the same time, storage has doubled from 32 to 64 gigabytes, and rear-facing camera capability has progressed from 3-megapixel photos and VGA video to 8-megapixel cameras supporting 1024p HD video at 30 frames/second. Mobile device manufacturers now offer front-facing cameras with applications like FaceTime, which enable mobile business

It took 16 years to reach 1 billion smartphones worldwide; it is estimated that it will take only three years for the next billion smartphone users to come onboard. The capacity challenge will rise exponentially.

videoconferencing and permit the mobile user, say, on overseas business travel to stay connected with home.

From 2009 to now, there has been an astonishing six generations of smartphone. In the three short years since the introduction of the first commercially successful tablet, we have seen five generations of this technology, taking us from 720p video capability only two years ago to 1024p HD video on a typical rear-facing camera today. With in excess of 1 billion smartphones on the planet, and with more than half of the population of North America owning a smartphone, feeding all of these pixels with media-rich content and delivering image and video data from the mobile back to the cloud, the upshot has been a network capacity and latency challenge. And while it took 16 years to reach 1 billion smartphones worldwide, it is estimated that it will take only three years for the next billion smartphone users to come onboard. The capacity challenge will rise exponentially.

As people have invented new services and applications that increasingly capable smartphones and tablets can support, the network has had to evolve and supply increased capacity and coverage, new levels of quality-of-experience and reduced latency to enhance the mobile experience for anything from videoconferencing and video streaming to media content downloading to the local device. Before boarding a long-haul flight, for example, you'll want to download the latest edition of *Xcell Journal* or *IEEE Communications Magazine*, a newspaper or two, other magazines, technical reports from the office or multiple videos to view later, at a more convenient time. The growth in online and interactive gaming and financial transactions has also made demands on capacity and latency.

Advanced socio-technical systems, such as Facebook for example, together with increasingly capable mobile devices have become a central part of daily life for billions of people worldwide. Social media users are actively using mobile applications more than ever before. A recent study by media agency Ruder Finn pointed out that 91 percent of the mobile subscribers engage in social computing applications, compared with 79 percent of desktop users. People in the United States, on the average, spend 2.7 hours per day on mobile devices, of which 45 percent post comments, 43 percent connect with friends, 40 percent share content with others and 38 percent share photos on social networking websites, making it an increasingly favorable platform for socializing.

One consequence of the changing nature of how people exploit mobility is that data traffic is now significantly greater than voice traffic on the wireless network. Some industry reports predict that the global mobile data growth will scale by approximately a factor of 40 in the coming years, going from 90 petabytes per month in 2009 to in excess of 3.5 exabytes per month in 2014.

STRUGGLING TO SCALE WITH BANDWIDTH DEMANDS

In response to these demands, and to the invention of new mobile applications and associated business models, wireless networks have made dramatic capacity improvements. Over the past three decades, since the conception of the 2G GSM system, the industry has achieved bit-rate improvements in excess of three orders of magnitude, corresponding to an order-of-magnitude throughput improvement for each of the past three decades. Clearly, the increasing smartphone and tablet computer population has resulted in substantial teletraffic growth, and it is anticipated that this growth will continue until 2020.

Since the first radio communications at the end of the 19th century, mankind has been on a path of developing technologies enabling people to communicate with anyone, anywhere and at any time using a range of media-rich services. However, despite significant advances in mobile user equipment and the wireless and wired networks themselves, the provisioning of a true telepresence capability is not yet within our grasp, as anyone who has experienced dropped connections, capacity and coverage problems, slow content download times, terminal battery-life issues and network latency disrupting the quality of the media experience would know. In addition to human-to-human and IP network type communications, we are also seeing a new class of service evolve, machine-to-machine (or machine-type communication) that will place further demands on network capabilities.

THE NETWORK EVOLVES

Standards organizations, such as 3GPP, along with industry conglomerates such as the Next Generation Network Mobility Alliance and research laboratories in industry and academia have been racing to define and drive mobile broadband 4G technologies such as LTE (long-term evolution) and the Evolved Packet Core to address all of these varied challenges. In the past four years, we have gone

from the standardization of the 3GPP Release 8 specification in late 2008—the first 3GPP standard to define next-generation capability making use of multicarrier OFDMA and SC-FDMA technology—through the first major evolution of LTE with the LTE-Advanced Release 10 specification, ratified in 2011. While LTE and LTE-A systems are still being deployed, the Release 12 working groups are planning the requirements and technologies for LTE-B.

One of the key challenges the wireless industry faces is extending capacity. Claude E. Shannon's famous channel-capacity formula—outlined in a seminal 1948 paper ("A Mathematical Theory of Communication") published when the comms pioneer worked at Bell Labs—informs us that bandwidth efficiency can only improve at the cost of reduced power efficiency. The law shows that capacity is a linear function of bandwidth but a logarithmic function of signal-to-noise ratio. This has led the research community to invent increasingly effective ways to use bandwidth within a complex set of constraints involving regulatory requirements, considerations for spectrum licensing, interoperability, multiple access, reliability, quality of service and spectral flexibility to the level of each individual piece of user equipment.

These constraints are some of the motivating principles that led to the multicarrier modulation scheme and evolved packet core employed in LTE, and they continue to be the motivators for features such as carrier aggregation in LTE-Advanced. But relying on channel bandwidth alone to address the capacity challenge is not enough.

Further capacity improvements have been made possible through the use of spatial-division multiplexing (SDM) MIMO, where (and in contrast to the logarithmic capacity formula) a linear increase in capacity is theoretically possible with respect to the number of antennas when the number of transmit and receive antennas is the same. While SDM can be employed to increase the throughput for a single user, spatial-division multiple access (SDMA) configurations can maximize the number of supported users in a cell by sharing the total system throughput among the users. MIMO transceivers, together with distributed or virtual MIMO, have permitted power-efficient or "green" solutions to the capacity problem. As the industry moves to LTE-B, it is likely that 3D MIMO, or full-dimension MIMO, will be embraced by both a future version of the 3GPP standard and by OEMs building macro basestations (Figure 1). Operators will deploy heterogeneous networks along with beamforming small cells that offer vertical and horizontal electrical-beam tilt to address coverage holes and minimize intercell interference.

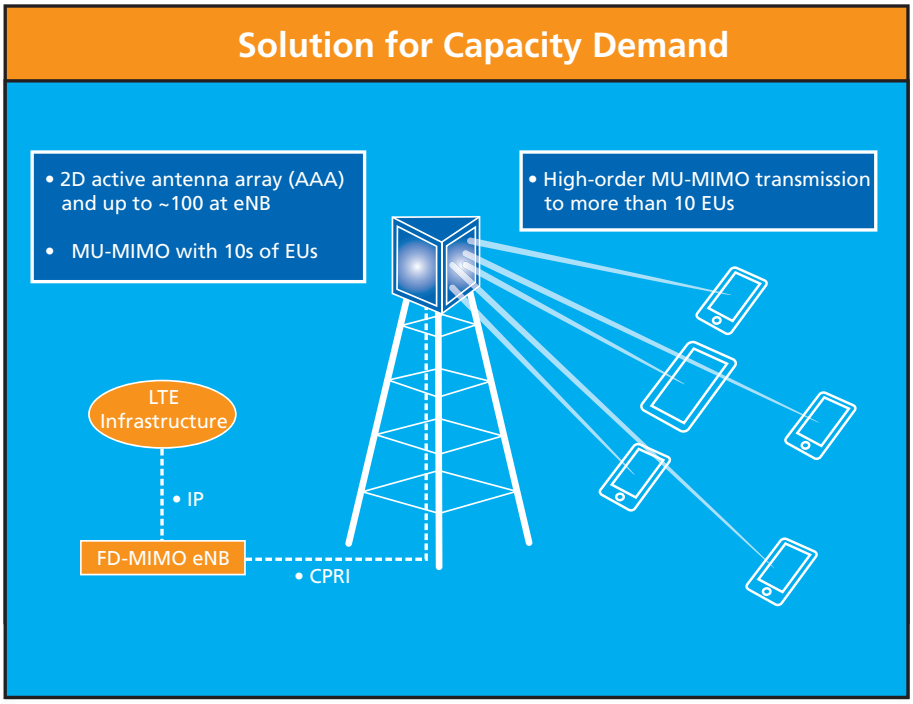


Figure 1 – Industrial and academic research communities, and the 3GPP PHY working group, are analyzing the potential use of massive, or 3D, MIMO for future-generation cellular access. Massive MIMO scales the antenna system by an order of magnitude compared with the systems of today, deploying hundreds of antennas.

However, augmenting carrier aggregation with advanced multi-antenna techniques is still not enough. It's difficult to realize the theoretical capacity improvements promised by spatial multiplexing in practice, and the reality is that implementation losses—for example through imprecise channel estimates—limit the gains promised by theory.

We are already seeing, and will continue to see at an accelerated pace, the deployment of heterogeneous networks comprising a macrocell augmented with relays and with a small-cell underlay to address issues of both capacity and coverage in dense urban-canyon environments, to extend in-building coverage and for use in environments where regulatory considerations prevent the installation of a macrocell. Heterogeneous networks (aka HetNets) and cell densification bring yet further challenges to the network designer in the form of backhaul, intercell interference and managing handover between cells or even to carrier-grade Wi-Fi.

Sixty years of research following Shannon's pioneering paper have led to advanced coding schemes—turbo

codes, for example—along with multi-antenna techniques, modulation systems, communication protocols and digital-IF processing that network companies can combine to realize systems that operate arbitrarily close to channel capacity. But there is still a long way to go to address the ferocious appetite for even more capacity, robust communications and reduced end-to-end latency. In addition to more flexible and sophisticated or smarter (if you will) use of spectrum by means of OFDMA and smart-antenna configurations, concepts such as enhanced intercell interference cancellation are coming online to assist with handover and maintaining the resilience of a connected device.

The rich set of concepts captured under the umbrella of self-organizing networks (SONs) is one of the "must-have" technologies on the LTE road map. There are multiple facets to SON, including automatic neighbor relations, load balancing, traffic growth and capacity management. The use of self-optimizing, self-healing technology promises energy savings, improved coverage and LTE parameter optimization, an example being the random-access channel, or RACH, optimization.

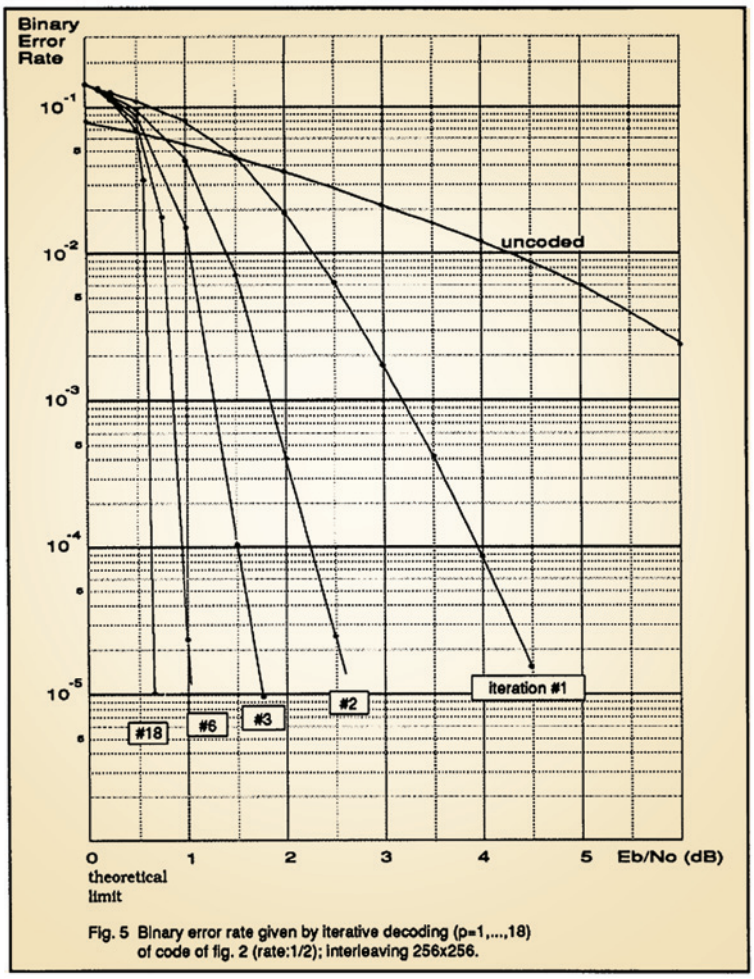


Figure 2 – The first-ever published waterfall plot for convolutional turbo codes, published by Berrou et al. in their famous 1993 paper. Turbo codes revolutionized the coding system employed in 3G and 4G wireless systems.

Another key aspect of SON is making the network smarter, with the end goal of minimizing OPEX-related drive tests. To improve their networks, operators often deploy engineers in the field to collect radio measurements, to discover problems such as coverage holes in the network and to determine whether there's a need to tune basestation or network parameters. However, such conventional drive tests are expensive, and the measurements they collect can only give a limited view, at one point in time, of the entire network. Member companies of the Next Generation Mobile Networks and 3GPP organization are actively discussing ways to address these challenges. The Release 10 specification introduced the concept of minimization of drive tests and is an important contribution to the area of SON that will assist with node insertion in the network and enabling automatic parameter tuning over the life of the node and network.

Yet another consideration for network design is the operating cost, with one of the significant contributors to OPEX, and actually CAPEX, being the RF platform, which includes the power amplifier, or amplifiers for multi-antenna systems. The green considerations for the RF electronics are multifaceted. Deployment of a nonlinear power amplifier is desirable from the vantage point of cost, but the non-constant-envelope OFDMA modulation scheme used in the LTE RAN downlink has the undesirable consequence of spectral regrowth when processed by a low-cost nonlinear amplifier. These highly undesirable out-of-band spectral emissions, which are delivered to neighboring spectrum, compromise nearby communication systems and breach regulatory requirements.

One solution is to use a large backoff and exercise only the linear portion of the amplifier transfer function. However, since we are also interested in the efficiency of the power amplifier or the effectiveness of the amplifier to convert DC power to the RF carrier, it's best to operate the amplifier with only a small output backoff, since the amplifier is most efficient at this bias point. Biasing the amplifier to minimize the backoff means moving closer to the compressive part of

The approach to solving for capacity, latency, coverage, quality of service, OPEX and CAPEX is going to be different for each mobile-network operator and cellular-infrastructure manufacturer.

the transfer function, which causes issues with multicarrier signals. The LTE OFDMA downlink waveform is formed as the weighted sum of orthogonal sampled sine waves. The signal formed by this sum exhibits a large peak-to-average power ratio. The amplitude of the real and imaginary time series tends to be Gaussian with a Rayleigh distributed envelope.

The Rayleigh envelope exhibits a large peak-to-average ratio, with peak values exceeding four times the average value, with a probability of 0.00035. To preserve the fidelity of the OFDMA time series and to avoid spectral artifacts due to power amplifier clipping, we may choose to operate the amplifier with the average signal level at one-fourth of full scale; however, the efficiency sacrificed in passing signals with a 4-to-1 peak-to-average ratio through a power amplifier is excessive. The peak power level would be 16 times the average power level. This means that an amplifier required to deliver 5 watts of average power must be capable of delivering 80 W of peak power.

Power amplifiers are very inefficient in their transduction process of turning DC power to signal power when they operate at small fractions of their peak power level. For a Class B amplifier, to do a first approximation and for a wide range of output power levels, the power pulled from the power supply is constant and approximately 35 percent of the peak power level. Thus, our 80-W amplifier will pull 28 W from the DC power supply and deliver 23 of these watts to its heat sink while delivering 5 W to the external load.

The answer to these dilemmas is digital signal processing. A midrange All Programmable device such as Xilinx's 28-nm Kintex™-7 410T, with 1,540 multiply-accumulators, operating at 491.52 MHz, can deliver a peak compute performance of 757 billion MACs per second. That is a 300,000x increase over the first generation of Harvard architecture, integrated DSP processors released in the early 1980s. Companies can use this DSP compute power to implement advanced crest-factor reduction to control the peak-to-average ratio of the transmission waveform, and to apply digital predistortion processing to linearize a low-cost, highly nonlinear Doherty power amplifier. In this way, designers control the overall cost of their equipment by defraying some of the high cost associated with building high-fidelity RF processing and amplification to the relatively low-cost world of digital signal processing, which unlike the former, has been able to benefit from Moore's Law process node scaling.

XILINX AND NEXT-GENERATION SMARTER NETWORKS

The approach to solving for capacity, latency, coverage, quality of service, OPEX and CAPEX is going to be different for each mobile-network operator and cellular-infrastructure manufacturer. Each unique solution will be a function of existing assets, expertise and different go-to-market strategies. A single hardware platform with a single SoC is not going to allow OEMs to differentiate, or to have a unified, or common platform approach that permits technology scaling from small-cell through to macrocell through to cloud RAN network architectures.

A conventional merchant silicon system-on-a-chip doesn't allow for the various architectures that an OEM might need to support, ranging from the conventional basestation to architectures that integrate the baseband into the radio head, or remote radio head architectures supporting the "baseband hotel" model exploiting the cloud. The go-to-market and technology considerations will even be different for a single OEM based on the geography of the deployed network. Moreover, now more than ever time-to-market governs whether a company is successful, wildly successful or obsolete. It comes down to selecting technology that is flexible, scalable and possesses the rich mix of embedded software, compute capability and connectivity to match the complex set of attributes that characterizes wireless networks. Beyond the silicon platform, design tools and intellectual-property (IP) libraries also play a crucial role in enhancing productivity.

Xilinx's 28-nm 7 series technology, and in particular the All Programmable Zynq SoC, bring to bear the connectivity, signal-processing and embedded computing capability that is required at the heart of next-generation networks. The FPGA compute fabric, and its unparalleled signal-processing capability, can be applied to OFDMA and SC-FDMA LTE baseband processing for realizing the high-speed math requirements of advanced receivers that might employ iterative techniques, exchanging statistical information between the channel decoder, equalizer and MIMO detector in a multi-antenna system. The Zynq SoC likewise supports future-generation systems that might exploit 3D MIMO as one option for increasing capacity and exceeding the minimum requirements defined by the ITU as part of the IMT-Advanced requirements.

Another application of FPGA signal processing is to address the cost, CAPEX and OPEX, and what we might

call green considerations for the network. With LTE-A and carrier aggregation, channel bandwidths have moved to 100 MHz. Crest-factor reduction (CFR) and power amplifier linearization techniques need to evolve to handle these wide channels, and this means more signal processing. The Communications Business Unit at Xilinx has been busy for some years now working on these problems and has developed CFR and digital-predistortion IP to service a wide range of air interface protocols.

Of course, not all basestations, whether macro or small cell, need to support all possible scenarios. That fact leaves OEMs with the dilemma of needing to rapidly build a suite of cost-effective solutions that can capture the frequency-planning requirements of a network operator. These requirements are a complex equation based on the parameters of spectrum licensing, spectrum fragmentation and the need to support multiple radio-access technologies from a single RF chain—the multi-RAT challenge. Further complicating the picture is the introduction of heterogeneous networks comprising a mixture of macro basestations, small-cell underlays and Wi-Fi offload. It's a question of scale and flexibility, two properties at the very heart of Xilinx All Programmable SoCs and FPGAs.

But there is more to the problem than signal processing in the PHY. LTE, and the future LTE-A and LTE-B networks, are getting smarter in so many dimensions, especially as software increasingly becomes a key consideration for anything from running self-organizing network applications and minimizing drive tests, performing LTE parameter optimization, implementing real-time performance management, collecting statistics for network and equipment planning purposes, through to running protocol stacks and O&M software.

With the introduction of the Zynq SoC in 2012, Xilinx put an industry-first computing paradigm into the hands of system designers by providing a combination of dual-Cortex™ A9 processing tightly coupled with a high-performance programmable logic fabric that can be used for signal processing, MAC-layer acceleration and connectivity such as Ethernet, CPRI and OBSAI. These devices, and other products in the portfolio, go directly toward meeting the diverse mix of compute requirements, ISA and datapath, scalability and flexibility needed for an OEM to build differentiated products and get them to market quickly and cost-effectively.

Backhaul in this emerging age will also be a challenge. As the capability of the radio-access network evolves, the backhaul requirements of the network increase dramatically. Different cell structures and deployment scenarios will use different backhaul

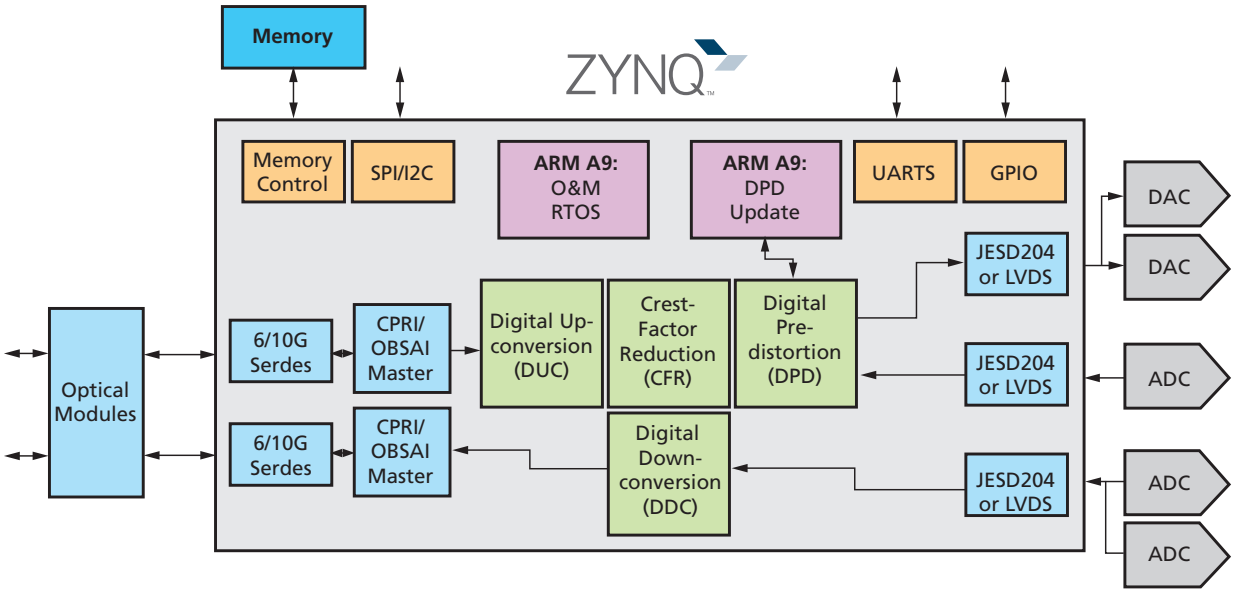


Figure 3 – The Zynq SoC enables the construction of a single-chip digital-IF processing chain incorporating the functions of up- and downconversion, crest-factor reduction, digital predistortion and connectivity.

technologies, ranging from Ethernet and optical fiber for some small-cell installations, through point-to-point microwave and e-band millimeter-wave links in other configurations. One way Xilinx is assisting OEMs in this area is with backhaul modem IP.

