ISSUE 91, SECOND QUARTER 2015

# Xcell journal

SOLUTIONS FOR A PROGRAMMABLE WORLD

## All Programmable Abstractions: Programming Your Way






XILINX
ALL PROGRAMMABLE™

www.xilinx.com/xcell

AVNET®

# X-fest On-Demand

PRESENTED BY AVNET

Accelerate | Growth | Technology

# Unrivaled Technical Training for FPGA, SoC, DSP & Embedded System Designers

**Don't miss the online technical training program featuring 12 technical courses, exhibits and demos, and a one-of-a-kind community forum**

For the first time ever, Avnet and Xilinx are excited to bring you X-fest On-Demand! Don't miss this opportunity to learn about recent innovations in technology and product development methodology from Xilinx and to experience a deep dive with Avnet technical experts into the key technology domains where Xilinx All Programmable products deliver unique, high-impact (and cost-savings) capabilities.

**TECHNICAL TRACKS ON:**

- **Design Essentials**
- **Techniques & Applications**
- **SoCs & IoT**

Visit the new site at www.xfest2014.com

XILINX ALL PROGRAMMABLE™

ANALOG DEVICES

life.augmented

TE connectivity Authorized Distributor

Texas Instruments

freescale

maxim integrated™

Micron

MathWorks

ON Semiconductor

samtec

SILICON LABS

SPANSION

Accelerating Your Success™

# Integrated Hardware and Software Prototyping Solution


HAPS and ProtoCompiler accelerate software development, HW/SW integration and system validation from individual IP blocks to processor subsystems to complete SoCs.

- Integrated ProtoCompiler design automation software speeds prototype bring-up by 3X
- Enhanced HapsTrak I/O connector technology and high-speed time-domain multiplexing deliver the highest system performance
- Automated debug captures seconds of trace data for superior debug visibility
- Scalable architecture supports up to 288 million ASIC gates to match your design size

To learn more visit: **www.synopsys.com/HAPS**

SYNOPSYS®
Accelerating Innovation

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 Judy Gelwicks
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/



# Setting the Stage for Improved Design Team Productivity

Welcome to the spring 2015 issue of *Xcell Journal*. Among the many great articles in these pages, you'll find three relating to a strategy Xilinx has dubbed "All Programmable Abstractions." The term refers to a new breed of high-level design-entry environments from Xilinx and Alliance members that enable the use of familiar software-programming models in FPGA design. These development environments make it easier for design teams to become productive and even enable those who have never programmed a Xilinx® All Programmable FPGA or SoC to use those devices without assistance from a hardware engineer.

In the cover story, I describe the evolution of Xilinx's development environments to better enable high-level design entry. This All Programmable Abstractions strategy started with Xilinx's release of the Vivado high-level synthesis (HLS) tool in 2012, as well as third-party design environments from Alliance members such as The MathWorks and National Instruments. The initiative has grown over the years, most recently with Xilinx's launch of three new development environments—SDNet™, SDAccel™ and SDSoC™— in our new SDx development environment line.

With these new development environments, entire design teams can be more productive and can create well-rounded systems upfront, at the architectural level. This saves time at the back end and gives teams a major head start on software development. But perhaps more impressively, these environments will also enable new users to add Xilinx FPGAs and SoCs to their systems arsenals and create new applications and innovations without assistance from FPGA experts. According to recent industry estimates, software engineers outnumber hardware engineers 10 to 1 worldwide. Thus, the SDx environments will enable a much wider audience than ever before to produce innovations based on our All Programmable devices.

This issue also contains two practical examples of the SDAccel development environment in action. You'll find those articles on pages 38 and 42.

Elsewhere in the issue, we are excited to showcase a contribution from a living legend in the computer-engineering field, Niklaus Wirth, on a high-level language he developed and ported to a Spartan®-3 FPGA platform. For those of you who don't recognize the name, Professor Wirth invented the Pascal language and several successors in an effort to turn programmers into system inventors.

Professor Wirth, now retired, revised and updated his book *Project Oberon* to help academics teach system programming to the next generation of computer science professionals. The processor he targeted for that lesson in his first version of the book is no longer in production. Failing to find any suitable commercial alternatives, Wirth designed his own using the low-cost, and thus priced-right-for-students, Spartan-3 development board from Digilent.

His article describes his design experience using the Spartan-3 board to revamp his Oberon programming language in hopes of inspiring a new generation of innovators.

Finally, in a great how-to article on page 54, Xilinx's Daniel Michek walks readers through use of the Vivado™ System Generator for DSP tool to create optimized hardware platforms.

I hope you enjoy the issue.

Mike Santarini
Publisher

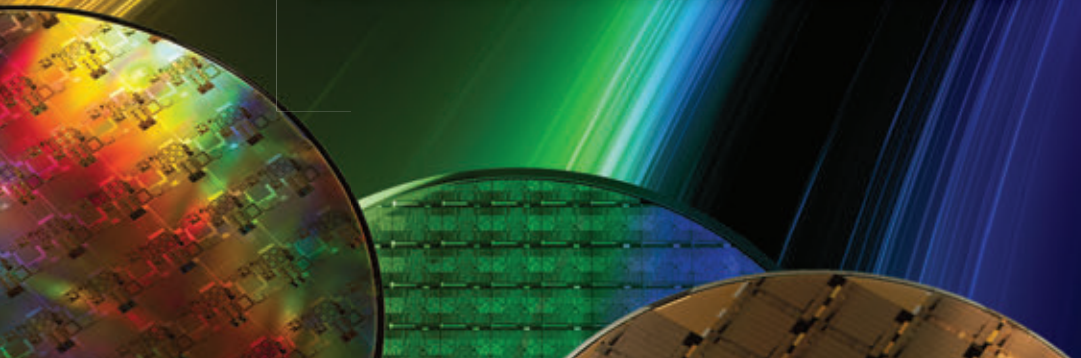
# Synplify Premier

## Achieve Best Quality of Results and Maximum Performance 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 and area goals.

- Unmatched quality of results
- Fastest turnaround time
- Fast and easy debug
- Results preservation from one run to the next

To learn more about how Synopsys FPGA design tools accelerate time to market, visit **www.synopsys.com/fpga**



SYNOPSYS®
Accelerating Innovation

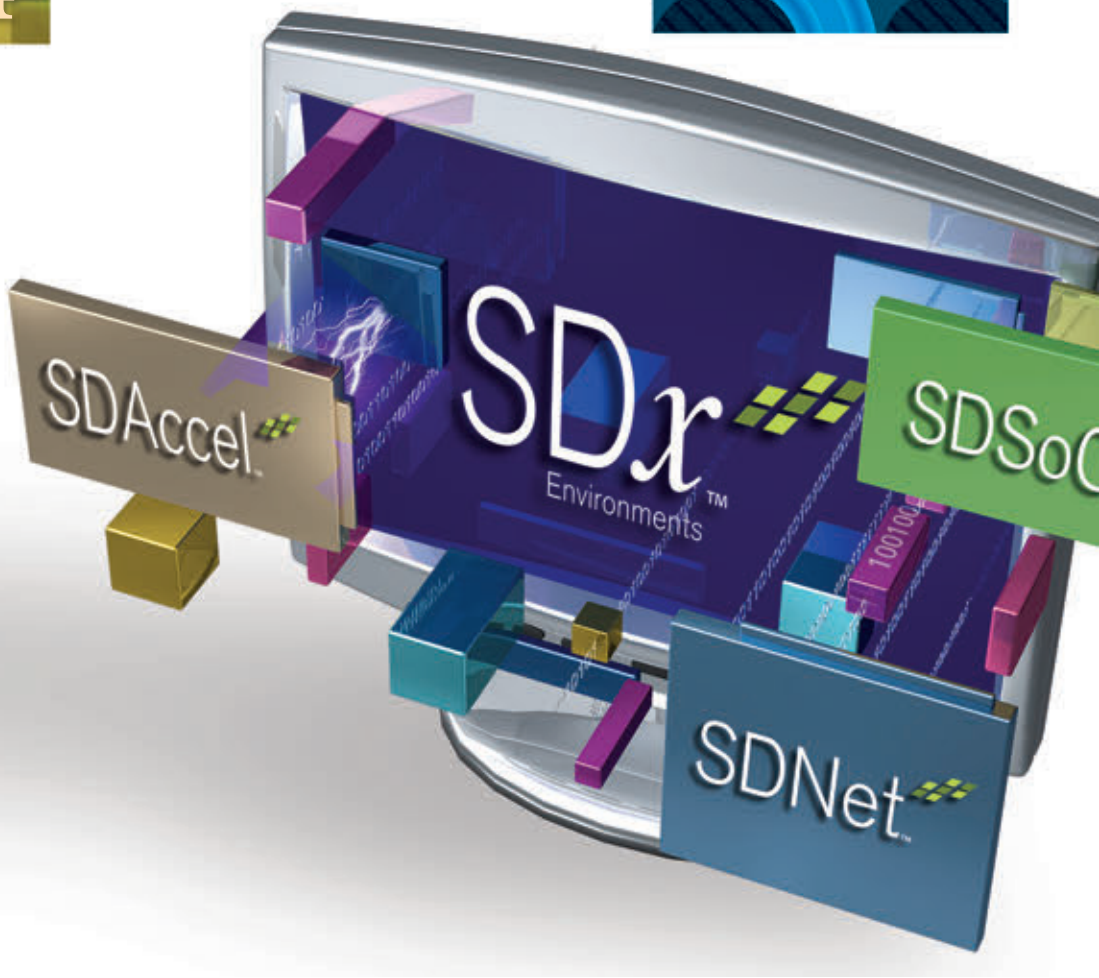
## VIEWPOINTS




## XCELLENCE BY DESIGN APPLICATION FEATURES








# Cover Story 8

All Programmable Abstractions:

Programming Your Way

## THE XILINX XPERIENCE FEATURES

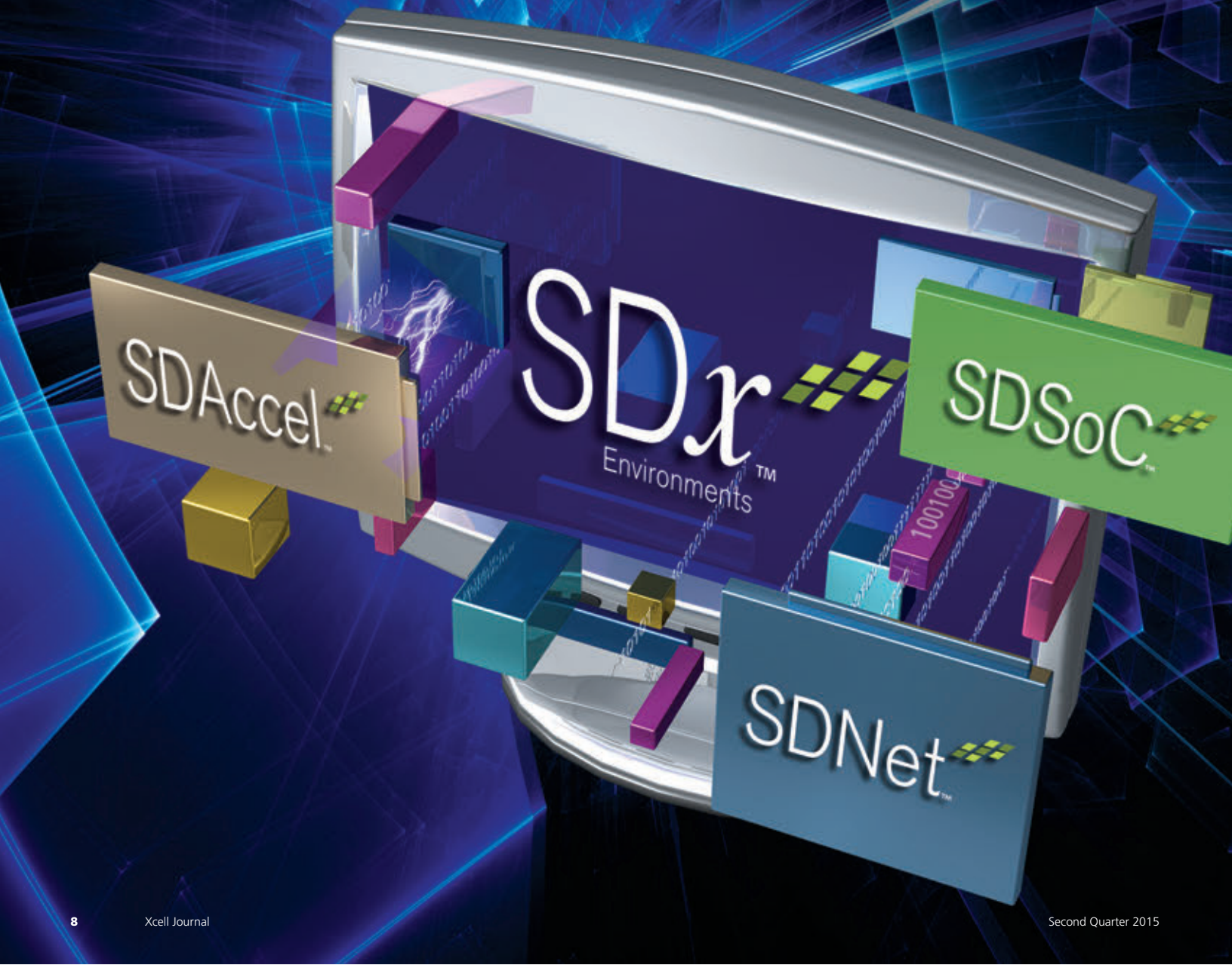





## XTRA READING

# All Programmable Abstractions: Programming Your Way

Xilinx's new SDx software-defined environments complement Vivado IPI, HLS and popular system-level design tools.



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

To enable new levels of design team productivity and expand the reach of its All Programmable FPGAs, SoCs and 3D ICs to a much larger user base of software engineers, Xilinx® Inc. recently unveiled two new additions to its SDx™ development environment family. The new SDAccel™ development environment enables data center equipment programmers—with no FPGA experience—to program Xilinx FPGAs for data center and cloud computing infrastructure equipment using OpenCL™, C or C++. The resulting FPGA-based equipment will offer far superior performance per watt (performance/watt) than GPU- and CPU-based equipment. Xilinx has also unveiled the SDSoC™ development environment, which enables software developers—again, with no FPGA experience—to create systems in C or C++ targeting Zynq®-7000 All Programmable SoC and UltraScale+™ MPSoC platforms from Xilinx and third-party platform developers.

The SDx environments are the latest deliverables from Xilinx's All Programmable Abstractions campaign. The initiative is designed to enable software engineers and system architects to easily program Xilinx devices with development environments tailored to their design needs.

"The combination of SDNet, SDAccel and SDSoC will provide familiar CPU-, GPU- and ASSP-like programming environments to system and software engineers," said Steve Glaser, senior vice president of the corporate strategy and marketing group at Xilinx. "For the first time, those engineers will be able to derive the unique benefits of All Programmable devices for customized acceleration, 10x to 100x performance per watt and any-to-any connectivity with the required security and safety for the next generation of smart systems. Xilinx is enabling these next-generation systems to be more connected, software defined and virtualized. They must also support software-based analytics and enable more computing in the cloud, often driven by pervasive video and embedded vision. This requires SDx software-defined programming environments and heterogeneous multiprocessing with new UltraScale FPGAs and MPSoCs."

The launch of the new SDAccel and SDSoC environments follows closely behind the spring 2014 release of the SDNet™ development environment, detailed in the cover story of *Xcell Journal* issue 87.

While the new SDx environments enable software engineers and system architects to program the FPGA portion of Xilinx devices, the environments will also enable design teams with hardware engineering resources to be more productive and quickly converge on an optimized system to improve time-to-market. With a working system design in hand, hardware engineers can focus on optimizing FPGA layout and performance for system efficiencies, while software engineers further refine application code.

## ALL PROGRAMMABLE ABSTRACIONS

When Xilinx began planning its 7 series All Programmable device family of FPGAs, 3D ICs and Zynq-7000 All Programmable SoCs back in 2008 under new CEO Moshe Gavrielov, it became evident that the rich capabilities of each of the members of the 7 series and future product lines would enable customers to place Xilinx's devices at the heart of their newest, most innovative products.

These All Programmable devices were far more sophisticated than the

glue-logic FPGAs of Xilinx's earlier years and enabled system functionality and end-product differentiation not achievable with any other architecture. To maximize the value of these new devices and jump ahead of the competition, management knew it was imperative for Xilinx to develop tools and methodologies that would enable system architects and even embedded-software developers—not just FPGA experts—to program Xilinx's newest devices. Further, the company would have to develop design environments for software engineers targeting key growth markets, and tailor those environments to the tools and flows these designers were accustomed to using. It would also be imperative to strengthen alliances with companies like The MathWorks and National Instruments, which already have established environments that enable nontraditional-FPGA users to leverage the power efficiency, performance and flexibility of Xilinx All Programmable devices.

For established customers, providing design environments for every member of the design team would guarantee efficiency and shorten time-to-market. If sophisticated enough, the environments would essentially "democratize" All Programmable FPGA and Zynq SoC design, enabling individual architects and software engineers who have no experience designing with FPGAs to program these devices without help from hardware designers. Worldwide, software engineers outnumber hardware engineers 10 to 1. Thus, Xilinx and Alliance Program partners providing such development environments (or the hardware platforms supporting them) would expand both their user base and their revenue.

That strategy, along with a subsequent development effort to enable software engineers and system architects to program Xilinx devices with environments tailored to their design needs, is what Xilinx calls All Programmable Abstractions (Figure 1).

## VIVADO HLS AND IPI: FIRST STEPS UP IN DESIGN ABSTRACTION

The first serious step up in design abstraction began in 2011 with Xilinx's acquisition of privately held high-level-synthesis (HLS) tool vendor AutoESL. The merger was followed in 2012 by Xilinx's public release of the HLS technology integrated into its ISE® Design Suite and Vivado® Design Suite tool flows. Xilinx selected the AutoESL technology after an exhaustive study by Berkeley Design Automation demonstrated that AutoESL's HLS tool was the easiest to use and offered the best quality of results of all the HLS tools available from the EDA industry. Its origins in the EDA world also meant that the use model for the tool was targeting ASSP and system-on-chip (SoC) system architects and well-rounded design teams who had experience in C and C++ programming along with a working knowledge of hardware-description languages (Verilog and VHDL) and chip design requirements.

With Vivado HLS, such broadly skilled architects and design teams can create algorithms in C and C++ while using Vivado HLS to compile and convert those algorithms into an RTL intellectual-property (IP) block. Then, an FPGA designer can take that block and others in RTL that they've created or licensed, and assemble the IP into a design using Xilinx's IP Integrator (IPI) tool. Next, the FPGA designer takes the assembled design through the Vivado flow, performing HDL simulation, timing and power optimization, placement and routing. Ultimately, the designer generates a netlist/bit file and configures the hardware of the targeted All Programmable device. If the design includes a processor (either the Zynq SoC or a soft MPU core), the configured device is then ready for embedded-software engineers to program.

To help these embedded-software engineers with the programming chore, Xilinx provides an Eclipse-based integrated design environment called the Xilinx Software Development Kit (SDK), which includes editors, compilers, debuggers, drivers and libraries targeting Zynq SoCs or FPGAs with Xilinx's 32-bit MicroBlaze™ soft core embedded in them. Introduced more than a decade ago, the SDK has evolved dramatically as Xilinx's integration of processors on its devices has transitioned

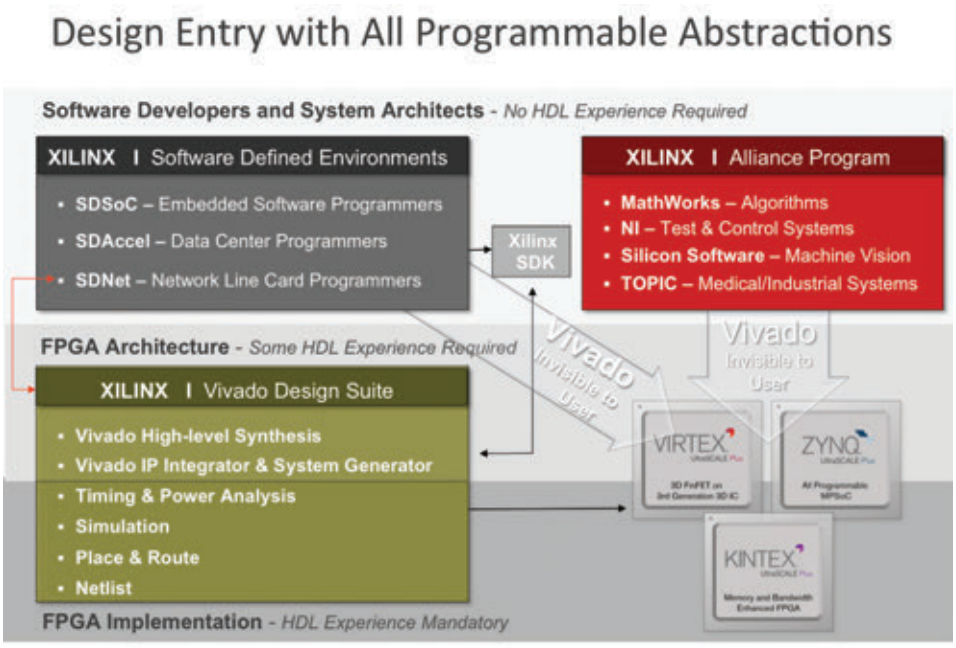



Figure 1 – With the new SDx design environments and those from Alliance members, Xilinx is enabling more innovators to leverage Xilinx All Programmable devices to add greater value to their next-generation products.

from hardened DSP slices and soft-core MCUs and MPUs (8, 16 and 32 bits) to a hardened 32-bit PowerPC® in Virtex®-4 and Virtex®-5 FPGAs; hardened 32-bit ARM® processors in the Zynq SoC; and 64-bit ARM processors in the upcoming Zynq UltraScale+ MPSoC.

## ALLIANCE MEMBERS BRING XILINX VALUE TO MORE USERS

For well over a decade, Xilinx has had very close partnerships with National Instruments and The MathWorks. Both of these companies offer unique high-level development environments tailored to the specific audiences they serve.

National Instruments (Austin, Texas) offers hardware development platforms fanatically embraced by control and test system innovators. Xilinx's FPGAs and Zynq SoCs power the NI RIO platforms. National Instruments' LabVIEW development environment is a user-friendly graphics-based program that runs the Vivado Design Suite under the hood so that National Instruments' customers need not know any of the details of FPGA design. Some perhaps don't even know a Xilinx device is at the heart of the RIO products. They can simply program their systems in the LabVIEW environment and let NI's hardware speed the performance of designs they are developing.

In the case of The MathWorks (Natick, Mass.), more than a decade ago the company added FPGA support to its MATLAB®, Simulink®, HDL Coder and Embedded Coder with Xilinx's ISE and Vivado tools running under the hood and completely automated. As a result, the users—who are mainly mathematician algorithm developers—could develop algorithms and speed algorithm performance exponentially, running the algorithms succinctly on FPGA fabric.

Over a decade ago, Xilinx added an FPGA architecture-level tool called System Generator to its ISE development environment and, more recently, the Vivado Design Suite, in order to enable teams who had FPGA knowledge to further tweak designs for additional algorithm performance gains. This combination of MathWorks and Xilinx technologies has helped customer companies produce thousands of innovative products.

More recently, other companies have begun contributing to the Xilinx environment as well. Xilinx recently welcomed two European companies, TOPIC Embedded Systems and Silicon Software, to its Alliance Program, specifically for their high-level development environments focused on medical and industrial markets.

TOPIC Embedded Systems (Eindhoven, the Netherlands) has a unique development environment called DYPLO, which was featured in *Xcell Journal* issue 89. Targeting the Zynq SoC and, soon, the Zynq MPSoC, the environment takes a singular approach to system-level design, enabling system architects to create a system representation of their design in C or C++ and run it on the Zynq SoC's dual-core ARM Cortex™-A9 processing system. When users find sections of the design that are running too slowly in software, they can drag and drop the sections to a window that converts the C into FPGA logic (running Vivado HLS under the hood) and places it in the Zynq SoC's programmable logic. They swap code snippets back and forth until they converge on an optimal system performance. TOPIC has seen first use in medical-equipment development and is starting engagements with manufacturers of industrial equipment.

Silicon Software (Mannheim, Germany) is the newest addition to the quartet of Alliance members enabling software engineers to leverage Xilinx All Programmable devices. The company's VisualApplets graphical image-processing design environment allows system architects and software engineers targeting Zynq SoC platforms to build new innovations in industrial machine vision applications—and do so without assistance from a hardware engineer. In Xilinx's booth at the SPS Drives trade show in 2013, Silicon Software demonstrated an optical-inspection system it developed with the VisualApplets environment. The demo showed a system performance speed-up of 10x by using the VisualApplets environment to move image-processing tasks from the Zynq SoC's processing system to the device's FPGA logic.

## DEMOCRATIZING FPGA AND SOC DESIGN WITH SDX

With the SDx software-defined development environments, Xilinx is building a series of higher-level design entry environments targeting software developers and system architects in key markets. Both SDAccel and SDSoC run the entire Vivado flow automatically under the hood, and require no direct use of Vivado tools and no hand-off to a hardware engineer. For its part, SDNet does not access Vivado HLS and has a unique two-step use model targeting line card architects.

In the first step, line card architects use an intuitive, C-like high-level language instead of arcane microcode to design the requirements and create a specification for a network line card. The SDNet development environment generates an RTL version of the design based on that specification. The flow then requires a hardware engineer to implement the RTL into the targeted FPGA.

In the second step, SDNet also allows network companies to test and update protocols in the high-level language and upgrade the functionality of the cards—even after deployment in the field—without hardware design intervention. The flow enables companies to quickly create and update cards, flexibility that's ideal for software-defined networks.

The two new environments, SDAccel and SDSoc, take the SDx philosophy into new application areas.

## SDACCEL OPTIMIZES DATA CENTER PERFORMANCE/WATT

According to a March 2014 article in the *Data Center Journal*, huge data centers

that form the heart of companies like Google, Facebook, Amazon and LinkedIn "consume upwards of 3 percent of the world's electric power, while producing 200 million metric tons of CO2." That enormous power consumption costs data centers more than $60 billion a year in electricity fees. The power consumption cuts deeply into the bottom-line profitability of even the biggest dot-com companies but also has an untold cost on the environment.

As more companies look to leverage cloud computing and analysis via Big Data; and as video and streaming media become more pervasive worldwide; and as more people join wireless networks and upgrade toward 5G, there is a relentless, exponentially increasing demand for more data centers with even better performance. The current trajectory, according to the *Data Center Journal* article, will bring data center traffic to 7.7 zettabytes annually by 2017. That means that data center power consumption will rise astronomically if left unchecked. Today the main culprit in this power consumption is the Intel x86 processors that form the bedrock of most data center equipment. MPUs today provide good, but not optimal, performance and high power consumption.

Software engineers, in abundant supply worldwide, find these MPUs the easiest devices to program. To solve the performance portion of the data center problem, many companies have been creating equipment with graphics processing units (GPUs) or CPU systems accelerated by GPU cards. GPUs have performance that's far superior to that of CPUs in data center applications but unfortunately, far worse power consumption. Together, the performance is extremely fast but the power consumption is abysmal.

To get the best of both worlds, many companies are turning to an FPGA-centric approach in which they pair FPGAs with other processors to maximize data center equipment performance.

A number of data center equipment vendors have demonstrated that a discrete FPGA paired with a discrete CPU raises power per card minimally but improves performance dramatically, yielding a significant improvement in performance/watt. Others believe that performance/watt can be further improved with a chip that combines an x86 processor core interconnected to FPGA logic on a single SoC. Some think that a seemingly even lower-power and equally high-performance solution would be to integrate FPGA logic with ARM 64-bit processor IP on a single SoC.

The main deterrent to using FPGAs in the data center has been programming. Data center developers are accustomed to programming x86-based architectures and typically come from a pure software-programming background. A first step designed to help developers move CPU programs to faster GPUs was the industry's open development of the OpenCL language. Over the last two years, OpenCL has evolved even further to enable customers to target FPGAs. This development is opening up new possibilities for future data center equipment architectures and even ubiquitous networks.

By launching the SDAccel environment, Xilinx has bridged that programming gap and paved the way to enabling data center engineers to use OpenCL, C or C++ to program FPGA-based platforms without requiring design intervention from hardware engineers. Tom Feist, senior director of design methodology marketing at Xilinx, said the SDAccel development environment for OpenCL, C and C++ enables data center programmers to build equipment with up to 25x better performance/watt than CPU- and GPU-based systems.

Feist said that in the SDAccel flow (see Figure 2, the SDAccel development environment demo), which targets x86 CPUs paired with acceleration cards running Xilinx's 20nm Kintex® UltraScale FPGAs, software developers identify applications that need acceleration, code and optimize the kernels in OpenCL, and then compile and execute the applications for the CPU. They then cycle to estimate kernels and debug them to come up with cycle-accurate models. Next, they use SDAccel to compile the code and implement it automatically in the FPGA (possible because Vivado runs under the hood). They can then run the application and validate the performance of the applications accelerated with the card vs. performance when running solely on the CPU. "They can run iterations in this cycle until they find the optimal balance between performance

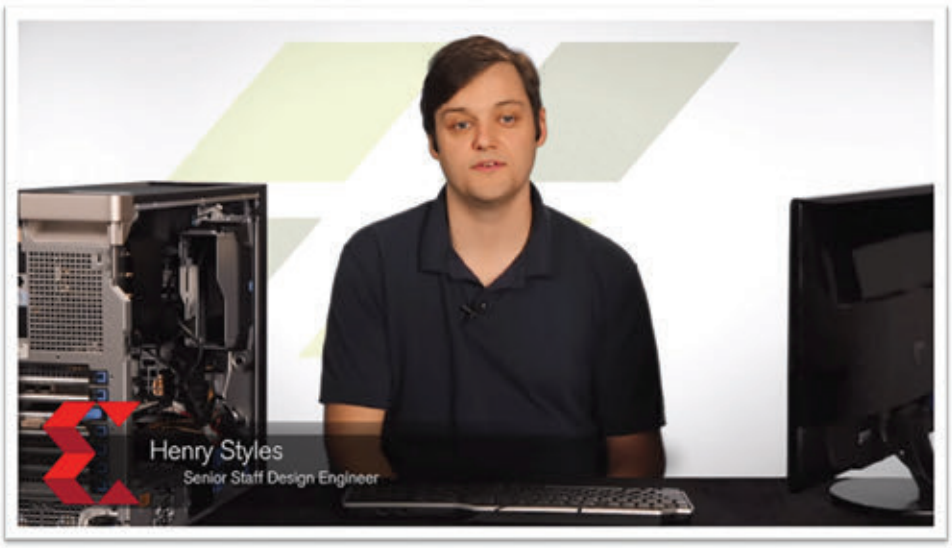



Figure 2 – In this SDAccel development environment demo, engineer Henry Styles shows how to use the SDAccel development environment for acceleration using a standard x86 64-bit workstation containing an Alpha Data ADM-PCIE-7V3 accelerator.


Figure 3 – In this SDSoC development environment demo, principal engineer Jim Hwang uses SDSoC to create a simple image-processing pipeline to detect motion and insert motion edges into a live HD 1080p video stream running at 60 frames per second.

and power and achieve up to 25x performance/watt improvement over CPU and GPU implementations," said Feist.

In this issue of *Xcell Journal*, Feist and colleagues contribute a detailed overview of the SDAccel environment and a second article showing SDAccel in action (see pages 38 and 42).

## SDSOC MULTIPLIES EMBEDDED-SYSTEM INNOVATORS

While SDNet enables line card developers to quickly create next-generation networks with a unique, "softly defined" approach, and while SDAccel allows data center equipment vendors to achieve the best performance/watt of their next-generation data centers, the new SDSoC development environment may well have the broadest impact among Xilinx's user base. That's because SDSoC targets the vast numbers of embedded-system design teams and specifically, their software engineers designing for the majority of markets Xilinx serves with its Zynq SoCs. The SDSoC environment enables users to now configure the logic—not just program the processor of embedded systems running Zynq SoC-based hardware platforms—in C and C++.

