

Xcell journal

ISSUE 89, FOURTH QUARTER 2014

SOLUTIONS FOR A PROGRAMMABLE WORLD

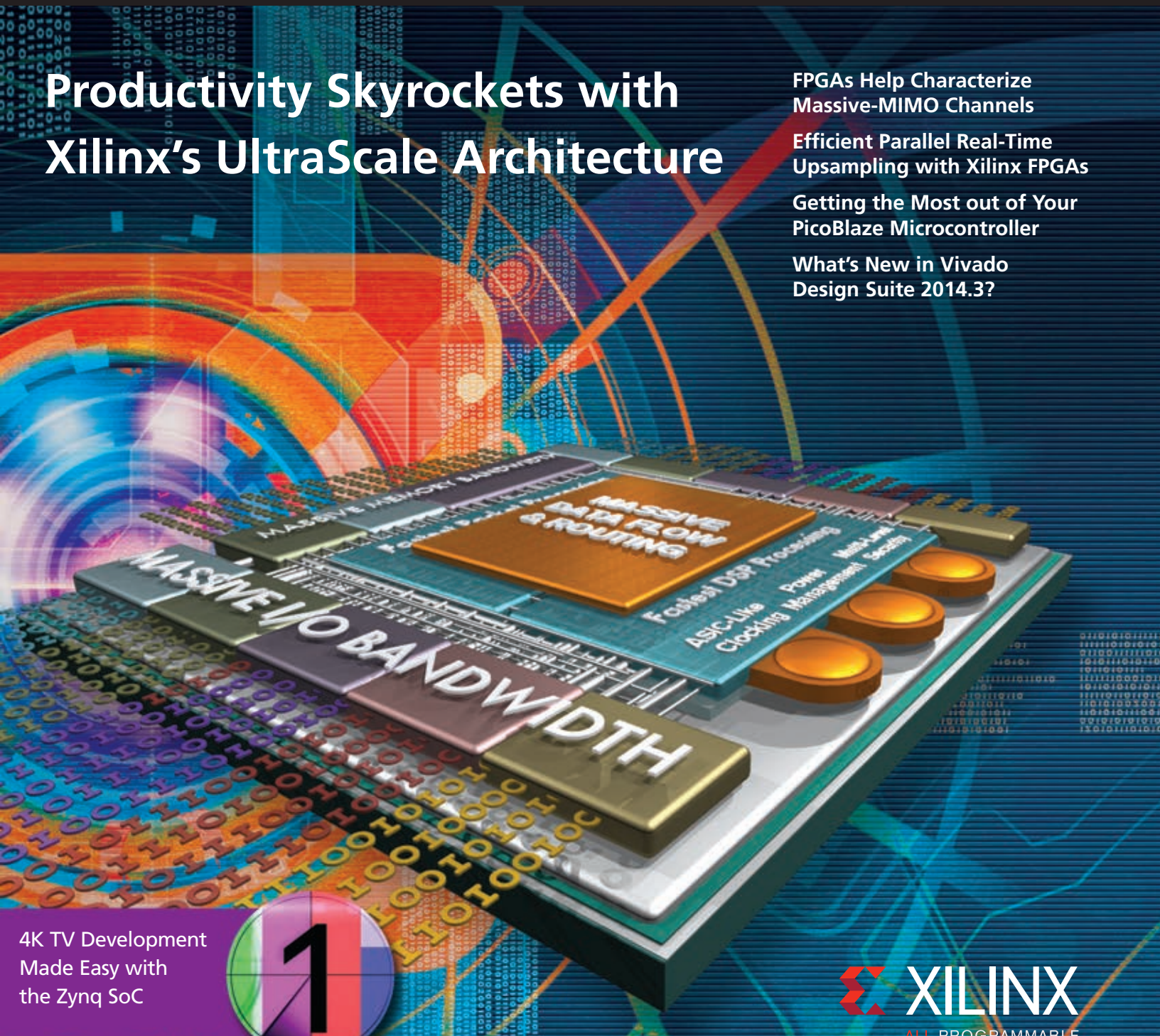
Productivity Skyrockets with Xilinx's UltraScale Architecture

FPGAs Help Characterize Massive-MIMO Channels

Efficient Parallel Real-Time Upsampling with Xilinx FPGAs

Getting the Most out of Your PicoBlaze Microcontroller

What's New in Vivado Design Suite 2014.3?

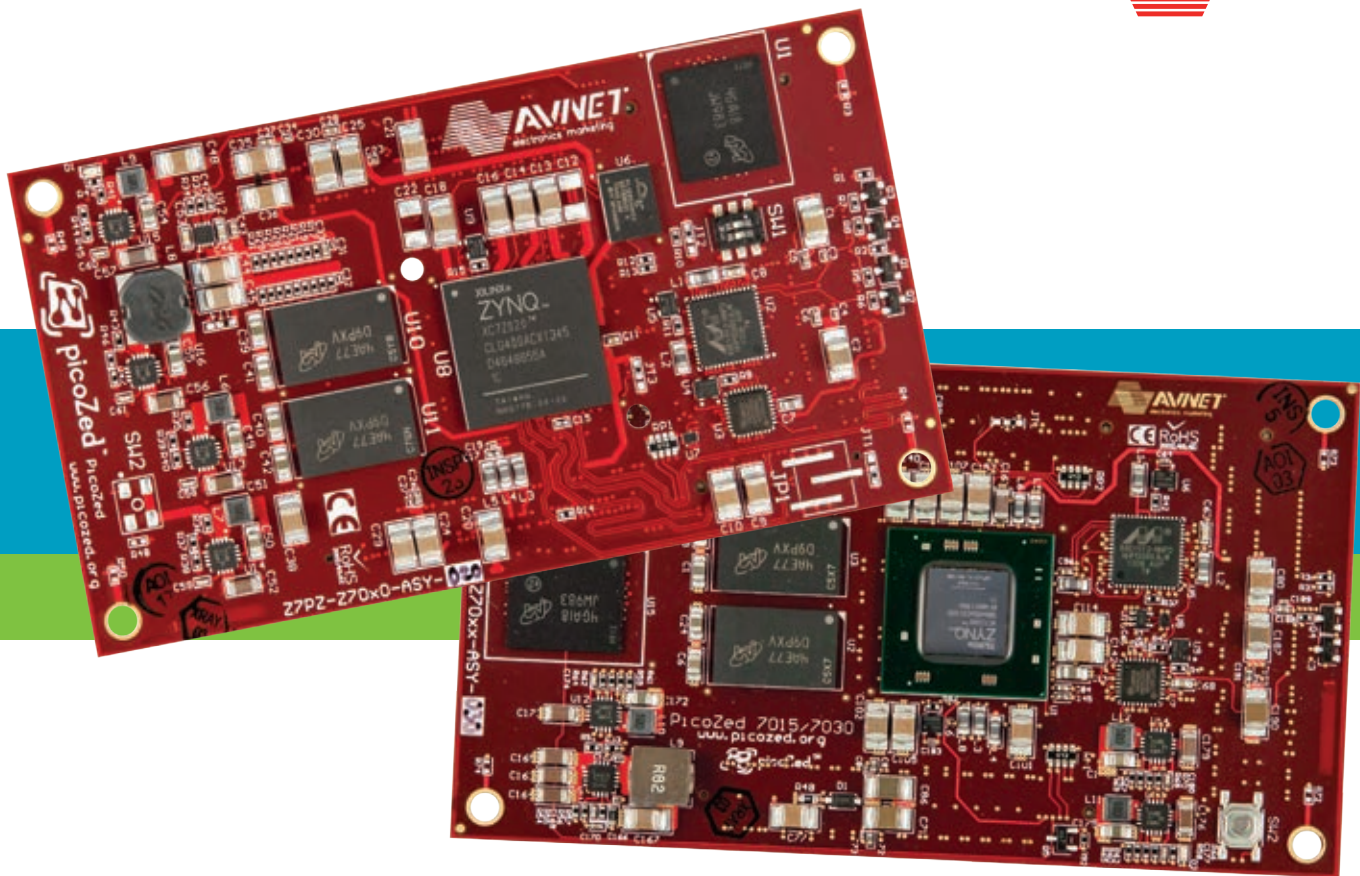


4K TV Development Made Easy with the Zynq SoC



 **XILINX**
ALL PROGRAMMABLE™

www.xilinx.com/xcell



PicoZed™ System-On-Module

Cost Effective, Scalable, Rugged, and Easy to Use



DESIGNED BY AVNET

PicoZed™ is a highly flexible, rugged, System-On-Module, or SOM that is based on the Xilinx Zynq®-7000 All Programmable SoC. It offers designers the flexibility to migrate between the 7010, 7015, 7020, and 7030 Zynq-7000 All Programmable SoC devices in a pin-compatible footprint. The PicoZed module contains the common functions required to support the core of most SoC designs, including memory, configuration, Ethernet, USB, clocks, and power. It provides easy access to over 100 user I/O pins through three I/O connectors on the backside of the module. These connectors also support access to dedicated interfaces for Ethernet, USB, JTAG, power and other control signals, as well as the GTP/GTX transceivers on the 7015/7030 models. The transceiver based 7015 and 7030 versions of PicoZed are a superset of the 7010/7020 version, adding four high-speed serial transceiver ports to the I/O connectors. Designers can simply design their own carrier card, plug-in PicoZed, and start their application development with a proven Zynq-7000 All Programmable SoC sub-system. Industrial Temperature PicoZed SOMs are built with components supporting extended temperatures of -40 to +85°C for harsh environments. www.picozed.org

Announcing HAPS Developer eXpress Solution

Pre-integrated hardware and software for fast prototyping of complex IP systems

- ✓ 500K-144M ASIC gates
- ✓ Ideal for IP and Subsystem Validation
- ✓ Design Implementation and Debug Software Included
- ✓ Flexible Interfaces for FMC and HapsTrak
- ✓ Plug-and-play with HAPS-70

Designs come in all sizes. Choose a prototyping system that does too. HAPS-DX, an extension of Synopsys' HAPS-70 FPGA-based prototyping product line, speeds prototype bring-up and streamlines the integration of IP blocks into an SoC prototype.

To learn more about Synopsys FPGA-based prototyping systems, visit www.synopsys.com/haps

Xcell journal

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

EDITOR Jacqueline Damian

ART DIRECTOR Scott Blair

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

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

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

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

Tomoko Suto, Japan
tomoko@xilinx.com

REPRINT ORDERS 1-800-493-5551



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/

© 2014 Xilinx, Inc. All rights reserved. XILINX, the Xilinx Logo, and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

The articles, information, and other materials included in this issue are provided solely for the convenience of our readers. Xilinx makes no warranties, express, implied, statutory, or otherwise, and accepts no liability with respect to any such articles, information, or other materials or their use, and any use thereof is solely at the risk of the user. Any person or entity using such information in any way releases and waives any claim it might have against Xilinx for any loss, damage, or expense caused thereby.

Embedded Methodology Manual Helps Teams Deliver Zynq SoC Designs UltraFast

I have great news for those of you targeting the Zynq®-7000 All Programmable SoC for your next design project: Xilinx has just published a must-read methodology manual called the [“UltraFast™ Embedded Design Methodology Guide” \(UG4046\)](#) and made it available as a free download.

The 211-page manual is a companion document to the [“UltraFast Design Methodology for the Vivado Design Suite” \(UG949\)](#), which Xilinx published in 2013 and we featured as the cover story for *Xcell Journal* issue 85.

Tom Feist, senior director of methodology marketing at Xilinx, said that where the “UltraFast Design Methodology for the Vivado Design Suite” focuses on helping all Vivado users employ best practices with the Vivado tools, the new “UltraFast Embedded Design Methodology Guide” is geared to helping entire design teams get the most out of Xilinx’s rich embedded tool box. With guidance from the manual, they can optimize their design practices and get their Zynq SoC-based designs to market faster.

“The ‘UltraFast Design Methodology for the Vivado Design Suite’ has been a great success. It has already proven to accelerate development from months to weeks with our users and has helped many design teams tap into the ASIC-design-class features of Vivado,” said Feist. “The ‘UltraFast Embedded Design Methodology Guide’ will further help embedded-design teams using our Zynq SoCs.”

Embedded-design teams typically have a mix of skill sets and team members, including system architects, hardware engineers and software engineers. “Each type of engineer has a certain way of doing things and the particular software or tools they use for designing,” Feist said. “And of course, the skill levels will vary too. This methodology manual has something for everyone.”

Feist said the manual will help designers better organize their Zynq SoC projects as a team and sharpen their individual Zynq SoC design skills. “It presents each member of the embedded-design team with the key principles, specific do’s and don’ts, and best practices they can employ to optimize their design processes,” said Feist.

The content is based on the best practices from Xilinx experts and design teams that have already delivered several Zynq SoC-based products to market. The manual is split into sections for each engineering discipline, but Feist said that all team members can glean valuable information and better organize their projects by reading the manual in its entirety.

The manual is composed of six chapters: System-Level Considerations, Hardware Design Considerations, Software Design Considerations, Hardware Design Flow, Software Design Flow and Debug. What’s more, the manual goes into the handoffs and relationships among the flows.

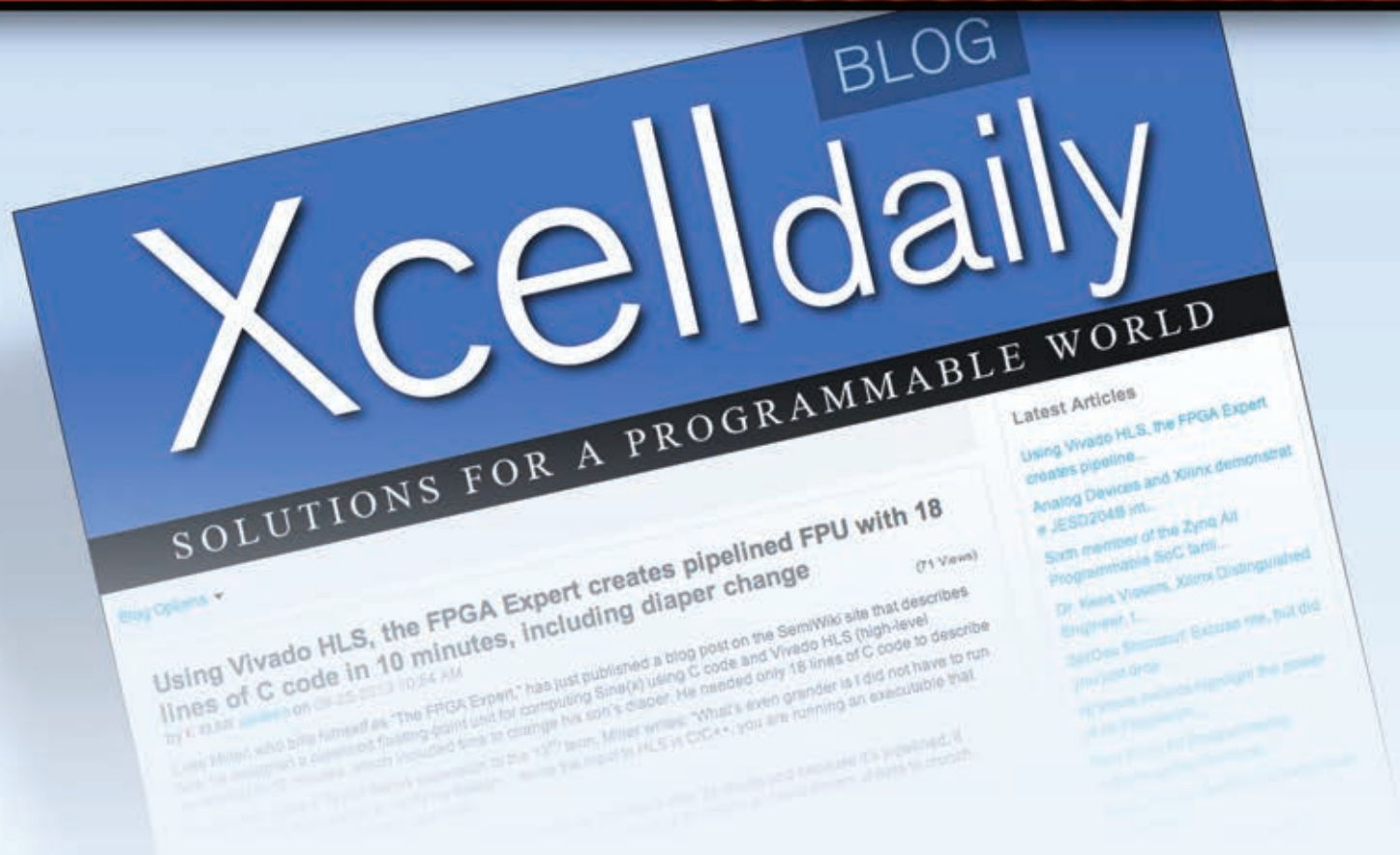
I hope you find the “UltraFast Embedded Design Methodology Guide” an enjoyable and informative read. Meanwhile, I invite you to also peruse the many great articles in this issue of *Xcell Journal*. Happy reading!



Mike Santarini
Publisher



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

- [Memory choices abound and new Xilinx White Paper clarifies your options](#)
- [Smart Vision Dev Kit pairs Zynq-based PicoZed SOM with Aptina-based 1.2Mpixel camera module](#)
- [miniSpartan6+ Kickstarter FPGA dev board update + video of HDMI I/O and processing](#)
- [New Zynq-based SDR Reference Design supports Public Safety and Military LTE Radio Development](#)

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

VIEWPOINTS

Letter from the Publisher

Embedded Methodology Manual
Helps Teams Deliver Zynq SoC
Designs UltraFast... **4**



XCELLENCE BY DESIGN APPLICATION FEATURES

Xcellence in UltraScale

UltraScale Brings Interlaken Onboard
for High-Bandwidth Designs... **14**

Xcellence in Wireless Communications

FPGAs Help Characterize
Massive-MIMO Channels... **18**

Xcellence in Video Processing

4K TV Development Made Easy
with the Zynq SoC... **26**

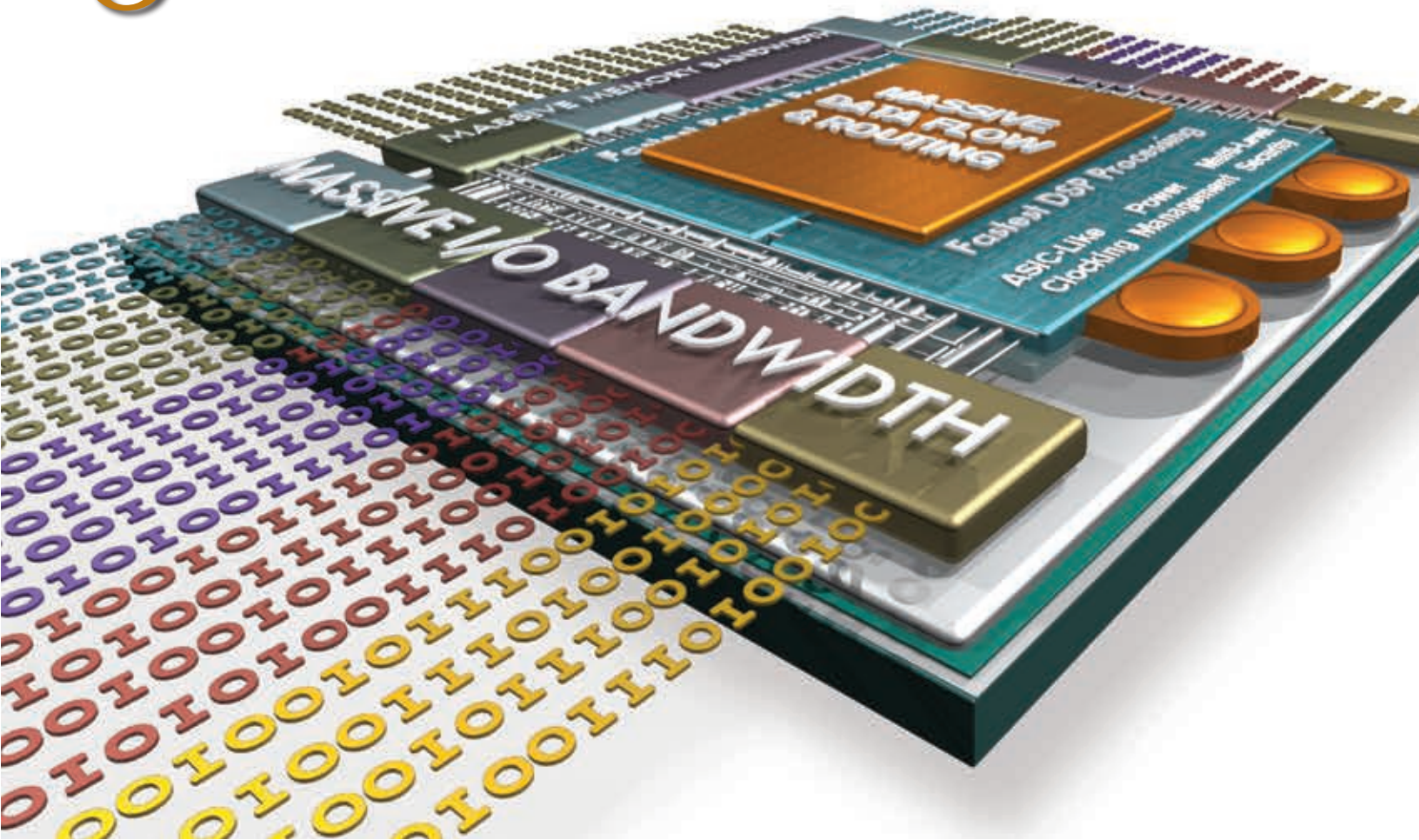
Xcellence in Astrophysics

FPGAs Aid Search for Dark Energy
with CHIME Telescope... **32**



Cover Story

8 Productivity Skyrockets with
Xilinx's UltraScale Architecture



THE XILINX XPERIENCE FEATURES

Xplanation: FPGA 101

Efficient Parallel Real-Time Upsampling with Xilinx FPGAs... **38**

Xplanation: FPGA 101

A Framework for FPGA Design Planning... **46**

Xplanation: FPGA 101

Faster Design Entry with Vivado IP Integrator and Xilinx IP... **50**

Xplanation: FPGA 101

Getting the Most out of Your PicoBlaze Microcontroller... **54**

Xperts Corner

Changing Utilization Rates in Real Time to Investigate FPGA Power Behavior... **60**



46



60

XTRA READING

Xtra, Xtra The latest Xilinx tool updates and patches, as of October 2014... **66**

Xpedite Latest and greatest from the Xilinx Alliance Program partners... **68**

Xclamations! Share your wit and wisdom by supplying a caption for our wild and wacky artwork... **70**



Productivity Skyrockets with Xilinx's UltraScale Architecture

by **Nick Mehta**

Senior Technical Marketing Manager
Xilinx, Inc.
nick.mehta@xilinx.com

SCANNING

The enhanced capabilities of the Xilinx UltraScale architecture combine with time-saving tools in the Vivado Design Suite to help you build superior systems faster.

M

Many markets and applications require a tremendous increase in system bandwidth and processing capability. Whether in wired or wireless communications, digital video or image processing, increased data throughput requirements have the same result: increasing traffic and demands on all system components. More data arrives on-chip through parallel and serial I/O. The data must then be buffered, again through both parallel I/O in the form of DDR memory and serial I/O in the form of serial memory standards such as Hybrid Memory Cube (HMC) and MoSys Bandwidth Engine. The data must then be processed in the logic and DSP before being transmitted to its next destination back through the parallel and serial I/O.

System processing requirements are becoming more complex for a number of reasons; larger data packets traveling at an increased data rate result in wider parallel data buses at increased frequency. To efficiently process the data, it is often necessary to build an entire system in a single device, thereby eliminating the latency and power consumption associated with sending large quantities of data between two FPGAs. The result is a need for ever-denser FPGAs with more capabilities. It is imperative that as these high-capability FPGAs are more heavily utilized, they maintain the ability to operate at their maximum possible performance, avoiding performance degradation as the device fills up.

Filling complex, high-capacity devices to high utilization could sound like a daunting task for the designer. Xilinx provides numerous solutions specifically aimed at reducing design time, freeing designers to

Data is transported to and from UltraScale FPGAs through a combination of high-performance parallel SelectIO and high-speed serial transceiver connectivity.

focus their efforts on what differentiates their product in their market.

THE ULTRASCALE ARCHITECTURE

To address the prevailing market challenges, Xilinx recently introduced the UltraScale™ architecture (Figure 1), providing unprecedented levels of system integration, performance and capability. Xilinx has used this new architecture to create two high-per-

formance FPGA families. The Xilinx® Virtex® UltraScale and Kintex® UltraScale families combine to address a vast spectrum of system requirements with a focus on lowering total power consumption through numerous innovative technological advancements. Sharing numerous building blocks, the UltraScale technology provides a scalable architecture, optimized for different market demands.

INCREASING SYSTEM BANDWIDTH

Before any signal processing or data manipulation can occur, data needs to reach its destination. Numerous serial and parallel protocols and standards exist today tailored to the specific needs of their target applications. The common theme among most standards is the desired increase in total data throughput, enabling vast quantities of information to move through a system at increasing data rates.

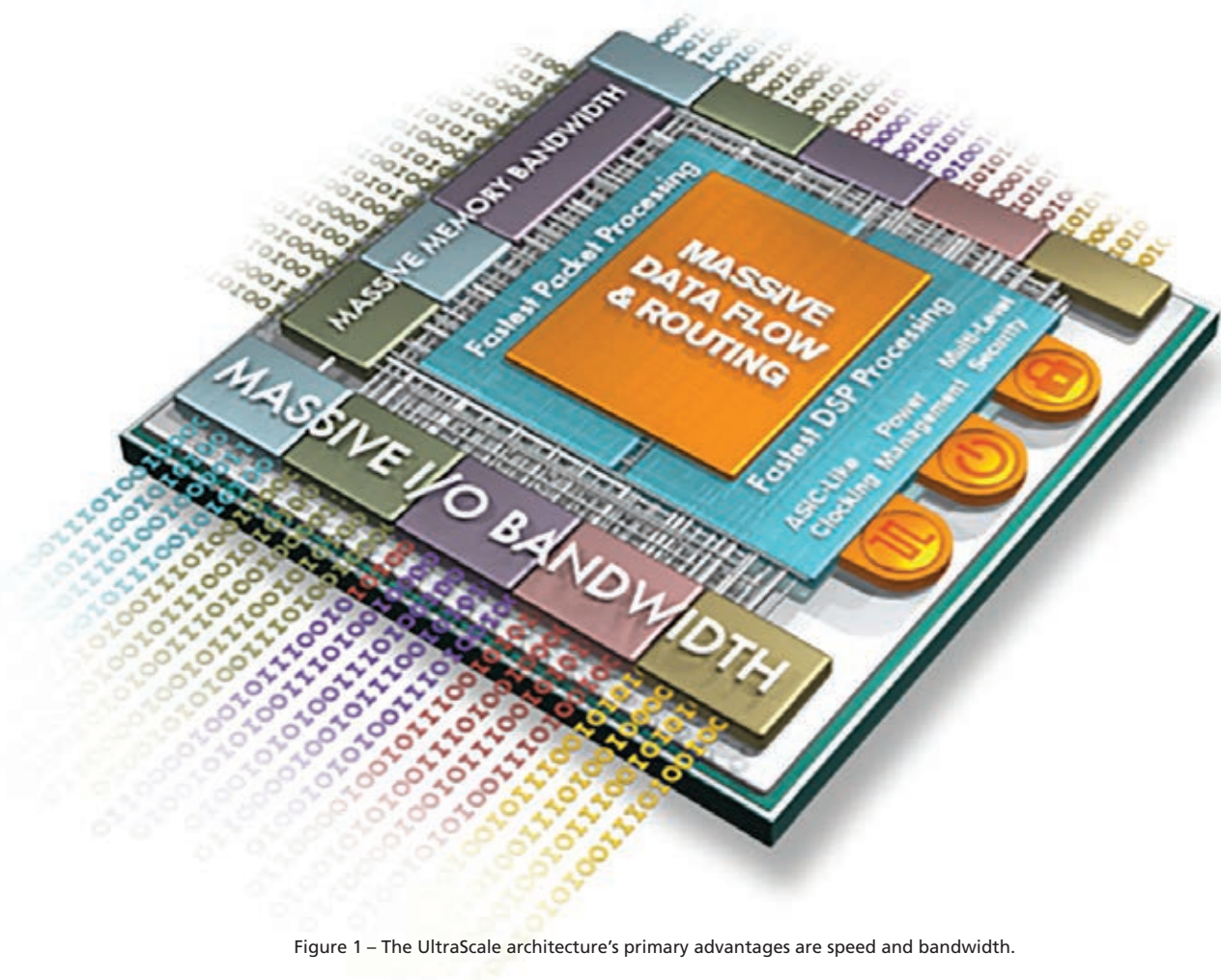


Figure 1 – The UltraScale architecture's primary advantages are speed and bandwidth.

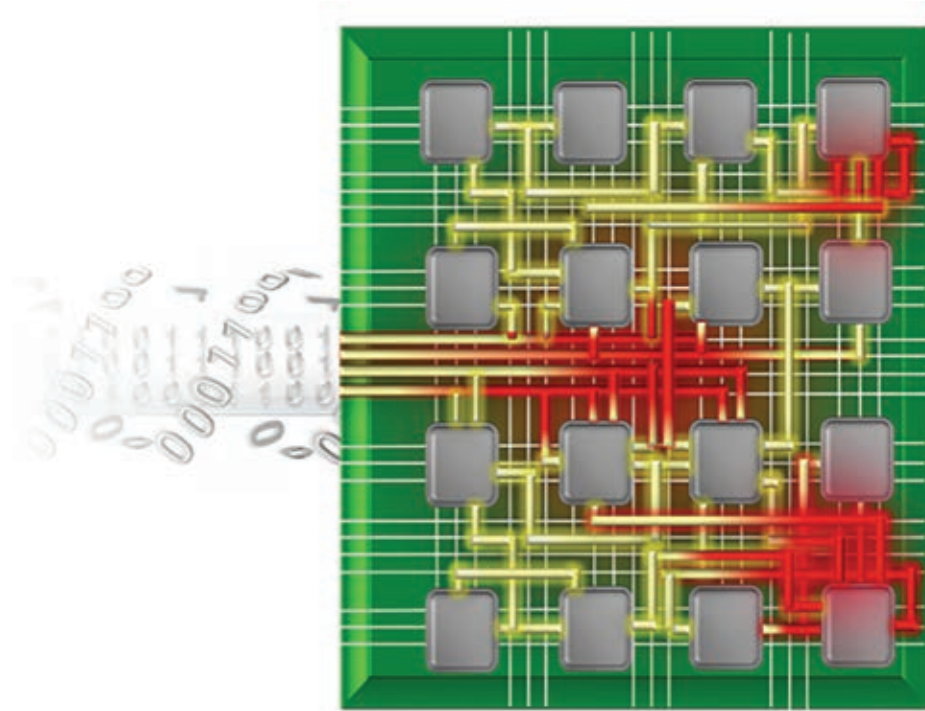


Figure 2 – The UltraScale architecture is capable of processing vast quantities of data.