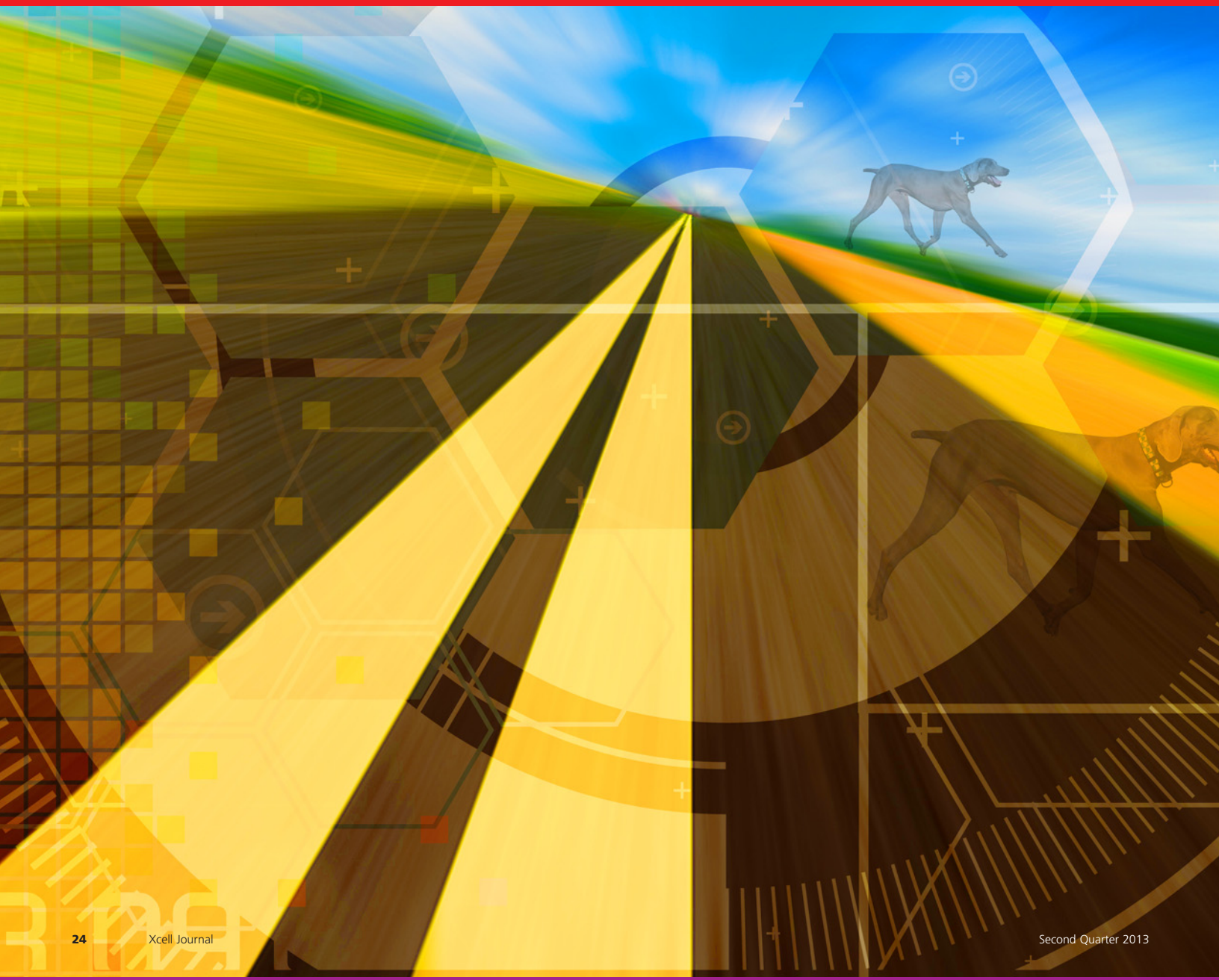
The recent acquisition of a company with expertise in this space means that Xilinx can now supply backhaul technology that not only competes with incumbent ASSP silicon, but goes a step further by providing the flexibility to scale up, or down, link capacity based on the exact needs of the problem at hand. For example, for some high-end scenarios, dual-mode transmission on both the vertical and horizontal polarizations of the electromagnetic wave can be used to increase link capacity or reliability. In other situations, a simpler, lower-data-rate and also lower-cost, single polarization scheme might be sufficient. The soft-modem IP can be integrated in a chip to realize any of the common configurations, 1+1, 2+0 and so on, in a flexible manner, enabling a customer with a progression of solutions that scale over performance and cost.

The Zynq SoC can bring all of these aspects of network design to reality in a single chip: software running on the ARM Cortex-A9 processors; RAN physical-layer functions and the backhaul modem implemented within the compute fabric and massively parallel array of multiply-accumulate engines; and connectivity protocols built using the high-speed serdes (Figure 3).

Along with advanced silicon and IP solutions, Xilinx has also recently brought a new generation of design tools to the market. One of the exciting advances in this area was with the 2012 public release of high-level synthesis technology: Vivado HLS. With this flow customers can write in C, C++ or SystemC, and compile to Xilinx silicon. This ability to program FPGAs from a high-level language enhances productivity and puts the All Programmable devices within reach of designers who don't speak hardware description languages as a first language. A new Vivado physical-implementation tool brings advances in quality of results, run-time and ease of use to customers.

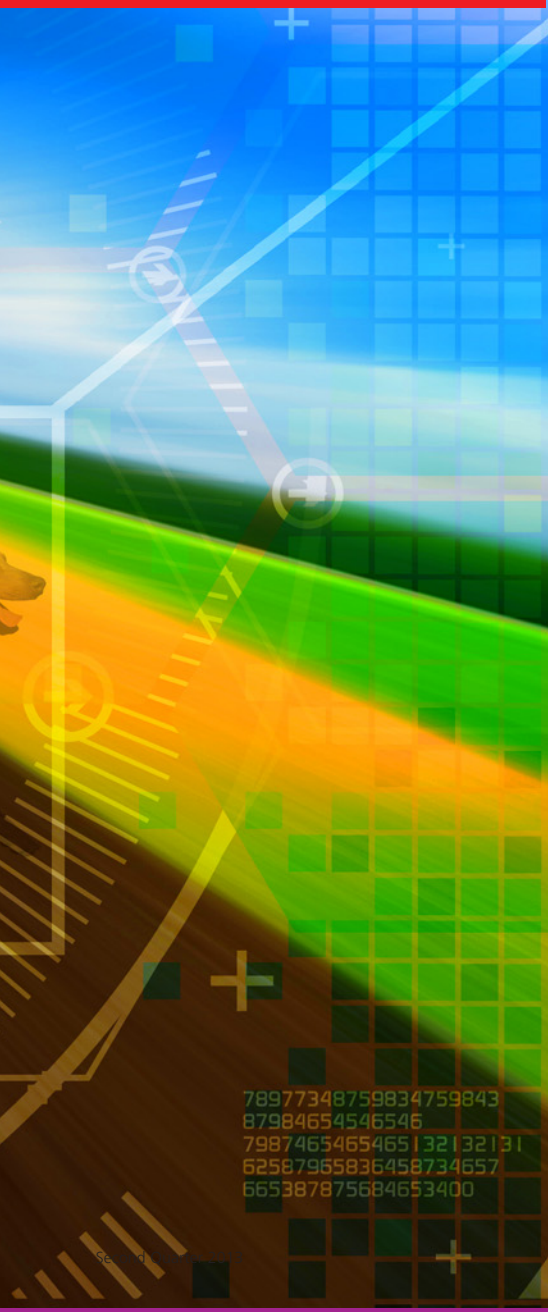
For more information on how Xilinx is helping customers build smarter networks, visit *http://www.xilinx.com/applications/smarter-networks/index.htm*.

Using OpenCV and Vivado HLS to Accelerate Embedded Vision Applications in the Zynq SoC



by Fernando Martinez Vallina
HLS Design Methodology Engineer
Xilinx, Inc.
Vallina@xilinx.com

José Roberto Alvarez
Engineering Director
for Video Technology
Xilinx, Inc.
jalvarez@xilinx.com



Pairing Vivado HLS with the OpenCV libraries enables rapid prototyping and development of Smarter Vision systems targeting the Zynq All Programmable SoC.

Computer vision has been a well-established discipline in academic circles for several years; many vision algorithms today hail from decades of research. But only recently have we seen the proliferation of computer vision in many aspects of our lives. We now have self-driving cars, game consoles that react to our every move, autonomous vacuum cleaners and mobile phones that respond to our gestures, among other vision products.

The challenge today is how to implement these and future vision systems efficiently while meeting strict power and time-to-market constraints. The Zynq™ All Programmable SoC can be the foundation for such products, in tandem with the widely used computer vision library OpenCV and the high-level synthesis (HLS) tools that accelerate critical functions in hardware. Together, this combination makes a powerful platform for designing and implementing Smarter Vision systems.

Embedded systems are ubiquitous in the market today. However, limitations in computing capabilities, especially when dealing with large picture sizes and high frame rates, have restricted their use in practical implementations for computer/machine vision. Advances in image sensor technologies have been essential in opening the eyes of embedded devices to the world so they can interact with their environment using computer vision algorithms. The combination of embedded systems and computer/machine vision constitutes embedded vision, a discipline that is fast becoming the basis for designing machines that see and understand their environments.

DEVELOPMENT OF EMBEDDED VISION SYSTEMS

Embedded vision involves running intelligent computer vision algorithms in a computing platform. For many users, a standard desktop-computing processing platform provides a conveniently accessible target. However, a general computing platform may not meet the requirements for producing highly embedded products that are compact, efficient and low in power when processing large image-data sets, such as multiple streams of real-time HD video at 60 frames/second.

Figure 1 illustrates the flow that designers commonly employ to create embedded vision applications. The algorithm design is the most important step in this process, since the algorithm will determine whether we meet the processing and quality criteria for any particular computer vision task. At first, designers explore algorithm choices in a numerical-computing environment like MATLAB® in order to work out high-level processing options. Once they have determined the proper algorithm, they typically model it in a high-level language, generally C/C++, for fast execution and adherence to the final bit-accurate implementation.

System partitioning is an important step in the development process. Here, through algorithm performance analysis, designers can determine what por-

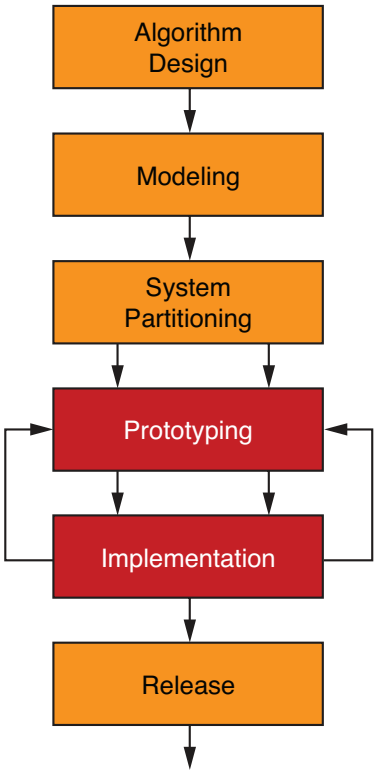


Figure 1 – Embedded vision system development process

tions of the algorithm they will need to accelerate in hardware given the real-time requirements for processing representative input data sets. It is also important to prototype the entire system in the target platform, so as to realistically measure performance expectations. Once the prototyping process indicates that a design has met all performance and quality targets, designers can then start implementing the final system in the actual targeted device. Finally, the last step is to test the design running on the chip in all use-case scenarios. When everything checks out, the team can release the final product.

ZYNQ SOC: SMARTEST CHOICE FOR EMBEDDED VISION

In the development of machine vision applications, it is very important for design teams to choose a highly flexible device. They need a computing platform that includes powerful general-purpose processing capabilities supporting a wide software ecosystem, along with robust digital signal-processing capabilities for implementing computationally demanding and memory-efficient computer vision algorithms. Tight integration at the silicon level is essential for implementing efficient and complete systems.

Xilinx® All Programmable SoCs are processor-centric devices that offer software, hardware and I/O programmability in a single chip. The Zynq SoC features an ARM® dual-core Cortex™-A9 MPCore™ processing system coupled with FPGA logic and key peripherals on a single device. As such, the device enables designers to implement extremely efficient embedded vision systems.

This level of integration between the processing subsystem, FPGA logic and peripherals in the Zynq SoC ensures faster data transfer speeds and a much lower power requirement and BOM cost than a system designed with individual components. It is feasible to implement systems in the Zynq SoC that require real-time processing for 1080p60 video sequences (1,920 x 1,080 RGB pictures at 60 frames/s) with processing capabilities in the hundreds of giga-operations per second.

To take full advantage of the many features of the Zynq SoC, Xilinx provides the Vivado™ Design Suite, an IP- and system-centric design environment that increases designer development productivity with the fast integration and implementation cycles needed to dynamically develop smarter embedded products. A component of that suite, Vivado HLS, allows you to take algorithms you've developed in C/C++ and compile them into RTL to run in the FPGA logic.

The Vivado HLS tool is particularly well-suited to embedded vision design. In this flow, you create your algorithms in C/C++; compile the algorithm or parts of the algorithm into RTL using Vivado HLS; and determine which functions are better suited to run in FPGA logic and which are better suited to run on the ARM processor. In this way, your design team can home in on the optimal performance for their vision systems running in the Zynq SoC.

To further help embedded vision developers to create Smarter Vision

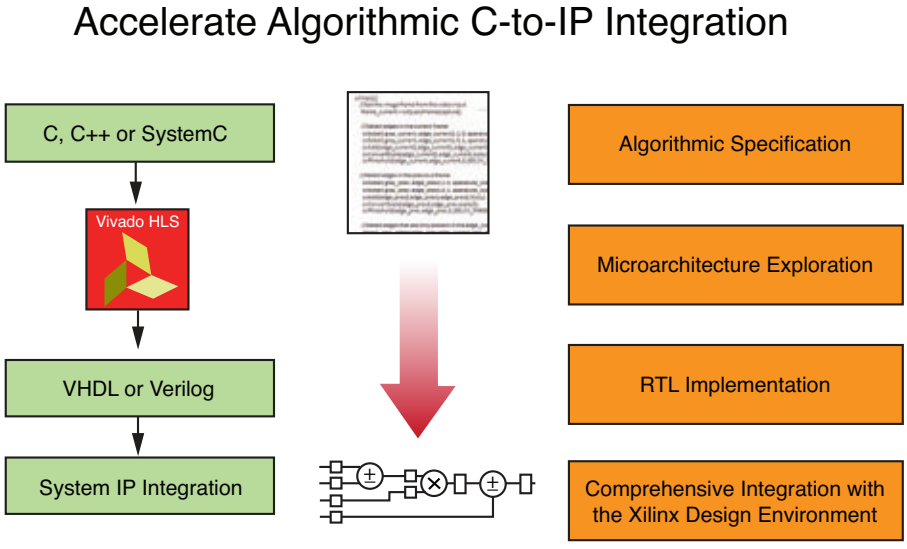


Figure 2 – High-level synthesis design flow

systems, Xilinx has added to Vivado support for the OpenCV libraries of computer vision algorithms. Xilinx has also introduced the new IP Integrator tools and SmartCORE™ IP to support these kinds of designs (see cover story, page 8).

OPENCV MAKES COMPUTER VISION ACCESSIBLE

OpenCV provides a path to the development of intelligent computer vision algorithms that are predicated on real-time performance. The libraries provide designers with an environment for experimentation and fast prototyping of algorithms.

The design framework of OpenCV is supported in multiple platform. But in many cases, to make the libraries efficient for embedded products requires implementation in an embedded platform that is capable of accelerating demanding routines for real-time performance.

While OpenCV was designed with computational efficiency in mind, it originated in traditional computing environments with support for multi-core processing. These computing platforms may not be optimal for an embedded application where efficiency, cost and power consumption are paramount.

FEATURES OF OPENCV

OpenCV is an open-source computer vision library released under a BSD license. This means that it is free to use in academic and commercial applications. It was originally designed to be computationally efficient on general-purpose multiprocessing systems, with a strong focus on real-time applications. In addition, OpenCV provides access to multiple programming interfaces like C/C++ and Python.

The advantage of an open-source project is that the user community is constantly improving algorithms and extending them for use in a wide variety of application domains. There

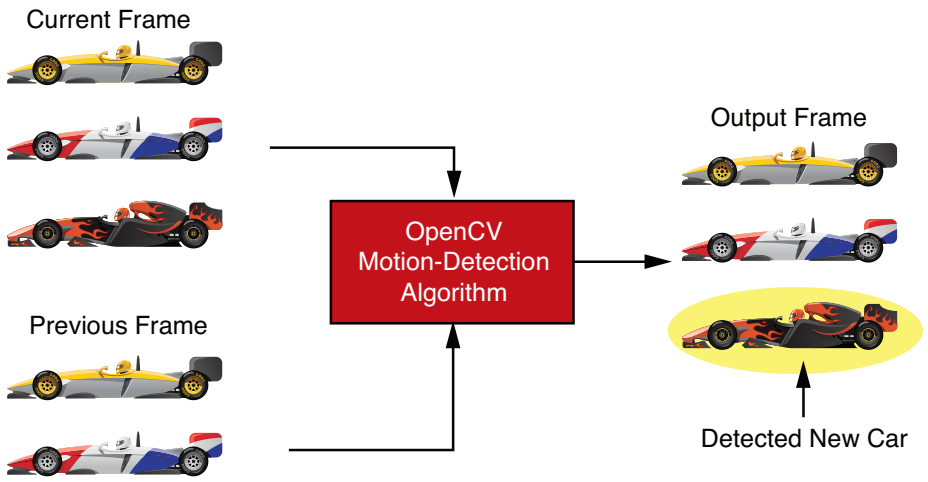


Figure 3 – Motion-detection example from the OpenCV library of algorithms

are now more than 2,500 functions implemented in OpenCV; here are some examples:

- Matrix math
- Utilities and data structures
- General image-processing functions
- Image transforms
- Image pyramids
- Geometric descriptor functions
- Feature recognition, extraction and tracking
- Image segmentation and fitting
- Camera calibration, stereo and 3D processing
- Machine learning: detection, recognition

For a more detailed description of OpenCV, please go to *opencv.org* and *opencv.willowgarage.com*.

ACCELERATING OPENCV FUNCTIONS USING HLS

Once you have partitioned the architecture of an embedded vision system to single out computationally demanding portions, HLS tools can help you accelerate these functions while still written in C++. Vivado HLS makes use of C, C++ or SystemC code to produce an efficient RTL implementation.

Furthermore, the Vivado IP-centric design environment provides a wide range of processing IP SmartCOREs that simplify connections to imaging sensors, networks and other necessary I/O interfaces, easing the process of implementing those functions in the OpenCV libraries. This is a distinct advantage from other implementation alternatives, where there is a need to accelerate even the most fundamental OpenCV I/O functionality.

WHY HIGH-LEVEL SYNTHESIS?

The Vivado HLS compiler from Xilinx is a software compiler designed to transform algorithms implemented in C, C++ or SystemC into optimized RTL for a user-defined clock frequency and device in the Xilinx product portfolio. It has the same core technology underpinnings as a compiler for an x86 processor in terms of interpretation, analysis and optimization of C/C++ programs. This similarity enables a rapid migration from a desktop development environment into an FPGA

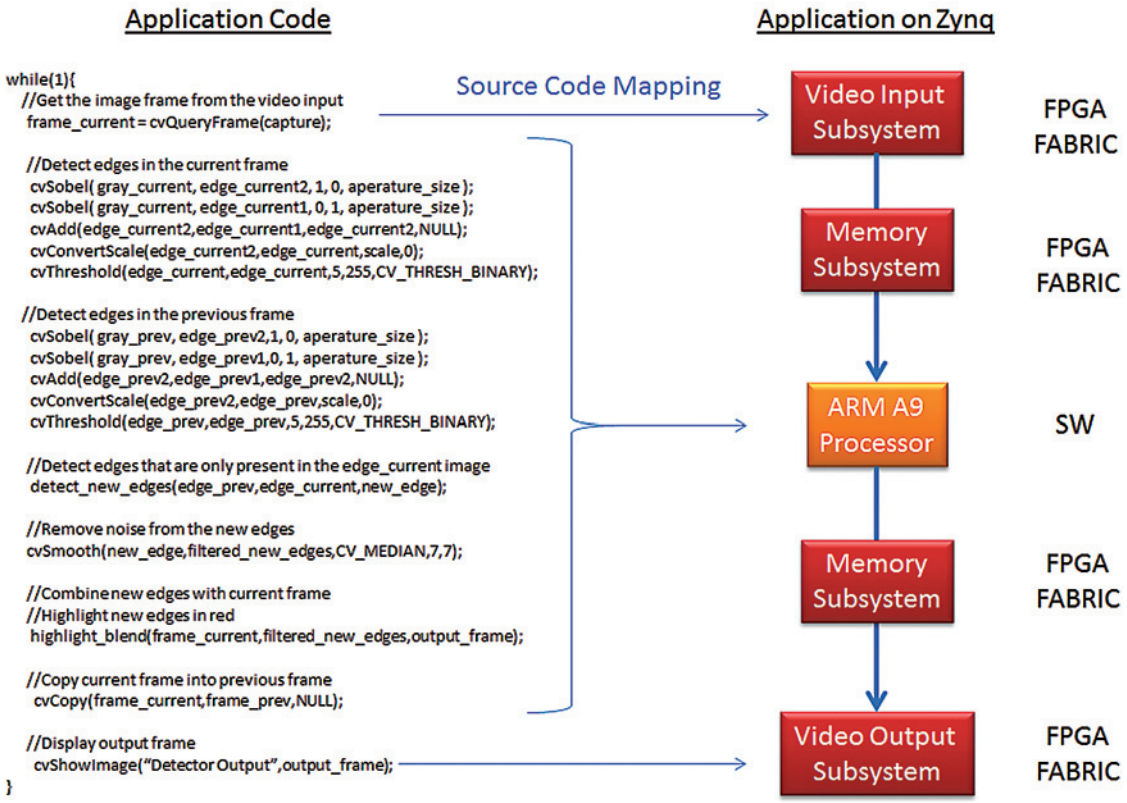


Figure 4 – Motion detection on the Zynq SoC using the ARM processor

implementation. The default behavior of Vivado HLS will generate an RTL implementation without the need for user input after you have selected the target clock frequency and device. In addition, Vivado HLS, like any other compiler, has optimization levels. Since the final execution target of the algorithm is a tailor-made microarchitecture, the level of optimizations possible in Vivado HLS is finer-grained than in a traditional compiler. The concept of O1 – O3 optimizations typical in software design for a processor is replaced with architecture-exploration directives. These directives draw on the expertise of the user to guide Vivado HLS in creating the best possible implementation for a particular algorithm in terms of power, area and performance.