"Software developers are accustomed to programming motherboards, ASSP platforms and ASICs without requiring a hardware engineer to do anything," said Nick Ni, SDSoC product manager. "With the SDSoC development environment, they can program Zynq SoC and MPSoC platforms in the same manner as they have with ASSPs. But what's unique is that with SDSoC, they can now create complete system designs in an Eclipse IDE environment using C or C++ targeting the Zynq SoC and Zynq UltraScale+ MPSoC platforms."

Ni said that in using the SDSoC environment, embedded-software developers create their designs in C or C++ and test to see what portions aren't running optimally on the Zynq SoC's processing system. They highlight the suspect code and command the SDSoC environment to automatically partition that code into the Zynq SoC's programmable logic to speed up the system performance. Ni said the SDSoC environment can move a software function to FPGA logic with the click of a button. And it doesn't require a hardware engineer to do it. The compiler in SDSoC will generate the entire Vivado project as well as the bootable software image for the targeted hardware platform.

"We are essentially enabling embedded-software engineers to become system engineers with the SDSoC environment for our Zynq SoCs," said Ni (see Figure 3, the SDSoC development environment demo).

The SDSoC environment leverages a macro compiler that takes the C/C++ code users have designated to accelerate in the Zynq SoC's logic. By running the Vivado Design Suite under the hood, the environment automatically turns the code into an IP block and configures that block into the device's logic, automatically generating a driver in software.

SDSoC provides board support packages (BSPs) for Zynq All Programmable SoC-based development boards including the ZC702 and ZC706, as well as third-party and market-specific platforms, such as the ZedBoard, MicroZed, ZYBO, and video and imaging development kits.

"We'll be adding more BSPs to SDSoC in the coming months, especially as more third-party platform companies develop systems with the Zynq SoC," said Ni. "The SDSoC not only expands the user base for Xilinx but also for companies developing platforms leveraging the Zynq SoC."

Ni is quick to note that while the main goal of the SDSoC environment is to enable the vast number of embedded-software designers to now create entire systems with Xilinx Zynq SoCs, users with traditional FPGA backgrounds and design teams can also greatly benefit from using the environment.

"It enables design teams to quickly design a system architecture in C and C++ and then try different configurations to get the optimal performance they require.," Ni said. "If they do have FPGA designers on their team, they can have them use Vivado tools to optimize the blocks and logic layout further."

For additional news about Alliance member support for the SDSoC environment, see the Xpedite section in this issue (page 66). For further information on Xilinx's All Programmable Abstractions, visit *http://www.xilinx.com/products/design-tools/all-programmable-abstrac tions.html.*

# Enabling a JPEG 2000 Network for Professional Video

**by Jean-Marie Cloquet**
Manager, Image Processing Division
Barco Silex
*jean-marie.cloquet@barco.com*

# A new reference design from Xilinx and Barco Silex offers JPEG 2000 video transport over Internet Protocol networks.

Because of its superior quality, JPEG 2000 has emerged as the standard of choice for the compression of high-quality video, including the transport of video in the contributing networks of television broadcasters. As a result, suppliers of video equipment have started adding JPEG 2000 encoders and decoders to a variety of transport solutions, supporting various interfaces and sometimes even using proprietary protocols.

This trend, however, has locked video service providers into the products of one or a few vendors. A solution to this dilemma arrived in April 2013 with the publication of the Video Services Forum's (VSF) TR-01 recommendation for transport of video over Internet Protocol (IP) networks, a specification for interoperable equipment. Xilinx and Barco Silex, a certified member of the Xilinx® Alliance Program, quickly joined forces to support the interoperability effort.

Barco Silex has now completed a reference implementation of the VSF TR-01 recommendation. The multichannel video-over-IP with JPEG 2000 solution was recently made public on the Xilinx website. It is based on intellectual-property cores from Xilinx and Barco Silex, and is ready to be customized and integrated by broadcast equipment OEMs. Honoring this effort, the National Academy of Television Arts and Sciences awarded Barco Silex a 2014 Technology & Engineering Emmy Award (Figure 1).

## LOOKING FOR SUPERIOR VIDEO COMPRESSION

JPEG 2000 supersedes the older JPEG standard and offers many advantages over its predecessor or other popular formats such as MPEG. By 2004, JPEG 2000 had become the de facto standard format for image compression in digital cinema through the Hollywood-backed Digital Cinema Initiatives (DCI) specification. The possibility of a visually lossless compression makes JPEG 2000 ideal for security, archiving and medical applications.

The broadcasting industry also took notice. Broadcasting and video service companies have huge amounts of live video that has to be transported to post-production and streaming facilities within their so-called contribution networks (Figure 2), without delay or loss of visual quality. Of particular interest for the professional-video industry, therefore, is the possibility of a visually lossless compression—that is, a compression scheme that retains the image quality and still allows efficient storage and transport.

In addition, the other innovations in JPEG 2000 also meant a step forward for the broadcasting industry. Each frame in the video stream is compressed individually as a still frame, in contrast to MPEG formats, which compress frames in groups. This single-frame compression technique results in a low latency but also makes for easy per-frame post-processing and editing. A JPEG 2000 stream may also be partially decompressed and viewed, allowing different applications and viewing experiences from the same stream.

Another big plus is the resilience against transmission errors in the stream. If transmission errors cannot be corrected using forward error correction (FEC), the errors will have a smaller visual impact after decoding compared with other codecs. Finally, JPEG 2000 preserves the image quality even after multiple encoding/decoding processes, which is of capital importance in contribution networks with various stages of video management.

Picking up on this interest, equipment suppliers soon started to implement JPEG 2000 encoders and decoders in their video gear. However, for the transport between locations, they still had a choice among a wide range of implementation options, including proprietary protocols. The drawback for video service providers was that they had to lock into the products of one or a few vendors, instead of setting up the best-matching, cost-efficient infrastructure.

## A RECOMMENDATION TO STANDARDIZE VIDEO TRANSPORT

So there was a clear demand from the service providers for a standardized transport to ensure better interoperability between existing and future equipment. What they needed was a transport that could be best organized over IP networks, which were becoming the prevailing network architecture, with standardized equipment ready for high-throughput data transport. Starting in 2007, the Society of Motion Picture and Television Engineers (SMPTE) published a standard for video transport over IP, which has been expanded since. SMPTE 2022 includes, among others, IP protocols for constant-bit-rate video signals in MPEG-2 transport streams (SMPTE 2022 1&2 for compressed video and SMPTE 2022 5&6 for uncompressed video).

Taking these specifications as its basis, the Video Services Forum in 2013 published its VSF TR-01 document, a technical recommendation titled "Transport of JPEG 2000 Broadcast Profile Video in MPEG-2 TS over IP." The VSF is an international association composed of service providers, users and manufacturers dedicated to interoperability, quality metrics and education for video networking technologies.

Any device that adheres to VSF TR-01 will take its input from an SDI (serial digital interface) signal, the legacy standard for uncompressed point-to-point video transport in the broadcast industry. The device will extract the active video, audio and ancillary data (for example, captions) and compress the video in JPEG 2000 format.

The resulting stream is multiplexed into an MPEG-2 transport stream together with the audio and ancillary data. This stream is again encapsulated according to SMPTE2022 in a Real-time Transport Protocol (RTP) stream and transmitted over IP to a receiving device. The receiver will de-encapsulate the RTP/IP stream, demultiplex the MPEG-2 transport stream, decode the JPEG 2000 and place the video, audio and ancillary data onto the output SDI signal.

## IMPLEMENTING AN FPGA-BASED REFERENCE SOLUTION

In September 2012, even before the VSF recommendation was published, Xilinx and Barco Silex announced a partnership to develop video-over-IP solutions. The goal was to offer a comprehensive platform of hardware-validated intellectual-property cores, reference designs and system integration services. In this effort, Barco Silex took on the role of system integrator, matching cores from Xilinx (SMPTE 2022, SMPTE SDI, Ethernet MACs) with its own high-performance JPEG 2000 and DDR3 memory controller cores. The goal was to enable OEMs of broadcast equipment to accelerate their product development and to add the latest video-over-IP capabilities to their existing products and those currently in development.

In this framework, the partners have now completed a reference design, composed of a four-channel transmitter-and-receiver platform (Figure 3). The transmitter is able to take up to four SDI high-definition (HD) streams (1080p30), optionally compress them with JPEG 2000 and send them over 1-Gbps (with compression) or 10-Gbps (uncompressed) Ethernet according to the VSF TR-01 standard. The receiver platform, conversely, can receive the IP stream, de-encapsulate and decompress it, and put it on up to four SDI HD links.

In the transmitter platform, Xilinx SMPTE SDI cores receive the incoming SDI video streams. On the uncompressed path, these SDI streams are multiplexed and encapsulated into fixed-sized datagrams by Xilinx's SMPTE 2022-5/6 video-over-IP transmitter core and sent out through the Xilinx 10-Gigabit Ethernet MAC (10GEMAC) and 10G PCS/PMA cores.

On the compressed path, the SDI streams first go to the JPEG 2000 encoder for compression. Next, they are encapsulated into MPEG-2 transport stream packets according to VSF TR-01 by the dedicated TS Engine core implemented by Barco Silex. Last, the SMPTE 2022-1/2 video-over-IP transmitter core packs the streams into fixed-size datagrams and sends them out through the 1G TEMAC. Alternatively, the streams can be multiplexed with uncompressed video channels on a 10-Gbit link using the 10GEMAC and 10G PCS/PMA cores.

On the receiver platform, the Ethernet datagrams of the uncompressed streams are collected at the 10GEMAC. The SMPTE 2022-5/6 video-over-IP receiver core filters the datagrams, de-encapsulates and demultiplexes them into individual streams, and outputs the SDI video through the SMPTE SDI cores. The Ethernet datagrams of the compressed streams are collected at the 10GEMAC, de-encapsulated by the SMPTE 2022-1/2 video-over-IP receiver core and by the TS Engine, and fed to the JPEG 2000 decoder. Its output video is converted to SDI and sent to the SMPTE SDI cores.

For each of the four channels, the uncompressed or compressed path can be chosen independently of what happens on the other channels.

## LAYING THE BASIS FOR INTEROPERABLE SOLUTIONS

The companies implemented the reference design in two platforms, one using the Zynq®-7000 All Programmable SoC and the other using the Kintex®-7 FPGA. But the blocks that are used can be integrated in solutions that address the complete range of OEM system requirements, from low-cost, high-volume applications to the most demanding high-performance applications. The intellectual-property cores that were used, such as Xilinx's SMPTE 2022 and Ethernet MAC LogiCORE™ blocks, are available for the full range of Xilinx FPGA systems, up to the UltraScale™ level.

For encoding and decoding, the reference design includes the Barco Silex JPEG 2000 encoder and decoder IP cores. These are silicon-proven, widely adopted single-FPGA solutions for high-performance, simultaneous multi-channel 720p30/60, 1080i, 1080p30/60 and 2K/4K/8K JPEG 2000 encoding and decoding. These cores also support the widest available spectrum of JPEG 2000 options in existence on the market. Essential in stitching multiple video streams together into a smooth, high-data-rate system is

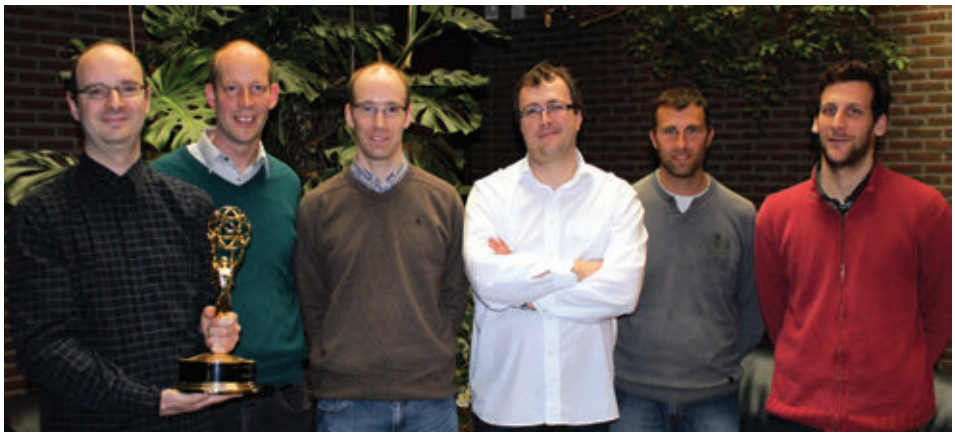


Figure 1 – The Barco Silex video team responsible for the reference design with their 2014 Technology & Engineering Emmy Award for Standardization and Productization of JPEG 2000 Interoperability. From left, they are Luc Ploumhans, Sake Buwalda, François Marsin, Jean-François Marbehant, Jean-Marie Cloquet and Vincent Cousin.

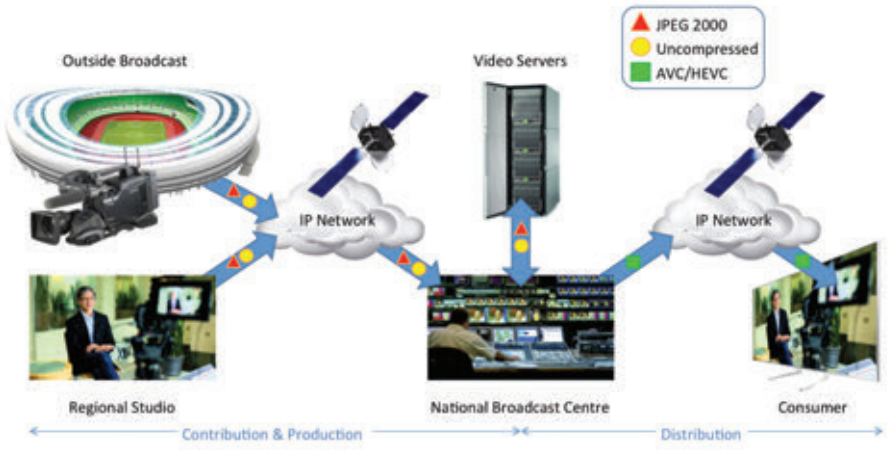



Figure 2 – The contributing networks of broadcasting companies

Barco Silex's DDR3 memory controller. This highly customizable controller is optimized to achieve high bandwidth by reordering accesses and mixing them to the different banks of the SDRAM.

The companies showed a first generation of this reference design in a public interoperability demonstration during the annual VidTrans conference held in February 2014 in Arlington, Va. During this test organized by the VSF, 10 companies (Artel, Barco Silex, Ericsson, Evertz, Imagine Communications, IntoPIX, Media Links, Macnica, Nevion and Xilinx) provided technology and equipment that was interconnected to show live transmission of 720p30 and 1080i60 HD content being compressed in real time using JPEG 2000 encoders and decoders.

A few months later, Barco Silex demonstrated that the reference design could also handle 4K and ultra-high-definition (UHD) signals. As one of the main standards proposed for next-generation video distribution, 4K video carries four times as many pixels as 1080p video, allowing a sharper view and a larger video display. Using the four input channels of the reference design in quad-SDI mode (4K carried over four SDI cables), it is now also possible to take as input a 4K signal and send it over the IP network. This makes the reference design ready for video resolutions up to 4K.

## FPGAS ADVANCE THE VIDEO INDUSTRY

The goal of the collaboration between Xilinx and video specialist Barco Silex was to leverage the power and flexibility of FPGA-based platforms in the professional-video market. By combining the JPEG 2000 cores of Barco and the transport cores of Xilinx, OEMs may produce and update standardized broadcast equipment quickly, making their products future proof in the process.

This reference design arrives at a time when the use of IP networks in the video industry is beginning to take off. The ability of OEMs to capture a share of that new market will depend on how fast they can get products out. With reprogrammable solutions based on Xilinx FPGAs, they can launch products even while standards are still evolving.

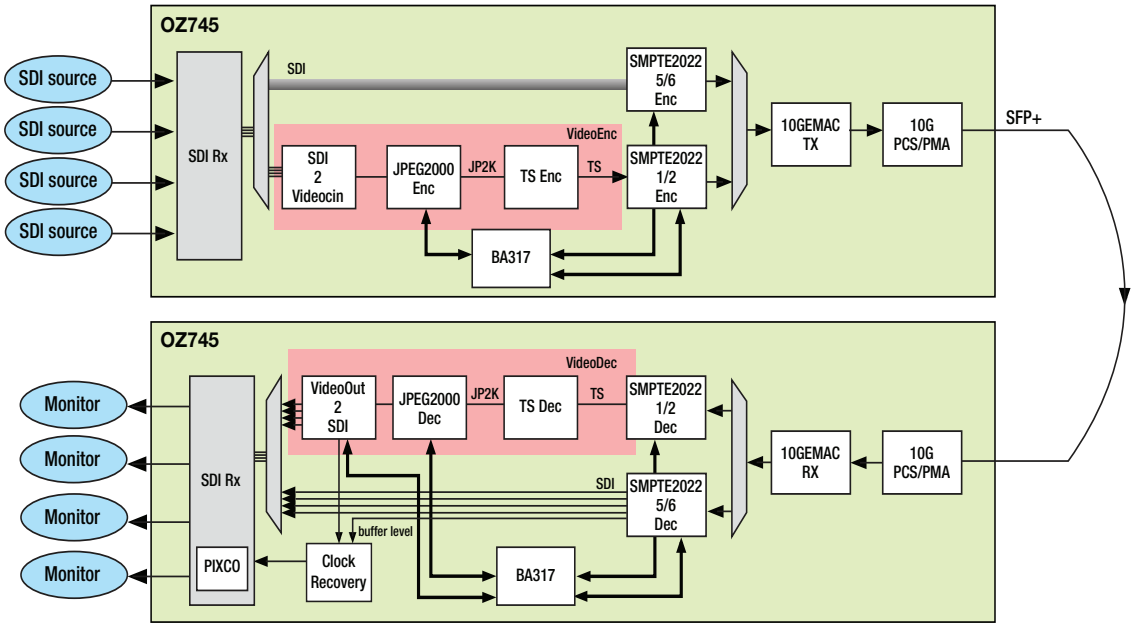
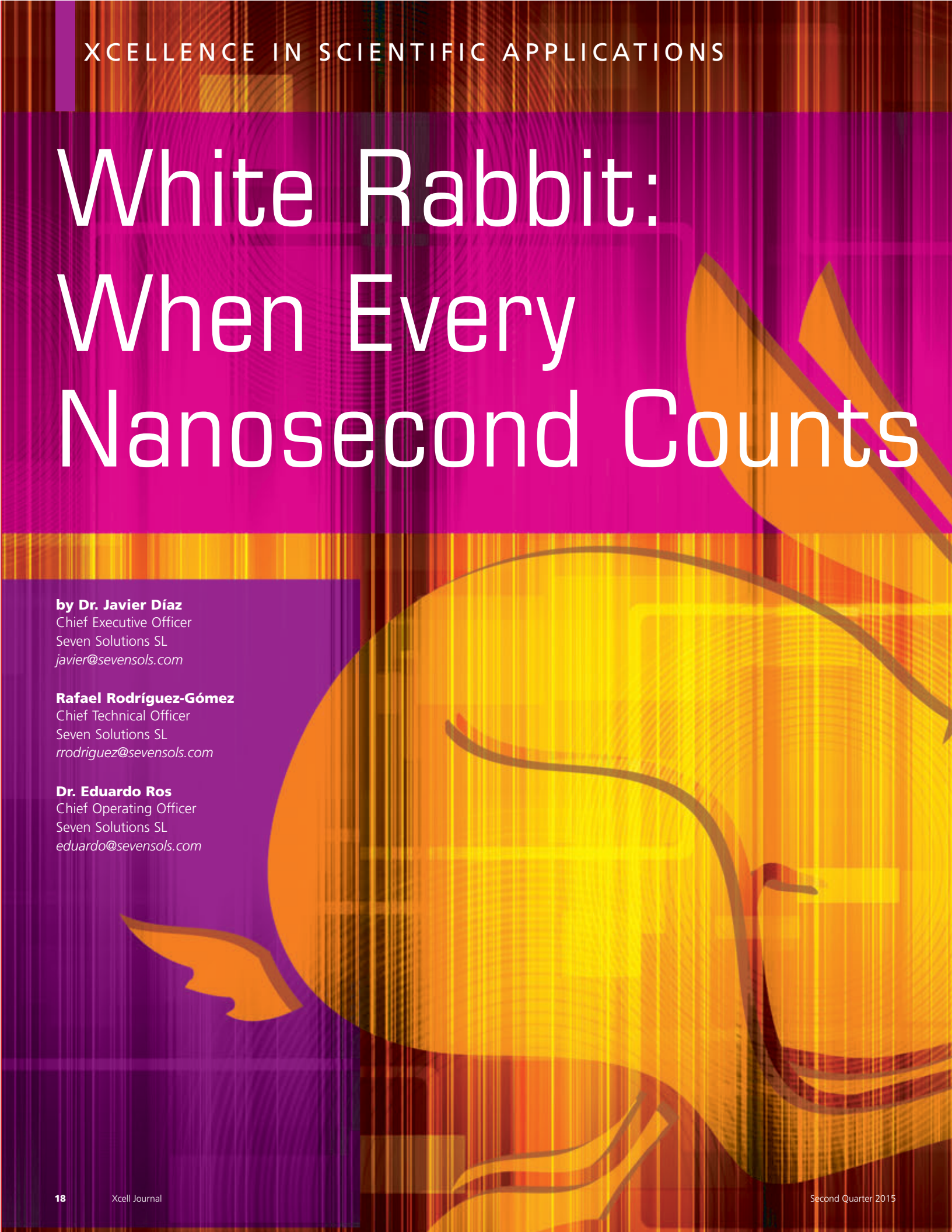



Figure 3 – Schema of the reference design with transmitter and receiver platform

# White Rabbit: When Every Nanosecond Counts

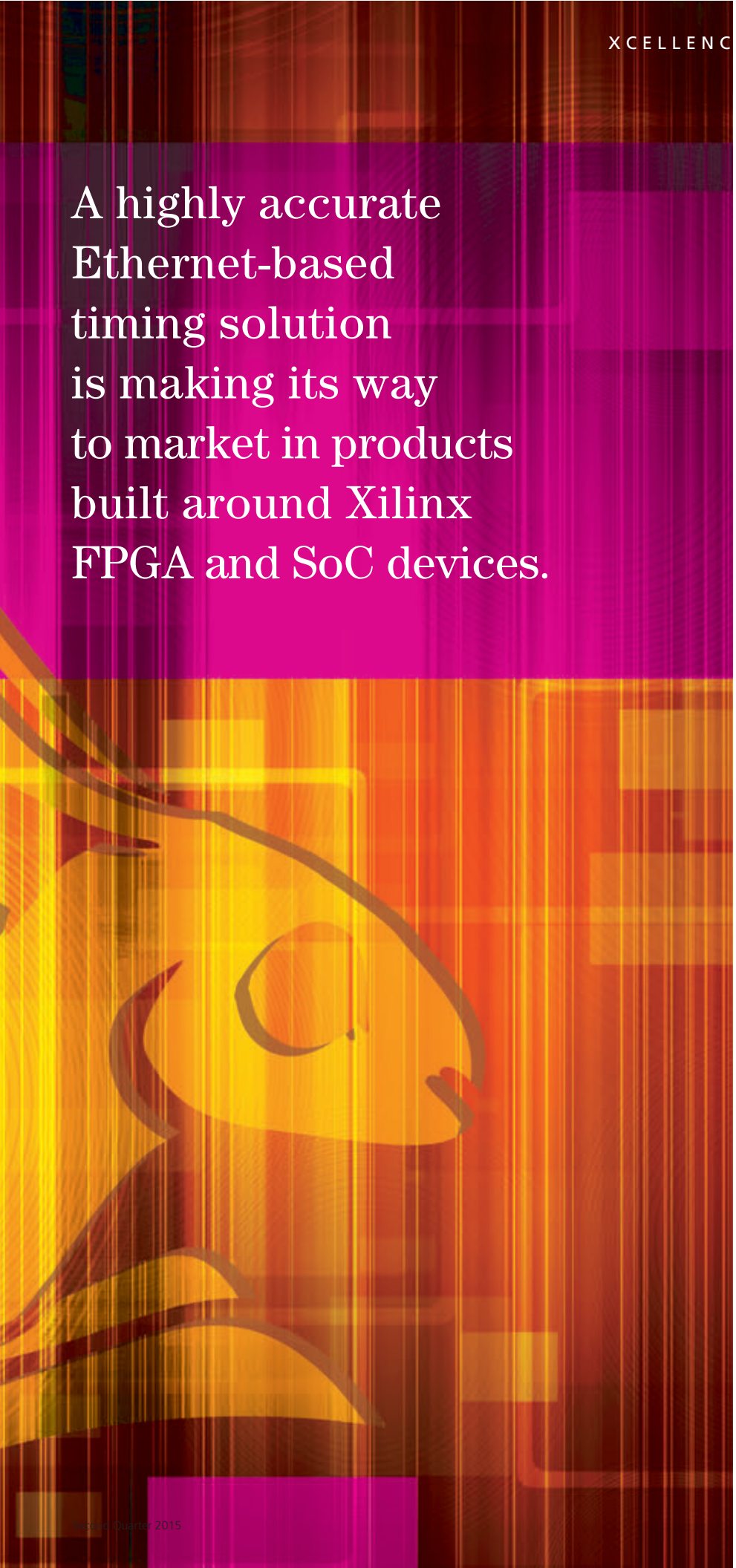


**by Dr. Javier Díaz**
Chief Executive Officer
Seven Solutions SL
*javier@sevensols.com*

**Rafael Rodríguez-Gómez**
Chief Technical Officer
Seven Solutions SL
*rrodriguez@sevensols.com*

**Dr. Eduardo Ros**
Chief Operating Officer
Seven Solutions SL
*eduardo@sevensols.com*

# A highly accurate Ethernet-based timing solution is making its way to market in products built around Xilinx FPGA and SoC devices.



The latest advances in telecommunications and informatics are pushing industrial time-transfer requirements significantly closer to those of scientific applications. For instance, upcoming 100G Ethernet networks and 5G mobile telecom require timing accuracy in the range of a few nanoseconds, while smart grids for electrical-power distribution require submicrosecond accuracy. Time-stamping for high-frequency trading (or stock trading in general) requires reliable mechanisms to distribute time from certified authorities to business centers. Finally, positioning services based on GNSS technologies, such as GPS or Galileo, benefit from high-accuracy synchronization mechanisms.

An Ethernet-based technology called White Rabbit, born at CERN, the European Organization for Nuclear Research, promises to meet the precise timing needs of these and other applications. Named after the time-obsessed hare in *Alice in Wonderland*, White Rabbit is based on, and is compatible with, standard mechanisms such as PTPv2 (IEEE-1588v2) and Synchronous Ethernet, but is properly modified to achieve subnanosecond accuracy. White Rabbit inherently performs self-calibration over long-distance links and is capable of distributing time to a very large number of devices with very small degradation.

Our startup, Seven Solutions SL, has developed the White Rabbit technology since its genesis in 2009 and brings WR product to market using Xilinx® All Programmable solutions. Our latest offering is the ZEN (Zynq® Embedded Node) board, a timing board intended to hold a high-precision reference clock that provides precise tim-

The price may make a solution based on highly accurate clocks, such as chip-scale atomic clocks, too costly for massive utilization.

ing information to other nodes while synchronizing itself in the framework of a White Rabbit network.

## A BRIEF HISTORY OF TIME

Physicists have always understood the importance of time and over the years have devised a wide variety of methods of measuring it. From simple techniques based on scanning the sky (sun clocks, star scanning) to complex mechanisms that rely on the properties of the subatomic world (atomic clocks), scientists have intensively worked toward developing accurate clocks. Existing clocks would neither gain nor lose 1 second in about 300 million years, and this accuracy is key in many applications, for instance for maintaining national metrology labs' time scales.

However, these extremely accurate clocks are expensive, very fragile and take up significant physical space. Therefore, they are not well suited in many real-world scenarios. In fact, most applications rely on electronics that include cheap clocks (crystal oscillators). For a few bucks we can choose among a large set of oscillators with very different specifications.

Oscillators are accurate enough for simple applications. But in many other application fields requiring synchronous communication or a global notion of time to operate simultaneously (distributed instrumentation), these "free-running clocks," not tied to one another, cannot not be used. Although designers can partially solve the problem by installing better oscillators, this is not always technically feasible. Individual clocks are still unsynchronized and even small frequency deviations make this approach invalid.

Yet, the target price may make a solution based on highly accurate clocks (like chip-scale atomic clocks, or CSACs) too costly for massive utilization. In those cases, an alternative approach is to distribute clock information from a reference clock (highly stable and typically expensive) to all the other elements in the network that need to be accurately synchronized. The question is, how can we do that?

## TIME TRANSFER TECHNOLOGIES

When you need to distribute time, there are many possible approaches. Note that distributing frequency (which involves sending the oscillator signal through a wire) is not the same as distributing phase (when events trigger at exactly the same instant in all the elements of a network).

We can solve the first problem (frequency distribution) by using, for example, a coaxial cable or an optical fiber to transmit the clock oscillation. In the second scenario (phase distribution), we can encode a pulse on the wire that transmits each second and use this pulse as a reference to know when a new second starts, a technique typically called a pulse-per-second (PPS) signal.

In addition, there is also a third problem. We may also need to provide the time—not just to have everything ticking at the same time or having the same reference about when to start the counting (phase), but also to ensure we have the same time in all the devices. Therefore, time value can be distributed by propagation of the time information from a central time server and then measuring the propagation time of this message and annotating it in each node. Having these three elements—frequency, phase (PPS) and time—we can say that the network is synchronized.

Current industrial solutions provide these properties in different ways. For instance, GPS devices provide a reference frequency (10 to 50 MHz), a PPS signal and a serial code to provide the time (typically based on the NMEA protocol). This approach is widely used in many systems that require accurate synchronization, since different instrumentation can be easily connected to different GPS receivers. But it uses a significant amount of low-level signals. In power grid applications, these values are provided based on a simple protocol called IRIG-B, which provides time and PPS information. In the past, the IRIG-B approach has been good enough to synchronize the power grid. However, nowadays it can't handle the "smart grid," which is becoming ever-more complex and contains new energy-monitoring applications that demand higher accuracy.

With the quasi-omnipresence of packet networks, previous mechanisms existing on switching networks have evolved and adapted to packet networks. SDH/SONET technology is little by little transforming in solutions based on the Precision Time Protocol (PTPv2, or IEEE-1588v2) plus Synchronous Ethernet (SyncE). PTPv2 is an industrial evolution of Network Time Protocol (NTP), the protocol the Internet uses to synchronize the computers throughout the network. PTPv2 relies on hardware time-stamping mechanisms that significantly improve the accuracy of time synchronization.

The second mechanism, SyncE, makes

it possible to encode the clock signal in the data carrier. In this way, transparently to the user, we are able to distribute time information and frequency to all the devices. Pairing PTPv2 with SyncE enables us to use packet networks for telecommunications seamlessly, and this combination is currently the most popular solution for industrial time distribution on telecommunications, power grid and automation applications. Note that some key problems related to phase propagation and system scalability are critical and remain unsolved.

## SCIENTIFIC APPLICATIONS AND BEYOND

Many applications require that reference clock source information be propagated to different destination points. Scientific facilities are probably the most demanding infrastructures requiring highly accurate time distribution. From CERN's LHC accelerator to large radio astronomy distributed facilities such as CTA, SKA or KM3NeT, all of them require ultra-accurate time and frequency distribution.

But next-generation IT and communications applications will also require highly accurate time transfer that is impossible to achieve with the current standard approaches. In the GPS world, to take one example, the measurement of satellite signal-propagation time is equivalent to the measurement of distances, so positioning and time accuracy measures are closely related. In general, GNSS is vulnerable to the problems of jamming or spoofing. Thus, when used for time distribution it is recommended that critical infrastructures use terrestrial alternatives (based on optical fibers) as complementary redundant mechanisms.

## WHITE RABBIT SOLUTION

White Rabbit (*http://www.whiterabbitsolution.com/*) is an extension of Ethernet for precise timing. It was conceived at CERN in 2009 as an open, collaborative software/hardware initiative, and the technology industry eagerly began participating in its development. Sources are available at the Open Hardware Repository (OHWR, *http://www.ohwr.org*), encouraging development by different companies and research institutions.

