

Xcell journal

ISSUE 78, FIRST QUARTER 2012

SOLUTIONS FOR A PROGRAMMABLE WORLD

Charge to Market with Xilinx 7 Series Targeted Design Platforms

FPGA-Based Automotive ECU Design
Fits International Standards

How to Build a
Self-Checking Testbench

ISE Design Suite 13.4
Available for Download

FPGA vs. DSP:
Which is best for
your design?

page 44



 **XILINX**
www.xilinx.com/xcell/



DESIGNED BY **AVNET**

Compact, Easy-to-Use Kit Demonstrates the Versatility of Spartan-6 FPGAs

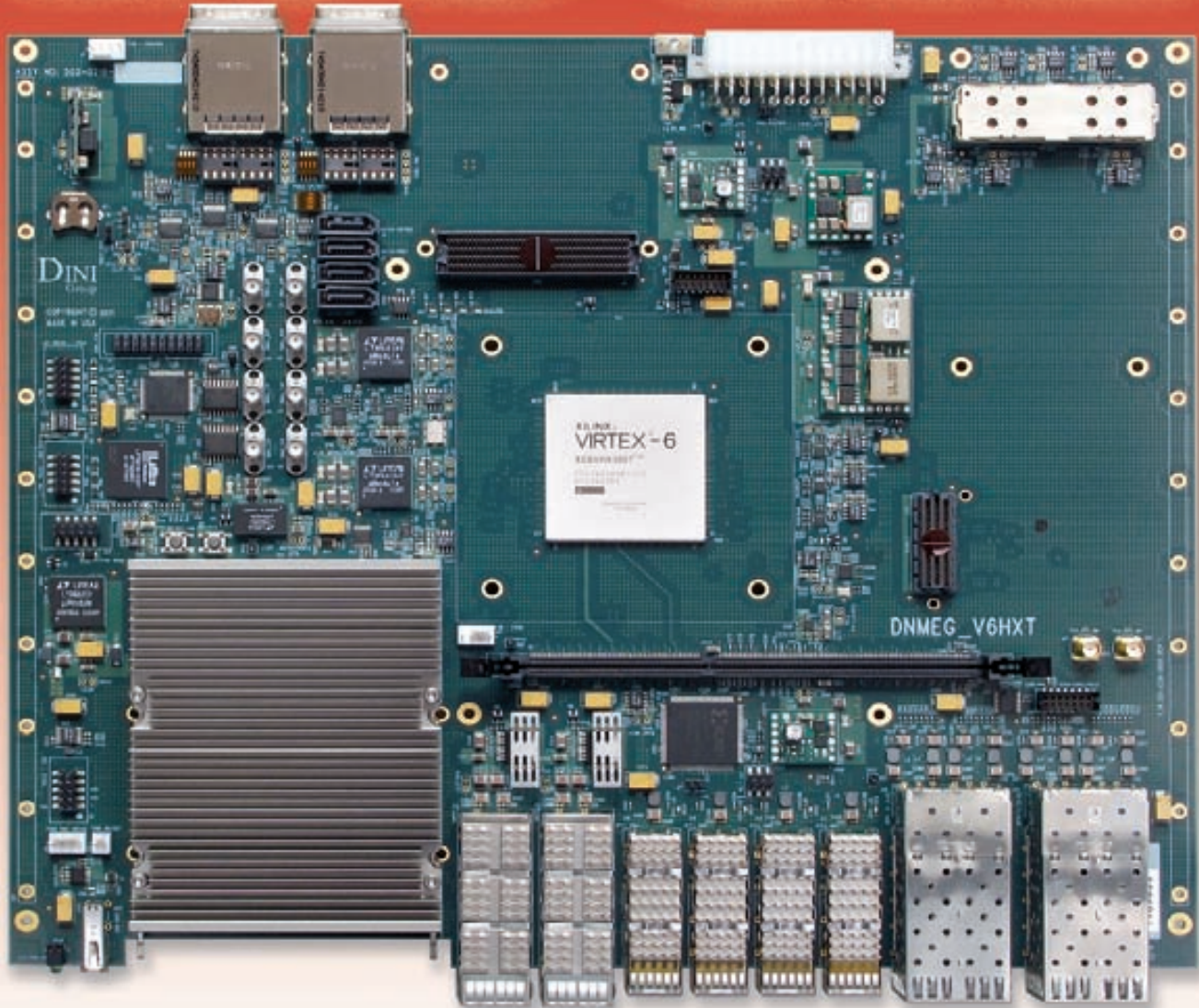
The low-cost Spartan®-6 FPGA LX9 MicroBoard is the perfect solution for designers interested in exploring the MicroBlaze™ soft processor or Spartan-6 FPGAs in general. The kit comes with several pre-built MicroBlaze “systems” allowing users to start software development just like any standard off-the-shelf microprocessor. The included Software Development Kit (SDK) provides a familiar Eclipse-based environment for writing and debugging code. Experienced FPGA users will find the MicroBoard a valuable tool for general-purpose prototyping and testing. The included peripherals and expansion interfaces make the kit ideal for a wide variety of applications.

Xilinx® Spartan®-6 FPGA LX9 MicroBoard Features

- Avnet Spartan-6 FPGA LX9 MicroBoard
- ISE® WebPACK® software with device locked SDK and ChipScope licenses
- Micro-USB and USB extension cables
- Price: \$89

To purchase this kit, visit
www.em.avnet.com/s6microboard
or call 800.332.8638.

250 GbE



Add it up: 10 GbE, 40 GbE, and 100GbE ports offer more capacity and versatility than any High Speed Serial I/O board – ever. The Xilinx, Virtex 6, HXT FPGA provides user program-ability and data rates to 11 GB/s. ASIC designers, Network engineers, and IP developers, can use this tool to prototype, emulate, model and test your most challenging projects. The board maybe used stand-alone, PCIe hosted, or plugged into any Dini Group ASIC prototyping board. Users will appreciate the scope of interfaces:

- CFP Module—100 GbE or Single/Dual 40 GbE
- 2-QSFP+ Modules—40 GbE
- 4-SFP+ Modules—40 GbE
- 8-SFP+ Modules—8 GbE

All FPGA resources are user available and supported by a 240-pin DDR3 UDIMM bulk memory. Daughter cards are accommodated for expansion, customization, and FMC (Vita-57). Put your high speed serial designs to work. Here is a Dini board just for you.

DINI
Group

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 Miyuki Takegoshi, Japan miyuki.takegoshi@xilinx.com
REPRINT ORDERS	1-800-493-5551



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

© 2012 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.

Peter Alfke Remembered, 1931-2011

Not long ago we learned of the death of our old friend and Xilinx colleague Peter Alfke. If you're a longtime Xilinx user or even new to FPGA design, chances are you've read some pearls of FPGA or general engineering wisdom from Peter at one time or another. He joined Xilinx in 1988 (employee number 109) as the director of application engineering and later that year started *Xcell Journal*.

Peter once told me that he came from a long line of educators and had a passion for teaching. That's why he loved being an FAE, started *Xcell* and later became the director of technical applications at Xilinx, where he was responsible for technical customer support, documentation and software QA. This passion for education and engineering was evident in his speeches and in his many writings—articles, forums and newsletters. In later years, he wrote the wonderfully practical “User Guide Lite” series that really cut to the good stuff, pointing users to the essentials of a given device. In the days when *EE Times'* *pldesignline.com* would track “most read” and “most popular” articles, he took great pride in seeing each new tome sitting atop both those lists—in some cases, for months.

In September 2009 at the age of 77, he had been struggling with a growing list of health issues, and so one day at lunch, he turned to his group of friends sitting around the table and said, “Boys, I've decided to retire before they have to take me out of here on a stretcher.” We threw him a retirement party and I wrote a piece for *EE Times* announcing his departure. Peter was touched by the response. He particularly loved one comment from a *comp.arch.fpga* member who wrote, “I use Xilinx because Peter Alfke told me to.”

Peter and I stayed in touch after his retirement and whenever a new issue of *Xcell Journal* came out, I would soon receive an e-mail or a call from Peter telling me what he liked or disliked about a particular article. At the same time, he would matter-of-factly mention that he was visiting a doctor to have, first, a hip replacement, then a part of his lung removed and later, treatment for spreading cancer. One day I received a contributed piece on FIFOs that was a bit over my head. I naturally asked Peter to give it a read. He answered within minutes.

Michael, this is a bad paper.

The author does not understand our asynchronous FIFO design, does not understand metastability issues and never explained the unorthodox features of his particular design.

He writes 10 times faster into than he reads from the FIFO. Nevertheless he worries about going empty ??? There must be some peculiar burst situations...

Our FIFO is very sophisticated in resolving asynchronous timing issues, but is very straightforward in its functionality: Everything you write into it, you must read out in the same sequence. Additional features, like idle codes, you should design as synchronous extensions on either side.

I am back out of Stanford Hospital, recovering at home, but stuff like this will drive me out of bed anytime.

*Cheers,
Peter*

Unfortunately, that was the last correspondence we had. And of course, I didn't run the article. Who could argue with a legend?



Peter Alfke (second from right) on a 2008 trip to CERN with Xilinx colleagues Patrick Lysaght (left) and Marc Defossez.



Mike Santarini
Publisher

Pass GO, Collect 3.6 Billion Samples per Second



X6 GSPS



Extremely Versatile

Complete chassis & embedded PC solutions available! Adapt to VPX, Cabled PCI-Express, PCI-Express, 64-bit PCI, and CompactPCI with our high-quality carriers.

Features

- Two 1.8 GSPS, 12-bit A/D Channels
- Single channel interleaved @ 3.6GHz
- +/-1V, AC-Coupled, 50 ohm, SMA Inputs
- Xilinx Virtex-6 SX315T/SX475T or LX240T
- 4 Banks of 1GB DRAM (4 GB total)
- Rugged levels available

VIRTEX⁶

wireless
ip cores



FrameWork Logic



805.578.4260 phone • www.innovative-dsp.com



**Innovative
Integration**
a subsidiary of ISI

... real time solutions!

VIEWPOINTS

Letter From the Publisher

Peter Alfke

Remembered, 1931-2011... 4

Xpert Opinion

Embedded Vision: FPGAs'

Next Notable Technology

Opportunity... 14



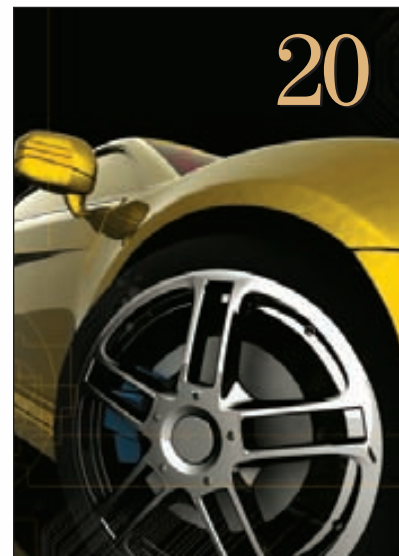
XCELLENCE BY DESIGN APPLICATION FEATURES

Xcellence in Automotive

FPGA-Based Automotive ECU
Design Addresses AUTOSAR,
ISO 26262 Standards... 20

Xcellence in Prototyping

Lowering the Barriers
to a Successful FPGA-Based
Prototype Project... 32



Cover Story

Charge to Market with
Xilinx® 7 Series Targeted
Design Platforms

8



THE XILINX XPERIENCE FEATURES

Xplanation: FPGA 101

Ins and Outs of Digital Filter
Design and Implementation... **36**

Xplanation: FPGA 101

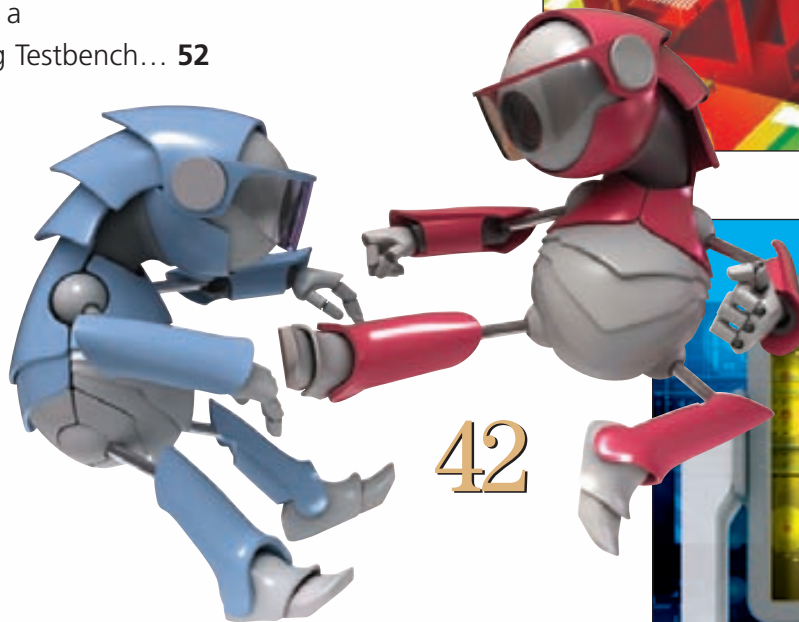
Using FPGAs to Solve Tough
DSP Design Challenges... **42**

Xplanation: FPGA 101

Tips and Tricks for Better Floating-Point
Calculations in Embedded Systems... **48**

Ask FAE-X

How to Build a
Self-Checking Testbench... **52**



XTRA READING

Tools of Xcellence A new class of
DC/DC turns down the noise... **58**

Xtra, Xtra The latest Xilinx
tool updates and patches,
as of January 2012... **64**

Xclamations! Share your wit and
wisdom by supplying a caption for our
techy cartoon, for a chance to win an
Avnet Spartan®-6 LX9 MicroBoard... **66**



Excellence in Magazine & Journal Writing
2010, 2011



Excellence in Magazine & Journal Design and Layout
2010, 2011

Charge to Market with Xilinx 7 Series Targeted Design Platforms

Bored with just a board? New Kintex-7 and Virtex-7 kits accelerate FPGA development.

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





To help customers evaluate which FPGAs best suit their needs and get their designs up and running fast, Xilinx Inc. has announced the release of the first three Targeted Design Platforms in support of its 28-nanometer, 7 series FPGAs. Base kits for the Virtex[®]-7 and Kintex[™]-7 FPGAs include Agile Mixed-Signal daughtercards as well as industry-standard FMC connectors. In addition to these base kits, Avnet is also releasing its Xilinx Kintex-7 FPGA DSP Kit for customers eager to take advantage of the fact that FPGAs can do the job of multiple DSPs and perform faster.

The three new kits represent the first deliverables of Xilinx's second-generation platforms for FPGA system development. Xilinx introduced its Targeted Design Platform (TDP) strategy back in 2009 in support of its 6 series devices (see cover story, *Xcell Journal* Issue 68). Xilinx describes the platform strategy as a pyramid consisting of four layers: base platforms, domain-specific platforms, market-specific platforms and, at the apex, customer designs (Figure 1). The TDP approach allows customers to take advantage of prebuilt solutions from Xilinx and the Xilinx ecosystem, and easily add these to their FPGA design projects. With the TDP as their design's foundation, customers can focus on features that will differentiate their products.

In the Xilinx TDP hierarchy, base platforms like the new Virtex-7 FPGA VC707 Evaluation Kit and Kintex-7 FPGA KC705 Evaluation Kit are the primary means for evaluating Xilinx's device families. On top of the base TDPs, domain-specific kits like the one from Avnet further help customers targeting designs in the DSP, embedded and connectivity domains. For even greater design acceleration, Xilinx also offers market-specific development platforms for segments such as aerospace and defense, automotive, consumer, wireless and wired communications, professional broadcast and industrial, scientific and medical. Versions of these market-specific kits for the 7 series are forthcoming.

All Xilinx TDPs include not only development boards and the software to program the FPGAs, but also internal and third-party IP, reference designs and documentation, and a broad portfolio of internally and partner-developed FPGA Mezzanine Card (FMC) daughtercards. All these assets combine into a powerful vehicle to help users get their designs working at a system level sooner than ever before possible.

Xilinx offers all three types of TDPs for its 6 series devices and is now beginning to roll out the TDPs for the 7 series. The VC707, KC705 and Kintex DSP Kit are the first of many 7 series kits that will debut in the coming months.

Mark Moran, senior strategic marketing manager with Xilinx, is emphatic in noting that TDPs are not your traditional development kits. All programmable chip vendors offer kits, which usually include a development board, a power cord, software and sometimes connectors.

"In a typical kit, you load the software into your PC, plug in the board, power it up and pretty much all you see is an LED light up to let you know it's working...and that's it," said Moran. "If you want more, you typically

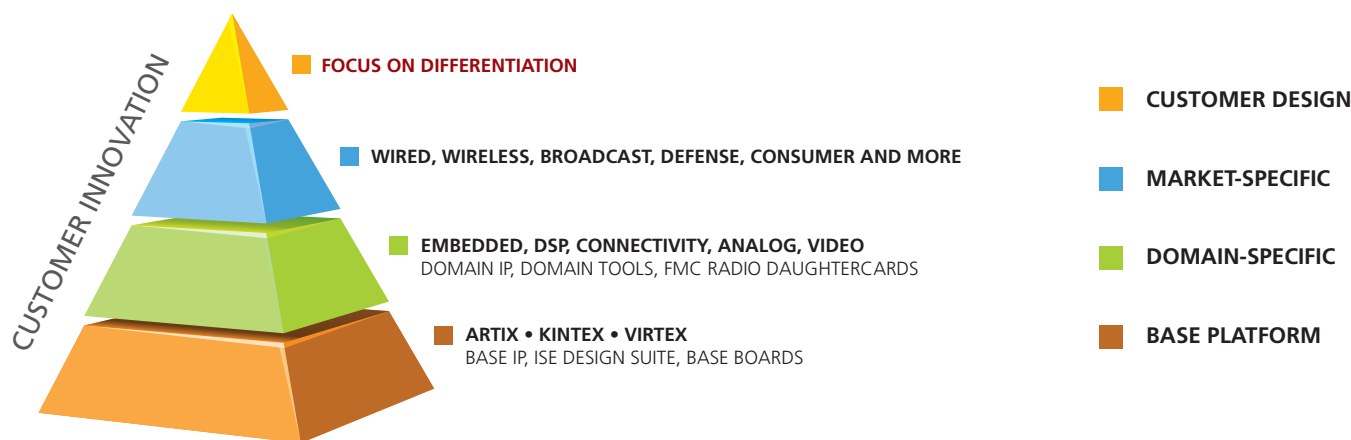


Figure 1 – The Xilinx Targeted Design Platform concept allows users to focus their design efforts on product differentiation and get products to market quickly.

have to go to the vendor's website to search for documentation and IP that's appropriate for your chip and board. In most cases, you'll be lucky if you find even basic building blocks. And if you do, they are often out of date. Some vendors may even require that you purchase standard IP blocks just to get up and running. Buying a kit at some vendors is a lot like buying a car without tires—you can sit in the driver's seat and turn it on, but you can't drive it anywhere."

Worse, many chip vendors offer a dizzying array of kits with overlapping purposes. That, he said, makes it hard for the user to discern which one is best, while at the same time making it difficult for the vendor to offer quality support. "With the Targeted Design Platform approach, Xilinx is offering a lot more than what folks find in a typical kit," said Moran. "Instead of a car with no tires, we are offering the board equivalent to a Ferrari that has race tires installed, a full tank of gas, is tuned up and ready to go—ready to go fast."

Evan Leal, senior product marketing manager, Targeted Design Platforms, at Xilinx, said that in addition to the board, connectors and design software, Targeted Design Platform kits include multiple types of reference designs, free IP and, in some cases, bundled FMC cards. There is immediate access to IP libraries and a catalog of plug-and-

play FMC daughtercards from Xilinx and third-party vendors. "Because this is our second TDP generation, we already have in place a very formidable ecosystem to support these new kits," said Moran. "Further, with this generation of devices we've made it even easier for customers to get up and running quickly." The hundreds of board, design service, IP and tool vendors that make up the Xilinx Alliance Program (<http://www.xilinx.com/alliance/index.htm>) provide a rich diversity of solutions and applications that enable Xilinx Targeted Design Platforms. Xilinx worked closely with 4DSP Inc., Analog Devices Inc., Avnet, Northwest Logic, The Mathworks, Texas Instruments and Xylon to bring the Kintex-7 FPGA and Virtex-7 FPGA kits to market.

PLUG-AND-PLAY FMC CARDS

In fact, one of the keys to Xilinx's TDP approach is the industry-standard FMC connectors on all kits starting with those supporting Virtex-6 and Spartan®-6 FPGA families. The addition of FMC connectors allows IP vendors, fellow chip makers and design services companies to field daughtercards that designers can use to quickly connect boards to Xilinx kits and test their designs within a system-level context.

Moran said that since its launch with the 6 series in 2009, the TDP

ecosystem now boasts more than 100 FMC cards from over 30 partners. In recent months, Xilinx has held plugfests in which more than 30 of these FMCs have been verified to work with the new 7 series platforms, covering a wide range of applications. In the coming months Xilinx anticipates a greater number of FMC cards will become available.

The 7 series also lends some inherent improvements to the TDP strategy. For example, Moran points out that with the introduction of the 7 series devices last year, Xilinx embraced the ARM® AMBA® AXI4 interface. AXI4 allows users to more easily integrate Xilinx, third-party and internally developed IP into Xilinx 7 series devices. It also allows customers and the Xilinx IP ecosystem to focus on one IP interconnect structure rather than several, so as to refine IP offerings and develop new ones faster.

Another key benefit of the new-generation TDPs comes from the fact that Xilinx implemented all its 7 series devices—the Virtex-7, Kintex-7 and Artix™-7—as well as the Zynq™-7000 Extensible Processing Platform devices on a scalable programmable logic architecture. Moran said this unified architecture allows users to implement their IP and migrate their designs more easily between FPGA families as their design requirements evolve. This

Even the Kintex devices now include Agile Mixed-Signal technology. AMS blocks allow users to monitor and test the internal blocks in the design and even implement analog functions on the FPGA itself instead of on discrete external circuitry. This improves performance, saves board space and reduces the bill of materials.

simplifies and streamlines design reuse and evaluation for end users.

Moran said that another huge bonus of the 7 series is that even the Kintex-7 devices now include Agile Mixed-Signal (AMS) blocks, which allow users to monitor and test the internal blocks in the design and even implement analog functions (such as ADCs) on the FPGA itself, instead of on discrete external circuitry. Doing so improves performance, saves board space and reduces the bill of materials. It also allows users to monitor the power consumption of the device and confirms the superiority of the Xilinx 7 series over competing devices, he said (see cover story, *Xcell Journal* Issue 76).

“This AMS block has been a feature that up until the 7 series we’ve only had in our Virtex devices,” said Moran. “With the 7 series, the AMS block is available in our Kintex devices as well. There will be a whole new group of users that will now be able to reap great benefits from these blocks.”

Moran said the first three 7 series TDPs are loaded with features to help users get up and designing extremely fast. The feature set of each of these kits is quite extensive and can be found on www.xilinx.com. Moran and Leal walked us through the high points of each.

THE VIRTEX-7 FPGA VC707 EVALUATION KIT

The Xilinx Virtex-7 FPGA VC707 Evaluation Kit is the base TDP for designers looking for a combination of break-

through performance, capacity and power efficiency required in the many applications the Virtex-7 FPGA serves, Moran said. These include advanced systems for wired and wireless communications, aerospace and defense, medical

RGMI and SGMII), an SFP/SFP+ transceiver connector, a GTX port (TX, RX) with four SMA connectors and a PCI Express® x8 edge connector.

For parallel connectivity, the kit has two FMC-HPC connectors (eight GTX



Figure 2 – The Xilinx Virtex-7 FPGA VC707 Evaluation Kit is the base TDP for designers looking for a combination of breakthrough performance, capacity and power efficiency required in the many applications the Virtex-7 FPGA serves.

and broadcasting markets (Figure 2).

This kit is based specifically on the VX485T-2 FPGA, a midsize Virtex-7 FPGA device.

In terms of software, it comes with an ISE® Design Suite Logic edition that is device-locked to the Virtex-7 VX485T-2 FPGA. For serial connectivity, it has Gigabit Ethernet (GMII,

transceivers, 160 single-ended or 80 differential user-defined signals). For memory, it includes SODIMM DDR3 memory at 1,600 Mbps, a 1-Gbit (128-Mbyte) BPI flash for PCIe® configuration, an SD card interface and an 8-kbyte IIC EEPROM.

Additional connectivity includes HDMI video out, a UART-to-USB

The ROHS-compliant Kintex-7 FPGA KC705 Evaluation Kit includes a number of high-impact and cost-saving features. It's built for maximum flexibility, to accelerate a broad range of applications.

bridge, a 2x16 LCD header, 8x LEDs, IIC and an analog mixed-signal port. "This evaluation kit comes with an AMS evaluation card to help designers quickly assess the value of the AMS feature," said Leal.

The kit also comes with reference designs and demonstrations, along with comprehensive documentation to allow users to get to work immediately. Example designs and demonstrations include a board diagnostic demo, ChipScope™ Pro Serial I/O Toolkit, IBERT transceiver test design, multi-boot reference design, PCI Express Gen 2 (x8) test design and DDR3 memory interface design.

In terms of documentation, the kit includes a Getting Started Guide, Hardware User's Guide and Reference Design and Example Guide. It also comes with schematics and UCF files, providing the information needed to accelerate board layout and development based upon best practices.

For further details about the board such as device configuration, clocking, controls and power, visit the Virtex-7 FPGA VC707 Evaluation Kit TDP kit page at <http://www.xilinx.com/products/boards-and-kits/EK-V7-VC707-G.htm>.

KINTEX-7 FPGA KC705 EVALUATION KIT

The base TDP for the Kintex-7 family is the Kintex-7 FPGA KC705 Evaluation Kit, featuring an XC7K325T-FF900-2 FPGA. Moran said the kit is built for maximum flexibility to help designers accelerate development of a broad range of applications, including radio/baseband, radar, Edge QAM, triple-rate SDI and others that demand

power-efficient, high-speed communications and processing (Figure 3).

The ROHS-compliant Kintex-7 FPGA KC705 Evaluation Kit includes a number of high-impact and cost-saving features. It comes with the ISE Design Suite: Logic Edition device-locked to the Kintex-7 XC7K325T FPGA.

Serial connectivity includes Gigabit Ethernet, SFP/SFP+ transceiver connector, a GTX port (TX, RX) with four SMA

connector, 8x LEDs, IIC, LCD header and an analog mixed-signal port.

Memory includes SODIMM DDR3 memory at 1,600 Mbps, a 1-Gbit (128-Mbyte) BPI flash for PCIe configuration, an SDIO-SD card interface, a 16-Mbyte quad SPI flash and an 8-kbyte IIC EEPROM.

The kit includes a board diagnostic demo, a ChipScope Pro Serial I/O Toolkit, an IBERT transceiver test



Figure 3 – The Kintex-7 FPGA KC705 Evaluation Kit is aimed at a broad range of applications that demand power-efficient, high-speed communications and processing.

connectors, a UART-to-USB bridge and a PCI Express x8 edge connector.

Parallel connectivity includes an FMC-HPC connector (four GTX transceivers, 116 single-ended or 58 differential—34 LA and 24 HA—user-defined signals) and an FMC-LPC connector (one GTX transceiver, 68 single-ended or 34 differential user-defined signals).

Additional connectivity includes HDMI video out, a 2x16 LCD display

design, a multiboot reference design, a PCI Express Gen 2 (x8) test design and a DDR3 memory interface design. "Further, this evaluation kit comes with a targeted reference design featuring PCIe x4 Gen 2 and DDR3 to enable designers to get started quickly integrating some of the most popular features used by FPGA designers," said Leal.

Documentation includes a Getting Started Guide, Hardware User's Guide,



Figure 4 – The Kintex-7 FPGA DSP Kit from Avnet targets customers looking to use FPGAs to do the job of multiple DSPs, and do it faster.

Reference Design and Example User Guide, along with schematics and UCF files. “Like the VC707, the KC705 Evaluation Kit comes with an AMS evaluation card to help designers quickly assess the value of the AMS feature,” said Leal.

It also comes with a number of cables and adapters (namely, a universal 12-volt power supply, two USB cables, one Ethernet cable and one HDMI-to-DVI adapter).