The user design flow for the HLS compiler is shown in Figure 2. At a conceptual level, the user provides a C/C++/SystemC algorithmic description and the compiler generates an RTL implementation. The transformation of a program code into RTL is divided into four major regions: algorithm specification, microarchitecture exploration, RTL implementation and IP packaging.

The algorithmic-specification stage refers to the development of the software application that will be targeted to the FPGA fabric. This specification can be developed in a standard desktop software-development environment and can make complete use of Xilinx-provided software libraries such as OpenCV. In addition to enabling a software-centric development flow, Vivado HLS elevates the verification abstraction from RTL to C/C++. The user can carry out complete functional verification of the algorithm using the original software. After RTL generation through Vivado HLS, the generated design is analogous to processor assembly code generated by a traditional software compiler. The user can debug at this assembly code level, but is not required to do so.

While Vivado HLS can handle almost all C/C++ code targeted at other software compilers, there is one restriction placed on the code. For compilation into an FPGA using Vivado HLS, the user code cannot include any run-time dynamic memory allocations. Unlike a processor, where the algorithm is bounded by a single memory architecture, an FPGA implementation has an algorithm-specific memory architecture. By analyzing the usage patterns of arrays and variables, Vivado HLS can determine the physical-memory lay-

out and memory types that will best fit the storage and bandwidth requirements in an algorithm. The only requirement for this analysis to work is that you explicitly describe all memory that an algorithm consumes in the C/C++ code in the form of arrays.

The second step in the transformation from a C/C++ implementation into an optimized FPGA implementation is the microarchitecture exploration. At this stage, you apply Vivado HLS compiler optimizations to test out different designs for the right mix of area and performance. You can implement the same C/C++ code at different performance points without having to modify source code. The Vivado HLS compiler optimizations or directives are how the performance of different portions of an algorithm are stated.

The final stage in the Vivado HLS compiler flow is RTL implementation and IP packaging. These are automatic stages inside the Vivado HLS compiler that do not require RTL knowledge from users. The details of optimized RTL creation for different devices in the Xilinx product portfolio are built into the compiler. At this stage, the tool is, for all intents and purposes, a pushbutton utility that has been thoroughly tested and verified to produce timing-driven and FPGA fabric-driven RTL. The output from the Vivado HLS compiler is automatically packaged in formats accepted by other Xilinx tools such as IP-XACT. Therefore, there are no additional steps in using an HLS-generated IP core in Vivado.

The OpenCV libraries from Xilinx provide a shortcut in the process of design optimization using Vivado HLS. These libraries have been precharacterized to yield functions capable of pixel-rate processing at 1080p resolutions. The optimization knowledge required to guide the Vivado HLS compiler is embedded into these libraries. Thus, you are free to quickly iterate between an OpenCV concept application in a desktop environment to a running OpenCV application on the Zynq SoC, which enables operations on the ARM processor and the FPGA fabric.

Figure 3 shows an overview of a motion-detection application developed in OpenCV. The goal of this design is to detect moving objects in a video stream by comparing the current image frame with the previous one. The first stage in the algorithm is to detect the edges in both frames. This data-reduction operation makes it easier to analyze the relative

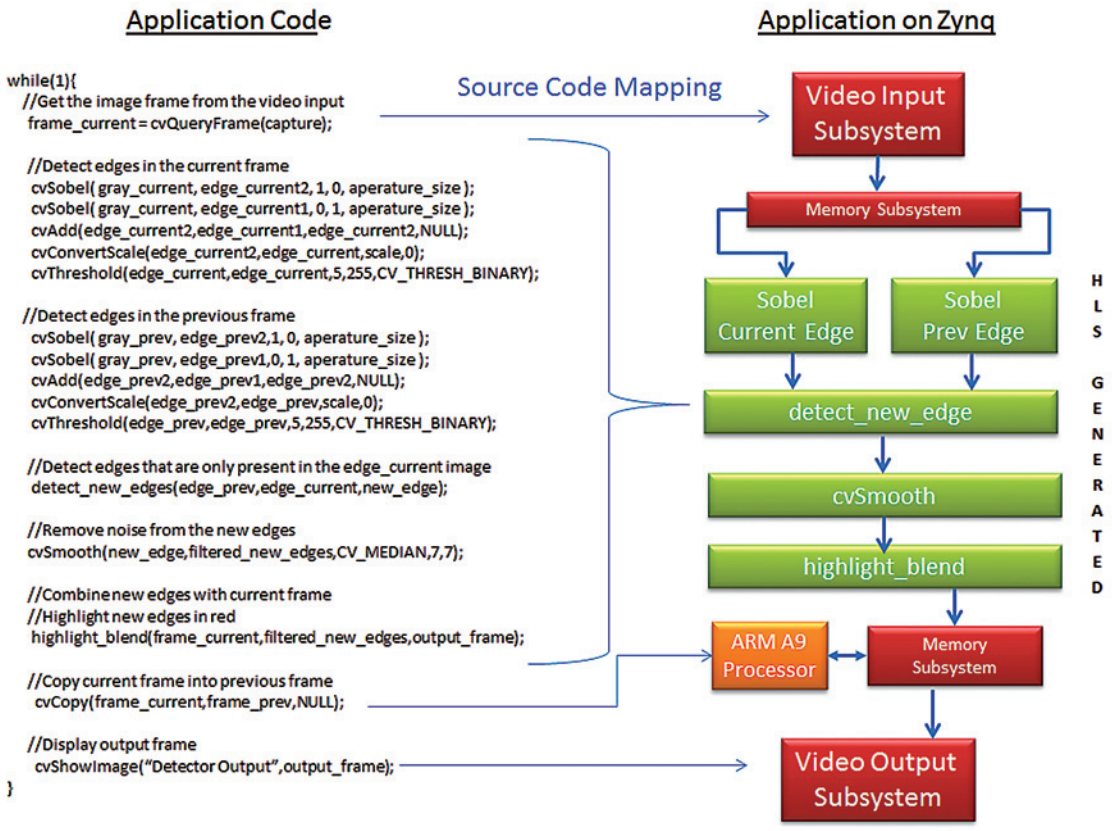


Figure 5 – Motion detection on the Zynq SoC using the programmable fabric

change between consecutive frames. Once the edge information has been extracted, edges are compared to detect edges that appear in the current image but not in the previous image. These new detected edges create a movement-detection mask image. Before the results of the new edge detection can be highlighted on the current image, you must take into account the effects of image sensor noise. This noise, which can vary from frame to frame, can lead to random false edges in the motion-detection mask image. Therefore, you must filter this image to reduce the impact of noise on the quality of the algorithm.

Noise reduction for this application is accomplished through the use of a 7x7 median filter on the movement-detection mask image. The central idea of a median filter is to take the middle value in a 7x7 window of neighboring pixels. The filter then reports back the median value as the final value for the center pixel of the window. After noise reduction, the movement-detection mask image is combined with the live input image to highlight moving edges in red.

You can fully implement the application to run on the ARM processing subsystem with the source code to Zynq SoC mapping shown in Figure 4. The only hardware elements in this implementation are for the "cvgetframe" and "showimage" functions. These video I/O functions are implemented using Xilinx video I/O subsystems in the FPGA fabric. At the time of the cvgetframe function call, the input side of the video I/O subsystem handles all the details, grabbing and decoding a video stream from an HDMI interface and placing the pixel data into DDR memory. At the time of showimage, this subsystem handles the transfer of pixel data from DDR memory into a video display controller in order to drive a TV or other HDMI-compliant video display device.

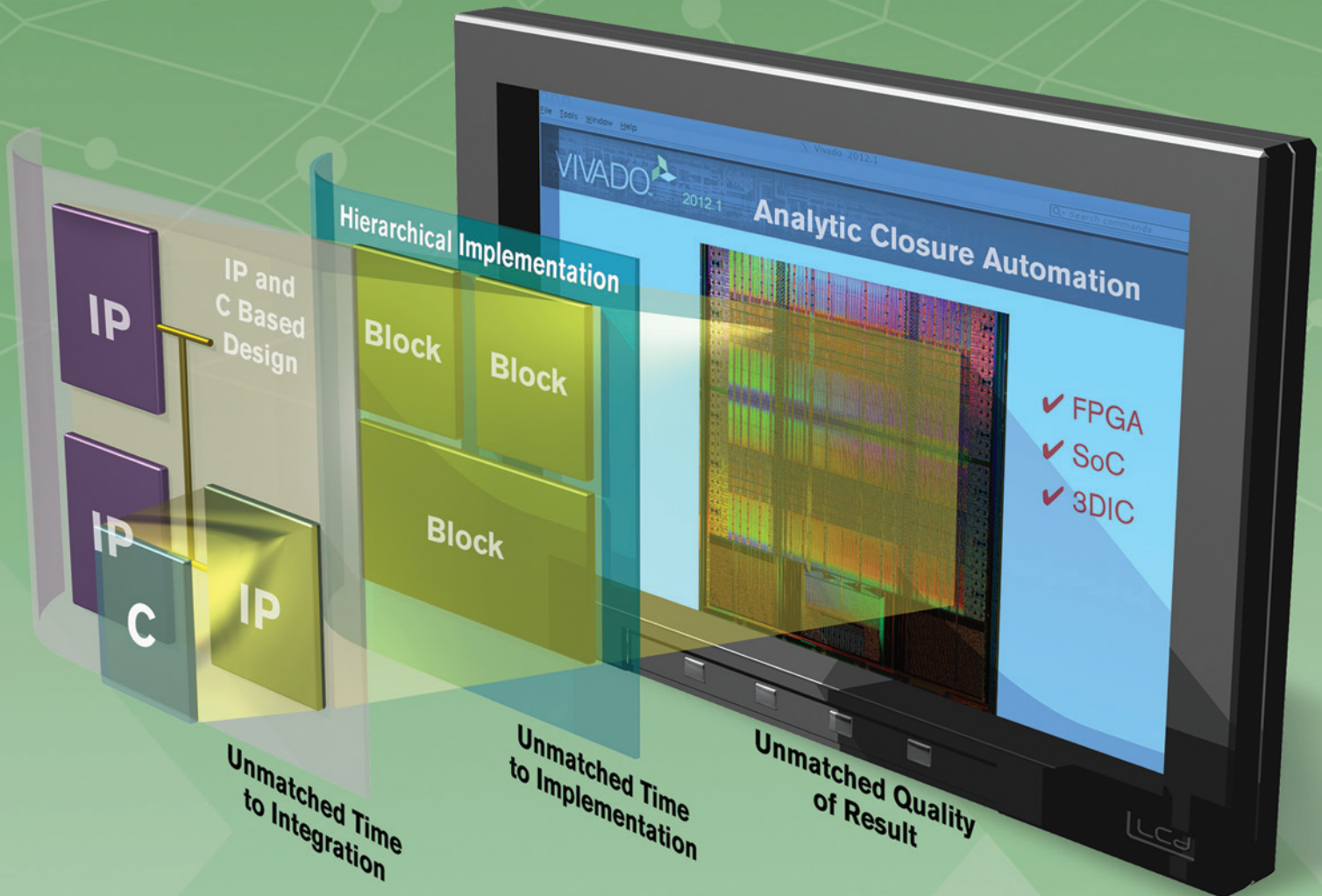
Vivado HLS-optimized OpenCV libraries for hardware acceleration enable the porting of the code in Figure 4 into a real-time 60-frame/s pixel-processing pipeline on the FPGA fabric. These OpenCV libraries provide foundational functions for the elements in OpenCV that require hardware acceleration. Without hardware acceleration—that is, if you were to run all the code in the ARM processors only—this algorithm has a throughput of merely 1 frame every 13 seconds (0.076 frames/s). Figure 5 shows the new mapping of the application after Vivado HLS compilation. Note that the video I/O mapping of the original system is reused. The computational core of the algorithm, which was previously executing on the ARM processor, is compiled into multiple Vivado HLS-generated IP blocks. These blocks, which are connected to the video I/O subsystem in Vivado IP Integrator, are optimized for 1080p resolution video processing at 60 frames/s.

The All Programmable environment provided by the Zynq SoC and Vivado Design Suite is well-suited for the design, prototyping and testing of embedded vision systems at the high data-processing rates required by the latest high-definition video technologies. Using the open-source set of libraries included in OpenCV is the best way to implement demanding computer vision applications in short development times. Since OpenCV libraries are written in C++, we use Vivado HLS to create source code that can be efficiently translated to hardware RTL in the Zynq SoC FPGA fabric, and used as convenient processing accelerators without sacrificing the flexibility of the design environment originally envisioned by OpenCV.

For more information on creating Smarter Vision designs with Vivado Design Suite, visit *http://www.xilinx.com/products/design-tools/vivado/index.htm*.

A Generation Ahead

Vivado Design Suite WebPACK™ Edition Delivers Unmatched Time to Integration and Implementation



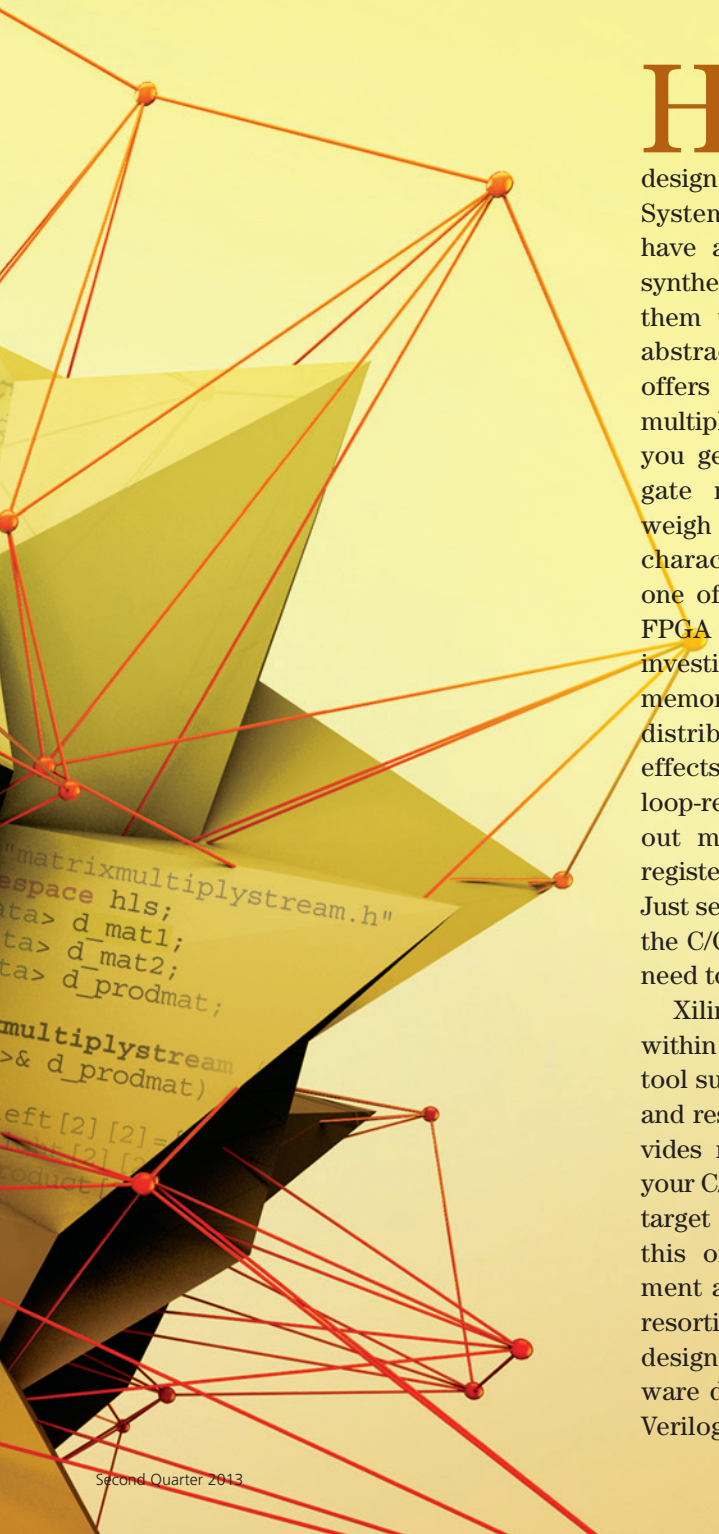
LEARN MORE

XILINX
ALL PROGRAMMABLE™

Xilinx High-Level Synthesis Tool Speeds FPGA Design

by Sharad Sinha
PhD Candidate
Nanyang Technological University, Singapore
sharad_sinha@pmail.ntu.edu.sg

Pairing Vivado HLS with high-level languages like C allows you to rapidly implement algorithms on FPGAs.



High-level synthesis (HLS) refers to an automated way of synthesizing a digital design beginning with its C, C++ or SystemC description. Engineers have a keen interest in high-level synthesis not only because it allows them to work at a high level of abstraction, but also because it offers the ability to easily generate multiple design solutions. With HLS, you get the opportunity to investigate numerous possibilities and weigh their area and performance characteristics before settling on one of them to implement on your FPGA chip. For instance, you can investigate the effects of mapping memories to Block RAM (BRAM) or distributed RAM, or explore the effects of loop unrolling and other loop-related optimizations—all without manually generating different register-transfer-level (RTL) designs. Just setting the relevant directives in the C/C++/SystemC design is all you need to do.

Xilinx has introduced an HLS tool within its newly released Vivado™ tool suite. Vivado HLS, a rebranding and respin of the AutoESL tool, provides many techniques to optimize your C/C++/SystemC code to achieve target performance. HLS tools like this one help you rapidly implement algorithms on FPGAs without resorting to a time-consuming RTL design methodology based on hardware description languages such as Verilog and VHDL.

To get a handle on how Vivado HLS operates from a user's perspective, let's take a walk through an end-to-end synthesis process, from design description (C/C++/SystemC) all the way to FPGA implementation, using matrix multiplication as our design example. Matrix multiplication is common in many applications and is extensively used in image and video processing, scientific computing and digital communications. All the results in this project were generated using Vivado HLS 2012.4, along with Xilinx® ISE® software (version 14.4) for physical synthesis and place-and-route. This flow also uses ModelSim and GCC-4.2.1-mingw32vc9 for RTL co-simulation.

Figure 1 shows a simple synthesis-oriented flow beginning with a C/C++/SystemC description of the design to be synthesized in an FPGA. The C/C++/SystemC testbench is the testbench needed to verify the correct functioning of this design. You will also need it for RTL and C co-simulation. Co-simulation involves verifying the generated RTL design (the .v or .vhd design) for functionality using this C/C++/SystemC testbench rather than an RTL testbench or a testbench written in a verification language like "e" or Vera. The clock period constraint sets the target clock period at which the design is supposed to run. The target FPGA device is the Xilinx FPGA on which the design will be mapped.

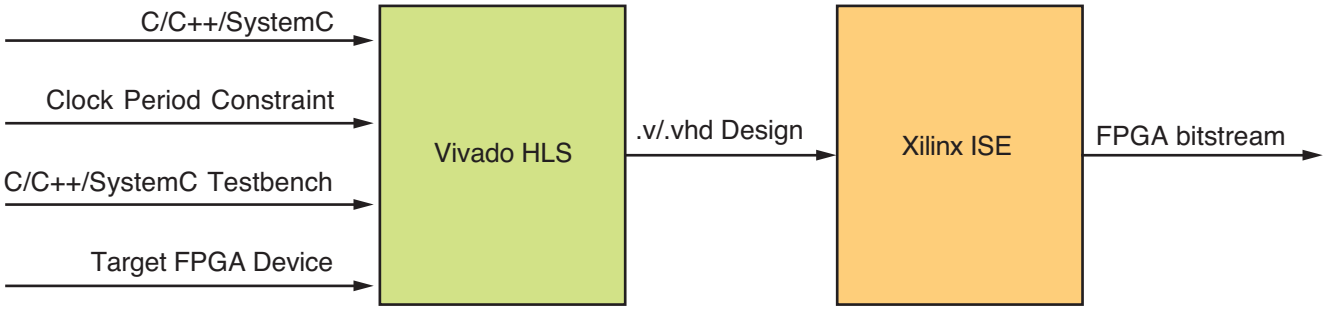


Figure 1 — FPGA synthesis flow using Vivado HLS

MATRIX MULTIPLICATION IN C

To get the most of our matrix multiplication example, we will explore various modifications of the C implementation of matrix multiplication to show their effect on the synthesized design. This process will highlight the important points you will need to be aware of when using HLS for prototyping as well as actual design. I will skip the steps involved in setting up a project, as you can easily find these in the tool documentation. Instead, I will focus on the design and implementation aspects.

In a typical Vivado HLS flow, you will need three C/C++ files: the source file, which includes the C function to be synthesized; the header file; and a file describing the testbench within a call to the main() function.

The header file includes not only the declaration of the function implemented in the source file, but also directives to support user-defined data types with specific bit widths. This allows a designer to use bit widths different from the standard bit widths defined in C/C++. For instance, an integer data type (int) is always 32 bits long in C. However, in Vivado HLS, you can specify a user-defined data type, "data," that uses only 16 bits.

Figure 2 shows a simple C function for matrix multiplication. Two matrices, mat1 and mat2, are multiplied to get matrix prod. For simplicity, all matrices are of the same size—namely, two rows by two columns.