From the very beginning, Seven Solutions (*www.sevensols.com*),

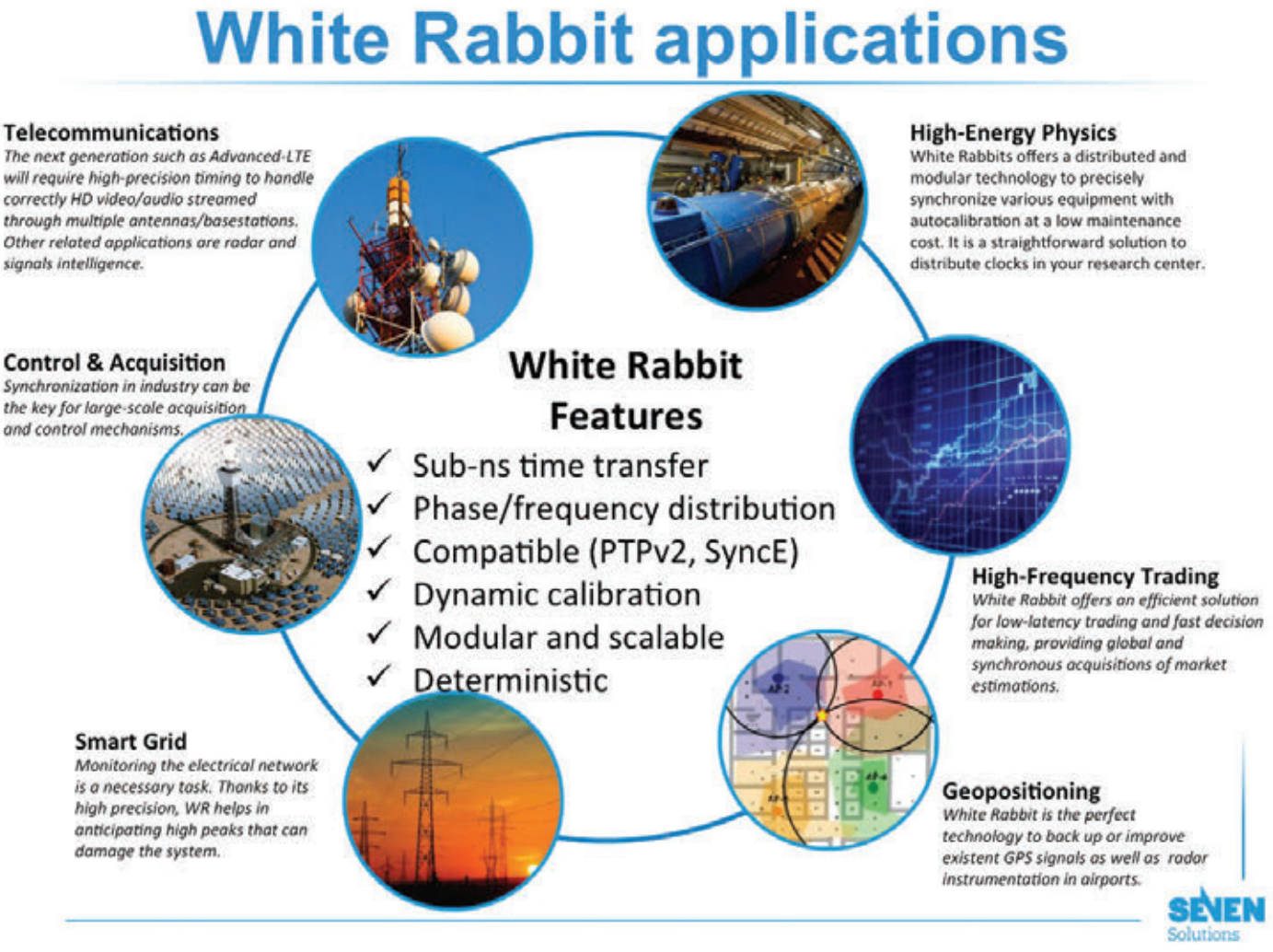



Figure 1 – The White Rabbit applications profile

based in Granada, Spain, has collaborated in the design of White Rabbit (WR) products including not only the electronics but also the firmware and gateware. The company also provides customization and turnkey solutions based on this technology.

As an extension of Ethernet, White Rabbit technology is being evaluated for possible inclusion in the next Precision Time Protocol standard (IEEE-1588v3) in the framework of a high-accuracy profile. Standardization would facilitate WR's integration with a wide range of diverse technologies in the future, as shown in Figure 1.

## INSIGHTS INTO THE WHITE RABBIT TECHNOLOGY

White Rabbit incorporates a number of mechanisms designed to optimize its timing accuracy within the framework of an extension of Ethernet (thus keeping the Ethernet communications structure). WR integrates PTP, Synchronous Ethernet and digital dual-mixer time difference (DMTD) phase tracking.

The new ZEN board from Seven Solutions shows how the key elements of White Rabbit come together in a product (Figure 2). Based on Xilinx's Zynq-7000 All Programmable SoC, the ZEN board includes the White Rabbit Core (WRC), along with a Gigabit Ethernet MAC implementation that's capable of providing a high-accuracy clock. Synchronization mechanisms implemented in the WRC include the following elements:

- Frequency synchronization (synthonization): This is obtained by using SyncE, which encodes the clock signal in the data carrier. To guarantee that all nodes use the same frequency, we employ a mechanism based on a local oscillator disciplined to the external clock that is recovered from the optical link.
- Phase synchronization: The physical clock of a node is retransmitted to the master element and vice versa so that the master can compare the phase of this signal (coming from the slave) with its own phase. The deviation should be equal to the propagation time of the signal through the fiber (properly measured using PTP). Having this information, the master can determine the phase difference between its own clock and the one from the slave, and request that the slave shift its phase to exactly the same value as the master. This process is done digitally by implementing a digital DMTD in the FPGA gateware.
- Time synchronization: This is a consequence of using the PTPv2 protocol, which measures the link propagation times and provides the global notion of time. White Rabbit also takes into account the asymmetries in the propagation time due to the utilization of different wavelengths in a bidirectional optical fiber for each communication transfer (back and forward in the loop), thus improving the accuracy of the standard PTP protocol. Since the frequency and phase have been previously synchronized, we can guarantee the global notion of time in all the devices within the White Rabbit network.

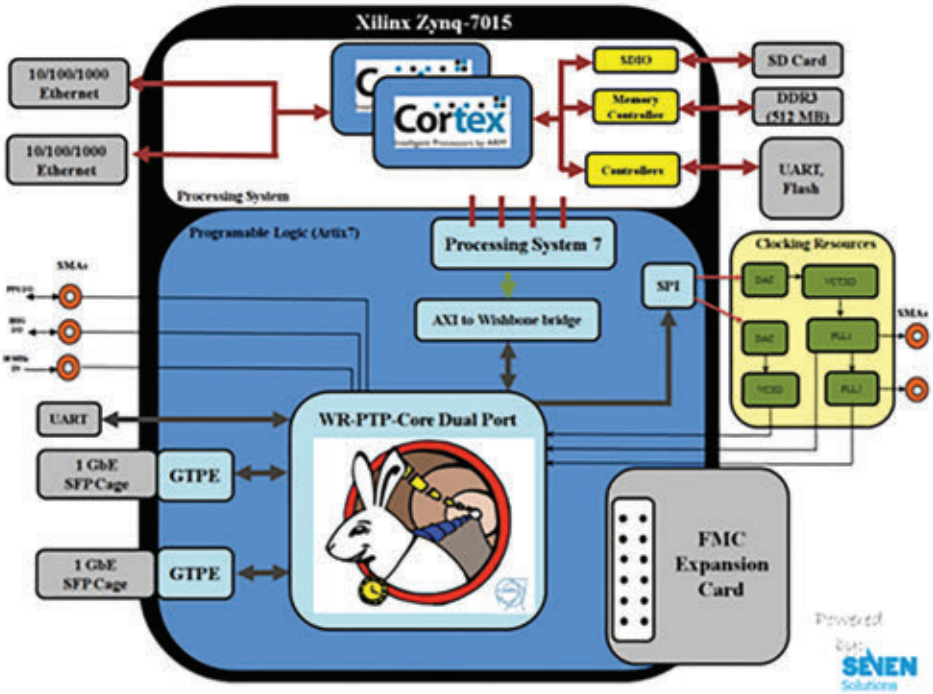



Figure 2 – White Rabbit gateware elements based on a Xilinx Zynq SoC device

All those operations are implemented in the WRC, partially using the proper FPGA gateware and partially using an embedded soft core. The White Rabbit products include the proper oscillators, PLL and timing electronics required to perform these various clock operations.

As a case study, let's take a more detailed look at the ZEN board. This board holds the Dual WRC (D-WRC), a revision of the original WRC developed by Seven Solutions in our new line of Xilinx 7 series products. The D-WRC is able to synchronize two White Rabbit nodes or to serve as an intermediate link in a daisychain network. In addition, the ZEN board includes high-precision, low-jitter and temperature-compensated clock resources controlled by the D-WRC.

Furthermore, the Zynq SoC's dual-core ARM® Cortex™-A9 processor,

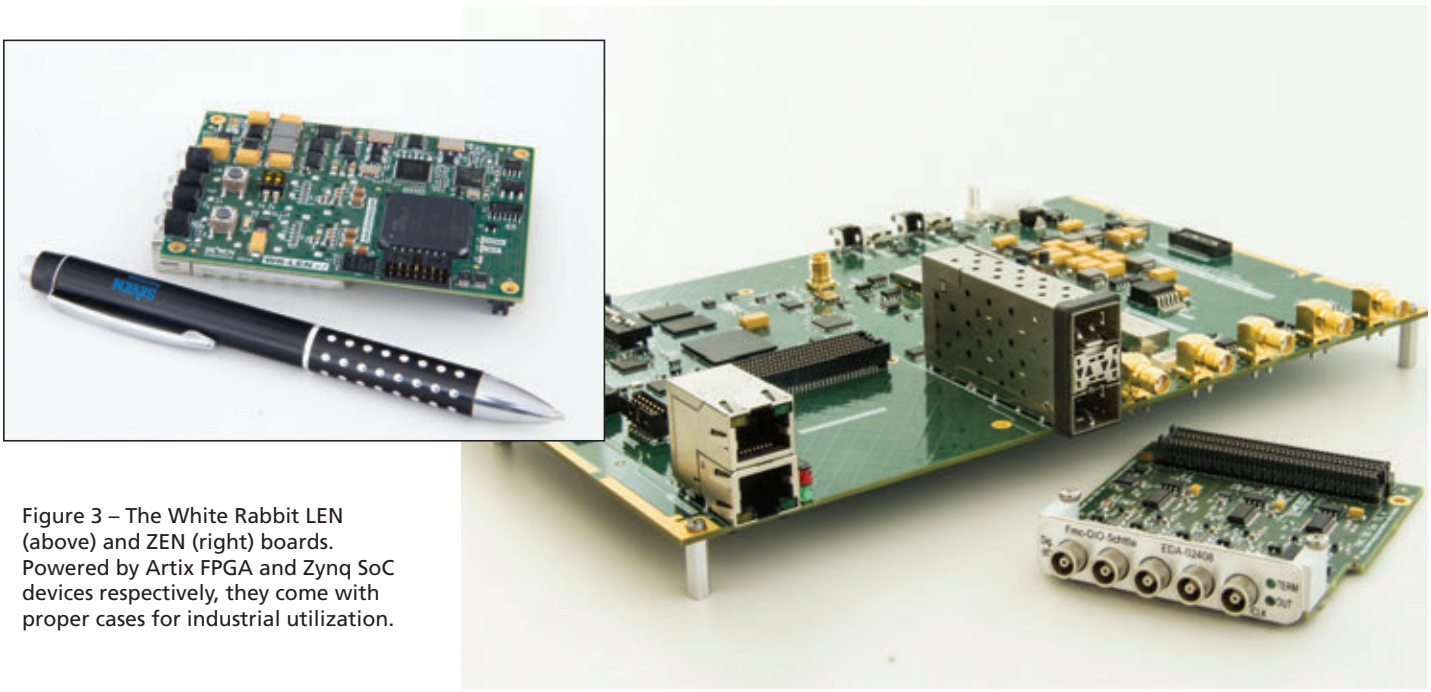


Figure 3 – The White Rabbit LEN (above) and ZEN (right) boards. Powered by Artix FPGA and Zynq SoC devices respectively, they come with proper cases for industrial utilization.

running under the Linux OS, facilitates the development of user applications. Having Linux onboard allows the utilization of new and interesting features such as Web services for configuration, SNMP support for status monitoring and remote firmware load and update.

The ZEN board is intended to function as a high-precision time provider. It offers a large amount of possibilities facilitated by diverse connections and expansions:

- IRIG-B I/O is the time of day used by the ZEN board. It is able to work as master or slave.
- Two 10/100/1000 Ethernet ports connected to the ARM processors can be used for diverse network applications and protocols (NTP, sNTP, PTPv2, management and so on).
- Two SFP cages are provided for plugging in WR-compliant links.
- SMA connectors enable the ZEN board to synchronize itself with more-precise clocks (for example, a GPS source or highly stable oscilators) to provide a diverse set of clocks synchronized by WR.
- An FMC connector makes it possible to plug in one of the mezzanine boards developed in the framework of the WR project or any other industrial board existing in the market. These FMC cards enhance the possibilities of the ZEN and allow many product configurations.
- Memory resources include SD, DDR3 and flash.
- Two UART-USB connectors are included for management and debugging in the D-WRC and Cortex processors.

In brief, the ZEN board offers the end user a node capable of reaching subnanosecond synchronization and of working in daisychain schemes, while also providing the best of the Zynq SoC and the new level of system-design capabilities it affords.

## WHITE RABBIT EQUIPMENT

The White Rabbit technology began its life in the Open Hardware community (Open Hardware Repository, OHWR) promoted by CERN. To accelerate the learning curve with this technology, Seven Solutions developed a White Rabbit Starting Kit consisting of a couple of Spartan®-6-based boards called SPEC, one of which can be configured as a master and the other as a slave. The idea was to encourage users to perform several early-evaluation experiments.

The most complex element of this technology is the switch. We developed (in collaboration with CERN, GSI and other partners) the 18-port White Rabbit Switch, designing the main board in the MicroTCA form factor. The core element was a Virtex®-6 (LX240T) FPGA. We paired this device with an external processor (ARM926E) running an embedded Linux OS to perform the high-level operations such as system updates, file management, etc. The switch uses 18 GTX links for SFPs and 40 GPIOs for general-purpose tasks (LEDS, SFP detection, etc.). This is a significantly complex product capable of distributing timing and also of handling data packets while using standard telecommunication tools.

Recently, Seven Solutions ported the White Rabbit core into the Xilinx Artix® FPGA family in the LEN board (Figure 3), enabling more cost-effective and energy-efficient solutions than existing

OHWR devices. Furthermore, we have just developed a White Rabbit offering based on the Zynq SoC devices. The WR-ZEN node, previously described, represents a complete and versatile system-on-a-chip approach, in which the node and the computer are all integrated in the same board. This solution will facilitate maintenance while reducing cost and improving system flexibility.

Current industrial products developed by Seven Solutions provide standard interfaces for management, configuration and monitoring, taking advantage of all the benefits of White Rabbit technology but with enhanced features, support and documentation.

## WHITE RABBIT APPLICATIONS

The first destination for White Rabbit technology was scientific applications. More recently, this technology has been integrated in several facilities and research projects in the framework of high-energy physics and distributed radio-astronomy facilities. White Rabbit is already being used in several particle accelerators (at CERN and GSI, among other institutions) and is also under consideration at different international scientific initiatives such as KM3NeT, HISCORE and others. Thus, the WR approach has been validated in highly demanding applications where accurate timing and frequency transference over distributed instrumentation on large facilities are critical.

In 2014, White Rabbit was also tested over long-distance links of 125 km by VSL in the Netherlands and 1,000 km by MIKES in Finland.

Accurate timing is demanded in a wide range of applications beyond the scientific fields. The smart power grid requires accurate and reliable timing, while high-frequency trading also relies on accurate and certified timing.

Many of these application domains currently depend on GPS timing signals, which are inherently vulnerable (due to environmental conditions and also accidental or malevolent jamming and spoofing). GPS should not be used for safety-critical infrastructures ("U.S. Air Force Chief Warns against Over-Reliance on GPS," *Inside GNSS News*, Jan. 20, 2010). And in this sense, White Rabbit represents an alternative that allows accurate timing and frequency transfer over terrestrial optical fibers. Standard telecommunication networks can be deployed, making this approach cost-effective.

Beyond the functional features of White Rabbit, Seven Solutions is developing new industrial products that target critical applications requiring high availability. Redundant power sources, hot-swappable fans, holdover oscillators and other techniques will allow their deployment in facilities that cannot afford system failure or long repair processes.

Figure 4 shows how to use WR-LEN as a distributed mechanism to provide

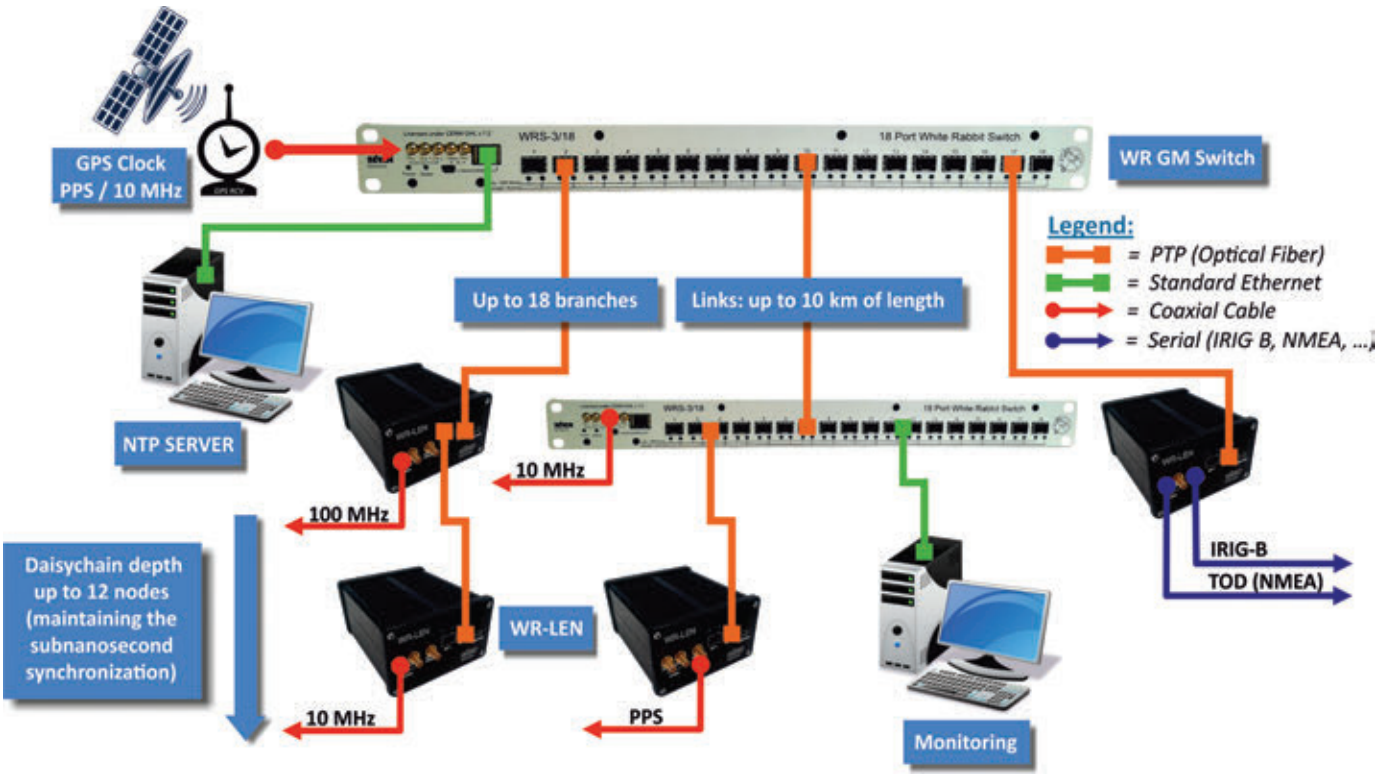



Figure 4 – A White Rabbit network providing accurate timing to different nodes. Daisychain configuration is allowed through WR-LEN nodes.

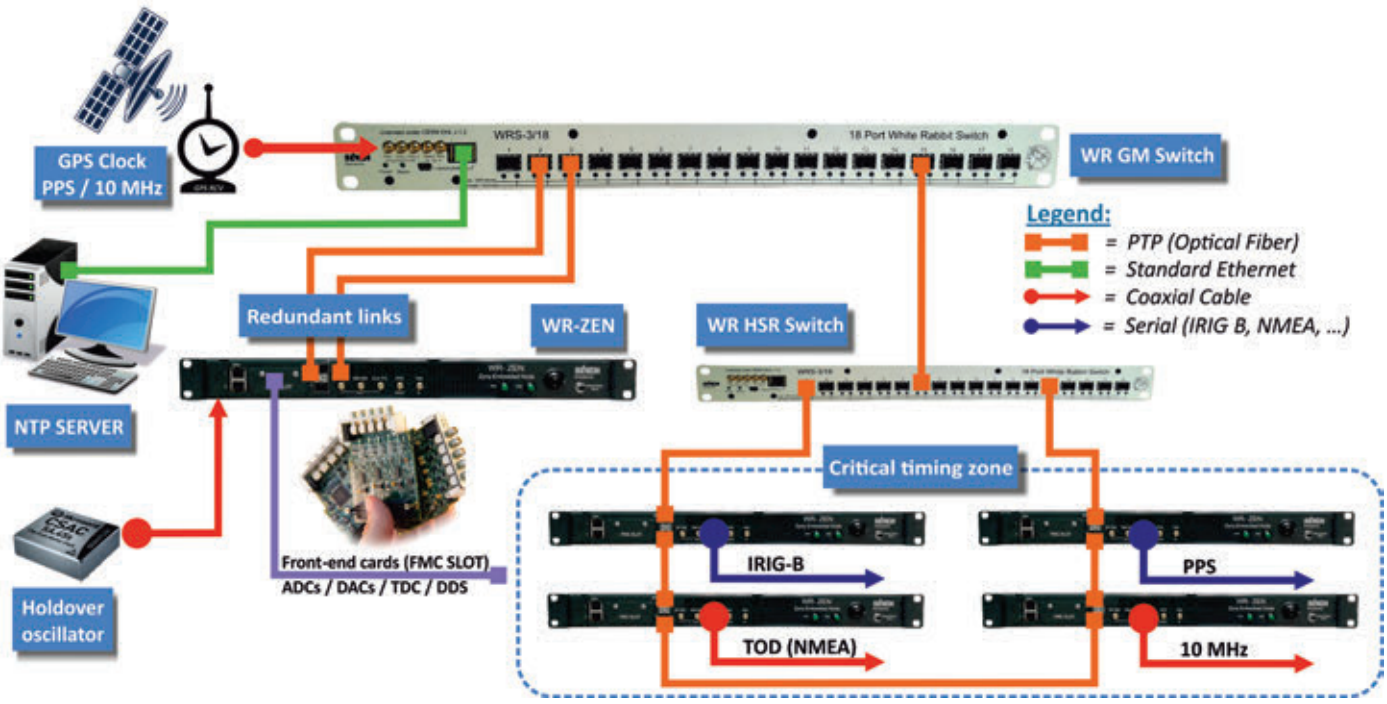



Figure 5 – Network configuration toward safety-critical systems based on the ZEN time provider

timing information in a simple and cost-effective way. The system is able to distribute timing in a GPS-like manner. It provides timing with the IRIG-B output format, which suits power grid applications. In addition, PTPv2 is also an available interface, a valid option since it can be integrated with PTP networks used on modern power grid facilities.

Figure 5 shows a network configuration based on a ZEN-powered time provider. In this case, much more powerful features are present. Redundant power supplies, redundant network topologies and holdover CSAC oscillators on critical elements are possible. Moreover, FMC expansion ports make it possible to add mezzanine cards to further develop additional sensing and control applications.

## FUTURE DEVELOPMENTS

White Rabbit is a promising technology that is able to solve incoming synchronization requirements for different end application fields including smart grids, telecommunications and high-frequency trading. WR solves problems like the phase synchronization and at the same time is able to synchronize clocks with subnanosecond accuracy for tens of thousands of devices distributed along large distances (hundreds of kilometers). As a result, White Rabbit allows ultrahigh-accuracy time transfer and, simultaneously, full data transfer without penalty.

These features, along with White Rabbit's scalability, will allow the development of a world-scale ground-based synchronization mechanism that can be used as back-end technology for GPS solutions (based on ground station antennas) and could open the door to novel applications such as autonomous automobiles or indoor navigation. The upcoming 100G telecom networks will benefit from mechanisms for accurate quality-of-service evaluation, while 5G wireless technologies can rely on WR to solve the well-known phase-synchronization problem.

These are just some examples of future applications in which White Rabbit may have a significant impact. Moreover, thanks to the potential standardization of White Rabbit within the IEEE-1588v3 profile (which is currently under consideration), it will be easy to adapt to multiple vendors. And we think that the most challenging applications are still to arise.

Seven Solutions is providing the first industrial products based on the White Rabbit technology. Our turnkey solutions allow an easy integration into user applications based on standard telecommunication interfaces and can be configured/read just by using standard software such as Web services or SNMP. Next, we will incorporate features for distributing RF generation (without requiring sending the reference frequency) or triggering event acquisition with high accuracy as required for many telecommunication applications.

# Evaluating the Linearity of RF-DAC Multiband Transmitters

Researchers at Bell Labs show how to create a flexible platform for rapid evaluation of RF DACs using Xilinx FPGAs, cores and MATLAB.

**by Lei Guan**
Member of Technical Staff
Bell Laboratories, Alcatel Lucent Ireland
*lei.guan@alcatel-lucent.com*

The wireless communications industry has entered into a new, all-in-one era, with every network operator seeking more-compact and multiband infrastructure solutions. The emerging RF-class data converters—namely, RF DACs and RF ADCs—architecturally make it possible to create compact multiband transceivers. But the nonlinearities inherent in these new devices can be a stumbling block.

For instance, nonlinearity of the RF devices has two faces in the frequency domain: in-band and out of band. In-band nonlinearity refers to the unwanted frequency terms within the TX band, while out-of-band nonlinearity is the undesired frequency terms out of the TX band.

For system engineers who are prototyping multiband transmitters using RF DACs, it is critical to ensure this key component meets the linearity requirement of the standards. Therefore, at the early prototyping stage, a flexible testing platform is fundamentally required to properly evaluate the nonlinear performance of the RF DACs regarding the multiband applications.

Here at Bell Labs Ireland, we have created a flexible software-and-hardware platform to rapidly evaluate the RF DACs that are potential candidates for next-generation wireless systems. The three key elements of this R&D project are a high-performance Xilinx® FPGA, Xilinx intellectual property (IP) and MATLAB®.

A few words here before starting this engineering story. In the design, we tried to minimize the FPGA resource usage while keeping the system as flexible as possible, so we were only interested in implementing the necessary functions. For setting up the whole testing system, we picked the latest Analog Devices RF-DAC evaluation boards (AD9129 and AD9739a) and the Xilinx ML605 evaluation board. The ML605 board comes with a Virtex®-6 XC6VLX240T-1FFG1156 FPGA device, which contains fast-switching I/Os (up to 710 MHz) and serdes units (up to 5 Gbps) for interfacing the RF DACs.

Now, let's take a closer look at how to use Xilinx FPGAs, IP and MATLAB to create this simple but powerful testing platform.

## SYSTEM-LEVEL REQUIREMENT AND DESIGN

The key purpose of this evaluation platform is to stimulate the RF DAC with various user-customized testing-data sequences. For this purpose, we designed two testing strategies: a continuous-wave (CW) signals test (xDDS) and a wideband signals test (xRAM).

Multitone CW testing has long been the preferred choice of RF engineers for characterizing the nonlinearity of RF components. Keeping the same testing philosophy, we created a tunable four-tone logic core based on a direct digital synthesizer (DDS), which is actually using a pair of two-tone signals to stimulate the RF DAC at two separate frequency bands. By tuning the four tones independently, we can evaluate the linearity performance of the RF DAC—that is, the location and the power of the intermodulation spurs in the frequency domain.

CW signal testing is an inherently narrowband operation. To further evaluate the RF DAC regarding wideband performance, we need to drive it with concurrent multiband, multimode signals, such as dual-mode UMTS and LTE signals at 2.1 GHz and 2.6 GHz, respectively. For that purpose, we created an on-chip BRAM array-based data storage core that has two subgroups in which to store the respective dual-band user data for repeated testing.

Figure 1 illustrates the simplified design diagram at the system level. As you can see, we are using a straightforward

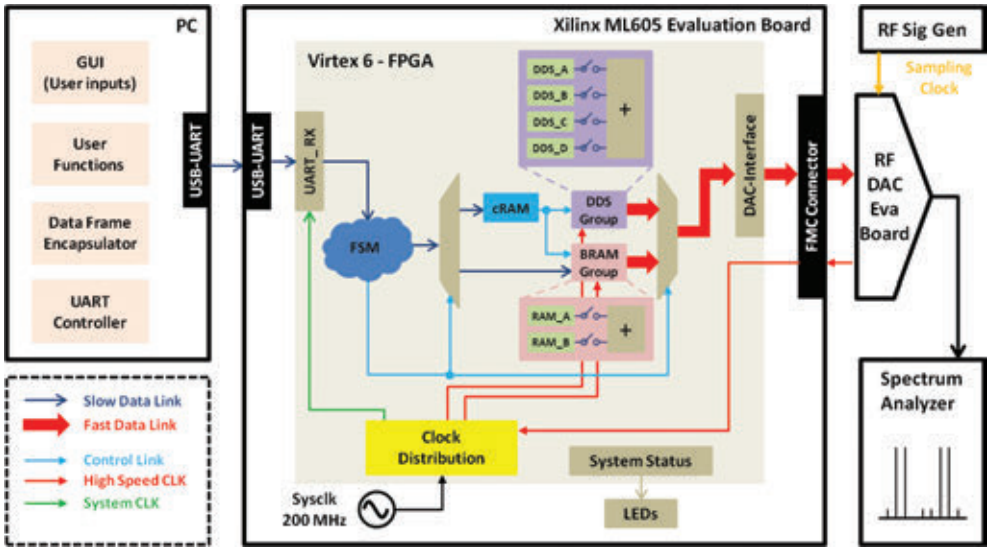



Figure 1 – Simplified block diagram of the platform at the system level

design strategy here, building up the platform as simply as possible and modularizing it with upgrading capability.

## HW DESIGN: XILINX FPGA CORE

The FPGA portion of Figure 1 outlines the implemented logic units the system fundamentally required. They include a clock distribution unit, a state machine-based system control unit and a DDS core-based multitone generation unit, along with two units built around Block RAM: a small BRAM-based control message storage unit (cRAM core) and a BRAM array-based user data storage unit (dRAM core). Also included are a UART serial interface toward the PC and a high-speed data interface toward the RF DAC.

The clock is the life pulse of the FPGA. In order to ensure that multiple clocks are properly distributed across FPGA banks, we chose Xilinx's clock-management core, which provides an easy, interactive way of defining and specifying clocks.

A compact instruction core built around a state machine serves as the system control unit. As shown in Figure 2, at the initial state (S0), a header detector unit is active, monitoring and filtering the incoming data byte from the UART receiver. The data bytes were generated and encapsulated into frames in MATLAB, as shown in Figure 3.

Basically, there are two types of data frames in the system. The frame with header "FF01" (cRAM frame) was used for transmitting phase-incremental values for DDSes and system control messages. The other frame, with the header "FF10" or "FF11" (dRAM frame), is used for delivering the user-customized data. States "S1x" process only the data with the header "FF01" for updating the phase-incremental values and executing the control instructions. States "S2x" and "S3x" are utilized for receiving and storing the user-customized data for two bands, respectively. The busy signal is used for continuously latching the data in until seeing the last stop bit at the end of the data sequences. The control messages, for example invoking single/multiple DDS or the user data sequences, are stored in the last two bytes of the cRAM frame. They will be executed at the rising edge of the `cRAM_rd_done` signal.