For further details about the board such as device configuration, clocking, controls and power, visit the Kintex-7 FPGA KC705 Evaluation Kit TDP kit page at <http://www.xilinx.com/products/boards-and-kits/EK-K7-KC705-G.htm>.

THE KINTEX-7 FPGA DSP KIT

In addition to these base platforms, Avnet is releasing the Xilinx Kintex-7 FPGA DSP Kit Targeted Design Platform for customers looking to use FPGAs to do the job of multiple DSPs—and do it faster. Built around the Kintex-7 KX325T FPGA, the TDP includes an integrated high-speed analog FMC to interface to real-world signals (Figure 4).

The high-speed analog daughter-card is the FMC150 by 4DSP. This card provides two 14-bit, 250-Msample/second A/D channels and two 16-bit, 800-MSPS D/A channels that can be clocked by an internal clock source or an externally supplied sample clock.

The Kintex-7 DSP kit ships with a targeted DSP reference design that includes a high-speed analog interface with digital upconversion (DUC) and digital downconversion (DDC). End users can place it right into their designs, saving weeks of development time.

For further details about the board such as device configuration, clocking, controls and power, visit the Kintex-7 FPGA DSP Kit page at <http://www.xilinx.com/products/boards-and-kits/AES-K7DSP-325T-G.htm>.

In the coming months, Xilinx will be releasing several new boards in support of its 7 series devices. To see what boards are available today and to learn more, visit <http://www.xilinx.com/products/boards-and-kits/index.htm>. To learn more about FMC daughter-cards, visit http://www.xilinx.com/products/boards_kits/fmc.htm. •

GigaBee Spartan-6 LX



- Scalable: 45K to 150K Logic Cells
- Gigabit Ethernet MAC and PHY
- 2 Independent DDR3 Memory Banks
- LVDS IO/s
- Very Small: 40 mm x 50 mm

TE0320 Spartan-3A DSP



- High-Speed USB 2.0
- 32-bit, 128 MByte DDR RAM

Common Module Properties

- On Board Power Supply
- Very Low Cost
- Long-Term Available
- Free Reference Designs
- Ruggedized for Industrial Usage
- Customized Versions Available
- Custom Integration Services

Development Services

- Hardware Design
- HDL Design
- Software Development



www.trenz-electronic.de

Embedded Vision: FPGAs' Next Notable Technology Opportunity

By Brian Dipert

Editor-In-Chief
Embedded Vision Alliance
dipert@embedded-vision.com

José Alvarez

Engineering Director, Video Technology
Xilinx, Inc.
jose.alvarez@xilinx.com

Mihran Touriguian

Senior DSP Engineer
BDTI (Berkeley Design Technology, Inc.)
touriguian@bdti.com



A jointly developed reference design validates the potential of Xilinx's Zynq device in a burgeoning application category.

What up-and-coming innovation can help you design a system that alerts users to a child struggling in a swimming pool, or to an intruder attempting to enter a residence or business? It's the same technology that can warn drivers of impending hazards on the roadway, and even prevent them from executing lane-change, acceleration and other maneuvers that would be hazardous to themselves and others. It can equip a military drone or other robot with electronic "eyes" that enable limited-to-full autonomous operation. It can assist a human physician in diagnosing a patient's illness. It can uniquely identify a face, subsequently initiating a variety of actions (automatically logging into a user account, for example, or pulling up relevant news and other information), interpreting gestures and even discerning a person's emotional state. And in conjunction with GPS, compass, accelerometer, gyroscope and other features, it can deliver a data-augmented presentation of a scene.

The technology common to all of these application examples is embedded vision, which is poised to enable the next generation of electronic-system success stories. Embedded vision got its start in traditional computer vision applications such as assembly line inspection, optical character recognition, robotics, surveillance and military systems. In recent years, however, the decreasing costs and increasing capabilities of key technology building blocks have broadened and accelerated vision's penetration into key high-volume markets.

Driven by expanding and evolving application demands, for example, image sensors are making notable improvements in key attributes such as resolution, low-light performance, frame rate, size, power consumption and cost. Similarly, embedded vision applications require processors with high performance, low prices, low power consumption and flexible programmability, all ideal attributes that are increasingly becoming a reality in numerous product implementation forms. Similar benefits are being accrued by latest-generation optics systems, lighting modules, volatile and nonvolatile memories, and I/O standards. And algorithms are up to the challenge, leveraging these hardware improvements to deliver more robust and reliable analysis results.

Embedded vision refers to machines that understand their environment through visual means. By "embedded," we're referring to any image-sensor-inclusive system that isn't a general-purpose computer. Embedded might mean, for example, a cellular phone or tablet computer, a surveillance system, an earth-bound or flight-capable robot, a vehicle containing a 360° suite of cameras or a medical diagnostic device. Or it could be a wired or wireless user interface peripheral; Microsoft's Kinect for the Xbox 360 game console, perhaps the best-known example of this latter category, sold 8 million units in its first two months on the market.

THE FPGA OPPORTUNITY: A CASE STUDY

A diversity of robust embedded vision processing product options exist:

microprocessors and embedded controllers, application-tailored SoCs, DSPs, graphics processors, ASICs and FPGAs. An FPGA is an intriguing silicon platform for realizing embedded vision, because it approximates the combination of the hardware attributes of an ASIC—high performance and low power consumption—with the flexibility and time-to-market advantages of the software algorithm alternative running on a CPU, GPU or DSP. Flexibility is a particularly important factor at this nascent stage in embedded vision's market development, where both rapid bug fixes and feature set improvements are the norm rather than the exception, as is the desire to support a diversity of algorithm options. An FPGA's hardware configurability also enables straightforward design adaptation to image sensors supporting various serial and parallel (and analog and digital) interfaces.

The Embedded Vision Alliance is a unified worldwide alliance of technology developers and providers chartered with transforming embedded vision's potential into reality in a rich, rapid and efficient manner (see sidebar). Two of its founding members, BDTI (Berkeley Design Technology, Inc.) and Xilinx, partnered to co-develop a reference design that exemplifies not only embedded vision's compelling promise but also the role that FPGAs might play in actualizing it. The goal of the project was to explore the typical architectural decisions a system designer would make when creating highly complex intelligent vision platforms containing elements requiring intensive hardware processing and complex software and algorithmic control.

BDTI and Xilinx partitioned the design so that the FPGA fabric would handle digital signal-processing-intensive operations, with a CPU performing complex control and prediction algorithms. The exploratory implementation described here connected the CPU board to the FPGA board via an

Ethernet interface. The FPGA performed high-bandwidth processing, with only metadata interchanged through the network tether. This project also explored the simultaneous development of hardware and software, which required the use of accurate simulation models well ahead of the final FPGA hardware implementation.

PHASE 1: ROAD SIGN DETECTION

This portion of the project, along with the next phase, leveraged two specific PC-based functions: a simulation model of under-development Xilinx video IP blocks, and a BDTI-developed processing application (Figure 1). The input data consisted of a 720p HD resolution, 60-frame/second (fps) YUV-encoded video stream representing the images that a vehicle's front-facing camera might capture. And the goal was to identify (albeit not "read" using optical character recognition, although such an added capability

would be a natural extension) four types of objects in the video frames as a driver-assistance scheme:

- Green directional signs
- Yellow and orange hazard signs
- Blue informational signs, and
- Orange traffic barrels

The Xilinx-provided IP block simulation models output metadata that identified the locations and sizes of various-colored groups of pixels in each frame, the very same metadata generated by the final hardware IP blocks. The accuracy of many embedded vision systems is affected by external factors such as noise from imaging sensors, unexpected changes in illumination and unpredictable external motion. The mandate for this project was to allow the FPGA hardware to process the images and create metadata in the presence of external disturbances with parsimonious

use of hardware resources, augmented by predictive software that would allow for such disturbances without decreasing detection accuracy.

BDTI optimized the IP blocks' extensive set of configuration parameters for the particular application in question, and BDTI's postprocessing algorithms provided further refinement and prediction capabilities. In some cases, for example, the hardware was only partially able to identify the objects in one frame, but the application-layer software continued to predict the location of the object using tracking algorithms. This approach worked very well, since in many cases the physical detection may not be consistent across time. Therefore, the software intelligent layer is the key to providing consistent prediction.

As another example, black or white letters contained within a green highway sign might confuse the IP blocks' generic image-analysis functions, there-

EMBEDDED VISION ALLIANCE SEES SUCCESS

Embedded vision technology has the potential to enable a wide range of electronic products that are more intelligent and responsive than before, and thus more valuable to users. It can add helpful features to existing products. And it can provide significant new markets for hardware, software and semiconductor manufacturers. The Embedded Vision Alliance, a unified worldwide organization of technology developers and providers, will transform this potential into reality in a rich, rapid and efficient manner.

The alliance has developed a full-featured website, freely accessible to all and including (among other things) articles, videos, a daily news portal and a multisubject discussion forum staffed by a diversity of technology experts. Registered website users can receive the alliance's monthly e-mail newsletter; they also gain access to the Embedded Vision Academy, containing numerous tutorial presentations, technical papers and file downloads, intended to enable new players in the embedded vision application space to rapidly ramp up their expertise.

Other envisioned future aspects of the alliance's charter may include:

- The incorporation, and commercialization, of technology breakthroughs originating in universities and research laboratories around the world,
- The codification of hardware, semiconductor and software standards that will accelerate new technology adoption by eliminating the confusion and inefficiency of numerous redundant implementation alternatives,
- Development of robust benchmarks to enable clear and comprehensive evaluation and selection of various embedded vision system building blocks, such as processors and software algorithms, and
- The proliferation of hardware and software reference designs, emulators and other development aids that will enable component suppliers, systems implementers and end customers to develop and select products that optimally meet unique application needs.

For more information, please visit www.embedded-vision.com. Contact the Embedded Vision Alliance at info@embedded-vision.com and (510) 451-1800. – Brian Dipert

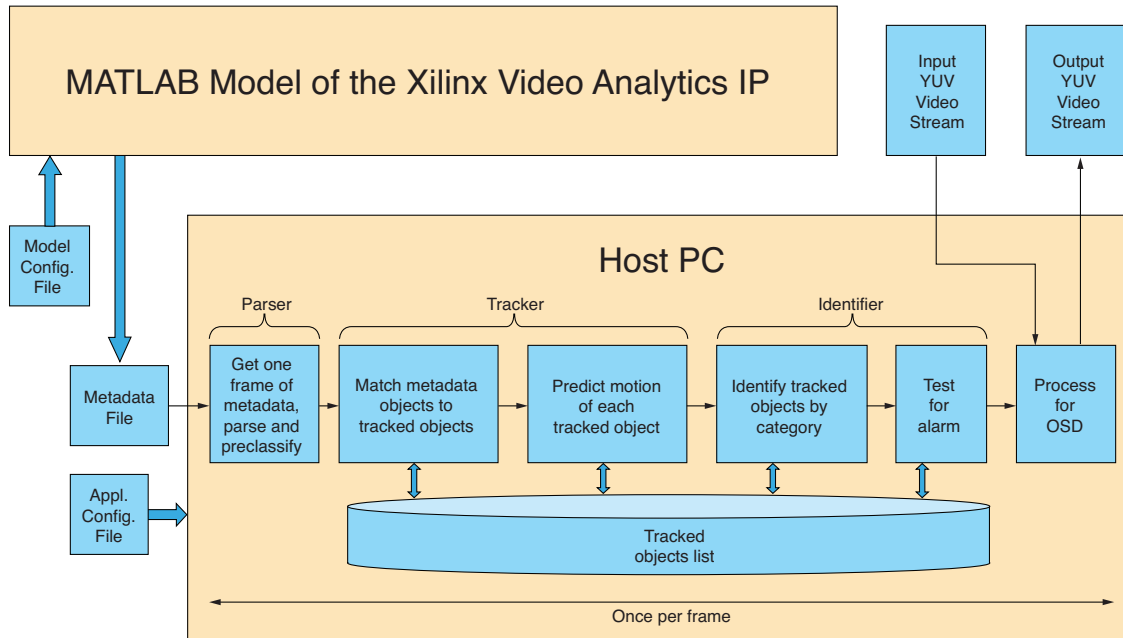


Figure 1 – The first two phases of BDTI and Xilinx's video-analytics proof-of-concept reference design development project ran completely on a PC.

PHASE 2: PEDESTRIAN DETECTION AND TRACKING

In the first phase of this project, the camera was in motion but the objects (that is, signs) being recognized were stationary. In the second phase targeting security, on the other hand, the camera was stationary but objects (people, in this case) were not. Also, this time the video-analytics algorithms were unable to rely on predetermined colors, patterns or other object characteristics; people can wear a diversity of clothing, for example, and come in various shapes, skin tones and hair colors and styles (not to mention might wear head-obscuring hats, sunglasses and the like). And the software was additionally challenged with not only identifying and tracking people but also generating an alert when an individual traversed a digital “trip wire” and was consequently located in a particular region within the video frame (Figure 3).

The phase 2 hardware configuration was identical to that of the earlier phase 1, although the software varied; a video stream fed simulation models

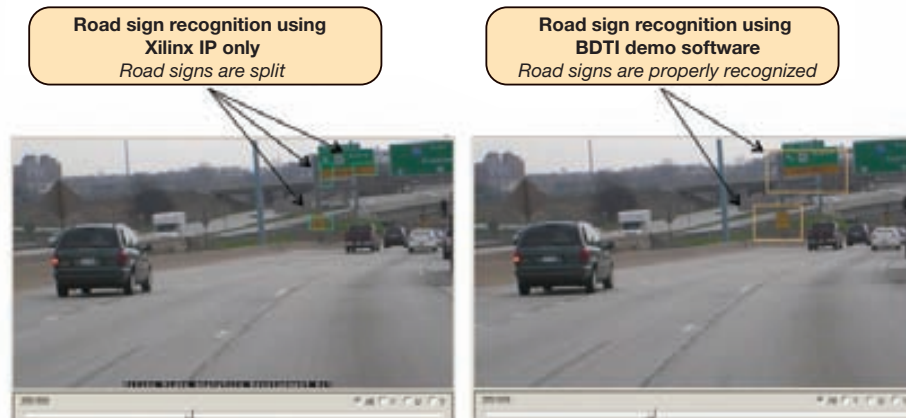


Figure 2 – Second-level, application-tailored algorithms refined the metadata coming from the FPGA's video-analysis hardware circuits.

by incorrectly subdividing the sign into multiple-pixel subgroups (Figure 2). The IP blocks might also incorrectly interpret other vehicles' rear driving or brake lights as cones or signs by confusing red with orange, depending on the quality and setup of the imaging sensor used for the application.

The BDTI-developed algorithms therefore served to further process the Xilinx-supplied metadata in an appli-

cation-tailored manner. They knew, for example, what signs were supposed to look like (size, shape, color, pattern, location within the frame and so on), and therefore were able to combine relevant pixel clusters into larger groups. Similarly, the algorithms determined when it was appropriate to discard seemingly close-in-color pixel clusters that weren't signs, such as the aforementioned vehicle brake lights.



Figure 3 – Pedestrian detection and tracking capabilities included a “trip wire” alarm that reported when an individual moved within a bordered portion of the video frame.

of the video-analytics IP cores, with the generated metadata passing to a secondary algorithm suite for additional processing. Challenges this time around included:

- Resolving the fundamental trade-off between unwanted noise and proper object segmentation

- Varying object morphology (form and structure)
- Varying object motion, both person-to-person and over time with a particular person
- Vanishing metadata, when a person stops moving, for example, is blocked by an intermediary

object or blends into the background pattern

- Other objects in the scene, both stationary and in motion
- Varying distance between each person and the camera, and
- Individuals vs. groups, and dominant vs. contrasting motion vectors within a group

With respect to the “trip wire” implementation, four distinct video streams were particularly effective in debugging and optimizing the video-analytics algorithms:

- “Near” pedestrians walking and reversing directions
- “Near” pedestrians walking in two different directions
- A “far” pedestrian with a moving truck that appeared, through a trick of perspective, to be of a comparable size, and
- “Far” pedestrians with an approaching truck that appeared larger than they were

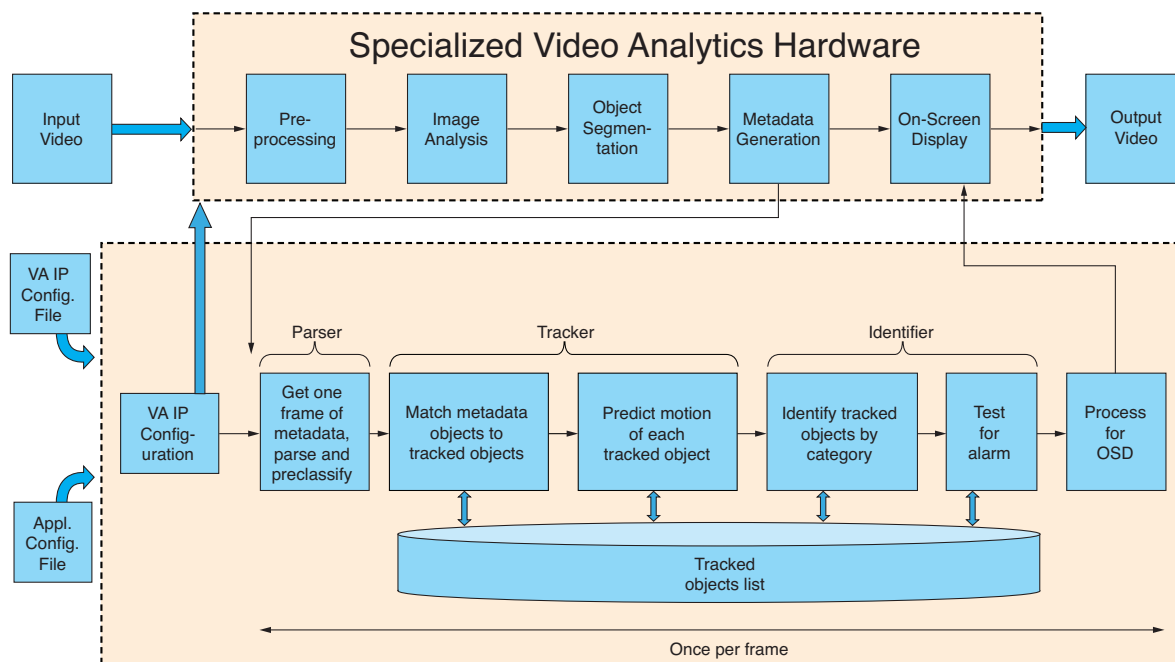


Figure 4 – The final phase of the project migrated from Xilinx’s simulation models to actual FPGA IP blocks. BDTI also ported the second-level algorithms from an x86 CPU to an ARM-based SoC, thereby paving the path for the single-chip Zynq Extensible Processing Platform successor.

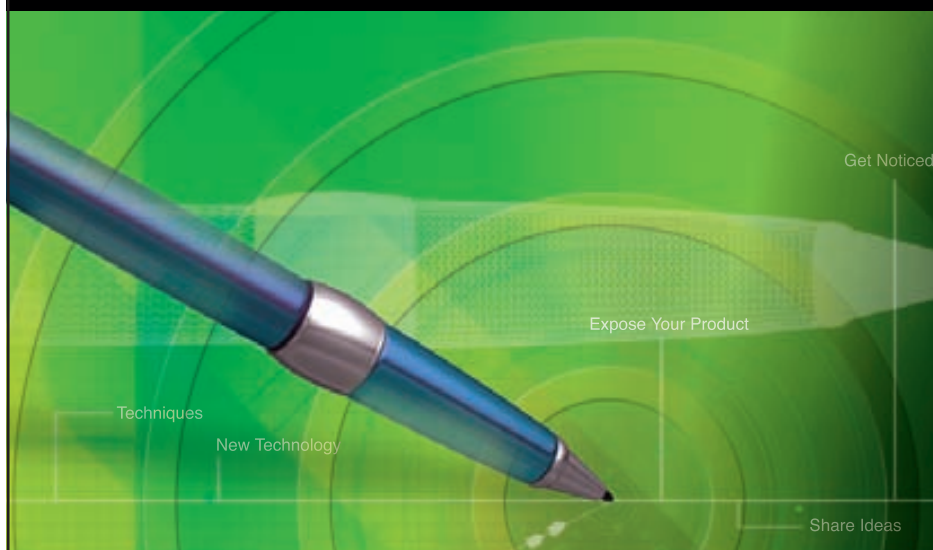
PHASE 3: HARDWARE CONVERSIONS AND FUTURE EVOLUTIONS

The final portion of the project employed Xilinx's actual video-analytics IP blocks (in place of the earlier simulation models), running on the Spartan®-3A 3400 Video Starter Kit. A MicroBlaze™ soft processor core embedded within the Spartan-3A FPGA, augmented by additional dedicated-function blocks, implemented the network protocol stack. That stack handled the high-bit-rate and Ethernet-packetized metadata transfer to the BDTI-developed secondary processing algorithms, now comprehending both road sign detection and pedestrian detection and tracking. And whereas these algorithms previously executed on an x86-based PC, BDTI successfully ported them to an ARM® Cortex™-A8-derived hardware platform called the BeagleBoard (Figure 4).

Those of you already familiar with Xilinx's product plans might right now be thinking of the Zynq™ Extensible Processing Platform, which combines the FPGA and Cortex-A8 CPU on a single piece of silicon. Might it be possible to run the entire video-analytics reference design on a single Zynq device? The likely answer is yes, since the Zynq product family includes devices containing sufficient programmable logic resources, and since the BDTI algorithms put only a moderate load on the ARM CPU core.

Embedded vision is poised to become the next notable technology success story for both systems developers and their semiconductor and software suppliers. As the case study described in this article suggests, FPGAs and FPGA-plus-CPU SoCs can be compelling silicon platforms for implementing embedded vision processing algorithms. ●●

GET PUBLISHED



WOULD YOU LIKE TO WRITE FOR XCELL PUBLICATIONS?

It's easier than you think!

Submit an article draft for our Web-based or printed publications and we will assign an editor and a graphic artist to work with you to make your work look as good as possible.

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

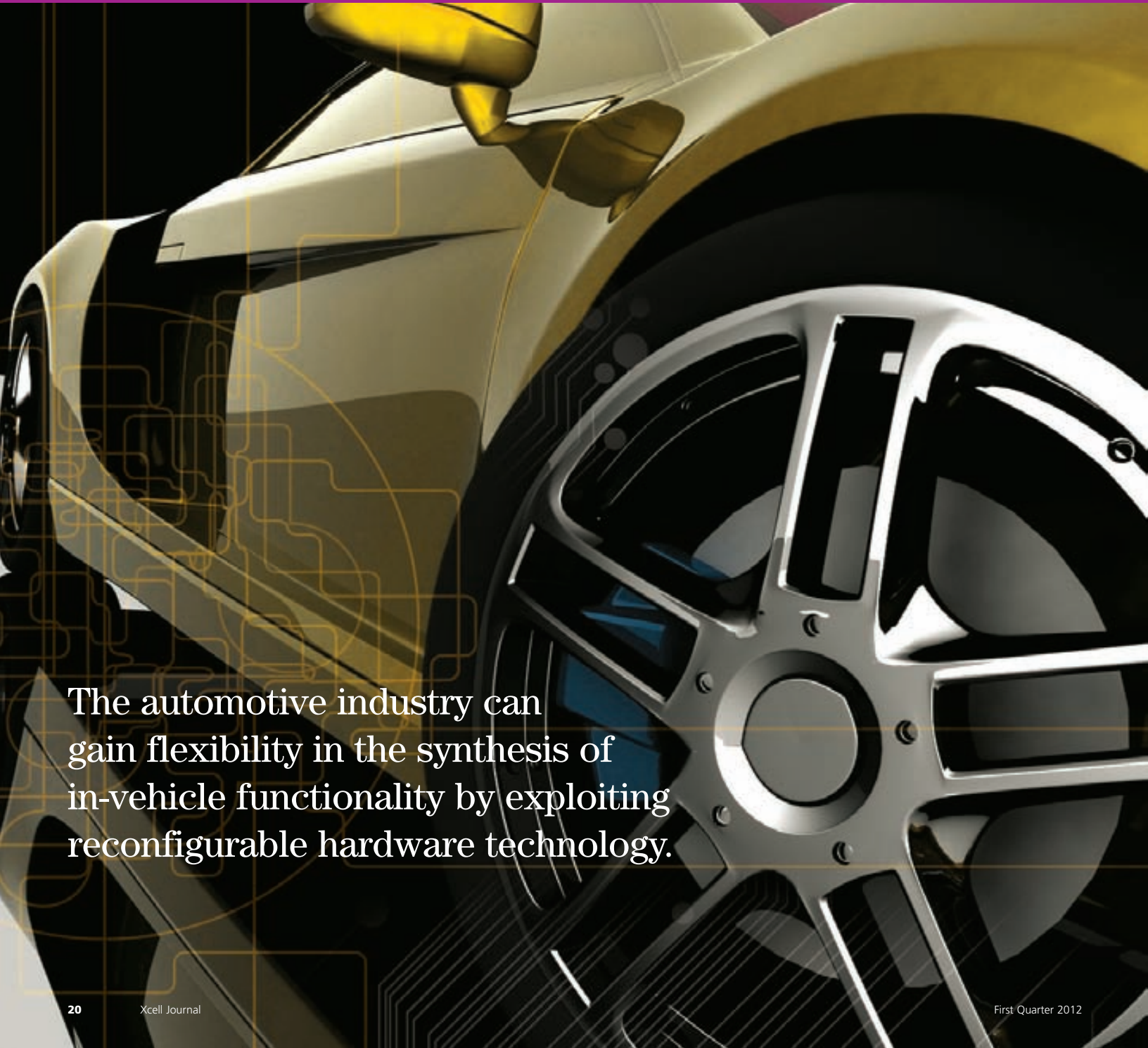
Mike Santarini
Publisher, Xcell Publications
xcell@xilinx.com



See all the new publications on our website.

www.xilinx.com/xcell

FPGA-based Automotive ECU Design Addresses AUTOSAR and ISO 26262 Standards



The automotive industry can gain flexibility in the synthesis of in-vehicle functionality by exploiting reconfigurable hardware technology.

by Francisco Fons

PhD Candidate
University Rovira i Virgili
Tarragona, Spain
francisco.fons@estudiants.urv.cat

Mariano Fons

PhD Candidate
University Rovira i Virgili
Tarragona, Spain
mariano.fons@estudiants.urv.cat



Automakers today are adding ever more advanced functions into the electronic control units (ECUs) distributed in their automobiles to improve the driving experience, enhance safety and, of course, outsell the competition. In such a landscape, the Automotive Open System Architecture (AUTOSAR) initiative and the international functional-safety standard ISO 26262 are fast forming the technical and architectural underpinnings of automotive ECU design.

As the density of automotive electronics increases to satisfy the growing functional demands of new vehicles, FPGA vendors are delivering bigger devices capable of integrating full applications, but that use less power and come in at more competitive prices than prior-generation devices. This trend points the way toward the further use of reconfigurable computing technology in the automobile industry.

We have devised a pioneering approach to designing an automotive ECU using a programmable FPGA device rather than an MCU-based platform as the foundation for an ECU that conforms to both the AUTOSAR and ISO 26262 standards. Our approach explores key features such as parallelism, customization, flexibility, redundancy and versatility of the reconfigurable hardware. After designing the concept, we hope to implement it in a prototype. For this purpose, the Xilinx® Zynq™-7000 Extensible Processing Platform—which combines a hard ARM® dual-core

Cortex™-A9 MPCore processor and a 28-nanometer Xilinx 7 series programmable logic device equipped with dynamic partial-reconfiguration capability—is an excellent candidate. This FPGA platform meets the needed requirements and also features on-chip communication controllers commonly used in the vehicle networks, like CAN and Ethernet.

NOVEL APPLICATIONS

The computational power present in an automobile is nowadays distributed through ECUs interconnected via a communication network. Over the coming years, such computation power is expected to rise due to the emergence of novel applications for motor vehicles. These include safety and driver-assistance features, car-to-car communications, comfort and control functions, entertainment and the large spectrum of hybrid-electrical technologies. No wonder, then, that the electronic content of vehicles is expected to grow. Analysts predict that the market for semiconductors in automotive applications will increase at a compound annual growth rate (CAGR) of 8 percent in the next five years. One of the fastest-growing segments relates to microcontrollers (MCUs) and programmable logic devices such as field-programmable gate arrays (FPGAs).