Data is transported to and from FPGAs based on the UltraScale architecture through a combination of the high-performance parallel SelectIO™ and high-speed serial transceiver connectivity. I/O blocks enable cutting-edge memory interfacing and network protocols through flexible I/O standards and voltage support. The different serial transceivers in the UltraScale architecture transfer data at up to 16.3 Gbps, providing all the performance required for mainstream serial protocols, and at up to 32.75 Gbps, enabling 25G+ backplane designs with dramatically lower power per bit than previous-generation transceivers. All transceivers in UltraScale FPGAs support the required data rates for PCI Express® Gen3 and Gen4, and integrated blocks for PCI Express enable FPGAs based on the UltraScale architecture to support up to x8 Gen3 Endpoint and Root Port designs.

CLOCKING AND BUFFERING DATA

All synchronous systems rely on one or more clock signals for circuit synchronization. Increasing system performance dictates higher clock frequency with larger device capacity, demanding

both improved clock flexibility and lower total clock power.

The UltraScale architecture contains powerful, rearchitected clock-management circuitry, including clock synthesis, buffering and routing components that together provide a highly capable framework to meet design requirements. The clock network allows for extremely flexible distribution of clocks within the FPGA to minimize the skew, power consumption and delay associated with clock signals. In addition, the clock-management technology is tightly integrated with dedicated memory-interfacing circuitry to enable support for high-performance external memories, including DDR4. Clock segmentation and new clock-gating granularity provide extra control of clock power consumption vs. existing FPGAs.

A dramatic increase in the number of global-capable clock buffers compared with previous and competing FPGAs provides a significant advantage to designers' productivity. Historically, it has been necessary to be frugal with global buffer usage, with only 32 global clock buffers located in the center of the FPGA. The UltraScale architecture has

a liberal distribution of global-capable clock buffers throughout the architecture, providing resources where they are required and reducing the need to be parsimonious. In addition, Xilinx has greatly simplified the types of clock buffers over previous FPGA generations, with all the clock-switching, clock-dividing and clock-enabling functionality maintained. The result is a wealth of flexible, capable clock buffers with all the functionality just where it's needed.

STORING, PROCESSING AND ROUTING DATA

The key to any system is its ability to process, manipulate and convert the data it receives (Figure 2). Increasing system complexity requires a combination of general-purpose fabric with more specialist functions dedicated to specific types of data processing.

There are many elements to the fabric of today's FPGAs: configurable logic blocks (CLBs) containing six-input lookup tables (LUTs) and flip-flops; DSP slices with 27x18 multipliers; and 36-kbit Block RAMs with built-in FIFO and ECC support. These resources are all linked together with an abundance of high-performance, low-latency interconnect.

In addition to logical functions, the CLBs provide shift register, multiplexer and carry-logic functionality as well as the ability to configure the LUTs as distributed memory to complement the highly capable and configurable Block RAMs. The DSP slices—with new 96-bit-wide XOR functionality, wider 27-bit pre-adder and 30-bit input—perform numerous independent functions including multiply-accumulate, multiply-add and pattern detect. In addition to the device interconnect, in devices enabled using second-generation SSI 3DIC technology, signals can cross between super-logic regions using dedicated, low-latency interface tiles. These combined routing resources enable easy support for next-generation data bus widths, allowing device utilization of better than 90 percent.

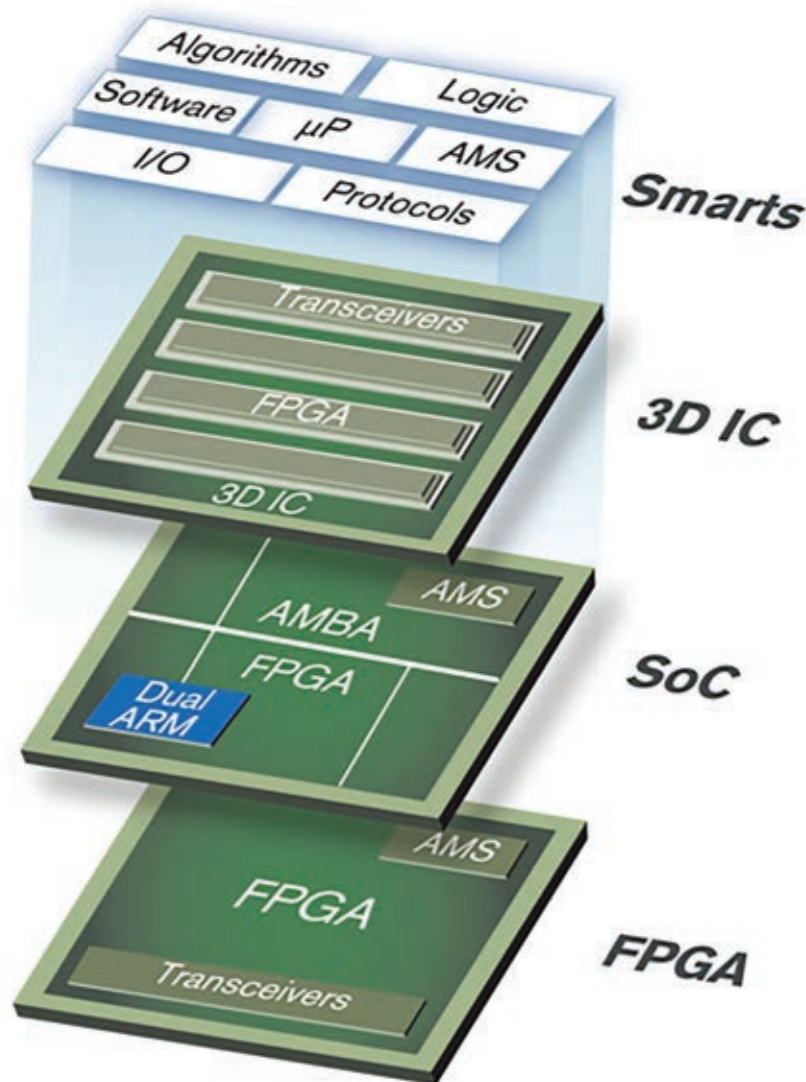


Figure 3 – UltraScale devices add popular functions to industry-leading technology.

EASING THE DESIGN CHALLENGE

The architectural enhancements delivered in the UltraScale architecture enable designers to pack more design in the same area but, in addition, device size is increasing. The result of being able to fit more design into a single device is a major benefit but places pressure on the design team to be able to implement their specified design quickly in order for their end product to achieve the fastest possible time-to-market. With the UltraScale architecture and the co-optimized Vivado® Design Suite, Xilinx delivers several time-saving and productivity-enhancing solutions.

INTEGRATING CORE FUNCTIONS

The flexibility of programmability is a valuable asset but, as with many things, it does not come free. A function built out of programmable resources can be larger, possibly even slower, than a block dedicated to the same function. Of course, by their very nature, FPGAs are predominantly programmable. But Xilinx FPGAs contain the right balance of dedicated-function, integrated IP to enable users to quickly implement commonly used functions (Figure 3). The UltraScale architecture contains integrated blocks for several popular communications protocols. There are multi-

ple integrated blocks for PCI Express, 100G Ethernet and 150G Interlaken in Kintex UltraScale and Virtex UltraScale devices, all fully tested and verified, providing guaranteed functionality.

In addition to the communication protocols, every I/O bank contains a programmable memory PHY, configurable by the Memory Interface Generator (MIG) tool. This is a great illustration of integrating when necessary. The memory PHY and some of the control logic are created as a programmable dedicated function, but the digital portion of the memory interface is built out of device fabric, providing all the necessary customization and support for different modes that a dedicated circuit could struggle to offer.

Within the device fabric are numerous other blocks, tailored to perform specific functions while maintaining an element of programmability. Designers can configure the block memories in numerous depths and widths, cascading to make larger, low-power arrays. The DSP slices have many modes that allow the user access to the various components of the block, depending on the chosen function. So there is a wealth of functionality across the UltraScale architecture beyond mere gates and registers.

CUSTOMIZABLE, REPEATABLE IP ENHANCES PRODUCTIVITY

Every design consists of various architectural building blocks connected together to build a system. Few functions are so established in the industry that it makes economic sense to provide them as dedicated, fixed-function blocks. Rather, the optimum method is to design a function to be built from programmable logic, verify that function and then reuse the function whenever it is needed. The concept of IP in this form has been around for many generations, but Xilinx has recently introduced several productivity enhancements (Figure 4).

PLUG-AND-PLAY IP

In 2012, Xilinx adopted the ARM® AMBA® AXI4 interface as the standard

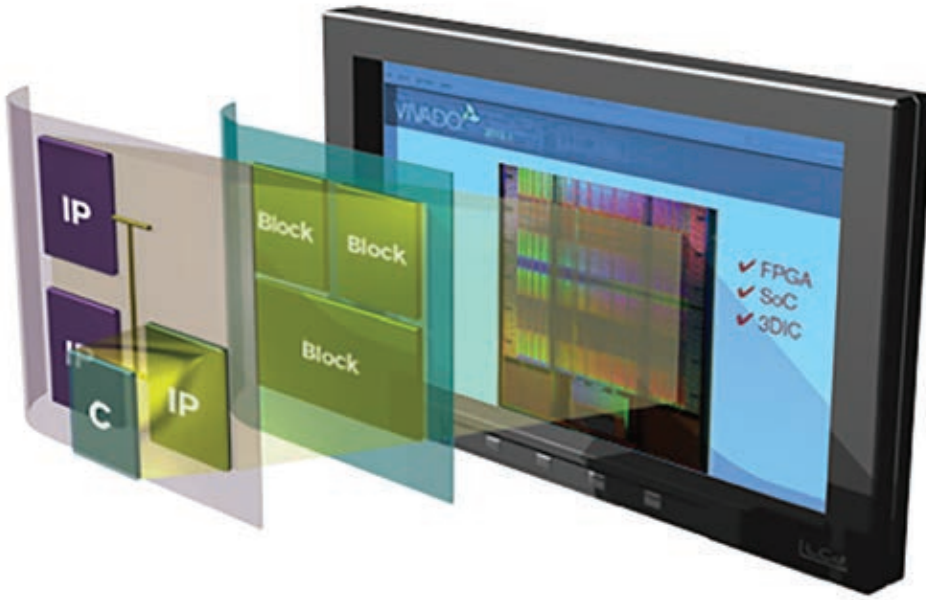


Figure 4 – Vivado tools accelerate the creation and implementation of complex designs.

interface for plug-and-play IP. Using a single, standard interface makes IP integration far easier than in the past, consolidating a broad array of interfaces into one, with designers no longer expected to understand numerous different interfaces. The UltraScale architecture continues to benefit from the flexibility and scalability of AXI4 interconnects to allow designers to achieve the fastest time-to-market while simultaneously optimizing IP for performance, area and power consumption with different AXI4 interconnect protocols, including AXI4-Lite and AXI4-Stream.

The Vivado IP Packager and IP Catalog leverage the IP-XACT standard, originally created by the SPIRIT Consortium as a standard structure for packaging, integrating and reusing IP within tool flows. IP-XACT is now an approved IEEE standard (IEEE1685-2009). The Vivado IP Packager makes a design with its constraints, testbenches and documentation available in an extensible IP catalog on a local or shared drive. The Vivado IP Catalog enables users to combine their own IP with Xilinx and third-party IP in order that all IP can be shared across a design team in a consistent and easy-to-use manner.

VIVADO IP INTEGRATOR

The Vivado IP Integrator (Figure 5) is an IP-centric design flow for accelerating time-to-system integration, making it quicker and easier to put together a system from its constituent parts. With an interactive graphical user interface, IPI provides intelligent autoconnection of IP interfaces, one-click IP subsystem generation and powerful debug capability, enabling designers to quickly and easily connect up any and all of the IP in their IP catalog. This capability enables designers to rapidly assemble complex systems that consist of design sources from many origins—some free, some purchased, some created in-house—with the knowledge that all the building blocks are correctly configured. Getting from concept to debug has never been faster.

In short, the UltraScale architecture includes architectural innovations in many key areas to successfully meet the aggressive demands of next-generation, high-performance designs. Ensuring the ability to implement designs with wide data buses at ever-increasing system frequencies, as UltraScale does, is a major piece of the puzzle. However, with device size and complexity increasing, it is critical to enable designers to continue ramping productivity. Providing a combination of integrated blocks and preverified IP, Xilinx provides the designer with all the tools necessary to implement a superior solution, faster. 🌈

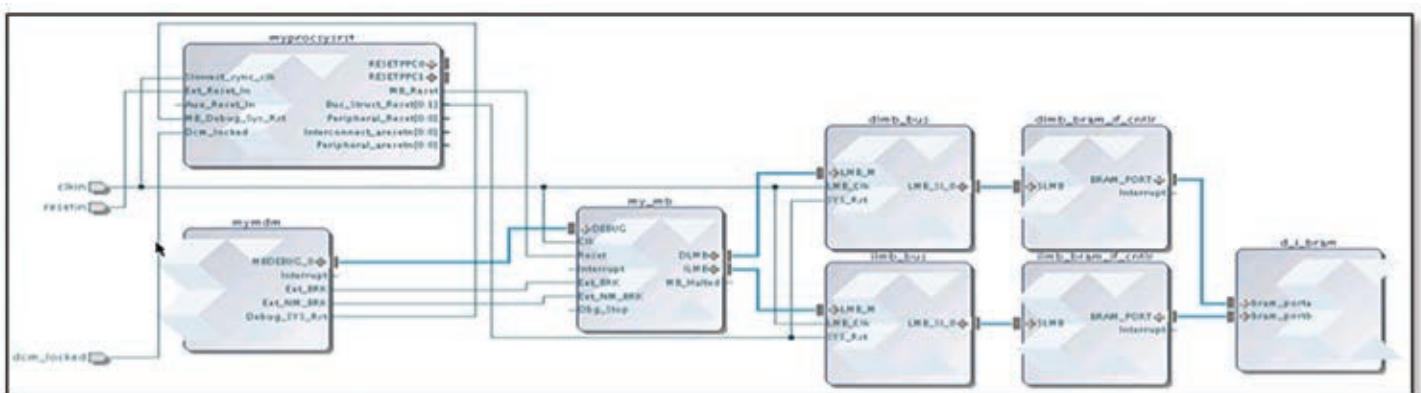


Figure 5 – Building a design in IP Integrator is a simple matter of connecting IP blocks.

UltraScale Brings Interlaken Onboard for High-Bandwidth Designs

Xilinx has integrated Interlaken IP on its UltraScale FPGAs to simplify interconnects in packet-processing systems.

by **Martin Gilpatric**

Transceiver Technical Marketing Manager
Xilinx, Inc.
martin.gilpatric@xilinx.com

B

Bandwidth is a funny thing. Ten years ago, most people barely understood the term “byte per second.” But today, on-line video, smartphones and all of the trappings of our modern interconnected society have pushed bandwidth into the common consciousness. These applications have laid bare to the public why bandwidth is important and why they want more of it.

Unfortunately, that high-level, top-of-the-stack perspective obscures the technological advances required to supply those additional bits and bytes to our homes and to the devices in our pockets. All of that data requires that our infrastructure must keep up. Interconnects that used to rely on 10-Gbps optics have moved to 40 to 100 Gbps, and now we are looking at 400 Gbps in the very near future.

INTERLAKEN IN THE INDUSTRY TODAY

Through all of this, the fundamental packet-processing tasks and architectures have not changed drastically. Similarly, many of the packaging, power and thermal constraints facing 100-Gbps systems are the same faced by their predecessors. This means that every component of the system needs to operate at a much higher rate. More important, the interconnects between these devices need to

be able to scale readily. Confounding this task is the multitude of suppliers for the FICs, NPUs, MACs and other ASSPs used in these systems. These devices use a variety of interconnect widths and rates to achieve the target throughput.

Now, Xilinx and the Interlaken protocol have stepped in to assist in the serial-interconnect bottlenecks between packet-processing functions (Figure 1). Interlaken arose from a joint effort between Cisco and Cortina, and was expressly designed to meet these challenges. Xilinx® FPGAs readily implement this standard via high-performance programmable logic and high-performance transceivers. Three things make Interlaken the optimal protocol for these situations: It is highly scalable, it was designed to work well with OTN and Ethernet systems, and it is an open protocol.

At the most fundamental level, Interlaken is designed to maximize

Potential Locations for Interlaken in Packet Processing

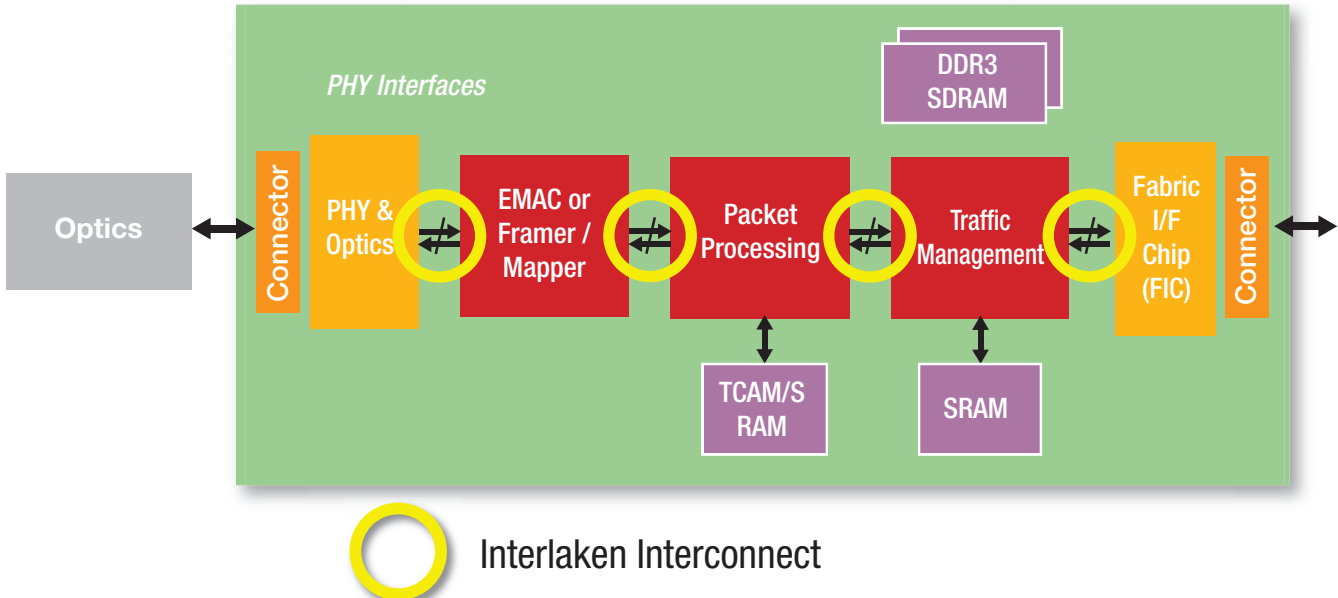


Figure 2 – With Virtex UltraScale’s 32.5-Gbps GTY high-performance transceivers (demonstrated in this video) and integrated Interlaken blocks, customers can create higher-performance, lower-power and more tightly integrated networks without sacrificing flexibility.

flexibility and scalability. There are no predefined line rates or lane widths mandated by the specification. This means, for example, that you can implement a 150-Gbps interface, enough to encapsulate a 100G Ethernet data path (with additional packet headers), in 12 lanes of 12.5 Gbps each or six lanes of 25 Gbps. In either case, the logical interface to Interlaken is maintained, simplifying the rest of the design. Furthermore, this flexibility makes it possible to adapt Interlaken for use in niche applications such as lookaside interfaces when working with TCAMs.

Regardless of the number of lanes the data is being transmitted across, the data is segmented and striped across those lanes. This approach has the advantage of lowering latency while working well with OTN streams or Ethernet packets. Within this segmentation, Interlaken allows for channelization, giving designers the flexibility to implement features like prioritization of packets while working within the protocol's existing feature set.

Interlaken's rich feature set and the fact that it was freely licensed have made for ready adoption. Also, including a variety of stakeholder companies in the original definition meant that Interlaken's feature set was ready to meet a wide selection of needs. A variety of ASSPs and IP quickly became available, easing both adoption and, of course, the use of Xilinx FPGAs.

XILINX AND INTERLAKEN: A PERFECT FIT

Xilinx has a reputation for making it easy for designers to adopt the latest high-performance standards. In the move to 100 Gbps and beyond, Xilinx devices have proven invaluable in leading packet-processing systems and test equipment. The availability of high-quality IP for 100G Ethernet MAC, OTU4 and Interlaken solutions complements Xilinx's high-speed FPGA fabric and world-class transceivers in providing flexible, powerful solutions for customers.



Figure 2 – With Virtex UltraScale's 32.5-Gbps GTY high-performance transceivers (demonstrated in this video) and integrated Interlaken blocks, customers can create higher-performance, lower-power and more tightly integrated networks without sacrificing flexibility.

Interlaken IP has been available for Xilinx FPGAs since the Virtex®-5 generation, originally from third-party vendors and then, after the acquisition of Sarance, through Xilinx itself. The UltraScale™ family of FPGAs is the fourth generation of Xilinx devices to offer Interlaken solutions, with one important advance.

In UltraScale FPGAs, Xilinx has integrated the Interlaken IP into the silicon itself. By making Interlaken a fixed feature, Xilinx frees the hands of designers by opening up more fabric logic and reducing the strain on timing that a soft implementation introduces. The integrated solution also saves both dynamic and static power without sacrificing flexibility. A single Interlaken block can implement as many as 12 lanes at any line rate up to 12.5 Gbps or up to six lanes all the way to 25 Gbps.

XILINX BUILDS ON INTERLAKEN'S ADVANTAGES

The integrated Interlaken IP is only one piece of what makes Xilinx UltraScale FPGAs so attractive. With a large number of feature-rich transceivers supporting a wide range of line rates, advanced high-speed programmable logic, integrated and soft IP for MAC solutions, and the capability to implement emerging standards, no other FPGA or ASSP can measure up.

The GTH and GTY transceivers found in the UltraScale FPGAs have a number of features that enable them to perform in a wide variety of situations. The GTH has line rates ranging from 500 Mbps to 16.3 Gbps, while the GTY can run all the way to 32.75 Gbps, ensuring that the FPGA can support any line rate required by a link partner ([see video demo on YouTube, Figure 2](#)). The equalization capabilities of these transceivers, including continuous-time linear equalization and decision-feedback equalization, allow users to connect to anything from another device on the board to optics or a device on the far side of a backplane.

To ease the bring-up of these high-speed interconnects, whether 100G Ethernet, OTU 4.4 or a wide Interlaken interface, all of the receiver equalization features are auto-adapting. This means that between the three taps of CTLE and 15 taps of DFE in the GTY, the millions of combinations that result are handled by the transceivers themselves. There's no need for the designer to hand-tune each link and maintain margin on those links over changes in process, voltage and temperature. With Interlaken supporting indefinitely wide interfaces, auto-adaptation keeps bring-up simple and robust.

Complementing the integrated Interlaken block is the UltraScale integrated 100G Ethernet MAC, or CMAC. With all of the same advantages as the integrated Interlaken block, the CMAC saves power, complexity and implementation time while providing a ready interface to today's 100G optics: CFP, CXP, CFP2, CFP4 and QSFP28. Connecting to these optics requires either a 10x10.3125-Gbps or 4x25.78-Gbps serial interface. The CMAC and the UltraScale transceivers support these two interfaces—GTH for 10.3125 Gbps and GTY for both line rates. The CMAC provides but one possible protocol to inter-

face off a line card. Xilinx provides IP that can be implemented in fabric for OTN applications or an assortment of Ethernet standards: 1G, 10G, 40G, 100G and even the emerging 400G standard.

Bridging between ASSPs that leverage a multitude of Interlaken configurations and implementing MAC functions are two important uses for an FPGA. But there is a lot more potential to explore in FPGA design. Savvy designers have started to intelligently optimize their systems by having the FPGA handle functions that are costly to perform in their choice ASIC or ASSP. These

functions can include varying levels of packet processing; the implementation of lookaside interfaces to a TCAM; or other features that set a product above the competition.

As technology always has, the world of high bandwidth continues its forward march. This means more throughput and new standards, two areas where Xilinx UltraScale devices are unbeatable. By integrating Interlaken into the UltraScale FPGAs, Xilinx enables existing 100G systems and all future systems to readily adopt new standards, enhance the features of existing architectures and better integrate systems across the board. 🌟

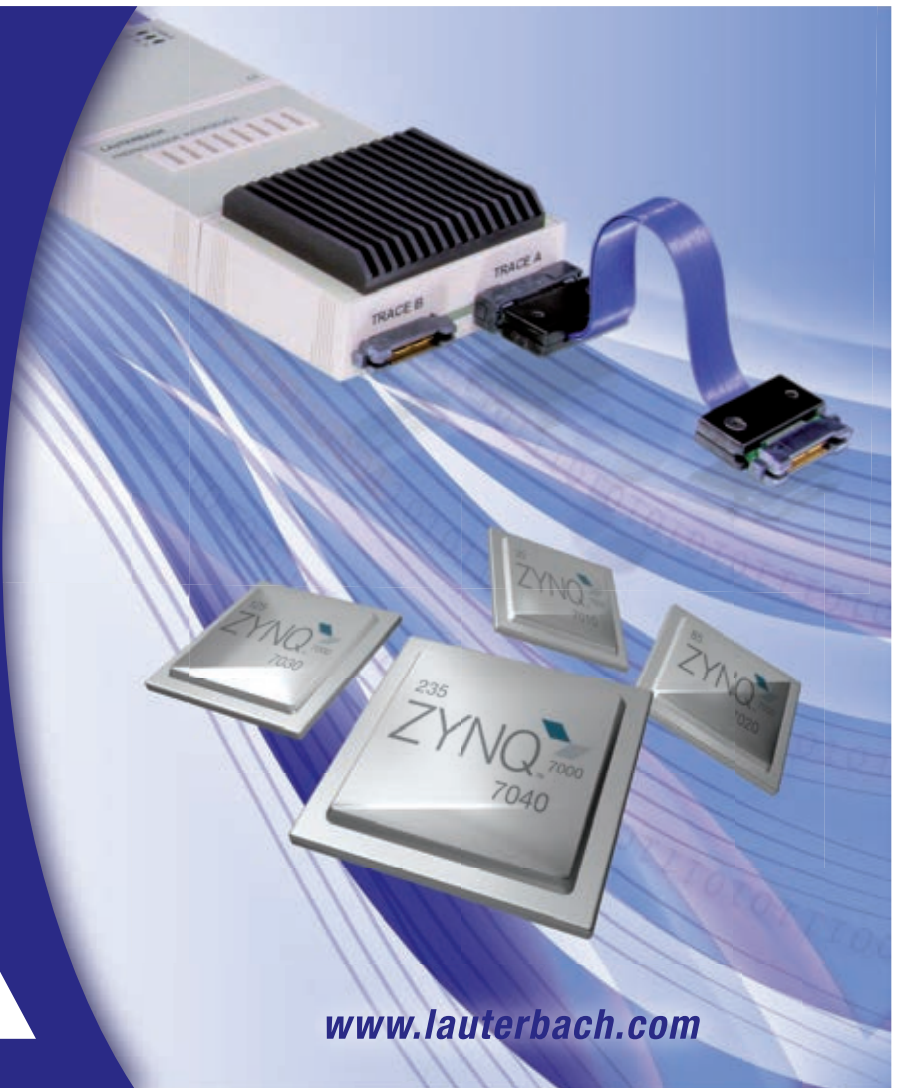
TRACE32®

Debugging Xilinx's Zynq™ -7000 family with ARM® CoreSight™

- ▶ RTOS support, including Linux kernel and process debugging
- ▶ SMP/AMP multicore Cortex™-A9 MPCore™s debugging
- ▶ Up to 4 GByte realtime trace including PTM/ITM
- ▶ Profiling, performance and statistical analysis of Zynq™'s multicore Cortex™ -A9 MPCore™

LAUTERBACH
DEVELOPMENT TOOLS

www.lauterbach.com



FPGAs Help Characterize Massive-MIMO Channels

by Patrick Murphy

President
Mango Communications
patrick@mangocomm.com

Clayton Shepard

Graduate Student
Rice University
cws@rice.edu

Lin Zhong

Associate Professor
Rice University
lzhong@rice.edu

Chris Dick

Chief DSP Scientist
Xilinx, Inc.
chris.dick@xilinx.com

Ashutosh Sabharwal

Professor
Rice University
ashu@rice.edu

A system built from 24 FPGAs, 96 antennas plus a custom 802.11 implementation enables the real-time study of multiuser-MIMO propagation environments.

Multiuser MIMO (MU-MIMO) is a wireless communications technique that leverages multiple antennas at infrastructure nodes, like basestations and access points, to serve many clients simultaneously. MU-MIMO is an integral part of upcoming wireless standards and is expected to deliver significant improvements in the performance of busy networks.

It is envisioned that with each new generation of wireless systems, the number of antennas at basestations will continue to grow, eventually leading to “massive MIMO” systems. The massive-MIMO approach extends the number of antennas at MU-MIMO basestations to tens or hundreds, seeking to improve performance while possibly simplifying the basestation’s signal processing. One scalable massive-MIMO technique is known as conjugate beamforming [1]. An early implementation of this strategy shows the potential for real-world gains [2].

Multiuser-MIMO techniques rely on accurate knowledge of the wireless propagation environment. An MU-MIMO infrastructure node can serve multiple users simultaneously only if it has accurate and recent measurements of the wireless channel to each user. Gathering this channel information in real time is challenging, and the performance impact of stale or inaccurate channel information can be severe.

We have devised an integrated system for massive-MIMO channel characterization that enables researchers to study the dynamics of channels in real time. The system uses the Xilinx® FPGA-based WARP hardware platform and [Mango Communications’ 802.11 Reference Design](#) at its core, scaled up to 24 FPGAs connected to 96 antennas using the Rice University Argos platform [2]. A custom Python framework developed by Mango Communications controls and gathers data from every node in the array in real time. This combination of Mango- and Rice-developed tools provides deep visibility into the wireless stack, including the raw channel data necessary for characterizing massive MIMO.

A key feature of the custom 802.11 implementation by Mango Communications is its ability to stream low-level baseband parameters, like AGC gains, channel estimates and raw packet contents (even packets with errors), from all receiving antennas in real time. This feature of the reference design allows the Rice Argos array to act as a standards-compliant 802.11 access point (AP), serving Internet to commercial Wi-Fi devices (for example, smartphones, tablets or laptops) while simultaneously gathering channel data between the array antennas and each client, all in real time. The Xilinx FPGAs are crucial to enabling real-time

Our massive-MIMO system, Argos, uses FPGAs to locally process the data in real time and significantly reduce the burden on the upstream processors.

processing at each antenna. They condense the data gathered from each antenna into per-client channel characteristics that a custom application can stream and analyze.