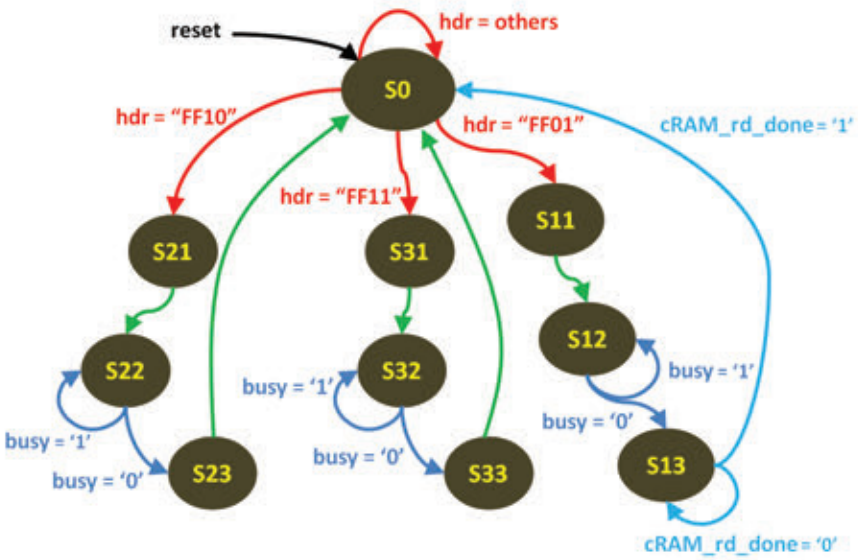



Figure 2 – Detailed design chart of the key state machine

We then instantiated four independent tone-generation units using Xilinx DDS cores, which we configured as the phase-incremental mode. The phase-incremental values for specific frequencies were generated in MATLAB and downloaded to the FPGA via the cRAM frame. We combined multiple tones via adders and then pipelined these tones to the next stage. Since the output of the DDS core was in two's complement binary format, a format-conversion unit may be needed if the RF DAC requires another data format, such as offset binary.

In general, high-performance on-chip BRAMs are always the first choice for creating small to medium-size user storage. For example, in this platform, we utilized Xilinx's Block Memory Generator core to build up two independent data RAMs for two frequency bands. Each of them is 16 bits in width and 192k in depth.

For communicating between the PC and the FPGA, we created a UART serial interface unit and configured it at a relatively low speed, 921.6 kbps, equivalent to 115.2 kbytes per second. It takes around 0.16 milliseconds and 3.33 seconds to transfer cRAM frames (18 bytes) and dRAM frames (~384k bytes), respectively.

Device vendors usually provide an example design of the high-speed data interface toward their chip in VHDL or Verilog format. It is not very difficult for an experienced FPGA engineer to reuse or customize the reference design. For example, considering our system's AD9739a and AD9129 RF DACs, a reference design of the parallel LVDS interface is available from Analog Devices. By the way, if there is no available example design from chip vendors, Xilinx has several very handy cores for the high-speed interfaces, such as CPRI and JESD204B.

## SW DESIGN: MATLAB DSP FUNCTIONS AND GUI

We chose MATLAB as the software host, simply because it has many advantages in terms of digital signal processing (DSP) capability. What's more, MATLAB also provides a handy tool called GUIDE for laying out a graphical user interface (GUI). So now, what do we need from MATLAB for this project?

We actually need a user interface associated with lower-level DSP functions and data flow control functions. The required DSP functions are a phase-incremental values calculator, baseband data

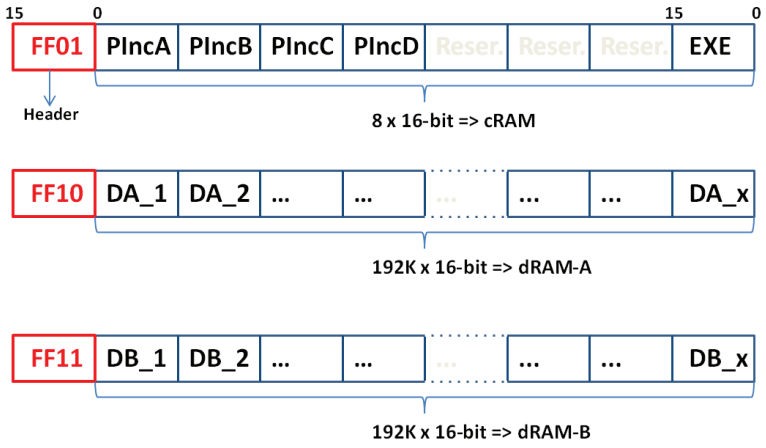



Figure 3 – Illustration of the data frame encapsulation

sequence generator and digital upconverter. The control functions are a data frame encapsulator, UART interface controller and system status indicators.

Figure 4 illustrates the GUI that we created for the platform. The key parameter of the RF DAC—namely, the sampling rate—should be defined first, and then you can select either xDDS mode or xRAM mode for stimulating the device. Next, at each subpanel, we can customize the parameters on the go to invoke the corresponding MATLAB signal-processing functions. In xDDS mode, you can calculate the phase-incremental value for the frequency tone `fc` with sampling rate `fs` by means of a simple equation, `phase_incr = fc*2^nbits/fs`, where `nbits` represents the number of binary bits used by DDS for synthesizing the frequencies. By pressing the "start" button, the generated phase-incremental values will be converted to the fixed-point format, encapsulated into a 2-byte data frame with different headers and control messages as shown in Figure 3, and then sent to the cRAM unit via UART and executed in the FPGA.

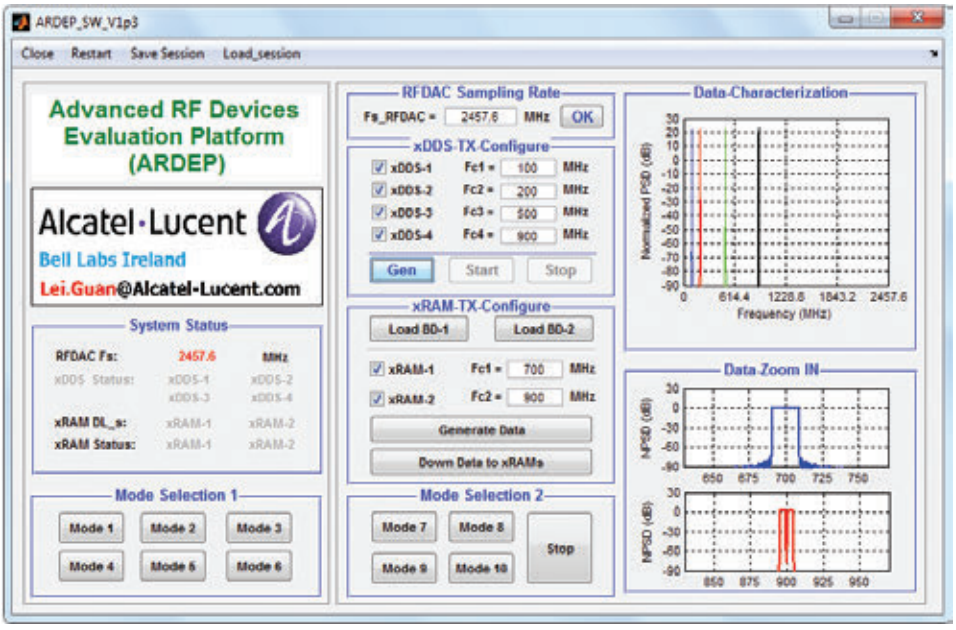



Figure 4 – Snapshot of the graphical user interface

In xRAM mode, we generated the baseband data sequences, normalized them to full scale (signed 16-bit) and upconverted them to the required frequency in MATLAB. After downloading the processed data to the dRAMs via UART, we can invoke the wideband signal testing by pressing the start button. Don't forget to configure the UART serial interface in MATLAB with the same protocol parameters used on the FPGA side.

Finally, we used a signal generator, the R&S SMU200A, to provide the sampling clock to logically "turn on" the RF DAC. And we connected the output of the RF DAC to a spectrum analyzer for evaluating the linearity performance of the RF DAC in the frequency domain.

## QUICK EVALUATION

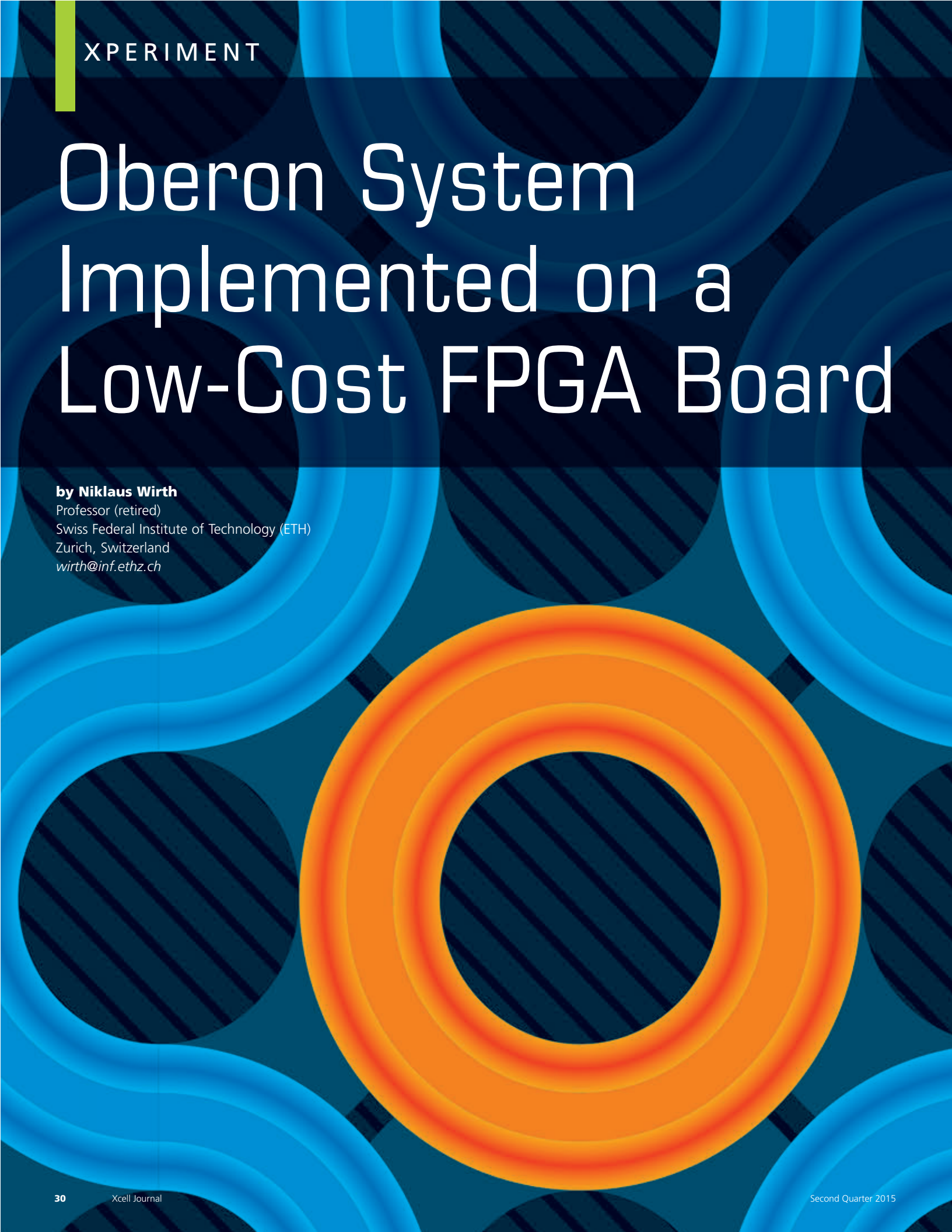
At the early-prototyping stage, evaluation of the key RF components regarding linearity performance can be a critical problem, but with our hardware/software platform you can manage this evaluation quickly without compromising performance. Then you can add in your RF power amplifier and use the proposed platform to evaluate the linearity of the cascaded system. After identifying the nonlinearities, you can implement some digital-predistortion algorithms to eliminate the unwanted nonlinearities of the cascaded system.

Properly using Xilinx IP cores in your FPGA design will significantly reduce the development cycle and increase the robustness of your digital system. Looking forward, we expect to upgrade the data interface module in the platform to the JESD204B standard to support the higher data rates required by multiple synchronized RF DACs. Meanwhile, we are migrating the FPGA host from Xilinx's ML605 to the Zynq®-7000 All Programmable SoC ZC706 evaluation kit. The Zynq SoC design will be a good choice for creating a standalone solution that does not need any external DSP and control functions on a separate PC.

For more information regarding the platform and digital predistortion, contact the author at *lei.guan@alcatel-lucent.com*.

# Oberon System Implemented on a Low-Cost FPGA Board

**by Niklaus Wirth**
Professor (retired)
Swiss Federal Institute of Technology (ETH)
Zurich, Switzerland
*wirth@inf.ethz.ch*



A Xilinx Spartan-3 board becomes the basis for revamping the author's Oberon programming language and compiler for use in software education.

In 1988, Jürg Gutknecht and I completed and published the programming language Oberon [1,2] as the successor to two other languages, Pascal and Modula-2, which I had developed earlier in my career. We originally designed the Oberon language to be more streamlined and efficient than Modula-2 so that it could better help academics teach system programming to computer science students. To advance this endeavor, in 1990 we developed the Oberon operating system (OS) as a modern implementation for workstations that use windows and have word-processing abilities. We then published a book that details both the Oberon compiler and and the operating system of the same name. The book, entitled *Project Oberon*, includes thorough instructions and source code.

A few years ago, my friend Paul Reed suggested that I revise and reprint the book because of its value for teaching system design, and because it serves as a good starting point to help would-be innovators build dependable systems from scratch.

There was a big obstacle, however. The compiler I originally developed targeted a processor that has essentially disappeared. Thus, my solution was to rewrite a compiler for a modern processor. But after doing quite a bit of research, I couldn't find a processor that satisfied my criteria for clarity, regularity and simplicity. So I designed my own. I was able to bring this idea to fruition because of the modern FPGA, as it

Xcell Journal *is honored to publish this article by industry legend Niklaus Wirth, who invented Pascal and several derivative programming languages and has pioneered some classic approaches to computer and software engineering. A recipient of the ACM Turing Award and of the IEEE Computer Pioneer Award, Professor Wirth has retired from teaching but continues to help educators develop and inspire the innovators of tomorrow.*

Choosing a Xilinx FPGA allowed me to update the system while keeping the design as close as possible to the original version from 1990.

allowed me to design the hardware as well as the system software. What's more, choosing a Xilinx® FPGA allowed me to update the system while keeping the design as close as possible to the original version from 1990.

The new processor is called RISC, and it was implemented on the low-cost Digilent Spartan®-3 development board, hosting a 1-Mbyte static RAM (SRAM) memory. The only system hardware additions I made were to add an interface for a mouse and an SD card to replace the hard-disk drive in the older system.

The book and the source code for the entire system are available on *projectoberon.com* [3,4,5]. Also available at the same site is a single file called S3RISCinstall.zip. This file contains instructions, an SD-card filesystem image and FPGA configuration bit files (in the form of PROM files for the Spartan-3 board's Platform Flash), together with construction details for the SD-card/mouse interface hardware.

## THE RISC PROCESSOR

The processor consists of an arithmetic-logic unit; an array of 16 registers of 32 bits; and a control unit with instruction register, IR and program counter PC. The processor is represented by the Verilog module RISC5.

The processor features 20 instructions: four for moving, shifting and rotating; four for logic operations; four for integer arithmetic; four for floating-point arithmetic; two for memory access; and two for branching.

RISC5 is imported by RISC5Top, the environment, which contains the interfaces to various (memory-mapped) devices and the SRAM (256M x 32 bit). The entire system (Figure 1) consists of the follow-

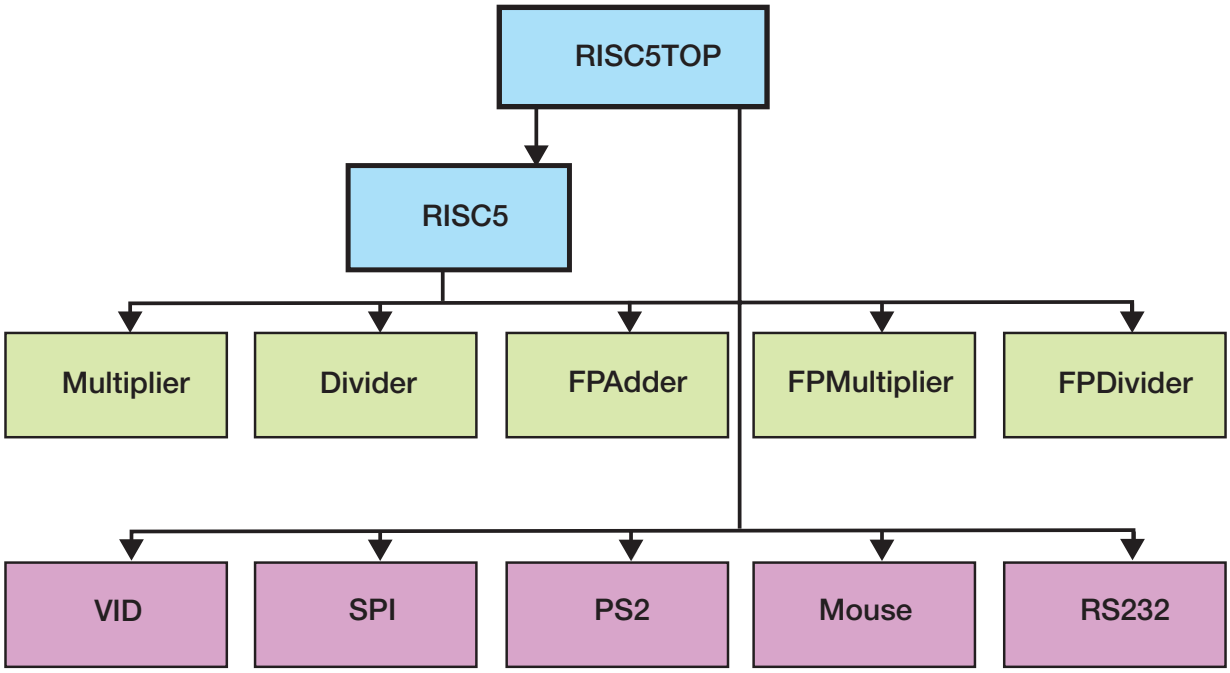



Figure 1 – Diagram of the system and the Verilog modules it contains

| | | |
|---|---|---|
| RISC5Top | environment | 194 |
| RISC5 | processor | 201 |
| Multiplier | integer arithmetic | 47 |
| Divider | | 24 |
| FPAdder | floating-point arithmetic | 98 |
| FPMultiplier | | 33 |
| FPDivider | | 35 |
| SPI | SD card and transmitter/receiver | 25 |
| VID | 1024 x 768 video controller | 73 |
| PS2 | keyboard | 25 |
| Mouse | mouse | 95 |
| RS232T | RS232 transmitter | 23 |
| RS232R | RS232 receiver | 25 |

ing Verilog modules (line counts shown):

I memory-mapped the black-and-white VGA display so that it occupies 1024 x 768 x 1 bit per pixel = 98,304 bytes, which is essentially 10 percent of the available main memory of 1 Mbyte. The SD card replaces the 80 Mbytes in the original system. It is accessed over a standard SPI interface, which accepts and serializes bytes or 32-bit words. The keyboard and the mouse are connected by standard, serial PS-2 interfaces. Furthermore, there is a serial, asynchronous RS-232 line and a general-purpose, 8-bit parallel I/O interface. Module RISC5Top also provides a counter, incremented every millisecond.

## THE OBERON OPERATING SYSTEM

The operating system software consists of a core that includes a memory allocator with a garbage collector and a file system, along with the loader, a text system, a viewer system and a text editor.

The module called "Oberon" is the central task dispatcher while "System" is the basic command module. An action is evoked by clicking the middle button on the text "M.P" in any viewer on the display, where P is the name of a procedure declared in module M. If M is not present, it is automatically loaded. Most text-editing commands, however, are evoked by simple mouse clicks, where the left button serves to set a caret, marking a text position, and the right button serves to select a text stretch.

The module "Kernel" contains the disk-store management and the garbage collector. I ensured that the viewers were tiled and do not overlap. The standard layout shows two vertical tracks with any number of viewers. You can enlarge them, make them smaller or move them by simply dragging their title bars. Figure 2 shows the user interface running on the monitor, along with the Spartan-3 board, keyboard and mouse.

The system when loaded occupies 112,640 bytes in module space (21 percent) and 16,128 bytes of the heap (3 percent). It consists of the following modules (line counts shown), as depicted in Figure 3:

| | | |
|---|---|---|
| Kernel | 271 | (inner core) |
| FileDir | 352 | |
| Files | 505 | |
| Modules (loader) | 226 | |
| Viewers | 216 | (outer core) |
| Texts | 532 | |
| Oberon | 411 | |
| MenuViewers | 208 | |
| TextFrames | 874 | |
| System | 420 | |
| Edit | 233 | |

It is remarkable that system initialization at power-on or reset takes only about 2 seconds. This includes a garbage-collecting scan of the file directory.

## THE OBERON COMPILER

The compiler, hosted on the system itself, uses the simple top-down recursive-descent parsing method. You can activate the compiler on a module's selected source text by using the command `ORP.Compile @`. The parser inputs symbols from the scanner delivering identifiers, numbers and special symbols (like BEGIN, END, + and so on). This scheme has proven to be useful and elegant in many applications. It is described in detail in my book *Compiler Construction* [6,7].

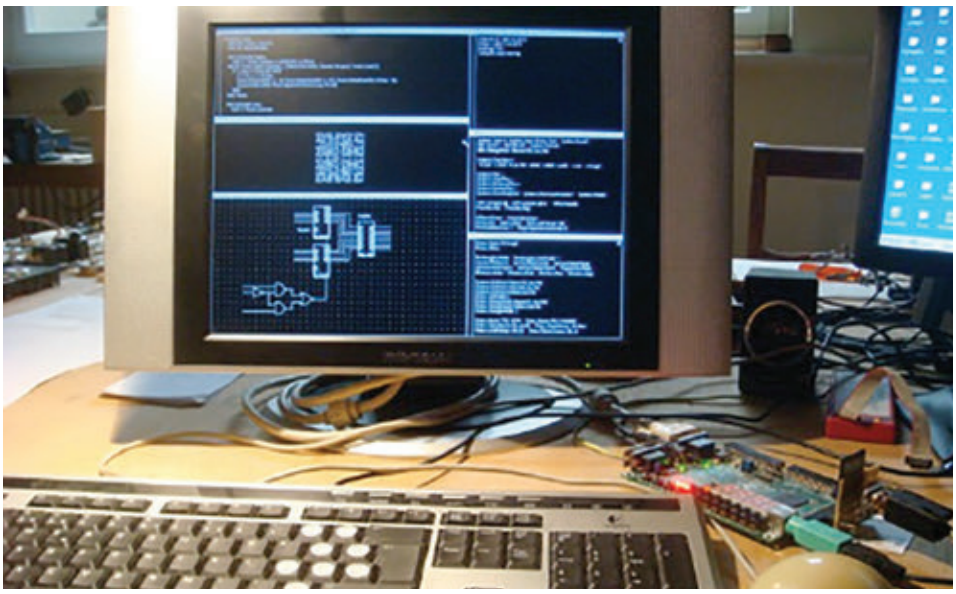


Figure 2 – Monitor showing user interface, with Spartan-3 board at right

The parser calls procedures in the code generator module. They directly append instructions in a code array. Forward-branch instructions are provided with jump addresses at the end of a module's compilation (fixups), when all branch destinations are known.

All variable addresses are relative to a base register. This is R14 (stack pointer) for local variables (set at procedure entry at run-time), or R13 for global and imported variables. The base addresses are loaded on demand from a system-global module table, whose address is held in register R12. R15 is used for return addresses (link), as determined by the RISC architecture. Thus, R0 - R11 are available for expression evaluation and for passing procedure parameters.

The entire compiler consists of four relatively small and efficient modules (line counts shown):

| | | |
|---|---|---|
| ORP | parser | 968 |
| ORG | code generator | 1120 |
| ORB | base def | 435 |
| ORS | scanner | 311 |

The compiler occupies 115,912 bytes (22 percent) of the module space and 17,508 bytes (4 percent) of the heap (before any compilation). Its source code is about 65 kbytes long. Compilation of the compiler itself

# The Lola HDL and its Translation to Verilog

The hardware-description language (HDL) called Lola was defined in 1990 as a means for teaching the basics of hardware design. This was the period when textual definitions started to replace circuit diagrams, and when the first FPGAs became available, although had not yet reached mainstream design. Lola was implemented by a compiler that generated bit files to be loaded onto FPGAs. Bit-file formats were disclosed by Algotronix Inc. and Concurrent Logic Inc. Both featured cells of rather simple structures, which seemed to be optimal for automatic placement and routing.

In the wake of my project to reimplement Oberon on an FPGA, the idea now popped up also to revive Lola. Because the cells of Xilinx's FPGAs feature a rather complex structure, we did not venture into the effort of implementing placement and routing, quite apart from the fact that Xilinx refuses to disclose its bit-file format for proprietary reasons.

The obvious path was to build a Lola compiler that does not generate proprietary bit files, but translates into a language for which Xilinx provides a synthesis tool. We chose Verilog. This solution implies a rather extravagant detour: First, a Lola module has to be parsed, then translated, then parsed again. With all these steps, we ensured that the Lola compiler had proper error reporting and type-consistency checking capabilities.

To drive the development of Lola-2, we had the ambition to reformulate in Lola all modules of the RISC5 processor. This has now been achieved.

## THE LOLA LANGUAGE

Lola is a small, terse language in the style of Oberon (see *http://www.inf.ethz.ch/personal/wirth/Lola/Lola2.pdf*).

For the sake of brevity, we show here only a single example of a Lola text (Figure 1). The unit of source text is called a module. Its heading specifies the name and its input and output parameters with their names and types. The heading is followed by

```
MODULE Counter0 (IN CLK50M, rstIn: BIT;
  IN swi: BYTE; OUT leds: BYTE);

TYPE IBUFG := MODULE (IN I: BIT; OUT O: BIT) ^;
VAR clk, tick0, tick1: BIT;
  clkInBuf: IBUFG;
REG (clk) rst: BIT;
  cnt0: [16] BIT;  (*half milliseconds*)
  cnt1: [10] BIT;  (*half seconds*)
  cnt2: BYTE;

BEGIN leds := swi.7 -> swi : swi.0 -> cnt1[9:2] : cnt2;
  tick0 := (cnt0 = 49999);
  tick1 := tick0 & (cnt1 = 499);
  rst := ~rstIn;
  cnt0 := ~rst -> 0 : tick0 -> 0 : cnt0 + 1;
  cnt1 := ~rst -> 0 : tick1 -> 0 : cnt1 + tick0;
  cnt2 := ~rst -> 0 : cnt2 + tick1;
  clkInBuf (CLK50M, clk)
END Counter0.
```

Figure 1 — Lola text showing a counter counting milliseconds and seconds, displaying the latter on the board's LEDs

takes only a few seconds on the 25-MHz RISC processor [8].

The compiler always generates checks for array indices and for references through pointers with value NIL. It causes a trap in case of violation. This technique ensures a high degree of security against errors and corruption. In fact, system integrity can be violated only by the use of operations in the pseudo-module SYSTEM, namely PUT and COPY. These operations must be restricted to driver modules accessing device interfaces, and they are easily recognized by SYSTEM in their import lists. The entire system is programmed in Oberon itself, and no use of assembler code is required.

I chose the Digilent Spartan-3 development board because of its low cost and simplicity, which makes it suitable for educational institutions to acquire sets of the kits for classrooms. An important advantage is the presence of static RAM on this board, which makes interfacing straightforward (even for byte selection). Unfortunately, newer boards all feature dynamic RAM, which is, albeit larger, very much more complicated to interface, requiring circuits for refresh and initialization (calibration). This circuitry alone can be as complex as the entire processor with static RAM. Even if a controller is provided on-chip,

a section containing declarations of local objects, such as variables and registers. Then follows the section defining the values of variables and registers through assignments. BYTE denotes an array of 8 bits.

## THE LOLA COMPILER

The compiler uses the simple top-down recursive-descent parsing method. It is activated on the selected Lola source text by the command `LSC.Compile @`. The parser inputs symbols from the scanner delivering identifiers, numbers and special symbols (like BEGIN, END, +, etc.). This scheme has proved to be useful and elegant in many applications. It is described in the book *Compiler Construction (Part 1* and *Part 2)*.

Instead of generating Verilog texts directly on the fly, statements are present in the parser which, as a side effect of parsing, generate a tree representing the input text in a way appropriate for further processing. This structure has the advantage that various, different outputs may easily be generated by calling on different translators. One of them is a translator to Verilog. The command is `LSV.List outputfile.v`. Another command may translate to VHDL, or simply list the tree. Yet another might generate a netlist for further processing by a placer and router.

Thus, the entire compiler consists of at least four relatively small and efficient modules (line counts shown):

| | | |
|---|---|---|
| LSS | scanner | 159 |
| LSB | base | 52 |
| LSC | compiler/parser | 503 |
| LSV | Verilog generator | 215 |

Instructions on translating from Lola to Verilog can be found at *http://www.inf.ethz.ch/personal/wirth/Lola/Lola-Compiler.pdf*.

## THE DIFFERENCE BETWEEN SOFTWARE AND HARDWARE 'PROGRAMS'

Many efforts have been undertaken in the past to make HDLs look like "ordinary" programming languages (PLs). We also note that HDLs typically have a counterpart in the set of PLs, adopting the respective style. Thus, Verilog has its ancestor in C, VHDL in Ada and Lola in Oberon. We consider it important to recognize the fundamental differences between the two classes, in particular in the presence of syntactic similarities or even identities. What are these fundamental differences?

In order to simplify an explanation, we restrict our analysis to synchronous circuits—that is, circuits in which all registers tick with the same clock. It is indeed a healthy design paradigm to stick to synchronous circuits in general, if possible. Then, quite obviously, all elements of a circuit operate concurrently, literally at the same time. Every variable and register is defined by one and only one expression (combinational circuit). Multiple assignments do not make sense. We can simply imagine every HDL program to be enclosed in a big repeat-forever clause, because the assignments to registers and variables are repeated in every clock cycle.

It was the ingenious idea of John von Neumann to introduce a processor architecture with a sequencer. The sequencer contains an instruction register, according to which (in every cycle) certain circuits are selected and others ignored, thus cleverly reusing the different parts (of the ALU). Now, cycles or steps have become inherently sequential, and it is possible to reassign values to the same variables, as the program counter relates them to certain places in the program and in the instruction sequence. It is this idea of the sequencer that makes it possible to execute enormous programs by relatively simple circuits.

In summary, Lola-2 is an HDL in the style of the PL Oberon. The compiler presented here translates Lola modules into Verilog modules. Lola's advantages lie in the language's simple and regular structure, and in the compiler's emphasis on type checking and improved error diagnostics. The entire set of modules for the RISC processor have been expressed in Lola (see *http://www.inf.ethz.ch/personal/wirth/Lola/index.html*).

*— Niklaus Wirth*

this counteracts our principle of laying everything open for inspection.

## CLOSING THOUGHTS

More than 40 years ago, C.A.R. Hoare remarked that in all branches of science and technology, students are exposed to many master examples of design before they are asked to experiment with their own attempts. Programming and software design stood in marked contrast to this sensible paradigm. Here the student was asked to write programs right from the start, before having read any examples.

The reason for this dire fact was that there existed almost no literature with sizable master examples. I therefore decided to remedy the situation somewhat, and I wrote the book on *Algorithms and Data Structures* in 1975. Subsequently (with J. Gutknecht), having been charged with the task of teaching a course on operating systems, I designed the system Oberon (1986-88).

Since then, the teaching of programming has not markedly improved, whereas systems have dramatically increased in size and complexity. Although the open-source endeavor is to be welcomed, it has not really changed the situation, since most programs have been constructed "to run," but not really for human consumption, for being understood.