While in-vehicle functions are increasing in number and sophistication, automotive manufacturers have seen the need to address the growing complexity of designing and managing these systems in an effective way. As a result, today both the AUTOSAR and ISO 26262 standards influence the way hardware and software systems are architected, designed and deployed in real automotive ECUs (see sidebar).

Founded by automakers in 2003, AUTOSAR aims to define a standard system software architecture for ECUs distributed in the vehicle. The goals of ISO 26262, meanwhile, center on functional safety—essentially, the

avoidance, or detection and handling, of faults in order to mitigate their effects and prevent the violation of any established system safety goal. Functional safety turns out to be one of the key issues in automobile development as new safety-critical functions arise, like driver assistance or dynamic control. The standard, which was ratified in 2011, supports the safe development of software and hardware.

The entire ECU design and development flow is thus governed by standards that demand systematic processes. Our work addresses the design of such a cost-effective embedded computing platform, with reconfigurable hardware technology enabling an optimized system architecture.

SYSTEM ARCHITECTURE

The AUTOSAR and ISO 26262 directives are mainly driven from a software development perspective and oriented toward computing platforms based on microcontroller units. However, the introduction of hardware/software co-design and reconfigurable computing techniques can bring some advantages in this arena. While standard MCUs are often the hardware platform of choice in automotive ECUs, the decreasing cost of new FPGAs, along with the fact that some of them harbor hard-core processors inside, makes these devices a solid solution for a massive deployment in this market. Moreover, the automotive trend of continually incorporating new embedded functionality points to a need for parallel computing architectures. That is particularly true in the infotainment sector today, where high-speed digital signal processing is opening doors to FPGA technology. Programmable logic suppliers like Xilinx and EDA tool vendors such as MathWorks show clear interest in this field.

Aiming to bring all the advantages of reconfigurable hardware to automotive applications, we describe the potential of this technology through

a use case focused on one of the most important ECUs found in the automobile computing network concerning deployment of end-user functions: a body controller module. This ECU, also called a body domain controller, is responsible for synthesizing and controlling the main electronic body functions found in the vehicle, such as the windshield wiper/washer, lighting, window lift, engine start/stop, exterior mirrors and central locking. Our goal was to design an AUTOSAR-compliant ECU system equipped with safety-critical functions on an FPGA platform.

from the software. In this way, AUTOSAR boosts the reuse of software by defining interfaces that are hardware independent. In other words, a software component written in accordance with the AUTOSAR standard should work on any vendor's microcontroller, provided it has been properly integrated into an AUTOSAR-compliant runtime environment.

This feature delivers increased flexibility to the automaker. Car manufacturers can exchange equivalent versions of the same software module developed by different suppliers throughout their vehicle platforms in a

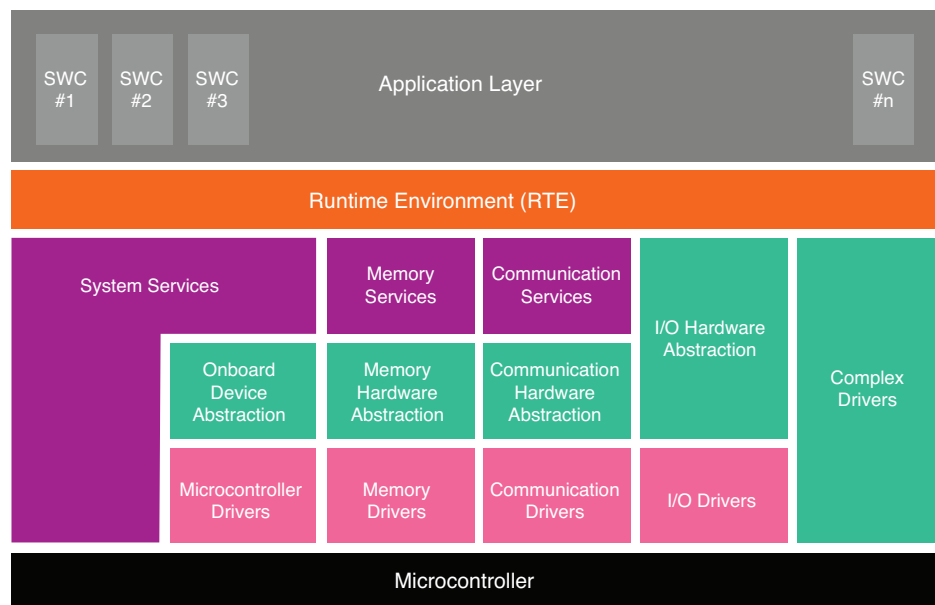


Figure 1 – The AUTOSAR layer-based model, from MCU to application layer

REAL-WORLD SCENARIO

The standardization of the ECU system architecture that AUTOSAR promotes is unavoidable if automakers are to manage the increasing functional complexity of vehicles in a cost-efficient way. It enables the high-level integration of functions distributed in ECUs and the reuse of software components. The main goal of AUTOSAR is to define a uniform ECU architecture that decouples the hardware

transparent way, thanks to inherent plug-and-play characteristics, and without causing side effects in the behavior of the rest of the functions within the vehicle. In the end, hardware and software are highly independent of each other. This decoupling occurs by means of abstracted layers interconnected through standard software APIs. Figure 1 shows the functional-layers breakdown that AUTOSAR defines.

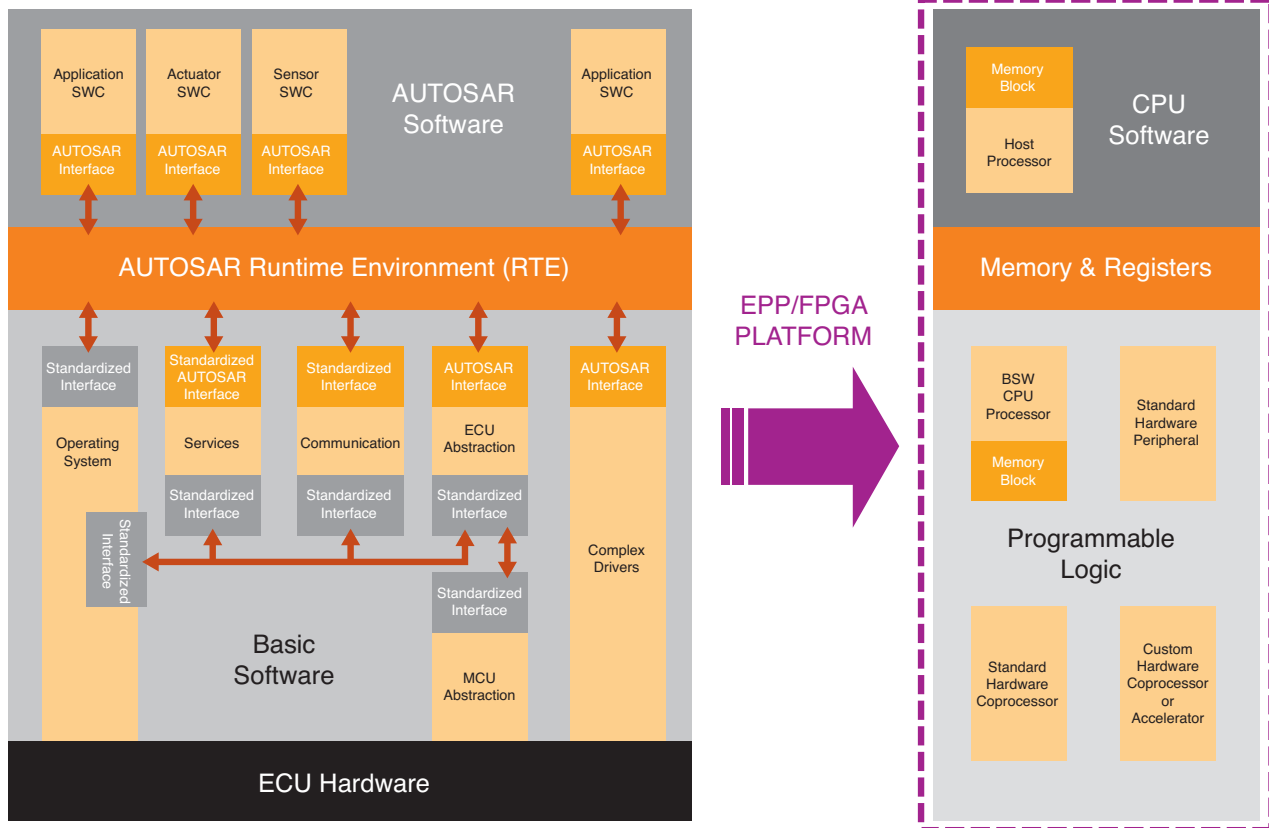


Figure 2 – Porting of the AUTOSAR ECU architecture to an FPGA platform

At the bottom, in black, is the hardware or physical layer, consisting of the MCU itself—that is, the CPU and some standard peripherals attached to it. Above the microcontroller, the basic software (BSW) is decomposed in three layers: microcontroller abstraction layer (MCAL) in pink, ECU abstraction layer (ECUAL) and complex drivers in green, and services layer (SRV) in purple. These three layers are organized, in turn, into several columns or stacks (memory, communication, input/output and so on).

Close to the hardware components is the microcontroller abstraction layer. As its name suggests, this layer abstracts the MCU. The goal is to have a hardware-independent API that handles the hardware peripherals present in the microcontroller. Next up, the ECU abstraction layer abstracts the other smart devices placed in the ECU board, typically in contact with the MCU (for example, system voltage regulator,

smart switching controllers, configurable communication transceivers and the like). Next, the third layer is the services layer. This layer is almost hardware independent and its role is to handle the different types of background services needed. Examples are network services, NVRAM handling or management of the system watchdog. With these three layers, AUTOSAR defines a set of basic software functions that sustain, under a specific hardware platform, all the functionality conceived from higher levels of abstraction to the automotive ECU.

The fourth layer, the runtime environment (RTE), provides communication services to the application software. It is composed of a set of signals (sender/receiver ports) and services (client/server ports) accessible from both the upper layer of the BSW and the application layer (APP). The RTE abstracts the application from the basic software and it clearly delimits

the line of the software-stacked architecture that separates the generic and exchangeable software code (APP) from the particular and hardware-dependent code (BSW). In other words, the RTE makes it possible to isolate the software application from the hardware platform; therefore, all software modules running above the RTE are platform-independent.

Above the RTE, the software architecture style changes from layered to component-based through the application layer. The functionality is mainly encapsulated in software components (SWCs). Hence, standardization of the interfaces for AUTOSAR software components is a central element to support scalability and transferability of functions across ECUs of different vehicle platforms. The standard clearly specifies the APIs and the features of these modules, except for the complex drivers. The SWCs communicate with

other modules (inter- or intra-ECU) only via the runtime environment.

As ECUs continue to integrate ever more functionality, FPGA devices can be a sensible alternative to single- or multicore MCUs. This overview of the different AUTOSAR layers may hint at the benefits that designers can extract from this architecture when deploying it in programmable logic. Let's take a closer look at how our design could

peripherals and coprocessors can coexist in hardware and be totally or partially managed in software.

Dedicated coprocessors or core processors are suitable from the point of view of functional safety too, since they can implement functionality with inherent freedom of interference in hardware, bringing a high level of flexibility—and even redundancy when required—to the system design. Also,

components, which implement the applications and communicate with the RTE through AUTOSAR interfaces.

Due to the inherent complexity of the AUTOSAR architecture, its deployment demands powerful embedded computing platforms. Today, the typical ECU implementation is based on a 32-bit single-core processor on an MCU platform. However, more and more, a single core is not going to be

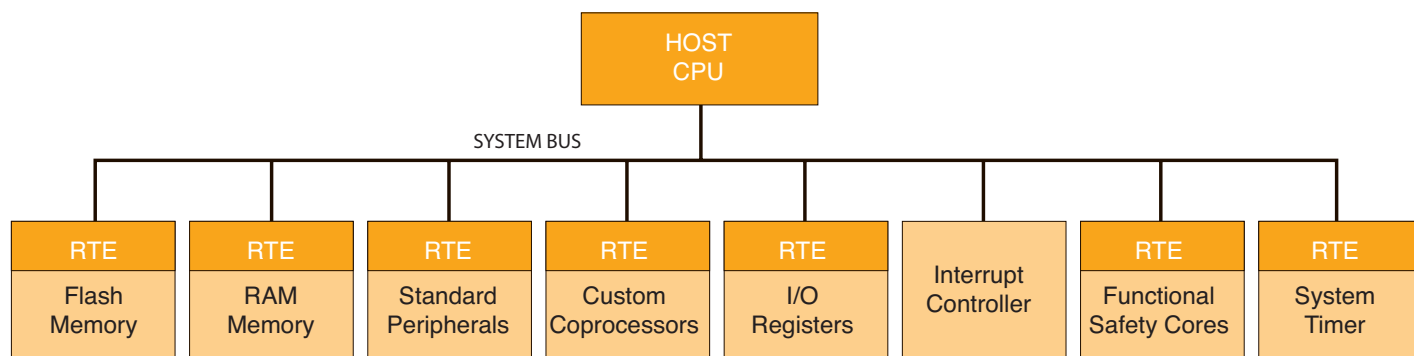


Figure 3 – Block diagram of an automotive ECU deployed in an FPGA

potentially implement a solution based on custom static hardware (flash- or SRAM-based FPGA technology), and then extend this approach to a runtime reconfigurable hardware implementation (SRAM-based partially reconfigurable FPGA).

ECU DESIGN ON FPGA-BASED STATIC HARDWARE

The AUTOSAR architecture fits well in an embedded system composed of a CPU, memory and programmable logic. The ECU platform requires one CPU or host processor to manage the application and process the different functions distributed in software components through the APP layer. At the same time, the MCU layer and part of the basic software layer can be synthesized in hardware in the programmable logic fabric. Hence, in addition to implementing standard peripherals attached to the CPU, other custom

the intermediate RTE layer can be synthesized in RAM blocks distributed along the FPGA or in flip-flops embedded in the logic cells of the device, as well as external memory. Moreover, it's easy to design the RTE signal interfaces to allow both read and write operations (via single-port memories) or to restrict the architecture to either read or write transactions only (by means of single dual-port memories with two independent read and write ports) as a protective measure against interference, like the counterpart sender and receiver software ports that AUTOSAR defines.

Figure 2 shows the proposed porting of the MCU-based AUTOSAR ECU architecture into an Extensible Processing Platform (EPP) or FPGA device, keeping a clear system partitioning in layers. Below the RTE layer are the operating system (OS), memory stack, communication stack, I/O stack and so on. Above it are the software

enough to deliver all the computational power demanded. Yet the use of multicore CPUs can degrade performance if they share the program/data memory through a multiprocessor bus with arbitration mechanisms, often necessitating a highly complex solution.

Instead, we propose a new alternative based on programmable logic and composed of only one single-core processor playing the role of host CPU but surrounded by more-intelligent peripherals, coprocessors or even slave processors. All these computing units can be instantiated in the FPGA fabric as new soft-core processors, such as the Xilinx PicoBlaze™ and MicroBlaze™, which run their own code from dedicated RAM blocks of the FPGA (separate soft-core processors with dedicated program memories), or as made-to-measure hardware accelerators. In both cases, the topology is one host CPU with smart peripherals that offload

The FPGA approach can attain system performance comparable to that of a multiprocessor platform but with the simplicity of a single-core processor, thanks to the use of powerful and autonomous custom coprocessors that work in parallel with the host processor.

some of the CPU's tasks, reducing the complexity of the system. Thus, the host CPU manages the whole APP layer in software while the custom peripherals take charge of the BSW layer and run independently of each other, in parallel and autonomously. Moreover, it's a good idea to design these peripherals to make the software execution of the host CPU more linear—that is, without creating excessive interrupts coming from the peripherals to request the CPU's attention via interrupt service routines. Figure 3 shows the block diagram of the system and its component breakdown into functional units synthesized in one FPGA device.

This approach can attain system performance comparable to that of a multiprocessor platform but with the level of simplicity (regarding software development and maintenance) of a single-core processor. This trade-off is possible by using dedicated hardware to build more powerful and autonomous custom coprocessors that work in parallel with the host processor.

Conceptually, to simplify the idea, it is possible to split the architecture of these systems into two main layers—the high layer and the low layer—separated by the RTE interface. The high layer corresponds to the application layer of AUTOSAR, composed of software components that manage the end-user functions in the vehicle. The low layer comprises the hardware and the basic software up to the RTE link. The application layer can represent, in relative

figures, around 90 percent of the high-level functionality within the vehicle, and all this source code—above the RTE—is reusable.

At the same time, the low layer comprises all those features that grant flexibility and versatility to the high layer. That is, the low layer performs the customization of all that reusable functionality in a particular hardware platform. As such, the high layer is essentially a set of software functions that implement the control of some vehicle loads, sensors and actuators by means of algorithms implemented in finite state machines (FSMs). These algorithms are executed cyclically by a CPU and scheduled in software tasks that the OS controls.

The low layer is also responsible for implementing the drivers of all the standard peripherals attached to the CPU—for example, A/D converter, PWM controller, timer or memory controller—to make the abstraction of the high layer feasible. This low layer involves the management of events that need to be served in real time. In this regard, programmable logic can bring some added value. The idea is to reach a host CPU able to process the application as a simple sequence of software functions not influenced by external events typically derived from hardware, but reading or writing RTE signals periodically to evolve the FSMs accordingly. The low layer would hide these hardware events and manage them, preprocessing them and updating certain signals

in the RTE or performing certain actions in real time as a result, following its specific tasks scheduling.

Attaching custom hardware controllers to the system CPU minimizes the need for shared resources, if such controllers can work in an autonomous way. From an OS point of view, this helps to reduce the system complexity (avoiding arbitration, latencies, retry mechanisms and the like).

Another advantage is that dedicated hardware can more simply implement certain functionality that is typically performed in software through multithreading, since concurrency is a feature more inherent to hardware than to software. Furthermore, the flexible hardware can be used to reduce execution time by hardwiring computationally intensive parts of the algorithm by means of parallel and pipelined hardware implementations instead of sequential software approaches on Von Neumann machines.

You can reduce the software complexity of an automotive ECU by granting a major level of intelligence to the peripherals and hardware coprocessors synthesized in the MCU and BSW layers, freeing up CPU time.

In parallel with the growing complexity of the ECU platforms, the number of I/O lines the system demands is also increasing. In this regard, an FPGA brings a clear advantage over a microcontroller since it typically offers far more user pins. This point is often relevant in MCU-based ECUs, because they need

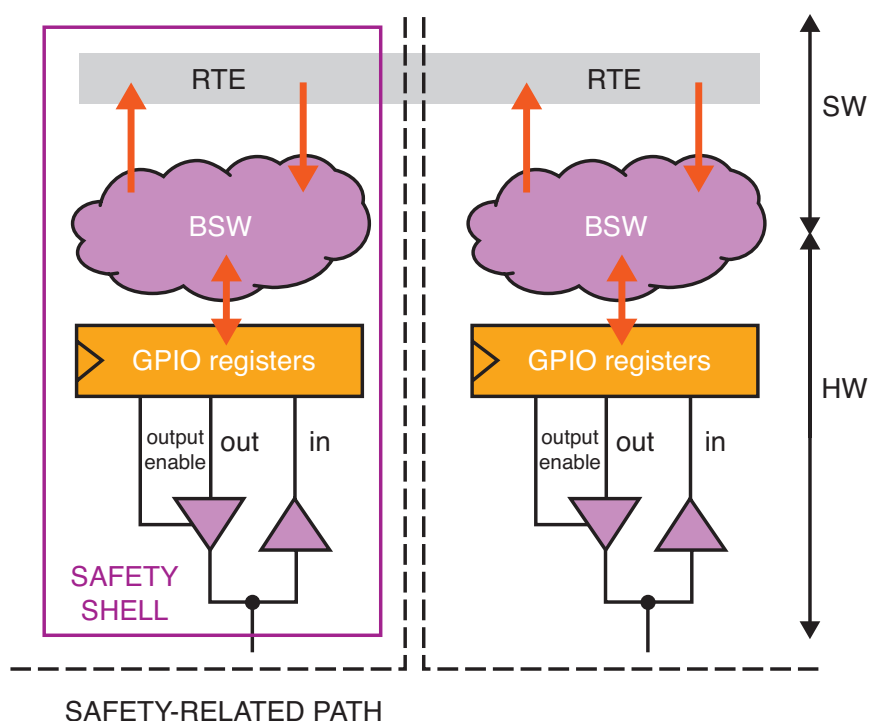


Figure 4 – Hardware/software co-design of a safety shell architecture isolates the safety-relevant ports from non-safety ports to guarantee there will be no interference.

to extend the MCU inputs and outputs with external chips that perform the parallel-serial data conversion, like digital shift registers or analog multiplexers. An FPGA lets you skip all these satellite components, reducing thus the bill of materials as well as the PCB dimensions of the electronic board.

State-of-the-art FPGA devices already incorporate analog-to-digital converters. This feature is interesting in automotive design since many ECUs make use of analog signals (for example, battery voltage) to implement part of the needed functionality. The presence of ADCs in programmable logic devices opens new application fields for FPGAs.

Like MCUs, FPGAs offer remote update capability. However, it is important to note that in this case, the bitstream downloaded into the FPGA relates not only to software code but also to hardware circuitry. This means

that, once the product is in production, it is still possible to change the hardware design by means of system updates or upgrades. The automotive industry appreciates such flexibility, since it also enables bug fixes (in this case, both hardware and software) after the product launch.

In any ECU that embeds a function qualified as safety-relevant under ISO 26262, the hardware and software involved in this implementation must fulfill a certain level of protection depending on how it is categorized. From the software point of view, it is necessary to demonstrate freedom from interference—that is, the non-safety-relevant code running in an ECU must not corrupt the operation of any code inside the same ECU classified as safety-relevant. This isolation is necessary to guarantee correct execution of safety-related and non-safety-related functions running side-by-side on the same processor. Often, it's easier

to manage these measures more flexibly in programmable logic than in MCUs.

Regarding memory protection strategies oriented to functional safety, it is necessary to guarantee write access to certain safety-related signals only from authorized safety software components. In the context of MCU devices, memory partitioning provides a fault-containment technique to separate software applications from each other to avoid any data corruption among them. Programmable logic will likely make it possible to implement a more efficient self-protection mechanism. It is possible to manage the RTE buffer related to safety signals through dedicated single dual-port memories so that the data is written from the write port and read from the read port. In this way, it is possible to implement dedicated hardware controllers that put different restrictions on writing or reading those signals from the software side. The same approach can be implemented with registers.

The possibility of introducing custom hardware solutions in the ECU system is a big advantage of the FPGA approach, especially for safety-related features. In this case, with regard to I/O pins and GPIO controllers, the pinout involved in safety functions can be grouped in made-to-measure I/O ports that are accessed exclusively by safety components inside the ECU, separated from the remaining pins of the device. This is a good way to decouple the safety-critical pins from the non-safety-critical pins of the system, ensuring freedom from interference by design. Any access to non-safety pins cannot corrupt the status of the safety pins, which are managed by safety-relevant code only. This idea is depicted in Figure 4.

Furthermore, it's also possible to tailor the size of each GPIO port to the needs of the application or the software component that handles it, skipping the step of converting a GPIO port into a physical resource that different

A safety strategy based on redundancy is another argument for choosing programmable logic, which makes it possible to instantiate several identical and independent processing engines multiple times in the same device.

applications share, as happens with MCU ports. In this way, in an FPGA each application managed by a different SWC (for example, window lift, wiper, exterior mirror, etc.) can have its specific port mapped in specific registers within the system memory map. In MCU platforms this is often not possible, since the ports have a fixed size (typically 8, 16 or 32 bits wide) and are addressed in a word-wise mode, not bit-wise. Therefore, this control register becomes a shared resource that several SWCs access along the program execution.

We can extend the same strategy used for the GPIO controller to other standard peripherals. In this manner, the function partitioning and isolation that AUTOSAR promotes at the high layer with SWCs can extend also to resources of lower layers by means of programmable hardware. Such a technique is impossible with frozen hardware solutions based on standard MCU devices.

The same decoupling strategy we described for MCU standard peripherals can be applied to all the channels or data paths of a safety function. This feature is particularly interesting as a way of implementing highly categorized safety goals—organized by ASIL level in ISO 26262 (see sidebar)—decomposing them in redundant partitions of a lower ASIL level so that each of these parts, performed in duplicate, is then implemented according to its new level. This safety strategy based on redundancy is another argument for choosing programmable logic, which makes it possible to instantiate several identical

and independent processing engines multiple times in the same device. Moreover, the fulfillment of a certain ASIL level is always clearer and easier to prove with architectural approaches (hardware) than by abstract software, especially features like freedom from interference, where design failures like a stack overflow or an incorrect handling of a data pointer in C programming language could introduce unexpected safety integrity problems to the system.

Another design advantage derived from the flexibility of programmable logic and applicable to functional safety is the possibility of implementing triple-modular redundancy (TMR) strategies. This is a commonly known method for single-event upset (SEU) mitigation in aerospace applications. Such a mitigation scheme has three identical logic circuits that perform the same task in parallel, with the corresponding outputs compared through a majority-voter circuit. Hardware offers a very efficient way to implement this strategy.

Also, in a market where cost and power consumption are huge concerns, some programmable logic devices, such as the Xilinx Zynq-7000 EPP, support several features to lower the overall system power consumption—some of them inherited from MCU devices. Features like the processing system power-on-only mode, sleep mode and peripheral independent clock domains can significantly reduce dynamic power consumption of the device during idle periods.

Certain programmable logic devices come with a hard-core processor

placed in the fabric, enabling designers to initially develop the whole system functionality in software, as they typically would do for an MCU-based platform, and then progressively add more hardware in the design, porting certain parts to programmable logic resources. This methodology enables the designer to build different versions of a solution and realize the advantages of synthesizing some functions in custom hardware with respect to a purely software-based approach.

ECU DESIGN ON RUNTIME RECONFIGURABLE HARDWARE

After exploring the advantages of implementing ECUs in static hardware and software via programmable logic, let's turn to designs that use SRAM-based FPGAs equipped with runtime partial-reconfiguration capability. PR technology offers additional benefits for automotive designers.

One big advantage is the fact that the system startup time can be reduced if the FPGA contains some PR regions that do not need to be configured at startup—for instance, when the ECU wakes up or at power-up. FPGAs that do not support active partial reconfiguration need to configure all the FPGA resources at power-up; runtime reconfigurable FPGAs, however, can be partially reconfigured by downloading a partial bitstream.

The large capacity of today's state-of-the-art FPGA devices results in considerable configuration time overhead to download the full bitstream at power-up. Runtime partial-reconfiguration technology can dramatically reduce this configuration latency. In

that case, it is possible to configure only a minimalist subsystem at power-up (that is, the bootloader and the portion of the system application immediately required) and keep the rest of the system idle until it is necessary to initialize it. Splitting this startup activity into two phases speeds up the initialization process in case the system needs a fast response at power-up or upon waking. For this purpose, the system architecture is decomposed into a static region and one or more partially reconfigurable regions (PRRs). The static region encompasses the

system responsible for carrying out the startup process, typically the host CPU, along with the reconfiguration engine and a data link to the bitstream repository. The other regions are described by specific partial bitstreams that can be downloaded later, when the application needs them.

Also, if PRR regions are disabled, then it is possible to reduce the power consumption of the device proportionally to the portion of area that is not in use. Power-saving modes are especially relevant in automotive battery-powered ECUs. For this reason, automotive MCUs make use of low-power modes to keep

the ECU power consumption to a minimum when the vehicle is inactive (that is, in sleep mode). Analogously, the use of blank bitstreams to disable portions of the FPGA when not required reduces logic activity and consequently, dynamic power consumption.

Automotive designers can also use a technique inherited from aerospace applications in systems based on run-time reconfigurable logic. Configuration scrubbing can recover the system from failures on SRAM resources originated by SEU or electromagnetic interference. Periodically reconfiguring the hardware peripherals guarantees that

TWO CRUCIAL STANDARDS

The automotive industry is already designing electronics with two crucial standards in mind: AUTOSAR, for handling embedded-system complexity by means of appropriate software and hardware architectures, and the upcoming ISO 26262, which governs functional safety. Relevant technical concerns adopted from ISO 26262 and released in AUTOSAR include the detection and handling of safety issues like hardware faults at runtime; abnormal timing and the broken logical order of execution of applications; and data corruption, among others.