The steps you will need to execute in the HLS flow are:

- Step 1: Build the project
- Step 2: Test the build
- Step 3: Synthesis
- Step 4: RTL co-simulation
- Step 5: Export RTL / RTL implementation

Step 1 compiles the project and tests for syntax errors and so on in the different design files. Step 2 tests the function to be implemented (present in the source file) for its functional correctness. In this step, you will execute the call to the function in the testbench and verify it as functionally correct. If the functional verification fails, you can go back and modify the design files.

Synthesis takes place in Step 3, as Vivado HLS synthesizes the function defined in the source file. The output of this stage includes Verilog and VHDL code (the RTL design) of the C function as well as estimates of resource utilization on the target FPGA and the estimated clock period with respect to the target clock period. Vivado HLS also generates estimates of latency as well as loop-related metrics.

Step 4 involves simulating the generated RTL using the C testbench. This step is called RTL co-simulation because the tool employs the same testbench that was used to verify the C source code to now verify the functional correctness of the RTL. For this step to be successful, your PATH environment variable on your system (Windows or Linux) should include the path to your ModelSim

```
void matrixmultiply(data matleft[2][2], data matright[2][2], data
product[2][2])

{
  data i,j,k;
  for(i=0;i<2;i++)
  {
    for(j=0;j<2;j++)
     {
      for(k=0;k<2;k++)
         {
         product[i][j]=product[i][j]+matleft[i][k]* matright[k][j];
         }
     }
  }
}
```

Figure 2 – Simple C code for 2 x 2 matrix multiplication

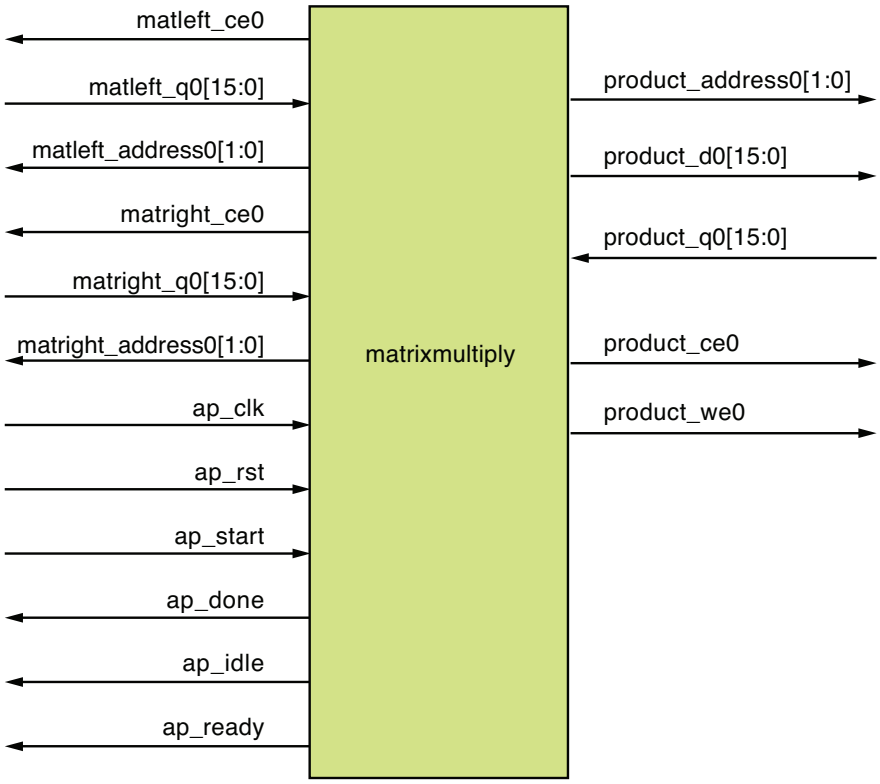


Figure 3 – Design resulting from the code in Figure 2

installation. Also, you must also have the package GCC-4.2.1-mingw32vc9 in your ModelSim installation folder.

Finally, Step 5 involves exporting the RTL as an IP block to be used in a bigger design and to be processed by other Xilinx tools. You can export the RTL as an IP-XACT-formatted IP block, as a System Generator IP block or as pcore-formatted IP for use with a Xilinx embedded design kit. While exporting the Vivado-generated RTL, you can evaluate its post-place-and-route performance by selecting the tool's "evaluate" option, thus resulting in RTL implementation. In this case Xilinx ISE runs from within the Vivado HLS tool itself. For this to happen, you need to set the path to your ISE installation in your PATH environment variable setting, and Vivado HLS will search for an ISE installation.

Of course, you are also free not to export the Vivado-generated RTL as an IP block in one of the three formats just described. The exported format files are available in three places: <project_directory>/<solution_number>/impl/<pcores> or <project_directory>/<solution_number>/impl/<sysgen> or <project_directory>/<solution_number>/impl/<ip>. You can as well use the Vivado-generated RTL in a bigger design or use it as a top-level design in itself. You need to take care of its interface requirements when instantiated in the bigger design.

When the C function in Figure 2 is synthesized, you get the RT-level implementation shown in Figure 3. As you can see, in this implementation, the elements of matrices 1 and 2 are read into the function and the elements of the product matrix are written out. Thus, this implementation assumes that memory external to the "matrixmultiply" entity is available for storing the elements of matrices 1, 2 and prod. Table 1 gives a description of the signals and Table 2 shows the design metrics.

In Table 1, start, done and idle signals are related to the finite state machine (FSM) that controls the datapath in the design. As you can see, the Vivado HLS-generated Verilog assumes that the operation starts with the start signal and that valid output data is available, with the ap_done signal going high from low. The Vivado HLS-generated Verilog/VHDL will always have at least three basic signals, ap_start, ap_done and ap_idle, along

Design Metric	Device: XC6VCX75TFF784-2
DSP48E	1
Lookup tables	44
Flip-flops	61
Best achieved clock period (ns)	2.856
Latency	69
Throughput (intitiation interval)	69

Table 1 – Design metrics for the design in Figure 3

Signal	Description
matleft_ce0	Chip enable for memory that stores matrix 1
matleft_q0[15:0]	16-bit element of matrix 1
matleft_address[1:0]	Address to read data from memory storing matrix 1
matright_ce0	Chip enable for memory that stores matrix 2
matright_q0[15:0]	16-bit element of matrix 2
matright_address[1:0]	Address to read data from memory storing matrix 2
product_ce0	Chip enable for memory that stores product matrix
product_we0	Write enable for memory that stores product matrix
product_d0[15:0]	Data written to memory that stores product matrix
product_q0[15:0]	Data read from memory that stores product matrix
product_address0[1:0]	Address from which data is to be read/written to in product matrix
ap_clk	Clock signal for design
ap_rst	Active high synchronous reset signal for design
ap_start	Start signal for start of computation
ap_done	Done signal for end of computation and output ready
ap_idle	Idle signal indicating that the entity (design) is idle
ap_ready	Indicates to the data-feeding stage that the design is ready for new input data; to be used in conjunction with ap_idle

Table 2 – Description of signals for the design in Figure 3

with the ap_clk signal. This means that no matter which design you implement using Vivado HLS, the latency of the design will constrain your streaming throughput. The design in Figure 2 has a latency of 69 clock cycles at a target clock period of 3 nanoseconds. That means that for this particular case, it will take 69 clock cycles for all the product matrix elements to be available. Hence, you cannot feed a new set of input matrices to your design before at least 69 clock cycles.

Now, the implementation as shown in Figure 3 is not probably what you would have in mind when you tried to implement matrix multiplication on an FPGA. You would probably want an implementation where you feed the input matrices, store and compute them internally, and then read out the product matrix elements. This is clearly not what the implementation in Figure 2 does. This implementation expects the input and the output matrices' data to be available in memories external to the implementation.

RESTRUCTURING THE CODE

The code in Figure 4 will serve your purpose. It will be part of the source file, which should be a C++ file and not a C file as earlier. You should include the relevant headers—hls_stream.h and ap_int.h—in the

In order to have a design where you can just stream in input matrices and stream out the product matrix, you need to implement read and write streams in your code. The streaming interfaces behave as FIFOs. By default, the depth of the FIFOs is 1.

header file matrixmultiply.h. Note that in Figure 2, when the source file was a C file, the header file included an ap_cint.h header. The header files ap_int.h and ap_cint.h help define user-defined data types with arbitrary bit widths for C++ and C source files respectively. The header file hls_stream.h file is required to make use of the stream interfaces and can only be used when the source file is in the C++ language.

In order to have a design where you can just stream in input matrices and stream out the product matrix, you need to implement read and write streams in your code. The code hls::stream<> stream_name is used to name the read and the write streams. Thus, d_mat1 and d_mat2 are read streams and d_product is a write stream. The streaming interfaces behave as FIFOs. By default, the depth of the FIFOs is 1. You will need to set the depths in the Vivado HLS directives pane by selecting the defined streams. For the code in Figure 4, each stream is four data units deep. Note here that the (i,j) loop executes before the (p,q) loop due to the sequential nature of C++ code. Hence, the d_mat2 stream will fill up after the d_mat1 stream.

Once you are done with the stream interface, you can select the matrices to be mapped to BRAM by applying the directive RESOURCE and choosing a core through the directives pane. Otherwise, the matrices will be implemented using flip-flops and lookup

```
#include "matrixmultiplystream.h"
using namespace hls;
stream <data> d_mat1;
stream <data> d_mat2;
stream <data> d_prodmat;

void matrixmultiplystream (stream<data>& d_mat1, stream<data>& d_mat2,
stream<data>& d_prodmat)
{
data matrixleft[2][2]={{0}};
data matrixright[2][2]={{0}};
data matrixproduct[2][2]={{0}};

data elementmat1;
data elementmat2;
data i,j,p,q,k;
for (i=0;i<2;i++)
 {
  for(j=0;j<2;j++)
  {
       matrixleft[i][j] = d_mat1.read();
  }
}

for (p=0;p<2;p++)
 {
  for(q=0;q<2;q++)
  {
       matrixright[p][q]=d_mat2.read();
  }
}
for(i=0;i<2;i++)
  {
    for(j=0;j<2;j++)
     {
      for(k=0;k<2;k++)
        {
matrixproduct[i][j] = matrixproduct[i][j]+matrixleft[i][k]*
matrixright[k][j];
        }
     }
  }


 for(i=0;i<2;i++)
   {
     for(j=0;j<2;j++)
      {

      d_prodmat << matrixproduct[i][j];
      }
   }
 }
```

Figure 4 – Restructured source code for matrix multiply

Signal	Description
d_mat1_V_read	Signals when the design is ready for inputs to matrix 1 (left matrix)
d_mat1_V_dout [15:0]	16-bit streaming element of matrix 1
d_mat1_V_empty	Signals to the design that no more elements for matrix 1 are left
d_mat2_V_read	Signals when the design is ready for inputs to matrix 2 (right matrix)
d_mat2_V_dout [15:0]	16-bit streaming element of matrix 2
d_mat2_V_empty	Signals to the design that no more elements for matrix 2 are left
d_product_V_din [15:0]	16-bit output element of product matrix
d_product_V_full_n	Signals that product matrix should be written to
d_product_V_write	Signals that data is being written to the product matrix
ap_clk	Clock signal for design
ap_rst	Active high synchronous reset signal for design
ap_start	Start signal to begin computation
ap_done	Done signal to end computation and signal output ready
ap_idle	Idle signal to indicate that the entity (design) is idle
ap_ready	Indicates to the data-feeding stage that the design is ready for new input data; to be used in conjunction with ap_idle

Table 3 – Description of signals for the design in Figure 5

Device: XC6VCX75TFF784-2			
Design Metric	Without BRAM or distributed RAM for matrices	With single-port BRAM for matrices	With distributed RAM (implemented in LUTs) for matrices
DSP48E	1	1	1
Lookup tables	185	109	179
Flip-flops	331	102	190
BRAM	0	3	0
Best achieved clock period (ns)	2.886	3.216	2.952
Latency	84	116	104
Throughput (initiation interval)	84	116	104

Table 4 – Design metrics for the design in Figure 5

tables (LUTs). Note that the directives pane will be active only if you keep the source file active in the synthesis view.

The implemented design for the code in Figure 4 is shown in Figure 5. Table 3 describes the signals available on this design's interface. In Table 3, d_product_V_full_n is an active low signal when the core is to be signaled that the product matrix is full. This would not be generally required in an implementation.

Table 4 shows the different design metrics after place-and-route for a clock period constraint of 3 ns, with and without mapping matrix arrays to BRAM or distributed RAM. You can see from Table 4 that the design fails to meet a timing constraint of 3 ns when matrices are mapped to single-port BRAM. This result has been deliberately included in the table to show that you can use this methodology to generate a variety of designs with different area-time metrics. You can also see from Table 1 that although the code in Figure 2 gives a latency of 69 clock cycles, which is less than the restructured design based on code in Figure 4, this design would require memory external to the matrixmultiply entity as explained earlier.

PRECISION OF THE IMPLEMENTATION

For the results shown here, I defined the data type "data" to be 16 bits wide. Hence, elements of all matrices (left, right and product) were only 16 bits wide. As a result, the matrix multiplication and addition operations were not done at full precision. You can choose to define another data type, data_t1, in the header file that would be 32 bits wide; all the elements of the product matrix would be of this data type, because a 16-bit number (element of left matrix) multiplied by another 16-bit number (element of right matrix) can be at most 32 bits wide. In this case, the resource utilization and timing results will differ from those in Tables 1 and 4.

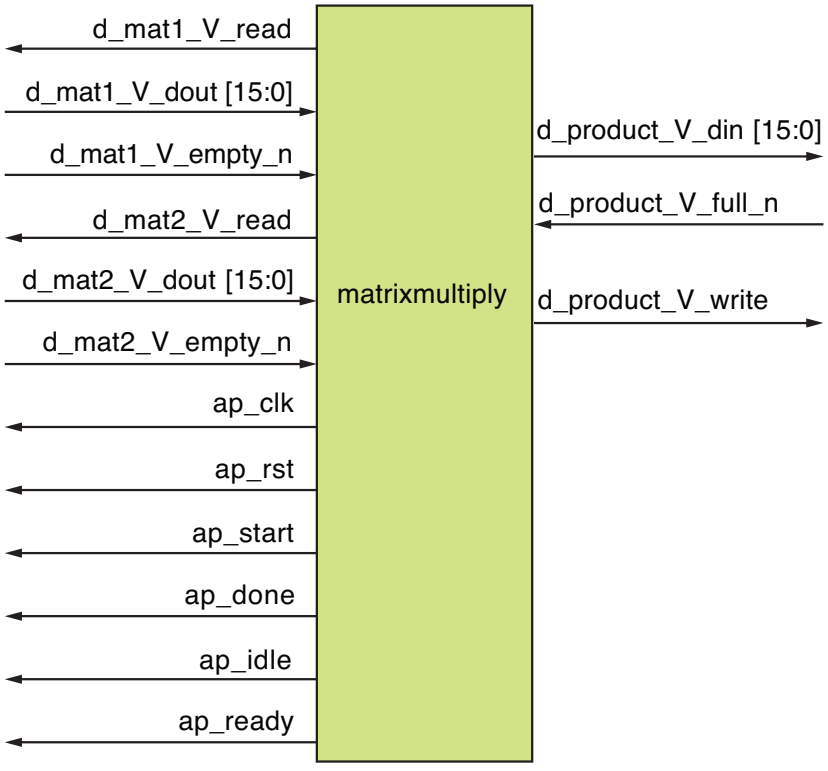


Figure 5 – Design resulting from the code in Figure 4

The restructured source code shows how multiple design solutions can arise from the same source files. In this case, one design solution had BRAM while the other one did not. Within each Vivado HLS project directory, you will see that Vivado HLS has generated separate directories for separate solutions. Within each solution directory will be a subdirectory named "impl" (for implementation). Here, within this subdirectory, you will have a directory titled "Verilog" or "VHDL," depending on which source code you used during the RTL implementation phase. This same subdirectory also contains a Xilinx ISE project file (file extension .xise). If the Vivado HLS-generated design is your top-level design, you can launch your solution in Xilinx ISE by clicking on this file, and then generate the post-place-and-route model for gate-level timing and functional simulation. You cannot do this simulation from within Vivado HLS.

After launching your solution in ISE, you should assign I/O pins to your design. Then you can select "generate programming file" in ISE Project Navigator to generate the bitstream.

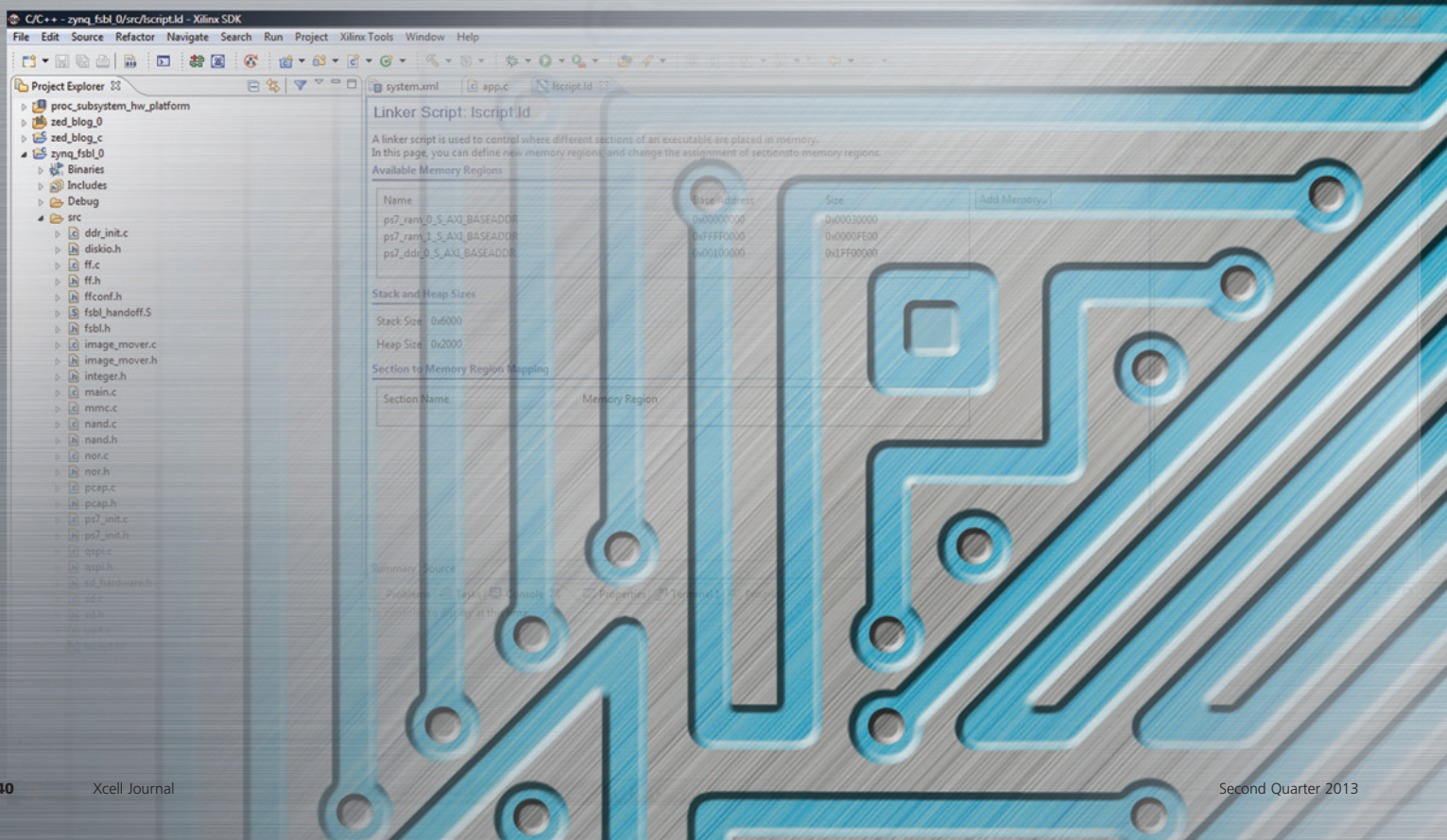
In this exercise, we have walked through an actual Vivado HLS end-to-end flow and then implementation on an FPGA. For many of the advanced features in Vivado HLS, you need to know your desired hardware architecture in order to restructure the source code. For further information, a couple of documents will prove helpful: the Vivado High-Level Synthesis Tutorial (UG871; *http://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug871-vivado-high-level-synthesis-tutorial.pdf*) and the Vivado Design Suite User Guide (UG002; *http://www.xilinx.com/support/documentation/sw_manuals/xilinx2012_2/ug902-vivado-high-level-synthesis.pdf*).

To read more by Sharad Sinha, follow his blog at http://sharadsinha.wordpress.com.

How to Configure Your Zynq SoC Bare-Metal Solution

Having developed your Zynq SoC application and tested it using the JTAG interface, the next step is to get the boot loader working.

by Adam Taylor
Principal Engineer
EADS Astrium
aptaylor@theiet.org



This is the second in a planned series of hands-on Zynq-7000 All Programmable SoC tutorials by Adam Taylor. A frequent contributor to Xcell Journal, *Adam wrote the cover story on Zynq SoC design in Issue 82 as well as the article on XPE and XPA in this issue (see page 46). He is also a blogger at All Programmable Planet. – Ed.*

Because of its unique mix of ARM processing clout and FPGA logic in a single device, the Zynq™-7000 All Programmable SoC requires a twofold configuration process, one that takes into account both the processor system and the programmable logic. Engineers will find that the configuration sequence differs slightly from that of traditional Xilinx® FPGAs. Nevertheless, the methodology is familiar and it's not at all difficult to generate a boot image and program the configuration memory.

Where standard FPGA configuration practices normally require only the FPGA bit file, you will need to add a second type of configuration file to get the maximum benefit from your Zynq SoC: the SW Executable and Linakble Format (ELF) file. The FPGA bit file defines the behavior of the programmable logic section of your design, while ELF file is the software program that the processing system will execute.

So let's have a look at how to implement a bare-metal (no operating system) software application on your Zynq SoC.

CONFIGURATION OVERVIEW

Within a Zynq SoC, the processing system (PS) is the master and therefore configures the programmable logic (PL) side of the device. (The only exception to this rule is if you use the JTAG interface for configuration.) This means you can power the processing system and have it operating while the programmable logic side is unpowered, if so desired, to reduce the overall system power. Of course, if you want to use the PL side of the Zynq SoC, you will need to power it too.