Let's take a closer look at the WARP hardware platform and the custom 802.11 implementation by Mango, as well as the conjugate beamforming strategy for MU-MIMO. Finally, we will also examine the characterization process, including the real-time collection of wireless channel measurements from Wi-Fi clients and the processing of channel data to estimate achievable MU-MIMO performance.

SYSTEM COMPONENTS

The Wireless Open-Access Research Platform (WARP) is a scalable and extensible programmable wireless platform, built from the ground up to prototype advanced wireless networks. WARP combines high-performance programmable hardware with an open-source repository of reference designs and support materials.

The WARP Project, founded in 2006 by Rice University professor Ashu Sabharwal, was originally funded by the National Science Foundation with ongoing support from Xilinx. The project has since grown into a self-sustaining

open-source effort with users around the world. Mango Communications spun off from the Rice University WARP Project in 2008 with the initial goal of manufacturing and distributing the Rice WARP hardware. In 2012 Mango released the entirely redesigned WARP v3 hardware. Today, Mango engineers are the most active contributors to the WARP repository and forums, providing ongoing development and support of open-source WARP designs.

A central component in our system for measuring massive-MIMO channels is Mango Communications' WARP v3 hardware platform. WARP v3 is designed for rapid, real-time prototyping of novel wireless designs. The hardware integrates a high-performance Xilinx Virtex®-6 FPGA, two flexible RF interfaces and multiple peripherals, including DDR3 DRAM and two 1-Gbps Ethernet interfaces. The WARP v3 board can be extended to four RF interfaces with Mango's dual-radio FMC module. This hardware configuration, shown in Figure 1, provides four fully programmable RF interfaces with independent digital baseband connections to the FPGA.

To study massive-MIMO systems, it's necessary to co-locate multiple WARP v3 nodes with shared power, clocking and Ethernet connectivity. This requirement has been addressed by the [Rice University Argos project](#). The Argos v2 array is a collection of 24 four-antenna WARP v3 nodes, pictured in Figure 2. The Argos array is designed to support a wide variety of massive-MIMO experiments and is perfectly suited to gather-



Figure 1 – The WARP v3 hardware with dual-radio FMC module provides a large FPGA, four RF interfaces, memory and two Ethernet connections.



Figure 2 – The Rice University Argos v2 array combines 24 quad-radio WARP v3 nodes with shared clocking and Ethernet connectivity.

ing channel measurements simultaneously across all 96 array antennas.

The FPGA on each WARP v3 node in the Argos array provides significant processing power close to the RF interfaces. In a massive-MIMO configuration like Argos, there is a tremendous amount of data to process. For example, when receiving 40 MHz of band-

width, each RF interface on WARP v3 generates a 960-Mbps sample stream (dual 12-bit 40-Msample/second ADCs). The full Argos array generates 96 times this amount—far more than can be streamed to a PC and processed in real time. Instead, the system uses FPGAs to locally process the data in real time and significantly reduce the burden on

the upstream processors. For our massive-MIMO channel-characterization design, this real-time processing is very important as it allows the design to continually measure channels and reliably observe submillisecond variations in channel characteristics. The custom FPGA design that performs this processing is the Mango Communications 802.11 Reference Design for WARP v3.

This reference design is a real-time FPGA implementation of the 802.11a/g medium-access control (MAC) layer and physical (PHY) layer. The design can interoperate with standard Wi-Fi devices, acting as an AP (serving Wi-Fi clients), a client (accessing a Wi-Fi AP) or a monitor (a receive-only passive observer of network activity). You can customize both the MAC and the PHY to explore new variations to the standard. This combination of interoperability and extensibility enables a wide range of wireless communications and networking experiments. The full source for the 802.11 reference design is available to users of WARP v3 hardware at no cost.

Figure 3 illustrates the reference design architecture. The design uses two Xilinx MicroBlaze™ cores to implement the high- and low-level MAC protocol in software. The MAC connects to two FPGA cores, which implement the PHY transmitter and receiver. We implemented these PHY cores in Xilinx System Generator. The transmitter core implements a complete bytes-to-waveform pipeline. It reads a packet payload from the MAC, creates the OFDM waveform and drives the waveform to the RF interface DACs. This pipeline includes encoding, scrambling, interleaving, IFFT and preamble insertion. The MAC specifies the modulation and coding rates per packet; all eight data rates specified in 802.11a/g are supported.

The receiver design implements the full waveform-to-bytes pipeline, including AGC, synchronization, FFT, channel estimation, equalization, detection and decoding. The receiver configures its

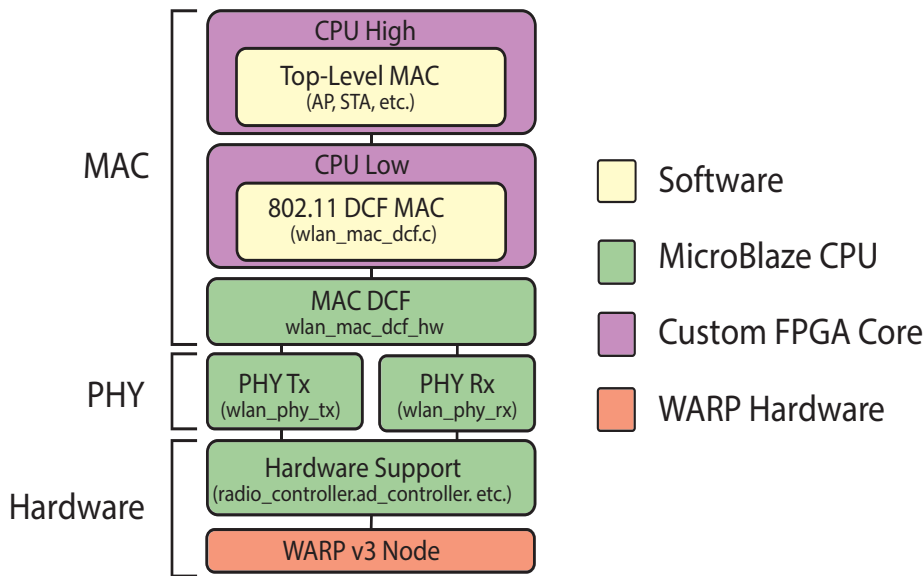


Figure 3 – The Mango 802.11 Reference Design architecture includes two Xilinx MicroBlaze CPUs for the MAC and custom System Generator cores for the PHY transmitter and receiver.

demodulation and decoding blocks automatically per packet using the RATE value in the packet's SIGNAL field. The receiver decodes packets of any rate fast enough to meet the standard's strict Rx-to-Tx turnaround requirements for transmitting an acknowledgement (ACK) in response to a reception.

One feature of our receiver design that is central to characterizing massive-MIMO channels is the channel-estimation subsystem. In a standard OFDM receiver, the channel estimator produces a complex channel coefficient per subcarrier. The equalizer uses these estimated coefficients to correct the channel's amplitude and phase degradations for each received data symbol. Our design additionally saves a copy of the channel estimates for every received packet to an on-chip memory area. The MAC treats these channel estimates as extra metadata about the received frame, along with standard information like Rx power, AGC gain selections, checksum status and antenna selection. The channel estimates are then copied to the higher-level MAC for further processing. Our

characterization platform gathers these estimates from every packet received by every node in the Argos array to assemble its real-time view of the massive-MIMO propagation environment.

WARPNET EXPERIMENTAL FRAMEWORK

The final piece for our massive-MIMO characterization system is a framework, dubbed WARPnet, for running experiments with large networks of WARP nodes. WARPnet is a custom Python package that uses a dedicated control connection to multiple WARP nodes. The framework allows a Python script running on a PC to remotely configure experiment parameters and retrieve experimental data, all in real time. WARPnet interacts with the Mango 802.11 Reference Design via the secondary Ethernet connection on each WARP v3 board. The upper MicroBlaze device processes the WARPnet commands, giving the framework direct access to the node's high-level MAC state and all data passed up from the low-level MAC and PHY.

For our massive-MIMO channel-characterization design, the WARPnet framework maintains a connection to every node in the Argos array. Each node is configured as an 802.11 monitor, capturing channel estimates from every received packet and offloading these packets via Ethernet for further analysis.

The full Python source code for WARPnet is open-source in the WARP repository.

UNDERSTANDING MULTIUSER MIMO

Basestations employing multiuser-MIMO techniques seek to create waveforms for many transmit antennas which, when combined by the wireless channel, deliver data to multiple users simultaneously. Creating multiuser waveforms requires sophisticated processing at the basestation. Many MU-MIMO techniques have been proposed. A common requirement of MU-MIMO designs is accurate knowledge of the wireless propagation characteristics from each basestation antenna to each client device.

One approach to MU-MIMO is known as zero-forcing, which in theory—and recently in practice [3]—has been shown to achieve significant performance gains over single-user techniques. The zero-forcing approach aims to maximize the signal-to-interference-plus-noise ratio (SINR) at each client's receive antenna. Maximizing SINR requires maximizing the power of the signal representing a user's payload (the "S") while minimizing the power of every other user's payload (the "I") in the waveform that arrives at the user's antenna. Zero-forcing requires very sophisticated processing at the basestation. With zero-forcing, the calculation of the transmit waveform for a given basestation antenna requires knowledge of the payload for every user and the wireless channel from every other antenna to every user. The computational complexity of this calculation increases significantly as the number of basestation antennas increases.

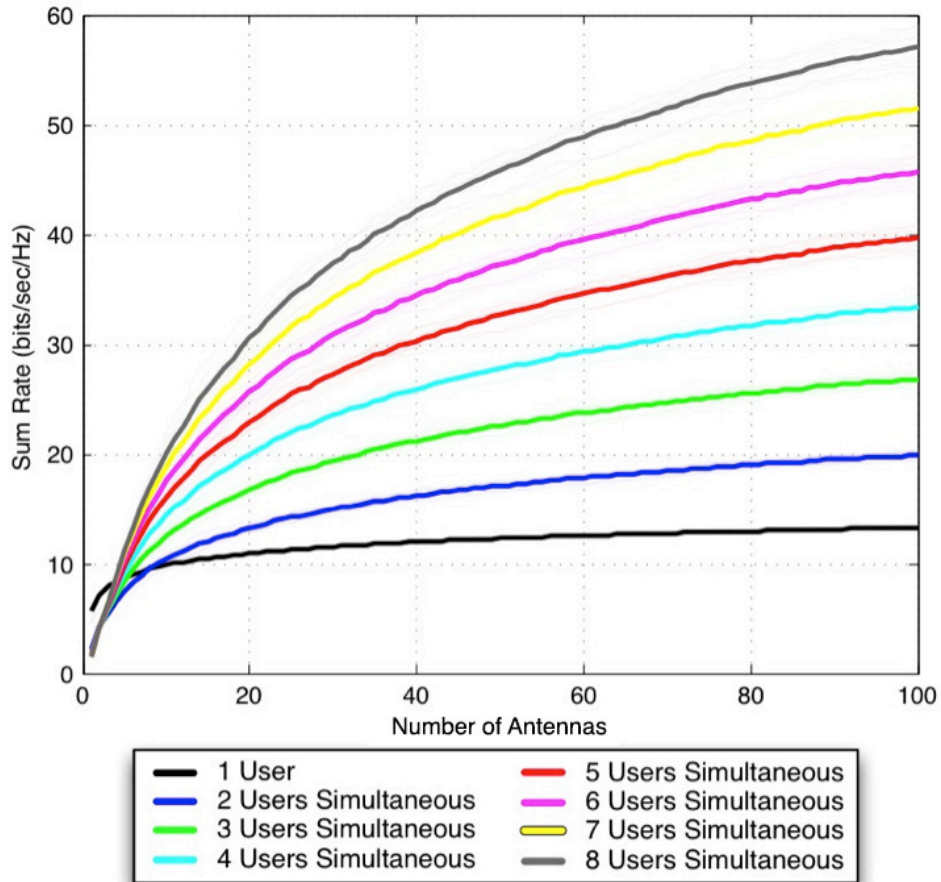


Figure 4 – Simulation of a multiuser-MIMO network shows significant rate improvements for many users when sufficient antennas are installed at the access point.

Conjugate beamforming [1] is an alternative MU-MIMO technique. In this approach, the basestation seeks to maximize the good signal power delivered to each client device without actively minimizing the interference power. In theory, the conjugate-beamforming approach of increasing SINR at each user by maximizing signal power (the “S” in SINR) while ignoring interference power (the “I” in SINR) improves with an increasing number of antennas. Further, the conjugate-beamforming calculation of each transmit antenna’s waveform does not require knowledge of any other antenna’s channel characteristics. Together these factors make conjugate beamforming well suited to massive-MIMO systems, where a basestation has many more antennas than users.

Consider the classic expression for Shannon channel capacity, $C = \log(1+\text{SINR})$. The capacity of the wire-

less channel, in bits/second/Hz, grows logarithmically with the SINR. Conjugate multiuser beamforming has two competing effects when the system adds more users and antennas. First, the presence of multiple antennas allows for an increase in received signal power, since each antenna can rotate its phase such that transmissions are constructively combined at the users’ receivers. Second, the presence of multiple transmissions to independent users creates an increase in interference power. The superimposed interference signals combine randomly. As the number of antennas increases, the growth in constructively combined signal power outruns the randomly combined interference power, increasing overall SINR.

The simulation results in Figure 4 illustrate the impact of increasing the number of basestation antennas on the overall network capacity when con-

jugate beamforming is employed. The simulation assumes a network with one basestation and eight users, with wireless channels modeled by independent and identically distributed Rayleigh fading. The simulation shows the overall network rate when one to eight users are served simultaneously vs. the number of antennas employed by the basestation. For a small number of antennas, we can see that conjugate beamforming to more than one user at a time is not beneficial. If basestations are limited to just a few antennas, traditional single-user beamforming with time-sharing may be superior to multiuser conjugate beamforming. As the number of antennas increases, more users can be supported, for substantial overall network rate improvements.

This simulation uses idealistic channel models to show that multiuser conjugate beamforming might be able to achieve performance gains. Whether these gains can be realized in a real system depends on the actual wireless channels between the basestation and client devices. Our MU-MIMO channel-characterization platform can measure the channels from the basestation to actual user devices in real time, providing a powerful tool to evaluate the real-world performance of MU-MIMO techniques.

PUTTING IT ALL TOGETHER

Now that we’ve explored the motivations for measuring massive-MIMO channels and the tools provided by the Rice Argos array, WARP hardware and Mango 802.11 Reference Design, let’s take a look at how to assemble these elements into a complete real-time, massive-MIMO channel-characterization platform.

The 24 WARP v3 nodes in the Argos array are configured with a custom version of the Mango 802.11 Reference Design. This version operates in receive-only monitor mode, attempting to receive packets on all four of the node’s antennas. For every packet reception, the node estimates the complex chan-

nel coefficient for each subcarrier, decodes the packet and sends the packet header and channel estimates via Ethernet for analysis. This processing flow is implemented in all 24 nodes in the array, with all nodes operating in parallel.

To communicate with standard Wi-Fi devices, the channel-measurement platform must also implement a standard 802.11 access point. Another WARP v3 node is used for this purpose, running the Mango 802.11 Reference Design in AP mode. This AP node serves as the 25th node in the Argos array. The AP advertises an open Wi-Fi network, accepts associations from commercial Wi-Fi devices and serves Internet access via its primary Ethernet connection.

This is the standard behavior of the AP profile in the Mango 802.11 Reference Design. To enable real-time channel measurements, this AP implements one additional function. Using the secondary Ethernet connection on the WARP v3 board, the AP node sends an Ethernet packet every time a Wi-Fi client joins or leaves the wireless network. The channel analysis application (discussed below) uses these association updates to maintain its local table of active clients.

CLIENT TRANSMISSIONS

One key challenge in gathering channel estimates from packets received from commercial Wi-Fi devices is ensuring the devices transmit often enough. Modern Wi-Fi devices often employ aggressive power-saving schemes, disabling their Wi-Fi radios when no application is requesting network access. The devices will periodically check in with the AP but possibly not frequently enough to ensure up-to-date channel estimates at the array.

We address the issue of infrequent client transmissions with two “hacks.” First, we modify the traffic indication map (TIM) field in beacons transmitted by the platform AP to inform all connected clients that new data packets are queued for them. The TIM field is normally used to support power saving at clients, allowing clients to briefly wake up in receive-only mode to receive the beacon, decode the TIM and resume low-power mode if no traffic is waiting. By listing every node in every beacon’s TIM field, nodes will sleep less often.

The second technique solicits client transmissions by using the ACK packets the client devices have transmitted. The array can extract channel esti-

mates from any packet a client transmits, including short ACKs. However, the 802.11 ACK packet only includes a destination MAC address, typically preventing the array from identifying the transmitting client.

We work around this issue by exploiting a quirk in the 802.11 MAC specification. The standard requires 802.11 devices to send positive acknowledgment packets after successfully receiving a unicast packet addressed to the client. The “must ACK” requirement applies even if the packet’s source address is not recognized. Thus, to trigger an ACK transmission by a client that contains an identifier unique to the client, the AP sends a data packet with a bogus (but unique) source address. Upon receipt, the client then transmits the ACK to the bogus (but unique) address the AP used. The array nodes receive this ACK and can unambiguously associate the resulting channel estimates with the transmitting client. This trick works remarkably well for triggering frequent updates of channel estimates at the array, and is only possible because of the full programmability of the Mango 802.11 Reference Design.

REAL-TIME ANALYSIS

The final component of the massive-MIMO channel-measurement platform is a custom application that gathers the array-channel estimates, computes achievable multiuser capacities and displays the results in real time. We developed this application in Objective-C, using native UDP sockets to interface with the WARP v3 nodes in the array and OS X graphics frameworks for plotting results.

The application has two primary views. The first displays the raw channel magnitudes that each array antenna gathers for every subcarrier, 4,992 data points in all (52 subcarriers x 96 array antennas). This view is a raw display of the channel data gathered by the array and serves primarily to convey the wide range of channel values observed by individual array antennas.

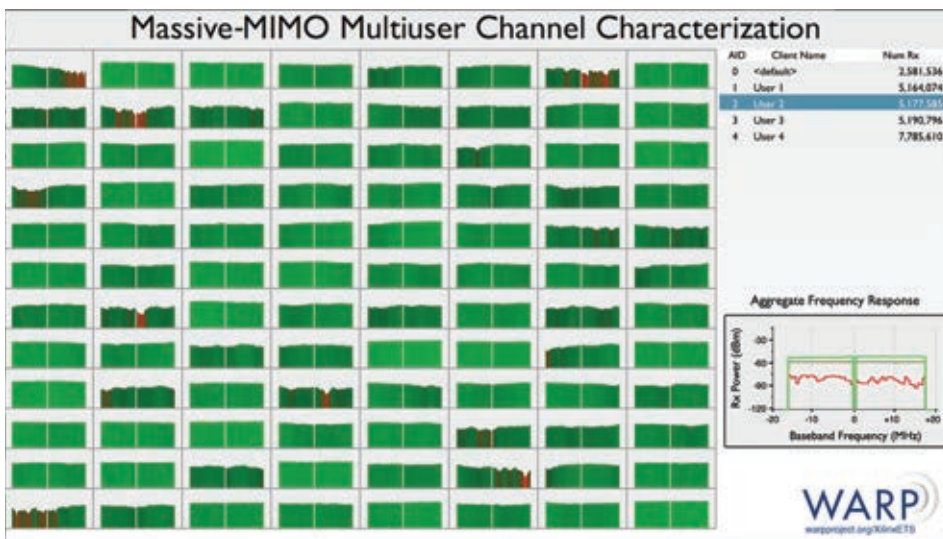


Figure 5 – Channel-magnitude view in our custom MU-MIMO channel-analysis application. Each bar plot shows magnitudes per antenna, per subcarrier.

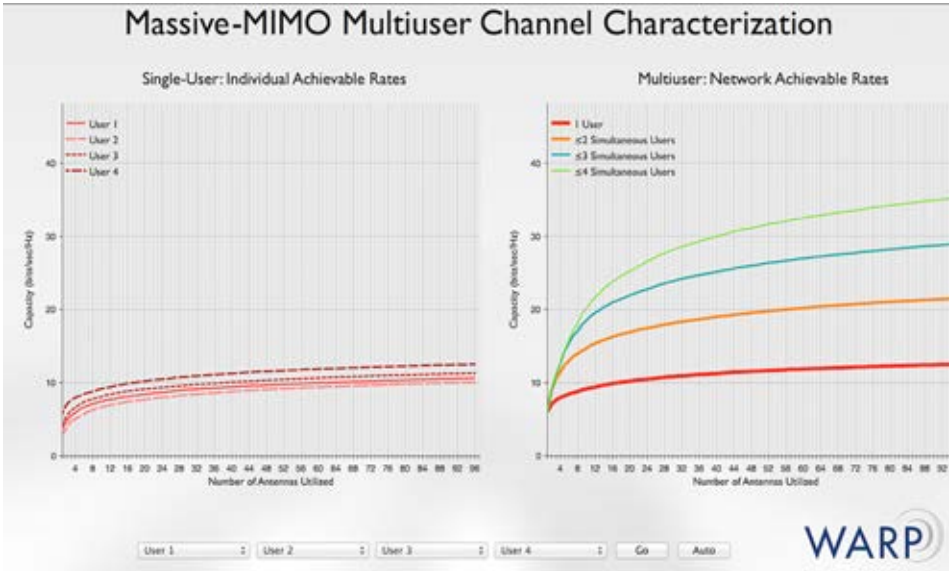


Figure 6 – Estimated network capacity for single-user and multiuser techniques were calculated from real MU-MIMO channel measurements the array had gathered, as displayed by our custom analysis application.

A screen shot of this view is shown in Figure 5. In practice, the view updates in real time (10 frames per second with active Wi-Fi clients).

The application’s second view displays results of capacity calculations based on the array’s channel estimates. This view is illustrated in Figure 6. Two capacity calculations are performed. The first plots the capacity to each user vs. the number of array antennas used. Each line on this plot approximates the achievable downlink capacity to a single user assuming the array used a subset of its antennas in a traditional, single-user beamforming configuration. The decreasing slope of each capacity curve with increasing antennas clearly demonstrates the diminishing benefit of many antennas with traditional, single-user wireless techniques.

The second plot shows the total network capacity if the array implements downlink multiuser beamforming techniques using a subset of its antennas. The trends on the four plots clearly highlight the benefit of additional antennas when multiuser techniques are employed. The increasing slope when

additional users are served highlights the “outside the log” (often labeled “pre-log” in MIMO literature) gain in network capacity with multiuser beamforming.

We demonstrated the massive-MIMO channel-measurement platform at the 2014 Xilinx Emerging Technology Symposium (ETS) in February. Details of this demonstration, including videos and links to supplementary material, are available at <http://warpproject.org/XilinxETS>.

REFERENCES

1. T.L. Marzetta, “Noncooperative Cellular Wireless with Unlimited Numbers of Base Station Antennas,” *IEEE Transactions on Wireless Communications*, vol. 9, no. 11, pp. 3590–3600, 2010
2. C. Shepard, H. Yu, N. Anand, E. Li, T. Marzetta, R. Yang and L. Zhong, “Argos: Practical Many-Antenna Base Stations,” *Proceedings of ACM MobiCom*, pp. 53–64, 2012
3. Q. Yang, X. Li, H. Yao, J. Fang, K. Tan, W. Hu, J. Zhang and Y. Zhang, “Bigstation: Enabling Scalable Real-time Signal Processing in Large MU-MIMO Systems,” *Proceedings of ACM SIGCOMM*, pp. 399–410, 2013

Everything FPGA.

1. MARS ZX3
Zynq™-7020 SoC Module

- Xilinx® Zynq-7020 SoC FPGA
- Up to 1 GB DDR3L SDRAM
- 16 MB QSPI flash
- 512 MB NAND flash
- USB 2.0
- Gigabit Ethernet
- 85,120 LUT4-eq
- 108 user I/Os
- 3.3 V single supply
- 67.6 × 30 mm SO-DIMM

VxWorks

2. MERCURY KX1
Kintex™-7 FPGA Module

- Xilinx® Kintex™-7 FPGA
- Up to 2 GB DDR3L SDRAM
- 16 MB QSPI flash
- PCIe® x4 endpoint
- 4 × 6.6/10.3125 Gbps MGT
- USB 3.0 Device
- 2 × Gigabit Ethernet
- Up to 406,720 LUT4-eq
- 178 user I/Os
- 5-15 V single supply
- 72 × 54 mm

3. FPGA MANAGER
IP Solution

→ USB 3.0

→ PCIe® Gen2

→ Gigabit Ethernet

Streaming,
made simple.

One tool for all FPGA communications. Stream data from FPGA to host over USB 3.0, PCIe®, or Gigabit Ethernet – all with one simple API.

4. PROFINET IP Core

- Optimized for Xilinx FPGA and Zynq SoC
- IRT cycle times as low as 31.25 µs
- Hardware IEEE 1588 PTP implementation
- Isochronous traffic can bypass the software stack
- Co-developed by ZHAW IneS and Enclustra

Enclustra. We speak FPGA.

Design Center • FPGA Modules
Base Boards • IP Cores

ENCLUSTRA
FPGA SOLUTIONS

Fourth Quarter 2014

Xcell Journal

25

4K TV Development Made Easy with the Zynq SoC

by **Roger Fawcett**

Managing Director

OmniTek

roger.fawcett@omnitek.tv



4K

Xilinx All Programmable technology can be a boon for designers of 4K video systems. Associated tools, IP and reference designs will help those new to FPGA design.

Ultrahigh-definition (UHD) TVs, also called 4K for their resolution level, are already widely available and 4K is proving a much more popular technology than 3D TV among consumers. But

the standards lag behind the uptake. Society of Motion Picture & Television Engineers (SMPTE) standards for 6-Gbps and 12-Gbps SDI, supporting 4K60 video, are only just being released, while HDMI™ 2.0 and DisplayPort supporting the same resolution are in the early stages of adoption. Given the significant consumer demand for 4K UHD TVs, many ad hoc standards have rushed in to fill the void.

Indeed, so much about 4K UHD TV is in a state of flux that it is essential for systems to be flexible enough to adapt to the developing standards. The way to ensure flexibility is to replace the time-honored chip sets and ASSPs long used for these designs with FPGAs and All Programmable systems-on-chip such as the Xilinx® Zynq®-7000 All Programmable SoC. These solutions offer the flexibility needed while also delivering performance comparable to that of ASICs.

At the same time, the size and performance of the latest FPGAs and SoCs present considerable design challenges, especially to engineers who are not particularly FPGA-savvy. While there are many similarities in the design of hardware and FPGA implementations, FPGA-based systems typically involve many more components. In addition, the inherent flexibility of firmware designs introduces extra complications.

Fortunately, Xilinx offers a lot of help for 4K TV designers—all at a much lower cost in both time and money than designing your system from scratch. But before delving into the details of how to use FPGA technology for 4K applications, let's first take a look at how 4K systems have managed to become so popular so quickly, and the issues that any 4K system has to address.

THE UPSIDES AND DOWNSIDES OF 4K

Ever since television was first invented, there has been a constant drive toward making the images it shows closer to real life. This effort typically comes down to providing bigger, better and faster video by increasing the resolution,

the frame rate or the dynamic range of the images (that is, how bright they can be)—plus, of course, attempting to achieve either a true 3D effect or at least a more immersive feel.

Increasing the resolution allows the images to be more detailed and makes it possible to display them on larger screens without the pixelation becoming obvious. Bigger screens give a more immersive feel. These are improvements that customers find easy to appreciate and hence, are willing to spend money on. The improvements brought about by increasing the frame rate (smoother motion) or the dynamic range (brighter lights and darker blacks), while compelling, have been slower to capture the consumer’s imagination so far.

The new 4K UHD TV represents a quadrupling of the number of pixels from the previously sought-after HD standards. Perhaps most important for customers, 4K allows them to upgrade to a much larger TV that offers a much more immersive feel without any obvious effect on the image quality.

There are, however, plenty of technical challenges intrinsic to developing systems to support 4K video. For a start, a frame size of 3,840 x 2,160 pixels delivered at frame rates of up to 60 Hz represents a 600-MHz pixel rate. It takes a very high-performance system to process this rate in real time. Then there are the different delivery configurations that are being

defined for 4K—all involving multiple data streams, some delivered multiplexed on the same cable, some on different cables—and the different technologies emerging to supply them: 4x3G; 6G-SDI and 12G-SDI; HDMI 1.4 and 2.0; DisplayPort 1.2; and V-by-One HS.

Another issue for designers is the need for any system to handle not just 4K but also many if not all of the video standards currently in use, including SD. In addition, the system must support conversion among these different standards with all the associated issues of up/down/cross-conversion, nonmatching color spaces, color correction, interlacing and deinterlacing, and cadence handling. An additional complication is that upconversion typically also needs to be followed by the application of so-called “super-resolution” enhancement techniques to counteract the image smoothing that inevitably results.

Other processes that may be needed include noise reduction, cropping and resizing—all to be done in real time. Some systems may also need to handle High-Bandwidth Digital Copy Protection (HDCP).

Furthermore, anyone needing to determine the quality of the broadcast transmissions will also need to generate appropriate eye and jitter displays, the technology for which becomes increasingly difficult to implement at higher bit rates.

FIRST LEVEL OF ASSISTANCE: 4K IP CORES

The first step in designing any system is to find ready-made blocks that you can usefully include in your design. In the FPGA world, the equivalent building blocks to the various chips available for inclusion in a PCB design are intellectual-property (IP) cores. So your first step is to identify what IP cores are available for use in your 4K UHD design.

OmniTek is a good source of IP cores for all types of video system design. The company, which is a certified member of the Xilinx Alliance Program, has a depth of experience in video processing, initially as developer of its own video test-and-measurement (T&M) systems. These systems needed dedicated hardware, which in turn led to the development of dedicated firmware blocks. Those firmware blocks are now also available as IP cores. The creation of OmniTek’s latest T&M system, the newly launched Ultra 4K Tool Box, led to the development of a range of 4K-capable IP cores, now available to third-party developers.

Two cores in particular are useful to designers of 4K systems: OmniTek’s OSVP v2 Scalable Video Processor and its Multi-Channel Streaming DMA Controller, both of which are available targeted for Xilinx 7 series FPGAs and Zynq SoCs. Both cores adopt the ARM® AMBA® AXI4 system interconnect standards.