INSIDE AUTOSAR

In recent years, electronic components have displaced mechanical and hydraulic systems within vehicles. The incursions continue, as designers begin to implement additional control, monitoring and diagnostic functions in software. In fact, electronics technology makes it possible to deliver new functions whose development would be costly or not feasible if using only mechanical and hydraulic solutions. These parts must meet stringent safety requirements to avoid faulty states and failures.

While software-related product failures have so far been relatively rare, continued growth in the use of software as part of manufactured products increases the system complexity and, together with shorter product development cycles, leads in the end to product failures. The automotive industry has attacked this problem by forming coalitions and enacting standards to ensure the application and creation of safe and reliable software.

The Motor Industry Software Reliability Association (MISRA), for example, is a consortium of vehicle manufacturers such as Ford and Jaguar Land Rover, component suppliers and engineering consultancies. By defining a

series of software programming rules, MISRA seeks to promote best practices in developing safety-related electronic systems in road vehicles and other embedded systems.

The Automotive Open System Architecture, or AUTOSAR, is a partnership of automotive manufacturers, suppliers and other companies from the electronics, semiconductor and software industries working together to develop a de facto open industry standard for automotive electrical/electronic (E/E) architectures, targeting several major issues. These include managing the growing complexity of automotive E/E systems associated with the continuous increase of functionality; improving flexibility for product modification, upgrade and update; improving scalability of solutions within and across product lines; improving the quality and reliability of E/E systems; and enabling the detection of errors in early design phases.

This initiative faces the challenge of having to integrate a growing amount of software and electronic technologies across a vast ecosystem of suppliers. By simplifying the exchange and update options for software and hardware, the AUTOSAR approach forms the basis for reliably controlling the rising complexity of the E/E systems in motor vehicles, as well as improving cost-efficiency without compromising quality.

The AUTOSAR initiative, founded in 2003, is a natural evolution of the older OSEK/VDX consortium—born one decade before and spearheaded by some German and French automakers—but now with more ambitious goals and extended to most of the automotive OEMs all over the world.

The core partners of AUTOSAR are the BMW Group, Bosch, Continental, Daimler, Ford, General Motors, PSA Peugeot Citroën, Toyota and the Volkswagen Group. In addition to these companies, more than 160 other members play an important role in the success of the partnership. Thus, under the slogan “cooperate on standards, compete on

the system is self-repaired in case of a malfunction. Moreover, the maximum time of the malfunction is restricted to the scrubbing period. This idea is commonly applied also in software as a protection measure against interference, for example, periodic reconfiguration of MCU peripherals.

Another promising feature derived from the flexibility of runtime partial-reconfiguration technology is fault recovery by means of function relocation in case of permanent or nonrepairing circuit failures placed in one specific 2D position of the FPGA resources, affecting for instance one

specific logic cell or RAM block. Once a hardware or software fault has been identified, it is possible to automatically relocate the required functionality to another part of the programmable logic device inside the same ECU, or even relocate it to another ECU at runtime. Although conceptually feasible, this feature is not totally supported by automatic tools today.

The most powerful characteristic of runtime reconfigurable computing technology applicable in the automotive sector is, beyond doubt, the time-multiplexing of functionality on shared hardware resources on the fly.

Time-sharing of functional applications processed in the same computing resources inside an ECU if both applications are mutually exclusive (for example, deployment of a lane-departure warning while a vehicle is running straight ahead and switching to a rear-camera view or park-assistance application when it's backing up) is an idea that could help to reduce the cost and complexity of such embedded systems, as well as freeing space and reducing weight in the vehicle.

This idea applies as well to the self-adaptivity of specific algorithms in

implementation," automotive manufacturers and suppliers work together to define an open and standardized system architecture aimed at being a breakthrough in automotive E/E design (<http://www.autosar.org>).

FUNCTIONAL SAFETY

Similarly, the IEC 61508 is a general standard of the International Electrotechnical Commission governing the functional safety of electrical, electronic and programmable electronic systems and components, accepted worldwide and in use since 2004, based on the various fields of application of safety-related systems.

Among other standards tailored for this purpose is one specifically aimed at functional safety in the automotive arena: the International Organization for Standardization's ISO 26262. Still under development but expected to be in place in 2012, this new standard is designed to support and facilitate all the development activities of safe products in the automotive industry. It encompasses safety activities during the concept phase, product development, production and operation.

In fact, functional safety—understood as the absence of unacceptable risk due to hazards caused by malfunctions in E/E systems—has become a key requirement in automotive design. This recently published standard is aligned to automotive-industry use cases and the definitions of acceptable risks, with the aim of preventing catastrophic failures. In this context, the term “risk” is defined as the combination of the probability of harm or damage occurring and its severity. During the engineering development phase, the standard requires that all potential hazards and risks be assessed in advance and that developers take suitable measures to minimize them. ISO 26262 includes guidance to avoid these risks by providing appropriate requirements and processes.

Following this approach, in a vehicle there are functions catalogued as safety relevant and non-safety relevant. Safety-relevant functions are all those in which a malfunc-

tion could put the driver at risk. For those functions that are considered safety relevant, the standard defines several possible levels of risk. That is, some functions are more critical than others from the point of view of ensuring a specific safety goal.

The standard defines a series of automotive-safety integrity levels (ASILs) based on the severity of possible accidents, the probability of exposure to certain driving situations and risk reduction due to external measures. It identifies four levels, from ASIL D through C, B and A, with D representing the most stringent and A the least stringent level. Each ASIL lists requirements or recommendations that carmakers and suppliers must satisfy to reduce an “intolerable risk” to a tolerable residual risk.

One example: If the steering column gets locked while the car is moving, the driver could have an accident, since he or she would not be able to turn the wheel. To reduce this risk to a tolerable level, the implementation of the steering-column control function should follow certain safety design criteria, according to the ISO 26262 standard and the ASIL level given to such a safety goal.

Depending on the level for each safety goal, both hardware and software developers have to consider specific safety measures in the implementation of those functions involved. In the case of an ASIL at the high (D or C) level, a common design method is to decompose safety requirements into redundant safety requirements to allow ASIL tailoring at a potentially lower ASIL level implemented by sufficiently independent elements. That is, the original safety requirements are redundantly implemented by different processors (typically MCUs), deploying redundant channels to minimize the probability of a catastrophic failure.

In the end, manufacturers and suppliers need to prove to the accreditation authorities that their E/E systems will deliver the required functionality safely and reliably according to the industry-specific regulations (<http://www.iso.org/iso>).

changing environmental or external conditions. For example, a given engine control algorithm can self-adapt, via partial reconfiguration, certain hardware blocks to perform optimally in any operating conditions of temperature or battery voltage. The same concept applies to communications systems, for instance

uration controller, while the PRR plays the role of a shared resource to swap different functional tasks or applications in and out at different times.

Finally, bringing the previous concept to its extreme, it is possible to envision a universal automotive ECU platform that can be configured in the manufacturing line and customized for a specific ECU

by electronics, and there is no end in sight. The car of the future will be based on very advanced software and hardware technologies enabling new features like autonomous driving, vehicle-to-vehicle communication, entertainment and improved safety. However, controlling the cost of automotive embedded systems is

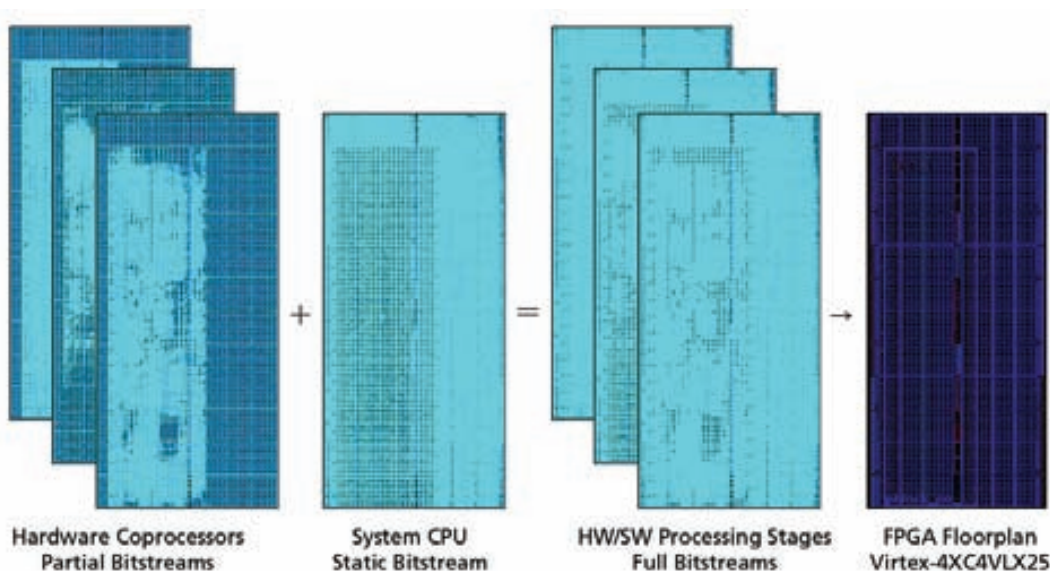


Figure 5 – Spatial and temporal partitioning of the automotive ECU application in a runtime reconfigurable FPGA composed of one PR region and one static region

designing a cryptographic controller able to specialize at runtime its security level in function of certain parameters. Another example would be making an ECC encoder/decoder IP that detects and corrects data transmission errors in noisy communication channels and is able to dynamically adapt its hardware architecture depending on the signal-to-noise ratio sensed.

Figure 5 shows an example of an ECU system deployed in a Xilinx Virtex[®]-4 FPGA composed of one static region and one PR region. The static region integrates a soft-core MicroBlaze processor and an ICAP-based reconfig-

urability inside the vehicle. This idea, technically feasible through reconfigurable hardware, would simplify the logistics in manufacturing plants and keep inventory to a minimum. That's because the module assembled in the production line should be, from the hardware point of view, the same for all the vehicles, all within a single platform design or product architecture (based on flexible hardware). Only the downloadable bitstream would make the ECU functional differentiation.

HIGHLY INTEGRATED ECUS

Around 90 percent of innovation in the automotive sector is nowadays driven

extremely important for automobile suppliers competing in a very high-volume industry. As a result, current trends are oriented toward reducing the number of ECUs present in the vehicle in exchange for delivering major functionality per unit. This is a goal that demands more-powerful computing platforms.

One approach shared by many industry players is to create highly integrated ECUs as domain controllers—that is, several core processors or microcontrollers in the same board sharing bus connections and other resources aimed at reducing system complexity from the whole-

vehicle perspective. This trend permits us to think about the possible use of reconfigurable hardware in the design of ECUs as a valid means of increasing computation parallelism and reducing PCB complexity while reaching a balanced solution in terms of performance/cost.

The approach presented in our work, although it is only an early concept, lays the groundwork for merging both AUTOSAR and ISO 26262 with runtime reconfigurable hardware to perform hardware/software co-design for a full automotive embedded ECU system. In fact, although today AUTOSAR does not cover reconfigurable hardware, we think this possibility cannot be dis-

carded in the future. Runtime reconfigurable SRAM-based FPGAs are already in use in aerospace applications despite their more difficult environmental conditions concerning susceptibility to SEU, and the automotive field has a history of adopting trends first seen in the aerospace field. Also, there exist already in the market some qualified design methodologies and tools aimed at enabling the implementation of FPGA-based safety-related systems, as well as the existence of one standard, already in place in the industry for some time and involving also FPGA devices, that regulates the design of components and systems in the avionic sector, such as the DO-254.

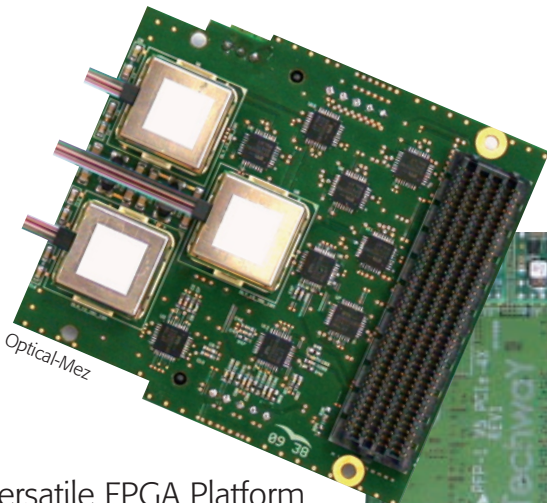
PARADIGM CHANGE DRIVEN BY CO-DESIGN

Therefore, this work spreads a computing paradigm change in the automotive field based on the replacement of a software-only solution by another alternative—one driven by hardware/software co-design and reconfigurable computing techniques in specific ECU scenarios where a purely software-based approach founded in Von Neumann MCUs is not feasible due to reasons of performance, complexity and safety. The continuous drop in the price of programmable logic technology—together with the incessant rise in performance such automotive electronic control units demand—should make this change viable in the not-too-distant future. 🌈

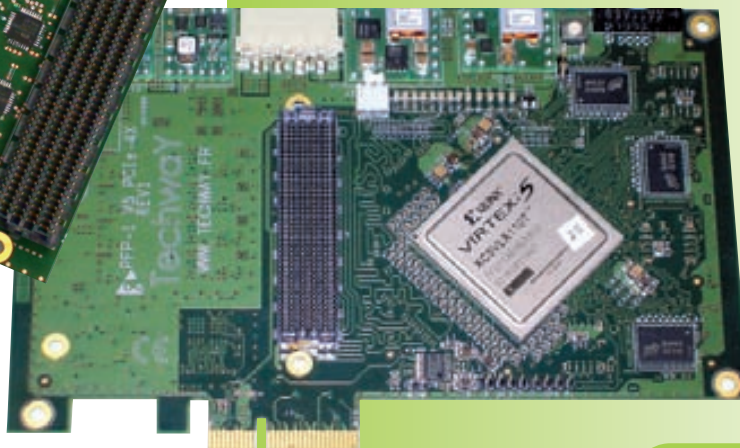
TechwayY

The way of innovation

Versatile FPGA Platform



- PCI Express 4x Short Card
- Xilinx Virtex Families
- I/O enabled through an FMC site (VITA 57)
- Development kit and drivers optimized for Windows and Linux



The Versatile FPGA Platform provides a cost-effective way of undertaking **intensive calculations** and **high speed communications** in an industrial environment.

www.techway.eu

Lowering the Barriers to a Successful FPGA-Based Prototype Project

The Xilinx Virtex-6 and Synopsys HAPS systems make prototyping practical for ASIC designs.

by **Troy Scott**

Senior Product Marketing Manager

Synopsys

troy.scott@synopsys.com

Recent surveys of the ASIC designer community show that more than 70 percent of all design projects use some form of FPGA-based prototype to provide a high-speed model of the design prior to the tapeout of a test chip. The prototype helps to start software development tasks and to close functional coverage earlier in the design cycle. In many organizations, the versatility and reuse benefits of a commercially available FPGA-based prototyping system have led to a much improved return on investment compared with custom circuit boards developed in-house.

The Synopsys HAPS High-Performance ASIC Prototyping System has a long pedigree of using Xilinx® Virtex® FPGAs as a logic host for ASIC prototyping. Starting in 1987 at HARDI Electronics AB of Lund, Sweden (now part of the Hardware Platform Group at Synopsys), the HAPS system of circuit boards has proven to be a versatile tool for thousands of prototyping projects. To understand why, it's instructive to take a deeper look at the functional elements of the Virtex-6 and the HAPS system architecture that have made this combination a state-of-the-art solution.

VIRTEX-6 IS IDEAL FOR ASIC PROTOTYPING

Throughout its history the HAPS prototyping systems have leveraged the Virtex family of devices, including the Virtex-II, Virtex-II Pro, Virtex-4, Virtex-5 and, most recently, Virtex-6 LX760 devices. The highest-capacity LX760 has proven its utility as a vehicle for ASIC prototyping for many companies. Designers can apply both the programmable and embedded-block functions of the device to the prototyping effort.

The primary LX760 logic building block, a slice, is composed of programmable lookup tables (LUTs) and storage elements used to implement combinatorial functions, along with small RAM blocks or shift registers. A total of 118,560 slices are available in the

LX760, providing a deep resource to host logic functions such as parity, XOR, AND, OR and synchronous logic of the original ASIC RTL. When considering the migration of arithmetic functions such as multipliers, accumulators and other DSP logic, the LX760 provides embedded signal-processing blocks that are far more area-efficient than slice-based logic cells. An LX760 provides 864 DSP blocks along

Designers can apply the programmable and embedded-block functions of the LX760 to the ASIC prototyping effort.

with the DSP48E1 slice, consisting of a 25 x 18-bit two's complement multiplier and 48-bit accumulator. The Synopsys Synplify Premier FPGA logic-synthesis tool automatically targets these essential building blocks and provides the best clock performance.

System-on-chip (SoC) designs include various memories provided by either the target ASIC fab's memory library or a memory compiler. In most cases FPGAs can represent these memories efficiently using slices for smaller register files, embedded block RAM for wider, deeper arrays or, for the largest memories, HAPS memory daughterboards as a host. Effective ASIC memory migration is part of the FPGA-based prototyping methodology, which often involves assigning an alternative FPGA-friendly implementation to ensure the most efficient use of on-chip RAM. Substitutions—driven, for example, by Verilog HDL 'define—help provide an easy way to switch between modules as the code is targeted to different implementations. In many cases ASIC memory compilers can generate an “emulation-friendly” light version of the model that excludes test logic so that the prototyping can more easily migrate to the prototype system. A Virtex-6 760LX provides up to 8,280 kbits of distributed slice-based RAM and 25,920 kbits of block RAM for hosting memory modules.

The Virtex-6 family provides up to nine embedded PLL-based mixed-mode clock managers (MMCMs) that can serve as frequency synthesizers for a wide range of frequencies, as a jitter filter for either external or internal clocks, and as deskew clocks. The Virtex-6 MCMM is capable of output frequencies from 4.69 MHz to 800 MHz. This is a key resource for the prototype

to achieve multimegahertz speeds of internal clocks. The Synopsys Certify multi-FPGA prototyping environment eases the implementation of ASIC clocks by targeting both onboard PLLs of the HAPS system and the embedded LX760 MMCMs with clock-distribution IP that's generated based on how a design is partitioned.

HAPS' MODULAR ARCHITECTURE APPLIED

The HAPS design team applied some key design criteria to balance the cost, performance and connectivity needs of a prototyping circuit board. For hardware-assisted verification tools such as an ASIC emulator, RTL debugging and verification was the primary application. But the prototype community was more concerned with “validation,” a process in which testing scenarios are vastly expanded. This required that the prototype performance be high enough to interface to real-world protocols like Gigabit Ethernet and double-data-rate (DDR) memory interfaces, and execute SoC processors to boot an operating system.

These implications convinced the hardware designers to provide a very open, modular architecture that maximized FPGA I/O access, along with a reconfigurable interconnect scheme

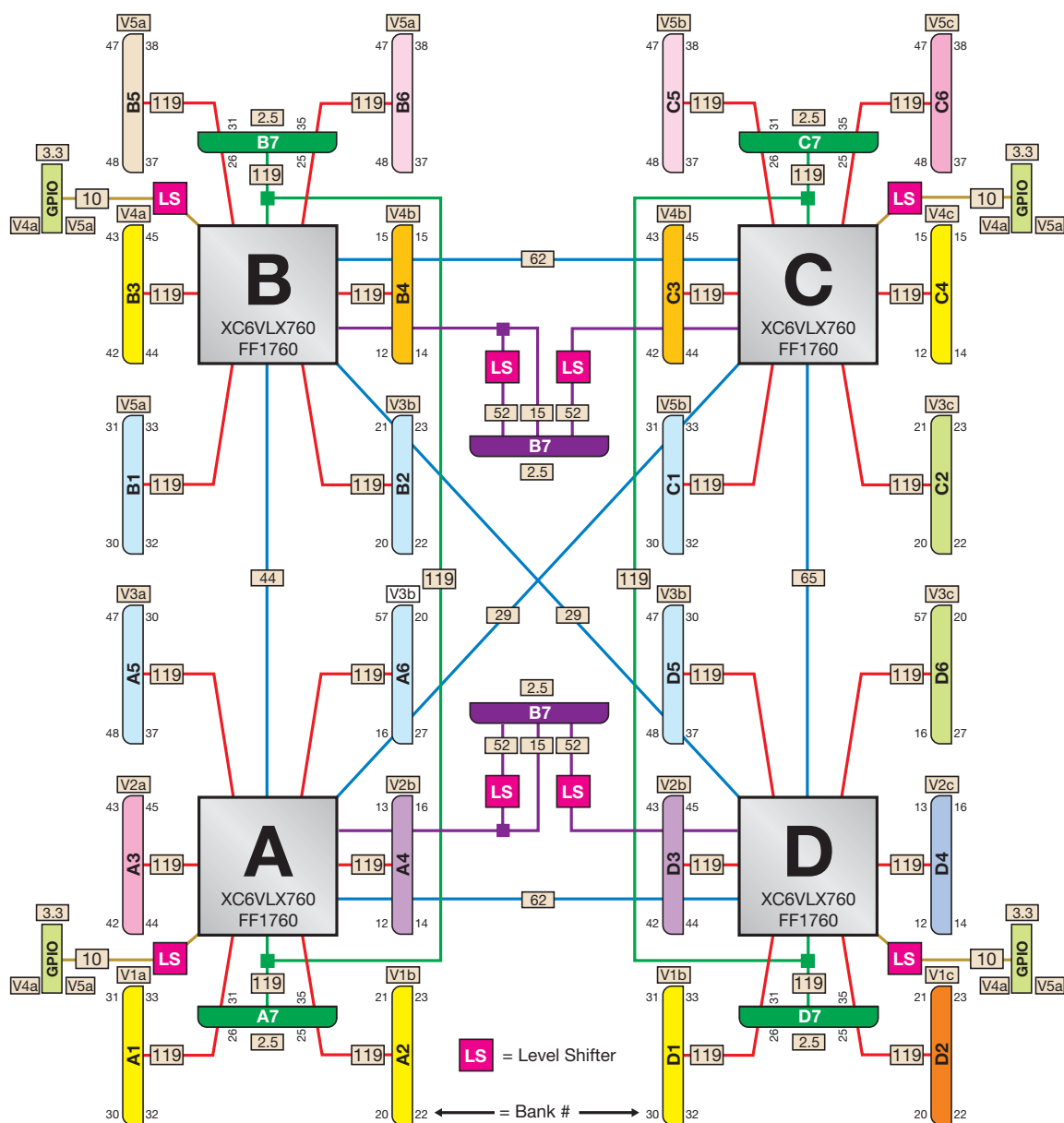


Figure 1 – I/O and inter-FPGA connections of the HAPS-64 system

between FPGAs. To maximize the I/O, control and power signal access of the Virtex I/O banks, we developed a special connector, the HapsTrak standard (see sidebar), specifically for the prototyping PCBs. Daughterboards host specialty physical interfaces for popular standards like PCIe®, DDR memory and USB. HAPS daughterboards use the same HapsTrak standard male/female connectors.

The combination of four XC6VLX760-FF1760 devices in the HAPS-64 system results in a “super-FPGA” platform with a capacity of

up to 18 million ASIC gates. With high logic capacity in place, the HAPS design team added HapsTrak connectors for I/Os and inter-FPGA buses into a regular matrix that would support stacking of systems or cabling. Each FPGA is connected to a group of these connectors and each connector has dedicated pins for power and clocks. Onboard “buried” routes provide additional FPGA connections. Interconnect boards create wider buses. Parts of the buses are linked to two pairs of HapsTrak connectors, which can be used to

expand the buses to other systems, daughterboards and test equipment. All HAPS systems have mating connectors on the bottom, so they can also be stacked together. This very open I/O scheme provides the ASIC prototyper with lots of expansion and connectivity options.

Clock management is a critical consideration for any circuit board design, and prototyping systems face similar issues. An onboard global clock distribution tree, which can be sourced with onboard PLLs or by an external clock generator, avoids

skew and provides adequate fanout buffer drive strength to ensure good integrity at high speed. The design provides a high-capacity clock routing pool of 299 differential clock inputs to connect among the FPGAs of the HAPS motherboard. Clock distribution circuits make it possible to chain together or stack as many as three HAPS-64 systems, expanding the system capacity up to 54 million ASIC gates.

Board reliability was an important issue for prototyping systems that may often be carried out of the controlled lab environment into the field and trade shows. HAPS systems are upgraded with a dedicated supervisor FPGA to control power and clock distribution, monitor voltage levels and manage data exchanges with a host computer via the Synopsys UMRBus interface.

BLENDING HARDWARE AND SOFTWARE VALIDATION TOOLS

What does the future hold for prototyping systems? With the emergence of co-emulation standards like SCE-MI and high-performance physical interfaces such as PCIe and Synopsys' Universal Multi-Resource Bus (UMRBus), it's now feasible to blend virtual prototypes written in C/C++/SystemC with the hardware prototype. This hybrid prototyping supports validation scenarios where

designers can join together IP in a much wider variety of model abstractions. The HAPS system architecture lends itself to being organized for best fit of the SoC project. On one case it may host a peripheral subsystem when it's critical to review real-world interfaces. In another, the FPGAs host a hardware execution engine to improve data throughput.

As commercial prototyping solutions like the HAPS systems have become mainstream validation methodologies, they must adhere to standards, making the tool easily deployable and allowing integration into existing flows and environments. In the case of prototyping, the system must also be reusable for many years and for a wide variety of projects and designs, so as to maximize the ROI of a commercial system. The flexibility of the HAPS architec-

ture derives from a modular concept that minimizes hardware when possible. These modular rules make the HAPS products easy to understand and easy to build, while delivering an enormous amount of freedom to prototype virtually any system. The robust and reliable hardware, together with the flexibility and I/O connectivity and the large collection of off-the-shelf hardware modules, makes it easy for ASIC designers to build their own customized ASIC prototype.

For further insight into design-for-prototyping, see the *FPGA-Based Prototyping Methodology Manual* (FPM) by Doug Amos, Austin Lesea and René Richter (<http://www.synopsys.com/fpm>). Learn more about Synopsys HAPS at www.synopsys.com/haps.

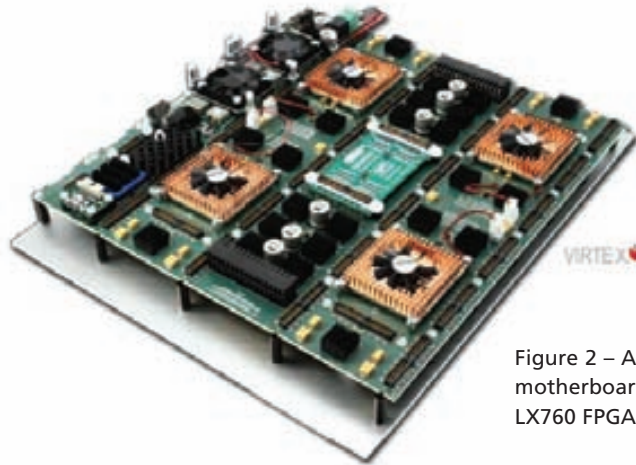


Figure 2 – A HAPS-64 motherboard with four Virtex-6 LX760 FPGAs onboard

GETTING CONNECTED WITH THE HAPSTRAK STANDARD

What gives the HAPS system its inherent flexibility and ability to generate a high-performance multimegahertz prototype is its modular concept and a high-density connector scheme called the HapsTrak standard.

Synopsys developed the connector jointly with Samtec, Inc. to provide density, signal integrity and reliability for PCBs that will be assembled and reassembled many times over their lifetime. We have upgraded HapsTrak over the years to address the scenarios in which prototype boards are applied and to keep pace with the innovations in FPGA programmable I/O performance and specialty circuits for double-data-rate (DDR) support. Electromechanical quality and reliability were top design criteria for the HAPS design team. The connector has impedance-matched traces to minimize signal reflections and provides a remote identification capability so that software applications can label and recognize connectors.

A single HAPS system consists of one, two or four FPGAs. In the HAPS-60 series, a Virtex-6 LX760 in an FF1760 package is the device of choice. Positioned near each FPGA are six HapsTrak II 120-pin connectors. Programmable I/O, clock and power signals are routed from each FPGA to the six connectors.