The software application and FPGA bit file are stored within the same configuration memory device attached to the processing system. The PS supports configuration by means of a number of nonvolatile memory types, including quad SPI flash, NAND flash, NOR flash and SD cards. You can also configure the system via JTAG, just like any other device.

Therefore, the Zynq SoC follows a typical processor boot sequence to configure both sides of the device, initially running from an internal boot ROM that cannot be modified. This boot ROM contains drivers for

the nonvolatile memories supported. You configure it by means of a header contained within the nonvolatile memory. The header marks the start of the configuration image and is the first thing the boot ROM looks for. The header defines a number of boot options that the boot ROM can implement, such as execute in place (not for all memories, however), first-stage boot loader (FSBL) offset and secure configuration. This header process ensures the boot ROM operates in the mode that is compatible with how the configuration memory has been formatted.

For designs, users have the option of either secure or no-secure methods of configuration. The boot ROM header supports and defines both modes. In secure configuration, the programmable logic section of the device must be powered up as the hard macros AES and SHA. You need both for decryption and must place them in the PL side of the device.

For the next stage of configuration, you will need to provide an FSBL that will configure the DDR memory and other peripherals on the processor as defined in Xilinx Platform Studio (XPS) before loading the software application and configuring the programmable logic. Overall, the FSBL is responsible for four major tasks:

- Initializing the processing system with the information that XPS provides
- Programming the PL side of the Zynq SoC if a bit file is provided
- Loading either a second-stage boot loader (SSBL), if an operating system is being used, or a bare-metal application into DDR
- Starting the execution of the SSBL or the bare-metal application

You program the PL side of the Zynq SoC via the Processor Configuration Access Port (PCAP), which allows both partial and full configuration of the programmable logic. This means you can program the FPGA at any time once the processing system is up and running. The PCAP also lets you read back and check for errors, if you are using the device in an environment where it may be subject to single-event functional interrupts (SEFIs).

To create a bootable image for our Zynq SoC, you will need at least the following functions:

1. Boot ROM header to control settings for the boot ROM (for example, execute in place, encryption, quad SPI configuration, FSBL offset and image length)
2. First-stage boot loader
3. Programmable-logic bit file
4. Software application (ELF file) for the processing-system side

Like all other Xilinx FPGAs, the Zynq SoC device uses a number of mode pins to determine which type of memory the program is stored within,

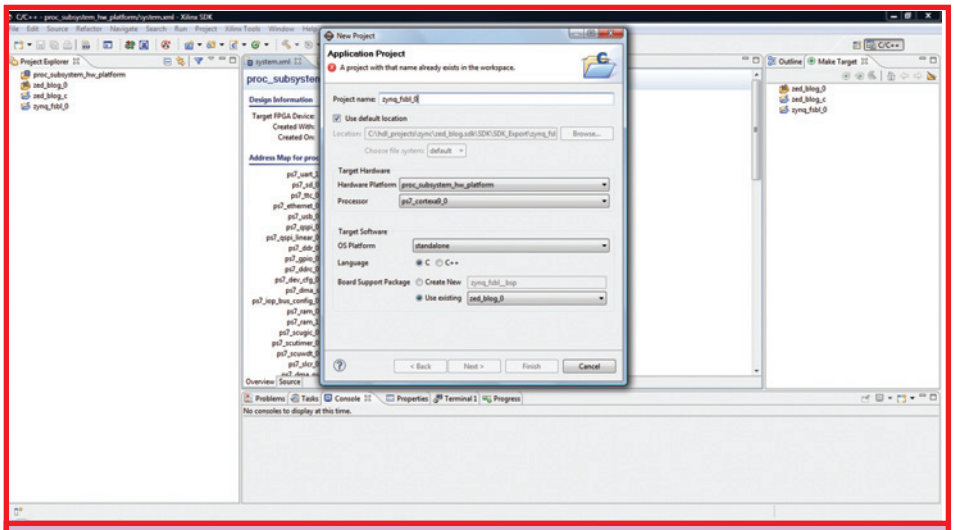


Figure 1 – Creating the first-stage boot loader project

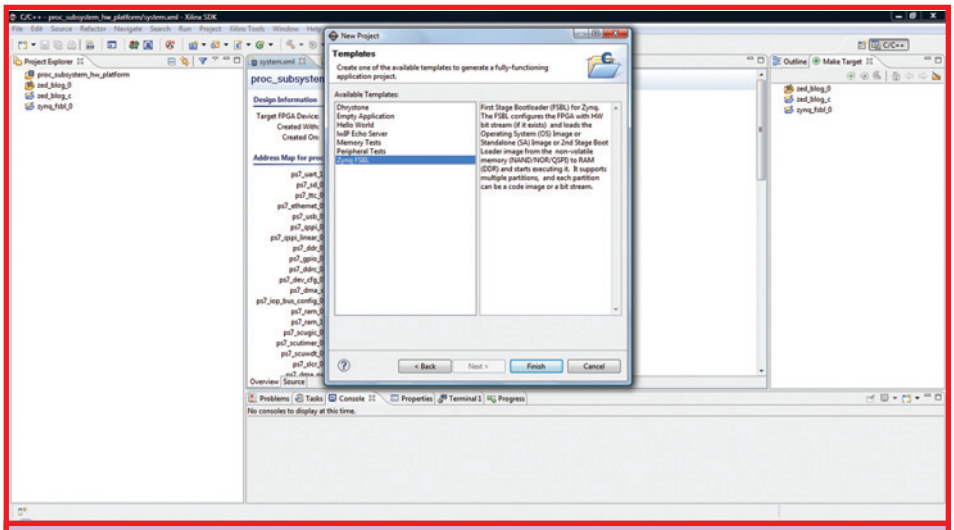


Figure 2 – Creating the FSBL from the template provided

along with other crucial system settings. These mode pins share the Multiuse I/O (MIO) pins on the PS side of the device. In all, there are seven mode pins mapped to MIO[8:2], with the first four defining the boot mode. The fifth defines whether the PLL is used or not, and the sixth and seventh define the bank voltages on MIO bank 0 and bank 1 during power-up. The first-stage boot loader can change the voltage standard defined on MIO banks 0 and 1 to the correct standard for the application; however, if you are designing a system from scratch, make sure the voltage used during power-up will not damage the device connected to these pins.

FIRST-STAGE BOOT LOADER CREATION

The next step in making a boot image is to create a first-stage boot loader. Thankfully, you don't have to do this manually—the FSBL that Xilinx has provided will load your application and configure your FPGA. You can, of course, customize the code provided to alter the boot sequence as desired for your application. Within the current Software Development Kit (SDK) workspace—the one that contains your project—create a new project using the new -> application project, as shown in Figure 1.

Choose whether you want to use C or C++, pick the board support package defined for your system and name the project—in this case, I used the name zynq_fsbl_0 for our example design.

On the next tab, select the Zynq FSBL option from the available templates as shown in Figure 2, and your FSBL project will be created. You are then nearly ready to create the boot image. If you have "compile" automatically selected, the FSBL will be compiled; if not, it will be compiled on demand later on.

However, we need to make a change to the linker script provided with the FSBL, as it has no idea where the DDR memory is located in the processor address space. Therefore, we need to open the lscrip.ld and add the DDR memory location into the file. The location of the DDR within the address space can be found in the linker script you have created for your application (see cover story, *Xcell Journal* Issue 82, for details on how to create this script). Otherwise, you can also find it in the system.xml file under the hardware platform definition.

Figure 3 shows the FSBL lscript.ld with the DDR address for the system added in. If you forget to add it, you will find that the boot loader will run and the FPGA will configure—but the application will not run if you have configured it to execute in DDR.

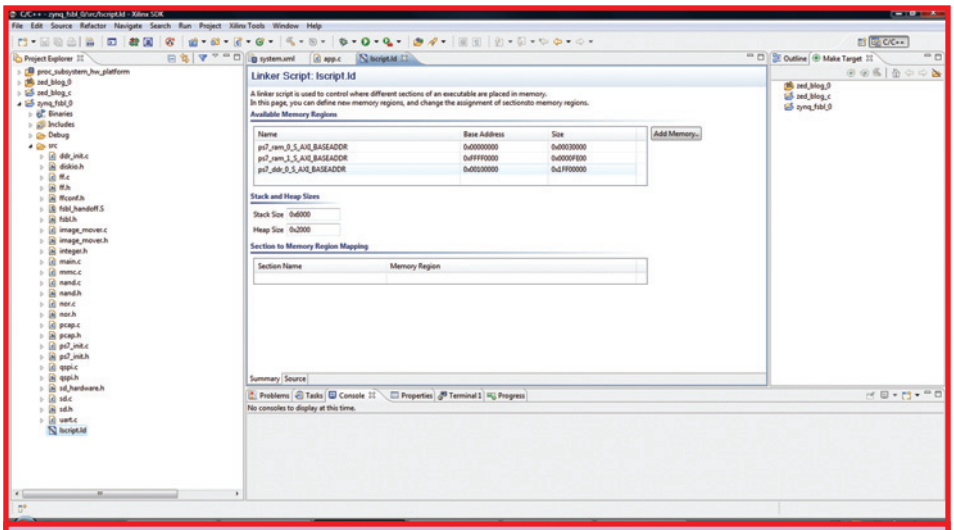


Figure 3 – Defining the DDR address space

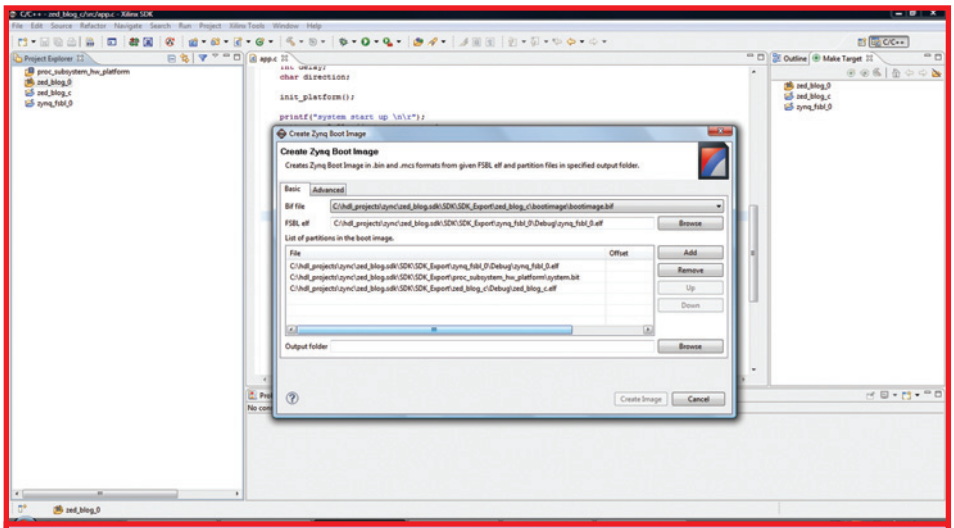


Figure 4 – Finding the complete files needed for the boot image

GENERATING A CONFIGURATION IMAGE

Looking under the Xilinx SDK Project Explorer, you should hopefully now

have the following four modules. Each should have a slightly different symbol identifying the type of module. Using Figure 4 as an example, you will see the following structure.

1. proc_subsystem_hw_platform – Named after the processing subsystem you created, this is the hardware definition of your file.
2. zed_blog_0 – This is the board support package you created.
3. zed_blog_C – This is the application itself.
4. zynq_fsbl_0 – This is the first-stage boot loader you have just created and modified the linker script for.

Since we are creating a bare-metal application (no OS), you will need three files to create a boot image, in this specific order: first-stage boot loader; FPGA programming bit file; and C application.

Creating a boot image is very simple within the SDK using the "Create Zynq Boot Image" option under the Xilinx tools menu. Once you have selected the boot image, a new dialog box will open, allowing you to choose the files needed, as seen in Figure 4.

It is important to stress that the FPGA bit file must always follow the FSBL. Clicking on "create image" will create both a *.bin and *.mcs file, which you can program into the target device.

PROGRAMMING THE CONFIGURATION MEMORY

You can program your configuration memory through either the SDK, using the "program flash" option under the Xilinx tools options, or by using Xilinx's iMPACT programming tools. For this example, we will be storing the configuration file within quad SPI flash. The QSPI interface is defined in the system assembly view of Xilinx Platform Studio. Hence, the hardware definition will contain the QSPI IP core and the location in the address map, allowing the first-stage boot loader to interface to this device and configure it.

The next stage in programming the device is to connect the Zynq SoC hardware system to your PC using the JTAG programming cable. Programming the configuration memory is then as simple as selecting the "program flash" option from the Xilinx tools menu within the SDK, as shown in Figure 5.

Navigate and select the MCS file you generated and enter an offset of 0x0 in the dialog box that appears. Then, after making sure your target hardware is powered on, click on "program" (Figure 6).

It may take a few minutes for the device to be programmed and verified (if you ticked the verify option). Once the process is complete, power down your hardware and reapply the power (alternatively, you can reset the processor). Provided you have the mode pins

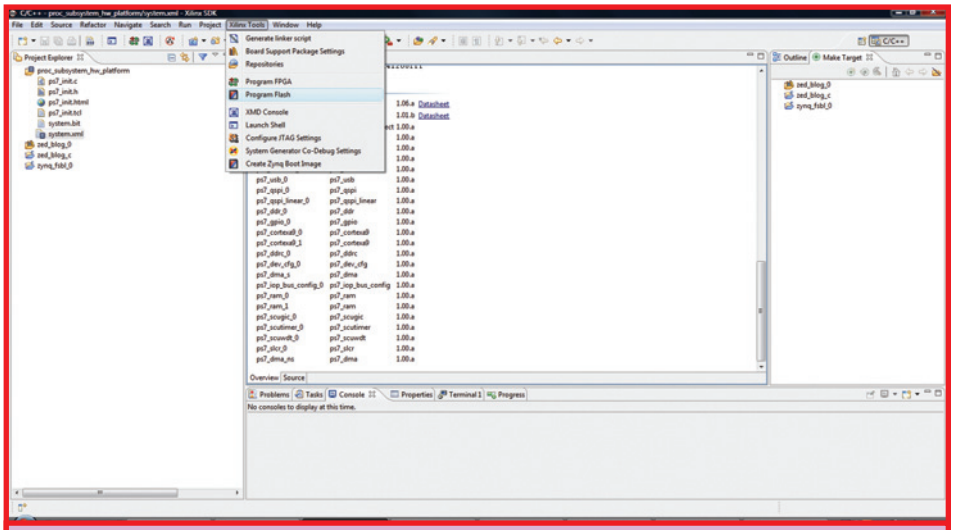


Figure 5 - Programming the configuration memory

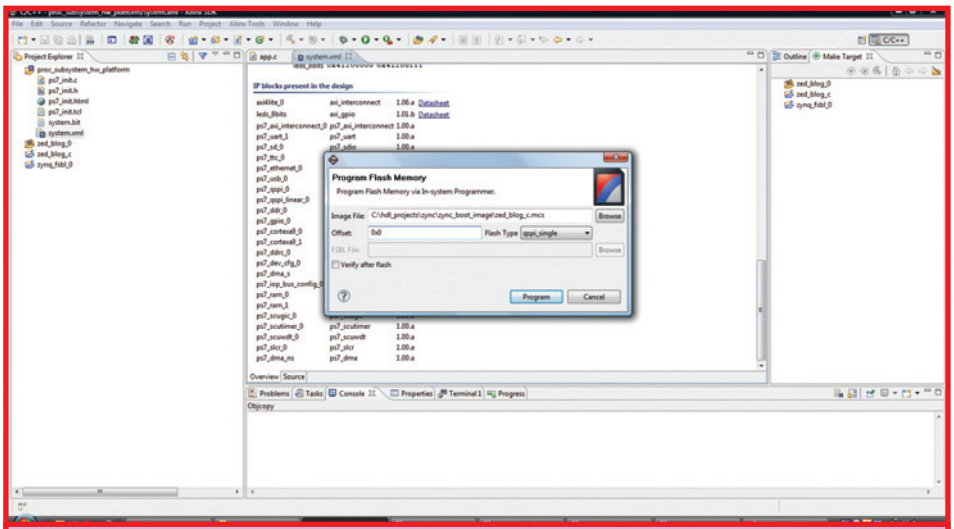


Figure 6 – Programming the flash dialog

configured correctly, your Zynq SoC system will spring to life, fully configured, and start to run the application software and FPGA design.

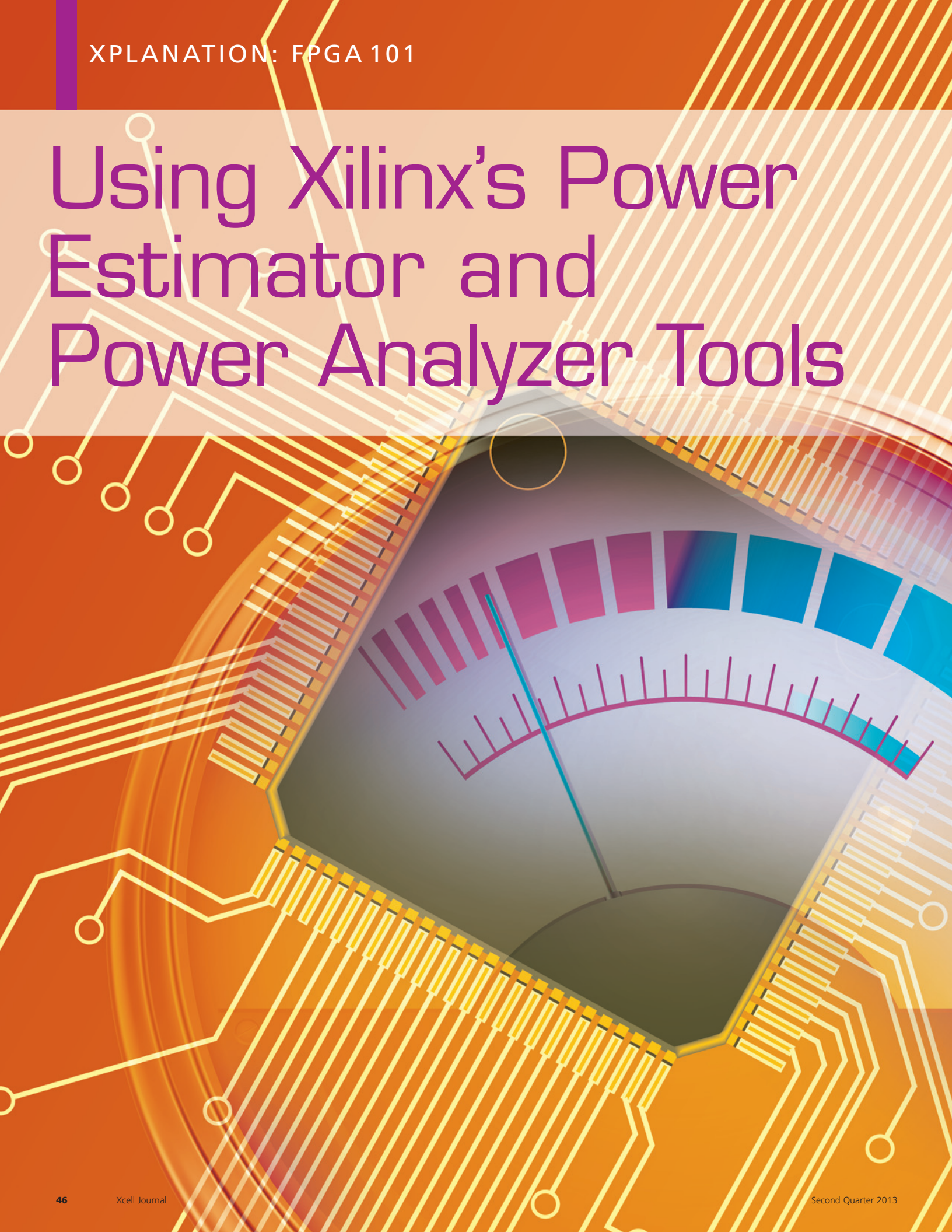
ADVANCED CONFIGURATION: PARTITION SEARCH AND MULTIBOOT

The example provided above walks you through the configuration of a single configuration image. However, it is possible for the boot ROM to perform a partition search should the initial image fail to load if the boot ROM checks fail. If this is the case, the boot ROM will look for another golden image located at a higher location within the configuration memory. The type of configuration memory you use will impact how much of the address space the boot ROM searches. If you are using an SD card, there is no partition search.

It is also possible for the Zynq SoC to load a different image from the configuration memory following a warm reset. To do this, you must define the multiboot address for the boot ROM either through the FSBL or at a later stage within the second-stage boot loader or the application. This allows the Zynq SoC's boot ROM following a warm reset to configure from the location toward which the multiboot address points, allowing a different image from the initial one to be loaded. Unlike in the partition search, these images can be located in any order within the configuration memory. However, following from power-on reset, the boot ROM will load the first valid image it finds within the configuration memory.

While different from the traditional FPGA configuration approach, Zync SoC configuration is not an arcane art. Many engineers will be familiar with the two-pronged methodology, and will find it very easy to generate a boot image and program the configuration memory using the tools provided within the Xilinx Software Development Kit.

Using Xilinx's Power Estimator and Power Analyzer Tools



Obtaining an accurate power estimation of your FPGA design can be critical for ensuring you have the correct power architecture.

by Adam Taylor
Principal Engineer
EADS Astrium
aptaylor@theiet.org

FPGAs are unlike many classes of components in that the power they will require on their core, auxiliary and I/O voltages depends upon the implementation of the design. Determining the power dissipation of the FPGA in your application is thus a little more complicated than just reading the datasheet. It can therefore be challenging to ensure you have the correct power architecture—one that takes into account not only the required quiescent currents, ramp rates and sequencing, but also has the ability to suitably power the end application while remaining within the acceptable junction temperature of the device.

What are XPE and XPA? These are two design tools provided by Xilinx that enable you to obtain accurate power analysis of your FPGA design. You will use the spreadsheet-based Xilinx® Power Estimator (XPE) in the early stages of your design and turn to Xilinx Power Analyzer (XPA) after completing the implementation. XPA allows you to perform further analysis on the power requirements of your design and assists in power optimization should it be required.

INITIAL STEPS

When you first launch a development project, it is rare that the complete FPGA design will exist (if you are lucky you may get to reuse some code, which will provide more accurate information for the power prediction). Therefore, the starting point for your power estimation will be the Xilinx Power Estimator spreadsheet (see *http://www.origin.xilinx.com/products/design_tools/logic_design/xpe.htm*). Your first power estimation will be based on the engineering team's initial thoughts on the number of clocks, logic and additional resources that the design will require.