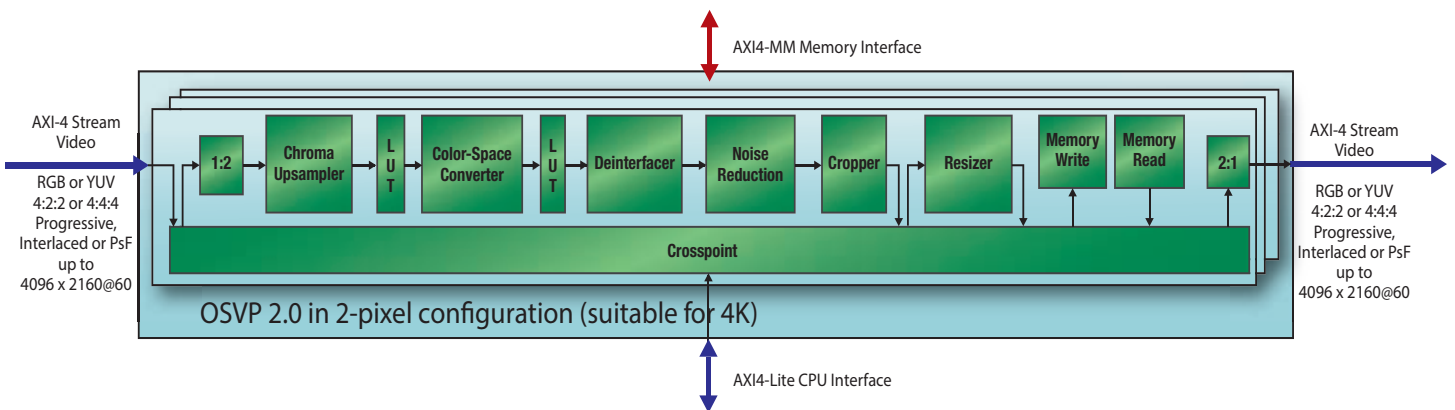


Figure 1 – Input channel architecture of OmniTek’s OSVP v2 Scalable Video Processor core

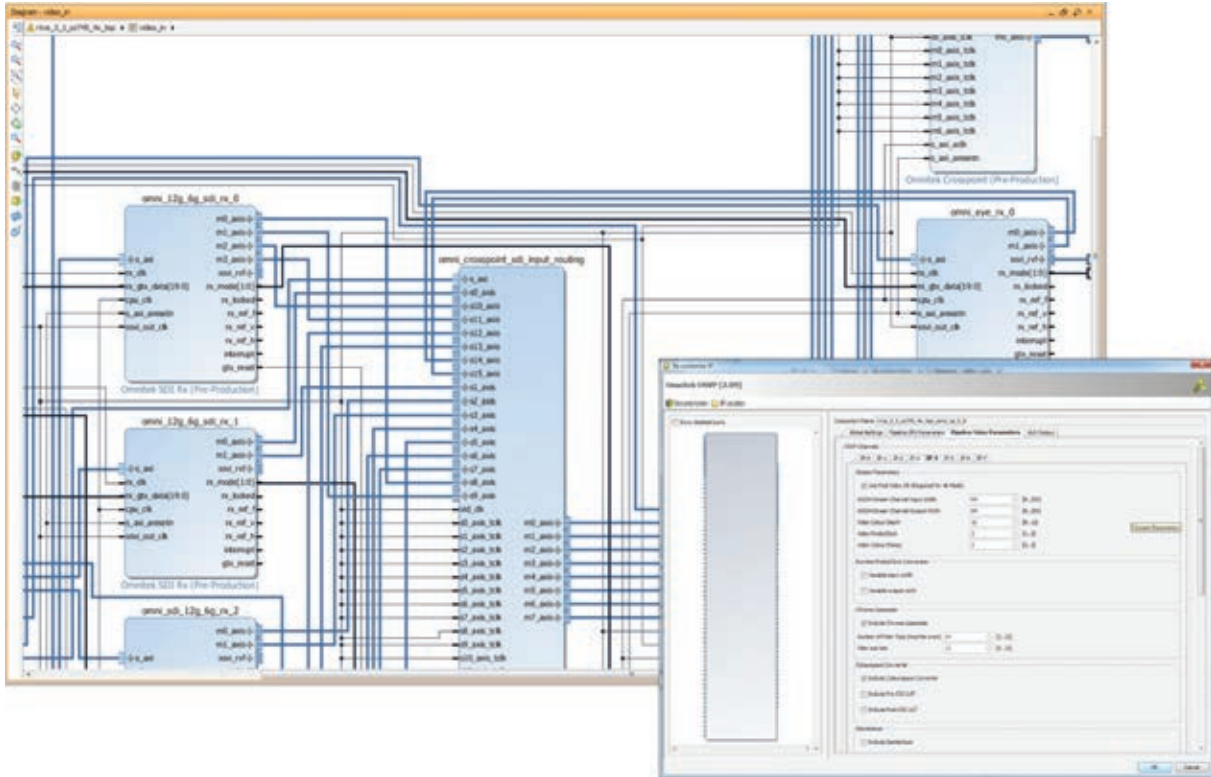


Figure 2 – The RTVE 3.1 video design as laid out within Vivado IP Integrator, together with Configuration window

The OSVP v2's facilities include six-axis color correction; motion- and edge-adaptive deinterlacing (complete with 3:2 and 2:2 film cadence detection and processing); the ability to resize and crop with image sharpening and smoothing; and noise reduction. Figure 1 shows a block diagram of this core. You configure the selection of processing facilities that are included at compile-time, while further details of the processing carried out by the OSVP v2 core can either be set at run-time or driven from software.

The OSVP v2 core comes as part of a suite that also includes a combiner for combining multiple video streams; an interlacer for producing output in interlaced formats; a dedicated crosspoint; and a chroma resampler that you can use to migrate between 4:4:4, 4:2:2 and 4:2:0 YCbCr. (A single implementation of the chroma resampler is able to convert from 4:4:4 to 4:2:2; from 4:2:2 to 4:2:0; from 4:2:2 to 4:4:4; or from 4:2:0 to 4:2:2.)

A single OSVP v2 core can process multiple video channels. The limiting factors are the resources offered by the FPGA or SoC on which it is implemented and the amount of SDRAM bandwidth available. For example, you could configure an OSVP core implemented on a Kintex[®]-7 XC7K325T FPGA to support up to eight inputs handling video at eight different HD video standards, eight color spaces and so on. At the same time, you could configure the output block for up to 16 progressive HD outputs. Alternatively, you could set up the output block to offer either a single 4K channel or a set of four channels that together offer Square Division (“quad”) or 2-Pixel Sample Interleave 4K.

Another challenge for those designing complex 4K systems is managing the many high-bandwidth memory accesses required in processing video. Sometimes the required video handling is offered alongside the video-processing block. For example, the OSVP v2 core includes a Multi-Port Video DMA

block that provides a highly efficient engine for handling video input/output.

Capturing and playing out one or more channels of 4K60 over PCI Express[®], however, requires a DMA controller optimized for handling streaming data across a PCIe[®] interface. OmniTek's Multi-Channel Streaming DMA Controller has a couple of key features to help here. The first is FIFO-based DMA (FDMA), which bypasses the need to transfer the data in and out of memory. The second is a series of design optimizations that allow the controller to make highly efficient use of the PCIe bandwidth, such as by prefetching of scatter-gather-mode descriptors and back-to-back packing of TLP packets.

Another IP core that OmniTek has developed for working with 4K UHD video is a block for unraveling the different streams that make up two-sample interleave forms of 4K video. Also, a drop-in replacement for the basic MIG SDRAM controller further improves the performance for UHD TV video applications.

THE PROGRAMMABLE ADVANTAGE

Further support for designers of FPGA- and SoC-based 4K video systems comes from Xilinx and it comes in three forms.

The first advantage resides in the Zynq SoC, a device that provides a powerful combination of hardware and software processing capabilities for high-performance video or image processing. The Zynq SoC integrates a feature-rich dual-core ARM Cortex™-A9 processing system with 7 series (28-nanometer) FPGA programmable logic in a single device. Users can either run processing algorithms on the ARM processors or offload them into FPGA hardware when acceleration is needed to achieve real-time operation.

The 300-MHz sustainable video-processing speed offered by the programmable logic of both Kintex-7 FPGAs and the Zynq SoC, combined with a

memory performance of 64-bit DDR3 at 1,600 Mbps, is critical for handling 4K video processing and 4K frame buffers. The Zynq SoC's DSP-rich programmable logic fabric gives DSP designers a highly flexible platform on which to implement signal-processing algorithms, while the tight coupling between the processor and the programmable logic allows the development of codec algorithms across both domains. Basing a design on a Zynq SoC also saves power and cost, since you are able to integrate in a single device what would otherwise take multiple ASSPs to accomplish.

Xilinx also offers significant connectivity support to enable 4K video system development, both through the range of built-in transceivers included on its FPGAs and SoCs and through its broad range of in-house connectivity IP. The Zynq 7045 SoC,

for example, offers up to sixteen 12.5-Gbps transceivers, allowing its use in conjunction with the 12G-SDI, 6-Gbps HDMI 2.0, 5.4-Gbps DisplayPort 1.2 and 10-Gbps Ethernet standards.

The third important contribution Xilinx makes is through the IP Integrator (IPI) tool associated with the Vivado® Design Suite. As illustrated in Figure 2, with the IPI, the task of linking IP blocks becomes similar to linking up chips on a printed-circuit board. It becomes particularly easy where (as with the OmniTek OSVP and DMA blocks) the interfaces on the IP blocks conform to the AMBA AXI4 interconnect protocols, which Xilinx has adopted as standard.

Even more power is available with the advent of Xilinx's new UltraScale™ (16-nm/20-nm) technology, which supports clock speeds up to many hundreds of gigabits per second and is being described as "ASIC-class." (For

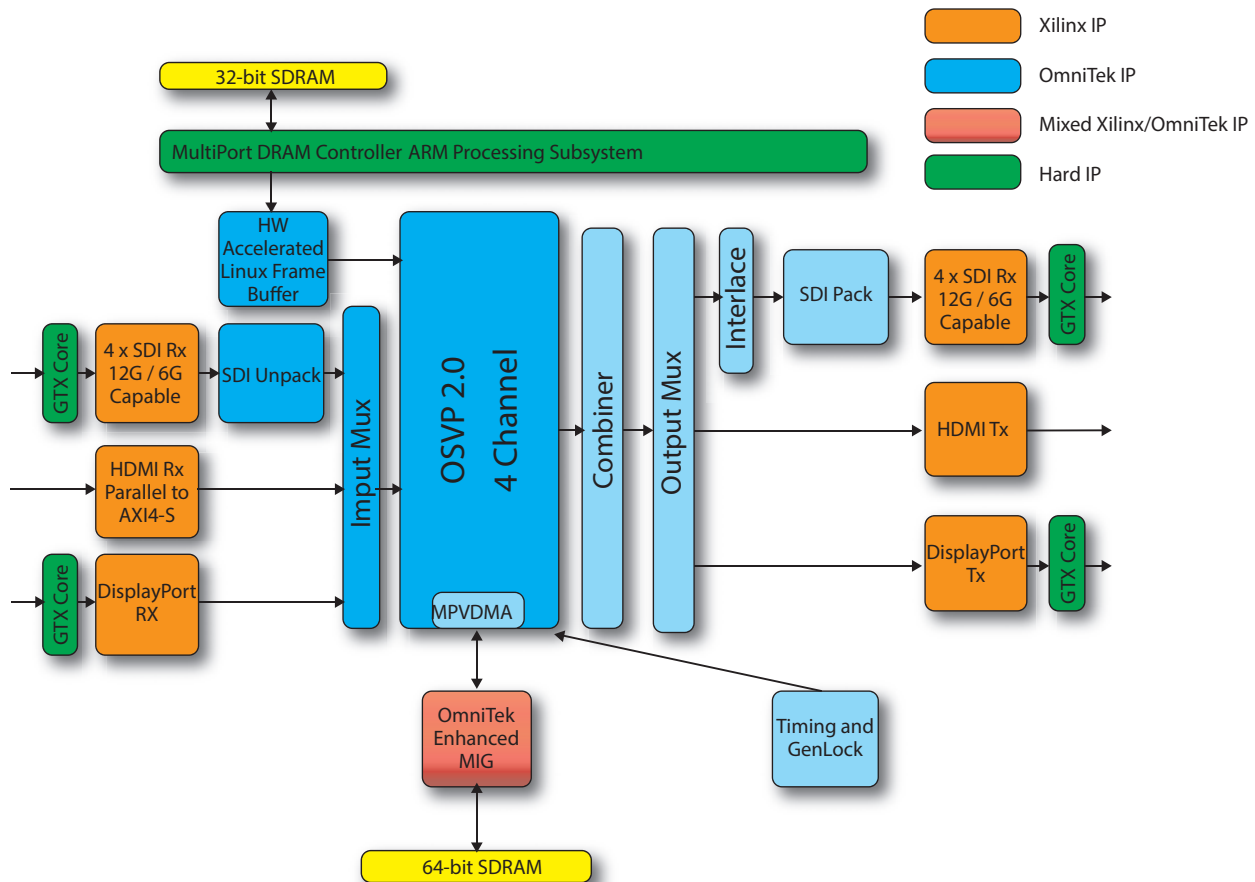


Figure 3 – Outline structure of the RTVE 3.1 Reference Design



Figure 4 – OmniTek's new Ultra 4K Tool Box, shown open to reveal the components upon which it is based

further information, see www.xilinx.com/products/technology/ultrascale.html). The UltraScale architecture enables the development of video systems not only at 4K but also beyond, with 8K on the horizon.

READY-MADE SYSTEMS TO ADAPT

While the building blocks offered by IP cores go a long way toward easing the task of creating video system designs, ready-made systems that you can adapt provide an even better starting point. For some time, Xilinx has been offering Real-Time Video Engine (RTVE) reference designs incorporating both Xilinx video and connectivity IP and IP blocks from OmniTek. These reference designs provide highly demonstrable, broadcast-quality video processing targeted to a wide range of video applications. The RTVE designs demonstrate both the functionality of these IP blocks and their easy interoperability, because they have all been designed to AXI4 interconnect standards.

Each new version of RTVE has extended the capabilities of the design by incorporating the latest IP blocks. The most recent version—RTVE 3.1—adds support for 4K video standards as defined by SMPTE 425-5:2014, DisplayPort 1.2, 6G-SDI and 12G-SDI. Figure 3 outlines the block diagram for this design.

The RTVE 3.1 design incorporates both of the OmniTek cores described above, along with the OmniTek interlacer, combiner and dedicated cross-point and some key components from Xilinx. It also comes with an API and an application to drive the RTVE engine from a Web-based interface. Both the firmware of the RTVE 3.1 design and the software of the application are available to customers in source form for use either as an illustration of how to go about designing a system using these tools or as the starting point for developing similar systems.

Also available to customers is the hardware platform for the RTVE 3.1, which comprises the OmniTek OZ745 Development Kit (based around a Xilinx

Zynq 7045 SoC) and an FMC expansion card. The FMC adds DisplayPort 1.2-compatible input and output ports, along with two SD/HD/3G/6G-SDI inputs and outputs. Together, those I/Os make it possible to extend the video standards supported to include 6G 4K and 12G 4K; 3G Level A and 3G Level B Square Division/Quad 4K; and 3G Level A and 3G Level B 2-Sample Interleave 4K.

The proof that these components can be used together to create a commercially viable system comes in the form of OmniTek's Ultra 4K Tool Box (Figure 4). The basic architecture of this 4K Tool Box is built around OmniTek's OZ745 Development Kit, FMC card and the firmware and associated application software of the RTVE 3.1. The 4K Tool Box not only provides up-, down- and cross-conversion of all video standards up to 4K60 and associated image correction, but also offers a wide range of displays including eye and jitter, gamut views and pixel data across all the many data streams that make up 4K images.

The Ultra 4K Tool Box is newly on the market but has already been purchased by a broad spectrum of customers, covering all areas of 4K processing from chip-set manufacture to test and measurement and broadcast. The uptake demonstrates the real interest in the new 4K standards across the video industry.

THE ULTIMATE ASSISTANCE

Along with these tools and IP, OmniTek also offers consultancy services to help customers get 4K designs up and running. The combination of leading-edge silicon technology and software tools from Xilinx and the video-processing and manufacturing expertise from OmniTek means that a designer of a video system starts within a complete development framework, with much easier integration capabilities and better support than they might have expected. The result is the muscle to get innovative and competitive products to market much more quickly. 🌟

FPGAs Aid Search for Dark Energy with CHIME Telescope

by Steve Leibson

Editor in Chief, Xcell Daily

Xilinx, Inc.

sleibso@xilinx.com

Canadian researchers are finding clues to the curious force known as dark energy by correlating signals received from across the universe.

A mysterious force is causing the universe to expand at an ever-increasing rate. Physicists don't really know what it is but they have named it: dark energy. There's no known way to directly detect dark energy experimentally. Evidence is strictly indirect. Observing and measuring periodic fluctuations in the distribution of baryonic matter, specifically neutral hydrogen, across hundreds of millions of light years may be among the best methods we have for studying dark energy. These variations, called baryonic acoustic oscillations (BAO), serve as a "standard ruler" for cosmological distance. A group of researchers based at universities in Canada are developing a radio telescope purpose-built for measuring BAO at the Dominion Radio Astrophysical Observatory in the heart of Canada's wine country on the south end of Okanagan Lake in British Columbia.

This telescope, called the Canadian Hydrogen Intensity Mapping Experiment (CHIME), reconstructs an image of the overhead sky without having to point at a specific direction. The telescope processes the signals received across a large array of radio antennas using a technique called interferometry. Historically, most interferometry telescopes have reconstructed images across a small region of the sky by mechanically steering several antennas, each in its own telescope dish, to point at that region. CHIME has no moving parts, so it may be thought of as a digital telescope. It reconstructs an image of the entire overhead sky by digitally processing the information its antennas have received as the Earth rotates through a 24-hour period. A Xilinx® Kintex-7 FPGA is a key component.

For CHIME to succeed, the team had to overcome sizable technological challenges similar to those encountered in commercial applications. For example, CHIME samples the voltage at each of its 2,560 antennas at 800 MHz and divides the samples into different frequency bands. Information from all

antennas moves over a gigantic Ethernet network to one processing location so that it can be used to reconstruct the sky image at that frequency. CHIME uses a custom-made network consisting primarily of high-speed serial transceivers connecting FPGAs point to point through low-loss, full-mesh custom backplanes. In total, CHIME combines 4,480 high-speed interconnects, each capable of operating at 10 Gbps, into what could be one of the world's largest purpose-built networks. CHIME's total information bandwidth is 8 Tbps.

Once the data for a particular observing frequency is brought together in one place, the information has to be correlated. Essentially, the data stream from each antenna needs to be multiplied with the data stream from every other antenna. This amounts to about 25,602 multiplications every 1.25 nanoseconds. Indeed, the "size" of a radio telescope correlator can be quantified as the square of the number of antennas times the frequency bandwidth, $NANT^2 \times BW$. Needless to say, the CHIME correlator is a monster.

These challenges are not unlike those encountered in advanced telecommunications hubs, where fast networking is needed to assemble data sent across fiber-optic trunk lines. Large numbers of rapid correlations are also used in financial market analysis, where correlating information from various indicators, with a range of time lags, provides valuable insight for fund managers.

Before examining the technology that has been developed to overcome these challenges, let's first take a moment to explore the science objectives of the project.

NASA'S EXPLANATION OF DARK ENERGY

According to NASA (<http://science.nasa.gov/astrophysics/focus-areas/what-is-dark-energy/>): "In the early 1990s, one thing was fairly certain about the expansion of the universe. It might have enough energy density to stop its expansion and recollapse,

it might have so little energy density that it would never stop expanding, but gravity was certain to slow the expansion as time went on. Granted, the slowing had not been observed, but, theoretically, the universe had to slow. The universe is full of matter and the attractive force of gravity pulls all matter together. Then came 1998 and the Hubble Space Telescope (HST) observations of very distant supernovae that showed that, a long time ago, the universe was actually expanding more slowly than it is today. So the expansion of the universe has not been slowing due to gravity, as everyone thought, it has been accelerating. No one expected this, no one knew how to explain it. But something was causing it.

“Eventually theorists came up with three sorts of explanations. Maybe it was a result of a long-discarded version of Einstein’s theory of gravity, one that contained what was called a ‘cosmological constant.’ Maybe there was some strange kind of energy-fluid that filled space. Maybe there is something wrong with Einstein’s theory of gravity and a new theory could include some kind of field that creates this cosmic acceleration. Theorists still don’t know what the correct explanation is, but they have given the solution a name. It is called dark energy.

“More is unknown than is known. We know how much dark energy there is because we know how it affects the universe’s expansion. Other than that, it is a complete mystery. But it is an important mystery. It turns out that roughly 68 percent of the universe is dark energy.”

Mapping matter density across the universe requires measuring the large-scale power spectrum of the 21-cm emission line from neutral hydrogen at a cosmological scale. Measuring the age of the universe when this light was emitted provides information on the cosmic expansion rate, which in turn is driven by dark energy.

Here’s how the Dark Energy Survey—a collaborative effort involving two dozen research institutions—describes BAO:

“Imagine dropping a pebble into a pond on a windless day. A circular wave travels outward on the surface. Now imagine the pond suddenly freezing, fixing these small ripples in the surface of the ice. In an analogous fashion, approximately 370,000 years after the Big Bang, electrons and protons combined to form neutral hydrogen, ‘freezing’ in place acoustic pressure waves that had been created when the universe first began to form structure. These pressure waves are called baryon acoustic oscillations and the distance they have traveled is known as the sound horizon. This distance is just the speed of sound times the age of the universe when they froze. Just as there is an increased air density in a normal sound wave, there is a slight increase in the chance of finding lumps of matter, and therefore galaxies, separated by the sound horizon distance. Today, this sound horizon distance is about 450 million light years, and it provides a standard ruler for cosmological distance measurements.”

(The Dark Energy Survey involves more than 120 collaborating scientists from 23 organizations who have banded together to survey a large portion of the sky visible from the southern hemisphere using the new Dark Energy Camera mounted on the Blanco 4-meter telescope at the Cerro Tololo Inter-American Observatory, perched high in the Chilean Andes.)

Measuring BAO across intergalactic distances coupled with red-shift data provides cosmologists with additional data about dark energy by mapping out the Hubble parameter—formerly known as the Hubble constant but no longer believed to be a constant over time. This information will help to explain how the universe’s expansion rate has changed over time, giving us more information about dark energy’s effects. By peering back to this key time when dark energy first began to overcome the pull of gravity, astronomers hope to obtain new insight that will provide hints about the theoretical nature of dark energy.

CHIME’S TECHNICAL DETAILS

CHIME is a stationary radio telescope array consisting of multiple adjacent parabolic cylinders; it depends upon the Earth’s rotation to scan the sky. CHIME Pathfinder is a smaller version of CHIME with 1/7 the collecting area—a pilot project—with 128 dual-polarization feed antennas and low-noise amplifiers (LNAs) spaced 30 cm apart along the focal lines of the telescope’s two parabolic cylinders. Briefly, signals from the feeds are bandpass-filtered to the 400- to 800-MHz band; the filtered analog signals are sent to 1.25-Gsample/second ADCs that operate at 800 Msamples/s. The sampled data is then fed to FPGAs that channelize the signals into 1,024 frequency bins. Each frequency bin is 390 kHz wide. The FPGAs then shuffle the channelized bins over a 10-Gbps full-mesh network to group like-frequency bins together. Grouped frequency bins are shipped off to a large bank of GPUs for real-time N^2 correlation and averaging. CHIME Pathfinder has already been built and tested. The full CHIME telescope is now under construction.

The total number of multiplications the CHIME telescope will perform will be $NANT^2 \times BW = 25602 \times 800 \text{ MHz} = 5242 \text{ teraMACs per second}$. CHIME Pathfinder, with just $NANT^2 \times BW = 2562 \times 800 \text{ MHz} = 52 \text{ teraMACs/s}$, will already be one of the world’s most powerful correlators. The full CHIME telescope correlator will be immense.

If the cosmological objectives of the CHIME experiment seem complex, the practical implementation of the experimental apparatus is significantly more straightforward, although the engineering of the equipment is state of the art.

IT STARTS WITH THE ANTENNAS

Both the CHIME and CHIME Pathfinder arrays consist of reflector antennas built from multiple parabolic cylinders. The reflective part of the antenna is made from galvanized steel mesh attached to an underlying structure that provides shape and support. Each CHIME Pathfinder parabola is 20 meters across and



Figure 1 – Walkways suspended above each reflector in the CHIME telescope carry the dual-polarization feeds and low-noise amplifiers.

37 meters long with a north-south orientation. The structure is not very different from that of any warehouse roof, except that the east-west cross-section is parabolic, the “roof” is made of wire mesh instead of sheet metal and it’s upside down, and there’s no floor.

The researchers chose wire mesh as a compromise between radio wave reflectivity and the ability to shed snow. (It’s in Canada, remember?) CHIME has no moving parts. Unlike a conventional radio telescope, it cannot be tipped over to shed accumulated snowfall. A coarse wire mesh allows snow to fall through, which avoids loss of observation time due to perturbed reflectivity. The CHIME Pathfinder has two such parabolic cylinders covering 1,500 m². The full CHIME telescope has five such cylinders but they’re 100 meters long instead of 37 meters. The full CHIME reflector covers 10,000 m².

A walkway suspended above each reflector and located at each parabolic cylinder’s focus line holds a linear series of polarized feed antennas and LNAs, with two LNAs per feed for the two polarized channels in each feed. Received signals are sent over coaxial cables to a shielded RF enclosure for filtering, fur-

ther amplification and processing. The CHIME Pathfinder has 64 dual-polarization feeds per parabolic cylinder—128 in total. The full CHIME telescope will have 1,280 dual-polarization feeds. The cylinders have an instantaneous field of view of approximately 100 degrees from north to south and 1 to 2 degrees from east to west. This narrow field of view sweeps the sky as the Earth turns.

Figure 1 shows the two CHIME Pathfinder reflectors and the focus-line walkway suspended above.

Each dual-polarization feed is made from printed-circuit boards in the shape of a cloverleaf. Tuned baluns combine differential signals from pairs of adjacent cloverleaf petals to form one single-ended output. The feed petals, balun stem and support base are all made from PCBs. Electrical losses in conventional circuit board materials are unacceptably lossy for astronomical instrumentation, so the group used Teflon-based PCB material for both the balun stem and support base.

EXOTIC ANALOG ELECTRONICS

The PCB feeds route the 400- to 800-MHz signal to the co-located LNAs, which have two gain stages. The first

gain stage employs an Avago ATF-54143 high-electron-mobility transistor (HEMT) followed by an Avago MGA-62563 RFIC. Each LNA drives 60 meters of coaxial cable, which terminates in a central shielded enclosure made from a sea container that holds the rest of the analog electronics including a custom Minicircuits 400- to 800-MHz bandpass filter and three more analog gain stages. Another 2 meters of coaxial cable deliver the filtered signals to ADC daughtercards plugged into digital back-end processing cards known as ICE motherboards. Figure 2 is a block diagram of the CHIME’s analog front end.

Each daughtercard holds two e2v EV8AQ160 quad 8-bit, 1.25-Gsample/s ADCs operating at 800 Msamples/s, making a total of 16 ADCs per ICE motherboard. The ICE motherboards are designed to be generic and they will be repurposed in the upgrade camera for the South Pole Telescope and for the Square Kilometer Array prototype. Figure 3 is a photo of the blue ICE motherboard with two red ADC daughtercards mounted on top.

The ADC daughtercards communicate with the ICE motherboard using LVDS signals and FMC-compliant high-pin-count connectors. The LVDS signals from the ADCs connect to a Xilinx Kintex-7 XC7K420T FPGA, which handles the data from all 16 ADCs on the two daughtercards. The ICE motherboard’s FPGA configuration was developed in VHDL using the Xilinx Vivado® Design Suite. The configuration is subdivided into modules that interconnect using the AXI4-Streaming bus protocol.

DATA SHUFFLING COURTESY OF FPGAS

The Kintex-7 FPGA on the ICE motherboard channelizes the input data from the ADCs into frequency bands using a custom polyphase filter bank followed by FFTs. The result is scaled to (4+4) bit complex values with saturation. Data streams from the 16 FPGA channelizers are then reordered by a crossbar module (also in the FPGA) that aligns the data

stream and then routes selected frequency bins to one of 16 output streams.

An FPGA-based data-shuffling module sends the reordered streams to each of 15 other ICE boards in a 9U crate called an ICEBOX. All 16 boards communicate over a mesh backplane using 19 of the Kintex-7 FPGA's high-speed GTX serdes transceivers operating at 15.5 Gbps. The FPGA can also receive trigger, synchronization and GPS-based time-stamp signals from the backplane. The ICE motherboard has two QSFP+ connectors that link to a GPU node and one SFP+ connector that connects to a control computer. Figure 4 shows a block diagram of the ICEBOX crate.

The FPGA's multigigabit transceivers communicate directly over the full mesh backplane creating a passive, low-cost, high-speed, data-shuffling network. The backplane is constructed of a low-loss material (Panasonic Megtron 6). Data sent over this network is en-

coded in 64B/66B format, scrambled to balance the DC content of the data and encapsulated in simple packets with a cyclic redundancy check (CRC) code to detect transmission errors. After the full data-shuffling transaction, each FPGA possesses a subset of 64 frequency bins from all 256 channelizers in the crate.

Consider the amount of traffic described in that last sentence. For the full-scale CHIME telescope: 2,560 channels x 8 bits/channel x 800 Msamples/s = 2 Tbytes/s.

That's a lot of data movement. According to Matt Dobbs, associate professor of physics and associate member of the Department of Electrical and Computer Engineering at McGill University and part of the CHIME collaboration, the large number of serial transceivers in each FPGA and their very low cost drove the design decision to implement the data-shuffle/corner-turn portion of the DSP algo-

rithms using the Kintex-7 FPGA's GTX transceivers and a full-mesh backplane. By contrast, he said, a commercial 10-Gbit Ethernet network solution would have been much more expensive and power hungry.

Each ICE board requires approximately 75 watts, so the entire ICEBOX needs about 1.2 kW plus some overhead for power-conversion efficiency. Figure 5 is a photo of a partially populated ICEBOX.

The 16 ICE board data streams pass through another crossbar that rearranges the data into eight output streams, each containing eight frequency bins from all of the ICE board channelizers. A custom array of eight 10-Gbps Ethernet UDP packet transmitters (designed to minimize FPGA resources by using a subset of the full UDP protocol) then sends these eight data streams to a GPU node through two QSFP+ optical connectors.