– Troy Scott

Ins and Outs of Digital Filter Design and Implementation

FPGAs make it easy to implement many types of digital signal-processing filters effectively.

by **Adam P. Taylor**

Principal Engineer
EADS Astrium
aptaylor@theiet.org

Filters are a key part of any signal-processing system, and as modern applications have grown more complex, so has filter design. FPGAs provide the ability to design and implement filters with performance characteristics that would be very difficult to re-create with analog methods. What's more, these digital filters are immune to certain issues that plague analog implementations, notably component drift and tolerances (over temperature, aging and radiation, for high-reliability applications). These analog effects significantly degrade the filter performance, especially in areas such as passband ripple.

Of course, digital models have their own quirks. The rounding schemes used within the mathematics of the filters can be a problem, as these rounding errors will accumulate, impacting performance by, for example, raising the noise floor of the filter. The engineer can fall back on a number of approaches to minimize this impact, and schemes such as convergent rounding will provide better performance than traditional rounding. In the end, rounding-error problems are far less severe than those of the analog component contribution.

One of the major benefits of using an FPGA as the building block for a filter is the ability to easily modify or update the filter coefficients late in the design cycle with minimal impact, should the need for performance changes arise due to integration issues or requirements changes.

FILTER TYPES AND TOPOLOGIES

Most engineers familiar with digital signal processing will be aware that there are four main types of filters. Low-pass filters only allow signals below a predetermined cutoff frequency to be output. High-pass filters are the inverse of the low pass, and will only allow through frequencies above the cutoff frequency. Bandpass filters allow a predetermined bandwidth of frequencies, preventing other frequencies from entering. Finally, band-reject filters are the inverse of the bandpass variety, and therefore reject a predetermined bandwidth while allowing all others to pass through.

Most digital filters are implemented by one of two methods: finite impulse response (FIR) and infinite impulse response (IIR). Let's take a closer look at how to design and implement FIR filters, which are also often called windowed-sinc filters.

So why are we focusing upon FIR filters? The main difference between the two filter styles is the presence or lack of feedback. The absence of

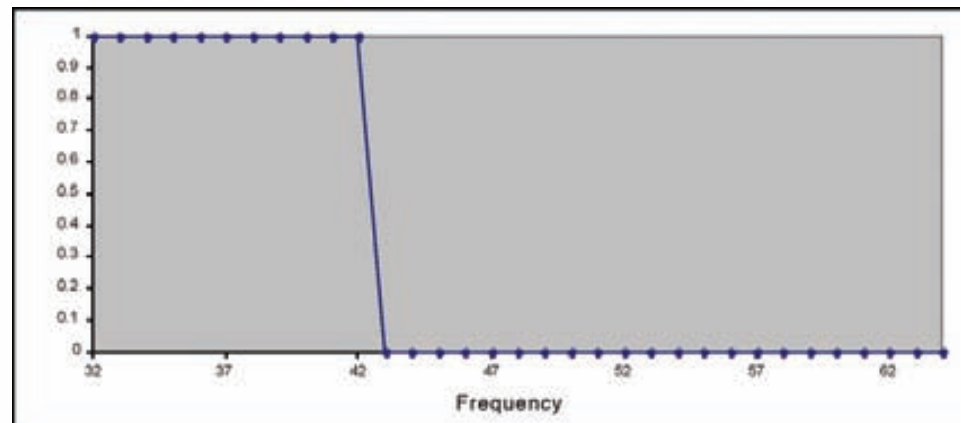


Figure 1 – Ideal low-pass filter performance, with abrupt transition from passband to stopband

feedback within the FIR filter means that for a given input response, the output of the filter will eventually settle to zero. For an IIR filter subject to the same input which does contain feedback, the output will not settle back to zero.

The lack of feedback within the filter implementation makes the FIR filter inherently stable, as all of the filter's poles are located at the origin. The IIR filter is less forgiving. Its stability must be carefully considered as you are designing it, making the windowed-sinc filter easier for engineers new to DSP to understand and implement.

If you were to ask an engineer to draw a diagram of the perfect low-pass filter in the frequency domain, most would produce a sketch similar to that shown in Figure 1.

The frequency response shown in Figure 1 is often called the “brick-wall” filter. That's because the transition from passband to stopband is very abrupt and much sharper than can realistically be achieved. The frequency response also exhibits other “perfect” features, such as no passband ripple and perfect attenuation within the stopband.

If you were to extend this diagram such that it was symmetrical around 0 Hz extending out to both +/- FS Hz (where FS is the sampling frequency),

and perform an inverse discrete Fourier transform (IDFT) upon the response, you would obtain the filter's impulse response, as shown in Figure 2.

This is the time-domain representation of the frequency response of the perfect filter shown in Figure 1, often called the filter kernel. It is from this response that FIR or windowed-sinc filters get their name, as the impulse response is what is achieved if you plot the sinc function

$$\frac{h[i] = \sin(2 * \text{PI} * \text{Fc} * i)}{i * \text{PI}}$$

Combined with the step response of the filter, these three responses—frequency, impulse and step—provide all the information on the filter performance you need to know to demonstrate that the filter under design will meet the requirements placed upon it.

FREQUENCY RESPONSE

The frequency response is the traditional image engineers consider when thinking about a filter. It demonstrates how the filter modifies the frequency-domain information.

The frequency response allows you to observe the cutoff frequency, stopband attenuation and passband ripple. The roll-off between passband and stopband—often called the transition

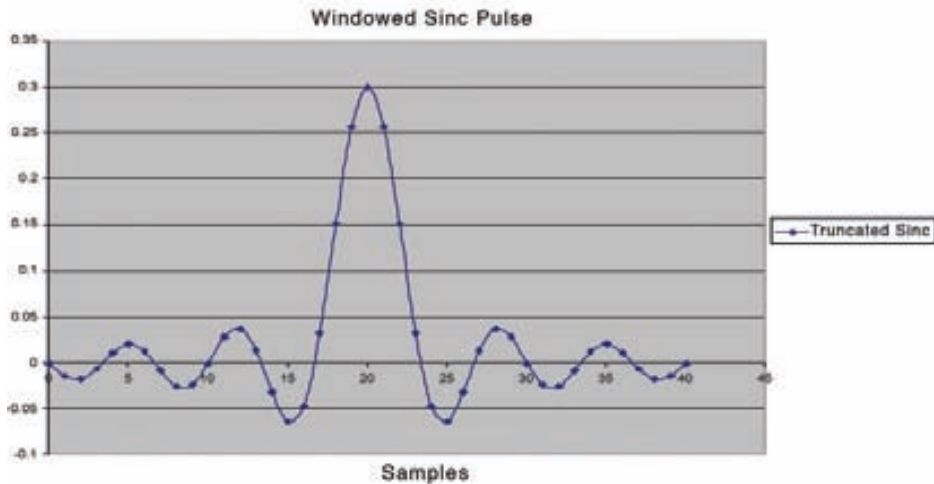


Figure 2 – IDFT or impulse response of the perfect low-pass filter

band—is also apparent in this response. Ripples in the passband will affect the signals being filtered. The stopband attenuation demonstrates how much of the unwanted frequencies remain within the filter output.

This can be critical in applications where specific frequency rejection is required, for example when filtering one frequency-division multiplexed channel from another in a communications system.

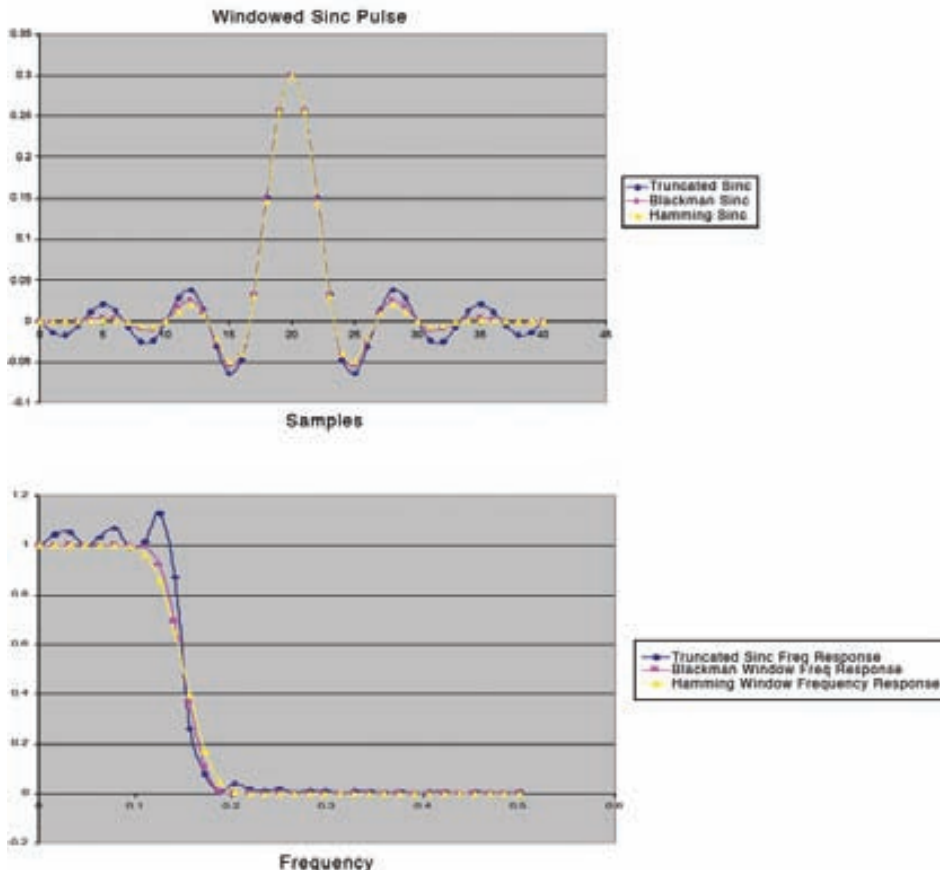


Figure 3 – Low-pass filter impulse responses (top chart) and frequency responses

IMPULSE RESPONSE

It is from the impulse response that the coefficients for your filter are abstracted. However, to achieve the best performance from your filter, the standard practice is to use a windowing function. Windowing is the technique of applying an additional mathematical function to a truncated impulse response to reduce the undesired effects of truncation.

Figure 2 demonstrates the impulse response extending out infinitely with ripples which, though they diminish significantly in amplitude, never settle at zero. Therefore, you must truncate the impulse response to $N + 1$ coefficients chosen symmetrically around the center main lobe, where N is the desired filter length (please remember that N must be an even number). This truncation affects the filter's performance in the frequency domain due to the abrupt cutoff of the new, truncated impulse response. If you were to take a discrete Fourier transform (DFT) of this truncated impulse response, you would notice ripples in both the passband and stopband along with reduced roll-off performance. This is why it is common practice to apply a windowing function to improve the performance.

STEP RESPONSE

The step response, which is obtained by integrating the impulse response, demonstrates the time-domain performance of the filter and how the filter itself modifies this performance. The three parameters of importance when you are observing the step response are the rise time, overshoot and linearity.

The rise time is the number of samples it takes to rise between 10 percent and 90 percent of the amplitude levels, demonstrating the speed of the filter. To be of use within your final system, the filter must be able to distinguish between events in the input signal; therefore, the step response must be shorter than the spacing of events in the signal.

Overshoot is the distortion that the filter adds to the signal as it is processing it. Reducing the overshoot in the

step response allows you to determine if the signal distortion results from either the system or the information that it is measuring. Overshoot reduces the uncertainty of the source of the distortion, degrading the final system performance, and possibly means the system does not meet the desired performance requirements.

If the signal's upper and lower halves are symmetrical, the phase response of the filter will have a linear phase, which is needed to ensure the rising edge and falling edge of the step response are the same.

It is worth noting at this point that it is very difficult to optimize a filter for good performance in both the time and frequency domains. Therefore, you must understand which of those domains contains the information you are required to process. For FIR filters, the information required is within the frequency domain. Therefore, the frequency response tends to dominate.

WINDOWING THE FILTER

Using a truncated impulse response will not provide the best-performing digital filter, as it will demonstrate none of the ideal characteristics. For this reason, designers use windowing functions to improve the pass-band ripple, roll-off and stopband attenuation performance of the filter. There are many window functions that you can apply to the truncated sinc, for example Gaussian, Bartlett, Hamming, Blackman and Kaiser—the list goes on. However, two of the most popular window functions are the Hamming and Blackman. Let's explore them in more detail.

Both of these windows, when applied, reduce the passband ripple and increase the roll-off and attenuation of the filter. Figure 3 shows the impulse and frequency responses for a truncated sinc, and with Blackman and Hamming windows applied. As you can see, both windows significantly improve the passband ripple. The roll-off of the filter is deter-

mined not only by the window but by the length of the filter—that is, the number of coefficients, often called filter taps.

Hamming window:

$$w[i] = 0.54 - 0.46 \cos(2\pi i/N)$$

Blackman window:

$$w[i] = 0.42 - 0.52 \cos(2\pi i/N) + 0.08 \cos(4\pi i/N)$$

Where i runs from 0 to N , providing a total of $N + 1$ points.

To apply these windows to the truncated impulse response, you must multiply the window coefficients with the truncated impulse response coefficients to produce the desired filter coefficients.

While the window type determines the roll-off frequency, a rule of thumb is that for a desired transition bandwidth the number of taps required is indicated by

$$N = 4 / BW$$

where BW is the transition bandwidth.

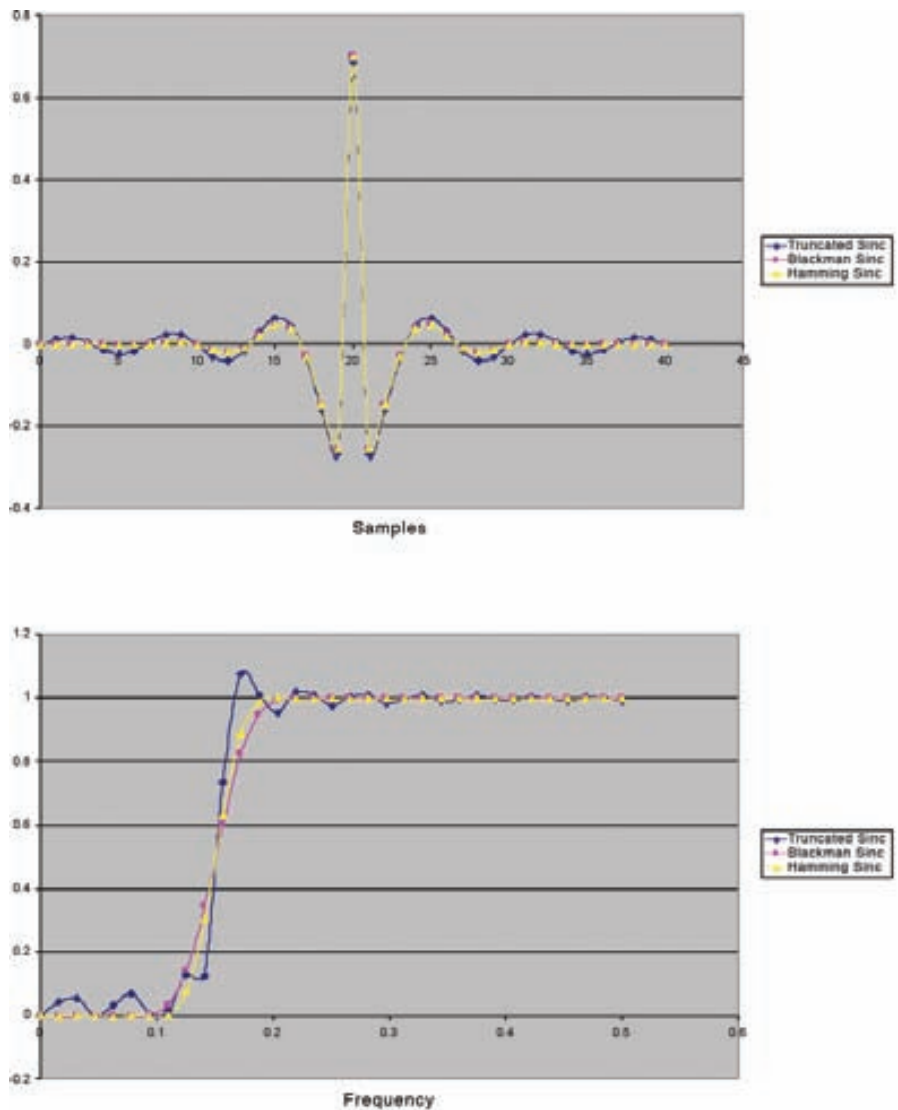


Figure 4 – High-pass filter impulse response (top) and frequency response using spectral inversion, with nonwindowed, and Hamming and Blackman windows applied

IMPLEMENTING DIFFERENT FILTER TOPOLOGIES

No matter what the end type of filter—bandpass, band reject or high pass—all of these start out with the initial design of a low-pass filter. If you know how to create a low-pass filter and a high-pass filter, you can then use combinations of the two to produce band-reject and bandpass filters.

Let's first examine how to convert a low-pass filter into a high-pass one. The simplest method, called spectral inversion, changes the stopband into the passband and vice versa. You perform spectral inversion by inverting

each of the samples while at the same time adding one to the center sample. The second method of converting to a high-pass filter is spectral reversal, which mirrors the frequency response; to perform this operation, simply invert every other coefficient.

Having designed the low- and high-pass filters, it's easy to make combinations that you can use to generate bandpass and band-reject filters. To generate a band-reject filter, just place a high-pass and a low-pass filter in parallel with each other and sum together the outputs. You can assemble a band-pass filter, meanwhile, by placing a

low-pass and a high-pass filter in series with each other.

AND NOW FOR THE DESIGN

The information we've reviewed so far details what windowed-sinc filters are, the importance of windowing and how to generate filters of different topologies. However, before you can implement your filter within an FPGA, you must generate a set of filter coefficients using one of several software tools, such as Octave, MATLAB® or even Excel. Many of these tools provide simplified interfaces and options, allowing you to design the filter with

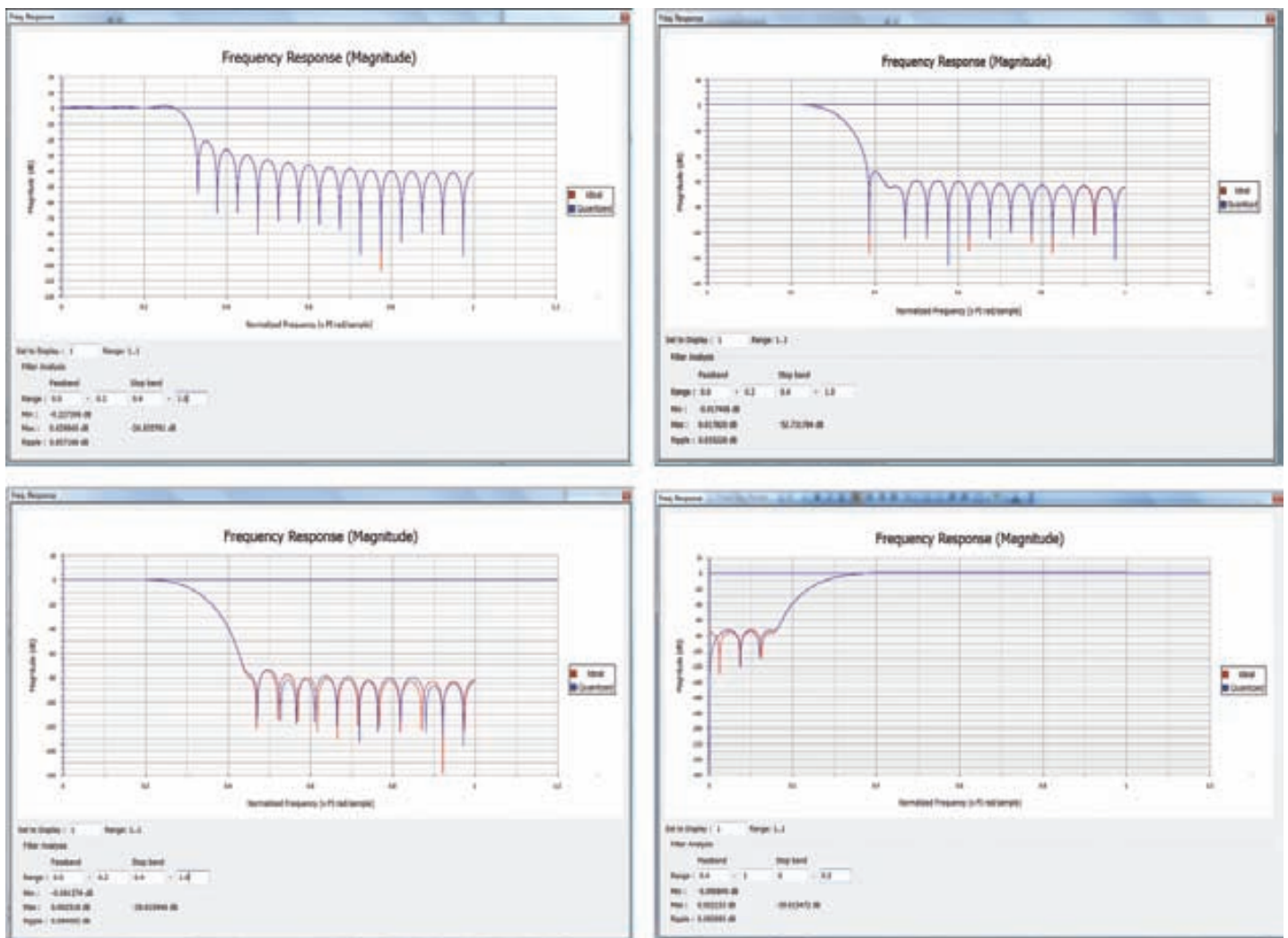


Figure 5 – Xilinx CORE Generator frequency responses: Top, truncated sinc (left) and Blackman window. Bottom, Hamming window (left) and Hamming-windowed high-pass filter

Digital filters find their way into many applications these days, and FPGAs offer a significant advantage to the system designer who needs to use them.

minimal effort, the FDA tool in MATLAB being a prime example.

Having generated a set of coefficients for the desired filter, you are ready to implement your filter within the FPGA. No matter the number of taps you have decided on, the basic structure of each stage of the FIR filter will be the same and will consist of a multiplier, storage and adder.

The method most engineers prefer, and the one that's by far the simplest, is to use the Xilinx® CORE Generator™ tool's FIR Compiler, which provides multiple options for customizing and for generating advanced filters. You can import the generated coefficients into the FIR Compiler as a COE file; this file will contain the coefficients for the filter with the radix also being declared.

```
Radix=10;
Coefdata =
-0.013987944,
-0.01735736,
-0.005971498,
0.012068368,
0.02190073,
```

Once you have loaded these coefficients, FIR Compiler will display the frequency response of the filter for the coefficients provided, along with basic performance characteristics such as stop-band attenuation and passband ripple.

Once you have completed your customization of the filter using the FIR Compiler tool, CORE Generator will produce all the necessary files to both implement the design and to simulate it during behavioral simulation prior to implementation, provided the user has access to the correct simulation libraries.

If you prefer, you can also implement the filter in HDL you have generated yourself. This will often be the choice if your final implementation will be in an ASIC and you are using the FPGA implementation as a prototyping system. In this case, the first step is to quantize the filter coefficients to use a fixed-number representation of the floating-point results. As the filter coefficients will represent positive and negative numbers, it is common practice to represent these in a two's complement format. Once these coefficients have been quantized, you can use them as constants within the HDL design.

Digital filters find their way into many applications these days, and FPGAs offer a significant advantage to the system designer who needs to use them. Windowed-sinc filters are very simple to design and implement using basic math tools along with either FPGA core-generation tools or directly in HDL. ●●

FPGA Modules

embedded world 2012
Exhibition Conference
Booth 104 - Hall 4A



New!

Mars AX3 XILINX Artix-7 FPGA • DDR3 SDRAM
Gigabit Ethernet • 108 I/Os

Designed for Linux



New!

Mars ZX2 XILINX Zynq EPP • DDR3 SDRAM
USB2 • Gigabit Ethernet • 108 I/Os



MA-MX1-KIT
\$ 549

Mars MX1 Kit

- Mars Starter Board
- Mars MX1 FPGA Module
- Reference Design

Prices in USD plus VAT+SH. Subject to change.

Universal Drive IP Core



Now in Beta Phase!

www.enclustra.com

ENCLUSTRA
FPGA SOLUTIONS


FPGA Design Center

Using FPGAs to Solve Tough DSP Design Challenges

by **Reg Zatrepalek**

DSP/FPGA Design Specialist
Hardent Inc.

rzatrepalek@hardent.com



A short, practical review of DSP vs. FPGA technology and a specific comparison of both architectures in a FIR filter application.

DSPs are invaluable in electronic system design for their ability to quickly measure, filter or compress analog signals on the fly. In doing so, they help the digital world communicate with the real world, the analog world. But as electronic systems become more elaborate, incorporating multiple analog sources to process, engineers are forced to make tough decisions. Is it better to use multiple DSPs and synchronize that functionality with the rest of the system? Or is it better to have one high-performance DSP handling multiple functions with elaborate software?

In many cases, today's systems are so complex that single-DSP implementations have insufficient processing power. At the same time, system architects simply can't afford the costs, complexities and power requirements of multiple-chip systems.

FPGAs have now emerged as a great choice for systems requiring high-performance DSP functionality. In fact, FPGA technology can often provide a much simpler solution to difficult DSP challenges than a standalone digital signal processor. To understand why requires a look back at the origins and evolution of the DSP.

MICROPROCESSORS FOR A SPECIALIZED PURPOSE

Over the past two decades, traditional DSP architectures have struggled to keep pace with increasing performance demands. As video systems make greater strides toward high definition and 3D, and communications systems push the limits of current technology to achieve higher bandwidth, designers need alternative implementation strategies. Hardware used to implement digital signal-processing algorithms may be categorized into one of three basic types of device: microprocessors, logic and memory. Some designs may require additional hardware for analog-to-digital (A/D) and digital-to-analog (D/A) conversion, and for high-speed digital interfaces.

Traditional digital signal processors are microprocessors designed to perform a specialized purpose. They are well-suited to algorithmic-intensive tasks but are limited in performance by clock rate and the sequential nature of their internal design. This limits the maximum number of operations per second that they can carry out on the incoming data samples. Typically, three or four clock cycles are required per arithmetic logic unit (ALU) operation. Multicore architectures may increase performance, but these are still limited. Designing with traditional signal processors therefore necessitates the reuse of architectural elements for algorithm implementation. Every operation must cycle through the ALU, either fed back internally or externally, for each addition, multiplication, subtraction or any other fundamental operation performed.

Unfortunately, in dealing with many of today's high-performance applications, the classical DSP fails to satisfy sys-

tem requirements. Several solutions have been proposed in the past, including using multiple ALUs within a device or multiple DSP devices on a board; however, such schemes often increase costs significantly and simply shift the problem to another arena. For example, to increase performance using multiple devices follows an exponential curve. In order to double the performance, two devices are required. To double the performance again takes four devices, and so on. In addition, the focus of programmers often shifts from signal-processing functions to task scheduling across the multiple processors and cores. This results in much additional code that functions as system overhead rather than attacking the digital signal-processing problem at hand.

A solution to the increasing complexity of DSP implementations came with the introduction of FPGA technology. The FPGA was initially developed as a means to consolidate and concentrate discrete memory and logic to enable higher integration, higher performance and increased flexibility. FPGA technology has become a significant part of virtually every high-performance system in use today. In contrast to the traditional DSP, FPGAs are massively parallel structures containing a uniform array of configurable logic blocks (CLBs), memory, DSP slices and some other elements. They may be programmed using high-level description languages like VHDL and Verilog, or at a block diagram level using System Generator. Many dedicated functions and IP cores are available for direct implementation in a highly optimized form within the FPGA.

The main advantage to digital signal processing within an FPGA is the ability to tailor the implementation to match system requirements. This means in a multiple-channel or high-speed system, you can take advantage of the parallelism within the device to maximize performance, while in a lower-rate system the implementation may have a more serial nature. Thus the designer can tailor the implementation to suit the algorithm and system requirements rather than compromising the desired ideal design to conform to the limitations of a purely sequential device. Very high-speed I/O further reduces cost and bottlenecks by maximizing data flow from capture right through the processing chain to final output.