Using XPE is very straightforward. Even better, the tool allows you to do plenty of "what if" implementations to determine the impact of various design choices on your power estimation. If your solution is power hungry, this capability can prove to be very important in helping you find the optimal implementation

The ability of XPE to predict the junction temperature of the device dependent upon the heat sinking, the airflow and the number of layers within the printed-circuit board is very useful at an early stage too. It can show whether your design can achieve derated junction temperatures for the intended implementation.

Within XPE, your first step is to complete the settings tab as accurately as you can. Along with selecting the device, pay particular attention that the package, speed grade and temp grade are correctly set—

likewise stepping, process and power mode, if applicable. All of these parameters will have a considerable impact on the overall power required—especially the process, which has the settings of "maximum" or "typical." The typical setting will give you the power your application would statistically see, while the maximum will give the worst case. It is important to ensure the power solution can handle the maximum case, but this can become challenging with larger devices that have higher current requirements.

You can also define the operating environment here as well, with the ambient temperature, heat sinking and airflow. Defining the maximum ambient temperature at this point will provide a more accurate power estimation, because the power required will increase as the device junction temperature rises. Including heat sinking, airflow or both will therefore lower your power estimation.

Also at this stage, don't overlook the Xilinx ISE® optimization setting. This setting will also have an impact on the power estimation, as different optimization schemes—for example, timing performance against area minimization—will result in varying implementations, each with its own resource usage and fan-out. They too will affect the power estimation.

The next stage of the estimation is to go through the tabs at the bottom of the XPE spreadsheet and fill in details on your proposed solution as accurately as possible. To ensure the most accurate estimation at an early stage, it is very important to define at least the resource usage, clock frequencies, toggle rate and enable rate. It is also a good idea to include a level of contingency to address the ever-present requirements creep.

This process will result in XPE providing an overall power and junction temperature estimation on the summary tab, like the one in Figure 1.

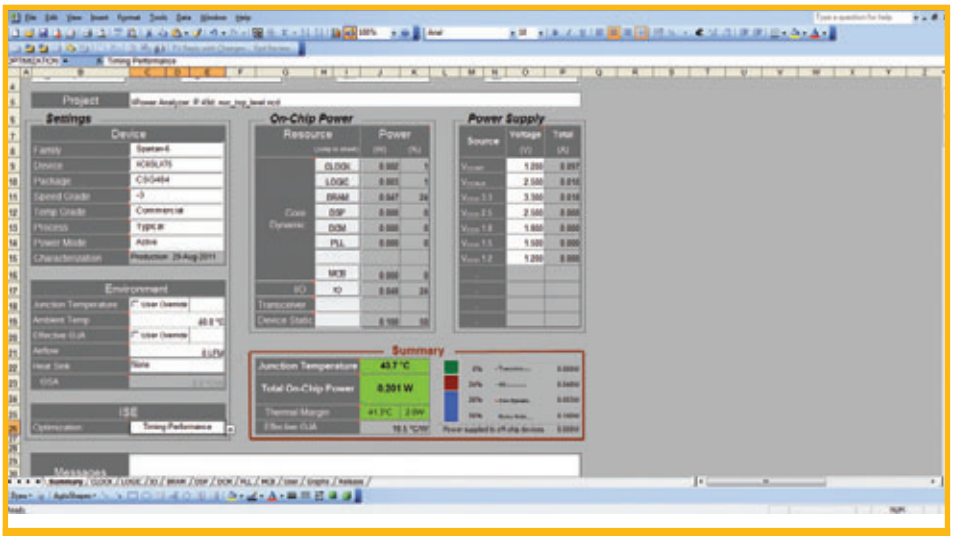


Figure 1 – Summary of the XPE device settings and power estimation

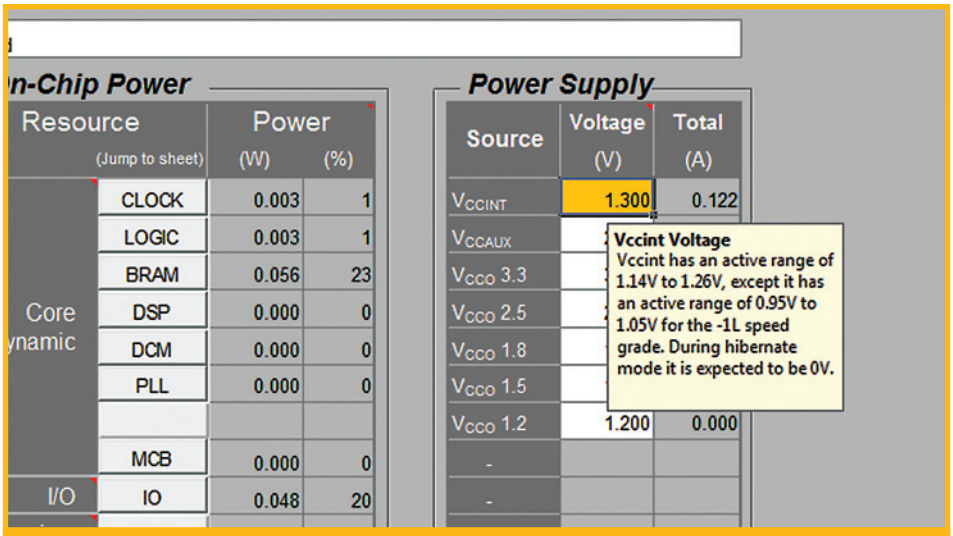


Figure 2 – The warning issued when your power supply voltage is outside of acceptable tolerance

Once your estimation is complete, you can, if you wish, export the settings from XPE for use later within the Xilinx Power Analyzer (XPA) by using the "export file" option on the summary page. This ensures the settings in XPA are aligned with those you initially used for the estimate.

As the development progresses through RTL production, trail synthesis and place-and-route, it is important to keep the estimation up to date as more-accurate information becomes available. Remember also to keep the hardware team—especially the group responsible for power engineering—informed of any change in the estimation. The power engineers should be able to provide worst-case maximum voltages for the supply rails, which can further improve the accuracy of your estimation. A higher worst-case maximum supply voltage will increase the power dissipation.

XPE is intelligent. It will color the voltage cells on your spreadsheet orange and issue a warning if the worst-case maximum supply voltage is outside of the acceptable tolerance for the device, as shown in Figure 2.

MOVING ON TO XPA

Once you have implemented the design, it is possible to obtain a much more accurate estimate of its power picture using the Xilinx Power Analyzer. How accurate this result will be depends upon the inputs you give the tool. You can open XPA by clicking on Analyze Power Distribution under the Place and Route Options within the processes window of the ISE design suite (see Figure 3).

Once XPA opens, you will again be presented with a summary screen similar to that in XPE (Figure 4). This is the place where you can define the environment and either create additional settings or import the settings from your XPE analysis.

To include the .xpa file or open a new project, you will need to use the following flow, which will include:

- A Native Circuit Design (NCD) file that defines the physical FPGA implementation
- A Settings File, to define the settings imported from XPE
- A Physical Constraints File (PCF), which contains the clocking information and mapping and UCF constraints
- A simulation file, either a Values Change Dump (VCD) or Switching Activity Interchange File (SAIF), which you can generate with your simulation tool, allowing XPA to obtain the switching information of the design

Naturally, the more information you include upfront, the more accurate the power estimation will be. Helpfully, XPA provides a confidence level on its estimation—either low, medium or high—for five categories: design implementation state, clock nodes activity, I/O nodes activity, internal nodes activity and device models.

These individual confidence levels contribute to the overall confidence level of the estimation. Again, rather helpfully, XPA will provide sugges-

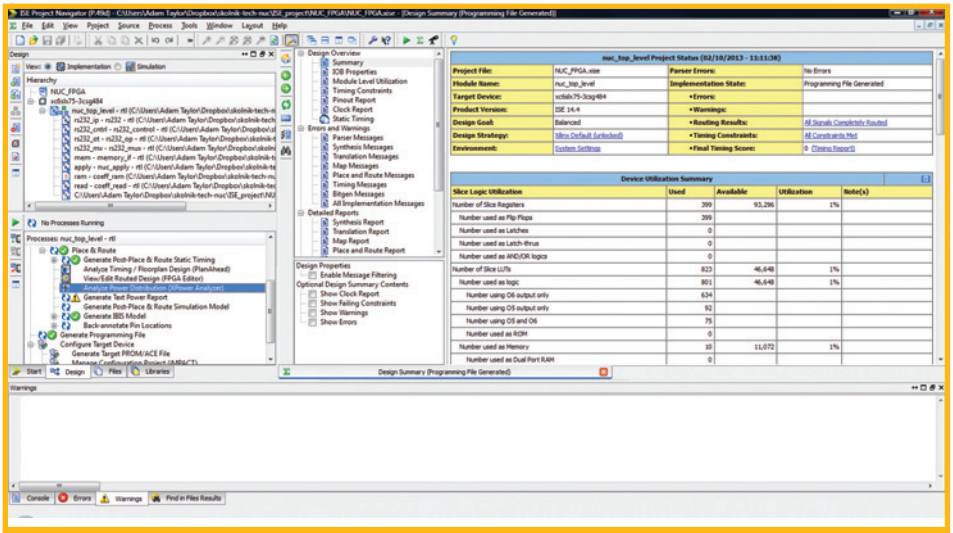


Figure 3 – Opening XPA from ISE

10 Steps to Accurate Power Estimation

To ensure you obtain the most accurate power estimation possible, here is a checklist that covers the most important steps.

1. Start with the correct environmental setting—stepping, speed grade, heat sinking, airflow and ambient temperature.
2. Ensure the process is set correctly to "typical" or "maximum" (for worst case).
3. Set the supply voltages to their worst-case maximum.
4. Make sure you have the most accurate toggle rate, fan-out, enable rate, I/O properties and clock rate.
5. Use a simulation VCD/SAIF file to provide the best information on the design performance within XPA. To obtain the best results, this file needs to be a gate-level simulation.
6. Keep the power estimation up to date as both the hardware and FPGA designs progress, so revisit it often.
7. Include contingency within your power estimation to address requirements creep in your application.
8. Be sure to define termination schemes and load capacitance correctly for the I/O in both XPA and XPE.
9. Make sure the confidence level reported by XPA is high.
10. Ensure the power solution meets all the other power requirements for the device and for reliability, and is not running at or near its load capacity.

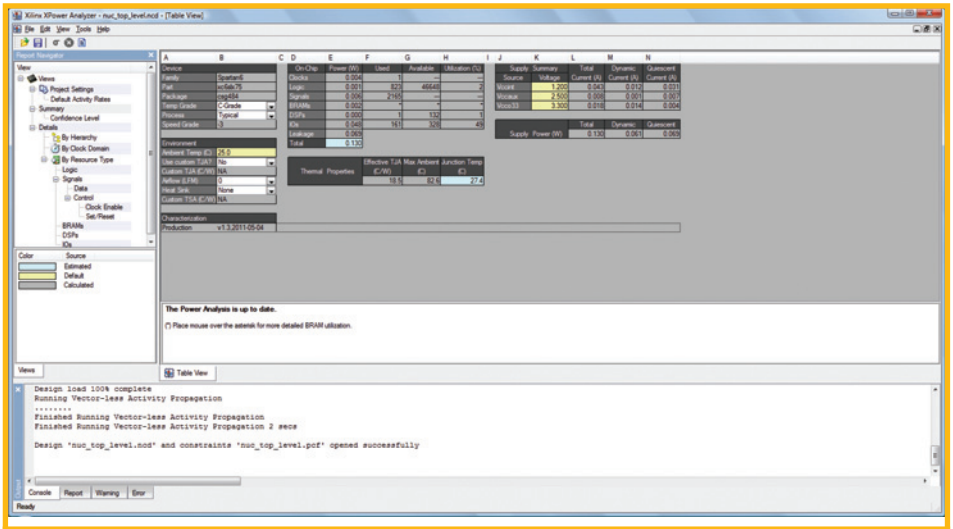


Figure 4 – The XPA summary screen

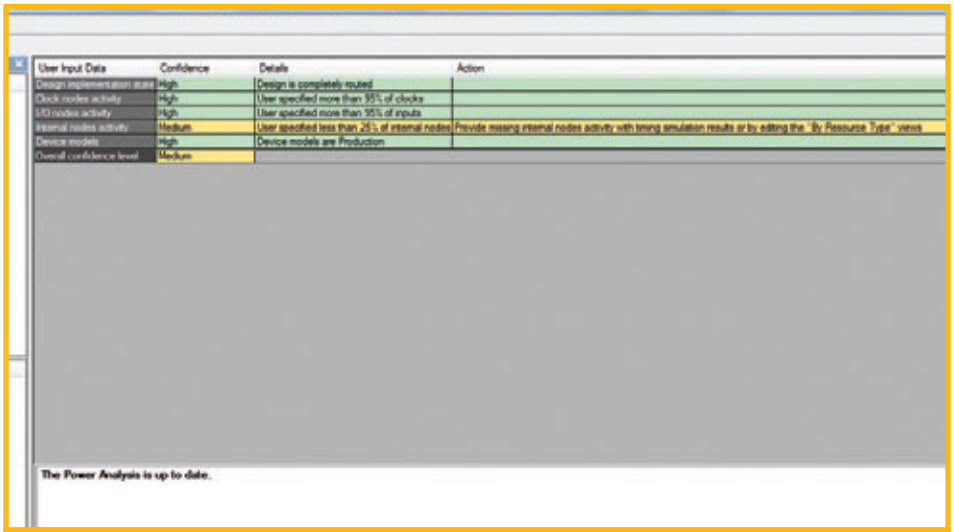


Figure 5 – XPA confidence-level report and suggested actions to improve it

tions on how to increase the confidence level of each individual category, hence boosting the overall confidence level (Figure 5).

To obtain the highest level of confidence in the power estimation, you are required to include a VCD or SAIF file from a gate-level simulation. This allows XPA to understand the behavior of the internal nodes, hence providing a better estimate.

Obtaining the VCD from your simulation is pretty straightforward, but depending on the tool you are using (Mentor Graphics' ModelSim, Xilinx's ISim, etc.), the format of the commands required might be slightly different. If you are using ModelSim, you can create a simple VCD file with the following syntax:

```
vcd file myvcd.vcd
   –define the file name and
    location if desired

vcd add /memory_if_tb/
uut_apply/*
   –the region of the design
    that is to be recorded
```

When you exit the simulation, the results will be saved into the VCD file, which you can then use with XPA. If you are using ISim, the format is a little different:

```
vcd dumpfile myvcd.vcd
   –define the file name and
    location if desired

vcd dumpvars -m
/memory_if_tb/uut_apply
   –the region of the design
    that is to be recorded

Run Simulation

vcd dumpflush
   –save results to the vcd
    file created
```

There are many more-advanced commands within the simulation tools that you can use to generate a VCD; these are detailed within the tool documentation itself.

Having included the simulation VCD file, the level of confidence of your power estimation should increase. If you cannot provide a simulation result (some can take a long time to run), then XPA will run using its internal analysis engine. For this to be accurate, it is important you again specify the toggle and enable rates.

Just as in XPE, it is also important to include the worst-case maximum supply voltages within XPA. It is possible as well to export the XPA design back into XPE to allow the engineering team to try out more experimental changes and determine their effect on the power and junction temperature estimates.

WHAT IF IT IS NOT OK?

Having included a simulation and provided the rest of the information, you will have achieved a high level of confidence in the power estimate and predicted junction temperature. In an ideal world, this will hopefully be acceptable for the application. However, should that not be the case, what other options are open to the

engineer to ensure your design has achieved the available power budget or required junction temperature?

To obtain the desired results, there are three places where we can affect the design: within the source, within the implementation or (in the case of junction temperature) within the physical-module design.

Within the design source, you can take the following steps:

1. Remove any asynchronous resets.
2. Include pipeline stages between large combinatorial functions.
3. Ensure you are using dedicated resources available within the device by either inference or instantiation.

The options available within synthesis and place-and-route are:

1. Set the optimization scheme for area minimization or power reduction.
2. Examine the design constraints to ensure they are not overly constraining the device.
3. Limit the fan-out of high-fan-out paths.
4. Utilize RAMs for things like state machines wherever possible.
5. Ensure retiming is enabled.

Hopefully, the above steps will reduce both the estimated power and the predicted junction temperature. However, if necessary you can reduce the junction temperature by using heat sinks or forced airflow, should the above not be sufficient. Doing so may, however, affect the mechanical design of the project.

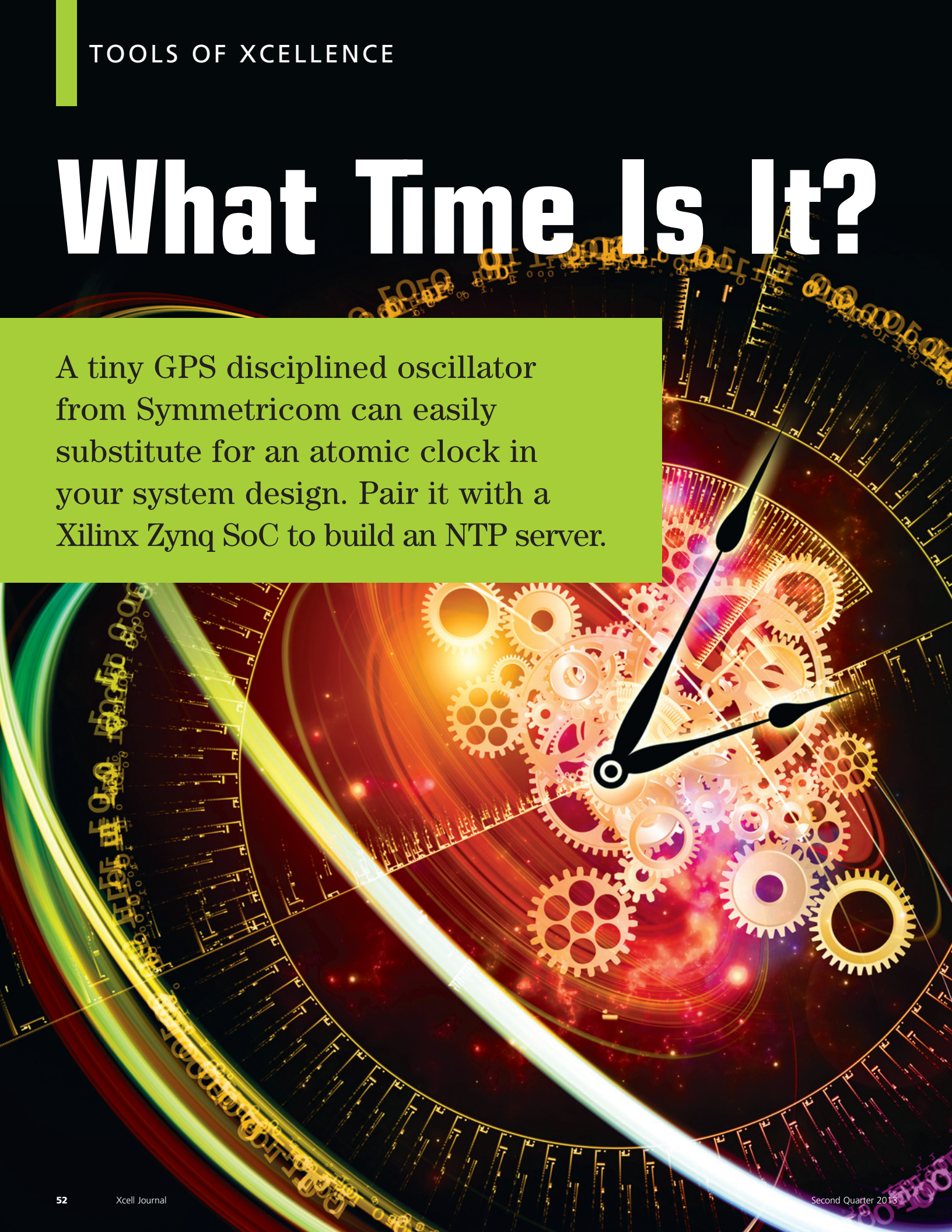
ACCURATE ESTIMATES

You can use Xilinx Power Estimator and Xilinx Power Analyzer to obtain accurate power estimations for an FPGA design, and information can be easily shared between the two tools. The accuracy of the power estimate will depend on the quality of data from both the simulation and your power-engineering team.

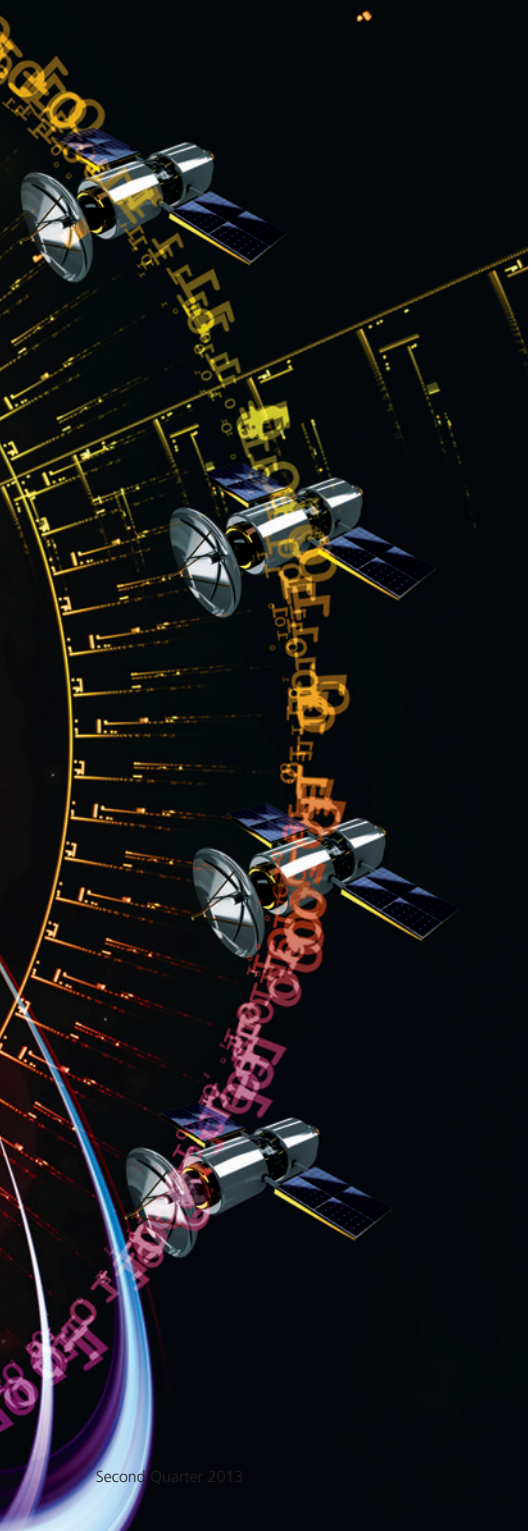
You should also provide sufficient contingencies to account for requirements creep and to ensure the solution is capable of addressing the power on currents and ramp rates of the voltage supplies.