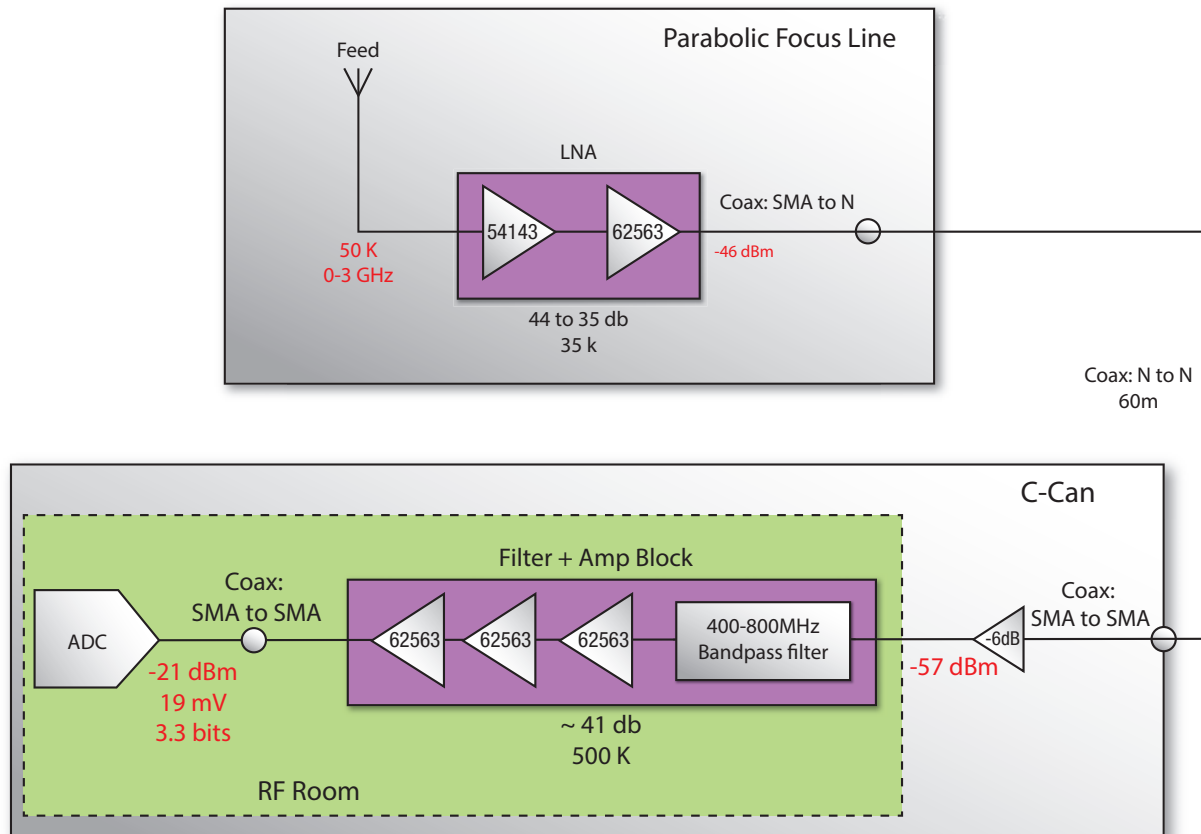


Figure 2 – Analog signals from the low-noise amplifiers travel over 60 meters of coax to an RF room inside a shielded, metal sea container.

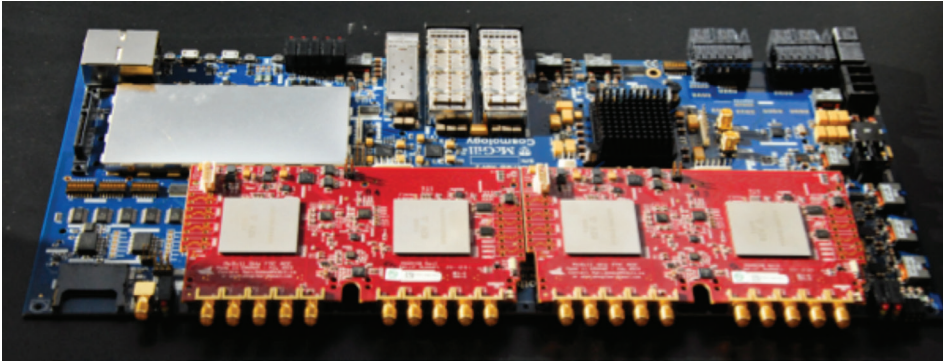


Figure 3 – The blue ICE motherboard with an onboard Kintex-7 FPGA supports 16 ADCs mounted on the two red daughtercards.

GPUS FOR CORRELATION

CHIME Pathfinder’s correlator consists of 16 identical GPU processing nodes. Each node processes 1/16 (25 MHz) of the full CHIME bandwidth. Each node is housed in a 4U rack-mount chassis and built primarily of high-end consumer-level components. Processing takes place in two AMD r9 280x GPUs and one r9 270x GPU. A pair of enterprise network interface cards (NICs)

receives a total of eight 10-Gbps network connections, streaming a total of 51.2 Gbps of radiometric data, along with associated headers and flags.

CHIME’s designers chose GPUs for this correlation task for ease of programming. The correlation operation takes place in a custom processing kernel written in the OpenCL language at the University of Toronto. However, energy efficiency and GPU cooling are



Figure 5 – This partially populated ICEBOX crate is part of the CHIME Pathfinder experiment.

significant concerns. Each GPU card requires 400 W. The CHIME Pathfinder needs 20 GPU boards with a power requirement of 8 kW. The full-scale CHIME needs 2,000 GPU cards and 200 kW of electricity, which could cost about \$150,000 per year. In addition, the CHIME design team tested several brands and types of GPUs for thermal performance and is presently evaluating direct-to-chip water cooling as a solution to the thermal issues associated with GPUs.

According to Dobbs, the team chose to use GPUs for spatial correlation to get maximum flexibility and faster implementation time at the expense of much higher power consumption. “Once we work all of this out, we might very well move the spatial correlation onto FPGAs,” he said. “We’re in new territory in terms of the image reconstruction algorithms, so flexibility is the key for now.”

For further information on the CHIME program, see “Canadian Hydrogen Intensity Mapping Experiment (CHIME) Pathfinder,” <http://arxiv.org/abs/1406.2288>. A NASA site offers further insight into dark energy: “Dark Energy, Dark Matter,” <http://science.nasa.gov/astrophysics/focus-areas/what-is-dark-energy/>.

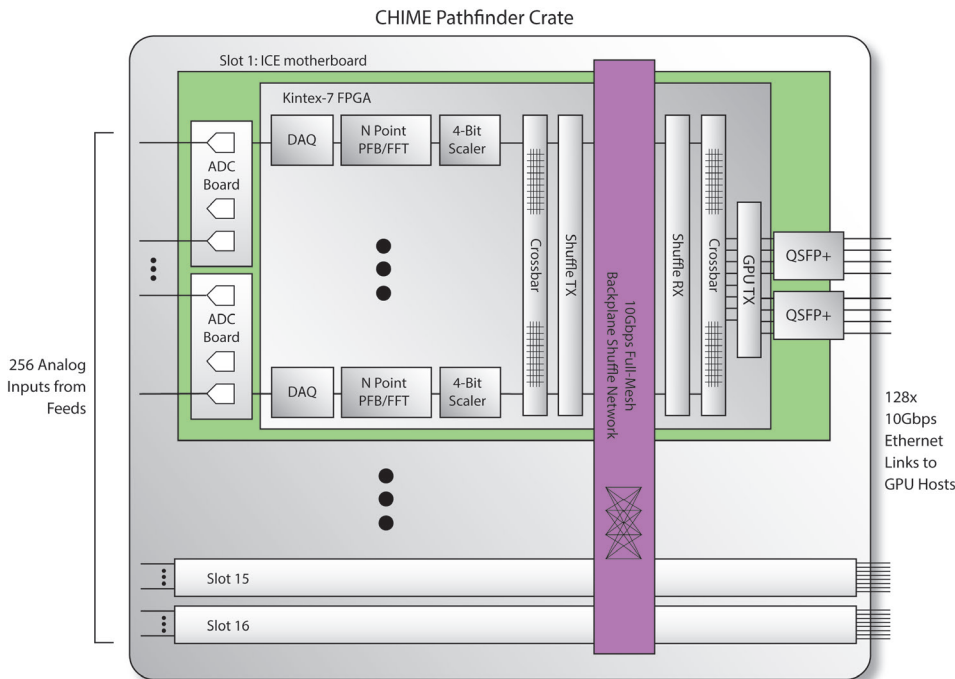


Figure 4 – An ICEBOX crate holds as many as 19 ICE motherboards.

Efficient Parallel Real-Time Upsampling with Xilinx FPGAs

by **William D. Richard**

Associate Professor

Washington University, St. Louis

wdr@wustl.edu

Here's a way to upsample by a factor of four in real time using a Virtex-6 device and the free WebPACK tools.

Upsampling is required in many signal-processing applications. The easiest way, conceptually, to upsample a vector of data by a factor of M is to zero-pad the discrete Fourier transform (DFT) [1] of the data vector with $(M-1)$ times as many zeros as there are actual frequency components and then transform the zero-padded vector back into the time domain [1, 2]. This approach is computationally expensive, however, and does not lend itself to efficient implementation inside FPGAs. The efficient, parallel, real-time upsampling circuit presented here produces M upsampled values per ADC clock, where M is the desired upsampling factor. Our Xilinx® Virtex®-6 XC6VLX75T FPGA implementation, which upsamples by a factor of $M=4$, serves as an example of the more general technique.

The general concept on which our parallel upsampling technique is based has been termed “windowed Sinc interpolation” by some authors, and it is described in several excellent papers in the literature [3, 4].

For the purposes of illustration, consider the example 16-MHz analog signal shown in Figure 1. This signal has the form:

$$f(t) = \cos(2\pi ft) * e^{(t * t)/\text{constant}}$$

Equation 1

If the signal shown in Figure 1 is sampled/quantized at 80 MHz using a 12-bit ADC driven to 97.7 percent of its full-scale input range, only five samples are taken per signal period, resulting in the sample data sequence shown in Figure 2. Upsampling this example data sequence by a factor of four, to an effective sample rate of 320 MHz, would provide 20 samples per signal period. While you can use the method described here to upsample by larger factors, we will upsample by $M=4$ for purposes of illustration.

Of course, it's possible to generate an (admittedly poorly) upsampled data vector with the desired number of samples by simply inserting $(M-1)$ zeros in between each actual sample value in the data sequence the ADC produces. This “zero insertion step” corresponds to a replication of the spectrum of the original signal in the frequency domain. By low-pass filtering the resulting “zero-padded” time-domain signal so as to eliminate

the “replications” of the desired spectrum in the frequency domain, you can obtain an upsampled data vector.

FIR FILTER DESIGN

An ideal (brick wall) low-pass filter in the frequency domain corresponds to convolution in the time domain with an infinite-extent Sinc function. Therefore, let’s run our zero-padded time-domain signal through a symmetric, low-pass FIR filter running at M times the ADC clock rate and performing an approximation to the desired convolution operation, using the topology shown in Figure 3 for an example 31-tap FIR filter. In this way, we can produce our upsampled data vector in real time. In Figure 3, R1, R2, ..., R31 represent registers clocked at M times the ADC clock rate, and C0, C1, ..., C15 represent the coefficients of the FIR filter.

It is important to note that most of the registers in the FIR filter shown in Figure 3 will contain zero, and not actual sample data, during any particular clock interval. For M=4, as an example, when R1 contains actual sample data, R2, R3 and R4 will contain zero. When R1 contains actual sample data, so will R5, R9, R13, R17, R21, R25 and R29, and the remaining registers will contain zero. During the next clock interval, R2, R6, R10, R14, R18, R22, R26 and R30 will contain actual sample data.

Since (M-1) out of every M samples moving through the FIR filter shown in Figure 3 are zero, you can collapse the filter and produce M outputs in parallel as shown in Figure 4 for the M=4 case when using a 31-tap FIR filter. With this implementation, the parallel FIR filter runs at the base ADC clock rate, not at M times the ADC clock rate.

You can specify the windowed Sinc function coefficients, Cw(n), shown in Figure 4 so as to minimize the number of

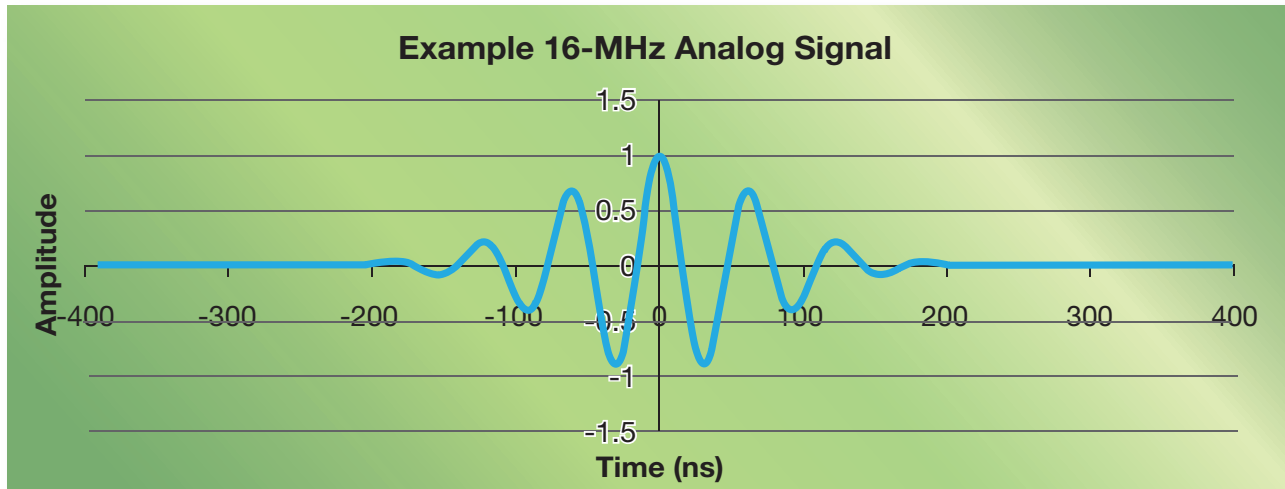


Figure 1 – This example 16-MHz signal illustrates the upsampling process.



Figure 2 – Here is the example sample data sequence that results from sampling the example analog signal of Figure 1 at 80 MHz, or five times per period, using a 12-bit ADC driven to 97.7 percent of its full-scale input range.

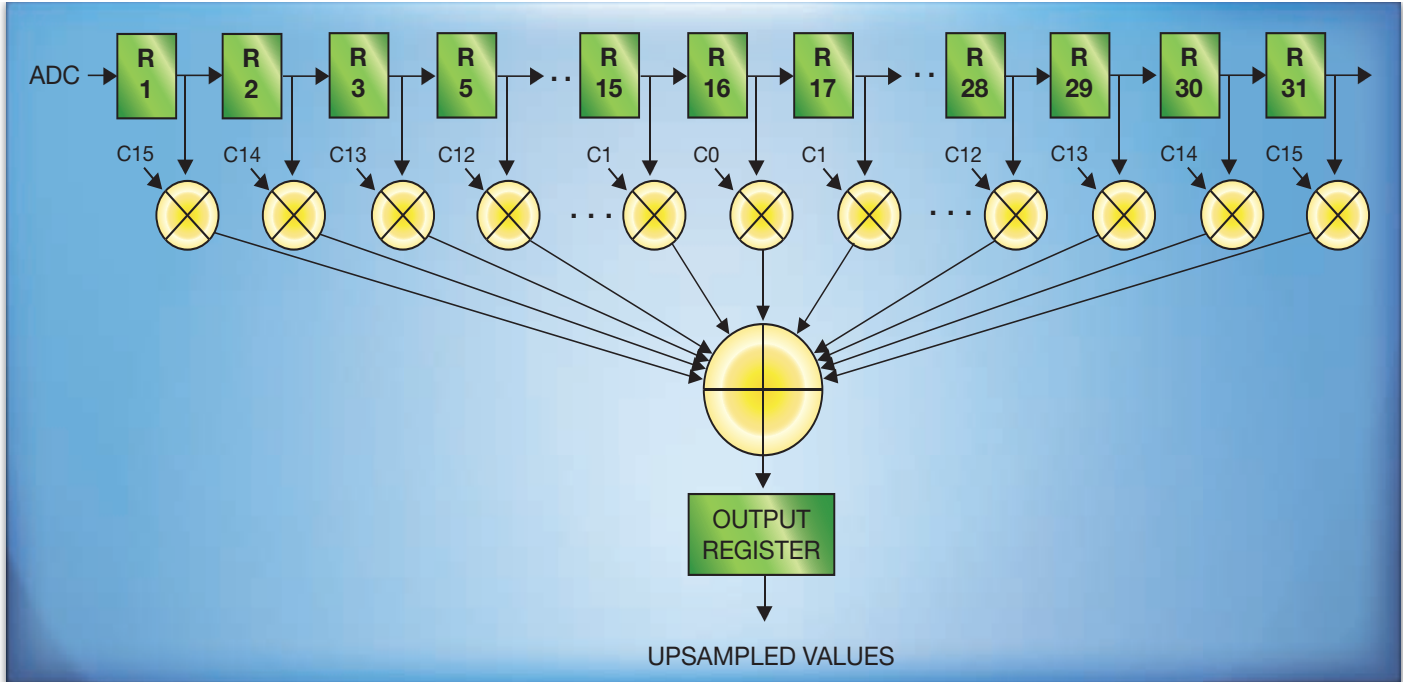


Figure 3 – You could use a 31-tap FIR filter to generate one upsampled data value per clock period if clocked at M-times the base ADC clock rate with zero insertion.

multipliers required to implement the FIR filter. For a T-tap, low-pass FIR filter, the optimal coefficients are given by:

$$C(n) = \text{Sinc}[(n * \pi) / M], n = 0 \text{ to } (T-1)/2.$$

Equation 2

Here, the Hanning window coefficients are given by:

$$H(n) = [1 - \text{COS}(2 * \pi * (n + ((T-1)/2)) / (T-1))] / 2, n = 0 \text{ to } (T-1)/2.$$

Equation 3

The windowed Sinc function coefficients, $C_w(n)$, are then found by multiplying corresponding values of $C(n)$ and $H(n)$, i.e.,

$$C_w(n) = C(n) * H(n), n = 0 \text{ to } (T-1)/2.$$

Equation 4

For $M=4$, when the coefficients for a 31-tap FIR filter are calculated as described above, $C_0 = 1.0$ and $C_4 = C_8 = C_{12} = C_{15} = 0$, the nine multipliers associated with these coefficients in Figure 4 are not needed. In addition, by recognizing that each coefficient is used twice to generate $UPSAMPLED\ VALUE(1)$, you can “fold” the implementation—add R1 to R8 before multiplying, for example—and eliminate four additional multipliers. The end result is a design that requires a total of only 18 multipliers to produce four upsampled values

per clock period. It is important to note that, as a result of the filter design technique described above, each original sample value exits the parallel filter unmodified.

We used the synthesizable VHDL [5] model in Figure 5 to evaluate the performance of the circuit shown in Figure 4. This VHDL implementation assumes 12-bit sample data, as an Analog Devices AD9670 eight-channel ultrasound front-end integrated circuit [6] might produce. Filter coefficients are represented as 25-bit fixed-point constants to match the size of the multipliers integrated onto the FPGA die. Input samples from the ADC are clocked into a register (R1 in Figure 4) connected to input pins, and upsampled output values use registers tied to output pins. Registers R2 to R8 are internal to the chip. Registers R1 to R8 are intentionally 15 bits wide so that the synthesized logic has the headroom to perform the calculation. The design checks for overflow or underflow and clamps the results so they remain within the valid range.

NO NEED FOR PIPELINING

Figure 6 plots the upsampled data sequence that results when the VHDL model is simulated using the ISim simulator in version 14.7 of the free Xilinx WebPACK™ tools [8] and fed the sampled/quantized 12-bit data sequence of Figure 2. Each of the original 12-bit samples is unchanged, as explained above, and three new samples have been inserted on the original waveform between each actual sample.

The worst-case error in the computed (upsampled) values from the ideal values in the original analog signal is 0.464 percent of the full-scale range, while the average error is 0.070

The placed-and-routed design used 19 DSP48E1 blocks but fewer than 1 percent of the Virtex-6's slices. It ran at 107 MHz without pipelining.

percent of the full-scale range. Of course, there is as much as 1/2 LSB of error in the sampled/quantized 12-bit source vector data values (or 0.012 percent of the full-scale range) due to the initial quantization step.

We implemented the upsampler in a Xilinx XC6VLX75T-3FF484 Virtex-6 FPGA [7] using version 14.7 WebPACK tools. The placed-and-routed design used 19 of the 288

DSP48E1 blocks in the part but fewer than 1 percent of the slices. The final upsampling circuit was able to run at 107 MHz. It was not necessary to pipeline the filter to achieve this performance. We also developed a pipelined version that ran at over 217 MHz.

Even though the XC6VLX75T-3FF484 is the smallest member of the Xilinx Virtex-6 family, it contains 288 DSP48E1

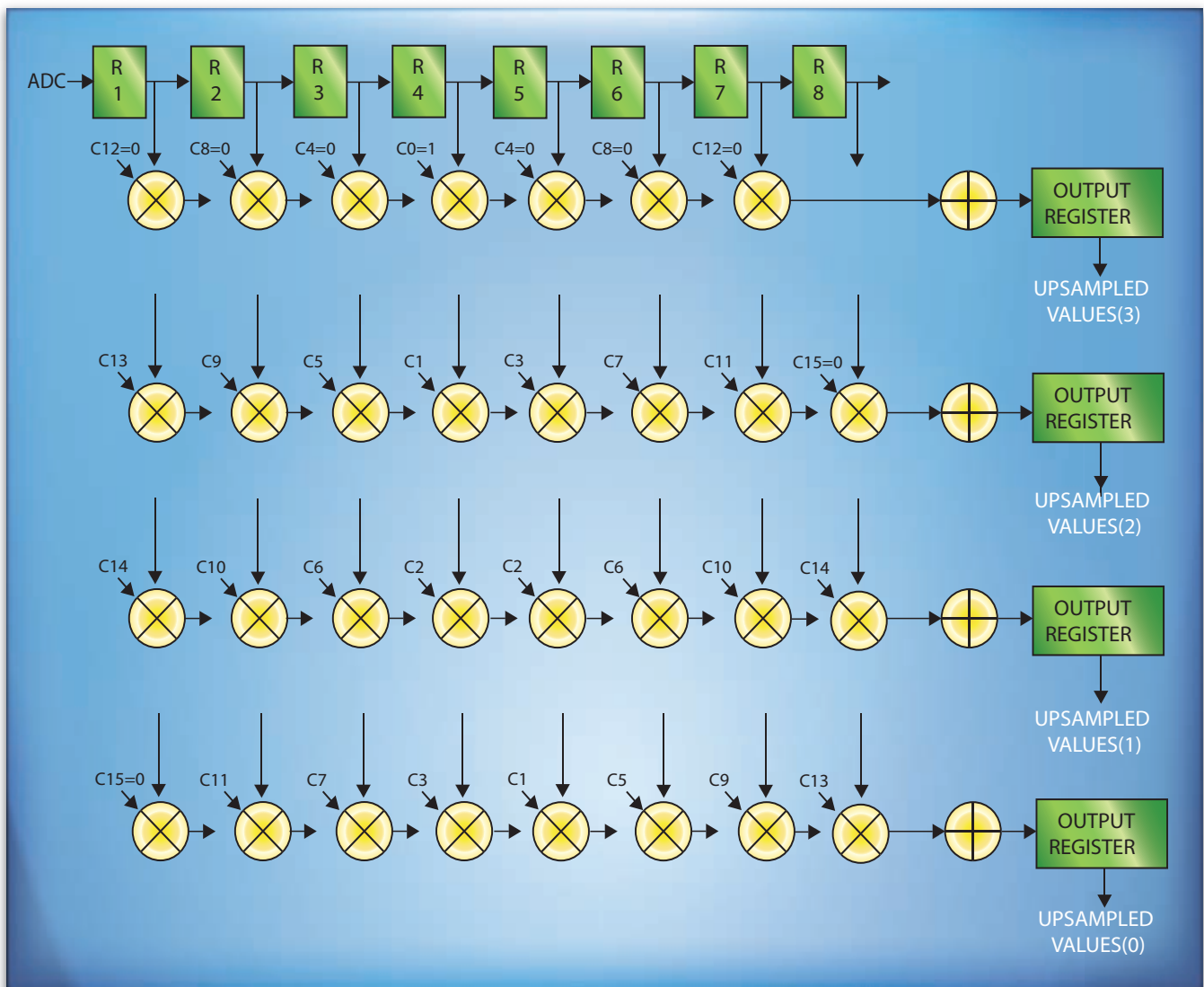


Figure 4 – By observing that only one out of every four registers in Figure 3 holds non-zero data during any given clock period, it is possible to collapse the filter and produce four outputs in parallel while running the filter at the base ADC clock rate.

```

LIBRARY IEEE ;
USE IEEE.STD_LOGIC_1164.ALL ;
USE IEEE.STD_LOGIC_ARITH.ALL ;
USE IEEE.STD_LOGIC_UNSIGNED.ALL ;

ENTITY upsample IS
  PORT (clk          : IN STD_LOGIC ;
        r_ext       : IN STD_LOGIC_VECTOR(11 DOWNTO 0) ;
        d0,d1,d2,d3 : OUT STD_LOGIC_VECTOR(11 DOWNTO 0)) ;
END upsample ;

ARCHITECTURE mine OF upsample IS
  SIGNAL r1,r2,r3,r4,r5,r6,r7,r8          : STD_LOGIC_VECTOR(14 DOWNTO 0) ;
  SIGNAL d0int,d1int,d2int                : STD_LOGIC_VECTOR(39 DOWNTO 0) ;
  CONSTANT c1 : STD_LOGIC_VECTOR(24 DOWNTO 0) := "1110001111110110011100110" ;
  CONSTANT c2 : STD_LOGIC_VECTOR(24 DOWNTO 0) := "1001101111101110000000011" ;
  CONSTANT c3 : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0100010101111101100111001" ;
  CONSTANT c5 : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0010001010010010011110000" ;
  CONSTANT c6 : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0010001110001110010111001" ;
  CONSTANT c7 : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0001001000101111000010111" ;
  CONSTANT c9 : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0000100011011001000000101" ;
  CONSTANT c10 : STD_LOGIC_VECTOR(24 DOWNTO 0) := "00001000000100110000100111" ;
  CONSTANT c11 : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0000001101110111011000010" ;
  CONSTANT c13 : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0000000011000100001100100" ;
  CONSTANT c14 : STD_LOGIC_VECTOR(24 DOWNTO 0) := "0000000001000001000111111" ;

BEGIN

  flops:PROCESS(clk)
  BEGIN
    IF (clk = '1' AND clk'EVENT) THEN
      r1 <= "000" & r_ext ;
      r2 <= r1 ;
      r3 <= r2 ;
      r4 <= r3 ;
      r5 <= r4 ;
      r6 <= r5 ;
      r7 <= r6 ;
      r8 <= r7 ;
      IF d0int(39) = '1' THEN
        d0 <= "0000000000000" ;
      ELSIF d0int(38) = '1' OR d0int(37) = '1' THEN
        d0 <= "1111111111111" ;
      ELSE
        d0 <= d0int(36 DOWNTO 25) ;
      END IF ;
      IF d1int(39) = '1' THEN
        d1 <= "0000000000000" ;
      ELSIF d1int(38) = '1' OR d1int(37) = '1' THEN
        d1 <= "1111111111111" ;
      ELSE
        d1 <= d1int(36 DOWNTO 25) ;
      END IF ;
      IF d2int(39) = '1' THEN
        d2 <= "0000000000000" ;
      ELSIF d2int(38) = '1' OR d2int(37) = '1' THEN
        d2 <= "1111111111111" ;
      ELSE
        d2 <= d2int(36 DOWNTO 25) ;
      END IF ;
      d3 <= r4(11 DOWNTO 0) ;
    END IF ;
  END PROCESS ;

  d0int <= r2*c11 - r3*c7 + r4*c3 + r5*c1 - r6*c5 + r7*c9 - r8*c13 ;
  d1int <= (r2+r7)*c10 - (r1+r8)*c14 - (r3+r6)*c6 + (r4+r5)*c2 ;
  d2int <= r2*c9 - r1*c13 - r3*c5 + r4*c1 + r5*c3 - r6*c7 + r7*c11 ;

END mine ;

```

Figure 5 – The VHDL source uses a single process and 25-bit fixed-point coefficients to implement the filter topology of Figure 4.

This straightforward FIR filter design methodology eliminates the need for sophisticated filter design tools while providing excellent results.

blocks with 25x18-bit multipliers integrated onto the die, or enough to theoretically implement 15 parallel upsampling FIR filters of the type shown in Figure 4. We have constructed a prototype annular-array ultrasound system that uses eight copies of the upsampler running at 80 MHz in an XC6VLX75T FPGA to upsample data from an eight-channel Analog Devices AD9670 ultrasound front-end chip prior to beam forming. In this system, the upsampler works as predicted by simulation and enables real-time beam forming using data upsampled to 320 MHz while running at the base AD9670 ADC clock rate of 80 MHz.

The largest Xilinx Virtex-6 FPGA, the XC6VVSX475T, contains 2,016 25x18-bit multipliers, allowing it to theoretically implement 106 upsampling filters of the type shown in Figure 4 in a single chip.