I continue to make the daring proposal that all programs should be designed not just for computers, but for human reading. They should be publishable. This is a task much harder than creating executable programs, even correct and efficient ones. It implies that no part must be specified in assembler code.

The result of ignoring this "human factor" is that in many places new applications are not carefully designed, but rather debugged into existence, sometimes with dismal consequences. An important ingredient to achieve understandability is to adhere to simplicity and regularity, and to renounce unnecessary embellishments, to avoid bells and whistles and to distinguish between conventional and convenient.

The small size of this system is witness to *how much can be achieved with how little*. The Oberon system's dimensions are ridiculously small compared with most modern operating systems, although it includes a file system, a text editor and a viewer (windows) management. The side ef-

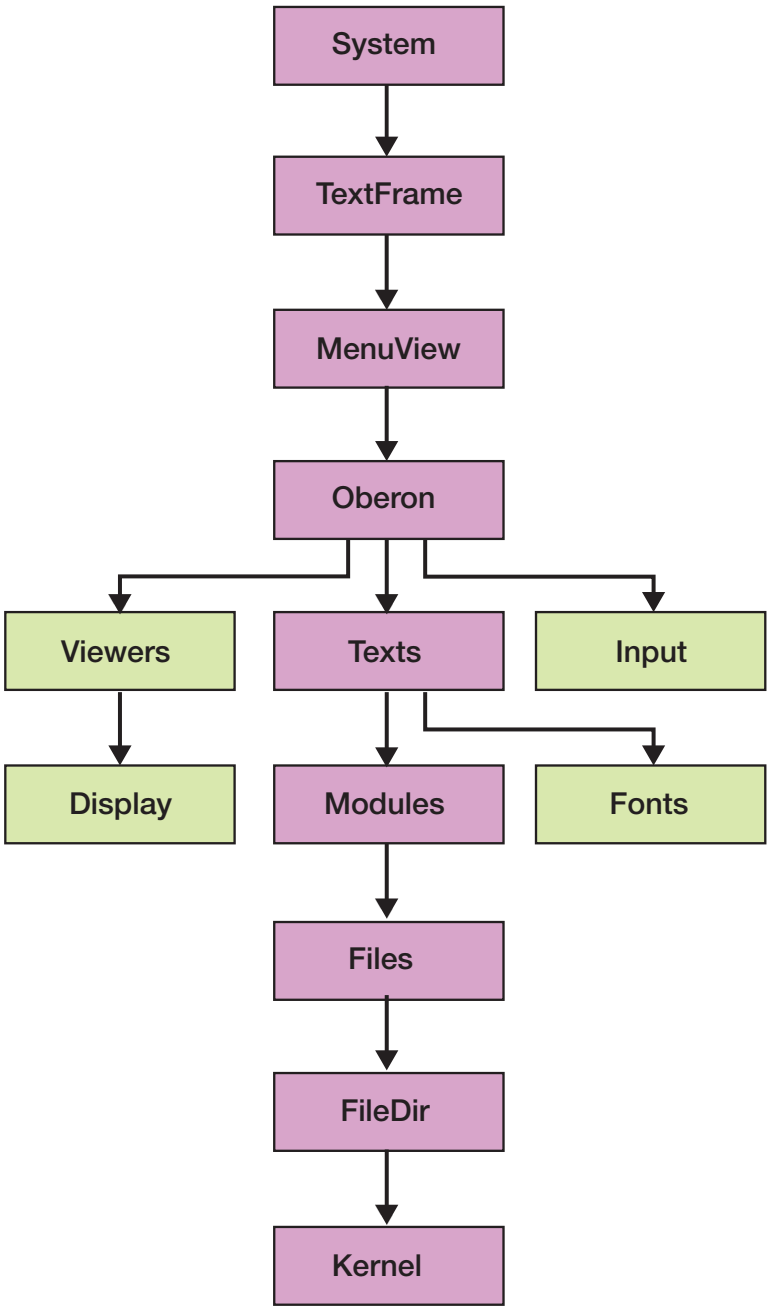



Figure 3 – The system and its modules

fect is that it rests on just a few simple rules, and that therefore it is easy to learn how to use it.

Finally, another advantage of this terseness is that one can safely build upon this basic system without being afraid of the existence of unknown features, such as back doors. This is an essential property in view of the increasing danger of attacks on the integrity of a system, indispensable for safety-critical applications. Notably, also the hardware of our system is free of such hidden parts. After all, no guarantee can be given for any system that is built upon a base that is so large that nobody can understand it in its entirety.


## ACKNOWLEDGEMENT

*I gratefully acknowledge the invaluable contributions of Paul Reed. He suggested that I re-edit the book* Project Oberon *and also suggested reimplementing the entire system on an FPGA. Paul was an essential source of encouragement. He thought of replacing the disk with an SD card, and he contributed the SPI, the PS-2 and the VID interfaces in Verilog.*


## References


1. *http://www.inf.ethz.ch/personal/wirth/Oberon/Oberon07.Report.pdf*
2. *http://www.inf.ethz.ch/personal/wirth/Oberon/PIO.pdf*
3. *www.inf.ethz.ch/personal/wirth/ProjectOberon/index.html*
4. *www.inf.ethz.ch/personal/wirth/Oberon/PIO.pdf*
5. *www.inf.ethz.ch/personal/wirth/Oberon/Oberon07.Report.pdf*
6. *http://www.inf.ethz.ch/personal/wirth/CompilerConstruction/CompilerConstruction1.pdf*
7. *http://www.inf.ethz.ch/personal/wirth/CompilerConstruction/CompilerConstruction2.pdf*
8. *http://www.inf.ethz.ch/personal/wirth/ProjectOberon/PO.Applications.pdf* (Ch. 12)

# Removing the Barrier for FPGA-Based OpenCL Data Center Servers

Xilinx's SDAccel environment delivers a CPU-like development and run-time experience on FPGAs, easing the data center design burden.

**by Devadas Varma**
Senior Engineering Director
SDAccel and Vivado High-Level Synthesis
Xilinx, Inc.
*dvarma@xilinx.com*

**Tom Feist**
Senior Director
Design Methodology Marketing
Xilinx, Inc.
*tfeist@xilinx.com*

Data centers today are the backbone of the modern economy, from the server rooms that power small to midsize organizations to the enterprise data centers that support U.S. corporations and provide access to cloud computing services. According to the Natural Resources Defense Council, data centers are one of the largest and fastest-growing consumers of electricity in the United States. In 2013, U.S. data centers consumed an estimated 91 billion kilowatt-hours of electricity—enough to power all the households in New York City twice over—and are on track to reach 140 billion kWh by 2020 [1]. Clearly, lowering power is essential for the scaling of data centers to improve reliability and lower operating costs.

Data center servers vary, depending on the server application. Many servers run for long periods without interruption, making hardware reliability and durability extremely important. Although servers can be built from commodity computer parts, mission-critical enterprise servers often use specialized hardware for application acceleration, including graphics processing units (GPUs) and digital signal processors (DSPs). Now, many companies are looking to add field-programmable gate arrays (FPGAs) for their highly parallel architecture and relatively low power consumption. Xilinx®'s new SDAccel™ development environment removes programming as a gating issue to FPGA utilization in this application by providing developers with a familiar CPU/GPU-like environment.

## IMPROVING PERFORMANCE/WATT

Public clouds such as Amazon Web services, Google Compute, Microsoft Azure, Facebook and China's Baidu have huge repositories of pictures and require very fast image recognition. In one implementation, Google scientists created one of the largest neural networks for machine learning by connecting 16,000 computer processors into an entity that they turned loose on the Internet to learn on its own. The research is representative of a new generation of computer science that is exploiting the availability of huge clusters of computers in giant data centers. Potential applications include improvements to image search, speech recognition and machine language translation. However, leveraging CPUs alone is not a power-efficient approach to data center design. For higher speed and lower power, alternative solutions are required.

Baidu, China's largest search-engine specialist, turned to deep-neural-network processing to solve problems in speech recognition, image search and natural-language processing. The company quickly determined that when neural back-propagation algorithms are used in online prediction, FPGA solutions scale far more easily than CPUs and GPUs while also reducing power [2].

The new generation of 28nm and 20nm high-integration FPGA families, such as Xilinx's 7 series and UltraScale™ devices, are changing the dynamics for integration of FPGAs into host cards and line cards in data center servers. Performance per watt can easily exceed 20x that of an equivalent CPU or GPU, while offering up to 50x to 75x latency improvements in some applications over traditional CPUs.

Teams with limited or no FPGA hardware resources, however, have found the transition to FPGAs challenging due to the RTL (VHDL or Verilog) development expertise needed to take full advantage of these devices. To resolve this issue, Xilinx has looked to the Open Computing Language, or OpenCL™, for a way to ease the programming burden.

## OPENCL CODE PORTABILITY

OpenCL was developed by Apple Inc. and promoted by Khronos Group [3] precisely to aid the integration of CPUs, GPUs, FPGAs and DSP blocks in heterogeneous designs. To enhance the OpenCL framework for writing programs that execute across heterogeneous platforms, leading CPU, GPU and FPGA vendors, including Xilinx,

# The SDAccel compiler delivers a 10x performance improvement over CPUs at 1/10 the power consumption of a GPU.

are contributing to development of both the language and its APIs.

The growing acceptance of OpenCL by CPU, GPU and FPGA vendors, server OEMs and data center managers alike is an indication that all parties recognize one overarching fact: C-based compilers for single-processor architectures can only offer small reductions in overall power dissipation within the server rack, even as processors turn to sub-20nm process technologies and add special power-saving states.

OpenCL is a framework for writing programs that execute across heterogeneous platforms consisting of CPUs, GPUs, DSPs, FPGAs and other processors. OpenCL includes a language (based on C99) for programming and an application programming interface (API) to control the platform and execute programs on the target device. OpenCL provides parallel computing using task-based and data-based parallelism.

## XILINX SDACCEL DEVELOPMENT ENVIRONMENT FOR OPENCL

Xilinx has worked for nearly a decade on the development of domain-specific specification environments. Concerns about data center performance from both data center managers and server/switch OEMs helped drive one such vertical development toward a unified environment for design optimization in data center applications. The result is SDAccel™, an OpenCL development environment for application acceleration.

The new Xilinx SDAccel environment (Figure 1) provides data center application developers with the complete FPGA-based hardware and OpenCL software. SDAccel includes a fast, architecturally optimizing compiler that makes efficient use of on-chip FPGA resources along with a familiar software-development flow based on an Eclipse integrated design environment (IDE) for code development, profiling and debugging. This IDE provides a CPU/GPU-like work environment.

Moreover, SDAccel leverages Xilinx's dynamically reconfigurable technology to enable accelerator kernels optimized for different applications to be swapped in and out on the fly. The applications can have multiple kernels swapped in and out of the FPGA during run-time without disrupting the interface between the server CPU and the FPGA for nonstop application acceleration.

SDAccel's architecturally optimizing compiler allows software developers to optimize and compile streaming, low-latency and custom datapath applications. The SDAccel compiler supports source code using any combination of C, C++ and OpenCL and targets high-performance Xilinx FPGAs. The SDAccel compiler delivers as much as a 10x performance improvement over high-end CPUs and one-tenth the power consumption of a GPU, while maintaining code compatibility and a traditional software-pro-

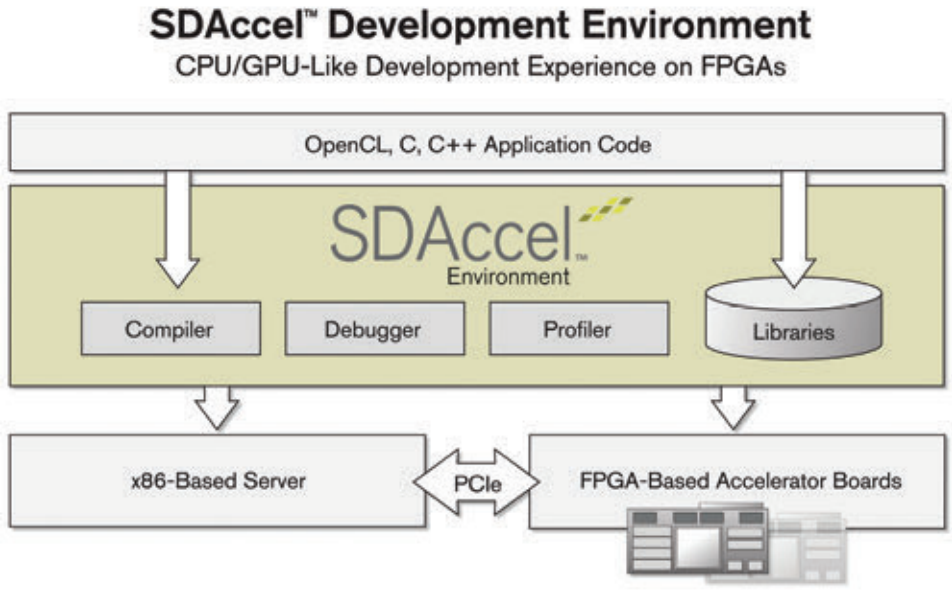



Figure 1 – The SDAccel environment includes an architecturally optimizing compiler, libraries, a debugger and a profiler to provide a CPU/GPU-like programming experience.

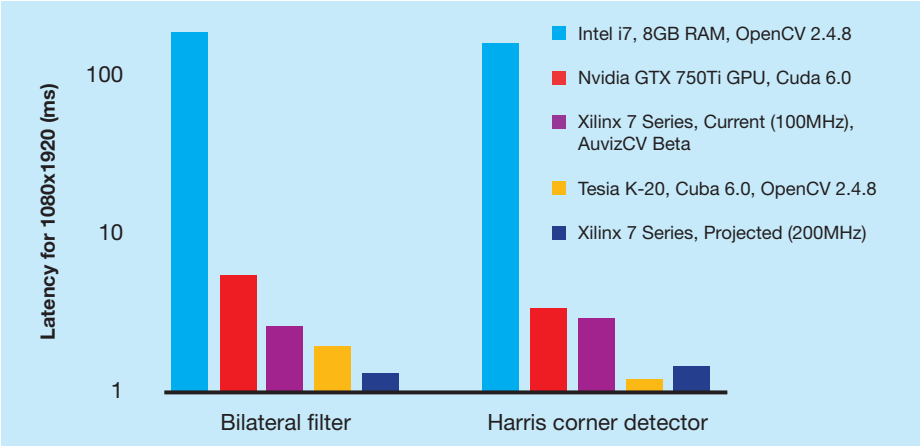



Figure 2 – Video-processing algorithms written in OpenCL for CPU, GPU and FPGA architectures run faster on FPGAs.

*(Benchmarks performed by Auviz using the AuvizCV library)*

gramming model for easy application migration and cost savings.

SDAccel is the only FPGA-based development environment that includes a wide variety of FPGA-optimized libraries for application acceleration. This library includes OpenCL built-ins, arbitrary-precision data types (fixed-point), floating-point, math.h, video, signal-processing and linear algebra functions.

On real-world computation workloads such as video processing with complex nested datapaths, it is clear that the inherent flexibility of the FPGA fabric has performance and power advantages when compared with the fixed architectures of CPUs and GPUs. As shown by the benchmarks seen in Figure 2, the FPGA solution compiled by SDAccel outperforms the CPU implementation of the same code and offers performance competitive with GPU implementations.

Both the bilateral filter and the Harris corner detector were coded using the standard OpenCL design paradigm in which data between kernels is transferred using device global memory. The FPGA implementation generated by SDAccel optimizes memory access by creating on-chip memory banks that are used for high-bandwidth memory transfers and low-latency computations. The creation and usage of these application-specific memory banks represent some of the architecturally aware capabilities of the SDAccel compiler.

## SOFTWARE WORKFLOW

FPGAs have long held the promise of increased algorithm performance at a lower power envelope than CPU and GPU implementations. Until now, that promise has been gated by the programming paradigm required to effectively use FPGAs. SDAccel eliminates this barrier by supporting a software workflow with in-system, on-the-fly reconfigurability that maximizes hardware acceleration ROI in the data center. SDAccel represents a unique and complete FPGA-based solution that far exceeds the capabilities and ease of use of competing point tools. For more information, visit *http://www.xilinx.com/products/design-tools/sdx/sdaccel.html.*

## REFERENCES

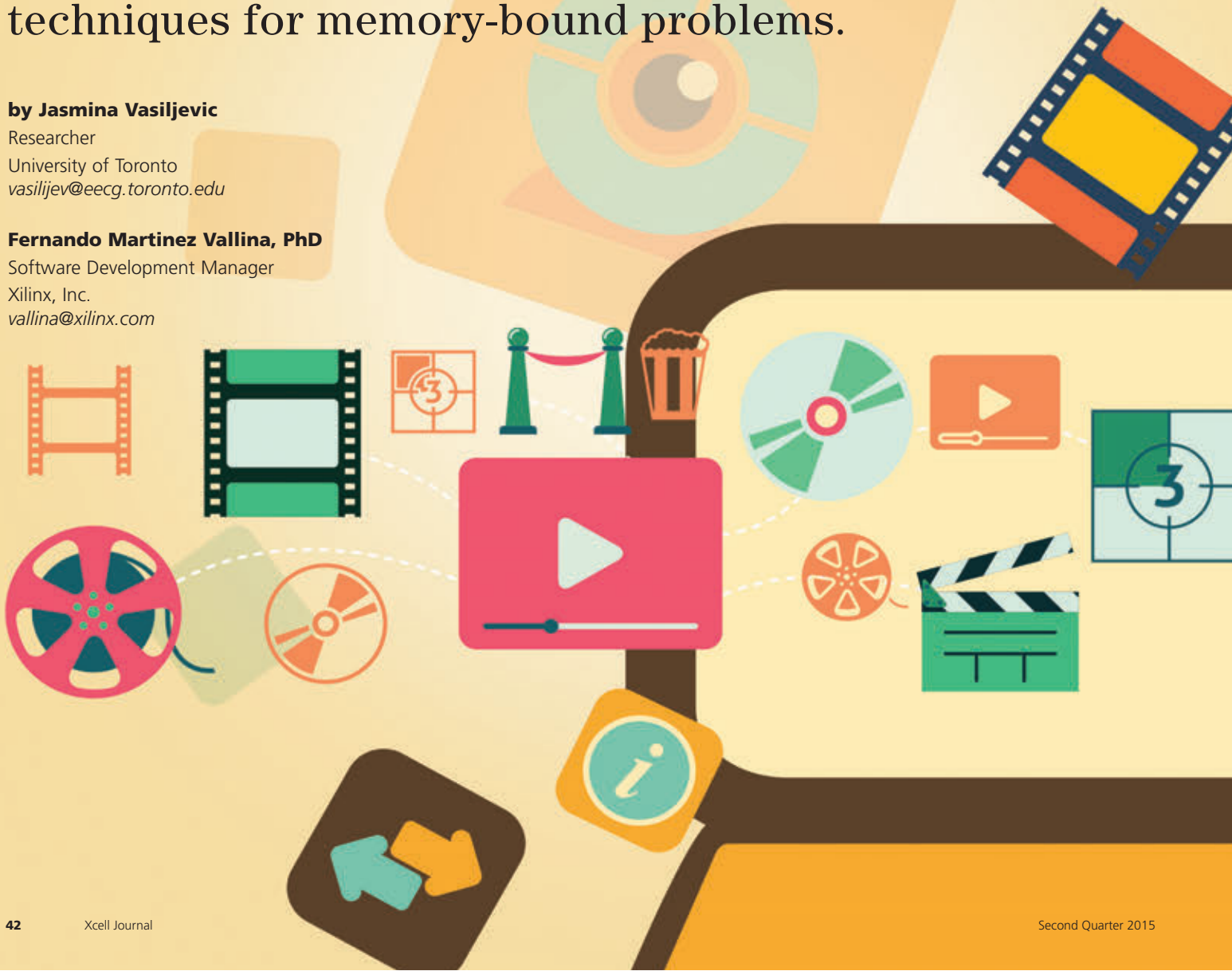
1. *http://www.nrdc.org/energy/data-center-efficiency-assessment.asp*
2. *http://www.pcworld.com/article/2464260/microsoft-baidu-find-speedier-search-results-through-specialized-chips.html*
3. *https://www.khronos.org/opencl*

# Optimizing an OpenCL Application for Video Watermarking in FPGAs

## Xilinx's SDAccel development environment offers optimization techniques for memory-bound problems.

**by Jasmina Vasiljevic**
Researcher
University of Toronto
*vasilijev@eecg.toronto.edu*

**Fernando Martinez Vallina, PhD**
Software Development Manager
Xilinx, Inc.
*vallina@xilinx.com*



Video streaming and downloading account for the majority of consumer Internet traffic and are a driving force behind cloud computing. The continually growing demand for this type of content is pushing video-processing applications out of specialized systems and into the data center. This shift in the deployment paradigm allows for the rapid scaling of computation nodes to accommodate the needs of the different compute-intensive stages of video content preparation and distribution, such as transcoding and watermarking.

We recently used Xilinx®'s SDAccel™ development environment to compile and optimize a video-watermarking application written in OpenCL™ for an FPGA accelerator card. Video content providers use watermarking to brand and protect their content. Our goal was to design a watermarking application that would process high-definition (HD) video at a 1080p resolution with a target throughput of 30 frames per second (fps) running on an Alpha Data ADM-PCIE-7V3 card.

The SDAccel development environment enables designers to take applications captured in OpenCL and compile them to an FPGA without requiring knowledge of the underlying FPGA implementation tools. The video-watermarking application serves as a perfect way to introduce the main optimization techniques available in SDAccel.

## VIDEO WATERMARKING WITH LOGO INSERTION

The main function of the video-watermarking algorithm is to overlay a logo at a specific location on a video stream. The logo used for the watermark can be either active or passive. An active logo is typically represented by a short, repeating video clip, while a passive logo is a still image.

The most common technique among broadcasting companies that brand their video streams is to use a company logo as a passive watermark, so that was the aim of our example design. The application inserts a passive logo on a pixel-by-pixel level of granularity based on the operations of the following equation:

```
out_y[x][y]  = (255-mask[x][y]) * in_y[x][y]  + mask[x][y] * logo_y[x][y]
out_cr[x][y] = (255-mask[x][y]) * in_cr[x][y] + mask[x][y] * logo_cr[x][y]
out_cb[x][y] = (255-mask[x][y]) * in_cb[x][y] + mask[x][y] * logo_cb[x][y]
```

The input and output frames are two-dimensional arrays in which pixels are expressed using the YCbCr color space. In this color space, each pixel is represented in three components: Y is the luma component, Cb is the chroma blue-difference component and Cr is the chroma red-difference component. Each component is represented by an 8-bit value, resulting in a total of 24 bits per pixel.

The logo is a two-dimensional image containing the content to be inserted. The mask is also an image, but it contains only the contour of the logo. The pixels in the mask are either white or black. White pixels in the mask indicate the logo insertion location, while black pixels indicate that the original pixel remains untouched. Figure 1 shows an example of the operation of the video-watermarking algorithm.

## TARGET SYSTEM AND INITIAL IMPLEMENTATION

The system on which we executed the application is shown in Figure 2. It is composed of an Alpha Data ADM-PCIE-7V3 card communicating with an x86 processor over a PCIe® link. In this system, the host processor retrieves the input video stream from disk and transfers it to the device global memory. The device global memory is the memory on the FPGA card that is directly accessible from the FPGA. In addition to placing the video frames in device global memory, the logo and mask are transferred from the host to the accelerator card and placed in on-chip memory to take advantage of the low latency of BRAM memories. Since this application uses a passive logo, only the data for the still image and placement location needs to be stored in the on-chip memories.

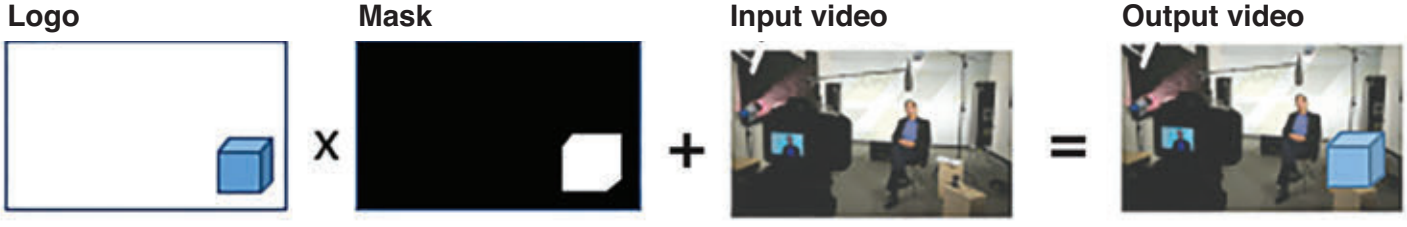



Figure 1 – The video-watermarking algorithm in action

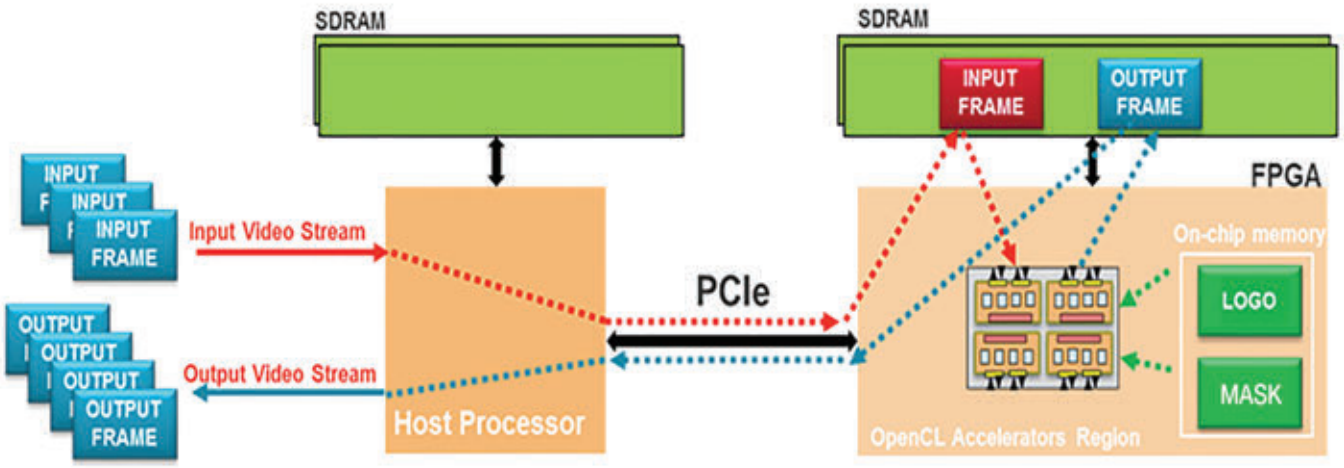



Figure 2 – System overview for the video-watermarking application

Once data has been set up, the host processor sends a start signal to the watermarking kernel in the FPGA fabric. This signal triggers the kernel to do three things: start fetching input video frames from device global memory, insert the logo at the location defined by the mask and place the processed frames back in device global memory to be fetched by the processor.

The coordination of data transfer and computation for every frame in the video stream is achieved by means of the code in Figure 3.

This code, which runs on the host processor, is responsible for sending a video frame to the FPGA accelerator card, launching the accelerator and then retrieving the processed frame from the FPGA accelerator card.

The first implementation of the watermarking algorithm for the FPGA is shown in Figure 4. This is a functionally correct implementation of the application but does not have any performance optimization or consideration for the capabilities of the FPGA fabric. As is, this code compiles in SDAccel and can be run on the Alpha Data card for a maximum throughput of 0.5 fps.

As you can see in the code of Figure 4, the watermarking algorithm is not a compute-intensive design. Most of the time is spent accessing memory to read and write video frames. Therefore, we focused on memory bandwidth when optimizing our example design.

## OPTIMIZING MEMORY ACCESS USING VECTORIZATION

One of the advantages of the FPGA fabric over other software-programmable fabrics is the flexibility and configuration of interconnect buses to memory. SDAccel creates custom-size datapaths and architectures to memory based on the application kernels. A higher memory bandwidth from the kernel can be inferred by modifying the code to consume multiple pixels at a time, a procedure referred to as vectorization.

The level of vectorization that is appropriate depends on the application and the FPGA accelerator card being used. In the case of the Alpha Data card, the interface to device global memory has a width of 512 bits, which matches the maximum AXI interconnect width available to a kernel in SDAccel. Given this boundary of 512 bits, the application is modified to process 20 pixels at a time (24 bits/pixel x 20 pixels = 504 bits). SDAccel offers full support for vector data types. Therefore, for this application, vectorizing the code is as simple as changing the data type of all arrays to char20, as shown in Figure 5, which results in a throughput of 12 fps.

## OPTIMIZING MEMORY ACCESS USING BURSTS

Although vectorization significantly improves the performance of the application, it was not enough to reach the goal of 30 fps. The application remains memory bound, because the kernel is issuing memory transfers of only 20 pixels at a time. To reduce the impact of memory constraints on the application, we had to modify the kernel code to generate burst read/write operations to memory for a data set larger than 20 pixels. The modified kernel code is shown in Figure 6.

The first modification to the kernel code is to define on-chip storage in the kernel to store a block of pixels at a time. The on-chip memories are defined by arrays declared in the kernel code. To start a burst transaction to memory, the code instantiates a memcpy command to move a block of data from DDR to BRAM storage inside of the kernel. Based on the size of on-chip memory resources and the amount of data to

be processed, a video frame is divided into 20 blocks of 1920 x 54 pixels, as shown in Figure 7.

Once the memcpy operations have placed the data for a block into the kernel arrays, the algorithm executes the watermarking algorithm on the block of data and places the results back into kernel arrays. The results of the block processing are transferred back to DDR memory using memcpy operations. This sequence of operations repeats for 20 times until all blocks in a given frame have been processed. As a result of this modification to the kernel code, the system performance is 38 fps, which exceeds the original goal of 30 fps.