As an example of how the FPGA stacks up, let's consider a

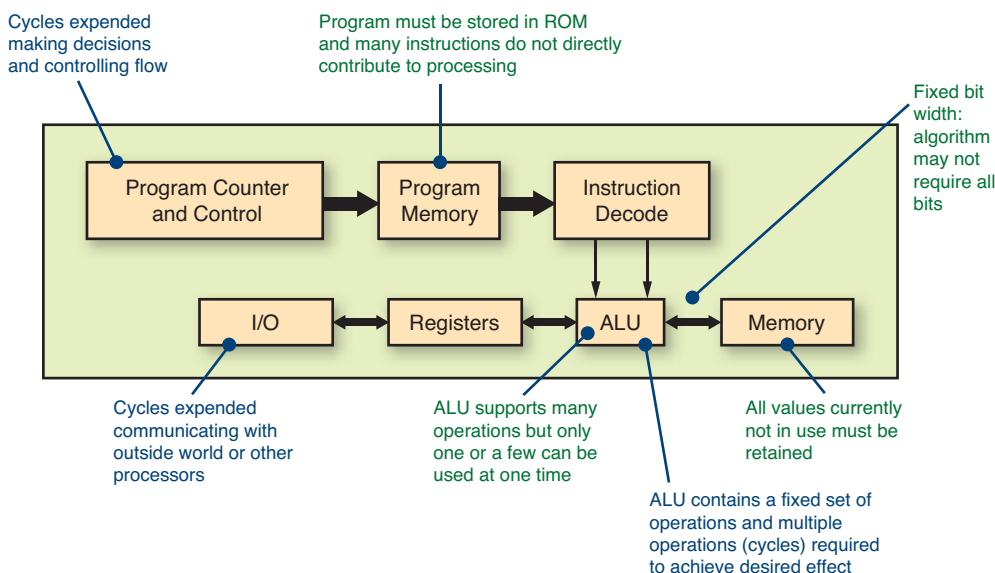


Figure 1 – Traditional DSP architecture

FIR filter implementation using both classical DSP and FPGA architectures to illustrate some of the strengths and weaknesses of each solution.

EXAMPLE: A DIGITAL FIR FILTER

One of the most widely used digital signal-processing elements is the finite impulse response, or FIR, filter. Designers use filters to alter the magnitude or frequency content of a data signal, usually to isolate or accentuate a particular region of interest within the sample data spectrum. In this regard, you can think of filters as a method of preconditioning a signal. In a typical filter application, incoming data samples combine with filter coefficients through carefully synchronized mathematical operations, which are dependent on the filter type and implementation strategy, and then move on to the next processing stage. If the data source and destination are analog signals, then the samples must first pass through an A/D converter, and the results fed through a D/A converter.

The simplest form of a FIR filter is implemented through a series of delay elements, multipliers and an adder tree or chain.

Mathematically, this equation describes the single-channel FIR filter:

$$Y_n = \sum_{i=0}^{N-1} k_{n-1} S_i$$

You can think of the terms in the equation as input samples, output samples and coefficients. If S is a continuous stream of input samples and Y is the resulting filtered stream of output samples, then n and k correspond to a particular instant in time. Thus, to compute the output sample $Y(n)$ at time n , a group of samples at N different points in time, or $s(n)$, $s(n-1)$, $s(n-2)$, ... $s(n-N+1)$, is required. The group of N input samples is multiplied by N coefficients and summed together to form the final result, Y .

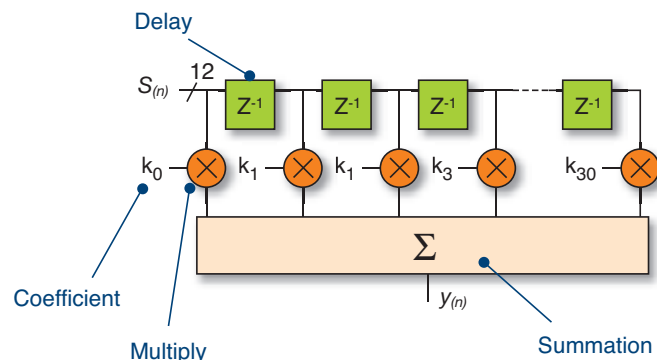


Figure 2 – FIR filter of length 31

Figure 2 is a block diagram for a simple 31-tap FIR filter (length $N = 31$).

Various design tools are available to help select the ideal length of a filter and the coefficient values. The goal is to select the appropriate parameters to achieve the required filter performance. The most popular design tool for choosing these parameters is MATLAB®. Once you have selected the filter parameters, the implementation follows the mathematical equation.

The basic steps for implementation of an FIR filter are:

1. Sample the incoming data stream.
2. Organize the input samples in a buffer so that each captured sample may be multiplied by each filter coefficient.
3. Multiply each data sample by each coefficient and accumulate the result.
4. Output filtered result.

A typical C program for implementing this FIR filter on a processor using a multiply-accumulate approach is shown in the code below.

```
/*
 * Capture the incoming data samples
 */
datasample = input();
/*
 * Push the new data sample onto the buffer
 */
S[n] = datasample;
/*
 * Multiply each data sample by each coefficient and accumulate the result
 */
y = 0;
for (i = 0; i < N; i++)
{
    y += k[i] * S[(n + i) % N];
}
n = (n+1) % N;
/*
 * Output filtered result
 */
output(y);
```

The implementation illustrated in Figure 3 is known as a multiply-and-accumulate or MAC-type implementation. This is almost certainly the way a filter would be implemented in a classical DSP processor. The maximum performance of a 31-tap FIR filter implemented in this fashion in a typical DSP processor with a core clock rate of 1.2 GHz is about 9.68 MHz, or a maximum incoming data rate of 9.68 Megasamples per second.

An FPGA, on the other hand, offers many different implementation and optimization options. If a very resource-effi-

cient implementation is desired, the MAC engine technique may prove ideal. Using a 31-tap filter as an example illustrates the impact of filter specifications on required logic resources. A block diagram of the implementation is shown in Figure 4.

Memory is required for data and coefficient storage. This may be a mixture of RAM and ROM internal to the FPGA. RAM is used for the data samples and is implemented using a cyclic RAM buffer. The number of words is equal to the

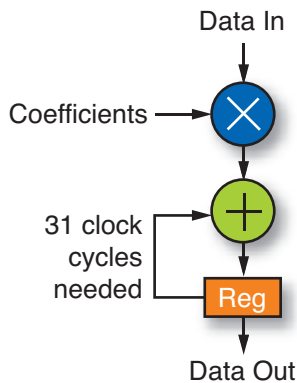


Figure 3 – MAC implementation in a classical DSP

number of filter taps and the bit width is set by sample size. ROM is required for the coefficients. In the worst case, the number of words will be the same as the number of filter taps, but if symmetry exists, this may be reduced. The bit width must be large enough to support the largest coefficient. A full multiplier is required since both the data sample and coefficient data change on every cycle. The accumulator adds the results as they are produced. The capture register is needed because the accumulator output changes on

every clock cycle as the filter is sampling data. Once a full set of N samples has been accumulated, the output register captures the final result.

When used in MAC mode, the DSP48 is a perfect fit. The input registers, output registers and adder unit are present in the DSP48 slice. The resources required for this 31-tap MAC engine implementation are one DSP48, one 18-kbit block RAM and nine logic slices. There are a few additional slices required for sample and coefficient address generation and control. If a 600-MHz clock were available in the FPGA, this filter could run at an input sample rate of 19.35 MHz, or 19.35 Msamples/s in a -3 speed grade Xilinx® 7 series device.

If the system specification required a higher-performance FIR filter, a parallel structure could be implemented. Figure 5 shows a block diagram of a Direct Form Type I implementation.

The Direct Form I filter structure provides the highest-performance implementation within an FPGA. This structure, which is also commonly referred to as a systolic FIR filter, uses pipelining and adder chains to exploit maximum performance from the DSP48 slice. The input is fed into a cascade of registers that acts as the data sample buffer. Each register delivers a sample to a DSP48 which is then multiplied by the respective coefficient. The adder chain stores the partial products that are then successively combined to form the final result.

No external logic is required to support the filter and the structure is extendable to support any number of coefficients. This is the structure that can achieve maximum performance, because there is no high-fanout input signal. The resources required to implement a 31-tap FIR filter are only 31 DSP48 slices. If a 600-MHz clock were available in the FPGA, this fil-

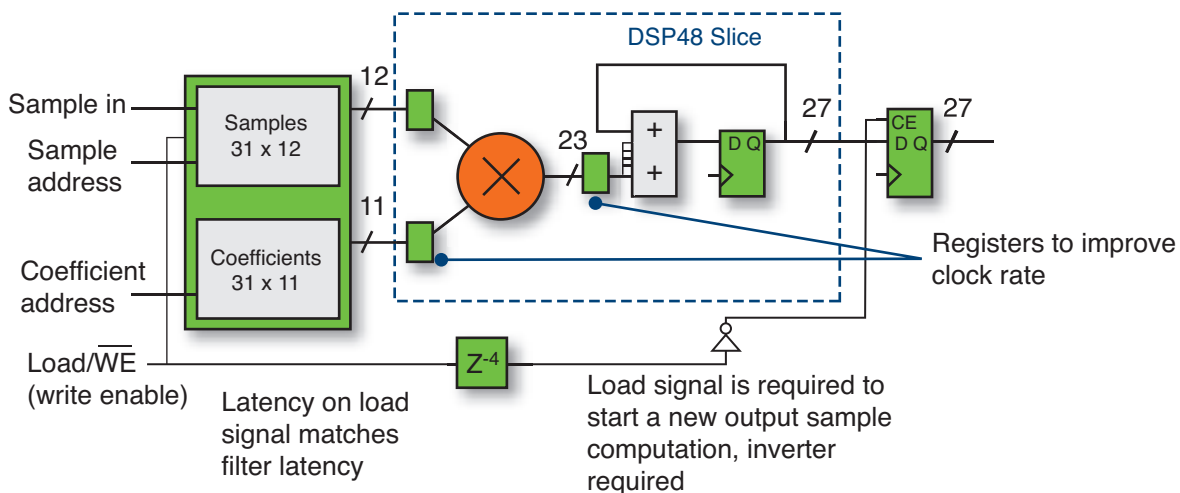


Figure 4 – MAC engine FIR filter in an FPGA

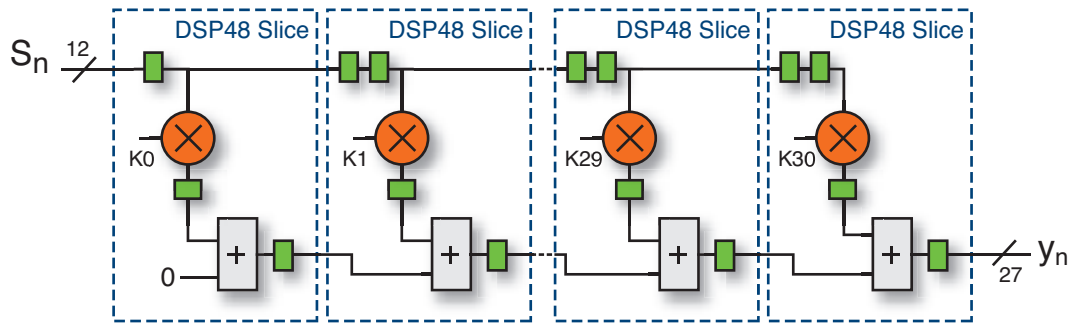


Figure 5 – Direct Form I FIR filter in an FPGA

ter could perform at an input sample rate of 600 MHz, or 600 Msamples/s, in a -3 speed grade 7 series device.

From this example, you can clearly see that the FPGA not only significantly outperforms a classic digital signal processor, but it does so with much lower clock rates (and therefore lower power consumption).

This example illustrates only a couple of implementation techniques for FIR filters in FPGA. The device may be further tailored to take advantage of data sample rate specifications that may fall in between the extremes of sequential MAC operation and full parallel operation. You may also consider additional trade-offs between performance and resource utilization involving symmetric coefficients, interpolation, decimation, multiple channels or multirate. The Xilinx CORE Generator™ or System Generator utilities will help you exploit all of these design variables and techniques.

DECIDING BETWEEN TRADITIONAL DSP AND FPGA

Conventional digital signal processors have been around for many years, and there are certainly instances where they present the best solution to a particular problem. If the system sample rate is below a few kilohertz and is a single-channel implementation, the DSP may be the obvious choice. However, as sample rates increase beyond a couple of megahertz, or if the system requires more than a single channel, FPGAs become more and more attractive. At high data rates the DSP may struggle to capture, process and output the data without any loss. This is due to the many shared resources, buses and even the core within the processor. The FPGA, however, can dedicate resources to each of these functions.

DSPs are instruction based, not clock based. Typically, three to four instructions are required for any mathematical operation on a single sample. The data must first be captured at the input, then forwarded to the processing core, cycled through that core for each operation and then released through the output. In contrast, the FPGA is clock based, so every clock cycle has the potential ability to perform a mathematical operation on the incoming data stream.

Since the DSP operates on instructions or code, the programming mechanism is standard C or, for higher performance, low-level assembly. This code may have high-level decision trees or branching operations, which may prove difficult to implement in an FPGA. A wide variety of legacy code exists to perform predefined functions or standards like audio and telephony codecs, for example.

FPGA vendors and third-party partners have realized the advantages of using FPGAs for high-performance DSP systems, and today many IP cores are available across most vertical markets including video, image-processing, communications, automotive, medical and military applications. Often it is simpler to break a high-level system block diagram into FPGA modules and IP cores than it is to map it into C code for DSP implementation.

MOVING FROM DSP TO FPGA

Examining a few key criteria may facilitate the decision between conventional DSP and FPGA (see Table 1).

It is widely accepted that software programmers outnumber hardware designers by a significant margin. The same is true for DSP programmers vs. FPGA designers. However, the transition for system architects or DSP designers to FPGA is not as difficult as software-to-hardware migration. Many resources are available to dramatically decrease the learning curve for DSP algorithm development and implementation in FPGAs.

The main hurdle is a paradigm shift from a sample- and event-based approach toward a clock-based problem description and solution. This transition is much easier to comprehend and apply if it is made at the system architecture and definition stage of the design process. It is not unusual for different engineers and mathematicians to be defining system architecture, DSP algorithms and FPGA implementation somewhat isolated from one another. This process is, of course, much smoother if each member has some knowledge of the challenges the other team members face.

In order to appreciate FPGA implementations, an architect need not be highly proficient at FPGA design. A fundamental understanding of the devices, resources and tools is

	FPGA	DSP
System sample rate	If the sample rate is more than a few MHz, or if multiple channels are required, FPGAs are the best choice.	For single-channel applications at sample rates of a few kHz, standalone DSPs will provide a more cost-effective solution.
System specification	A specification developed from or including a block diagram may map more easily into modules in the FPGA.	If the system spec is based on a C model, DSP may be easier to code into similar functional modules.
Data rate	If the I/O data rates are greater than a few Mbytes/second, the FPGA will be easier and more efficient to implement.	For low data rates extra cycles are usually available, so DSPs will not be bandwidth limited.
Data width or precision	Because of its highly uniform parallel structure, the FPGA is the clear choice for high-data-width or high-precision systems.	There may be no significant limitations if the data width is identical to the processor bus width and no potential for bit growth exists within the algorithm.
Decision trees or conditional branches	State machines may provide an easy alternative to conditional branches. If the required structure is a decision tree, embedded processor cores or MicroBlaze™ may provide the optimum solution.	DSPs are usually easier to implement with branches.
Floating point	FPGAs can have floating-point cores. Often high-precision fixed-point implementations can supplant floating-point requirements, and may in fact provide a higher-performance solution.	Conventional DSPs may have floating-point cores.
Legacy code	Mechanisms to adapt C code for use in FPGAs are now available.	If much C code already exists, a traditional DSP may be quicker to code but may struggle to increase performance from the legacy system specification.
Cores, libraries and third-party IP	Extensive parameterizable IP cores and libraries optimized for FPGAs are available from Xilinx and third-party vendors. Licensing agreements are usually simpler and easier to negotiate for FPGA implementations.	Many cores and libraries exist for conventional DSPs. May involve per-use royalty fees or complex and restrictive revenue formulas.

Table 1 – Key criteria in DSP vs. FPGA choice

all that is required. These steps can be reduced through the many focused courses that are available.

The exact steps will vary depending on an engineer's background and expertise. Specifically in the DSP category are classes in algorithm development, efficient implementation and System Generator design. If you wish to become proficient with DSP design in FPGA, a great starting point would be three courses offered by Hardent and other Xilinx Authorized Training Partners: DSP Primer, Essential DSP

Implementation Techniques for Xilinx FPGAs and DSP Design Using System Generator.

Hardent also offers general courses describing Xilinx devices, HDL design entry languages, optimization techniques, and design and debug strategies. Specialized courses and workshops focus on high-speed I/O considerations, embedded processing and DSP design techniques.

Consult www.hardent.com/training for more on the Hardent training schedule. 🌈

Tips and Tricks for Better Floating-Point Calculations in Embedded Systems

Floating-point arithmetic does not play by the rules of integer arithmetic. Yet, it's not hard to design accurate floating-point systems using FPGAs or FPGA-based embedded processors.

by Sharad Sinha

PhD Candidate

Nanyang Technological University, Singapore

sharad_sinha@gmail.com

Engineers tend to shy away from floating-point calculations, knowing that they are slow in software and resource-intensive when implemented in hardware. The best way to understand and appreciate floating-point numbers is to view them as approximations of real numbers. In fact, that is what the floating-point representation was developed for. The real world is analog, as the old saying goes, and many electronic systems interact with real-world signals. It is therefore essential for engineers who design those systems to understand the strength as well as the limitations of floating-point representations and calculations. It will help us in designing more reliable and error-tolerant systems.

Let's take a closer look at floating-point arithmetic in some detail. Some example calculations will establish the fact that floating-point arithmetic is not as straightforward as integer arithmetic, and that engineers must take this into account when designing systems that operate on floating-point data. One important technique can help: using a logarithm to operate on very small floating-point data. The idea is to gain familiarity with a few flavors of numerical computation and focus on design issues. Some of the references at the end of this article will offer greater insight.

As for implementation, designers can use Xilinx® LogiCORE™ IP Floating-Point Operator to generate and instantiate floating-point operators in RTL designs. If you are building floating-point software systems using embedded processors in FPGAs, then you can use the LogiCORE IP Virtex®-5 Auxiliary Processor Unit (APU) Floating Point Unit (FPU) for the PowerPC 440 embedded processor in the Virtex-5 FXT.

FLOATING-POINT ARITHMETIC 101

IEEE 754-2008 is the current IEEE standard for floating-point arithmetic. (1) It overrides the IEEE 754-1985 and IEEE 754-1987 versions of the standard. The floating-point data represented in the standard are:

- Signed zero: +0, -0
- Nonzero floating-point numbers represented as $(-1)^s \times b^e \times m$ where:
 - s is +1 or -1, thus signifying whether the number is positive or negative,
 - b is a base (2 or 10),
 - e is an exponent and
 - m is a number of the form $d_0.d_1d_2\text{-----}d_{p-1}$, where d_i is either 0 or 1 for base 2 and can take any value between 0 and 9 for base 10. Note that the decimal point is located immediately after d_0 .
- Signed infinity: $+\infty$, $-\infty$
- Not a number (NaN) includes two variants: qNaN (quiet) and sNaN (signaling)

The term $d_0.d_1d_2\text{-----}d_{p-1}$ is called the “significand” and e is called “exponent.” The significand has “p” digits where p is the precision of the representation. IEEE 754-2008 defines five basic representation formats, three in base 2 and two in base 10. Extended formats are also available in the standard. The single-precision and double-precision floating-point numbers in IEEE 754-1985 are now referred to as binary32 and binary64 respectively. For every format, there is the smallest exponent, emin, and the largest exponent, emax.

The possible range of finite values that can be represented in a format is determined by the base (b), precision (p) and emax. The standard defines emin as always equal to 1-emax:

- m is an integer in the range 0 to b^{p-1} .

For example, for $p = 10$ and $b = 2$, m is from 0 to 1023.

- e, for a given number, must satisfy $1-\text{emax} \leq e+p-1 \leq \text{emax}$.

For example, if $p = 24$, $\text{emax} = +127$, then e is from -126 to 104.

It should be understood here that floating-point representations are not always unique. For instance, both 0.02×10^1 and $2.00 \times 10^{(-1)}$ represent the same real number, 0.2. If the leading digit—that is, d_0 —is zero, the number is said to be “normalized.” At the same time, there may not be a floating-point representation for a real number. For instance, the number 0.1 is exact in the decimal number system. However, its binary representation has an infinite repeating pattern of 0011 after the decimal. Thus, it is difficult to represent it exactly in floating-point format. Table 1 gives the five basic formats as per IEEE 754-2008.

ERROR IN FLOATING-POINT CALCULATIONS

Since there is only a fixed number of bits available to represent an infinite number of real numbers, rounding is inherent in floating-point calculations. This invariably results in rounding error and hence there should be a way to measure it to understand how far the result is off when computed to infinite precision. Let us consider the floating-point format with $b = 10$ and $p = 4$. Then the number .0123456 is represented as $1.234 \times 10^{(-2)}$. It is clear that this representation is in error by .56 ($=0.0123456-0.01234$) units in the last place (ulps). As another example, if the result of a floating-point calculation is $4.567 \times 10^{(-2)}$ while the result when computed to infinite precision is 4.567895, then the result is off by .895 ulps.

“Units in the last place” is one way of specifying error in such calculations. Relative error is another way of measuring the error that occurs when a floating-point number approximates a real number. It is defined as the difference between the real number and its floating-point representation divided by the former. The relative error for the case when 4.567895 is represented by $4.567 \times 10^{(-2)}$ is $.000895/4.567895 \approx .00019$. The standard requires that the result of every floating-point computation be correct within 0.5 ulps.

FLOATING-POINT SYSTEM DESIGN

When developing designs for numerical applications, it is important to consider whether the input data or constants can be real numbers. If they can be real numbers, then we

Parameter	Binary format (base, b = 2)			Decimal format (base, b = 10)	
	binary32	binary64	binary128	decimal64	decimal128
P, digits	24	53	113	16	34
emax	+127	+1023	+16383	+384	+6144

Table I – Five basic formats in IEEE 754-2008

need to take care of certain things before finalizing the design. We need to examine if numbers from the data set can be very close in floating-point representation, very big or very small. Let us consider the roots of a quadratic equation as an example. The two roots, α and β are given by the following two formulas:

$$\alpha = (-b + \sqrt{b^2 - 4ac}) / 2a ; \quad \beta = (-b - \sqrt{b^2 - 4ac}) / 2a$$

Clearly b^2 and $4ac$ will be subject to rounding errors when they are floating-point numbers. If $b^2 \gg 4ac$, then $\sqrt{b^2} = |b|$. Additionally, if $b > 0$, β will give some result but α will give zero as result because the terms in its numerator get canceled. On the other hand, if $b < 0$, then β will be zero while α will give some result. This is erroneous design, since roots that become zero unintentionally because of floating-point calculations, when used later in some part of the design, can easily cause a faulty output.

One way to prevent such erroneous cancellation is to multiply the numerator and denominator of α and β by $-b - \sqrt{b^2 - 4ac}$. Doing so will give us:

$$\alpha' = 2c / (-b - \sqrt{b^2 - 4ac}) ; \quad \beta' = 2c / (-b + \sqrt{b^2 - 4ac})$$

Now, when $b > 0$, we should use α' and β to get the values of the two roots and when $b < 0$, then we should use α and β' to obtain their values. It is difficult to spot these discrepancies in design. That's because on an integer data set, the usual formulas will give the correct result yet they will cause the design to return an incorrect result in the case of floating-point data.

Another interesting example comes during the calculation of the area of triangles. Heron's formula for calculating the area of a triangle gives the area A of a triangle if its sides are known using the following equation:

$$A = \sqrt{s(s-a)(s-b)(s-c)}$$

where a, b and c are the sides of a triangle and $s = (a+b+c)/2$.

For a flat triangle—that is, one in which one side is approximately equal to the sum of two other sides, say $a \approx b+c$ —then $s \approx a$. Since two nearby numbers will be subtracted in $(s-a)$, $(s-a)$ can become equal to zero. Or if only s or only a has a rounding error, it can result in a large ulps error. It is better to rewrite these kinds of equations in the format given below, which was first suggested by the mathematician and computer scientist William Kahan (2) in 1986:

$$A = (\sqrt{((a+(b+c))(c-(a-b))(c+(a-b))(a+(b-c)))) / 4 ; a \geq b \geq c$$

Let us see the result when computed using both Heron's and Kahan's formulas for the same set of values. Let $a = 11.0$, $b = 9.0$, $c = 2.04$. Then the correct value of s is 11.02 and that of A is 1.9994918. However, the computed value of s will be 11.05, which has an error of only 3 ulps, the computed value of A using Heron's formula is 3.1945189, which has a very large ulps error. On the other hand, the computed value of A using Kahan's formula is 1.9994918, which is exactly equal to the correct result up to seven decimal places.

HANDLING VERY SMALL AND VERY BIG FLOATING-POINT NUMBERS

Very small floating-point numbers give even smaller results when multiplied. For a designer, this could lead to underflow on a system and the final result would be wrong. This problem is acute in disciplines that deal with probabilities like computational linguistics and machine learning. The probability of any event is always between 0 and 1. Formulas that would give perfect results with integer numbers would cause underflow when working with very small probabilities. One way to avoid underflow is to use exponentiation and logarithms.

You can effectively multiply very small floating-point numbers as follows:

$$a \times b = \exp(\log(a) + \log(b))$$

where the base of exp and log should be the same. However, note that if a and b are both probabilities, they both lie between 0 and 1 and hence their logarithm will fall in the range from $-\infty$ and 0. If you want to deal with only positive probability, multiply the logarithm of the probability by -1 and the results can be suitably interpreted. The other advantage of working with log values is related to speed, since addition, a speedier process, replaces multiplication.

Engineers often come across situations where it's necessary to multiply very big numbers in a series, for instance:

$$k = \prod_{i=1}^v m_i$$

The value of k can easily overflow in some cases. One way to deal with such cases is to compute k as follows:

$$k = \exp\left(\sum_{i=1}^v \log(m_i)\right)$$

Of course, it does complicate the calculation a bit, but if ensuring that the design does not fail for such corner cases is paramount, it is a small cost to pay.

SOME COMMON FLOATING-POINT SUBTLETIES

Floating-point arithmetic does not strictly follow certain rules of mathematics that integer numbers adhere to. The mathematical rule of associativity for addition and multiplication does not apply to floating-point numbers, nor does the distributive rule. This is because of two reasons: first, there can be underflow or overflow when the order is changed, and second, there is the problem of round-off error. Here are some examples for floating-point numbers a , b and c :

Example 1: $a \times (b \times c) \neq (a \times b) \times c$

The expression on the right-hand side (RHS) can overflow or underflow when a and b are multiplied. Also, since the product of a and b is computed first, it can have a different round-off error than the product of b and c on the left-hand side (LHS), resulting in different results from the two expressions.

Example 2: $a = (b - c) + c$ is not equivalent to $a = b$

The RHS expression $(b - c)$ can have a value equal to zero if they are sufficiently close in floating-point representation. In that case $a = c$. The round-off error in $(b - c)$ can lead to a result different from b .

Example 3: $a = b + b \times c$ is not equivalent to $a = b \times (1.0 + c)$

Due to round-off error, the expression $(1.0 + c)$ can have a result that will change the value so as to be different from the initial expression.

Example 4: $b = c / 10.0$ is not equivalent to $b = c \times 0.1$

Again, $c / 10.0$ can have a different round-off error than $c \times 0.1$, resulting in different values of b . Also, the number 10.0 has an exact representation in base 2 while the number 0.1 has an inexact representation. The number 0.1 has an infinite repeating pattern of 0011 in its binary representation. Such numbers, which cannot be represented finitely in a particular base, are said to be inexact numbers in that base, while those that can be are called exact numbers.

Assuming c can always be represented as an exact number, if it is divided by an exact number in one case but multiplied by an inexact number in the other case, the results will be different. Since c itself can be exact or inexact depending on how it was calculated earlier, the equivalence relation does not hold between the two expressions above.