It is possible to upsample by a factor of $M=4$ in real time using an FIR filter implemented in a Xilinx XC6VLX75T-3FF484 FPGA that runs at 107 MHz when the filter is designed using the efficient, parallel topology presented here. The original data samples pass through the filter unmodified, and $(M-1) = 3$ upsampled values are produced in parallel. This straightforward FIR filter design methodology eliminates the need for sophisticated filter design tools while providing excellent results. Straightforward exten-

sion of the ideas presented here could be used to upsample by larger factors or to reduce the error in the computed upsampled values by using an FIR filter with more taps. 🌈

REFERENCES

1. A.V. Oppenheim, R.W. Schaffer, *Discrete-Time Signal Processing* (Prentice Hall, Englewood Cliffs, NJ, 1989)
2. H. Stark, J.W. Woods, I. Paul, "An investigation of computerized tomography by direct Fourier inversion and optimum interpolation," *IEEE Transactions Biomedical Engineering* 28, 496-505 (1981)
3. R.W. Schaffer, L.R. Rabiner, "A digital signal processing approach to interpolation," *Proceedings of the IEEE* 61, 692-702 (1973)
4. R. Crochiere, L.R. Rabiner, *Multirate Digital Signal Processing*, (Prentice-Hall, Englewood Cliffs, NJ, 1983)
5. D. Pellerin, D. Taylor, *VHDL Made Easy!* (Prentice-Hall, Upper Saddle River, NJ, 1997)
6. Analog Devices AD9670 Octal Ultrasound AFE with Digital Demodulator Datasheet Rev Sp0 (Analog Devices, 2013)
7. Virtex-6 Family Overview DS150 (v2.3) (Xilinx, Inc., 2011)
8. ISE In-Depth Tutorial UG695 (v13.1) (Xilinx, Inc., 2011)

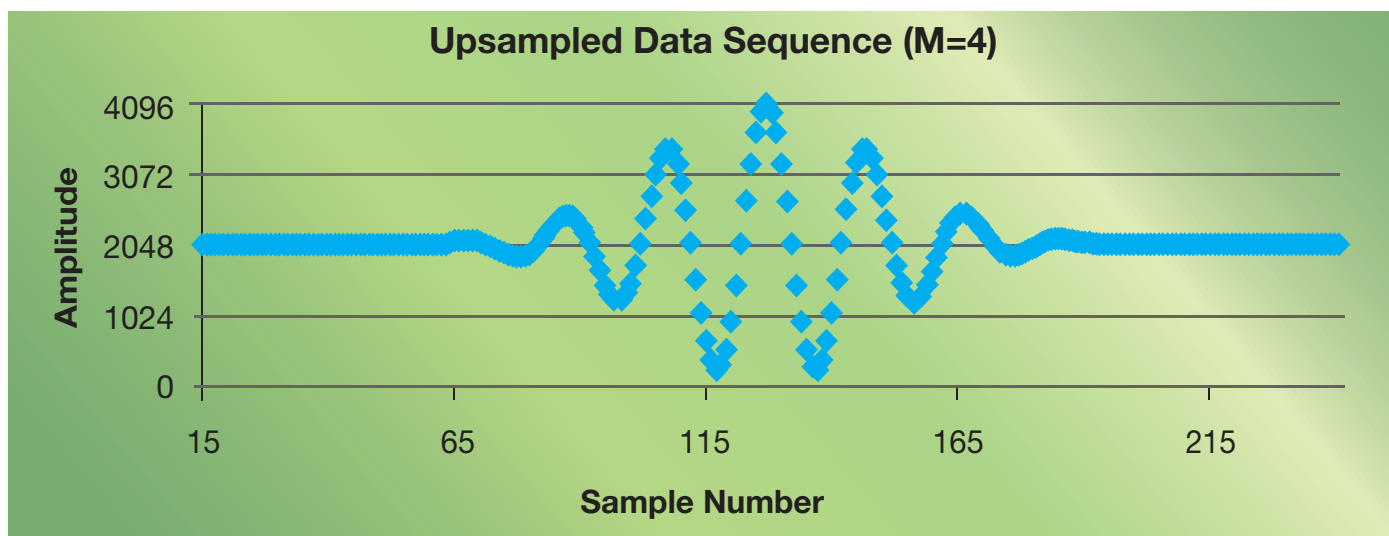
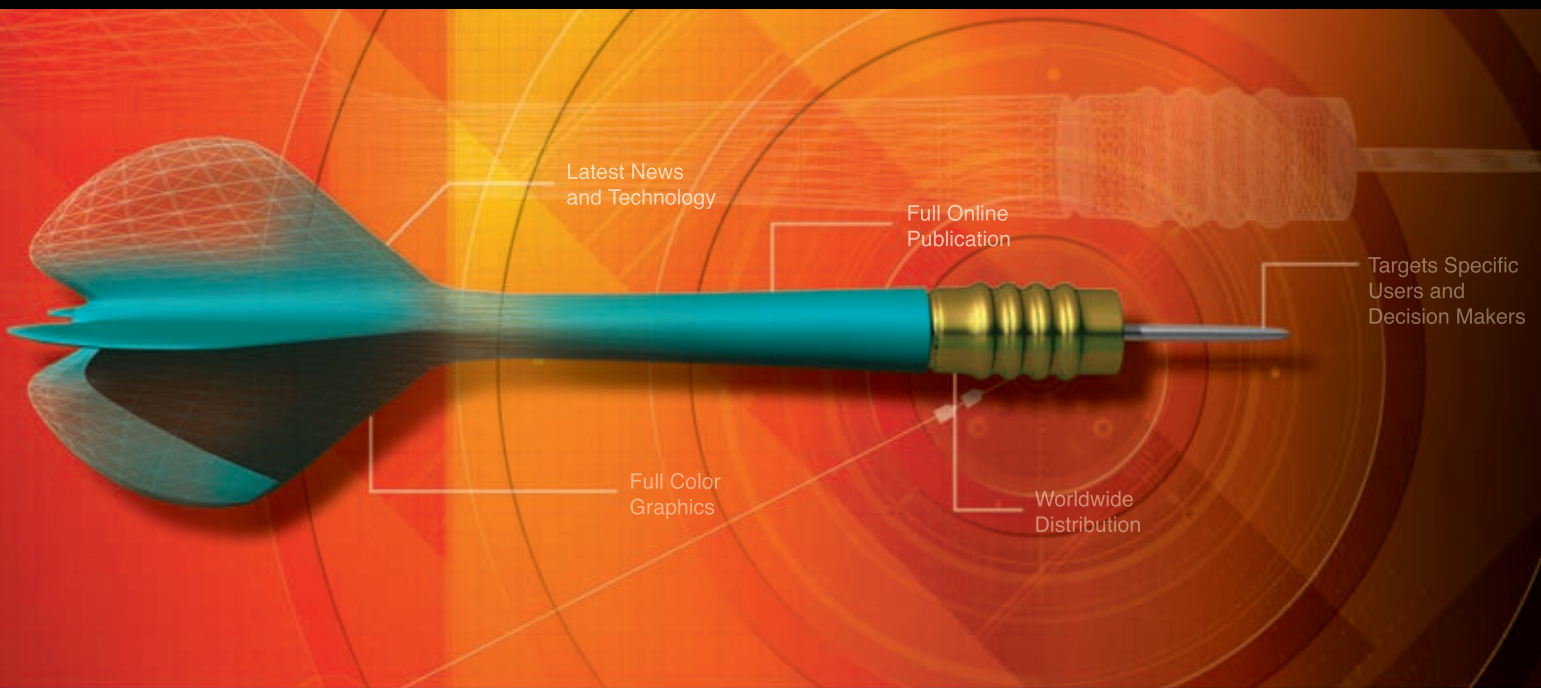


Figure 6 – This graph shows the upsampled data sequence produced by the VHDL model.

GET ON TARGET



LET XCELL PUBLICATIONS HELP YOU GET YOUR MESSAGE OUT TO THOUSANDS OF PROGRAMMABLE LOGIC USERS.

Hit your marketing target by advertising your product or service in the Xilinx *Xcell Journal*, you'll reach thousands of engineers, designers, and engineering managers worldwide!

The Xilinx *Xcell Journal* is an award-winning publication, dedicated specifically to helping programmable logic users – and it works.

We offer affordable advertising rates and a variety of advertisement sizes to meet any budget!

Call today :
(800) 493-5551
or e-mail us at
xcelladsales@aol.com

Join the other leaders in our industry
and advertise in the *Xcell Journal*.



www.xilinx.com/xcell/



A Framework for FPGA Design Planning

by **Jeffrey Lin**

Senior Manager, Global Communications Services Group
Xilinx, Inc.
jeffrey.lin@xilinx.com

This time-tested
FPGA development
framework is your
route to flawless
project execution.



The capabilities and performance of modern FPGAs have long since reached the point where these devices now sit in a central location within systems and serve as a primary data-processing engine for many products.

Given the important role FPGAs play in so many applications, it is more important than ever to approach FPGA design with a formal and methodical development process. The purpose of such a process is to avoid costly and time-consuming design deficiencies discovered late in the development cycle that can have a potentially disastrous impact on schedules, cost and quality.

The Xilinx Global Communications Services Group has long been using a time-proven design framework to develop and deliver turnkey FPGA designs to Xilinx customers for products ranging from medical-imaging processing engines to self-learning network switch engines. It is a framework we have developed and evolved over the course of designing, developing and delivering hundreds of FPGA designs.

The framework we use covers everything from system architecture considerations to FPGA development and test planning. Let's take a closer look at this framework, with a focus on the FPGA hardware, in hopes that other engineering teams might find it a useful tool for tackling complex FPGA design projects of their own.

FRAMEWORK OVERVIEW

The framework is an iterative, top-down methodology for designing hardware in FPGAs. First, the planning starts at the system architecture level to determine FPGA functionality. Then we progressively refine the features implemented in the FPGA using the known capabilities and performance of the FPGA device.

In addition, the implementation of large FPGA designs necessitates well-defined development, simulation and verification plans. The framework aids us in crafting those plans as well. In short, the framework can be summarized as shown in the flow chart in Figure 1. The focus for this discussion is on the Planning and Documentation portion (top section).

SYSTEM ARCHITECTURE

In the context of this discussion, system architecture refers to the division of functionality among system software and hardware. In particular, the focus is on the subdivision of the hardware functionality into FPGA and other microchip components, with the assumption that product-level requirements are already defined. For example, a marketing or product-definition organization may have already weighed in with product requirements.

In the system architecture phase, the idea is to bring clarity to how these product requirements are to be realized in a physical product. For the FPGA, the primary decision is what features and functions should be implemented in the programmable device and furthermore, what is reasonably achievable in an FPGA.

By defining the high-level requirements of the FPGA up front, you can avoid costly design and requirements changes before you have proceeded too far down the development process. At this early stage, the definition of the system architecture will guide you to several important decisions that affect both development time and product cost.

At this level of discussion, only the general outline of the FPGA features is required. The detailed features and implementation requirements will be defined in a later stage during the FPGA requirements definition. Participants in this discussion should include someone familiar with the system-level requirements, someone with system-level architecture design knowledge and someone familiar with the features and capabilities of FPGAs.

In the context of the FPGA, there are 10 important questions to answer.

1. What is the feature list to be implemented in the FPGA?
2. What are the technical trade-offs for implementing features in the FPGA vs. using non-FPGA components?
3. What is the design effort/cost to implement in FPGA vs. non-FPGA components?
4. What custom features or processing are required?
5. What does the flexibility of the FPGA bring to the functions?
6. What future-proofing risk mitigation should you consider?
7. Can features be consolidated within the FPGA from multiple non-FPGA components?
8. Based on the design features to be implemented, what are the FPGA device choices?
9. Can the features be implemented in an FPGA?
10. What non-FPGA devices are required and how do these devices interface to the FPGAs?

FPGA ARCHITECTURE

The FPGA architecture is the micro-architecture and chip-level data-flow design at the physical level on the FPGA device. Your team should design this architecture concurrently with the system-level architecture for device sizing, selection and feasibility.

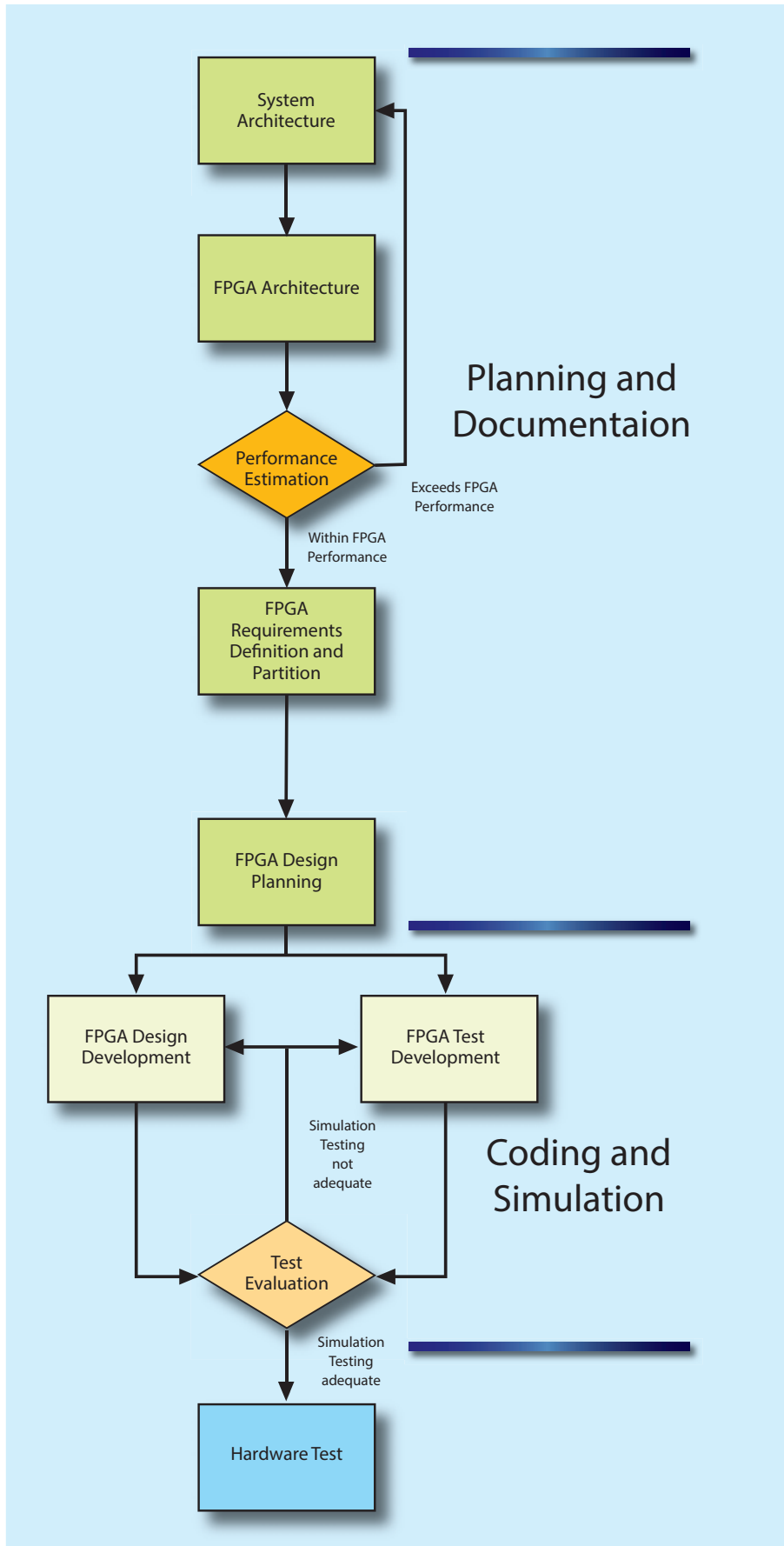


Figure 1 – The FPGA development framework

The purpose of defining the FPGA architecture is to ensure the system architecture requirements can be implemented with accurate, realistic and achievable design requirements in the FPGA.

This level of discussion requires intimate knowledge of the features and capabilities of the FPGA fabric and resources. It should therefore be undertaken with participants who are experienced FPGA designers. At this stage you must consider the decisions for performance targets of the FPGA, potential risk areas, as well as available FPGA fabric resource utilization.

During the FPGA architecture definition, it's possible you might find that the system-level requirements and architecture are unachievable or high-risk for implementation in the FPGA. In this case, you must reevaluate and update the system architecture to create a set of high-level requirements that are achievable in the FPGA.

Ask yourself what existing IP can be used and what must be created. You also need to examine I/O requirements and how to map the clock domains and features onto the FPGA clock resources. Other key questions include how to place the GT resources on the FPGA, whether cross-SLR data flow is accounted for in SSI devices and whether target clock frequencies are realistic for the design functionality. Finally, you must also assess whether your target performance goals are realistic for the selected FPGA.

FPGA REQUIREMENTS DEFINITION AND PARTITION

The FPGA requirements definition-and-partition phase is closely related to the system and FPGA architecture and is driven by the decisions made at those stages. The FPGA requirements definition refers to the detailed requirements to be implemented in the FPGA and will serve as the definitive feature list for the design and test-engineering teams to achieve, design and implement. This job differs from the system and FPGA architecture requirements in that the FPGA

requirements are precise. The list defines the minute details of the FPGA rather than the division of functionality between the different components of the system, or the data flow through the FPGA.

The purpose of this stage is to clearly define exactly what the FPGA engineering team should implement and test. Here, you will translate the high-level system and FPGA architecture requirements into exact requirements for implementation. The benefits of doing this are twofold. First, discretely defining the FPGA requirements will highlight any limitations of the system and FPGA architecture and any previously unconsidered or unforeseen conditions. Second, this step will pave the way for smooth execution of the FPGA design development and test.

To properly outline FPGA requirements, you must have a clear definition that is concise and easily distilled into discrete requirements. We recommend labeling or numbering the requirements and defining them as a basic description easily determined to be achieved or not, rather than as a high-level, ambiguous requirement. You can use any industry-standard or proprietary format so long as it is clear and concise.

Avoid vague or loosely defined terms such as “fast” or “small.” Instead, stick with specific targets such as “400 MHz” or “4.2k flip-flops.” The goal of this definition is to ensure the document can be distributed to development engineering teams with no prior knowledge of the system or FPGA architecture to implement without the need for constant clarification. Ask yourself whether each requirement is clear, concise and unambiguous and whether it contains all the necessary information to avoid a need for constant clarification. Also, do the requirements include pin-out and I/O definition? Are all high-level requirements decomposed into fundamental design elements? Can a design team not involved with the early system architecture definition use the requirements to develop an FPGA? And finally, can a test and verification team use the document to develop

test platforms and tests to validate each requirement with a definitive pass or fail?

FPGA DESIGN PLANNING

This stage of the framework is planning for the actual development of the FPGA hardware and to ensure features and development are completed in step with other parts of the overall product development.

The goal at this point is to properly map the current state of the system-level and FPGA-level requirements and architecture into a development plan. After going through the previously listed planning stages, there are typically two scenarios development teams will now encounter.

Scenario 1 is a case where the system and FPGA architecture and requirements are well understood and are finely detailed, resulting in an FPGA design development phase (that is, HDL coding) and test development phase (simulation testbench) that will proceed smoothly with minimal requirements changes.

Scenario 2 is a case where the system architecture and FPGA requirements are still in flux. Designs in this category will encounter multiple changes and revisions during the design and test development phases of the cycle.

While everyone shoots for Scenario 1, more often than not, people will find themselves in Scenario 2—clearly, a more difficult situation to manage.

The overall goal of design planning should be to reach Scenario 1 at this stage of the development cycle. In Scenario 1, the development of the FPGA is straightforward and is a matter of scheduling the implementation and test of design features.

In Scenario 2, the most important management task is to ensure you have a well-understood process in place to evaluate and determine what changes should be implemented, and the impact of each change to the overall development schedule. There are several program-management philosophies and techniques that you can employ here. The most important point is that such evaluation and impact assessments are made.

In terms of FPGA-specific planning and development, one of the benefits of FPGAs is the ability to revise and download a hardware platform to a prototype PCB multiple times. Design teams should take full advantage of this capability. Therefore, the recommended development plan is one that will incrementally add features to a working design. The idea is to start with a basic design that can enable the major communication interfaces to function without all of the requirements implemented.

The benefits of such an approach are twofold. First, you will ensure there is always a working design that you can use to debug a PCB and the larger system. And second, debugging the actual FPGA design will be easier, since you can check newly added features to ensure they do not interfere or break the currently operational design.

In parallel with the FPGA design development, it is vitally important to have a sound simulation environment plan for the resulting FPGA design. Investing in the development of a robust simulation environment will not only result in a reduction of design bugs, it can also significantly reduce lab debug time by allowing you to replicate real-life data flow so as to reproduce error conditions in simulation and quickly isolate and determine the root causes.

Development of a robust test and simulation environment is just as complex as development of the FPGA design itself and should be planned for and given consideration as such.

The Xilinx Global Communications Services Group has honed our FPGA development framework through consistent application over hundreds of FPGA designs. The result is a practical design approach that delivers results, is easily understood and is widely applicable to many different development tasks. The benefits of using this framework in developing your next FPGA design will be accurate overall development schedules, fast hardware bring-up and, ultimately, an on-time product launch. 🌟

Faster Design Entry with Vivado IP Integrator and Xilinx IP

by **Duncan Cockburn**

Staff Design Engineer
Xilinx, Inc.
duncan.cockburn@xilinx.com

Here's how to optimize Xilinx cores for use with Vivado IPI in a CPRI remote radio head design.

M

Modern FPGA-based designs use an increasing amount of intellectual property (IP), both in variety and number of instances. The Vivado® Design Suite's IP Integrator (IPI) tool and Xilinx® communications IP are making it easier to quickly connect these IP blocks together.

To illustrate the power of the IPI approach, let us consider the example of a wireless remote radio head (RRH). Situated near the antenna, the RRHs form part of a cellular communications network. They are normally connected with optical fiber upstream to a baseband transceiver station and optionally downstream to further RRHs, thus implementing a multihop topology (Figure 1).

The Common Public Radio Interface (CPRI) protocol is widely used in linking these RRHs together. Let's create an example design with one uplink CPRI port and three downlink CPRI ports, and connect them. We can accomplish

the majority of this job with IPI. The result will form a major component in the overall design. We will use a Kintex®-7 device, which is an excellent fit in this application due to its low power, low cost and high performance. The GTX transceivers in -2 speed-grade All Programmable Kintex FPGAs and Zynq®-7000 SoCs make it possible to use the 9.8-Gbps CPRI line rate.

Figure 2 shows what we will create within IPI. We can create the block design and instance the required IP from the IP catalog. The CPRI cores are available in the standard Xilinx IP catalog and have been optimized for the sharing of resources where possible and for ease of use in IPI. The switches are custom IP.

IP CORE RESOURCE SHARING

One of the challenges customers encounter when using multiple instances of IP is how to share resources efficiently. A number of communications IP cores support the "shared logic" feature. In the case of the CPRI core, we can configure the IP with sharable logic resources inside the core or we can omit these shared resources. If they are included in the core, they will provide the necessary outputs to let us connect them to the cores that have excluded the logic.

Users with specialized requirements may wish to exclude this logic on all their cores and implement their own.

In our design, we have configured CPRI cores to run at 9.8 Gbps. At this line rate, it is necessary to use an LC-tank-based oscillator for the transceiver clock. Transceivers in the Kintex-7 device are arranged in quads, with each transceiver quad consisting of four transceiver channels and one LC-tank-based quad phase-locked loop (QPLL). It is necessary for all the cores to share the QPLL and the clock generated by the uplink clocking. Figure 3 shows the QPLL and clock output ports on the uplink core customized with shared logic connected to the appropriate input ports on a downlink CPRI core that has been customized with it excluded.

ROUTING DATA BETWEEN CPRI CORES

We have also instantiated the IQ switch and the Ethernet switch to allow data to be routed between the cores.

Control and management data in the CPRI network is transmitted via an Ethernet subchannel. The Ethernet switch in the system makes it possible to issue firmware updates or commands remotely and transmit them to any node. The IP was designed to use as few logic resources as possible, since a fully featured Ethernet switch in this situation is not necessary.

The IQ switch provides the ability to route any IQ sample between CPRI cores with deterministic latency. An important feature for multihop radio systems is the ability to accurately measure the link delay, and the CPRI standard defines a method to facilitate this measurement.

CONNECTING INTERFACES WITH IPI

IPI bus interfaces map a defined set of logical ports to particular physical ports on the IP. If we use interfaces wherever possible, we move from connecting many signals to connecting a few interfaces. Common bus interfaces on IP are those that conform to the ARM® AXI specification, such as AXI4-Lite and AXI4-Stream. This elevation of abstraction makes design entry

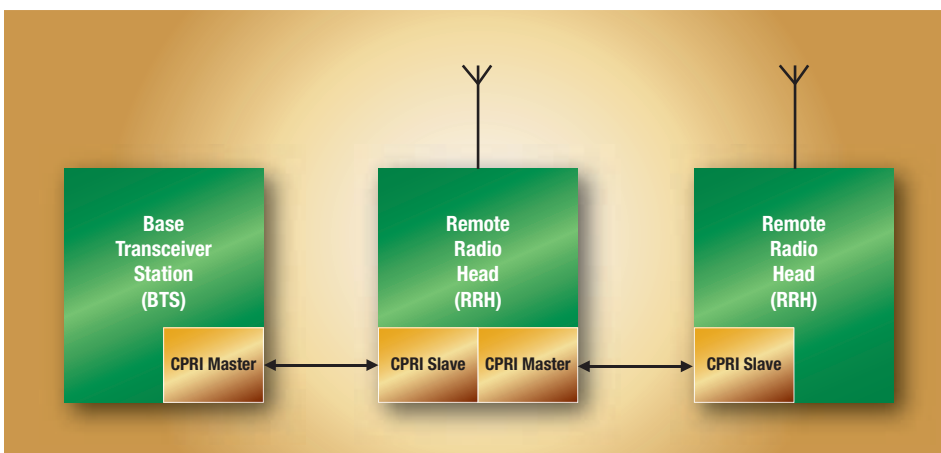


Figure 1 – Diagram of a multihop topology

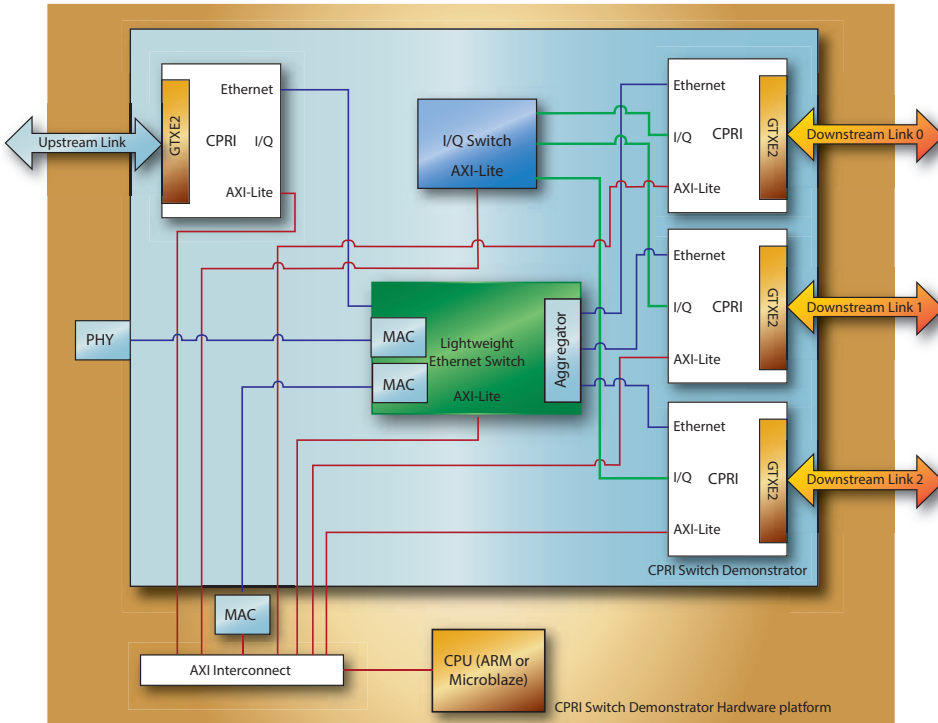


Figure 2 – CPRI switch hardware platform

easier and faster, and also allows you to take advantage of design rule checks for the interface. Vivado IP Packer will allow you to use your own IP within IP Integrator and to take advantage of interfaces in your own design.

IPI makes it easy to connect interfaces together. Simply click on the interface and IPI will indicate what it can connect to. Drag the connection line to the desired end point and the connection will be made. This technique allows you to connect many signals with just a couple of clicks.

Figure 4 shows the Ethernet switch providing a number of AXI4-Stream interfaces, two GMII interfaces and an AXI4-Lite interface. The streaming interfaces allow direct connection to the CPRI cores and this removes the need for internal buffering on the CPRI core. The GMII interfaces allow connection to an Ethernet PHY, which could be useful

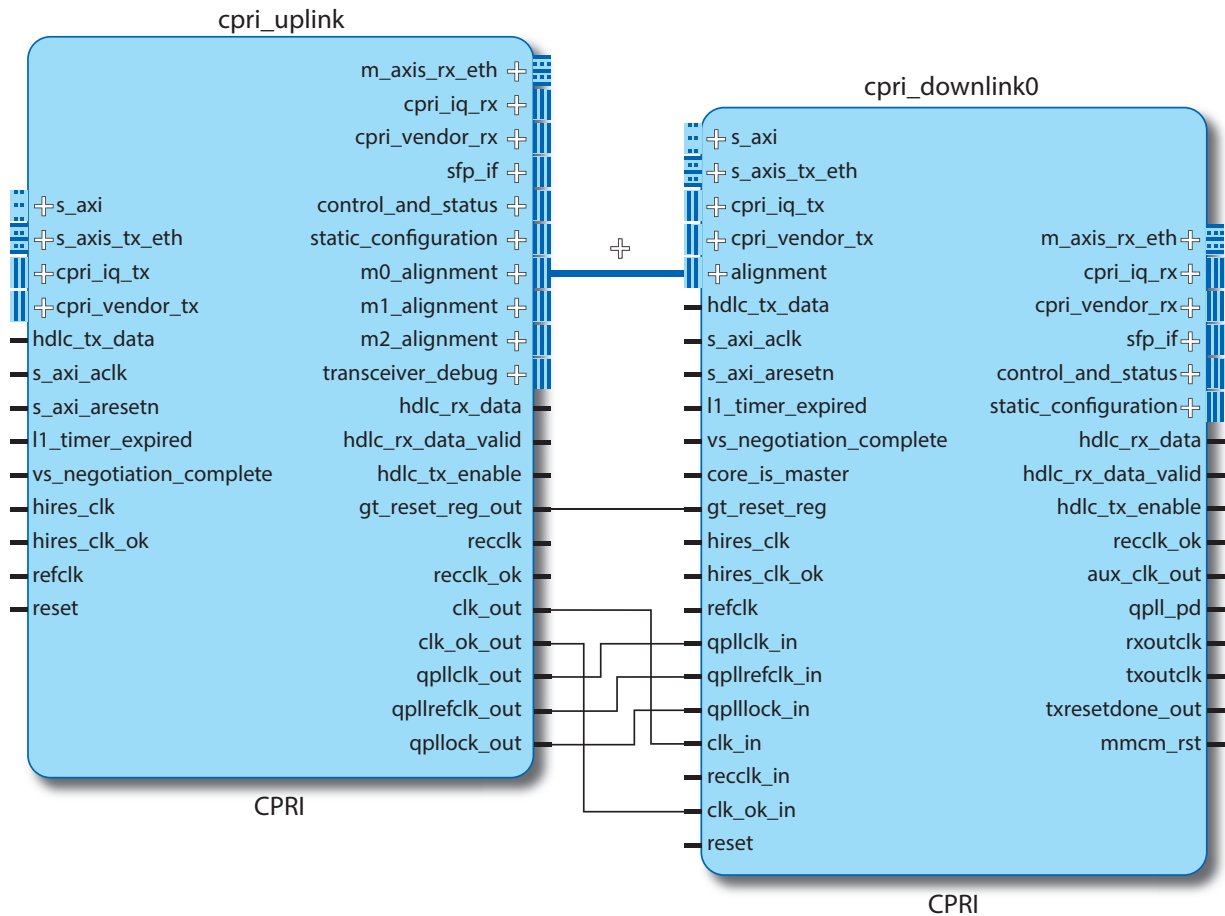


Figure 3 – Shared logic connections for a QPLL

for an engineer in the field debugging a network issue. The AXI4-Lite management interface provides access to the address table mapping and other configuration options such as the address table aging interval.

Continuing in this fashion, we can build up our system, connecting the interfaces within IPI. You have the flexibility to use whatever entry method works best for you. In addition to using the GUI to link interfaces, you can also opt to directly issue commands through the Tcl console or source them from a script. Every time you do something in the GUI, the resulting command will be echoed back.