**Host Code**

SEND FRAME / COMPUTE FRAME / RECEIVE FRAME

```
for (i=0; i<FRAMES; i++) {
  // Send a video frame to the FPGA device
  err = clEnqueueWriteBuffer(commands, d_frin, CL_TRUE, 0,
        sizeof(int) * LENGTH_FRAN, h_frin, 0, NULL, &writeEvent0;
  clWaitForEvents(1, &writeEvent);

  // Run logo insertion on the input frame
  err = clEnqueueTask(commands, kernel_load_block, 0, NULL,
        &kernelEvent);
  clWaitForEvents(1, &kernelEVent);

  // Read the output frame back to CPU
  err = clEnqueueReadBuffer( commands, d_frout, CL_TRUE, 0,
        sizeof(int) * LENGTH_FROUT, h_frout, 0, NULL, &readEvent);
  clWritForEvents(1, &readEvent);
}
```

Figure 3 – Code to coordinate each frame's data transfer and computation

```
for (y=0; y<FRAMES_HEIGHT; y++) {
  for (x=0; x<FRAMES_WIDTH/20; x++) {

        i = y*FRAME_WIDTH/20 + x;
        out_y[i]   = (255-mask[i]) * in_y[i]   + mask[i] * logo_y[i];
        out_cr[i]  = (255-mask[i]) * in_cr[i]  + mask[i] * logo_cr[i];
        out_cb[i]  = (255-mask[i]) * in_cb[i]  + mask[i] * logo_cb[i];
  }
}
```

Figure 4 – Initial implementation of the watermarking kernel

```
for (y=0; y<FRAMES_HEIGHT; y++) {
  for (x=0; x<FRAMES_WIDTH/20; x++) {

        i = y*FRAME_WIDTH/20 + x;

        out_y[i]   = (255-mask[i]) * in_y[i]   + mask[i] * logo_y[i];
        out_cr[i]  = (255-mask[i]) * in_cr[i]  + mask[i] * logo_cr[i];
        out_cb[i]  = (255-mask[i]) * in_cb[i]  + mask[i] * logo_cb[i];
  }
}
```

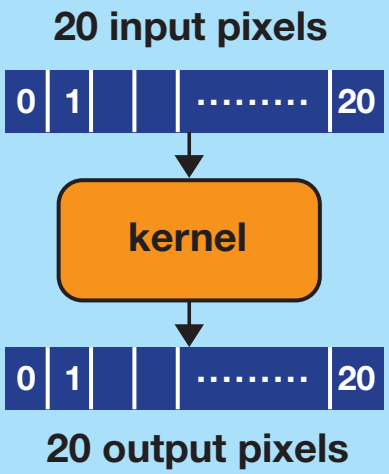



Figure 5 – Kernel code after vectorization

```
char20 l_in_y[BLOCK_WIDHT*BLOCK_HEIGHT/20];
char20 l_in_cr[BLOCK_WIDHT*BLOCK_HEIGHT/20];
char20 l_in_cb[BLOCK_WIDHT*BLOCK_HEIGHT/20];

char20 l_out_y[BLOCK_WIDHT*BLOCK_HEIGHT/20];
char20 l_out_cr[BLOCK_WIDHT*BLOCK_HEIGHT/20];
char20 l_out_cb[BLOCK_WIDHT*BLOCK_HEIGHT/20];

for (block_id=0; block_id<FRAME_HEIGHT/BLOCK_HEIGHT; block_id++){

  memcpy(l_in_y, in_y + block_id*BLOCK_WIDHT*BLOCK_HEIGHT/20, BLOCK_WIDHT*BLOCK_HEIGHT*3);
  memcpy(l_in_cr, in_cr + block_id*BLOCK_WIDHT*BLOCK_HEIGHT/20, BLOCK_WIDHT*BLOCK_HEIGHT*3);
  memcpy(l_in_cb, in_cb + block_id*BLOCK_WIDHT*BLOCK_HEIGHT/20, BLOCK_WIDHT*BLOCK_HEIGHT*3);

  for (y=0; y<BLOCK_HEIGHT; y++) {
    for (x=0; x<BLOCK_WIDTH/20; x++) {

      i = y*BLOCK_WIDTH/20 + x;

      l_out_y[i]  = (255-mask[i]) * l_in_y[i]  + mask[i] * logo_y[i];
      l_out_cr[i] = (255-mask[i]) * l_in_cr[i] + mask[i] * logo_cr[i];
      l_out_cb[i] = (255-mask[i]) * l_in_cb[i] + mask[i] * logo_cb[i];
  }

  memcpy(out_y, l_out_y + block_id*BLOCK_WIDHT*BLOCK_HEIGHT/20, BLOCK_WIDHT*BLOCK_HEIGHT*3);
  memcpy(out_cr, l_out_cr + block_id*BLOCK_WIDHT*BLOCK_HEIGHT/20, BLOCK_WIDHT*BLOCK_HEIGHT*3);
  memcpy(out_cb, l_out_cb + block_id*BLOCK_WIDHT*BLOCK_HEIGHT/20, BLOCK_WIDHT*BLOCK_HEIGHT*3);
}
```

On-chip memory storage for the input block

On-chip memory storage for the ioutput block

Loop through 20 blocks

PHASE 1

PHASE 2

PHASE 3

Figure 6 – Kernel code optimized for burst data transfers

## BROADLY APPLICABLE

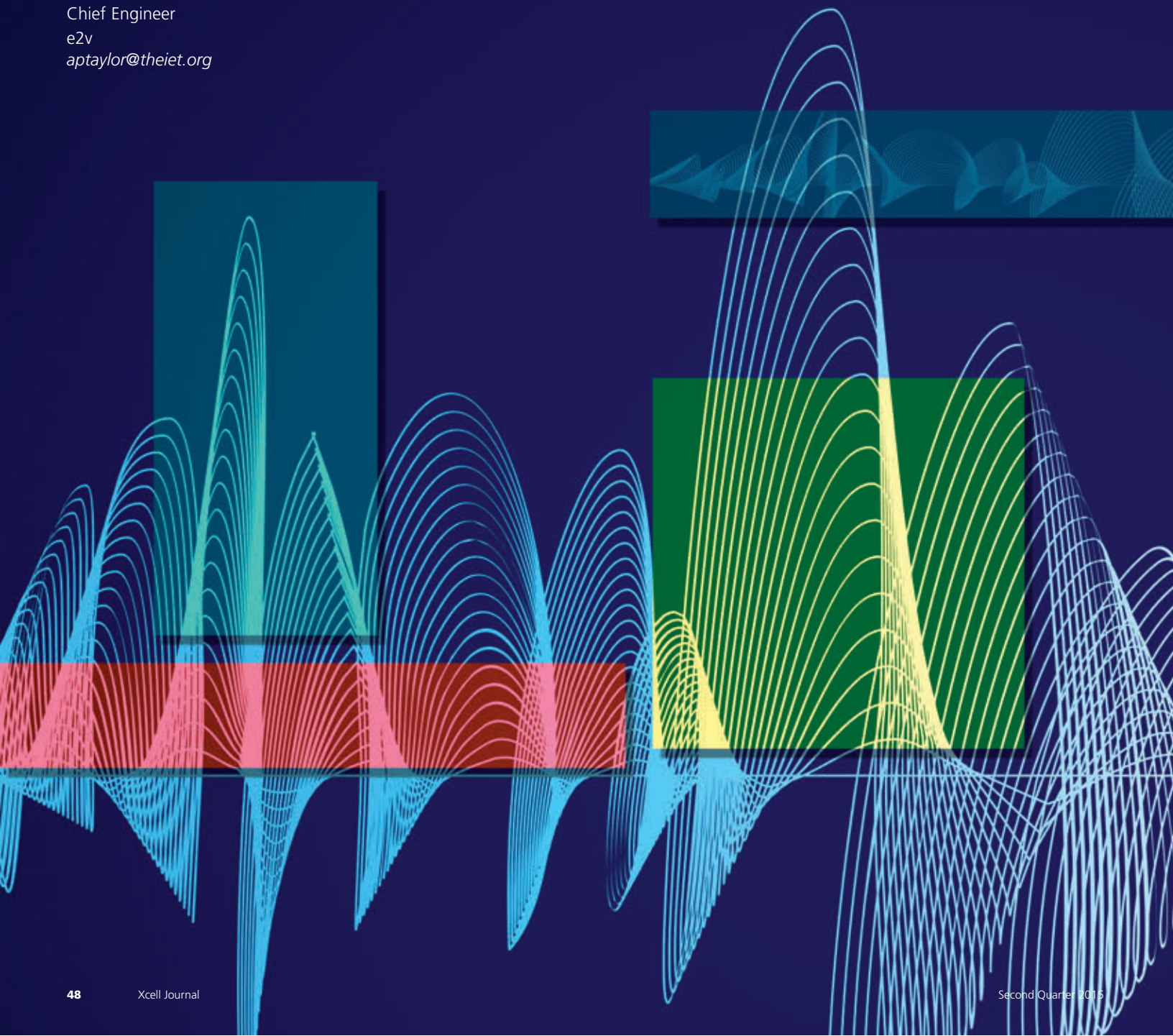
The optimizations necessary when creating applications like this one using SDAccel are software optimizations. Thus, these optimizations are similar to the ones required to extract performance from other processing fabrics, such as GPUs. As a result of using SDAccel, the details of getting the PCIe link to work, drivers, IP placement and interconnect became a non-issue, allowing us as designers to focus solely on the target application.

The optimizations we made in our watermarking application are applicable to all designs compiled using SDAccel. In fact, video watermarking provides a great "how to" introduction to the optimization methods Xilinx has made available in SDAccel.

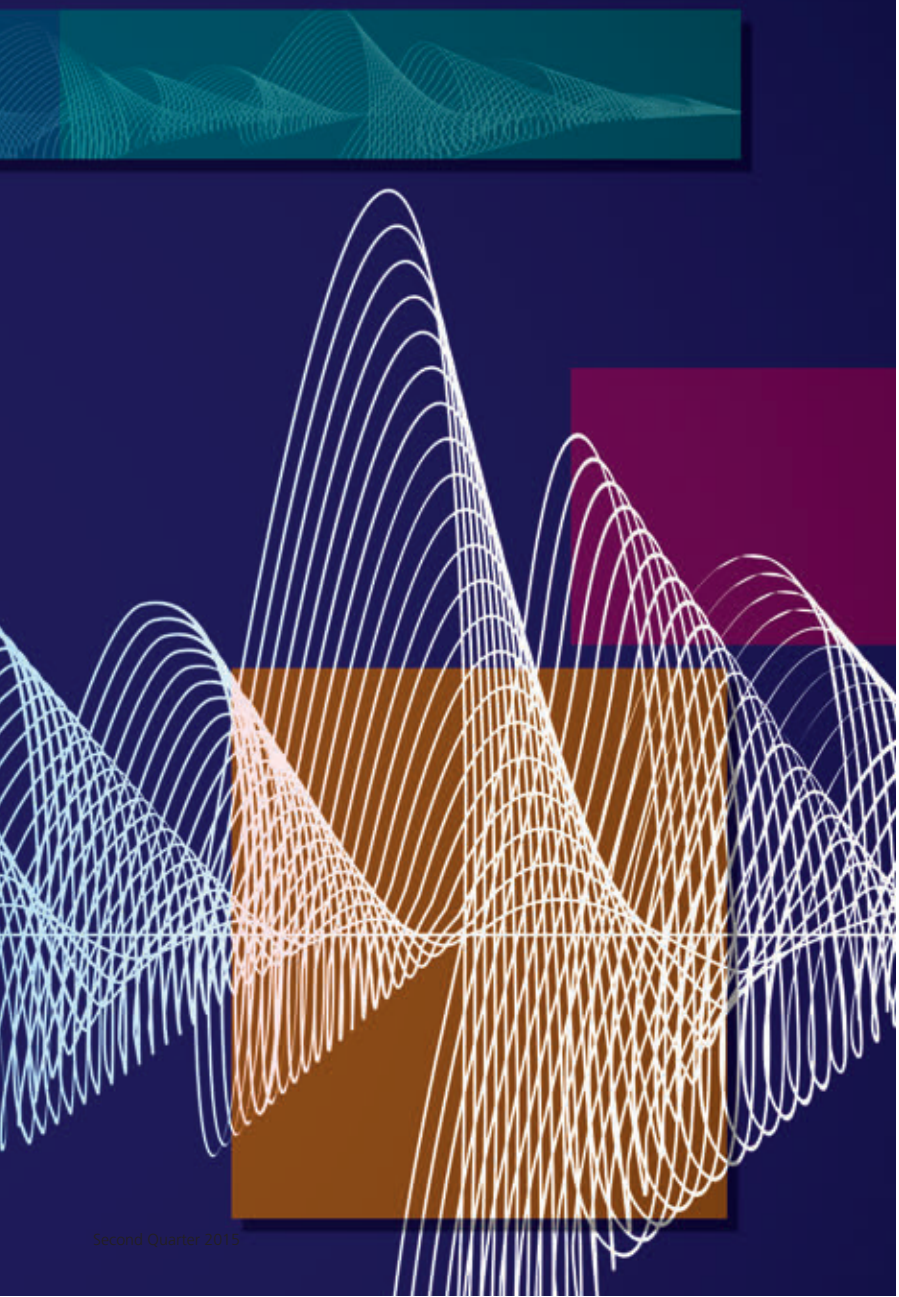

Figure 7 – Video frame partitioning into blocks

# Coming to Grips with the Frequency Domain

**by Adam P. Taylor**
Chief Engineer
e2v
*aptaylor@theiet.org*



The ability to work within the frequency domain is a necessity in a number of applications. Here's how the frequency domain factors into FPGA designs.



For many engineers, working in the frequency domain does not come as naturally as working within the time domain, probably because the frequency domain is associated with complicated mathematics. However, to unlock the true potential of Xilinx® FPGA-based solutions, you need to feel comfortable working within both of these domains.

The good news is that it's not as daunting as you might initially think to master the ins and outs of the frequency domain. Custom modules that you design yourself or IP modules available on the marketplace will help you transform to and from the frequency domain with ease. Methods also exist that make it possible to implement high-speed processing within the frequency domain.

## TIME OR FREQUENCY DOMAIN?

As engineers, we can examine and manipulate signals in either the time domain, where we analyze signals over time, or the frequency domain, where the analysis takes place with respect to frequency. Knowing when to do which is one of the core reasons an engineer is required on a project.

Typically in electronic systems, the signal in question is a changing voltage, current or frequency that has been output by a sensor or generated by another part of the system. Within the time domain, you can measure a signal's amplitude, frequency and period, along with more interesting parameters such as the rise and fall times of the signal. To observe a time domain signal in a laboratory environment, it is common to use an oscilloscope or logic analyzer.

However, there are parameters of a signal that are present within the frequency domain. You must analyze them there in order to access the information contained within. Here, you can identify the frequency components of the signal, their

Depending upon the type of signal—repetitive or nonrepetitive, discrete or nondiscrete—there are a number of methods you can use to convert between time and frequency domains.

amplitudes and the phase of each frequency. Working within the frequency domain also makes it much simpler to manipulate signals due to the ease with which convolution can be performed. Convolution is a mathematical way of combining two signals to form a third. As with the time domain analysis, if you wish within the laboratory environment to observe a frequency domain signal, you can use a spectrum analyzer.

For some applications, you will wish to work within the time domain, for example in systems that monitor the voltage or temperature of a larger system. While noise may be an issue, taking an average of a number of samples will in many cases be sufficient. However, for other applications it is preferable to work within the frequency domain. For example, signal-processing applications that require the filtering of one signal from another or demand that the signal be separated from a noise source are best analyzed within the frequency domain.

Working within the time domain requires little post-processing on the quantized digital signal as the sampling takes place within the time domain. By contrast, working within the frequency domain first requires the application of a transform to the quantized data to convert from the time domain. Similarly, to output the post-processed data from the frequency domain, you will need to perform the inverse transform again back to the time domain.

## HOW DO WE GET THERE?

Depending upon the type of signal—repetitive or nonrepetitive, discrete or nondiscrete—there are a number of methods you can use to convert between time and frequency domains, including Fourier series, Fourier transforms and Z transforms. Within electronic signal processing and FPGA applications in particular, you will most often be interested in one transform: the discrete Fourier transform (DFT), which is a subset of the Fourier transform. Engineers use the DFT to analyze signals that are periodic and discrete—that is, they consist of a number of n bit samples evenly spaced at a sampling frequency that in many applications is supplied by an ADC within the system.

At its simplest, what the DFT does is to decompose the input signal into two output signals that represent the sine and cosine components of that signal. Thus, for a time domain sequence of N samples, the DFT will return two groups of N/2+1 cosine and sine wave samples, respectively referred to as the real and imaginary components (Figure 1). The real and imaginary sample

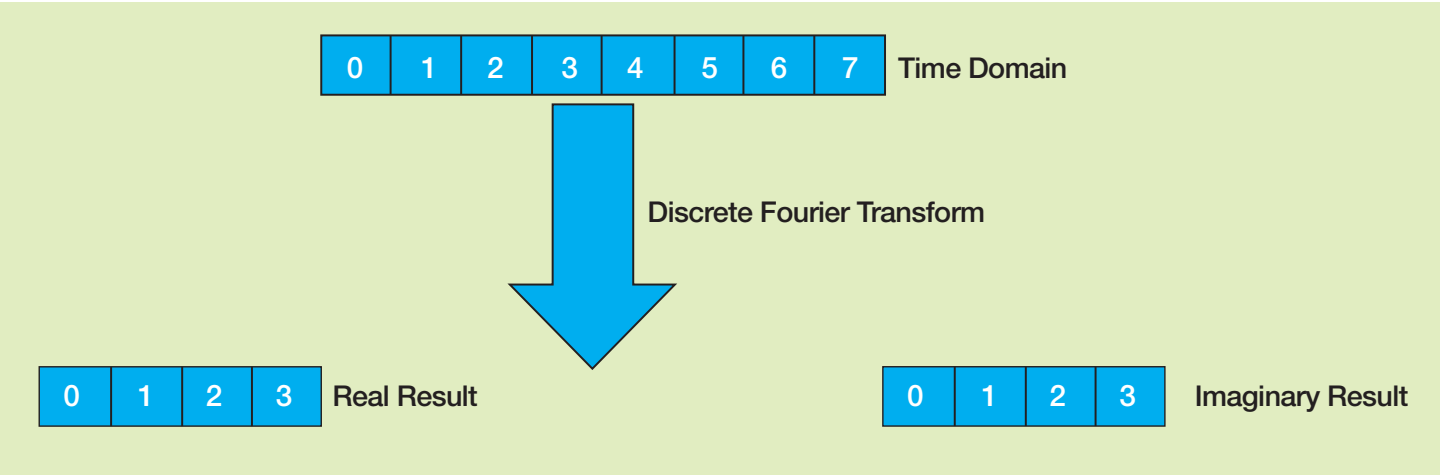



Figure 1 – N bits in time domain to n/2 real and imaginary bits in the frequency domain

width will also be n/2 for an input signal width of n bits.

The algorithm to calculate the DFT is pretty straightforward, as seen in the equation below:

$$ReX[k] = \sum_{i=0}^{N-1} x[i] \cos\left(\frac{2\pi ki}{N}\right)$$

$$ImX[k] = -\sum_{i=0}^{N-1} x[i] \sin\left(\frac{2\pi ki}{N}\right)$$

where x[i] is the time domain signal, i ranges from 0 to N-1 and k ranges from 0 to N/2. The algorithm is called the correlation method and what it does is to multiply the input signal with the sine or cosine wave for that iteration to determine its amplitude.

Of course, you will wish at some point in your application to transform back from the frequency domain into the time domain. For this purpose, you can use the synthesis equation, which combines the real and imaginary waveforms to re-create a time domain signal as such:

$$x[i] = \sum_{k=0}^{N/2} Re\bar{X}[k] \cos\left(\frac{2\pi ki}{N}\right) + \sum_{k=0}^{N/2} Im\bar{X}[k] \sin\left(\frac{2\pi ki}{N}\right)$$

However, *ReX* and *ImX* are scaled versions of the cosine and sine waves. Therefore, you will need to scale them. Hence, ImX[k] or ReX[k] is divided by N/2 to determine the values for *ReX* and *ImX* in all cases except when ReX[0] and ReX[N/2]. In this case, they are divided by N. This is, for obvious reasons, called the inverse discrete Fourier transform, or IDFT.

Having explored the algorithms used for determining the DFT and IDFT, it may be helpful to know what you can use them for.

You can use tools such as Octave, MATLAB® and even Excel to perform DFT calculations upon captured data, and many lab tools, such as oscilloscopes, are capable of performing DFT upon request.

However, it is worth pointing out that both the DFT and IDFT above are referred to as real DFT and real IDFT in that the input is a real number and not complex. Why you need to know this will become apparent.

## WHERE DO WE USE THESE TRANSFORMS?

From telecommunications to image processing, radar and sonar, it is hard to think of a more powerful and adaptable analysis technique to implement within an FPGA than the Fourier transform. Indeed, the DFT forms the foundation for one of the most commonly used FPGA-based applications: It is the basis for generating the coefficients of the finite input response (FIR) filter (see *Xcell Journal* issue 78, "Ins and Outs of Digital Filter Design and Implementation").

However, its use is not just limited to filtering. The DFT and IDFT are used in telecommunications processing to perform channelization and recombination of the telecommunication channels. In spectral-monitoring applications, they are used to determine what frequencies are present within the monitored bandwidth, while in image processing the DFT and IDFT are used to handle convolution of images with a filter kernel to perform, for example, image pattern recognition. All of these applications are typically implemented using a more efficient algorithm to calculate the DFT than the one shown above.

All told, the ability to understand and implement a DFT within your FPGA is a skill that every FPGA developer should have.

## FPGA-BASED IMPLEMENTATION

The implementation of the DFT and IDFT as described above is often done as a nested loop, each loop performing N calculations. As such, the time it takes to implement the DFT calculation is

$$DFTtime = N * N * Kd\,ft$$

where Kdft is the processing time for each iteration to be performed. Clearly, this can become quite time-consuming to implement. For that reason, DFTs within FPGAs are normally implemented using an algorithm called the fast Fourier transform. This FFT has often been called the the most important algorithm of our lifetime, as it has had such an enabling impact on many industries.

The FFT differs slightly from the DFT algorithms in that it calculates the complex DFT—that is, it expects real and imaginary time domain signals and produces results in the frequency domain which are n bits wide as opposed to n/2. This means that when you wish to calculate a real DFT, you must first set the imaginary part to zero and then move the time domain signal into the real part. If you wish to implement an FFT within your Xilinx FPGA, you have two options. You can write an FFT from scratch using the HDL of your choice or you can use the FFT IP provided within the Vivado® Design Suite IP Catalog or another source. Unless there are pressing reasons not to use the IP, the reduced development time resulting from using the Xilinx core should drive its selection.

The basic approach of the FFT is to decompose the time domain signal into a number of single-point time domain signals. This process is often called bit reversal as the samples are reordered. The number of stages that it takes to create these single-point time domain signals is calculated by Log2 N, where N is the number of bits, if a bit reversal algorithm is not used as a short-cut.

These single-point time domain signals are then used to calculate the frequency spectra for each of these points. This operation is pretty straightforward, as the frequency spectrum is equal to the single-point time domain.

It is in the recombination of these single frequency points that the FFT algorithm gets complicated. You must recombine these spectra points one stage at a time, which is the opposite of the time domain decomposition. Thus, it will again take Log2 N stages to re-create the spectra, and this is where the famous FFT butterfly comes into play.

When compared with the DFT execution time, the FFT takes

$$FFTtime = K\,fft * N\,Log2\,N$$

which results in significant improvements in execution time to calculate a DFT.

When implementing an FFT within an FPGA, you must also take into

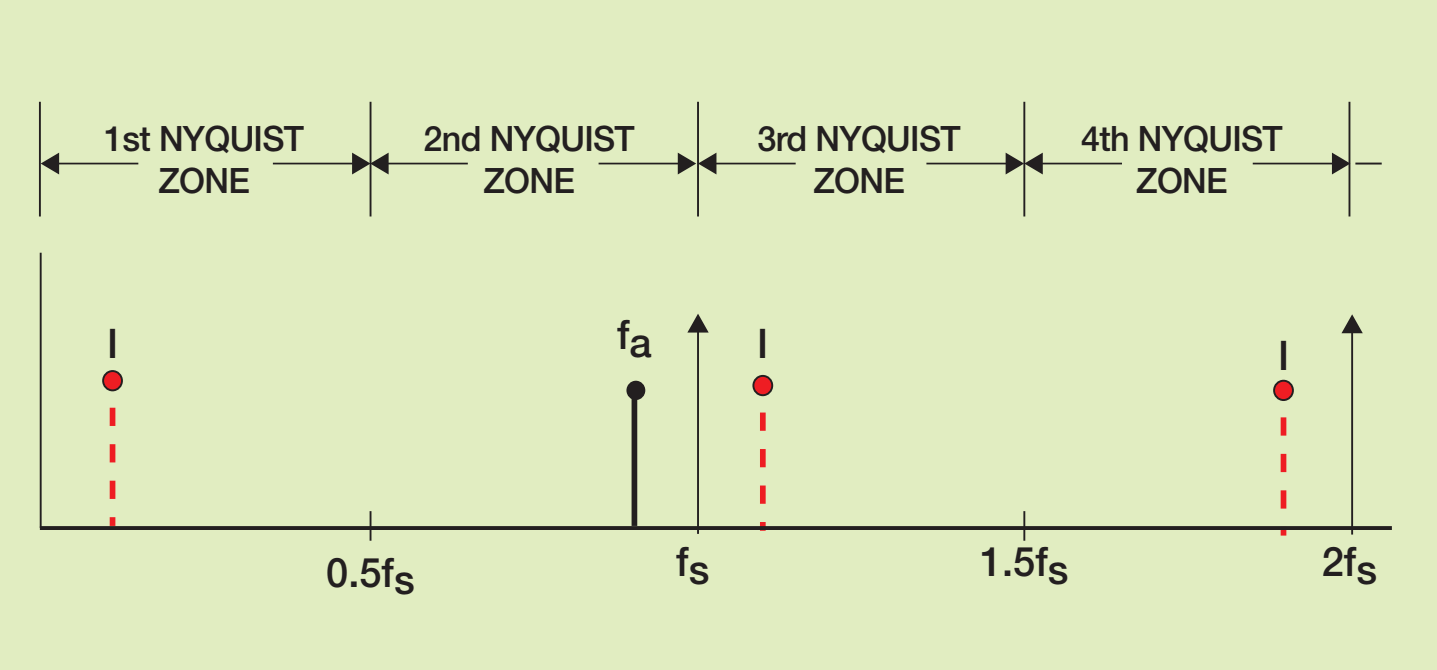



Figure 2 – Nyquist zones and aliasing

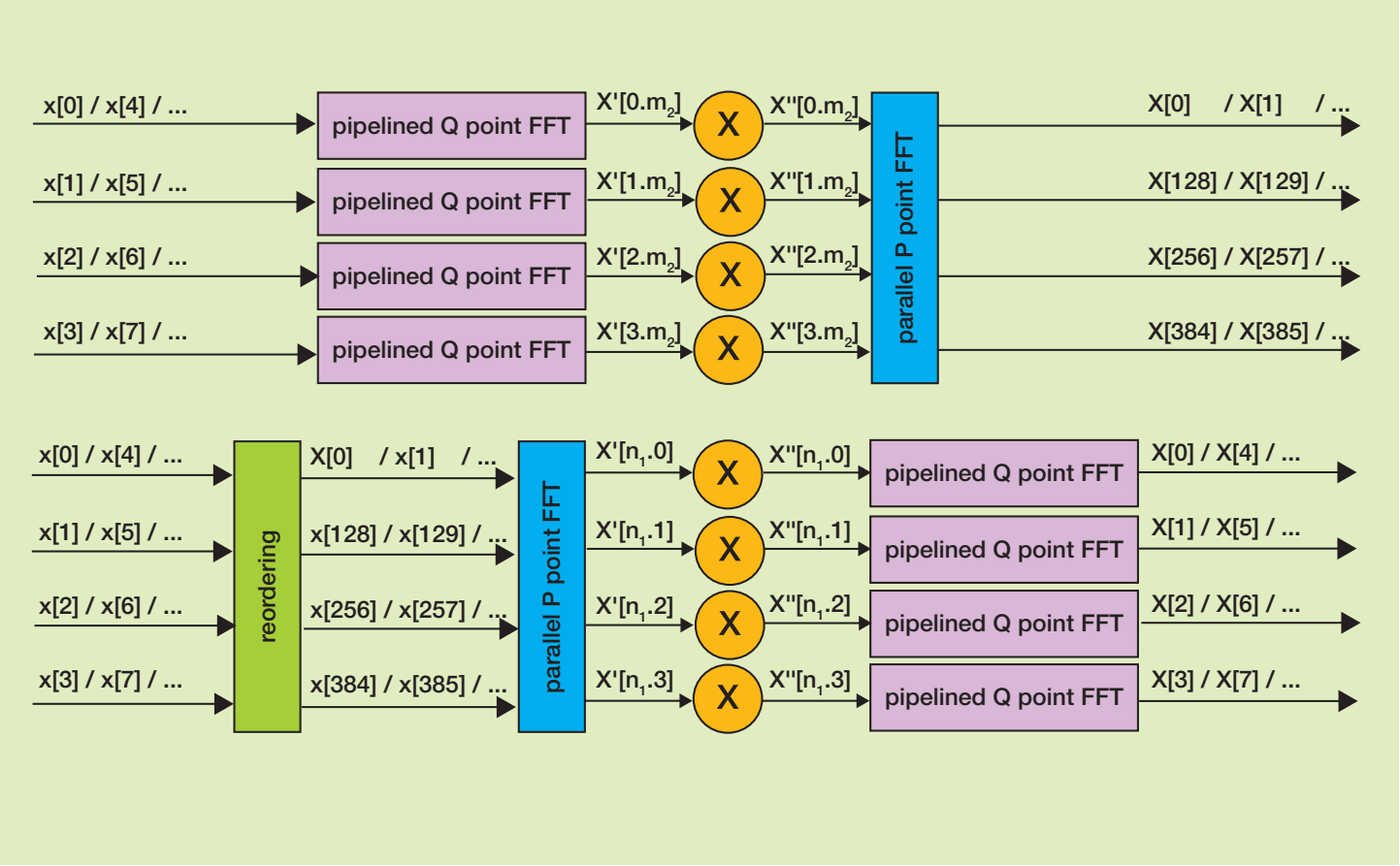



Figure 3 – Split and combined FFT structures

account the size of the FFT. The size will determine the noise floor below which you cannot see signals of potential interest. The FFT size will also determine the spacing of the frequency bins. Use the equation below to determine the FFT's size:

$$FFTNoise\ Floor\ (dB) = 6.02n + 1.77 + 10log10(\frac{FFTsize}{2})$$

where n is the number of quantized bits within the time domain and FFTSize is the FFT size. For FPGA-based implementation, this is normally a power of two—for example, 256, 512, 1,024, etc. The frequency bins will be evenly spaced at

$$Bin\ Width = \frac{\frac{Fs}{2}}{FFTsize}$$

For a very simple example, a sampling frequency (FS) of 100 MHz with an FFT size of 128 would have a frequency resolution of 0.39 Hz. This of course means frequencies within 0.39 Hz of each other cannot be distinguished.

## HIGHER-SPEED SAMPLING

Many applications for FFTs within FPGAs and higher-performance systems operate at very high frequencies. High-frequency operation can present its own implementation challenges.

At high frequencies, the Nyquist sample rate (sampling at least two samples per cycle) simply cannot be maintained. Therefore, a different approach is needed. An example would be using an ADC to sample a 3-GHz full-power-bandwidth analog input with a 2.5-GHz sample rate. Using Nyquist-rate criteria, signals above 1.25 GHz will be aliased back into the first Nyquist zone to be of use. These aliased images are harmonic components of the fundamental signal and thus contain the same information as the non-aliased signal, as shown in Figure 2.

To determine the resultant frequency location of the harmonic or harmonic content, you can use the algorithm below:

*Fharm = N × Ffund*
*IF (Fharm = Odd Nyquist Zone)*
*Floc = Fharm Mod Ffund*
*Else*
*Floc = Ffund-(Fharm Mod Ffund)*
*End*

where N is the integer for the harmonic of interest.

Continuing our example further, with a sample rate of 2500 MHz and a fundamental of 1807 MHz, there will be a harmonic component at 693 MHz within the first Nyquist zone that we can further process within our FFT.

Having grasped the basics about the frequency spectrum, the next crucial factor to consider is the way you interface these ADC and DAC devices to the FPGA. It is not possible for the data from the ADC to be received at FS/2 where in the example above, the sampling frequency is 2.5 Gbps. For this reason, high-performance data converters use multiplexed digital inputs and outputs that operate at a lower data rate with respect to the converter's sample rate, typically FS/4 or FS/2.

Having received the data from the FPGA in a number of data streams, the next question is how you can process the data internally within the FPGA if you wish to perform a DFT. One common method used for a number of applications, including telecommunication processors and radio astronomy, is to use combined or split FFT structures, as shown in Figure 3.

While this application is more complicated than a straightforward FFT, such an approach makes it possible to achieve the higher-speed processing.

As you can see, working within the frequency domain is not as difficult as you may initially think, especially when there are IP modules to help in transforming to and from the frequency domain. Moreover, a number of methods are available that will enable you to implement high-speed processing.

# Marrying SoC Platform Designs with System Generator for DSP

**by Daniel E. Michek**
Senior Manager, System-Level Product Marketing
Xilinx, Inc.
*daniel.michek@xilinx.com*

The Vivado System Generator tool easily connects into platform designs to leverage board interfaces and processing systems.

FPGA applications are ever evolving, and the FPGA design flows are evolving along with them. No longer do we use FPGAs as simple glue logic or even as the heart of a signal-processing chain that marries intellectual property (IP) to proprietary back-end interfaces. Instead, FPGAs are transitioning into programmable systems-on-a-chip, a combination of hardware designed to act as processor peripherals alongside high-level software running on powerful APUs. This is an architecture we refer to as Xilinx® All Programmable SoCs.

In order to take advantage of this new flow, we need to shift the design methodology from top-down RTL, as in the earliest days of the FPGA, to a bottom-up flow centered around IP development and standardized connections such as ARM®'s Advanced eXtensible Interface (AXI). As the interfaces evolve from custom to common, we reduce the effort expended on verifying interactions between the data path and the platform design.