Example 5: $x / x = 1.0$

This equation does not hold in cases where x can be zero, infinite or not a number. NaN cases arise when invalid mathematical operations are performed, such as taking the square root of a negative number or dividing a zero by another zero. It is

the programmer's responsibility to decide what to do next in case of a NaN assertion. Note that as per IEEE 754-2008, NaNs are represented as floating-point numbers with the exponent $\text{emax} + 1$ and nonzero significands. Since you can place system-dependent information into the significand, the result is a bunch of NaNs and not a unique NaN. Thus, when dealing with floating-point code, do not replace x / x with 1.0 unless you are sure that x will not be zero, infinite or NaN.

Example 6: $x - x = 0.0$

This expression is not true if x is infinite or NaN. Hence, do not replace $x - x$ with 0.0 if you are not sure about what values x can take. Engineers often implement such replacement strategies in compilers as optimization steps. However, as can be seen, these kinds of optimizations can lead to erroneous results.

Example 7: $x + 0 = x$

The above equality does not hold when x itself is -0 (remember that there are +0 and -0). This is because $(-0) + (+0) = +0$ in the default rounding direction in IEEE 754-2008, which is round to even.

The IEEE standard defines +0 and -0 so as to find the sign of an answer. For instance $4 \times (+0) = +0$ and $(+0) / (-4) = -0$. For comparison purposes $+0 = -0$ and hence statements like $\text{if}(x = 0)$ need not be concerned with the distinction between +0 and -0.

FURTHER INSIGHT

Along with Kahan's work, engineers looking to explore some of the subtleties involved in working with floating-point data will gain further insight from D. Goldberg's *What Every Computer Scientist Should Know About Floating-Point Arithmetic*, Revision A (Mountain View, Calif., June 1992). Also, there are many excellent references explaining IEEE 754-2008 in detail as well as other standards that affect floating-point design, like ISO/IEC 9899:201x. [3]

Electronics industry sectors like avionics and safety-critical embedded systems require development of very accurate and error-free hardware and software. Such pieces of hardware deal with a lot of floating-point data, and therefore it is essential that designers appreciate the need to respect floating-point arithmetic as a little different from integer arithmetic. 🌈

REFERENCES

1. IEEE Std 754-2008, *IEEE Standard for Floating Point Arithmetic*
2. W. Kahan, "Calculating Area and Angle of a Needle-like Triangle," unpublished. Draft available at <http://www.cs.berkeley.edu/~wkahan/>
3. N1548, ISO/IEC 9899:201x, *Programming languages - C*

How to Build a Self-Checking Testbench

Testbenches, especially those you will use often or for a number of similar projects, should be self-checking and time-agnostic. Here are some tips on the best way to construct them.

by William Kafig

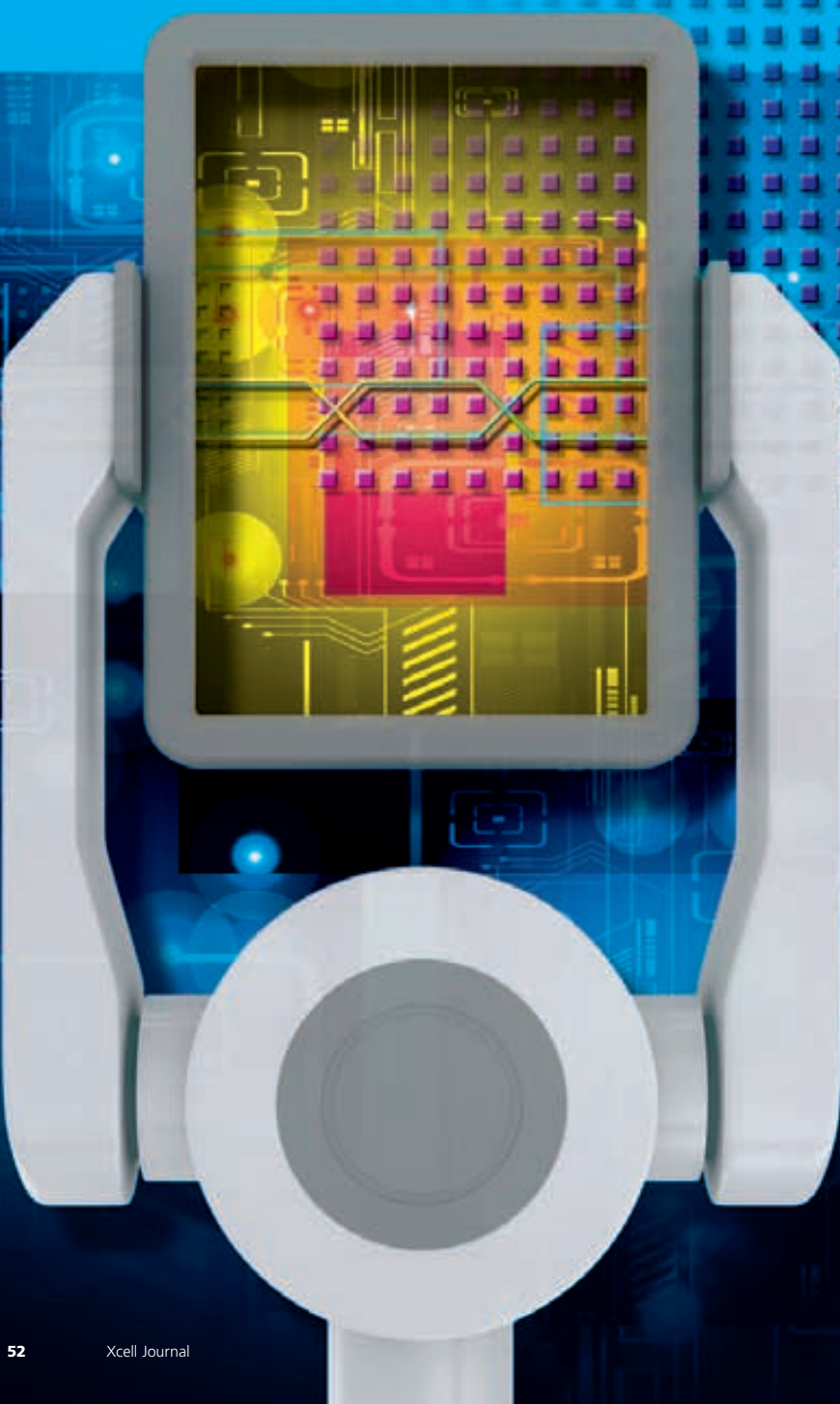
Senior Content Development Engineer
Xilinx, Inc.

bill.kafig@xilinx.com

A testbench, as it's known in VHDL, or a test fixture in Verilog, is a construct that exists in a simulation environment such as ISim, ModelSim or NCsim. Simulation enables a unit under test (UUT)—typically, your synthesizable FPGA design—to connect to virtual (simulated) components such as memory, communication devices and/or CPUs, and be driven with a known set of stimuli. These stimuli cause the UUT to react and interact with the virtual components. You can view both the stimulus and the reaction as waveforms in the simulation environment.

Here's quick example to illustrate how to implement a testbench using a simple 8-bit up/down with reset as the FPGA design (UUT). The testbench provides clock, up/down, enable and reset control signals. Figure 1 shows how to connect the UUT (central gray box) to a testbench.

The various functions on the left side of the diagram provide stimulus for the UUT which, in turn, produces a series of waveforms displayed in the simulation environment. Figure 2 shows a quick screen shot of the waveform view, both zoomed in and zoomed out. How clearly do you see the results? Do you see the values ascending, then descending on the count_out_pins? What if we zoom in (blown-up circle)?



Now we can see the values of the outputs, but we also need to validate each and every single value. That would be 256 values for increment and another 256 for the decrement, along with the special cases of enable off and reset and how these control signals affect the count value. While tolerable for a small design, this is a painful and ineffective means of verifying a larger, more sophisticated design.

Indeed, verifying a significant FPGA design can be more difficult and time-consuming than creating the synthesizable design. Given a known set of stimuli, a design should *always* produce a predictable collection of results—regardless of whether the simulation is behavioral, netlist or full timing (post place-and-route).

There are certainly many ways to create a stimulus set and drive it into the simulation—direct input from the simulation console, input from a text or binary file, scripted stimulus (.do or .tcl) and so on (Figure 3).

The question becomes, after the simulation has run, how do you verify that it was successful—that the generated output matches the predicted output as driven by the stimulus? Certainly you could pore over many dozens to hundreds (or thousands) of waveforms, but this is extraordinarily tedious and time-consuming, not to mention error-prone. You could dump thousands of lines of text to disk, thus requiring someone to examine each and every message—likewise tedious and error-prone.

You could even write a testbench that painstakingly describes each output waveform and tests specific values of the waves at strategic points in time. This type of testbench could then “automatically” detect errors. A wonderful thing, yes? Unfortunately, not only would you have to significantly modify this testbench each time the input was changed, but time does not always hold a fixed meaning in the

context of simulation. Behavioral simulation has little concept of how a design may actually be implemented in hardware and no concept of routing delays; netlist simulation knows how the design is implemented, but is likewise ignorant of routing delays. Only post-place-and-route simulation (full timing simulation) can factor in all the delays for an accurate timing model, but this type has the longest simulation time by far. Thus, it is easy to see that the *exact absolute* time cannot hold constant through all these types of simulations.

This means that if a testbench looks for a specific value (output) at a specific point in time, that value is likely not to be in the same place in time in

time the stimulus (test vectors) change, you will need to recode the waveform checker module. Certainly you can use data files as the metric for the waveform checker, but the tedium of calculating the various values and times is still extremely costly.

Clearly, this static approach is not the path to take.

THE ‘TIME-AGNOSTIC,’ SELF-CHECKING TESTBENCH

We’ve just looked at a number of ways *not* to write a testbench. Let’s now consider some specifications for an “ideal” time-agnostic, self-checking testbench.

“Time-agnostic” in this context means that the type of simulation

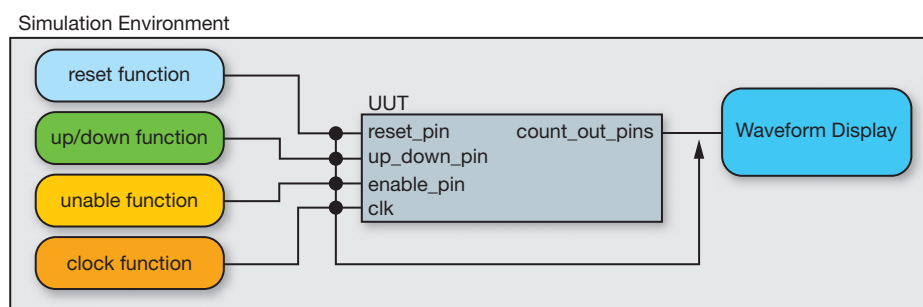


Figure 1 – Simple example of a UUT—that is, your synthesizable design—and how it connects in a testbench



Figure 2 – Zoomed-out waveform shows “big picture,” but lacks details. Zoomed-in waveform shows details, but no context.

the behavioral simulation as it is in the post-place-and-route simulation.

Writing a separate verification testbench for each type of simulation is possible, but demands a significant time investment, not to mention the time spent in figuring out what is “correct” to begin with. Additionally, each

(behavioral, netlist, post place-and-route) and hence the specific timing has no impact on the verification of the UUT. Self-checking means that the testbench is capable of generating its own valid output for each set of stimuli without the user having to explicitly enter it.

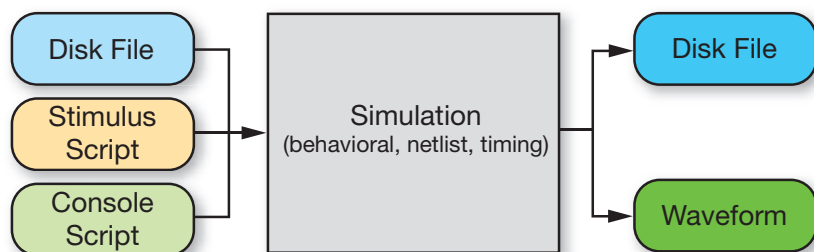


Figure 3 – Generic testbench construction

To use the language of specification, the testbench: shall be valid regardless of the type of simulation being performed; shall operate with any set of valid test vectors; and shall report the status of the various tests in the stimulus set in a clear and easy-to-read format.

You can achieve these specifications by using a technique that creates a parallel flow to the UUT. Stimulus, regardless of its source, is fed to both the UUT and the parallel path. This parallel flow comprises a *behavioral model* of the UUT followed by an asynchronous FIFO that adjusts for any time discrepancies between the behavioral model and the UUT.

A “waveform comparator” detects the new output of the UUT and pulls the next value from the behavioral model’s FIFO. These two values are

then compared and the result is registered, clearly and unambiguously, highlighting the matching and non-matching behaviors between the paths. Figure 4 illustrates a typical model of this type of testbench.

THE CORE OF THE DESIGN

The core of the design consists of the UUT and the known-good behavioral model (KGBM) of that UUT. The UUT is simple enough—it is the synthesizable design you are trying to implement. The KGBM behaves according to the specification of the UUT, but is coded using behavioral constructs.

By constructing a “perfect” version of the UUT using behavioral modeling techniques—which *don’t have to be synthesizable*—you (or rather, your waveform comparator) can quickly identify any differences between the

outputs of the UUT and the KGBM. Behavioral modeling is generally easier (and faster) to code, as it doesn’t need to meet the rigors of synthesis and timing closure.

Construct the behavioral model hierarchically. Once you have created the lowest levels, you can simulate them piecemeal with their own small testbenches to verify proper behavior of the overall model, in the same way as you would simulate a synthesizable design. These small testbenches generally do not need to be time-agnostic or self-checking. Ad hoc (manual inspection of the waveforms) is usually sufficient to validate them.

CREATING STIMULUS

You can create stimulus in a number of ways, starting with console input. Many simulation environments allow the user to “force” values into signals. While this technique may be appropriate for some designs or certain signals, larger, more complex designs benefit from easily repeatable inputs so that it’s easier to perform analysis and debugging.

You can also script stimulus using “tcl” or “do” scripts, depending on which simulation tool you are using. While this approach has a number of benefits, you need to take into account the type of

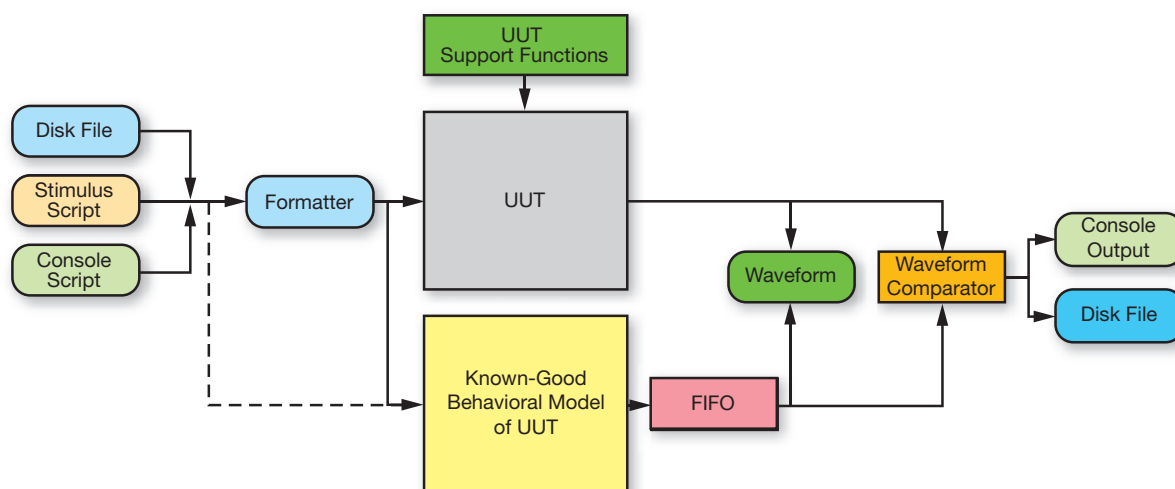


Figure 4 – Generic block diagram for a time-agnostic, self-checking testbench

data that you are pumping into the UUT/KGBM. Scripting is especially attractive in cases where there are a number of inputs that can be simulated in an unpredictable way, such as jitter and pseudo-random noise or data.

The disk file stimulus, provided by a simple text file, is an ideal mechanism for loading simple data streams. An ASCII text file merely contains the values that will be pumped into the design. As an example, let's use the WaveGen design,

just the data itself. This means you can feed the “raw” stimulus directly to the KGBM and use a formatter to package the stimulus for the UUT.

The UUT support-functions block that you've coded for your specific design supplies the necessary clocks, resets and other control signals required to keep the UUT running so that it can process the incoming data. Since the KGBM is not synthesizable, all timing and reset signals can be con-

When the UUT presents its outputs (which will be later in time than the KGBM), the waveform comparator block reads the next piece of data from the FIFO buffering the KGBM data. In this fashion, the relative performance differences between the UUT and the KGBM become irrelevant; therefore, it doesn't matter if the UUT is simulated from the source-level code, from the netlist or from the post-place-and-route design. The performance of the

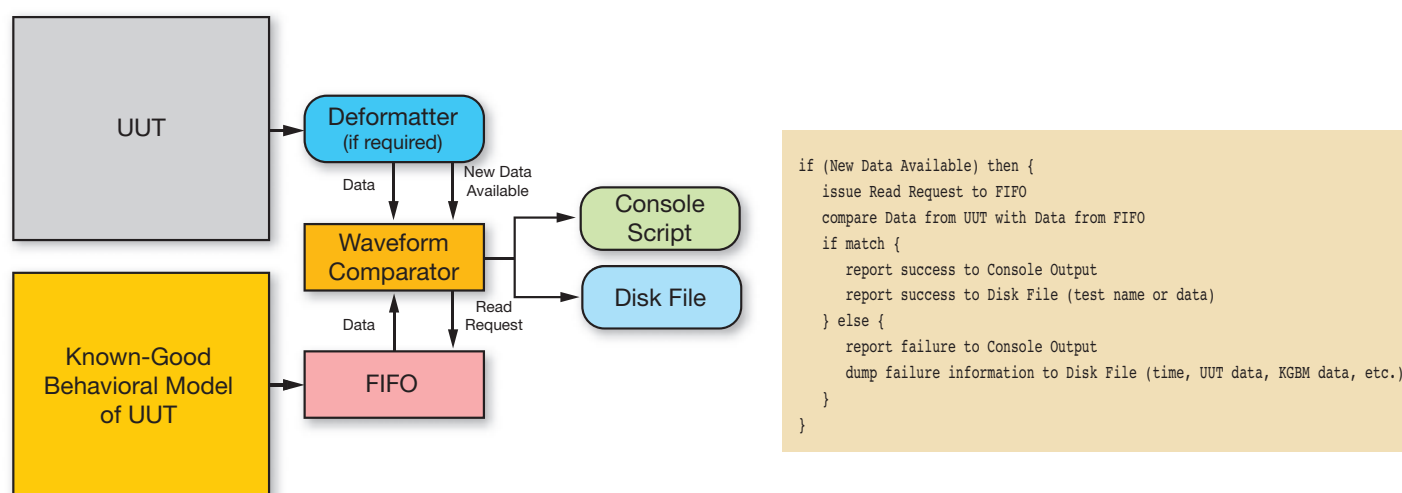


Figure 5 – Block diagram details and pseudo code for the waveform comparator

which Xilinx provides as a reference design with the ISE® tool suite and uses in many customer education classes.

Regardless of the source of the stimulus, you must convert the information into a form that the UUT can digest. You may need a “formatter” of some sort to handle this task. The KGBM, because it needs to model only the *behavior* and not the actual process of the UUT, may be able to use the input stimulus directly. For example, if the UUT needs to process data contained in an Ethernet packet, then the formatter must collect the data from the stimulus source, create a packet and then send the packet to the UUT. The KGBM, on the other hand, doesn't need any header information (unless it needs to make decisions based on this information), but rather

tained within the model or shared with the UUT.

THE ASYNCHRONOUS TIMING FIFO

The UUT is constrained by the speed of its clocks and the latency of the architecture; the KGBM is not. Since the KGBM is constructed as a behavioral model, it is the *function* that is important and not necessarily the timing (the exception being any output that other devices use, such as RS-232, Ethernet and the like). Since the KGBM will produce an output before the UUT, some mechanism must be present to align the outputs of the UUT and KGBM so that they are presented to the waveform comparator block simultaneously. Asynchronous FIFOs excel for this type of task.

UUT is automatically compensated for and no modifications to the testbench are required when the user changes the type of UUT simulation.

WAVEFORM COMPARATOR

As with the stimulus, you may need to compensate for the format of the outputs. The WaveGen example provides its key output in the form of an RS-232 stream. The KGBM can output in “character” form. This implies that there must be some type of UART receiver or equivalent “deformatter” built into the waveform comparator (Figure 5).

With the information from the UUT and the KGBM, the waveform comparator checks to see if there is a match. If there is, data can be marked as “successful” and written out to

either the simulator console output or a disk file. If there is a mismatch between the UUT and KGBM outputs, then the comparator can display an appropriate error message and alert the user to this discrepancy. As with good data, you can see this mismatch immediately on the console or save it to an ASCII disk file for later analysis. The advantage to textual output, whether via the console or an ASCII file, is that of data filtering. Often the user just wants to know if the simulation was successful or not. When not successful, it is helpful to be pointed to the time index in the simulation where the mismatch was found so that you may glean further information

resentation may yield additional information for debugging as it contains all the monitored information at every cycle of the simulator. It is because of this vast amount of data that the user benefits through textual messages to the console and disk to more quickly “see” a successful result, or pinpoint any mismatches.

WAVEGEN EXAMPLE

The WaveGen reference design is a simple design that illustrates several important concepts used in the development of an FPGA, including crossing time domains, synchronizing asynchronous data and multicycle paths, among others. The basic function of the

'Ideal'	VHDL Module Name	Verilog Module Name
Formatter	tb_uart_driver.vhd	tb_uart_driver.v
UUT	wave_gen.vhd	wave_gen.v
KGBM	tb_wave_gen_model.vhd	tb_cmd_gen.v
FIFO	tb_fifo.vhd	tb_fifo.v
Waveform Comparator	tb_resp_checker.vhd	tb_resp_checker.vhd

Table 1 – Mapping of 'ideal' testbench to WaveGen testbench source code

from the waveform view. Generally, you should send only simple and highly relevant information to the console and more detailed information to the text file. In this way you can quickly scan the output of the console to see which tests passed and which failed. You can then reference details of both in the text file.

You may also elect to halt the simulation when a mismatch is found. This is also achievable through the waveform checker module.

For those who enjoy reading waveforms, the ISim simulator tool, along with most other simulators, will display what is happening graphically as a default. This graphical rep-

resentation of WaveGen design is to load a pattern into memory and “play” it back out at varying rates, thus forming a simple waveform generator. For demo boards so equipped, the output can be played through a speaker at audio rates.

The WaveGen design receives an ASCII text command and data via an RS-232 input. Therefore, the stimulus file needs to contain a set of commands to store a pattern in the memory, read some of the pattern back out (as verification that data was properly loaded into memory), configure the “playback” rates and initiate the automatic playback. Table 1 shows the VHDL and Verilog modules needed for the WaveGen testbench.

EASIER CODING

Testbenches, especially those that you will use often or for a number of projects, should be self-checking and time-agnostic. Self-checking is derived by driving the stimulus (inputs) from either a file or script which provides consistent, repeatable and documentable input into the testbench; and the use of a waveform comparator that compares the output of a known-good behavioral model of the design with the UUT. The comparison can be automated and can report the success and failure of various sets of inputs to the console or a text file (or both), thus providing a quick mechanism for the designer to check success or failure.

You can code the KGBM using behavioral modeling techniques rather than strictly synthesizable constructs. Coding in this fashion is easier (hence the time developing the model is significantly shorter than developing the UUT) and produces outputs earlier than the UUT without a significant impact on simulation time. Theoretically, the output values of the KGBM and UUT should match for every input set; therefore, it is easy to detect and report differences. 🌟



Bill Kafig is a senior content development engineer working in the Xilinx Global Training Solutions team. He is currently neck-deep preparing for the release of the

Zynq™ training material and has previously been involved with Xilinx's partial reconfiguration, PCIe®, C-language, VHDL and other classes.

Prior to his joining Xilinx more than four years ago, Bill logged 25+ years of experience spreading the gospel of programmable logic throughout North America. He has done time as a project manager, systems engineer, consultant, FAE and digital grunt for a certain government agency.

Reach him at williamk@xilinx.com.



ALDECTM



Your Global Verification Solutions Provider

VHDL, Verilog, SystemVerilog
OVM/UVM, VMM
Code and Functional Coverage
DSP Co-Simulation (MATLAB® & Simulink®)
Emulation (Hardware-Assisted Verification)

Headquarters-US

2260 Corporate Circle
Henderson, NV 89074
USA

Phone: +702.990.4400
E-mail: sales@aldec.com

Europe

Mercia House
51 The Green, South Bar
Banbury, OX16 9AB
United Kingdom

Phone: +44.1295. 20.1240
Email: sales-eu@aldec.com

Israel

Even Yehuda 40500
6 Macabi St.
POB 2521
Israel

Phone: +972.5.2257.3422
E-mail: sales-il@aldec.com

Japan

Shinjyuku Estate Bldg. 9F
1-34-15, Shinjyuku
Shinjyuku-ku, Tokyo 160-0022
Japan

Phone: +81.3.5312.1791
Email: sales-jp@aldec.com

China

Suite 2004, BaoAn Building
#800 DongFang Road
PuDong District
Shanghai City, 200122, P.R. China

Phone: +86.21.6875.2030
Email: info@aldec.com.cn

India

#4123, 1st Floor
6th Cross, 19th Main
HAL II Stage Indira nagar
Bangalore, 560008, India

Phone: +91.80.3255.1030
Email: sales-in@aldec.com

Taiwan

Room 920, 8f, no.8
Lane 360, sec.1
Neihu Rd., Taipei
Taiwan

Phone: +886.2.2659.9119
E-mail: sales-tw@aldec.com

New Class of DC/DC Turns Down the Noise

The PowerSoC provides the efficiency of a switched-mode converter with the solution footprint and noise characteristics of a linear regulator.

by Michael G. Laflin

Director of Marketing
Enpirion, Inc.
milaflin@enpirion.com

Austin Lesea

Principal Engineer
Xilinx, Inc.
Austin.Lesea@xilinx.com

Conversion efficiency is driving FPGA system designers away from linear regulators and toward switched-mode DC/DC converters. While switched-mode DC/DCs offer dramatic increases in efficiency, they also require a much more complex design, while increasing the parts count and footprint. Most significantly for high-speed I/O, switched-mode DC/DC converters are also a source of noise.

A new class of DC/DC converter called the PowerSoC can minimize the various components of noise, and can power high-speed I/O with performance equivalent to or better than that of a linear regulator.

Introduced by Enpirion in 2004, the PowerSoC integrates the entire DC/DC converter into a single IC package including the controller, gate drivers, MOSFET switches, high-frequency decoupling and, most important, the inductor. Most PowerSoCs require only input and output filter capacitors, so that the overall solution is simple and small.

A very simple synchronous switched-mode DC/DC converter consists of a pair of MOSFET switches, an inductor, and input and output filter capacitors. Figure 1 shows the converter during the switching cycle and its associated DC and AC current paths. When SW1 is closed (SW2 open), current flows from the source through the inductor and to the load, while the input and output filter capacitors “shunt” the high-frequency AC currents. When SW2 is closed (SW1 open), energy stored in the inductor sources the current to the load

through the second half of the switching cycle. The opening and closing of these switches and flow of the high-frequency AC currents create noise.

NOISE AND STRATEGIES FOR ITS MITIGATION

A stepdown DC/DC converter effectively “chops up” a DC voltage into an AC voltage and then filters it back to a pseudo-DC voltage. This process introduces noise of four types: ripple voltage on the converter DC output, ripple voltage on the converter input supply,

radiated electromagnetic interference and conducted EMI.

Every passive electrical component has, besides its basic function (resistance, capacitance, inductance), a parasitic component of the other two: an equivalent series resistance (ESR) and an equivalent series inductance (ESL), in the case of a capacitor. For a resistor, an equivalent series inductance and an equivalent parallel capacitance are present as well.