What Time Is It?

A tiny GPS disciplined oscillator from Symmetricom can easily substitute for an atomic clock in your system design. Pair it with a Xilinx Zynq SoC to build an NTP server.



by Austin Lesea
Principal Engineer
Xilinx Labs
austin.lesea@xilinx.com



What if you need to know what time it is, or what frequency you have, or need to provide a precise frequency in your system? A few months back I obtained a sample of a small assembly, measuring roughly 1 x 1 x 0.25 inch, that seems to perform magic: a GPS disciplined temperature-compensated crystal oscillator (GPS-TCXO) from Symmetricom.

The evaluation board has an RS232-to-USB chip on the motherboard that converts the interface to/from the TCXO and its GPS receiver device. Installing the software to support the USB is probably the only real effort required, other than finding a place to stick the magnet base so that the GPS antenna can view some satellites in the sky.

Cellular basestations, radio systems and computer systems all require either precise time, or precise frequencies, to operate. One big advantage of using a fixed GPS system is that, because it isn't moving, it gives a better frequency reference, a more precise location and more precise time information than a system that is not fixed. If you have a mobile application some of that precision gets lost, but by using the TCXO you still wind up with a really nice system.

WHAT IS PRECISE?

An atomic clock is not really a clock, but an atomic oscillator with an active or passive maser. A cesium clock is the typical frequency source used for what is known as a "primary reference." It provides an accurate and precise frequency, which varies by no more than $1E^{-11}$. Typical cesium references operate down to perhaps $1E^{-12}$, but have a wander of about +/-100 nanoseconds associated with them (but no drift). Such a reference is also sometimes referred to as a Stratum 1 clock.

The GPS constellation consists of many such precise clocks in orbit (mostly rubidium sources that do not wander, but do drift a tiny bit), steered and corrected periodically by the world's stable time sources. Some 200 of the world's most accurate clocks get "voted" in a complex mathematical process, and then we all agree what time it was some 30 days ago and apply corrections.

Thus, a 10-MHz temperature-compensated crystal oscillator steered by the GPS constellation becomes a precise Stratum 1 time/frequency/position reference while locked to the satellites, and when the satellites are lost, it then drifts. However, even this drift may be anticipated and canceled if you have a history of the drift and knowledge of your temperature (the temperature compensation is not perfect, and is easily observed in operation).

NOW, FOR SOME REALLY SMALL NUMBERS

After a month of operation plugged into my desktop machine at work and, from time to time, into a laptop at home, my little board settled into its routine. When it first turned on, it was within $+/-1E^{-10}$, but after a few weeks the inherent drift of my unit was clearly visible, about $3.031E^{-12}$, which is being compensated for, so the delivered precision is less than $3E^{-12}$ (Allan variance[v] for ~6 hours; the Allan, or two-sample, variance is a measure of frequency stability in clocks, oscillators and amplifiers). See Figure 1. This is about three times better than a cesium clock, which wanders about a bit more but would have zero drift unless it were broken. The drift of a TCXO is not constant, because crystals do age. So the drift is now down to $3.029E^{-12}$ after six months.

In the event that you cannot acquire any satellites, there is another alternative: the Network Time Protocol. NTP provides time stamping over the Internet.

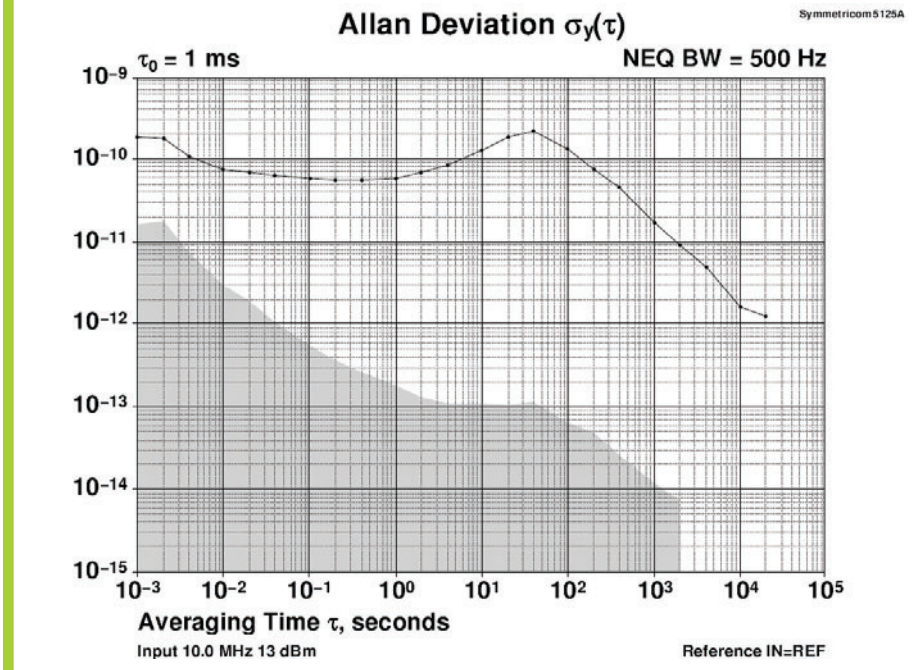


Figure 1 – The Allan variance from a test run of another unit

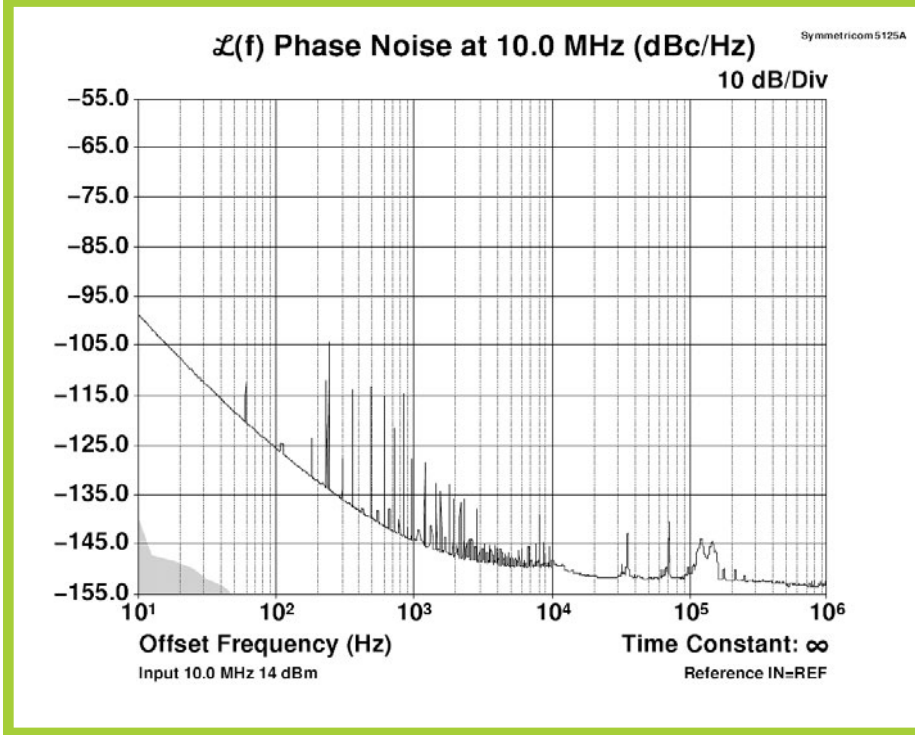


Figure 2 – Chart demonstrates the spectral purity of the outputs

The pulse-per-second (1-PPS) output had a six-month set of equally impressive statistics: a median of -397 picoseconds (it is averaging just under 400 ps of late, which could be removed as a fixed delay) with a variance of $1.72E^{-16}$.

Such levels of precision easily allow you to measure the effect of gravity and velocity on time (both of which the GPS satellites correct for as they orbit around the Earth, affected by both gravity and acceleration).

The delivered frequency is remarkably stable and constant while it is locked, which is almost 100 percent of the time. Why does it not stay locked all the time? My window faces southeast, so all satellites directly to the northwest are blocked from view of the antenna. Even so, the time spent with no satellites in view is only a few minutes a day and has little to no effect on the performance.

The receiver needs a minimum of four satellites in view to operate, and it usually sees 10 or 11 in a clear sky view. If the GPS signal is lost, I will start to see the intrinsic drift of my unit: $\sim 3.03E^{-12}$. Since every crystal is different, your unit will have its own drift. Now that I know that drift is there, I could modify the 10-MHz output with an external synthesizer with a reverse correction while the satellites are not being tracked. I could then turn off that correction when the satellites come back in view.

WHAT ELSE?

I am also able to download the latitude, longitude and altitude, and form a long-term running average. After six months, you could say that I really know exactly where my antenna is located! Of

course, I can see it from my desk, so perhaps I should say I can locate it with reference to WGS84, the World Geodetic System standard, on Google Earth. If I were in a mobile application, the receiver would also report velocity.

If I needed to make precise measurements, I would use the 10 MHz as a reference. Most high-end lab equipment has a back-panel input for an external 10-MHz reference, so the instruments are able to lock to that reference, and thus all their measurements are traceable back to the international standard.

If I were building a frequency meter, counter or RF signal source, I would put such a module in there, and pretty much forget about ever having to calibrate it. Of course, you have to occasionally get a view of the sky or place the instrument in the window so it can see as many satellites as possible. The spectral purity of the outputs is also impressive, as seen in Figure 2 (also from a test report of another unit).

NO GPS? WHAT THEN?

In the event that you cannot acquire any satellites, there is another alternative: the Network Time Protocol. NTP provides time stamping over the Internet so that you may measure time or frequency, or control a frequency source. So, in a future frequency meter one could envision a wireless 802g modem for the Internet, a GPS disciplined oscillator and the necessary control electronics (perhaps a Xilinx® Zynq™-7000 All Programmable SoC device). Such an instrument would always be accurate and would always know just how accurate it was. It would be able to diagnose itself and report whether it had any problems. I need a view of the sky for an hour or two, please….

You could port an NTP server to the Zynq SoC platform, as there is public code available. Running an NTP server on one of the ARM® cores in the Zynq device under a Linux operating system is a fairly straightforward task.

For more on the Symmetricom TCXO, see *http://www.symmetricom.com/*. Additional information on the Zynq-7000 SoC is available at *http://www.xilinx.com/products/silicon-devices/soc/zynq-7000/index.htm.*

Application Notes

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

XAPP897: DESIGNING VIDEO STREAMING SYSTEMS USING ZYNQ-7000 AP SOC WITH FREERTOS

http://www.xilinx.com/support/documentation/application_notes/xapp897-video-streaming-system-freertos.pdf

This application note by Dinesh Kumar describes how to create video systems for Digital Visual Interface (DVI) input and video test pattern generator (TPG) input using Xilinx® native IP in the Zynq™-7000 All Programmable (AP) SoC. The reference design, which is targeted for the ZC702 evaluation board, configures video IP cores with a processing frame rate of 60 Hz and 1,920 x 1,080 resolution, and displays system-level bandwidth utilization and video latency as metrics. In this way, designers can create complex, high-performance video systems using the Zynq-7000 AP SoC, with one input from the DVI and one input from the TPG.

This application note demonstrates the application of FreeRTOS, one of the two recommended operating systems for the Zynq-7000 AP SoC (Linux is the other). FreeRTOS is a free operating system that consists of only a few files. It is easy to port, use and maintain. FreeRTOS supports multiple threads or tasks, mutexes, semaphores and software timers. In the reference design, the main application runs in one of the FreeRTOS threads while another thread is created to gradually change the transparency of the on-screen display (OSD) to show the blending effect.

The design uses two AXI video direct-memory access (VDMA) cores to simultaneously move four streams (two transmit video streams and two receive video streams), each in a 1,920 x 1,080 frame size at 60 frames/second and 24 data bits per pixel (RGB). A TPG with a video timing controller (VTC) block drives one VDMA, while incoming video from DVI-In drives the other. The data from both streams to memory map (S2MM) paths of the VDMA cores are buffered into DDR, read back by the MM2S channel of the AXI VDMA and sent to a common OSD core that multiplexes or overlays multiple video streams to a single output video stream. The output of the OSD core drives the onboard HDMI video display interface through the color space converter.

The reference design is created with the Xilinx Platform Studio (XPS) in the Vivado™ System Edition 2012.4. Software created with the Xilinx Software Development Kit runs on the ARM® dual-core processor and implements control, status and monitoring functions. The reference design has been fully verified and tested on hardware.

XAPP1078: SIMPLE AMP RUNNING LINUX AND BARE-METAL SYSTEM ON BOTH ZYNQ SOC PROCESSORS

http://www.xilinx.com/support/documentation/application_notes/xapp1078-amp-linux-bare-metal.pdf

The Zynq-7000 All Programmable SoC contains two ARM Cortex™-A9 processors that can be configured to concurrently run independent software stacks or executables. This application note describes a method of starting up both processors, each running its own operating system and application, and allowing the two to communicate through shared memory.

The Zynq-7000 SoC's Cortex-A9 processors share common memory and peripherals. Asymmetric multiprocessing (AMP) is a mechanism that allows both processors to run their own operating systems or bare-metal applications with the possibility of loosely coupling those applications via shared resources. The reference design includes the hardware and software necessary to run both Cortex-A9 processors in an AMP configuration: CPU0 runs Linux and CPU1 runs a bare-metal application. Author John McDougall has taken care to prevent the CPUs from conflicting on shared hardware resources. This document also describes how to create a bootable solution and how to debug both CPUs.

XAPP890: ZYNQ ALL PROGRAMMABLE SOC SOBEL FILTER IMPLEMENTATION USING THE VIVADO HLS TOOL

http://www.xilinx.com/support/documentation/application_notes/xapp890-zynq-sobel-vivado-hls.pdf

This application note describes how to generate the Sobel edge-detection filter in the Zynq-7000 All Programmable SoC ZC702 Base Targeted Reference Design (TRD) using the Vivado high-level synthesis (HLS) tool. The techniques described by authors Fernando Martinez Vallina, Christian Kohn and Pallav Joshi present the fundamental flow for integrating an IP block generated by Vivado HLS into a Zynq SoC-based system.

The Vivado HLS tool provides a methodology for migrating algorithms from a processor onto the FPGA logic. In the context of Zynq devices, this means moving code from the ARM dual-core Cortex-A9 processor to the FPGA logic for acceleration. The code implemented with the HLS tool in hardware represents the computational bottleneck of the algorithm. For this application note, the computational bottleneck is the Sobel edge-detection algorithm running at 60 frames per second on a resolution of 1080p. The authors describe how to take a C description of the algorithm, generate RTL with the HLS tool and integrate the resulting block into a hardware system design. While this application note focuses on the Sobel IP block, the techniques described apply to all designs in which a Vivado tool-generated IP block is integrated into a Zynq SoC.

XAPP896: SMPTE2022-5/6 HIGH-BIT-RATE MEDIA TRANSPORT OVER IP NETWORKS WITH FORWARD ERROR CORRECTION ON KINTEX-7 FPGAS

http://www.xilinx.com/support/documentation/application_notes/xapp896-k7-smpte2022-56-fec.pdf

This application note by Gilbert Magnaye, Josh Poh, Myo Tun Aung and Tom Sun covers the design considerations of a video-over-IP network system using the performance features of the LogiCORE™ IP SMPTE 2022-5/6 video-over-IP transmitter and receiver cores. The design focuses on high-bit-rate native media transport over 10-Gbit/second Ethernet with a built-in forward error correction engine. The design is able to support up to three SD/HD/3G SDI streams.

The reference design has two platforms: transmitter and receiver. The transmitter platform design uses three LogiCORE SMPTE SDI cores to receive the incoming SDI video streams. The SMPTE 2022-5/6 video-over-IP transmitter core receives the SDI streams, multiplexes them and encapsulates them into fixed-size datagrams before sending them out through the LogiCORE IP 10-Gigabit Ethernet MAC. On the receiver platform, the Ethernet datagrams are collected at the 10-Gigabit Ethernet MAC.

The DDR traffic passes through the AXI interconnect to the AXI 7 series memory controller. A MicroBlaze™ processor is included in the design to initialize the cores and read the status. The reference design targets the Xilinx Kintex™-7 FPGA KC705 evaluation kit.

XAPP596: 4K2K UPCONVERTER REFERENCE DESIGN

http://www.xilinx.com/support/documentation/application_notes/xapp596-4k2k-up-converter-ref-design.pdf

In the digital display market, the next innovation wave—ultrahigh-definition (UHD) 4K2K—is now emerging. Getting to market faster than the competition with 4K2K viewing experiences is the challenge for product development designers. Xilinx Kintex-7 FPGA Display Targeted Reference Designs give designers immediate access to the power efficiency and price/performance of 28-nanometer 7 series FPGA devices for efficiently managing increased bandwidth and algorithm complexity. An upconverter is among three Xilinx reference designs that will enable customers to concentrate on product differentiation by providing basic infrastructure for 4K2K digital display signal processing.

The HDTV-to-4K2K upconverter reference design described in this application note by Yasushi Tatehira enables upconversion from 1080p high-definition TV to 4K2K progressive images. The upconversion results in showing HDTV content, which is very popular in broadcasting and packaged media, on a 4K x 2K flat-panel display. The reference design is built from two pieces of LogiCORE IP: the video scaler and the on-screen display (OSD).

This reference design, which is part of the Kintex-7 FPGA-based Display Targeted Design Platform, makes it possible to show HDTV content on a 4K2K monitor or 4K2K flat panel, providing design engineers with the base for their product designs. With this foundation addressing the mandatory upconversion, engineers can focus their efforts on differentiating their 4K2K digital TVs, displays and projectors.

XAPP891: AXI USB 2.0 DEVICE: DEMONSTRATING PERFORMANCE FOR BULK AND ISOCHRONOUS TRANSFERS

http://www.xilinx.com/support/documentation/application_notes/xapp891-7series-axi-usb-2-0.pdf

This application note demonstrates the performance measurement of the Xilinx USB 2.0 high-speed device with AXI for bulk and isochronous transactions. The test system generated is based on a Kintex-7 FPGA. The performance is measured with two separate host drivers for bulk and isochronous transactions. Authors Ravi Kiran Boddu and Dinesh Kumar describe how to develop a

USB system and corresponding ELF files for these bulk and isochronous transactions.

The AXI USB 2.0 device enables USB connectivity for a design using minimal resources. This interface is suitable for USB-centric, high-performance designs, bridges and legacy port replacement operations. The USB 2.0 protocol multiplexes many devices over a single, half-duplex serial bus. Running at 480 Mbits/second (high speed) or at 12 Mbps (full speed), the AXI USB 2.0 device is designed to be plug-and-play. The host controls the bus and sends tokens to the device specifying the required action.

The AXI USB 2.0 device supports up to eight endpoints. Endpoint 0 is used to enumerate the device with control transactions. The seven remaining user endpoints can be configured as bulk, interrupt or isochronous. Also, endpoints can be configured as input (to the host) or output (from the host). The user endpoints' data buffers are unidirectional and are configured by the configuration-and-status register of the respective endpoint. The size of the buffers can be configured from 0 to 512 bytes for bulk, 64 bytes for interrupt and up to 1,024 bytes for isochronous endpoints.

XAPP554: XADC LAYOUT GUIDELINES

http://www.xilinx.com/support/documentation/application_notes/xapp554-xadc-layout-guidelines.pdf

The Xilinx analog-to-digital converter (XADC) is a precision mixed-signal measurement system. As with any other mixed-signal/analog circuitry, a suboptimum printed-circuit board (PCB) layout can have a big impact on performance. In this application note, author Cathal Murphy details a number of simple layout guidelines to help a board designer achieve the best possible performance from the XADC.

The first step is to stagger the via associated with each BGA ball to maximize the routing space for differential signals and shields. Then:

- Route the N and P lines together at the minimum PCB spacing where possible, to maximize the benefits of differential sampling that the XADC offers.
- Place all decoupling and anti-aliasing capacitors as close to the FPGA pins as possible.
- Minimize the impedance on the power supply and ground lines.
- Minimize any common impedance between the RefN line and the GNDADC of the XADC, preferably using a star connection.

Following these guidelines will maximize the probability of achieving excellent XADC results on the first board iteration.

XAPP1077: PHYSICAL LAYER HDMI RECEIVER USING GTP TRANSCEIVERS

http://www.xilinx.com/support/documentation/application_notes/xapp1077-phy-hdmi-rx-gtp.pdf

The High-Definition Multimedia Interface (HDMI) I/O standard utilizes 3.3-volt terminated transition-minimized differential signaling (TMDS). Although TMDS signaling can be received natively using the Spartan®-6 FPGA SelectIO™ interface, the GTP transceivers can increase performance. The focus of this application note is on techniques to enable systems using the higher-bandwidth capability of GTP receivers.

It is possible to use an external passive network to adapt the GTP transceivers to receive a signal that complies with HDMI technology. Authors Paolo Novellini and Rodney Stewart present, analyze and compare two alternative passive networks from both a theoretical and a practical point of view. The authors tested both networks on a custom development board to explore their signal integrity limits and to confirm theoretical expectations. Although the results are general, the chip-to-chip use case is the one they primarily consider in this application note. The target data rate used for HDMI is 1.45 Gbps.

XAPP586: USING SPI FLASH WITH 7 SERIES FPGAS

http://www.xilinx.com/support/documentation/application_notes/xapp586-spi-flash.pdf

This application note by Arthur Yang describes the advantages of selecting a serial peripheral interface (SPI) flash as the configuration memory storage for the Xilinx 7 series FPGAs. The document includes the required connections between the FPGA and the SPI flash memory, and the details necessary to select the proper SPI flash.

The SPI flash is a simple, low-pin-count solution for configuring 7 series FPGAs. Support of indirect programming enhances ease of use by allowing in-system programming updates through reuse of connections already required for the configuration solution. Although some other configuration options permit faster configuration times or higher density, the SPI flash solution offers a good balance of speed and simplicity.