Xilinx's System Generator for DSP has evolved too. This tool, which is part of the Vivado® Design Suite, integrates the new bottom-up design methodology by incorporating DSP data paths into platform designs constructed with the Vivado IP Integrator tool. Let's take a closer look at how design automation using System Generator is enabling high-performance designs to take advantage of platform connectivity.

## BUILDING AN ALL PROGRAMMABLE PLATFORM FRAMEWORK

We start our new design flow by defining the platform framework in which we want to house the data path. The Vivado tool suite is board aware and we will take advantage of available board automation to build our new platform design.

A platform design or platform framework, as shown in Figure 1, is the basic collection of processor- and board-level interfaces, along with the logic combining them. We use the platform framework as the basis for our system-level design, the shell, and this leaves us with the room for our data path. Block and connectivity automation will link the processing systems through IP peripherals to the board-level interfaces. DSP data paths or software accelerators packaged in the IP Catalog can then take advantage of Xilinx's Designer Assistance automation to easily tie into our platform framework of processors and, by extension, interfaces to external devices.

## CREATING THE DATA PATH AS IMPORTABLE IP

Our ultimate goal is to make the data path accessible to the All Programmable platform framework. If we wanted to start from scratch, we could create the data path with standardized interfaces. By quickly marking a gateway port as an AXI4-Lite interface as shown in Figure 2, or simply naming our ports to match a standard connection like AXI4-Stream on our Simulink® diagram, System Generator will take care of adding the extra logic to our design and collecting the common signals into interfaces as it packages the design for the Vivado IP Catalog.

But a new method allows us to use the platform framework to tailor-fit a plug-in that marries to the All Programmable design. We use automation to determine which interfaces exist within a platform design, which interfaces associate to the board and which interfaces create a plug-in for the DSP data path. Since the goal is to convert the data path into IP that connects to the platform framework, we do not need to focus on board-level interfaces but instead, on standardized AXI interfaces. Each interface not associated at the board level converts into System Generator gateways. While acting as

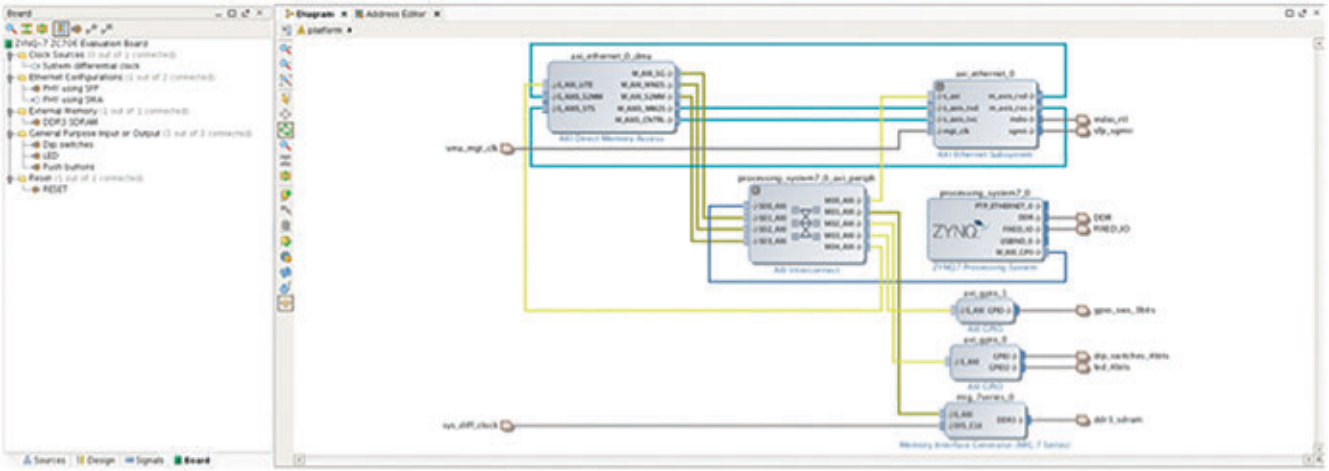


Figure 1 – Example of a platform framework connecting a processing system to board-level interfaces

simple signals in the System Generator world, these gateways produce AXI interfaces to connect to the platform design when we export it to the IP Catalog.

As one example, AXI4-Lite interfaces create independent read and write signals that share a commonly addressable register interface when exported to the Vivado tool suite. A trivial copy-and-paste gives us more direct registers available to the processor at an address offset through the same interface. At the same time, we

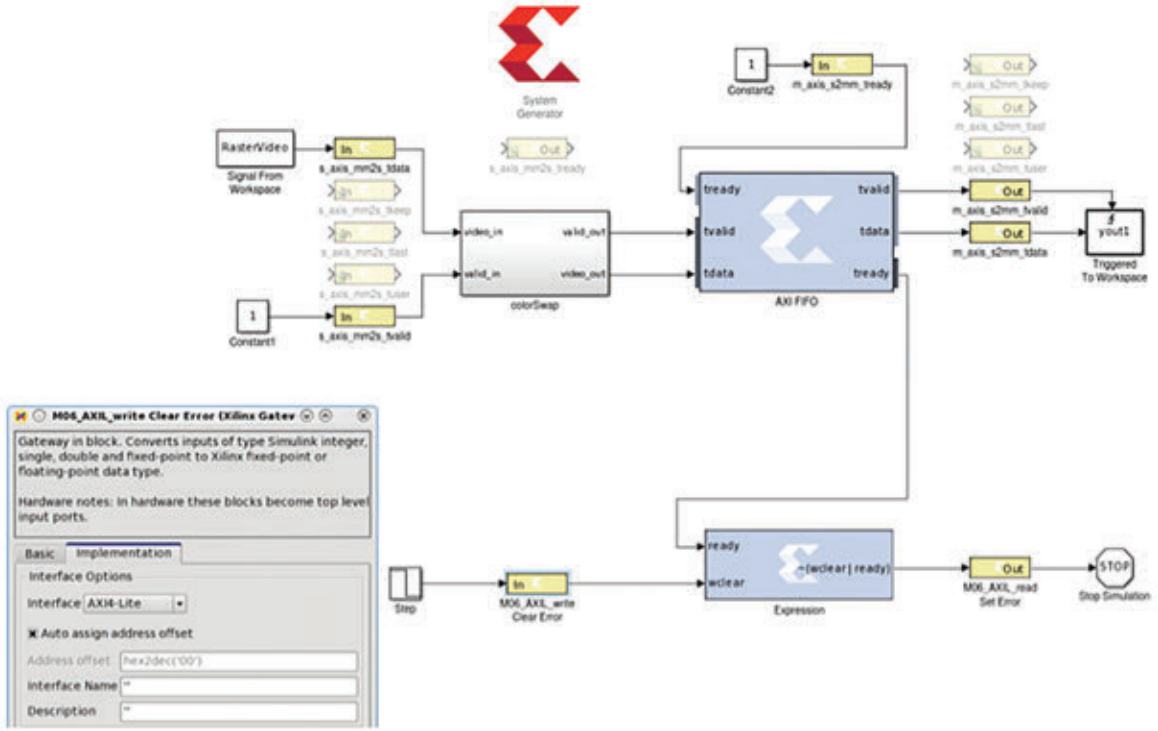



Figure 2 – Automating gateways into AXI4-Lite and AXI4-Stream interfaces

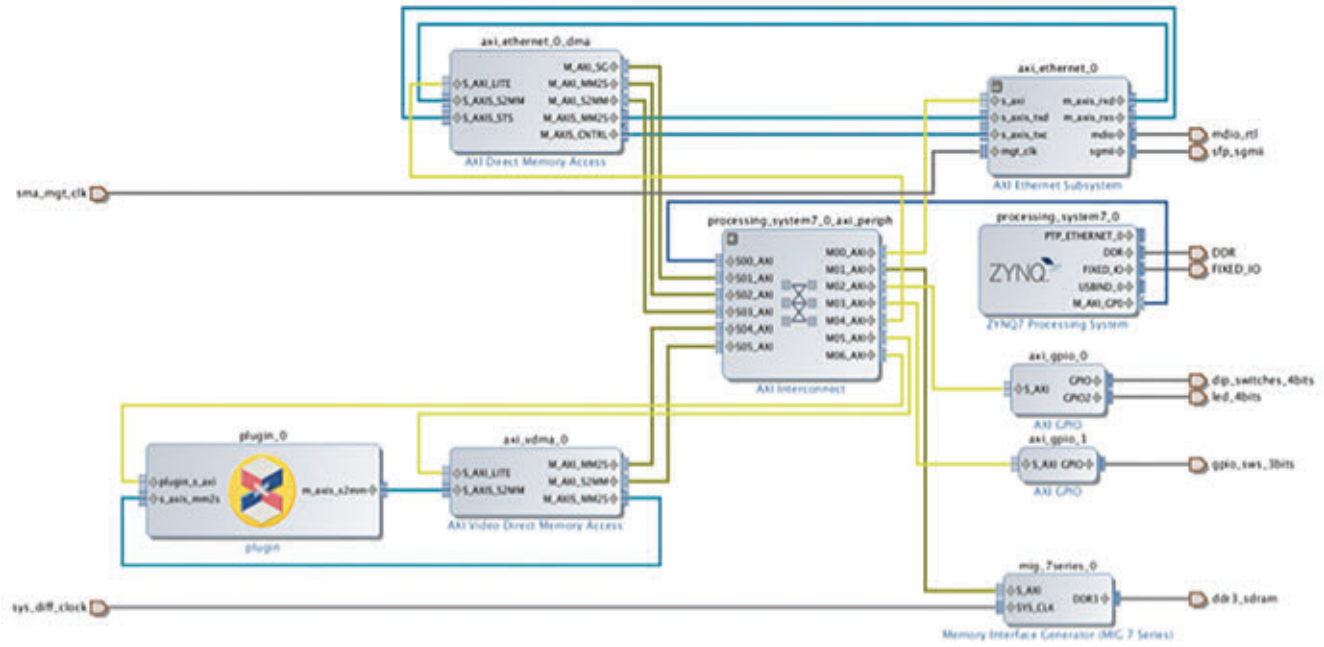



Figure 3 – Platform-based system that connects DSP data path to the platform framework

automatically generate software driver APIs to read or write to those registers.

If an AXI4-Stream interface is available from the platform design, System Generator will add appropriately matched gateways to the model. AXI4-Stream interfaces are extremely flexible and contain many signals. ACLK is the clock source associated with this interface, but this signal is directly implied as the abstracted system clock for this portion of the data path. TVALID is the signal that denotes when the interface is valid. Other signals are optional. System Generator will add those that exist in the originating stream interface to our model, but we can delete or add signals to match our internal requirements.

In the model shown in Figure 2, we see that our data path only cares about TDATA (the data sent across the interface) and TVALID on the S_AXIS interface. To remove unnecessary signals, we comment out the unused gateways for this model, since default values will drive the signal connections in IP Integrator.

The AXI4-Lite and AXI4-Stream signals easily simulate and verify with the bottom-up methodology. The AXI4-Lite interfaces model as simple gateways that we source and sink from the myriad of Simulink blocks, a simple abstraction of passing data from one side of the gateway to the other. Likewise, the AXI4-Stream interfaces are merely a collection of signals that follow simple handshake rules for passing data from one IP core to another.

Our simulation modeling is only challenged by the optional ports we use on our interface. If we accept data on every cycle and process it though our data path without interruption, we will not need the TREADY handshake signal. The simplified model sends each element of a vector through the TDATA gateway from Simulink's Signal to Workspace token. When a full handshake is desired, we model it with an AXI4-Stream FIFO to buffer our data as shown on the M_AXIS interface in Figure 2.

The tailoring automation acts as a starting point to create IP that interfaces to the platform framework. But System Generator is flexible and allows us to add or delete portions of, or even complete, AXI interfaces. The end result is a conversion of the data path into IP for reuse among multiple system-level designs.

After adding our logic, our last step is to export the data path we built with System Generator for DSP back into the Vivado IP Catalog. This action allows easy connection of the interfaces, whether working with RTL or within IP Integrator. Additionally, we generate drivers for use in the SDK and attach models for simulation with golden test vector data to the IP. And because we have prior knowledge of the platform framework when we created our DSP data path, we automate the incorporation of the model and the platform design, as shown in the completed system in Figure 3.

## REDUCING SIMULATION RISK

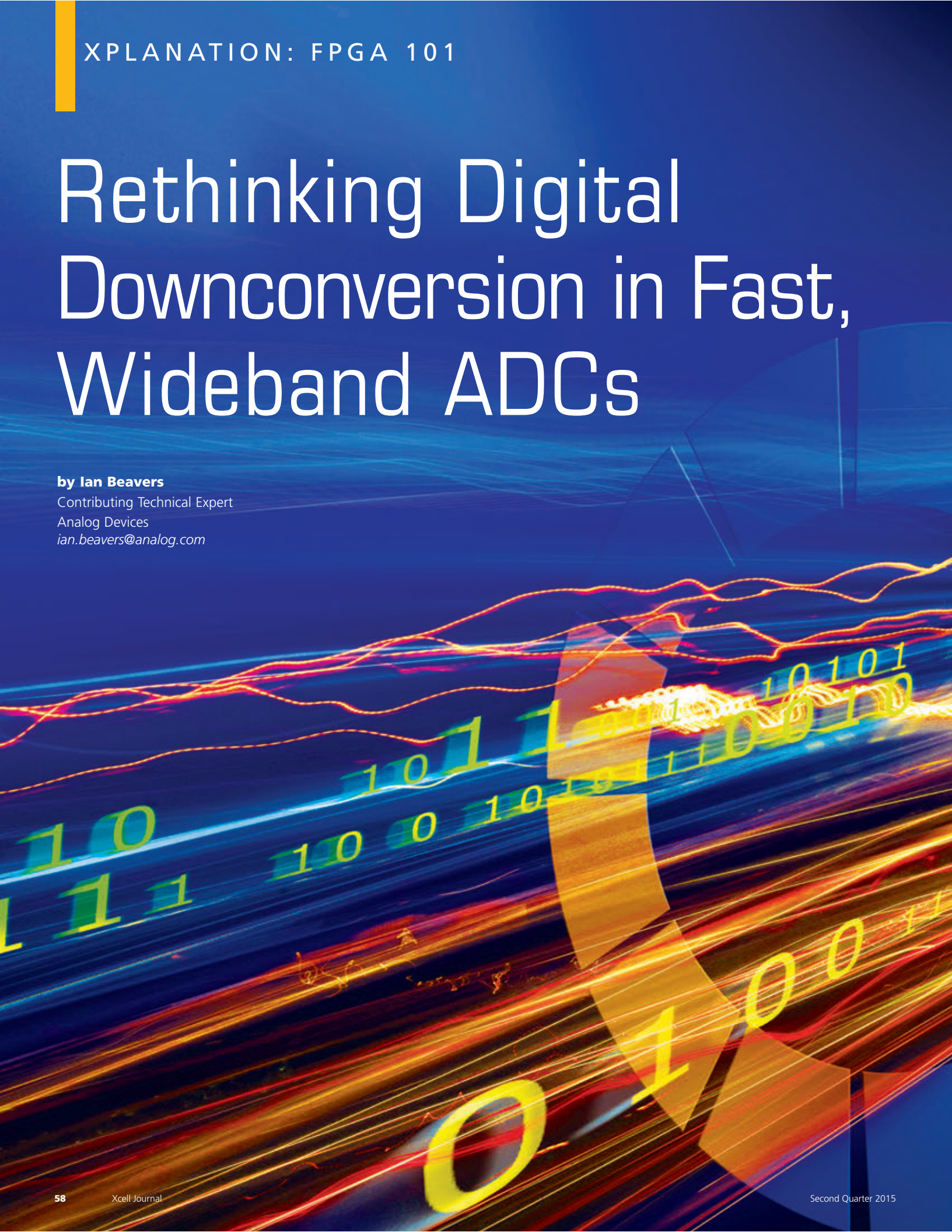
Generating a complete system-on-chip that incorporates hardware accelerators, DSP data paths or custom logic is a challenge. Confirming that the data path will work as expected by simulating in a bottom-up fashion introduces a large risk, and ensuring that we maintain platform interface bandwidth to support the data path is equally daunting.

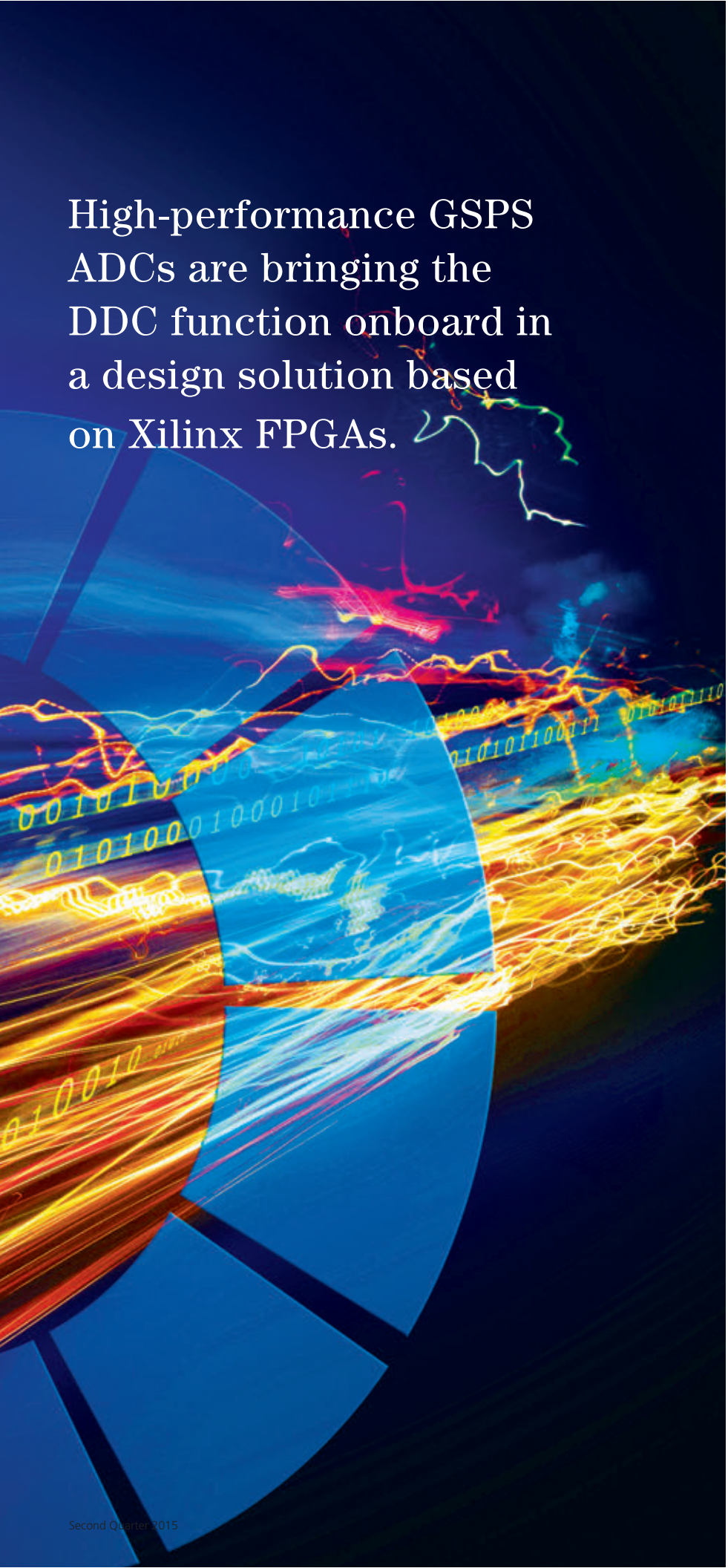
By utilizing standardized interfaces, we develop IP that reduces simulation risk. That's because the interactions at the interface level abstract away, so we may focus on verifying the internal data path. And finally, by leveraging smart automation for board, block and connectivity, we generate the platform-based system that meets our needs and integrates our custom data path.

More information about System Generator for DSP is available at *www.xilinx.com/products/design-toolsvivado/integration/sysgen.html*. If you have any questions or comments, call the author, Daniel Michek, at (858) 207-5213, or e-mail *daniel.michek@xilinx.com*.

# Rethinking Digital Downconversion in Fast, Wideband ADCs

**by Ian Beavers**
Contributing Technical Expert
Analog Devices
*ian.beavers@analog.com*





# High-performance GSPS ADCs are bringing the DDC function onboard in a design solution based on Xilinx FPGAs.

Wideband gigasample-per-second (GSPS) analog-to-digital converters offer many performance benefits to high-speed acquisition systems. These ADCs provide a wide frequency spectrum of visibility across high sample rates and input bandwidths. However, while some applications need a wideband front end, others require the ability to filter and tune to a narrower band of spectrum.

It can be inherently inefficient for an ADC to sample, process and burn the power to transmit a wideband spectrum, when only a narrow band is required in the application. An unnecessary system burden is created when the data link consumes a large bank of high-speed transceivers within a Xilinx® FPGA, only to then decimate and filter the wideband data in subsequent processing. The Xilinx FPGA transceiver resources can instead be better allocated to receive the lower bandwidth of interest and channelize the data from multiple ADCs. Additional filtering can be done within the FPGA's polyphase filter bank channelizer for frequency-division multiplexed (FDM) applications.

High-performance GSPS ADCs are now bringing the digital downconversion (DDC) function further up in the signal chain to reside within the ADC in a design solution based on Xilinx FPGAs. This approach offers several new design options to a high-speed system architect. However, because this function is relatively new to the ADC, there are design-related questions that engineers may have about the operation of the DDC blocks within GSPS ADCs. Let's clear up some of the more-common questions so that designers can begin using this new technique with more confidence.

To get the full performance benefit of DDCs, the design must also contain a filter-and-mixer component as a companion to the decimation.

## WHAT IS DECIMATION?

In the simplest definition, decimation is the method of observing only a periodic subportion of the ADC output samples, while ignoring the rest. The result is to effectively reduce the sample rate of the ADC by downsampling. For example, a decimate-by-M mode in an ADC outputs only the first of Mth samples, while discarding all the other samples in between. This method continues to repeat for each multiple of M.

Sample decimation alone will only effectively reduce the sample rate of the ADC and correspondingly act as a low-pass filter. Without frequency translation and digital filtering, decimation will merely fold the harmonics of the fundamental and other spurious signals on top of one another in the frequency domain.

## WHAT IS THE ROLE OF THE DDC?

Since decimation by itself does not prevent the folding of out-of-band signals, how does the DDC make this happen?

To get the full performance benefit of DDCs, the design must also contain a filter-and-mixer component that's used as a companion to the decimation function. Digital filtering effectively removes the out-of-band noise from the narrowly defined bandwidth that is set by the decimation ratio. The typical digital filter implementation for a DDC is a finite impulse response (FIR) filter. This filter is a function only of the past inputs, since there is no feedback. The passband of the filter should match the effective frequency spectrum width of the converter after the decimation.

## HOW WIDE SHOULD THE DDC FILTERS BE?

The decimation ratios for DDCs are typically based on integer factors that are

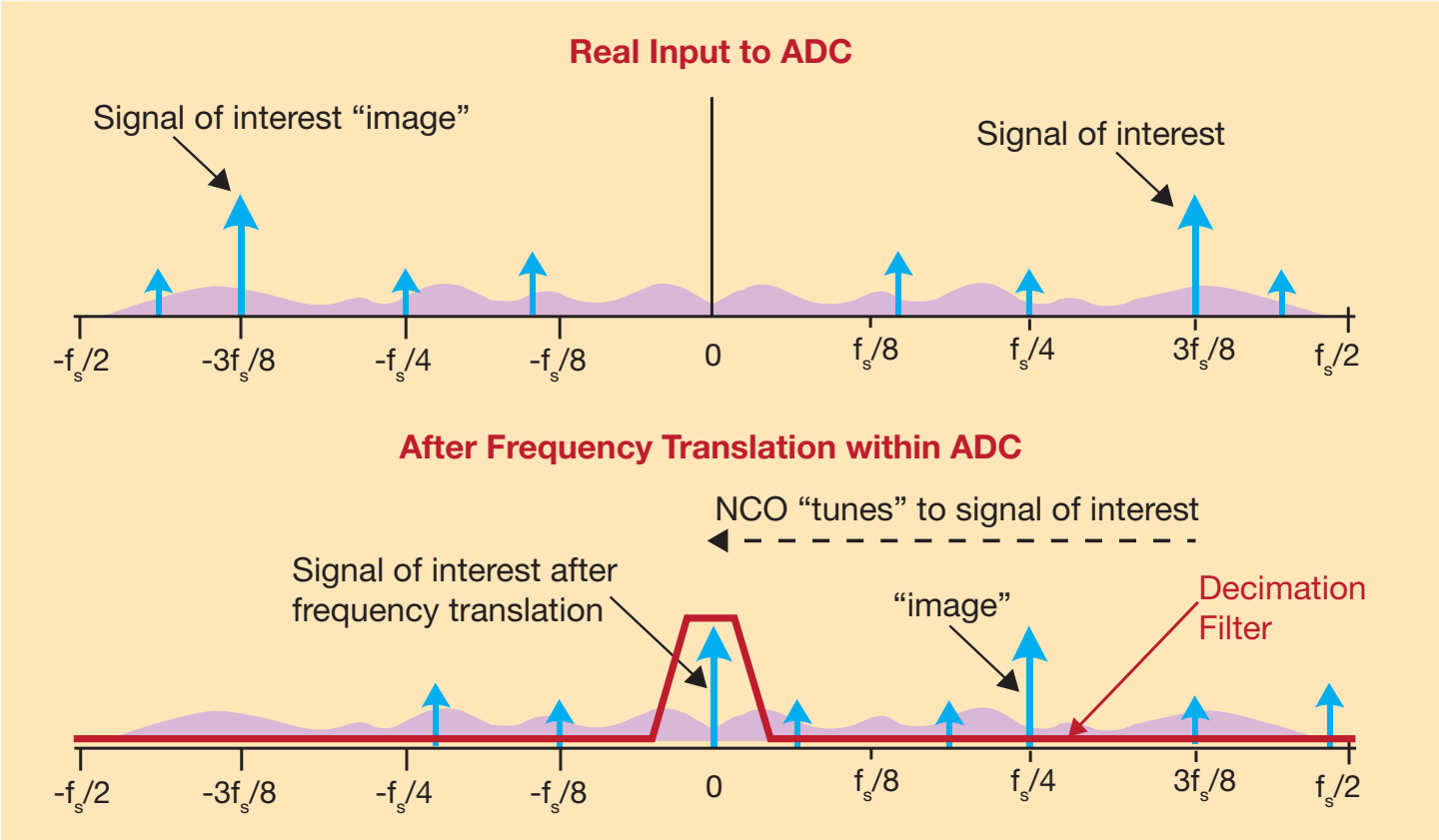



Figure 1 – Frequency translation using a low-pass filter and NCO effectively achieves a bandpass filter at the frequency of interest. Frequency planning ensures that unwanted harmonics, spurs and images fall out of band.

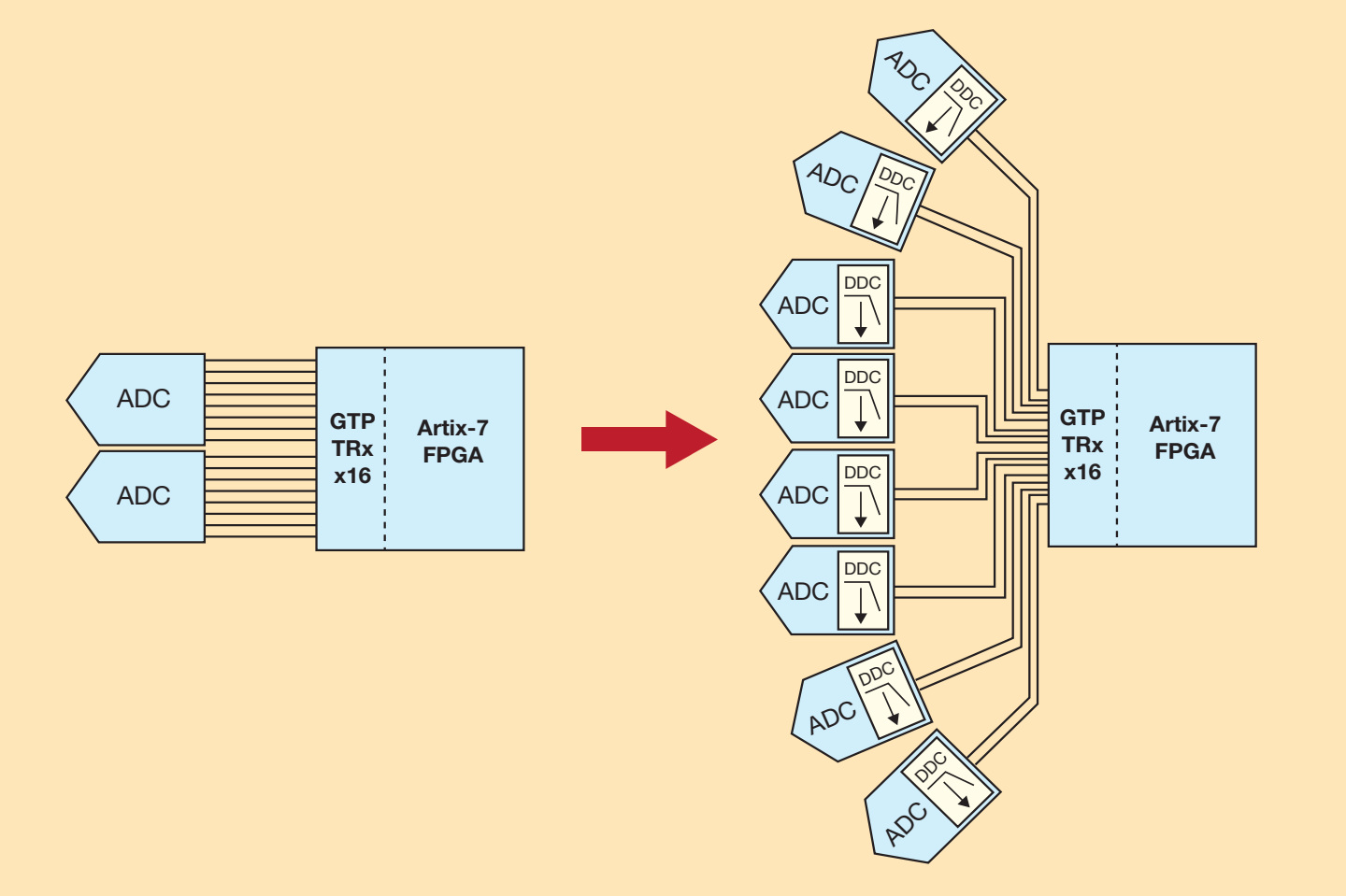



Figure 2 – The use of DDCs with a decimation factor of 8 allows the same 16 GTP 6.6-Gbps transceivers of the Xilinx Artix-7 to support eight ADCs with decimated I/Q data on two lanes of JESD204B each vs. only two ADCs that each output full bandwidth over eight lanes.

powers of 2 (2, 4, 8, 16, etc.). However, the decimation factor could actually be any ratio based on the DDC architecture, including fractional decimation. In the case of fractional decimation, an interpolation computation block is typically needed ahead of decimation to achieve a rational fraction ratio.

Ideally, the digital filter would precisely match the decimation frequency bandwidth and filter everything outside that band. However, a practical effective filter width will not exactly match to the full bandwidth of the decimation ratio. The filter width will therefore be some percentage of the decimation frequency, such as 85 percent or 90 percent. For example, the usable bandwidth of a decimation factor-of-8 filter may be practically the sample rate divided by 10, or fs/10. The DDC filtering stage must provide a low passband ripple and a high stopband alias rejection.

## IS THE FREQUENCY FIXED?

The next question is whether the DDC filters are fixed in frequency, or if they can be tuned and centered on a particular band of interest.

We have discussed the decimation and filtering stages of DDCs. But this is only valuable if the desired frequency is within the filter passband from DC. If that is not the case, then we need a way to tune the filter to a different part of the frequency spectrum to observe the signal of interest. The narrow bandwidth can be tuned within the first or second Nyquist zone by a numerically controlled oscillator (NCO). An NCO offers a method to tune and mix the filter band to a different portion of the wideband spectrum (Figure 1).