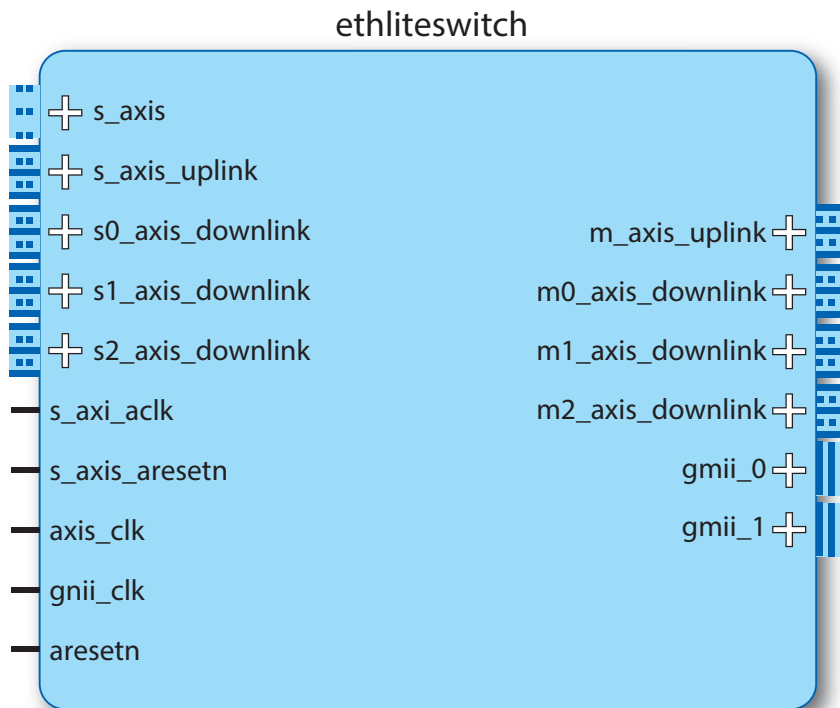
You can also export the entire design when you have finished creating it with the command "write_bd_tcl." This command will create a Tcl file that can be sourced to create the entire block design from scratch and can be easily used as part of a scripted build flow. All of the IP in the design provides an AXI4-Lite man-

agement interface to allow the cores to connect to a host processor. Intelligence built into IPI allows connection automation. With this mechanism, IPI will recognize that the AXI4-Lite interface on our IP will connect to the AXI bus interconnect and automatically configure the appropriate address ranges and connect the bus for us. You can then connect this bus to the host processor with the aid of IPI. The host processor in our case is a MicroBlaze™, but if using a Zynq SoC series device it could be easily changed to leverage the ARM CPU.

FURTHER GAINS COMING

Vivado IP Integrator capabilities are growing rapidly and with that growth, further gains will be achieved. With the right IP, we can put together whole sub-systems quickly and reap the rewards.

For more information on the CPRI, Ethernet switch or IQ switch IP, contact Per-minder Tumber at Xilinx Wireless Communications (*permind@xilinx.com*).



CPRI 6 Port Lightweight Ethernet Switch

Figure 4 – Ethernet switch symbol with interfaces

FPGA

Boards & Modules

EFM-02

FPGA module with USB 3.0 interface. Ideal for Custom Cameras & ImageProcessing.



- ▶ Xilinx™ Spartan-6 FPGA XC6SLX45(150)-3FGG484I
- ▶ USB 3.0 Superspeed interface Cypress™ FX-3 controller
- ▶ On-board memory 2 Gb DDR2 SDRAM 64 Mb Dual SPI flash
- ▶ Samtec™ Q-strip connectors 191 (95 differential) user IO

EFM-01

Low-cost FPGA module for general applications.



- ▶ Xilinx™ Spartan-3E FPGA XC3S500E-4CPG132C
- ▶ USB 2.0 Highspeed interface Cypress™ FX-2 controller
- ▶ On-board memory 4 Mb SPI flash
- ▶ Standard 0.1" pin header 50 user IO

CESYS
Hardware • Software • HDL-Design
www.cesys.com

Getting the Most out of Your PicoBlaze Microcontroller

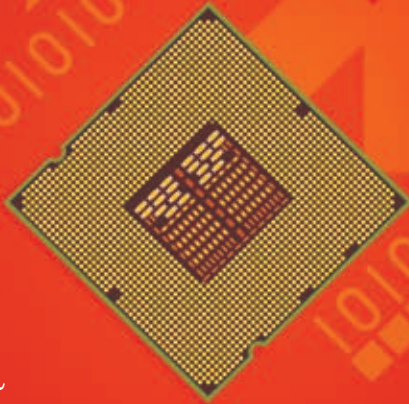
by **Adam P. Taylor**

Head of Engineering – Systems

e2v

aptaylor@theiet.org

Many FPGA applications can benefit from using a simple soft-core processor to ease the generation of sequential control structure.



The PicoBlaze™ is a compact 8-bit soft-core microcontroller that FPGA engineers instantiate within their selected Xilinx® FPGA. Once implemented, this core is completely contained within the FPGA fabric using only logic slices and Block RAMs; it requires no external volatile or nonvolatile memory.

Thanks to its small implementation footprint, it is possible for an FPGA to contain multiple PicoBlaze instantiations, with each instantiation used to implement control structures typically created by state machines. The result is a reduction in the development time along with a standardized approach to control structure generation. Thanks to the underlying high performance of the Xilinx FPGA fabric, PicoBlaze instantiations are often capable of outperforming many discrete 8-bit microcontrollers.

Let's take a look at how we can best utilize this handy device within our designs.

PICOBLAZE ARCHITECTURE

Before we can use the core, it is first a good idea to understand a little about its architecture. PicoBlaze is a very sim-

ple 8-bit microcontroller that is based around a RISC architecture (see Figure 1). The controller has a 12-bit address port, which means it can address as many as 4,096 memory locations. Each address location contains an 18-bit instruction that defines the operation the core must perform. Inputs and outputs to and from the core are possible via two 8-bit ports (one input, one output). The controller also provides an 8-bit identification port, allowing for up to 256 peripherals to be read from or written to. There is also a size-selectable scratchpad that can be 64, 128 or 256 bytes. As with all micros, PicoBlaze contains an arithmetic logic unit and support for one interrupt. These capabilities mean the controller offers many advantages to the FPGA design engineer.

One of the most important aspects of PicoBlaze is its highly deterministic nature, which means all instructions require two clock cycles for execution and interrupts are serviced with four clock cycles maximum. (You can find more detailed information on the PicoBlaze architecture within the Xilinx user guide that comes with your download.)

WHY USE PICOBLAZE?

FPGA applications usually require a combination of parallel and sequential operations, with data flow being predominantly parallel and control structures predominantly implemented as sequential structures, for example state machines (see *Xcell Journal* issue 81, "[How to Implement State Machines in Your FPGA](#)"). However, complex control structures if implemented as a state machine can become unwieldy, increasing the verification time and making modifications later in the development cycle more difficult. Complicated state machines also take more time to develop and if several are required, this time can be considerable.

You can also use PicoBlaze for serial communication control over RS232, I2C and SPI. In fact, anything that you would use a typical 8-bit micro for can be implemented within a PicoBlaze, with the upside of higher performance. Engineers have used PicoBlaze to implement PID controllers in control systems. They have used it with I2C, SPI or parallel DACs to create reference waveforms that range from simple square, sawtooth and triangular to more com-

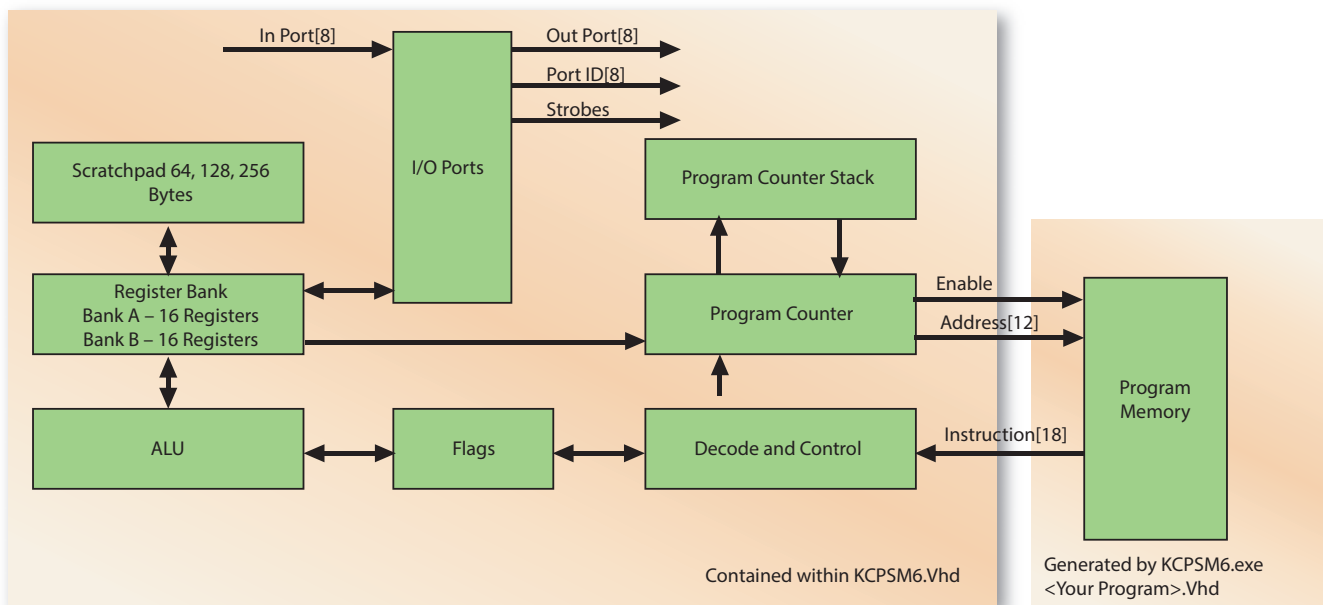


Figure 1 – PicoBlaze architecture, with processor in left box and memory at right

plicated sine/cosine waveforms (using the shift-and-add CORDIC algorithm).

Instantiating a PicoBlaze microcontroller within your FPGA to implement these sequential functions can result in a faster development time and allow for simplified modifications later in the development life cycle. And of course, being a soft core, PicoBlaze also helps address obsolescence issues and it encourages design reuse as ASM modules are developed.

FIRST PICOBLAZE INSTANTIATION

You can quickly implement a PicoBlaze within your design by following a few initial steps. First, make sure you have the latest version of the microcontroller for the device you are targeting. Xilinx makes these available from the [PicoBlaze Lounge](#), offering versions that support the latest 7 series devices to those that work with older Spartan®-3 and Virtex®-4 devices.

Having downloaded the correct version of the processor, extract those files into your working directory and ensure that you read the “readme” file, paying close attention to the setting of the PATH and Xilinx environment variables as required. Within your working directory, you will now notice the following files or applications, along with the usual readme and license files and user guides.

- **KCPSM6.Vhd:** This is the actual source code for the PicoBlaze.
- **KCPSM6.exe.** This is the assembler program that you will use to generate the machine code and memory files required.
- **ROM_Form.vhd.** The assembler executable uses this file to generate the VHDL file your created program will reside in.
- **KCPSM6_design_template vhd.** This is a template instantiation of a PicoBlaze processor.
- **All_kcpsm6_syntax.psm.** This file is a definition of all assembler commands and syntax.

```

NAMEREG s0,led ;rename S0 register to led
;As 8 bit processor we need four delay loops 256 * 256 * 256 *
256 = 4294967296
CONSTANT max1, 80 ;set delay
CONSTANT max2, 84 ;set delay
CONSTANT max3, 1e ;set delay
CONSTANT max4, 00 ;set delay
main: LOAD led, 00; load the led output register with 00
flash: XOR led, FF; xor the value in led register with FF i.e.
toggle
        OUTPUT led,01; output led register with port ID of 1
        CALL delay_init; start delay
        JUMP flash; loop back to beginning
delay_init: LOAD s4, max4;
            LOAD s3, max3;
            LOAD s2, max2;
            LOAD s1, max1;
delay_loop: SUB s1, 1'd; subtract 1 decimal from s1
            SUBCY s2, 0'd; carry subtraction
            SUBCY s3, 0'd; carry subtraction
            SUBCY s4, 0'd; carry subtraction
            JUMP NZ, delay_loop;
            RETURN

```

Figure 2 – Snippet of assembler code for a program to flash LEDs

```

kcpsm6.exe
KCPSM6 Assembler v2.63
Ken Chapman - Xilinx Ltd - 20th December 2013

Enter name of PSM file: test.psm

Reading top level PSM file...
C:\hdl_projects\picoblaze\test.psm

A total of 21 lines of PSM code have been read

Checking line labels
Checking CONSTANT directives
Checking STRING directives
Checking TABLE directives
Checking instructions

Writing formatted PSM file...
C:\hdl_projects\picoblaze\test.fnt

Expanding text strings
Expanding tables
Resolving addresses and Assembling Instructions
Last occupied address: 00E hex
Nominal program memory size: 1K (1024) address(9:0)
Occupied memory locations: 15
Assembly completed successfully

Writing LOG file...
C:\hdl_projects\picoblaze\test.log
Writing HEX file...
C:\hdl_projects\picoblaze\test.hex
Writing VHDL file...
C:\hdl_projects\picoblaze\test.vhd

KCPSM6 Options.....
R - Repeat assembly with 'test.psm'
N - Assemble new file.
Q - Quit

```

Figure 3 – Using the KCPSM6 assembler to generate your memory files

At the very simplest level, you need declare only two components within your design.

For our example design, the final step is to create a new project in the ISE® Design Suite within which we can instantiate the PicoBlaze and its program memory if you are not adding them to an existing project.

Once we have completed the above steps, we are ready to start creating a PicoBlaze processor within our application. At the very simplest level, you need declare only two components within your design: the processor itself and the program memory as shown in Figure 1 (processor is within the left box and memory the right box, to provide context). Of course, if you have more than one instantiation you will have a number of memory components, each containing a different program. However, the first thing we need to do is understand the development flow of a typical project.

DEVELOPMENT FLOW

Creating your first PicoBlaze instantiation is simple. The first step is to create a blank text file using an editor like Notepad++. This file should have the file extension .PSM—for instance, test.psm. You program the microcontroller using the PicoBlaze assembler. Xilinx provides detailed information on this syntax in the file All_kcpsm6_syntax.psm, which comes with your download. However, it is easy to understand and learn this syntax. Figure 2 is an example of an assembler code snippet, which is a simple program to flash LEDs at 2-Hz frequency with a 40-MHz clock.

Once you are happy with your assembler program, the next stage is to run this program through the assembler executable that came with your download. Doing so will generate a

memory file (VHDL for use within your FPGA), a log file and a hex file whose use we will look at later. Figure 3 shows the assembler process having been run for the code snippet above. Having run the assembler, you are now in a position to instantiate the PicoBlaze within your FPGA.

You are now in possession of the two VHDL files required: KCPSM6.vhd and the VHDL file created by the assembler program containing your application (in this case, test.vhd). The second stage is to declare the two components (KCPSM6 and Memory) within your VHDL design and instantiate them as shown in Figure 4. This simple VHDL example can be seen in the code snippet in Figure 5, which implements a PicoBlaze that will flash LEDs on an LX9 Spartan® development board.

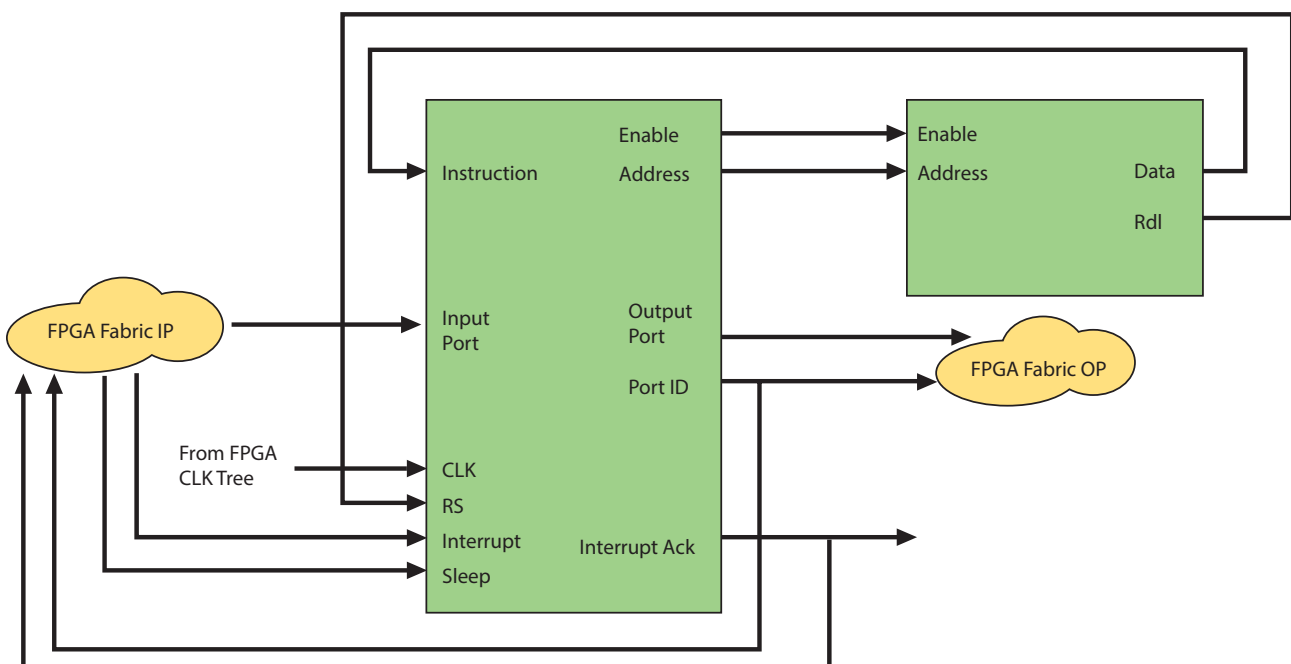


Figure 4 – PicoBlaze context diagram

SIMULATION AND VERIFICATION

Once you have instantiated the design files within your application, you will of course want to verify the performance of the system or module within a simulation environment before you progress to synthesis and implementation. As PicoBlaze uses logic slices and Block RAMs, it can be simulated very simply in programs like Mentor Graphics' ModelSim or Xilinx's ISim in ISE (or Xsim in the Vivado® Design Suite if that is where you are implementing your PicoBlaze).

Since the Block RAM contains the instructions for your program, simulation is simple. Essentially, all you need to provide is a clock and other inputs and outputs as required by the instantiation, Figure 6 shows ISim results for a PicoBlaze simulation and shows the two clock cycles between the loading of instructions.

UPDATING YOUR PROGRAM

One of the great benefits of having the PicoBlaze contained within the FPGA (and bit file) and is that following configuration of the FPGA the core will start executing the program within its RAM. However, in some cases you may need to modify the program the core is executing. While you could rerun the implementation stage including an updated memory file, depending upon the complexity of the remaining design this may take considerable time, especially if you are only trying out possibilities in the lab. It is therefore possible to update the program memory the core uses to modify the program and try it out before you rerun the implementation stage using the JTAG loader program also provided with your download.

The initial step in using the JTAG loader is to enable it within your design setting. Use the generic `C_JTAG_LOADER_ENABLE : integer := 1` within your instantiation of one program memory. Note that you can set this parameter for only one memory instantiation within your design at a time.

Having enabled this facility within your design, you must first select the correct version for the operating system you are using from the JTAG_loader di-

```

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

-- Uncomment the following library declaration if instantiating
-- any Xilinx primitives in this code.
--library UNISIM;
--use UNISIM.VComponents.all;

entity pico_wave_top is
    Port ( clk : in STD_LOGIC;
          led : out STD_LOGIC_VECTOR (3 downto 0));
end pico_wave_top;

architecture Behavioral of pico_wave_top is

component kcpsm6 is
    generic( hwbuidl : std_logic_vector(7 downto 0) := X"00";
            interrupt_vector : std_logic_vector(11 downto 0) := X"3FF";
            scratch_pad_memory_size : integer := 64);
    port ( address : out std_logic_vector(11 downto 0);
          instruction : in std_logic_vector(17 downto 0);
          bram_enable : out std_logic;
          in_port : in std_logic_vector(7 downto 0);
          out_port : out std_logic_vector(7 downto 0);
          port_id : out std_logic_vector(7 downto 0);
          write_strobe : out std_logic;
          k_write_strobe : out std_logic;
          read_strobe : out std_logic;
          interrupt : in std_logic;
          interrupt_ack : out std_logic;
          sleep : in std_logic;
          reset : in std_logic;
          clk : in std_logic);
end component kcpsm6;

component test is
    generic( C_FAMILY : string := "S6";
            C_RAM_SIZE_KEYWORDS : integer := 1;
            C_JTAG_LOADER_ENABLE : integer := 1);
    Port ( address : in std_logic_vector(11 downto 0);
          instruction : out std_logic_vector(17 downto 0);
          enable : in std_logic;
          rd1 : out std_logic;
          clk : in std_logic);
end component test;

SIGNAL instruction : std_logic_vector(17 DOWNTO 0);
SIGNAL address : std_logic_vector(11 DOWNTO 0);
SIGNAL enable : std_logic;
SIGNAL rd1 : std_logic;
SIGNAL kcpsm6_output : std_logic_vector(7 downto 0);
SIGNAL port_id : std_logic_vector(7 downto 0);
SIGNAL write_strobe:std_logic;
begin

ram_inst : test PORT MAP (
    address => address,
    instruction => instruction,
    enable => enable,
    rd1 => rd1,
    clk => clk);

```

```

pico_inst : kcpsm6 PORT MAP (
  address => address,
  instruction => instruction,
  bram_enable => enable,
  in_port => (OTHERS =>'0'),
  out_port => kcpsm6_output,
  port_id => port_id,
  write_strobe => write_strobe,
  k_write_strobe => open,
  read_strobe => open,
  interrupt => '0',
  interrupt_ack => open,
  sleep => '0',
  reset => rdl,
  clk => clk);

output_ports: process(clk)
begin
  if rising_edge(clk) then
    if write_strobe = '1' then
      -- 4 LEDs at port address 01 hex Spartan LX9
      Development board
        if port_id(0) = '1' then
          led <= kcpsm6_output(3 DOWNTO 0);
        end if;
      end if;
    end if;
  end process;
end Behavioral;

```

Figure 5 – Code snippet for a PicoBlaze that will flash LEDs on an LX9 Spartan development board

rectory and copy it to your working directory (where you hex file is located). Now you can open a command window and navigate to your working directory and use the command below.

jtagloader -l <Your Project Name>.hex

Note: I have renamed the version of the executable for my OS jtagloader.exe.

This action will download the hex file created when you ran the assembler on your latest PSM file, with results as shown in Figure 7. As this file downloads, you will notice that the JTAG loader halts the core execution and downloads the new program to memory before releasing the core reset, at which point it starts running your new program.

Once you are happy with the updated behavior of the PSM file, you can rerun the implementation and bit file generation, ensuring that the next time the device is configured it will execute the updated program. 🌈



Figure 6 – ISim simulation results



Figure 7 – JTAG loader in action

Changing Utilization Rates in Real Time to Investigate FPGA Power Behavior

Power management in FPGA designs is always a critical issue. Here's a new way to measure estimated power on a real FPGA device.

by Ahmet Caner Yüzügüler

Hardware Design Engineer
Aselsan
acyuzuguler@aselsan.com.tr

and Emre Sahin

Senior Hardware Design Engineer
Aselsan
emresahin@aselsan.com.tr



Modern FPGA chips are very capable of developing high-performance applications, but power management in these designs is usually a limiting factor. The resource usage of an FPGA device mostly determines the capacity and processing rate of the design, but adding resources increases the power consumption. Higher power dissipation brings more operating costs, larger area requirements and higher junction temperature, which the designer must address with more airflow and cooling systems.

Since the total power consumption of a board or system is so important, designers must set a power budget, juggling trade-offs between resource usage and power concerns. The ability to predict the amount of potential power consumption of the system beforehand therefore gives the designer a head start.

For preimplementation power evaluation, Xilinx offers some tools that estimate the power consumption virtually based on user entries or synthesis reports. One of them is the Xilinx® Power Estimator (XPE) spreadsheet. This Excel-based power-estimation tool lets you enter design properties such as resource usage, toggle rates and clock frequency as input and calculates the estimated power according to the device information. Another commonly used tool is Xilinx Power Analyzer (XPA). After placement and routing, XPA imports the generated NCD file and estimates the power with implementation details and simulation results instead of user inputs for more accurate estimations.

As an alternative, we have devised a new method to measure the estimated power of an FPGA design on a real device. In order to imitate different implementation scenarios, we have created a generic, device-free VHDL design within which it is possible to change the number of activated resources—namely, DSP slices, Block RAMs and slice registers—along with their operational conditions (junction temperature, clock frequency and toggle rates) through a serial channel while the FPGA is running. Our technique monitors

the current and voltage levels of the supply rail simultaneously so that we can easily observe the dynamic power characteristics of a device for different resource-usage and ambient conditions.

We have implemented the design on a Xilinx KC702 Evaluation Board. But you can implement it on any other device as well, with a few minor changes, so long as the FPGA device supports the IP cores used in the design.

Another possible way to use our technique might be as a testing VHDL design for FPGA boards. Let us imag-

ine we have recently designed a general-purpose Kintex®-7 board. The customer may implement any design on the board with an unknown amount of resource capacity used in changing ambient conditions. To ensure the stability and robustness of the board, we need to force the operating limits of the device in the highest and lowest required ambient temperatures and verify that the FPGA can work for different resource-usage cases. However, testing every resource-utilization scenario would require a new VHDL design from

scratch and take too much time. Our suggested method gives designers the flexibility to manipulate the test design as required in real time.

IMPLEMENTATION DETAILS

We have tried to keep our implementation as simple as possible to prevent generating slice LUTs used as logic, which consume power that we cannot control. The simplest way to implement slice registers is to combine them as a shift register block. Figure 1a is a block diagram of how slice registers are im-

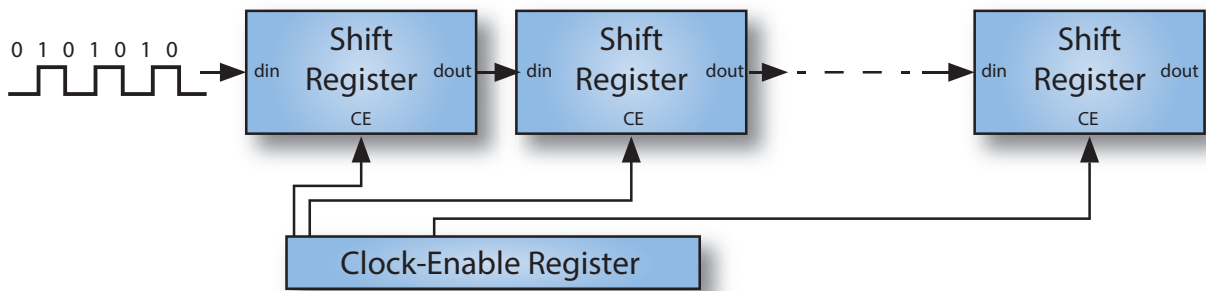


Figure 1a

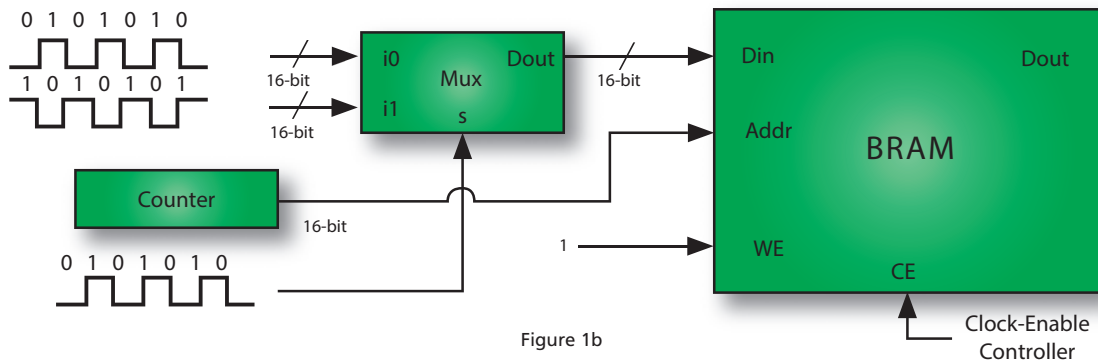


Figure 1b

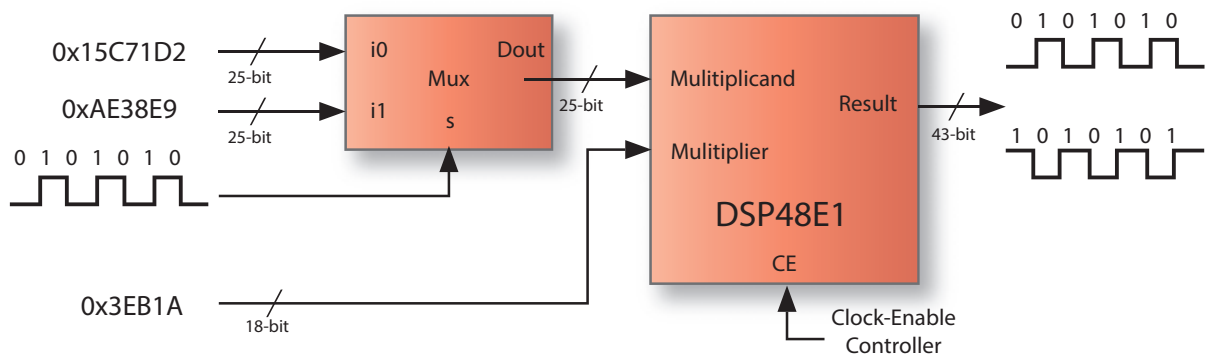


Figure 1c

Figure 1 — Implementation block diagrams: (a) slice registers, (b) Block RAMs and (c) DSP slices

plemented. One simple point to note here is that, when we attempt to create slice registers, synthesis tools tend to use slice LUTs as 32-bit shift registers (SRL32) instead of slice registers. You can use the following VHDL attribute to force the synthesis software to utilize slice registers instead:

```
attribute SHREG_EXTRACT : string;
attribute SHREG_EXTRACT of <signal_name> : signal is "no";
```

The LogiCORE™ Block Memory Generator creates Block RAMs as single-port RAMs. A 16-bit word, whose bits are continuously toggled, is constantly written into a random address of the activated BRAMs to keep them occupied. Figure 1b is a block diagram of one BRAM component. Similarly, we use DSP slices to multiply a 25-bit multiplicand and an 18-bit multiplier that produces a 43-bit output, which is the maximum word width a single DSP48E1 can handle. The multiplicand of all DSP components is regularly altered to make the DSPs dissipate dynamic power, as shown in Figure 1c. However, synthesis tools attempt to remove the resources in the synthesis process, since the outputs of these modules are not connected to any output pin. You can use the following attribute to keep the resources untrimmed:

```
attribute KEEP : string;
attribute KEEP of <signal_name>: signal is "TRUE";
```

In this scheme, you activate resources by controlling their clock-enable signals. Clock-disabled resources consume so little power that they can be considered as not utilized in the design at all. More specifically, power consumption of 50 DSP slices with a 100 percent toggle rate is 0.112 watts, whereas it is 0.001 W when their clock-enable signals are deasserted. The result is to obtain approximately the same outcome for different utilization rates. As a consequence, a design with each resource component controlled with a clock-en-

able signal over serial communication simulates changing utilization rates instantaneously without the need for new design and implementation stages.