Output ripple is the byproduct of the shunting off, or flow, of the AC ripple current through the output filter capacitor. Figure 2 shows the small-signal model of the output filter capacitor and the contribution of each element of the model to the output ripple waveform. Note that the ESL of the output filter capacitor combines with the parasitic inductance of the PCB traces’ return, and the internal parasitic inductance of the converter, to create the total ESL of the output filter loop. The ESL creates the large high-frequency spikes through inductive “ringing.”

Most DC/DC converter supplier datasheets show low-pass-filtered ripple waveforms and are thus generally unreliable as an indication of the actual ripple that would be measured on the PCB for a given application.

Fundamentally, to reduce output ripple you can either reduce the magnitude of the ripple current, or reduce the ESR and ESL of the capacitor and the ESL of the PCB traces. Operating at higher switching frequencies will reduce the ripple current for a given inductor value and allows you to use smaller, low-ESR/ESL ceramic capacitors. However, a higher switching frequency increases switching losses in the MOSFET switches and will affect efficiency.

Placing multiple capacitors in parallel can reduce ESR/ESL in the same way that placing resistors in parallel reduces their combined resistance. But adding capacitors increases ESL in the PCB and will increase the PCB real estate the converter consumes.

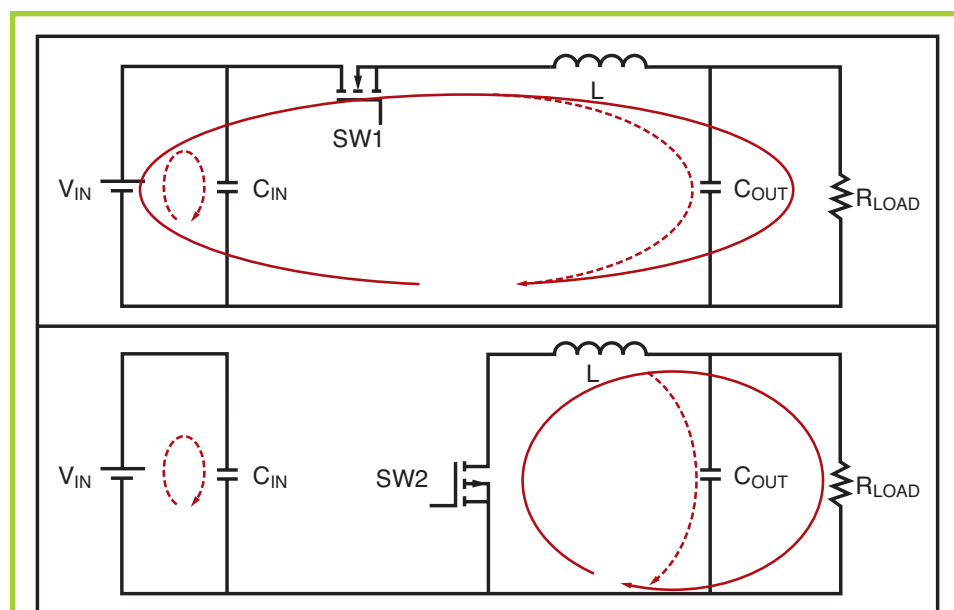


Figure 1 – In this simplified diagram of a synchronous stepdown DC/DC converter, the solid red line shows the flow of DC currents while the dotted red line shows the flow of high-frequency AC current.

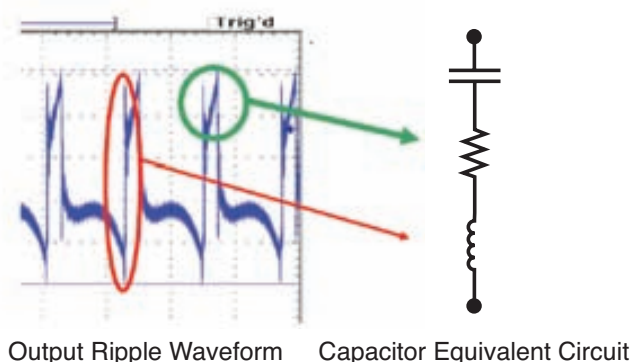


Figure 2 – Output voltage ripple components and sources

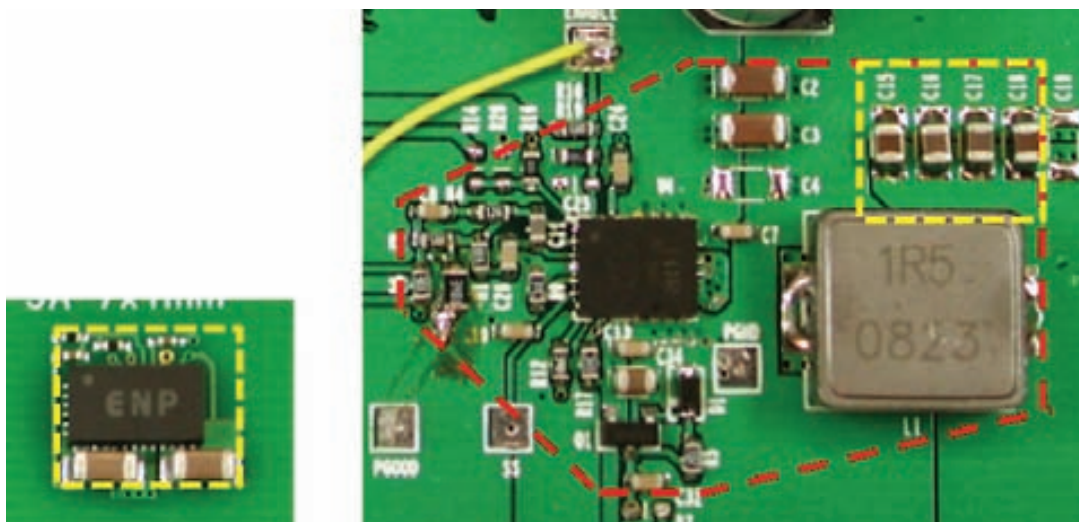


Figure 3 – Converter solution footprint comparison for a typical 4A case. The PowerSoC (left) has much smaller input and output AC current loops and is 1/7 the size of the typical discrete implementation. The dotted yellow rectangle in the photo at right demonstrates the size of the PowerSoC relative to a discrete DC/DC (dotted red line).

You can curb PCB ESL by using smaller filter components (inductor and capacitors) to reduce the length of the PCB. Unfortunately, the smaller inductor will generally result in higher ripple currents without increasing switching frequency.

Another possibility is to use second-stage filtering, such as placing a ferrite bead and capacitor between the DC/DC output filter section and the target load. The disadvantage of this approach is that the additional lossy element will affect regulation and could decrease efficiency.

INPUT VOLTAGE RIPPLE

As the SW1 MOSFET opens and closes, current flows from the source (VIN) with a near-rectangular pulsed waveform. The rise and fall times can be very fast, on the order of a very few nanoseconds.

Much in the same way that the output ripple arises from the ESR and ESL of the output filter capacitor and PCB trace ESL, the input ripple results from the input filter capacitor's ESR and ESL, along with the ESL of the supply PCB trace. However, the magnitude of the input current ripple is much larger, with large changes in current vs. time

(di/dt). Therefore, the impact of PCB inductance is much more important and the input filter capacitor must tolerate higher RMS currents. This high, fast switching current is also the primary source of conducted and radiated EMI.

As with the output filter capacitor, operating at a higher switching frequency allows the use of smaller, lower-ESR/ESL ceramic input filter capacitors. The same cautions apply with regard to higher switching losses.

One mitigation strategy is to minimize parasitic inductances in the input filter loop. The primary way to accomplish this is to place the filter capacitor as close to the DC/DC as possible and make the PCB traces as short and wide as possible. You should generally not place the input filter capacitor on the opposite side of the PCB and connect it to the DC/DC using vias. This will introduce a large amount of inductance in the current loop.

RADIATED EMI

Radiated EMI results from the high, fast switching currents flowing through the input AC current loop. Recall from your electromagnetic-fields courses that the radiation efficiency of a loop antenna is a function

of the loop radius relative to the radiation wavelength.

$$P_{RAD} = \eta \frac{\pi \left(\frac{2\pi r^2}{\lambda} \right)^4}{12} |I_0|^2$$

The equation gives the power radiated by a loop antenna of radius r and wavelength λ ; η is a free-space constant. Note that there is an r^8 relationship with loop radius while the wavelength has a λ^4 relationship. Hence, there is a significant advantage in operating at higher frequencies if it allows you to use smaller components that result in a smaller input current loop radius.

The best mitigation strategy for radiated EMI is to reduce the radius of the input AC current loop. You can do so by switching at higher frequencies that enable the use of smaller ceramic filter capacitors. The same caveat regarding higher switching frequency applies here—namely, higher switch loss.

CONDUCTED EMI

Conducted EMI comes from two primary sources. The first is from the fast switching input currents being pulled

A specialized deep-submicron high-frequency LDMOS process provides low switching loss and enables complete integration of control, drive and switching elements. The low switching loss makes high switching frequencies, typically 5 MHz, possible.

from the input voltage rail, which can cause both supply ripple (differential-mode) and ground bounce (common-mode) EMI. The other significant source of conducted EMI results from the coupling of the inductor magnetic-flux leakage onto PCB traces on the board.

Here, the first mitigation strategy is to properly size the input filter capacitor to supply or filter the high-frequency AC currents so as to minimize the currents on the supply rail.

Also, minimize the parasitic inductance and ESL in the input AC current loop. You can accomplish this by operat-

wide as possible to reduce trace inductance. Finally, use shielded inductors to reduce flux leakage.

POWERSOC AS A STRATEGY TO MITIGATE NOISE

In manufacturing its PowerSoC devices, Enpirion uses a specialized deep-submicron high-frequency LDMOS process to provide low switching loss and to enable complete integration of control, drive and switching elements. The low switching loss makes high switching frequencies, typically 5 MHz, possible.

High-density, high-permeability, low-profile magnetics provide minimal AC

output AC loops very small, reducing ripple and EMI.

The package layout is structured to further minimize the radius of the input and output AC filter loops and thus minimize radiated and conducted EMI and ripple. Package design includes RF techniques to minimize parasitic impedances within the internal circuit elements to keep high-frequency AC currents contained inside the package.

Figures 3 and 4 provide a comparison between PowerSoC and discrete DC/DC converter implementations.

POWERING ROCKETIO USING POWERSOC

We designed and built a daughtercard for the Xilinx® Virtex®-5 development board. We took jitter measurements with the Enpirion devices powering the development board and with linear regulators, with and without second-stage filtering on the PowerSoCs. The PowerSoC came in at 77.2 and 78.3 picoseconds with and without the second-stage filter, respectively. Jitter for the linear regulator was 78 ps.

Clearly, PowerSoCs represent a powerful new tool for the FPGA designer. The devices significantly reduce the many issues encountered when changing over from linear regulator-based voltage converters to the more efficient switched-mode types. PowerSoCs offer similar footprints and ease of design as linear regulators, while providing switched-mode converter efficiencies—but without the noise and complexity of a discrete converter implementation. ●●

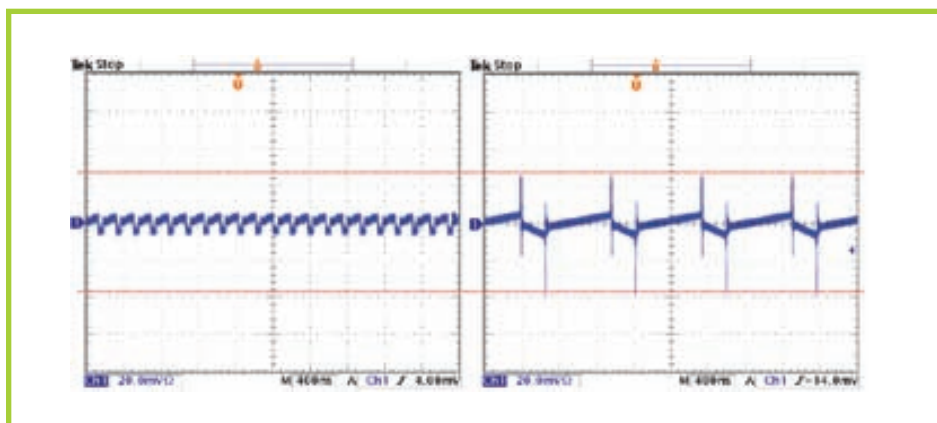


Figure 4 – Output ripple voltage comparison between PowerSoC (left) and discrete DC/DC implementation (right). Ripple measured on vendor evaluation boards using same equipment and technique. Measurement bandwidth is 500 MHz.

ing at higher switching frequencies that enable the use of low-ESL ceramic capacitors; these in turn will enable a smaller loop radius. Once again, the same caveats apply with regard to higher switching frequency as switch loss.

In addition, make PCB traces for the input filter capacitor as short and

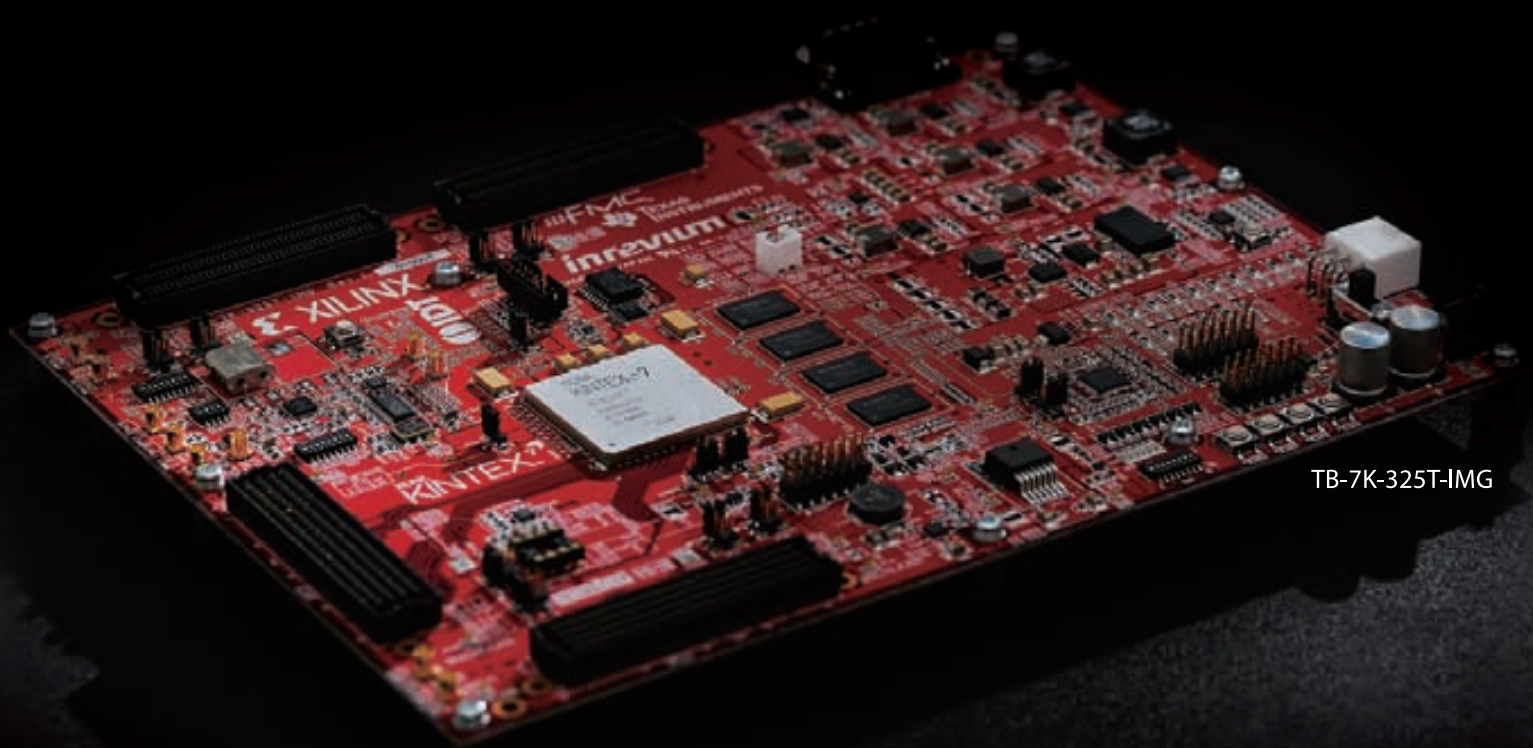
loss with low DC resistance. The magnetics and magnetic structures exhibit self-shielding properties that reduce flux leakage. The high switching frequency allows the inductor to be physically very small, enabling the use of small input and output filter capacitors. This in turn makes the input and

Kintex™-7 FPGA ACDC

(Acquisition, Contribution, Distribution and Consumption) 1.0 Baseboard



A Revolutionary Innovation Platform



TB-7K-325T-IMG

The Kintex-7 FPGA ACDC (Acquisition, Contribution, Distribution and Consumption) 1.0 Baseboard. The processing performance and flexibility provided by 28nm Kintex-7 FPGAs allows manufacturers to build stunningly immersive technology into new consumer TV screens that come with features such as multiple windows/picture-in-picture, 3D graphics for games and ultra realistic viewing that goes far beyond HDTV.

- ▶ **FPGA** : Kintex-7 FPGA XC7K325T-FFG900
- ▶ **Memory** : DDR3 2Gbit x 4 (address shared)
- ▶ **FMC connector**
 - HPC (High Pin Count) x 2
 - LPC (Low Pin Count) x 2
- ▶ **SERDES** : 16ch x 12.5Gbps (GTx)



TOKYO ELECTRON DEVICE LIMITED

Headquarter : Yokohama , Japan
E-mail:psd-sales@teldevice.co.jp
<http://www.inrevium.jp/eng/x-fpga-board/index.html>

US Offices : Fremont , CA
E-mail:psd-nasales@teldevice.co.jp

China Offices : Shanghai
E-mail:sales@teldevice.cn



ALLIANCE PROGRAM
PREMIER MEMBER

FPGAs Made Accessible

JTAG-HS1™

Lowest-cost, highest-speed programming solution for Xilinx® devices

- Compatible w/ all Xilinx tools & devices
- Up to 30 Mbits/sec on JTAG bus
- Supports 6 & 14-pin programming headers
- Volume pricing available

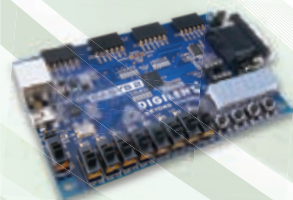
\$49.99

JTAG-SMT1™

Embedded programming for Xilinx® devices

- Compatible w/ all Xilinx tools & devices
- Up to 30 Mbits/sec on JTAG bus
- Surface mounts like any other component
- Volume pricing available

\$27.99
(100+)



BASYS 2™

The Ultimate FPGA Starter

- Spartan® 3E
- USB-powered
- Ideal first time FPGA users

\$49.00
(Student Price)



NEXYS 3™

Spartan®-6 power at an entry-level price

- Spartan®-6 LX16
- 32MB PCM & 16MB RAM
- 10 / 100 Ethernet, USB UART
- USB host for mouse & keyboard

\$119.00
(Academic Price)



ATLYS™

Advanced video processing w/ Spartan-6 power

- Spartan®-6 LX45
- 128MB DDR2 & 16MB PCM
- Gigabit Ethernet, USB UART
- 4 HDMI ports, AC97 audio

\$199.00
(Academic Price)

(509) 334-6306



DIGILENT®
BEYOND THEORY

www.digilentinc.com

Xilinx Tool & IP Updates

Xilinx is continually improving its products, IP and design tools as it strives to help designers work more effectively. Here, we report on the most current updates to the flagship FPGA development environment, the ISE® Design Suite, as well as to Xilinx® IP. Product updates offer significant enhancements and new features to four versions of the ISE Design Suite: the Logic, Embedded, DSP and System Editions. Keeping your installation of ISE up to date is an easy way to ensure the best results for your design. On Jan. 18, 2012, Xilinx released ISE Design Suite 13.4 and made it available for download at www.xilinx.com/download.

For more information or to download a free 30-day evaluation of the ISE Design Suite, visit www.xilinx.com/ise.

DOCUMENTATION NAVIGATOR

Xilinx Documentation Navigator is designed to help you intuitively find and download documentation customized to your needs. You can also trust that you are reading the latest information through the built-in alert mechanism. The alert system lets you know when new documentation is available for download and when downloaded documentation is updated on the Web. You can download Xilinx Documentation Navigator from the Xilinx Support Site at <http://www.xilinx.com/support>.

ISE DESIGN SUITE: LOGIC EDITION

Front-to-Back FPGA Logic Design

Latest version number: 13.4

Date of latest release: January 2012

Previous release: 13.3

Revision highlights:

The ISE Design Suite 13.4 release enhances support for Artix™-7 FPGAs, and also provides support for the Virtex®-7 XT devices with partial reconfiguration.

MicroBlaze™ Microcontroller System (MCS)

With the 13.4 release, Xilinx introduces the MicroBlaze Microcontroller System (MCS). MicroBlaze MCS delivers the power of a MicroBlaze-based processing system in all ISE Design Suite Editions and ISE WebPACK™. Designers can now quickly and easily design a microcontroller without the need for an ISE Embedded Edition license or for Xilinx Platform Studio. MicroBlaze MCS features a fixed-configuration MicroBlaze core, general-purpose input and output peripherals, fixed and general-purpose timers, UART, interrupt controller and I/O connector. The Software Design Kit supports

MicroBlaze MCS. Now both logic designers and embedded designers can make use of the same device support and leverage the power of the same industry-leading MicroBlaze soft microprocessor.

RX Margin Analysis in ChipScope™

Pro: This release includes a new feature enabling users to analyze the margin of high-speed serial I/O channels using 2D statistical eye scan algorithms via the ChipScope Pro Serial I/O toolkit. RX Eye Scan, through the monitoring of 7 series FPGA GTX transceivers, delivers optimal signal quality and a lower bit-error ratio (BER). Additional use modes enable the diagnosis of the effects of equalization settings.

PlanAhead™ design and analysis

tool: This release provides usability improvements, extending user productivity in synthesis and implementation and assisting users in closing designs. The tool also supports intelligent clock-gating, team design and fifth-generation partial reconfiguration technology now being offered in Artix-7 and Virtex-7 XT devices, further managing power consumption.

ISE DESIGN SUITE: EMBEDDED EDITION

Integrated Embedded Design Solution

Latest version number: 13.4

Date of latest release: January 2012

Previous release: 13.3

Revision highlights:

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

Synopsys VCS support: New to the Embedded Edition is Synopsys VCS simulation support for the Embedded Development Kit (EDK). The VCS Simulator provides another processing option, further increasing productivity and reducing verification time.

ISE DESIGN SUITE: DSP EDITION

For High-Performance DSP Systems

Latest version number: 13.4

Date of latest release: January 2012

Previous release: 13.3

Revision highlights:

All ISE Design Suite editions include the enhancements listed above for the Logic Edition.

Updates specific to the DSP Edition include Mentor Graphics Questa Advanced Simulator support for the HDL import flow. Other enhancements include a 7x simulation performance speedup for FIR Compiler v6.3, AXI4 support, and a Viterbi decoder, convolution encoder and RS encoder.

XILINX IP UPDATES

Name of IP: ISE IP Update 13.4

Type of IP: All

Targeted application: Xilinx develops IP cores and partners with third-party IP providers to decrease customer time-to-market. The powerful combination of Xilinx FPGAs with IP cores provides functionality and performance similar to ASSPs,

but with flexibility not possible with ASSPs.

Latest version number: 13.4

Date of latest release: January 2012

Installation instructions:

www.xilinx.com/ipcenter/coregen/ip_update_install_instructions.htm

Listing of all IP in this release:

www.xilinx.com/ipcenter/coregen/13_4_datasheets.htm

Revision highlights:

CORE Generator™ IP is equipped with Artix-7 and Virtex-7 XT support. Artix-7 and Virtex-7 XT device family support in the ISE 13.4 tools is public access. The set of cores supporting Artix-7 and Virtex-7 XT provides preproduction support for these two device families.

Updates to existing IP:

- **Bus Interface and I/O**

- PCI v4 - Artix-7 support
- XAUI v10.1 - DXAUI (4 x 6.25G) support for 7 series FPGA devices

- **Communication and Networking**

- Aurora 64B/66B - AXI4-Stream support and support for Virtex-7 and Kintex™-7 GTX transceivers

- **7 Series FPGA Transceiver Wizard v1.6:**

The wizard added the following protocol support for 7 series GTH transceivers: CEI-6 and Interlaken at 6.5 Gbps; 10GBase-KR at 10.3125 Gbps; CEI-11 at 11.1 Gbps; CAUI at 10.3125 Gbps; and OTU-4 at 11.18 Gbps. An additional “start from scratch” feature for 7 series GTH transceivers enables users to configure the GTH primitive for other protocols.


In addition, Xilinx has modified the wizard's GUI to display 20/40 as the internal data path width for 8B10B encoding, and updated XAUI and RXAUI (bringing optional ports to the top-level wrapper). Release 13.4 also adds QSGMII and PCIe® Gen1/Gen2 Protocol (PIPE wrapper delivery through wizard) protocol support for 7 series GTX transceivers. Xilinx has updated the attribute to support General ES 3.1 Silicon (GTX) using characterization updates.

Additional IP supporting AXI4 interfaces:

The latest versions of CORE Generator IP have been updated with production AXI4 interface support. For more details, see http://www.xilinx.com/ipcenter/axi4_ip.htm. In general, the AXI4 interface is supported by the latest version of an IP for Virtex-7, Kintex-7, Virtex-6 and Spartan®-6 device families. Older “production” versions of IP continue to support the legacy interface for the respective core on Virtex-6, Spartan-6, Virtex-5, Virtex-4 and Spartan-3 device families only.

For general information on AXI4 support, see <http://www.xilinx.com/ipcenter/axi4.htm>

13.4 CORE Generator enhancements

- 2x improvement in CORE Generator startup speed
- “Get License” button on the “View License Status” dialog provides quick access to the Xilinx Product Licensing site, enabling users to generate license keys for evaluation IP and IP for which they have purchased licenses. 

Xpress Yourself in Our Caption Contest

DANIEL GUIDERA



If you've got an urge to let your sense of humor take wing, here's your opportunity. We invite readers to step up to our verbal challenge and submit an engineering- or technology-related caption for this cartoon by Daniel Guidera, showing an unusual avian visitor to the conference room. The image might inspire a caption like "I told Steve not to open the windows so wide!"

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 and two runners-up will each receive an Avnet Spartan®-6 LX9 MicroBoard, an entry-level development environment for evaluating the Xilinx® Spartan®-6 family of FPGAs (<http://www.xilinx.com/products/boards-and-kits/AES-S6MB-LX9.htm>).

The deadline for submitting entries is 5:00 pm Pacific Time (PT) on April 1, 2012. So, put down your bird seed and get writing!

JOEL OWENS, an engineering technician at Rockwell Collins Inc. at Dyess Air Force Base in Texas, won first place with this caption for the cartoon showing food service overkill from Issue 77 of *Xcell Journal*:



DANIEL GUIDERA

"Do you get the feeling our deadline is about to be pushed up?"

Congratulations as well to our two runners-up. Like our winner, they will each receive an Avnet Spartan®-6 LX9 MicroBoard.

"Don't get me wrong... I like it, but this is not what I had in mind when I asked you to build a java repository."

– *Luther Davis, electronics engineer, Northrop Grumman Electronic Systems*

"...and finally, in regard to our recent request to allocate more person-hours to the project...I believe management has given us their answer."

– *Jeff Saarela, electrical engineer, Astronautics Corporation of America*

Now Available!

The industry's first methodology manual for FPGA-based prototyping of SoC Designs



FPGA-Based Prototyping Methodology Manual:

Best Practices in Design-for-Prototyping

Go to:

<http://www.synopsys.com/fpmm>

- ▶ Purchase the book or download a free ebook version
- ▶ Become involved with the online community for prototypers



Twice the performance. Half the power.



Innovate without compromise with the Xilinx 7 series FPGAs. The new 7 series devices are built on the only unified architecture to give you a full range of options for making your concepts a reality. Meet your needs for higher performance and lower power. Speed up your development time with next-generation ISE® Design Suite. And innovate with the performance and flexibility you need to move the world forward.

www.xilinx.com/7

ARTIX⁷
LOWEST SYSTEM COST

KINTEX⁷
BEST PRICE / PERFORMANCE

VIRTEX⁷
HIGHEST SYSTEM PERFORMANCE