A digital tuning word provides a fractional divider of the sample rate with a frequency placement resolution defined by the number of bits used in the digital tuning word that allows the band of interest to be mixed. The tuning word has the tuning range and resolution to place the filter where it is needed. A typical NCO tuning word may be up to 48 bits of resolution across two Nyquist bands of the sampled frequency, which is adequate for most applications.

The NCO is accompanied by a mixer. Operating much like an analog quadrature mixer, this device performs the downconversion of real and complex input signals by using the NCO frequency as a local oscillator.

The filter follows the frequency translation stage. After the carrier band of interest is tuned down to DC, the filter effectively lowers the sample rate while providing sufficient alias rejection from unwanted adjacent carriers around the tuned bandwidth of interest.

# The use of a single decimate-by-8 DDC allows the same Xilinx Artix-7 FPGA system to support four times more ADCs.

When mixing a real input signal down to baseband, 6 dB of signal loss is introduced due to the filtering of the negative image. The NCO introduces an additional small insertion loss. The total loss of a real input signal mixed down to baseband is typically slightly more than 6 dB. The NCO allows the input spectrum to be tuned to dc, where it can be effectively filtered by the subsequent filter blocks to prevent aliasing. The DDC may also contain an independently controlled digital gain stage. A gain stage allows the system to enable +6 dB or more of gain to center the dynamic range of the signal within the full scale of the output bits.

## INTERPROCESSOR INTERRUPTS

The decimation of the ADC samples removes the need to send unwanted information downstream in the signal chain to eventually get discarded anyway. Therefore, since this data is filtered out, it reduces the output data bandwidth needed on the back end of the ADC. This amount of reduction is offset by the increase in data from both the I/Q data output. For example, a decimate-by-16 filter with both I and Q data would reduce the wideband output data by a factor of 8.

This minimized data rate reduces the complexity of system layout by lowering the number of output JESD204B lanes from the ADC. The reduction in ADC output bandwidth can allow the design of a compact system that otherwise may not be achievable. For example, in a case where system power and size limit a board to using a single FPGA, the number of high-speed serial transceivers supported can limit the number of ADCs without the use of DDCs.

For a case where only a narrow bandwidth is observed in this system, decimation within the ADCs helps remove this limitation. The use of a single decimate-by-8 DDC allows the same Xilinx Artix®-7 FPGA system to support four times more ADCs by reducing the output bandwidth of the ADCs to just two output data lanes. For this particular case, as many as eight ADCs using DDCs could now be designed with the same existing 16 GTP transceivers in the Artix-7 FPGA (Figure 2). This allows more efficient use of Xilinx FPGA resources as a multichannel digital receiver for a set of FDM channels.

## DO DDC FILTERS AFFECT SNR AND SFDR?

The next question to examine is how the analog performance of signal-to-noise ratio (SNR) and spurious-free dynamic range (SFDR) change when the DDC filters are on vs. when they are off.

Since the wideband noise of the converter is filtered out and only a narrow spectrum is observed, we should expect the signal power relative to the observed noise to be higher. The dynamic range of the ADC will be better within the passband of the filter. The improved SNR by use of the DDC is inherently an advantage of decimating and filtering the wideband spectrum.

Digital filtering by the DDC is used to filter the noise outside of a smaller bandwidth. The SNR calculation of the ADC must then include a correction factor for this filtering that accounts for the processing gain of the filtered noise. Using a perfect digital filter, for every power-of-two reduction in bandwidth, the processing gain due to the filtered noise will increase by +3 dB:

*Ideal SNR (with processing gain) = 6.02*N + 1.76 dB + 10log10(fs/(2*BW))*

A distinct advantage of using DDCs is the ability to have the harmonics of the fundamental signal fall outside the band of interest. With proper frequency planning, digital filtering will prevent the harmonics from being seen within the narrow DDC bandwidth and therefore will increase the SFDR performance of the system.

In the systems where only a narrow bandwidth is needed, a DDC provides ADC processing gain by filtering out the wideband noise. This increases the signal-to-noise ratio seen within the bandwidth of interest. An additional benefit is that, with proper frequency planning, the typically dominant second- and third-order harmonics of the fundamental fall outside the tuned bandwidth of interest and are digitally filtered. This increases the SFDR of the system.

Sampling theorems dictate that harmonics or other higher-order system spurs can fold back around the end of each Nyquist band. This is also true for DDCs, which have the potential for unwanted second- or third-order harmonics to fold back into the passband and decrease the SFDR. Therefore, to navigate around such

sampling problems, you should implement your system frequency plan for the DDC passband filter width and NCO tuning position.

## ARE EXTERNAL FILTERS REQUIRED?

System ADCs using internal DDCs can use additional analog filters, as would normally be done without DDC filtering. For wideband applications, the DDCs provide some relaxation in the filtering that's needed at the front end of the ADC.

The digital filtering within the DDC will do some of the work and relax what otherwise would require a strict front-end anti-alias analog filter. However, a wideband front end will now allow multiple uses for the DDC to either observe multiple bands simultaneously or even sweep the band of interest with the NCO to find a changing input signal.

## CAN AN ADC PROVIDE MULTIPLE DDCS?

The final question for engineers contemplating internal digital downconversion with an FPGA is whether an ADC provides just one DDC. The answer is no; in fact, multiple bands can be observed.

For multiple DDCs within an ADC, each can have its own NCO that tunes to separate bands across the Nyquist zone. This scheme makes it possible to observe multiple frequency bands simultaneously and removes the burden on the system FPGA transceiver count and decimation blocks, which can be reassigned to other processing activities such as channelizing multiple ADCs for FDM systems.

High-speed ADCs now have the processing power to bring the DDC function up the signal chain. For those systems that do not need to use the full bandwidth of a wideband Nyquist-rate ADC, the DDC operation filters the unwanted data and noise. This can improve the SNR and SFDR of the signal acquisition. The lower bandwidth reduces the data interface burden to the transceivers of an FPGA, like Artix-7, and allows for the design of more-complex signal-acquisition systems.

# What's New in the Vivado 2015.1 Release?

*Xilinx is continually improving its products, IP and design tools as it strives to help designers work more effectively. Here, we report on the most current updates to Xilinx design tools including the Vivado® Design Suite, a revolutionary system- and IP-centric design environment built from the ground up to accelerate the design of Xilinx® All Programmable devices. For more information about the Vivado Design Suite, please visit* www.xilinx.com/vivado.

*Product updates offer significant enhancements and new features to the Xilinx design tools. Keeping your installation up to date is an easy way to ensure the best results for your design.*

*The Vivado Design Suite 2015.1 is available from the Xilinx Download Center at* www.xilinx.com/download.

## VIVADO DESIGN SUITE 2015.1 RELEASE HIGHLIGHTS

The latest release of the Vivado Design Suite includes the new Vivado Lab Edition; interactive clock domain crossing (CDC) analysis; accelerated simulation flows; advanced system performance analysis in the Xilinx Software Development Kit (SDK); and new devices including the XCVU440.

### Vivado Lab Edition

The new Vivado Lab Edition is a no-cost, lightweight programming and debug edition of the Vivado Design Suite. It includes the Vivado Device Programmer, Vivado Logic and Serial I/O Analyzer, as well as memory debug tools. Lab Edition is intended for use in laboratory environments where the full-featured Vivado Design Suite is not required. As such, it is 75 percent smaller than the complete Vivado Design Edition, which considerably reduces lab setup time and system memory requirements. For design teams that require remote debug or programming over Ethernet, the Vivado Design Suite 2015.1 also provides a standalone hardware server that occupies less than 1 percent of the complete Vivado Design Edition.

### Interactive Clock Domain Crossing Analysis

The interactive CDC capability enables debug of CDC issues earlier in the design, reducing expensive in-system debug cycles. Combined with interactive timing analysis and cross-probing features, the Vivado Design Suite provides unparalleled timing analysis and debug functionalities, accelerating time-to-market.

### Vivado Simulator and Third-Party Flows

Advancements in the simulation flows reduce the LogiCORE™ IP compile times by more than half. Overall simulation performance is 20 percent faster than in previous releases. Simulation flows are fully integrated with those of Xilinx Alliance Program members Aldec, Cadence Design Systems, Mentor Graphics and Synopsys.

### Xilinx SDK Advanced System Performance Analysis

Xilinx has extended the SDK with the ability to analyze the performance and the bandwidth of a Zynq®-7000 All Programmable SoC design, including key performance metrics for the processing system (PS) as well as bandwidth analysis between the PS, the programmable logic (PL) and external memories. System-modeling designs using AXI traffic generators are provided for the ZC702 and ZC706 evaluation boards.

### Device Support

**New Devices**

The following UltraScale™ devices are introduced in this release:

- Virtex® UltraScale devices: XCVU125, XCVU190, XCVU440

**General Access**

- Kintex® UltraScale devices: XCKU035, XCKU060, XCKU115
- Virtex UltraScale devices: XCVU065

**Early Access (Contact your local Xilinx sales representative)**

- Virtex UltraScale devices: XCVU160

### Licensing

Vivado Design Suite 2015.1 features updates to licensing. With Vivado License Borrow, clients may borrow one floating seat and lock to machine for non-network use of the Vivado tools for a specific period (for activation licenses only). Also included is virtual-machine support for activation-based licenses. This release introduces one-step activation licensing. If you are using Vivado License Manager for client (node-locked) licenses, and are connected to the Internet, Vivado License Manager downloads and installs activation licenses automatically.

## VIVADO DESIGN SUITE: DESIGN EDITION UPDATES

### Partial Reconfiguration and Tandem Configuration

The Partial Reconfiguration Controller IP is now available for any user of partial reconfiguration (PR) in 7 series, Zynq SoC or UltraScale devices. This IP is the heart of a PR system, fetching from memory and delivering to the configuration port partial bitstreams when hardware or software trigger events occur. The latest release has expanded support for UltraScale devices, and supports implementation for KU115, VU125 and VU190 devices, as well as the previously supported KU040, KU060 and VU095 devices. Tandem PROM and Tandem PCIe® are available for the same UltraScale devices that now have PR support (KU115, VU125 and VU190).

For more information, see PG193.

### Vivado IP Integrator

Release updates include a bottom-up synthesis flow option for faster design iterations. Each piece of IP is synthesized by itself and only changed if the IP needs to be synthesized again. A new layout optimized for IP Integrator is available in the Vivado IDE. The result is up to a 50 percent reduction in project flow time including design generation and validation, along with improvements to revision control ease of use. The release also offers support for saving a design in the validated state so that validation does not need to be rerun during generation. The search in the "Add IP…" window has been enhanced and there is now quick access to IP details.

See the Vivado Design Suite 2015.1 Release Notes for more information.

## VIVADO DESIGN SUITE: SYSTEM EDITION UPDATES

### Vivado High-Level Synthesis

The System Edition boasts new synthesizable C++ library functions with a special focus on software-defined radio applications: numerically controlled oscillator (NCO), QAM modulator and demodulator. See UG902 for more information.

### System Generator for DSP

Release updates include advanced support for hardware co-simulation burst mode, which accelerates simulation to increase performance by 100x. Improved timing analysis allows cross-probing to quickly identify failing paths. A new capability parses an SoC platform design from Vivado IP Integrator to tailor a complementary set of gateways for easy and accurate IP development. Enhanced support for multiple AXI4-Lite interfaces enables independent register alignment to clock domains. Finally, the tool now supports MATLAB® 2015A.

See the Vivado Design Suite 2015.1 Release Notes for more information.

## XILINX INTELLECTUAL PROPERTY (IP) UPDATES

Xilinx has launched its next generation of video-over-Internet Protocol connectivity and professional video solutions, enabling "any media over any network," for the broadcast and pro A/V markets.

### Video-over-Internet Protocol Connectivity

Xilinx defines and deploys video over Internet protocols for contribution and distribution networks with the provision of the SMPTE ST 2022-1,2,7 and SMPTE ST 2022-5,6,7 cores and reference designs. ST 2022-1,2,7 and ST 2022-5,6,7 cores are now available from Xilinx. To learn more, visit *http://www.xilinx.com/products/intellectual-property/ef-di-smpte2022-12.html* and *http://www.xilinx.com/products/intellectual-property/ef-di-smpte2022-56.html*.

The video-over-IP FEC Engine l is available in the Vivado Design Suite 2015.1 release. Xilinx provides free system-level reference designs for all cores with application notes at *http://www.xilinx.com/esp/broadcast/refdes_listing.htm*.

### New Suite of 'Any to Any' Video Connectivity Solutions

The release includes HDMI 1.4/2.0; DisplayPort 1.2 with support for HDCP; and the new 6G/12G versions of SDI. These Xilinx-developed and supported interfaces will allow developers to adhere to the latest industry standards for 4K/Ultra-HD systems.

The video connectivity solutions are available from Xilinx in the Vivado Design Suite 2015.1 release and will be supported on Artix-7, Kintex-7, Virtex-7, Virtex-7X and Kintex UltraScale FPGAs, and on Zynq-7000 All Programmable SoCs. The video-processing suite is now available from Omnitek and targets Kintex-7 and Kintex UltraScale FPGAs and Zynq SoCs. To learn more, visit *http://omnitek.tv/sites/default/files/OSVP.pdf*.

## LEARN MORE

### QuickTake Video Tutorials

Vivado Design Suite QuickTake video tutorials are how-to videos that take a look inside the features of the Vivado Design Suite and UltraFast™ Design Methodology. New topics include:

- What's New in Vivado 2015.1
- Simulating with Cadence IES in Vivado
- Designing with UltraScale Memory IP
- Using Board Automation with IP Integrator

See all Quick Take Videos at *www.xilinx.com/training/vivado*.

### Training

For instructor-led training on the Vivado Design Suite, UltraFast Design Methodology and more, visit *www.xilinx.com/training*.

**Download Vivado Design Suite 2015.1 today at** *http://www.xilinx.com/download*.

# Latest and Greatest from the Xilinx Alliance Program Partners

*Xpedite highlights the latest technology updates from the Xilinx Alliance Program 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 some highlights.***

## DAVE'S BORA NOW SUPPORTS SDSOC DEVELOPMENT ENVIRONMENT

DAVE Embedded Systems (Porcia, Italy) has collaborated with Xilinx to make sure its BORA module supports Xilinx's SDSoC development environment. The BORA module is built around Xilinx's Zynq®-7000 All Programmable SoC. The SDSoC support will allow BORA users to develop their software algorithms quickly and easily. Hardware-accelerated functions within BORA are implemented in programmable logic but can be invoked transparently from software applications running on the Zynq SoC's dual-core ARM® Cortex™-A9 processing system. Accelerated functions—written in C, C++ or SystemC—can be moved from the software domain to the FPGA fabric on top of an existing implementation.

DAVE demonstrated the BORA system in February at Embedded World 2015 in Nuremberg, Germany. The demonstration consisted of IP (namely, an LCD controller) that was developed with classic tools and IP generated by the SDSoC design environment.

For more information, please visit *http://www.dave.eu*.

## XYLON USES SDSOC FOR MICROZED-BASED VISION PLATFORM

Xylon (Zagreb, Croatia) develops logicBRICKS IP cores that help customers stay at the forefront of technology innovations in image processing and computer vision. To provide reusable IP that smoothly integrates into Xilinx All Programmable SoCs and MPSoCs and implements evolving video-processing, object-detections and video-analytics algorithms, Xylon's designers have become experts with the latest Xilinx design tools and technologies. Xylon has been working closely with Xilinx as Xilinx developed its innovative SDSoC development environment as an alternative to manual RTL coding.

Using the new development environment for only a couple of weeks, Xylon was able to develop a board support package (BSP) for a MicroZed board-based vision platform. And by using the legacy logicBRICKS IP as the C-callable RTL IP, Xylon designed a re-

al-time facial-features tracking system that the company demonstrated at Embedded World 2015.

For more information, please visit *http://www.logicbricks.com/*.

## IVEIA PUTS SDSOC THROUGH ITS PACES ON EDGE-DETECTION ALGORITHM

Leveraging Xilinx's SDSoC development environment, iVeia (Annapolis, Md.) demonstrated at Embedded World one of its pure C/C++ reference designs for processing high-definition video. Using iVeia's Atlas-I-Z7e system-on-a-module and video development kit, the company ran a canny edge-detection algorithm in real time on a GigE Vision high-definition camera.

Two implementations of the identical C/C++ code were demonstrated running side-by-side, one compiled using the standard C/C++ compiler, the other using the SDSoC development environment's full-system optimizing compiler. The performance of the SDSoC development environment's implementation was nearly two orders of magnitude faster than the standard implementation. Using the SDSoC environment to profile and partition the design, functions identified as repetitive and compute-intensive were targeted for the programmable logic while the more-complex statistical algorithms remained on the processing system.

For more information, please visit *http://www.iveia.com/*.

## ADI ADOPTS SDSOC TO CREATE RADIO REFERENCE DESIGNS

Three of the main challenges of any SoC-based software-defined radio (SDR) platform are the partitioning of the design between software and HDL, the implementation of the data-processing algorithms in HDL code and the integration of this HDL code in the main system design. The Xilinx SDSoC development environment addresses these challenges by providing an integrated development environment within which users can implement their entire design in C/C++ code, select which parts of the design they would like to have implemented in programmable logic and then let the tool generate the software and HDL portions of the design and bind everything together to create the final system design.

Analog Devices (Norwood, Mass.) has started to adopt the SDSoC development environment to create reference designs for its SDR FMCOMS2/3/4/5 platforms based on the AD9361/AD9364 agile RF transceivers. The first reference design to be released shows how to implement a direct-digital-synthesis (DDS) IP core in the SDSoC development environment and integrate the generated IP into the Analog Devices FMCOMMS HDL platform and Linux environment. In this way, the output of the DDS can be transmitted and received back over the air by the FMCOMMS card and displayed in the Analog Devices IIO Scope Linux application.

The DDS IP is implemented entirely in C code following the Xilinx high-level synthesis (HLS). The DDS function is called in a C main loop to generate the bindings for the data flow between the IP and the rest of the design. Based on this code, the SDSoC development environment is able to synthesize the DDS IP, integrate it into the Analog Devices HDL platform and generate all the necessary files in order to be able to run the Analog Devices Linux distribution on the Zynq-7000 All Programmable SoC. Analog Devices demonstrated the DDS HLS IP with the Zynq SoC SDR kit at Embedded World.

## DORNERWORKS TIPS XEN HYPERVISOR FOR ZYNQ ULTRASCALE+ MPSOC DESIGNS

Xilinx has selected DornerWorks to provide an open-source Xen hypervisor solution to its customers along with comprehensive customer support and engineering services. The Xen hypervisor is a well-established virtual-machine monitor for running multiple operating systems simultaneously, making Xen an obvious choice for upcoming products based on the Zynq UltraScale+™ MPSoC targeting next-generation cloud, data center, and wireless and wired network computing. Xen provides the virtualization of applications and of guest operating systems running on the Zynq MPSoC's quad-core ARM Cortex-A53 processor.

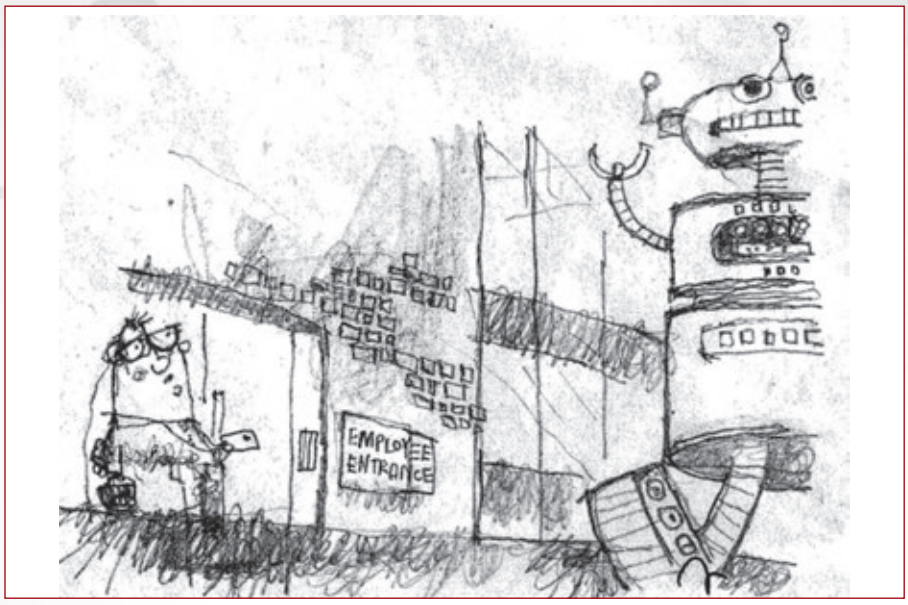
"DornerWorks brings proven, solid technology expertise on Xen hypervisor solutions and has strong service capabilities for both embedded processing designs and FPGA logic designs," said Hugh Durdan, vice president of portfolio and solutions marketing at Xilinx. "Our partnership with DornerWorks on the Xen hypervisor solution for the Zynq UltraScale+ MPSoC will enable our customers to adopt a virtualized, secure OS platform quickly and accelerate product development cycles."

Development with a hypervisor not only requires the Xen kernel, but also a configured dom0 (the privileged system domain) and configured guest domains with their own guest operating systems. DornerWorks will provide sample distributions of complete packages and documentation on how to spin your own customized systems.

To complete the hypervisor ecosystem, DornerWorks also offers a range of services to its customers, including FPGA design services.

For details, visit *http://dornerworks.com/services/XilinxXen*.

# Xpress Yourself in Our Caption Contest



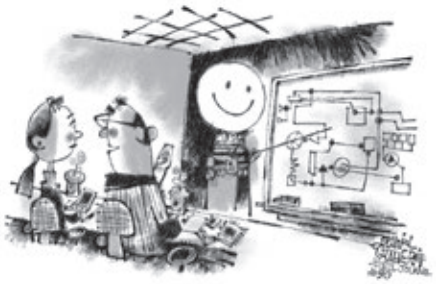

DANIEL GUIDERA

Even robots have their limits. The mechanical man clanking his way across this factory floor looks like he's had enough—driven berserk, perhaps, by faulty programming. How do you interpret his plight? Let us know by writing an engineering- or technology-related caption for our cartoon featuring this modern-day tin man. The image might inspire a caption like "George watched the new janitorial robot walk off the job in a huff, but couldn't fix the Roomba AI program fast enough to stop him."

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 a Digilent Zynq Zybo board, featuring the Xilinx® Zynq®-7000 All Programmable SoC (*http://www.xilinx.com/products/boards-and-kits/1-4AZFTE.htm*). Two runners-up will gain notoriety, fame and have their captions, names and affiliations featured in the next issue.

The contest begins at 12:01 a.m. Pacific Time on April 17, 2015. All entries must be received by the sponsor by 5 p.m. PT on July 1, 2015.

**BARRY MEAKER,** design engineer at Boeing Company (Seattle), won a shiny new Digilent Zynq Zybo board with this caption for the smiley-face cartoon in Issue 90 of *Xcell Journal*:



"Ever since Bob announced he's retiring in a month, he seems different."

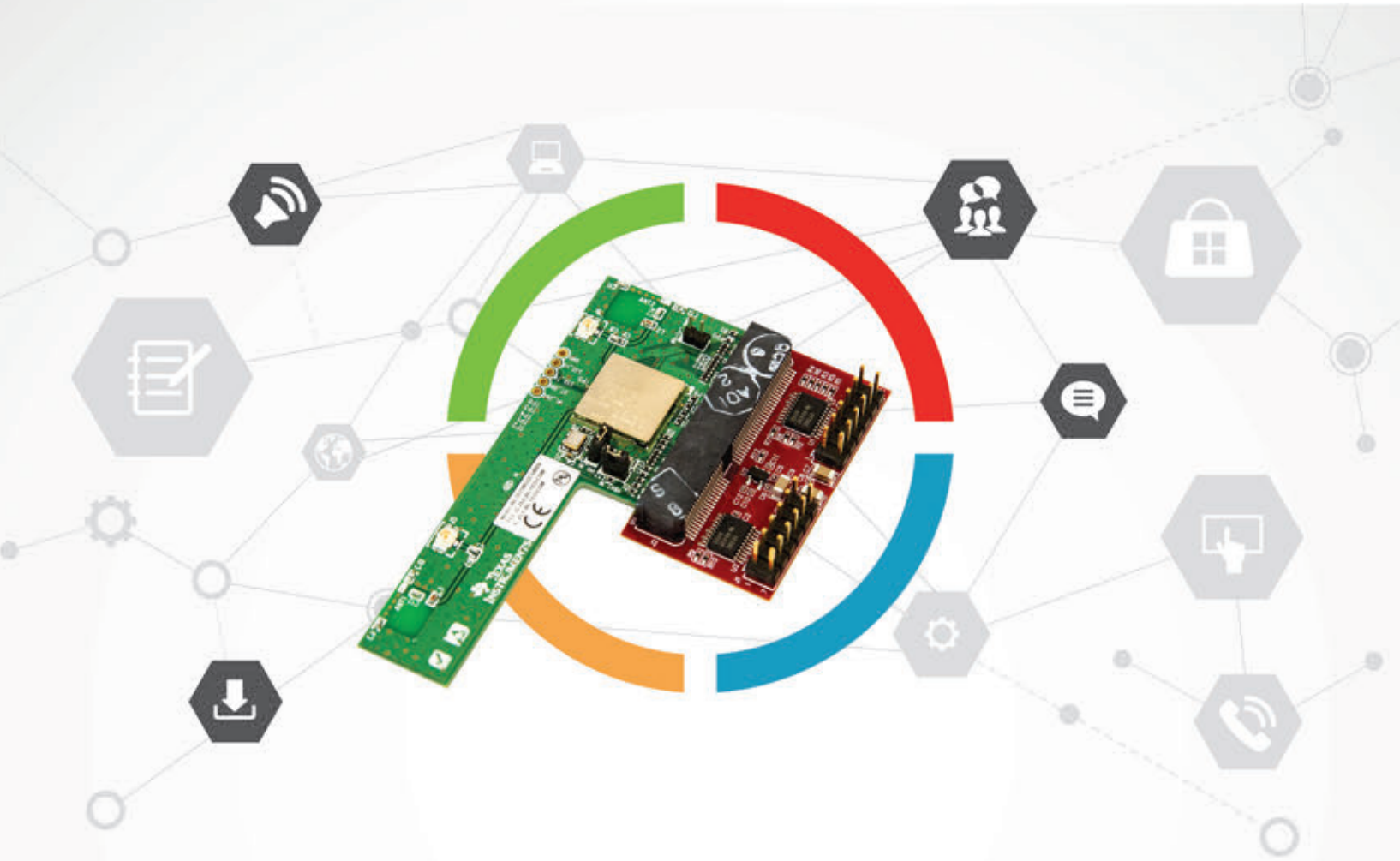
**Congratulations as well to our two runners-up:**

"It's a Logic High."

*— Aaron Algiere, electrical engineer, Harris Corp., Melbourne, Fla.*

"It's obviously unstable. Too much positive feedback."

*— Glenn Dixon, engineer, L-3 Communications, Salt Lake City*



# Need Wireless Connectivity in your Next SoC Design?

Adding wireless to your next Zynq®-7000 All Programmable SoC design just got easier with Avnet's new WiLink® 8 Adaptor and example Linux reference design for MicroZed™.

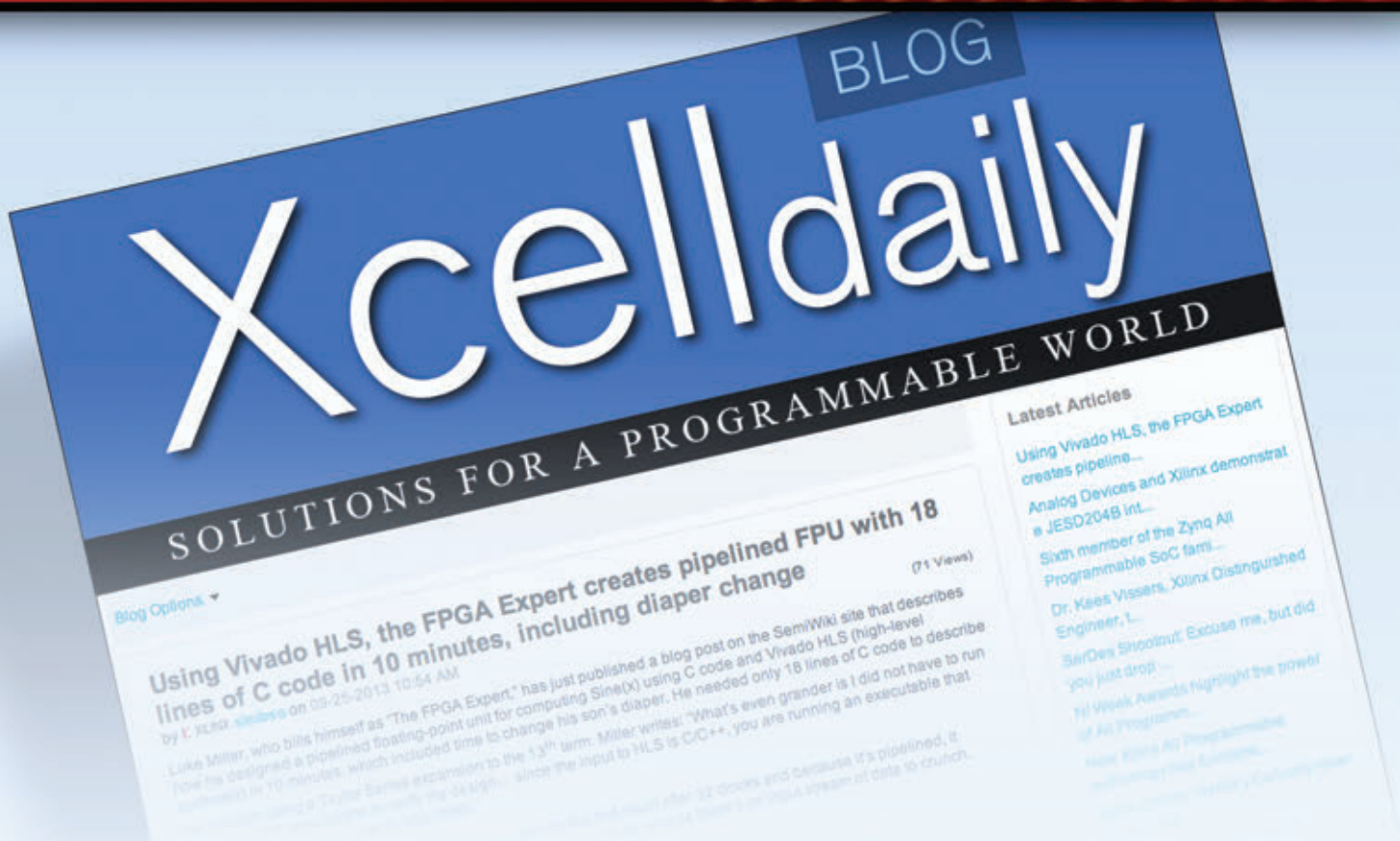
---

**Order your kit today at:**
www.zedboard.org/product/wilink-8-adaptor



Accelerating Your Success™

# *Xcell Journal* Adds New Daily Blog




Xilinx has extended the Award Winning Journal and added an exciting new *Xcell Daily Blog*. The new site provides dedicated readers with a frequent flow of content to help engineers leverage the flexibility and extensive capabilities of Xilinx products, ecosystem, and customers to create All Programmable and Smarter Systems.

## Recent

- *Adam Taylor's MicroZed Chronicles Part 77 – Introducing the Zynq SoC's Ethernet*
- *NGCodec demos HEVC H.265 Real-Time Encoder running on Kintex-7 FPGA at NAB 2015*
- *New 27-page White Paper covers the basics of design for functional safety using FPGAs and the Zynq SoC*
- *Zynq-based ONetSwitch open-source SDN Kickstarter project completes successful funding campaign*
- *Sumitomo Electric's 4x25G QSFP28 LR4 optical module and Virtex UltraScale FPGA drive 10km of fiber at OFC 2015*

Visit Blog: www.forums.xilinx.com/t5/Xcell-Daily/bg-p/Xcell

XILINX
ALL PROGRAMMABLE.