One important factor that affects power consumption considerably is the average toggle rate of the resources. The toggle rate is the number of transitions

of the output for a specific resource per clock cycle. You can achieve a constant 100 percent toggle rate by alternating inputs of the resources that make the output change its state in every cycle. For a DSP slice, for example, you would adjust the toggle rate by choosing two multiplicands, which are alternated consecutively, and one multiplier, so that each bit of output changes in each cycle. The alternation rate of inputs determines the toggle rate of a resource; thus, we control the toggle rate of a resource type on the fly via serial channel.

Power consumption is also directly proportional to the clock frequency. We have utilized a Mixed-Mode Clock Manager (MMCM) to generate clocks with varying frequencies. A set of registers determines the frequency, phase and duty cycle of the MMCM output, which would normally only be initialized when

the design is implemented and a bit-stream file is generated. However, the MMCM's Dynamic Reconfiguration Port allows us to change such characteristics as the output clock while the FPGA is running. Clock phase and duty cycle are not of interest, since phase does not affect power consumption and duty cycle does not vary in most cases.

Frequency, on the other hand, is highly correlated with power dissipation. The internal mechanism of an MMCM works in such a way that VCO frequency is divided by the value of the

CLKOUT0_DIVIDE register to obtain the desired output frequency. We assign new values to this register by following a state machine algorithm, as explained in [Xilinx application note XAPP888](#), for a glitch-free transition. At the same time, we change the clock frequency of the design in real time according to the user input in the graphical user interface (GUI). Then, we can easily observe the power characteristics of the design for different frequencies.

Power behavior under high-temperature conditions is another important issue to observe and verify cautiously. Core temperature of the silicon depends on the design of the board, processing rate, ambient temperature, heat sink and airflow provided by the fan. In our design, we partially control the junction temperature by changing the speed of the fan located on top of the FPGA chip with a simple PWM on-off controller. You can also use an external heating tool such as a heat gun to reduce heating time.

On-chip sensors measure the core temperature and an analog-to-digital converter, XADC, is generated through the LogiCORE XADC Wizard. When you enter a reference temperature value in the GUI, it transmits to the device through the serial channel and compares it to the core temperature measured with on-chip sensors. The speed of the fan is controlled to stabilize the junction temperature at the desired level. In this way, you can observe power consumption and plot it over core temperature to confirm that it meets the power budget of your design and stays within critical limits for the required heating profile.

A graphical user interface is designed for communication with the FPGA device to easily change the parameters explained above. Figure 2 shows a screen shot of the GUI. We first connect to the device via a standard UART with one of the available ports. Then, we can open Xilinx Power Estimator to compare its estimated values with the measured ones. The GUI starts to plot a power vs. time

graph immediately for both estimated and actual power with the default values of the parameters. You can modify those parameters on the related panels and specify the number of utilized slice resources with their toggle rates at the Resource panel. The Utilization Sweep panel allows you to sweep the utilization rate of a particular resource from 0 to 100 percent with equal intervals and plot the power vs. utilization graph with the measured power values for each interval.

Similarly, you can change the output of the MMCM and sweep the clock frequency on the Clock Select panel. In the next column, the supply voltage of the FPGA chip is measured and plotted continuously on the Vccint panel. The Core Temperature panel, meanwhile, shows and plots core temperature. The user can enter a reference temperature to change junction temperature in the provided box.

Voltage and current levels of the power supply are measured with Texas Instruments' UCD9248 Digital PWM System Controller, integrated on the KC702 board. This multirail and multiphase PWM controller for power converters supports the Pow-

er Management Bus (PMBus) communication protocol. Its PWM signal drives a UCD7242 integrated circuit that regulates Vccint supply voltage. A set of PMBus commands is used to configure IC functions. The UCD7242 possesses on-chip voltage- and current-sensing circuitry and communicates with the UCD9248. Our GUI software continuously transmits the corresponding PMBus commands to the device, while the voltage and current values of the DC/DC converter are received back in a standard PMBus data format. We then convert the incoming bytes to actual values and obtain the voltage and current levels of the power supply.

Running a complex and congested design in high temperatures may disturb the supply voltage levels. Unfortunately, FPGA chips have a narrow tolerable range of input voltages. Exceeding these levels can cause loss of functionality or permanent damage on the chip. Hence, voltage-level stability during challenging operating conditions is another factor to test after you have designed a board. The Vccint panel is intended to observe and verify that the power supply sys-

tem of the designed board is capable of tolerating voltage variations for overwhelming processing rates under high temperatures.

One important point designers should always keep in mind is that every device has different power dissipation due to manufacturing process variations. For example, after a Virtex-7 FPGA chip is produced, a minimum voltage supply level that is sufficient to operate the chip properly is determined in test stages and written into E-fuses. Adjusting voltage regulator output to this minimum voltage level changes static power consumption. Thus, an offset in static power may cause a difference in results between XPE and our design. For instance, static power consumption of a Kintex-7 chip estimated in XPE differs by 0.7 W between typical and worst-case calculations.

GOOD FIT WITH XPE

We have tested our design on many different design scenarios with varying factors to ensure its accuracy and reliability. In the meantime, the estimation results of XPE are also plotted on the same graphs with our measurements to compare estimated power dissipation to actual measurements.

The power behavior of the Kintex-7 device is observed by sweeping the resource usage rates starting from zero to the maximum value initially implemented on the design. Figure 3 shows the change in power dissipation as the utilization rate of a resource type is swept. Straight and dashed lines represent actual power measurements and XPE estimations respectively. We can see that the two curves follow each other very closely even when the utilization rate and power consumption are at high values.

These results show that Xilinx Power Estimator calculations are consistent with real measurements and predict the power consumption accurately. We can also conclude that our proposed method works as expected

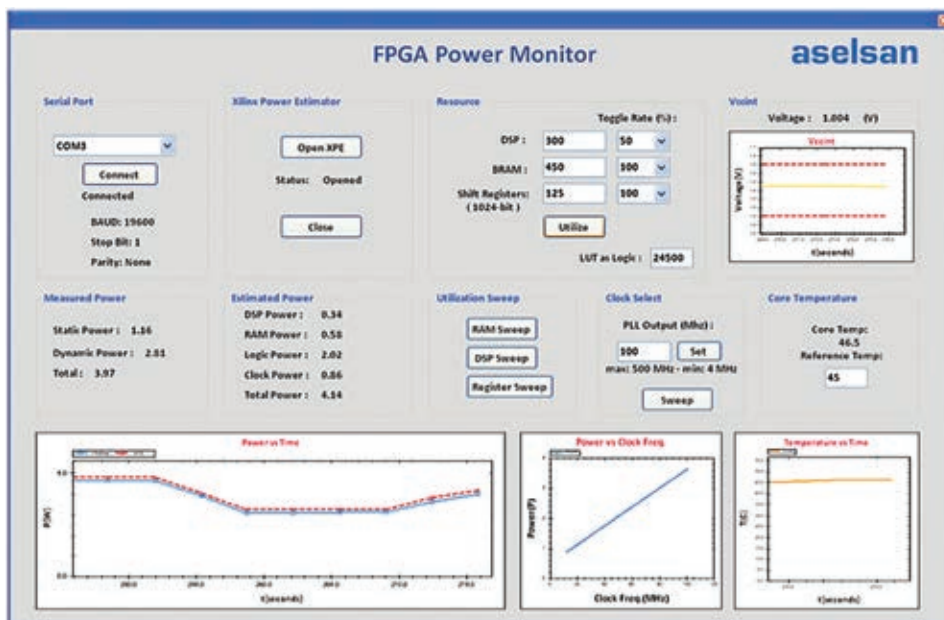



Figure 2 – Our graphical user interface


and can be an alternative to XPE. An even better option is to use these two methods collaboratively. Testing a design with both XPE and our method enables you to double-check the results and ensures you do not make any mistake while modifying the numbers at XPE. For instance, if you do not enter the average fan-out number of a clock correctly at XPE, or simply misunderstand a concept and fill a box inaccurately, the resulting plots would not overlap, signaling that an error exists so that you can correct the entries and prevent a misleading estimation.

The rated specifications of an FPGA board, such as maximum power dissipation or allowable core temperature range, are always included in the datasheet of the board. However, a design with high resource utilization and clock frequency can exceed these rated values. Thus, it's important to verify that power regulators supply enough current to the device

and that the cooling system is sufficient to keep the temperature within critical values for a possible FPGA design. Our method makes it possible to test the reliability of the board for high-performance requirements by increasing resource usage and clock frequency. Such a test helps determine the maximum signal-processing rate possible in designs implemented on this board. Alternatively, it gives us an idea about whether we need to upgrade board capabilities and the cooling mechanism of the system.

Now that test results confirm that our method is a reliable way to manipulate resource usage and evaluate power consumption, engineers have another option for power management in predesign stages. With some further improvements, such as a JTAG interface, fully device-free VHDL code or a common current-sensing system, this design would be a key tool for FPGA projects. ●●








XEM7350 KINTEX 7

- ✓ Available K70T, K160T, and K410T
- ✓ FMC HPC Carrier
- ✓ USB 3.0 transfers over 340 MB/s
- ✓ 8 Gigabit Transceiver Lanes
- ✓ 512 MiB DDR3
- ✓ Excellent JESD204B Host


FrontPanel SDK

Software API and Driver



Device Firmware



FPGA IP

www.opalkelly.com

Power vs. Utilization

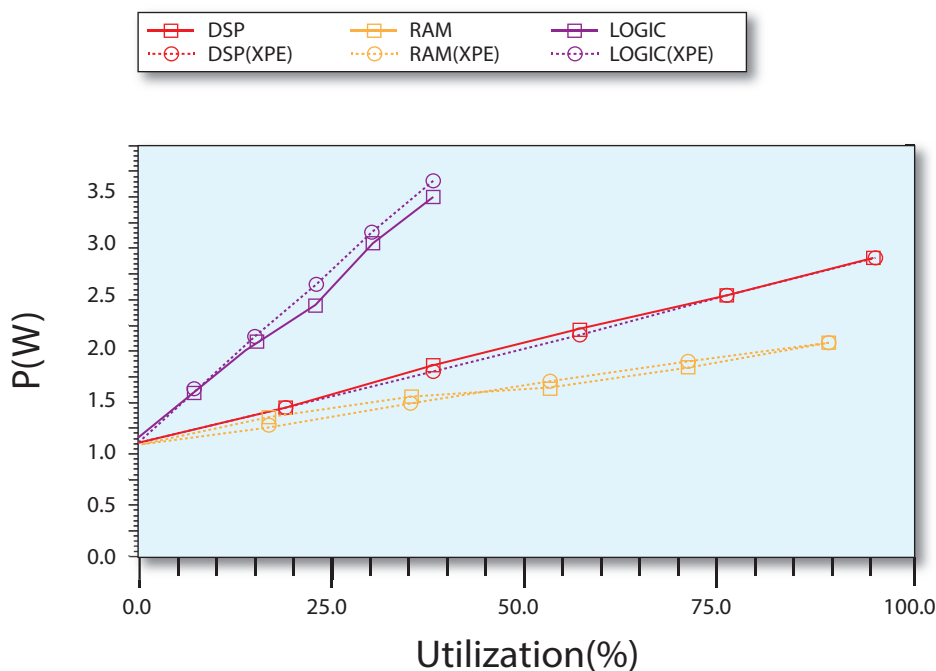


Figure 3 – In this graph plotting power vs. utilization, straight lines represent actual power measurements and dashed lines the XPE estimates.

What's New in the Vivado 2014.3 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 2014.3 is available from the Xilinx Download Center at www.xilinx.com/download.

VIVADO DESIGN SUITE 2014.3 RELEASE HIGHLIGHTS

The latest release of the Vivado Design Suite features expanded support for Virtex® UltraScale™ devices, along with 20 percent compile-time improvements for 7 series devices. Overall productivity enhancements include quality-of-results improvements in Vivado high-level synthesis (HLS); automated connectivity in Vivado IP Integrator between streaming and memory-mapped AXI interconnects; and extensions to the Vivado IP Catalog. [See the Vivado Design Suite 2014.3 Release Notes for more information.](#)

General Access

- Kintex® and Virtex UltraScale: XCKU035, XCK U040, XCKU060, XCKU075, XCKU100, XCKU115, XCVU065, XCVU080 and XCVU095

Early Access

- Virtex UltraScale SSI devices: XCVU125, XCVU160, XCVU190 and XCVU440
- Contact a Xilinx field application engineer for access to these devices.

VIVADO DESIGN SUITE: DESIGN EDITION UPDATES

Vivado IP Integrator

Vivado IP Integrator provides the next level of design assistance with automation of the connectivity between streaming and memory-mapped AXI interconnects, as well as extensions to the Vivado IP Catalog.

Partial Reconfiguration

Xilinx partial-reconfiguration technology now offers initial support for UltraScale devices (supports implementation for KU040 and VU095 devices only). Bitstream generation is disabled until ES2 silicon (Virtex UltraScale) or production silicon (Kintex UltraScale) is available.

Xilinx has also added new bitstream generation features, including a per-frame CRC option that inserts CRC values for partial bitstream integrity checking. Users can request only specific partial bitstreams with the use of `write_bitstream -cell`. For more information, see the section on partial reconfiguration in the “Vivado Design Suite User Guide.”

VIVADO DESIGN SUITE: SYSTEM EDITION UPDATES

Vivado High-Level Synthesis

Xilinx has enhanced the quality-of-results (QoR) in Vivado HLS, with an average of 10 percent better fmax and 5 percent lower LUT utilization, along with AMBA® AXI-4 interface inference. QoR improvements include enhanced DSP block support, optimized control structures for loops and advanced user control for operator latency. Designers can use the latter feature to model memory port latency.

Interface enhancements for AXI4 memory-mapped and streaming interfaces include improved modeling for arbitrary precision types. Users can access size information. Also, the main page is context sensitive and now directly accessible from the directive editor.

System Generator for DSP

This tool now supports MATLAB® 2014B and offers enhanced Ethernet hardware co-simulation, including free running-clock support and board part block labeling. Expanded support for multiple asynchronous clock domains, through metastability registers or user-defined RTL, allows a more complete system-level design. Other enhancements include:

- System Generator for DSP now offers improved AXI4-Stream interface handling for user-defined RTL import (black box).
- Improved cross-probing between your model and the waveform viewer enables faster debug and verification.
- Upgraded performance models for CORDIC and DSP48 macros decrease simulation run-times.
- Improved caching support accelerates simulation modeling of complex IP super-sample-rate FIR compiler, making it possible to implement gigahertz sample-rate filters.

INTRODUCING THE NEW ULTRAFast EMBEDDED DESIGN METHODOLOGY

Xilinx expands the UltraFast™ Design Methodology with the new “UltraFast Embedded Design Methodology Guide,” enabling embedded-design teams to make informed decisions in the creation of smarter systems leveraging Zynq®-7000 All Programmable SoCs. The guide covers key principles, specific do’s and don’ts, best practices and tips on avoiding pitfalls. It also includes use cases outlining experiences and learning gained from system development inside and outside of Xilinx.

Best practices focus on the entire design team consisting of system architects and software engineers, as well as hardware engineers. Download the new “UltraFast Embedded Design Methodology Guide” today at http://www.xilinx.com/support/documentation/sw_manuals/ug1046-ultrafast-design-methodology-guide.pdf.

Vivado QuickTake Tutorials

Vivado Design Suite QuickTake video tutorials are how-to videos that take a look inside the features of the Vivado Design Suite. New topics include: What’s new in 2014.3, Vivado Activation Floating-License Generation, Licensing and Activation Overview, Using the UltraScale Memory Controller IP and Using report_cdc to Analyze CDC Structural Issues.

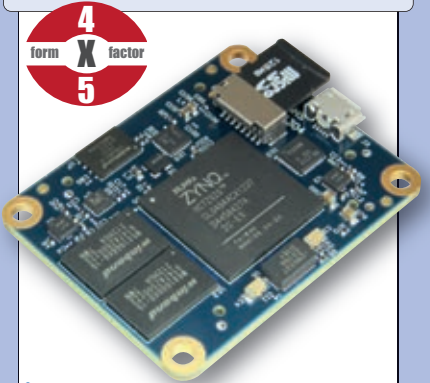
See all Quick Take Videos at www.xilinx.com/training/vivado.

Vivado Training

For instructor-led training on the Vivado Design Suite, please visit www.xilinx.com/training.

Download Vivado Design Suite 2014.3 today at <http://www.xilinx.com/download>.

All Programmable FPGA and SoC modules



rugged for harsh environments
extended device life cycle

Available SoMs:

ZYNQ™ KINTEX™
ARTIX™ SPARTAN™

Platform Features

- 4x5 cm compatible footprint
- up to 8 Gbit DDR3 SDRAM
- 256 Mbit SPI Flash
- Gigabit Ethernet
- USB option

 XILINX®

ALLIANCE PROGRAM
CERTIFIED MEMBER — BASE

Design Services

- Module customization
- Carrier board customization
- Custom project development



difference by design

www.trenz-electronic.de

Latest and Greatest from the Xilinx Alliance Program Partners

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

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

ALPHA DATA RELEASES XILINX ULTRASCALE BOARD FOR FPGA ACCELERATION

Certified Alliance Member [Alpha Data](#) has collaborated with Xilinx to provide one of the very first UltraScale™ FPGAs in a commercially available product suitable for large-scale deployment. The company's catalog of off-the-shelf boards now includes Xilinx's 20-nanometer Kintex® UltraScale devices, enabling a new class of accelerator cards. Alpha Data's [ADM-PCIE-KU3](#) is a high-performance, reconfigurable, half-length, low-profile, x16 PCIe® form-factor board based on the Xilinx Kintex UltraScale KU060 ASIC-class FPGAs.

Users can create applications for the board using the Vivado® Design Suite, with high-level synthesis (HLS) and OpenCL. Customers are provided with IP and reference designs for rapid development. The board's features can be demonstrated and evaluated right out of the box in Windows or Linux systems: Gen3 PCIe/DMA, DDR3, self-test, board control/status and more.

The ADM-PCIE-KU3 is the latest in the highly successful line of Alpha Data's Xilinx FPGA-centric products, the result of more than a decade of experience and partnership with Xilinx. The ADM-PCIE-KU3 features 16 Gbytes in two independent channels of DDR3-1600 ECC memory capable of 25-Gbyte/second bandwidth. The product has dual QSFP+ ports supporting

10G/40G Ethernet, dual SATA connections, voltage/temperature/current control and monitoring, and a passive air-cooled heat sink, all on a small 6.6 x 3.8-inch PCIe board.

The ADM-PCIE-KU3 is competitively priced and is available for order immediately. It is currently offered with the Kintex UltraScale KU060-2 ES device, and will be available with KU060-2 production silicon from mid-2015. The board ships with a comprehensive software development kit (SDK), including example designs and an evaluation of the PCIe/DMA IP core.

Additional IP and full PCIe/DMA cores are available to license. Further product details and ordering options for the ADM-PCIE-KU3 are available at <http://www.alpha-data.com/ku3>.

NATIONAL INSTRUMENTS' NEW SYSTEM-ON-MODULE IS POWERED BY THE XILINX ZYNQ SOC

The [sbRIO-9651](#) system-on-module (SOM) from Alliance Member [National Instruments](#), based on the Xilinx Zynq-7000 All Programmable SoC, combines a fully tested and validated hardware design with a complete middleware solution and the NI Linux Real-Time operating system. This combination significantly reduces development time, design risk and time-to-market. The NI SOM also provides an alternative to hardware description languages (HDLs) that simplifies the task of interfacing to I/O and communicating data. The product gives design teams the customizability of a SOM without the increased time and risk of developing custom software.

Unlike other SOMs, which offer limited deployment-ready software, the NI sbRIO-9651 integrates a validated board support package (BSP) and device drivers with the NI Linux Real-Time OS. As an alternative to HDLs, NI [LabVIEW](#) system-design software provides a graphical development environment with thousands of functions and IP blocks for both processor and FPGA logic development. Key benefits include:

- The NI SOM ships with a complete middleware solution out of the box to remove the time and risk associated with developing an embedded OS, custom software drivers and other common software components.
- LabVIEW FPGA integration eliminates a design team's need for HDL expertise, making powerful FPGA technology more accessible than ever before.
- NI Linux Real-Time, a robust Linux-based RTOS, gives design teams access to an extensive community of applications and IP.
- A complete development kit includes a reference carrier board along with design files for reuse.

- Design teams can use NI's CompactRIO technology to quickly prototype their applications and then deploy them with the same code used for prototyping, which saves significant time and effort.
- The SOM product is based on an industrial-grade Zynq-7020 SoC, and is designed, tested and validated for reliable long-term deployment and industrial applications.

TOPIC AND XILINX RELEASE COMPLETE DESIGN PLATFORM FOR MEDICAL SYSTEMS

Premier Alliance Member [Topic Embedded Systems](#) and Xilinx have released a platform for development and deployment of medical systems that combines the power of Topic's [Dypllo](#) framework and the Zynq SoC. Medical-device manufacturers can drag and drop sensor algorithms between the processor and FPGA fabric to instantly view trade-offs in performance and power consumption.

Topic's Dypllo framework integrates with the Vivado Design Suite and leverages high-level synthesis and partial re-configuration to generate all required hardware and software programming files. Dypllo then acts as a middleware component to the operating system that abstracts the hardware from the application layer and manages dynamic hardware and software configuration at run-time. Resulting designs achieve system performance and power savings over processor-only implementations.

The platform includes Topic's [Miami system-on-module](#), which comes with a Zynq XC7015 or ZC7030 device. The Florida carrier board has a 10-inch TFT touchscreen, SD Card slot, Wi-Fi, Bluetooth, Ethernet and HDMI. An onboard TI ADS1298 analog-to-digital converter supports sixteen 24-bit, 32-kHz channels for ECG/EMG/EEG electrodes. The hardware complies with ISO13485, EN 55021 Class A and IEC 61000-4-2 EMC standards, dramatically reducing development risk.

Complete development kits are available from Topic starting at \$3,450. The kits include a ZC7030 Miami SOM, Florida carrier board, eight electrodes, trial license of Dypllo, medical reference design and Linux kernel in a robust flight case. The Miami SOMs are also available for separate purchase. For more information, contact info@TopicEmbeddedProducts.com.

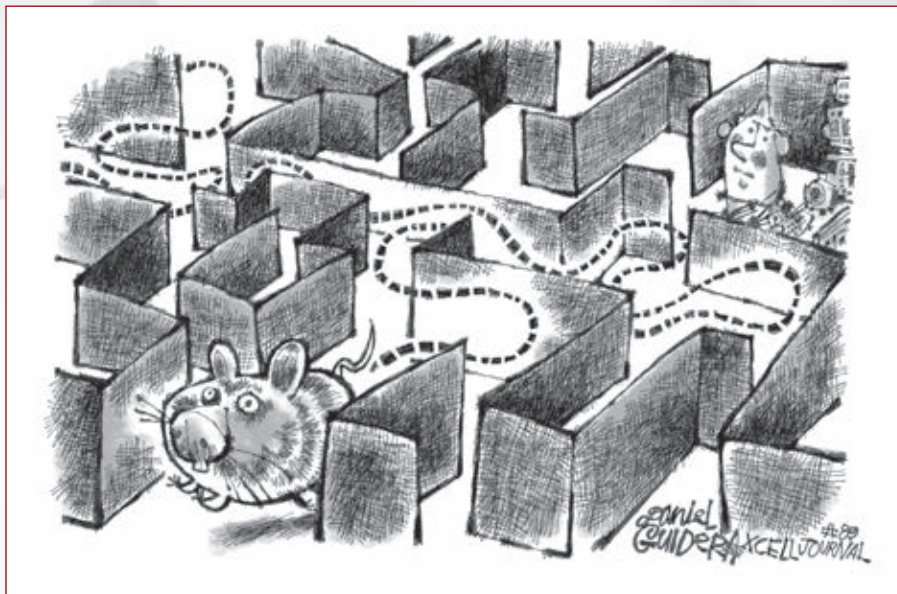
EASING IP EVALUATION REQUESTS TO ACCELERATE DESIGN AND INTEGRATION TIME FOR XYLON IP

The latest version of Vivado IP Integrator offers a pushbutton IP-evaluation request capability for IP cores from Xilinx Alliance members. Customers using the 2014.3 release will now have access to the logicBRICKS evaluation IP cores from Premier Alliance Member [Xylon](#), with IP from other alliance members to be added in future releases.

Seven Vivado-optimized logicBRICKS IP cores are certified as Vivado UltraFAST™ compatible and are optimized for Xilinx All Programmable FPGAs and SoCs. They are listed in the Vivado IP Catalog with a link to download the evaluation versions. This availability enables the rapid evaluation of efficient image-processing, video-processing and interface IP, further expanding the Vivado IP Catalog and shortening the overall design cycle. The cores include:

- [logiVIEW Perspective Transformation and Lens Correction Image Processor](#)
- [logi3D Scalable 3D Graphic Accelerator](#)
- [Bitmap 2.5D Graphics Accelerator](#)
- [logiCVC-ML Compact Multilayer Video Controller](#)
- [logiI2C Serial Bus Controller](#)
- [logiI2S Audio I2S Transmitter/Receiver](#)
- [logiSDHC SD Card Host Controller](#)

Xpress Yourself in Our Caption Contest



DANIEL GUIDERA

If your office floor plan is vaguely reminiscent of a hamster's maze, then you'll be inspired to submit an engineering- or technology-related caption for this cartoon showing a racing rodent caroming around a high-tech labyrinth. The image might inspire a caption like "The new layout of cubicles at Superior Semiconductor felt pretty homey to certain employees of the biotech firm next door."

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 a cool, Xilinx-branded gift from our swag closet.

The contest begins at 12:01 a.m. Pacific Time on Nov. 4, 2014. All entries must be received by the sponsor by 5 p.m. PT on Jan. 5, 2015.

So, get off your exercise wheel and start writing!

MIHAI ALBU, an electrical engineer at Stapla Ultrasonics (Wilmington, Mass.), won a shiny new Digilent Zynq Zybo board with this caption for the yoga cartoon in Issue 88 of *Xcell Journal*:



"By carefully Planning Ahead his 'OM' timings, George was finally able to transition to the next state of consciousness."

Congratulations as well to our two runners-up:

"The boss said I had to meet the stretch goals ..."

— David Su, software engineer, Intel Corp. (Santa Clara, Calif.)

"...his spiritual state of mind gained him 20 dB of SNR."

— Ali Alsaqqa, PhD student at the University at Buffalo



Synplify Premier

Put yourself in the driver's seat
for Xilinx FPGA Design

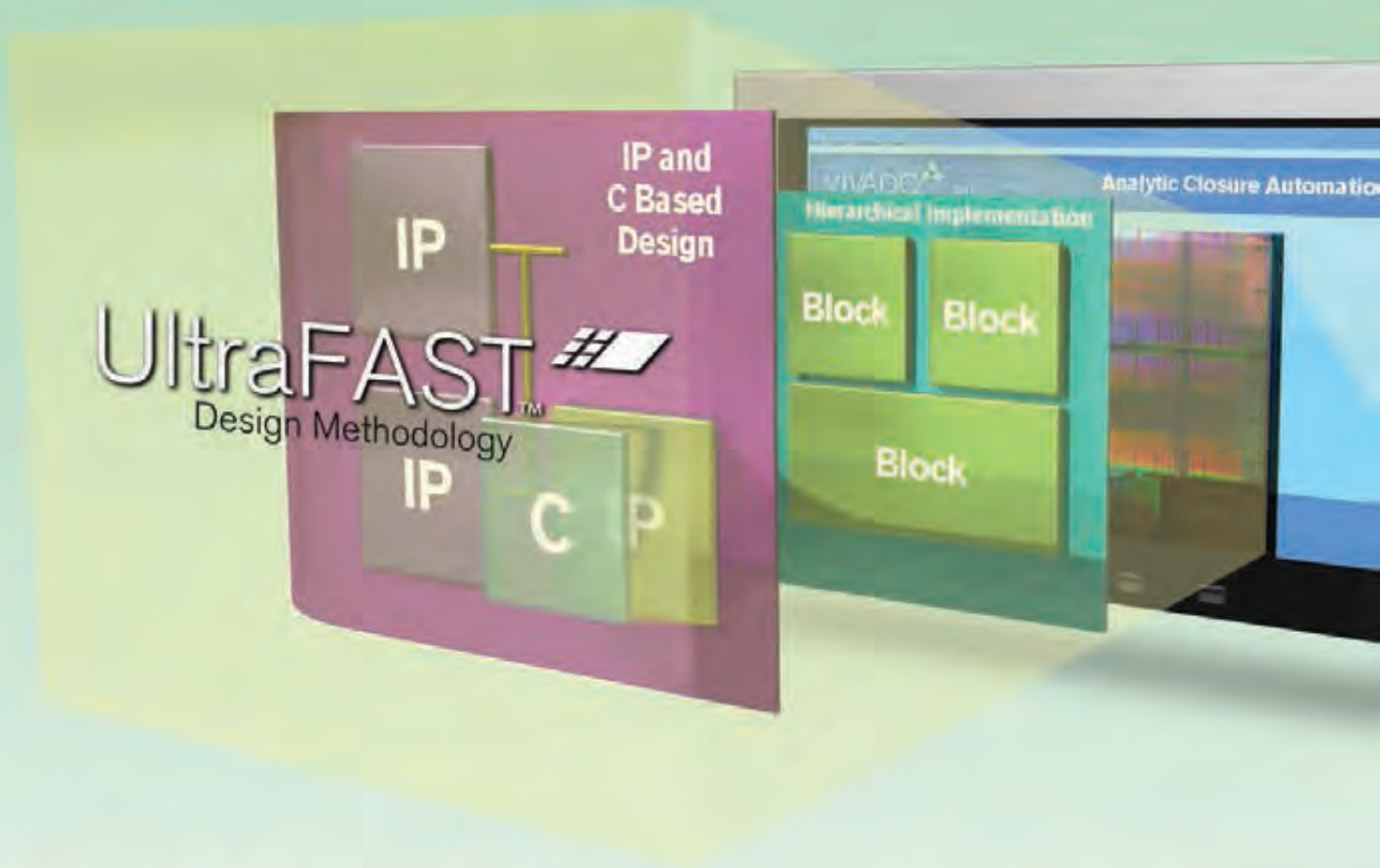
- ▶ Unmatched quality of results for Xilinx® FPGAs
- ▶ Fastest turnaround time, without timing performance compromise
- ▶ Results preservation from one run to the next
- ▶ Implementation control when you need it

To find out more and test drive Synplify Premier® today, visit

www.synopsys.com/fpgaqualityofresults

Xilinx Introduces

The **UltraFast™** Design Methodology for the Vivado® Design Suite



The **UltraFast Design Methodology** from Xilinx enables accelerated and predictable design cycles.

VIVADO

XILINX
ALL PROGRAMMABLE.

Learn More: www.xilinx.com/ultrafast