XAPP797: THROUGHPUT PERFORMANCE MEASUREMENT

http://www.xilinx.com/support/documentation/application_notes/xapp797.pdf

This application note discusses the SPI bandwidth measurement for 1 Mbyte of data, writing and reading from the

SPI flash in the Enhanced Quad mode of the AXI Quad SPI IP core. The document is based on using the KC705 board with Numonyx SPI memory, which, with a few modifications in the software example files, can be tested on any other board.

Authors Sanjay Kulkarni and Prasad Gutti show how to measure the performance of the system by writing and reading 1 Mbyte of data to and from SPI flash. These systems are built using Xilinx Platform Studio (XPS), v14.4, which is part of the ISE® Design Suite System Edition. The design also includes software, built using the Xilinx Software Development Kit, that runs on a MicroBlaze processor subsystem and implements control, status and monitoring functions. The main focus of this application note is to measure the SPI bandwidth where the core is configured in Quad SPI mode with an SPI clock rate of 40 MHz.

XAPP1084: DEVELOPING TAMPER-RESISTANT DESIGNS WITH XILINX VIRTEX-6 AND 7 SERIES FPGAS

http://www.xilinx.com/support/documentation/application_notes/xapp1084_tamp_resist_dsgns.pdf

Keeping one step ahead of the adversary, whether military or commercial, is a continuous process that involves understanding the potential vulnerabilities and attacks, and then developing new mitigation techniques or countermeasures to combat them. By taking advantage of various Xilinx FPGA anti-tamper (AT) features, a systems engineer can choose how much AT to include with the FPGA design, enabling individual silicon AT features or combining a number of them.

This application note provides anti-tamper guidance and practical examples to help the FPGA designer protect the intellectual property (IP) and sensitive data that might exist in an FPGA-enabled system. Tamper resistance needs to be effective before, during and after the FPGA has been configured by a bitstream. Sensitive data can include the configuration data that sets up the functionality of the FPGA logic, critical data or parameters that might be included in the bitstream, along with external data that is dynamically brought in and out of the FPGA during post-configuration normal operation.

Author Ed Peterson summarizes the silicon AT features available in the Virtex-6 and 7 series devices and offers guidance on various methods you can employ to provide additional tamper resistance.

XAPP739: AXI MULTI-PORTED MEMORY CONTROLLER

http://www.xilinx.com/support/documentation/application_notes/xapp739_axi_mpmc.pdf

Designers use a multiported memory controller (MPMC) in applications where several devices share a common memory controller. This is often a requirement in many video, embedded and communications applications, where data from multiple sources moves through a common memory device, typically DDR3 SDRAM memory. This application note by Khang Dao and Dylan Buli demonstrates how to create a basic DDR3 MPMC design using the ISE Design Suite Logic Edition tools, including Project Navigator and CORE Generator™. The idea is to create a high-performance MPMC by combining the Memory Interface Generator (MIG) IP block and the AXI Interconnect IP block, both provided in the ISE Design Suite Logic Edition.

The AXI interfaces used in this example design consist of AXI4, AXI4-Lite and AXI4-Stream, all of which provide a common IP interface protocol framework for building the system. The example design, a full working hardware system on the Virtex-6 FPGA ML605 evaluation platform board, implements a simple video system in which data from a video test pattern generator loops in and out of memory multiple times before being sent to the DVI display on the board. The DDR3 memory therefore acts as a multiported memory shared by multiple video frame buffers.

XAPP593: DISPLAYPORT SINK REFERENCE DESIGN

http://www.xilinx.com/support/documentation/application_notes/xapp593_DisplayPort_Sink.pdf

This application note by Arun Ananthapadmanaban and Vamsi Krishna describes the implementation of a DisplayPort™ sink core and policy maker reference design targeted for a MicroBlaze processor in the Spartan-6 FPGA Consumer Video Kit. The reference design is a loop-through system that receives video from a DisplayPort source via the receive link, buffers the video data and retransmits it over the DisplayPort transmit link. The policy maker performs several tasks, such as initialization of GTP transceiver links, register probing and other features useful for bring-up and use of the core. The application controls both the sink and source of the reference design and communicates with the monitor (sink) connected on the transmit port of the reference design using the auxiliary channel. The reference design uses DisplayPort source and sink cores generated from the Xilinx CORE Generator tool, along with a policy maker and frame buffer logic using external memory.

Latest and Greatest from the Xilinx Alliance Program Partners

Xpedite highlights the latest technology updates from the Xilinx Alliance partner ecosystem.

The Xilinx® Alliance Program is a worldwide ecosystem of qualified companies that collaborate with Xilinx to further the development of All Programmable technologies. Xilinx has built this ecosystem, leveraging open platforms and standards, to meet customer needs and is committed to its long-term success. Alliance members—including IP providers, EDA vendors, embedded software providers, system integrators and hardware suppliers—help accelerate your design productivity while minimizing risk. Here are reports from five of these members.

CUSTOM VISION ALGORITHMS FROM MATHWORKS

MathWorks (Natick, Mass.), provider of the MATLAB® and Simulink® tools, offers several add-ons integral for use with Xilinx hardware for embedded vision. The company's Computer Vision System Toolbox provides algorithms and tools for the design and simulation of computer-vision and video-processing systems. HDL Coder generates portable, synthesizable Verilog and VHDL code from MATLAB functions, Simulink models and Stateflow charts. HDL Verifier automates the verification of HDL code on Xilinx FPGA boards by enabling FPGA-in-the-loop (FIL) testing. In addition, MathWorks recently released a hardware support package for working with the Xilinx Zynq™-7000 All Programmable SoC.

"Smarter vision applications are constantly evolving, adding new features like analytics, recognition and tracking," said Ken Karnofsky, senior strategist at MathWorks "These features require improved system performance and greater design flexibility for hardware and software implementation. Over the last few years, MathWorks and Xilinx have enabled custom vision algorithms developed with MATLAB and Simulink to be easily integrated with available IP and implemented on Xilinx hardware, including the Zynq-7000 SoC for hardware/software coprocessing. This workflow accelerates development by bridging the divide between algorithm exploration and system implementation."

Ken Karnofsky
MathWorks

Please visit *www.mathworks.com/zynq* for more information.

NI LABVIEW SIMPLIFIES FPGA PROGRAMMING

National Instruments' LabVIEW changes the rules of FPGA programming, delivering new technologies that convert graphical block diagrams into digital hardware circuitry, simplifying development and shortening time-to-market for advanced control, monitoring and test applications.

The Austin, Texas, company offers frame grabbers with a user-programmable FPGA in the image path as well as embedded systems, such as CompactRIO, that have image-processing capabilities combined with FPGA-enabled I/O. CompactRIO combines an embedded controller for communication and processing, plug-in I/O modules and a reconfigurable chassis housing the user-programmable FPGA. System designers use the LabVIEW graphical development platform for programming floating-point processors and FPGAs for image processing, digital and analog I/O, communication and more within a single development environment.

Jamie Smith
National Instruments

"Xilinx All Programmable FPGA technology is at the center of the NI LabVIEW RIO architecture," said Jamie Smith, director of embedded systems marketing at NI. "LabVIEW lets engineers who are familiar with FPGAs take advantage of their inherent benefits of performance, versatility and determinism."

Learn more about NI's approach to FPGA-based design at *http://www.ni.com/fpga/*.

ZYNQ SoC-BASED VIDEO ENGINE FROM OMNITEK

The OSVP IP block from IP vendor OmniTek (Basingstoke, U.K.) provides complete multivideo format conversion and compositing. All interfaces conform to ARM®'s AXI4 protocol. The IP supports up to eight channels of simultaneous chroma upsample, color-space conversion, deinterlacing and resizing, with overlay graphics capability.

The OZ745 is a video development platform based around the Xilinx Zynq-7045 SoC. The kit includes all the basic components of hardware, design tools, IP, preverified reference designs and a board support package to rapidly develop video- and image-processing designs. OmniTek also supplies software and firmware IP cores along with design services to further accelerate time-to-market.

Roger Fawcett
OmniTek

"At OmniTek, we are extremely proud of our design team's skills in image-processing algorithm design for optimum FPGA and CPU partitioning," said managing director Roger Fawcett. "The latest Zynq SoC devices provide a single-chip solution for such designs. We have exploited this efficient architecture in the Real Time Video Engine 2.1, which combines highly optimized Xilinx and OmniTek FPGA and software IP."

Please visit *http://www.omnitek.tv/sites/defaul/files/OSVP.pdf* and *http://www.omnitek.tv/sites/default/files/OZ745.pdf* for more information.

VIDEO AND GRAPHICS IP FROM XYLON

The power of the logicBRICKS video IP cores from Xylon (Zagreb, Croatia) are best demonstrated by the logiADAK Automotive Driver Assistance kit. Built around the Xilinx Zynq-7000 All Programmable SoC, this kit makes it easy to combine hardware-accelerated video-processing features implemented in programmable logic with high-level, complex control algorithms running on the ARM processor system in a single chip.

Graphics cores for full-range implementations of 2D and 3D graphics processing units (GPUs) seamlessly blend with the video IP cores and can be immediately used with different operating systems (e.g., Linux, Microsoft Windows Embedded Compact 7 and Android).

Davor Kocacec
Xylon

"More than 15 years ago, when we started developing our logicBRICKS IP cores for video processing with Xilinx FPGAs, no one could imagine today's smart vision applications running on a single Xilinx All Programmable device," said Davor Kocacec, CEO of Xylon. "Our easy-to-use IP cores, software and design services enable our customers to quickly build large portions of their next Xilinx-based SoC and fully concentrate on key differentiating features."

More information about Xylon products and solutions for embedded vision, as well as a number of preverified reference designs for download, can be found at *www.logicbricks.com*.

NEWS FROM THE EMBEDDED VISION ALLIANCE

The Embedded Vision Alliance (Walnut Creek, Calif.) is an industry partnership formed to inspire and empower design engineers to create more capable and responsive products through integration of vision capabilities.

According to the alliance's founder, Jeff Bier, computer vision has long been a niche technology, because computer vision equipment has been big, expensive and complex to use. Recently, however, products like the Microsoft Kinect and vision-based automotive safety systems have demonstrated that computer vision can now be deployed even in cost-sensitive applications, and in ways that are easy for nonspecialists to use.

"We use the term 'embedded vision' to refer to the incorporation of visual intelligence into a wide range of systems, creating 'machines that see and understand,'" said Bier. "One of the key factors enabling the current transition from the niche technology of computer vision to pervasive embedded vision is more capable and efficient processors. Xilinx has been a leader in this realm, developing new devices, like the Zynq-7000 SoC family, that deliver an enormous amount of programmable processing power in a very modest price, power and size envelope."

Jeff Bier
Embedded Vision Alliance

The Embedded Vision Alliance provides training videos, tutorial articles, code examples and an array of other resources (all free of charge) on its website, *www.Embedded-Vision.com*.

What's New in the Vivado 2013.1 Release?

The Vivado™ Design Suite provides a highly integrated design environment with a completely new generation of system-to-IC-level features, including high-level synthesis, analytical place-and-route and an advanced timing engine. These tools enable developers to increase design integration and implementation productivity.

VIVADO IP INTEGRATOR: ACCELERATED TIME TO IP CREATION AND INTEGRATION

To accelerate the creation of highly integrated, complex designs in All Programmable FPGA devices, Xilinx has delivered the early-access release of the Vivado IP Integrator (IPI). Vivado IPI accelerates the integration of RTL, Xilinx® IP, third-party IP and C/C++ synthesized IP. Based on industry standards such as the ARM® AXI interconnect and IP-XACT metadata for IP packaging, Vivado IPI delivers intelligent correct-by-construction assembly of designs co-optimized with Xilinx All Programmable solutions.

Built on the foundation of the Vivado Design Suite, IP Integrator is a device- and platform-aware interactive, graphical and scriptable environment that supports IP-aware automated AXI interconnect, one-click IP subsystem generation, real-time DRC, interface change propagation and a powerful debug capability. When targeting a Zynq™-7000 All Programmable SoC, embedded design teams can now more rapidly identify, reuse and integrate both software and hardware IP targeted for the dual-core ARM processing system and high-performance FPGA fabric. To obtain an early-access license, please contact your local sales representative.

To see a video of the IP Integrator creating an IP subsystem, please visit *http://www.xilinx.com/training/vivado/creating-ip-subsystems-with-vivado-ip-integrator.htm*.

VIVADO HIGH-LEVEL SYNTHESIS LIBRARY ENHANCEMENTS

To accelerate C/C++ system-level design and high-level synthesis (HLS), Xilinx has enhanced its Vivado HLS libraries with support for industry-standard floating-point math.h operations and real-time video-processing functions. More than 350 active users and upwards of 1,000 customers evaluating Vivado HLS will now have immediate access to video-processing functions integrated into an OpenCV environment for embedded vision running on the dual-core ARM processing system. The resulting solution enables up to a 100x improvement in the performance of existing C/C++ algorithms through hardware acceleration. At the same time, Vivado HLS accelerates system verification and implementation times by up to 100x compared with RTL design entry flows.

When targeting a Zynq-7000 All Programmable SoC, design teams can now more rapidly develop C/C++ code for the dual-core ARM processing system, while compute-intensive functions are automatically accelerated in the high-performance FPGA fabric.

VIVADO DESIGN SUITE DEVICE SUPPORT

Vivado now supports Zynq-7000 All Programmable SoC devices, including the 7Z030 and 7Z045 (support requires early access to the IP Integrator). The Vivado Design Suite supports all 7 series devices and the 2013.1 release includes these updates:

- Production-ready: Virtex®-7 7VX690T, 7VX1140T, 7VX330T, 7VX415T and 7VX980T
- Defense-grade: Kintex™-7Q (7K325T and 7K410T) and Virtex-7Q (7V585T and 7VX485T)
- General ES ready: Virtex-7 7VH580T and 7VH870T

The Vivado Design Suite, WebPACK™ Edition is a free download that provides support for Artix™-7 (100T and 200T) and Kintex-7 (70T and 160T) devices, and for Zynq-7000 All Programmable SoC (early-access Z010, Z020 and Z030) devices.

Q: ***WHAT ARE THE DIFFERENT EDITIONS OF THE VIVADO DESIGN SUITE?***

A: The Vivado Design Suite is now available in three editions: Design, System and WebPACK—the no-cost, device-limited version of the Vivado Design Suite Design Edition.

In-warranty Vivado Design Suite customers will receive the ISE® Design Suite at no additional cost.

The Vivado and ISE Design Suites are now separate downloads and installers. The Vivado Design Edition includes the ISE Embedded Edition; the Vivado System Edition includes the ISE System Edition, which also includes Vivado High-Level Synthesis and System Generator for DSP.

For license generation, please visit *www.xilinx.com/getlicense*.

Q: ***SHOULD I CONTINUE TO USE THE ISE DESIGN SUITE OR MOVE TO VIVADO?***

A: ISE Design Suite is an industry-proven solution for all generations of Xilinx's All Programmable devices. The Xilinx ISE Design Suite continues to bring innovations to a broad base of developers, and extends the familiar design flow for 7 series and Xilinx Zynq-7000 All Programmable SoC projects. ISE 14.5, which brings new innovations and contains updated device support, is available for immediate download.

The Vivado Design Suite 2013.1, Xilinx's next-generation design environment, supports 7 series devices and Zynq All Programmable SoC (early-access) devices. It offers enhanced tool performance, especially on large or congested designs.

Xilinx recommends that customers starting a "new" design contact their local FAE to determine if Vivado is right for the design. Xilinx does not recommend transitioning during the middle of a current ISE Design Suite project, as design constraints and scripts are not compatible between the environments.

For more information, please read the Vivado 2013.1 and ISE 14.5 release notes.

Q: ***IS VIVADO DESIGN SUITE TRAINING AVAILABLE?***

A: Vivado takes full advantage of industry standards such as powerful interactive Tcl scripting, Synopsys Design Constraints, SystemVerilog and more. To reduce your learning curve, Xilinx has rolled out new instructor-led classes that will show you how to use the Vivado tools. To learn more about instructor-led courses, please visit *www.xilinx.com/training*.

Q: ***IS THERE A WAY TO EXPLORE THE FEATURES OF THE VIVADO DESIGN SUITE ONLINE?***

A: Vivado Quick Take Tutorials provide a fast review of specific features of the new design environment. Topics include flow overview, system-level design, synthesis, design analysis, I/O planning and high-level synthesis. New topics are added regularly. Visit *www.xilinx.com/training/vivado*

Xpress Yourself in Our Caption Contest

DANIEL GUIDERA

Heavens to Murgatroyd! If you're looking to Xercise your funny bone, here's your opportunity. We advise readers to duck for cover before taking on our verbal challenge and submitting an engineering- or technology-related caption for this cartoon showing an asteroid about to crash into an engineering lab. The image might inspire a caption like "3D printing experiment run amok."

Send your entries to *xcell@xilinx.com*. Include your name, job title, company affiliation and location, and indicate that you have read the contest rules at *www.xilinx.com/xcellcontest*. After due deliberation, we will print the submissions we like the best in the next issue of *Xcell Journal*. The winner will receive an Avnet Zedboard, featuring the Zynq™-7000 All Programmable SoC (*http://www.zedboard.org/*). Two runners-up will gain notoriety, fame and a cool, Xilinx-branded gift from our swag closet.

The contest begins at 12:01 a.m. Pacific Time on April 15, 2013. All entries must be received by the sponsor by 5 p.m. PT on July 1, 2013.

So, get writing!

WARREN TUSTIN, a design engineer at Agilent Technologies (Colorado Springs, Colo.), won a shiny new Avnet Zedboard with this caption for the Tarzan cartoon in Issue 82 of *Xcell Journal*:

"It always was a bit noisy in the lab at the start of swing shift."

Congratulations as well to our two runners-up:

"Tarzan switches careers from ape man to code monkey."

– Joseph Trebbien, software engineer

"You know things are slow in the Valley when the headhunters start making house calls."

– David Pariseau, Technology Plus, Inc. (Los Altos, Calif.)

Trust Synopsys' FPGA Synthesis Solutions to deliver the fastest time-to-market for your FPGA design

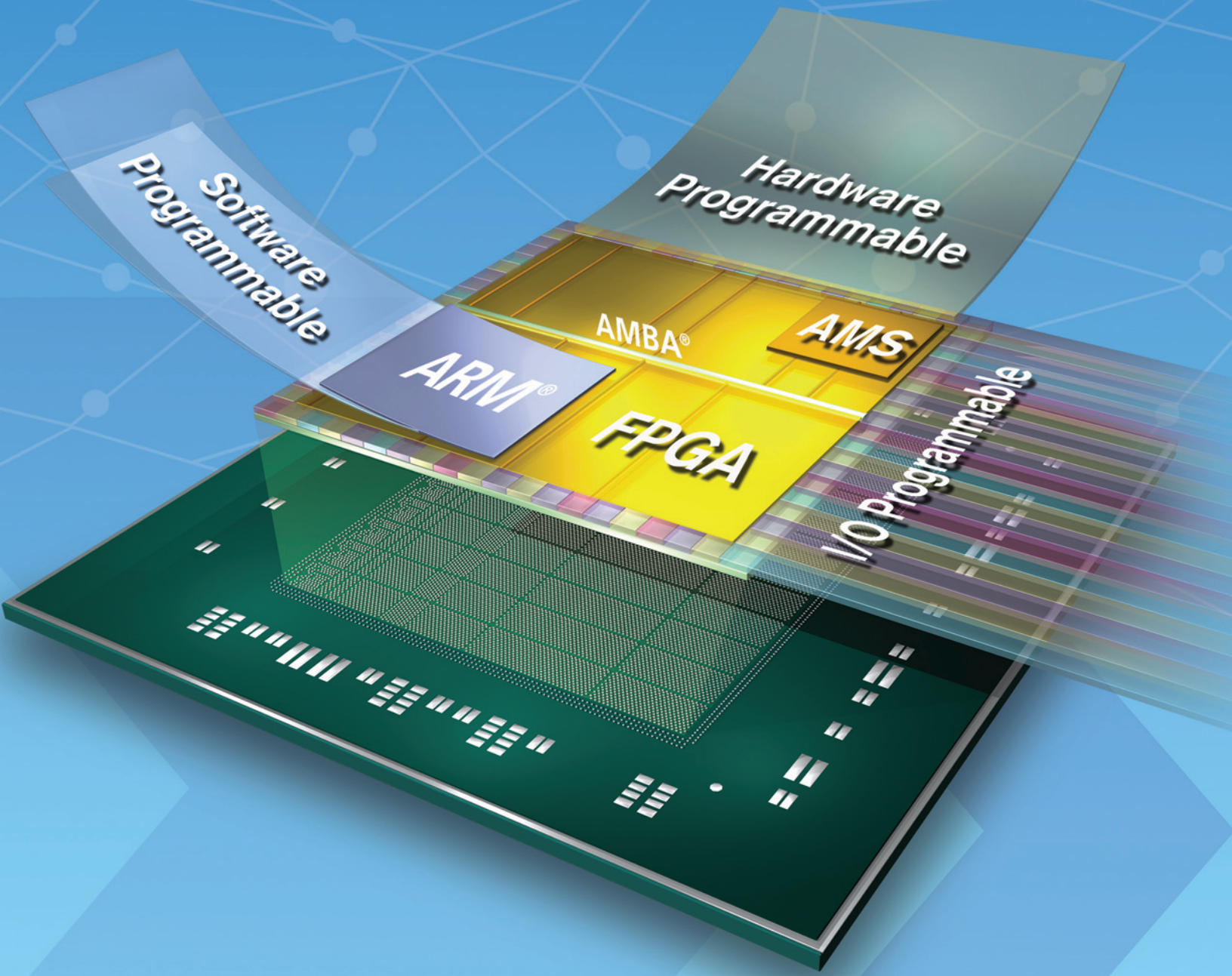
FPGAs keep getting bigger, but your schedule is not. There is no time to waste on numerous design iterations and long tool runtimes. Use the hierarchical and incremental techniques available in Synopsys' Synplify® software to bring in your schedule and meet aggressive performance goals.

To learn more about how Synopsys FPGA design tools accelerate design bring-up, visit **www.synopsys.com/fpgafastturnaround**.

SYNOPSYS®
Accelerating Innovation

A Generation Ahead

Hardware, Software and I/O Programmable SoC



LEARN MORE

XILINX
ALL PROGRAMMABLE™

PN 2539