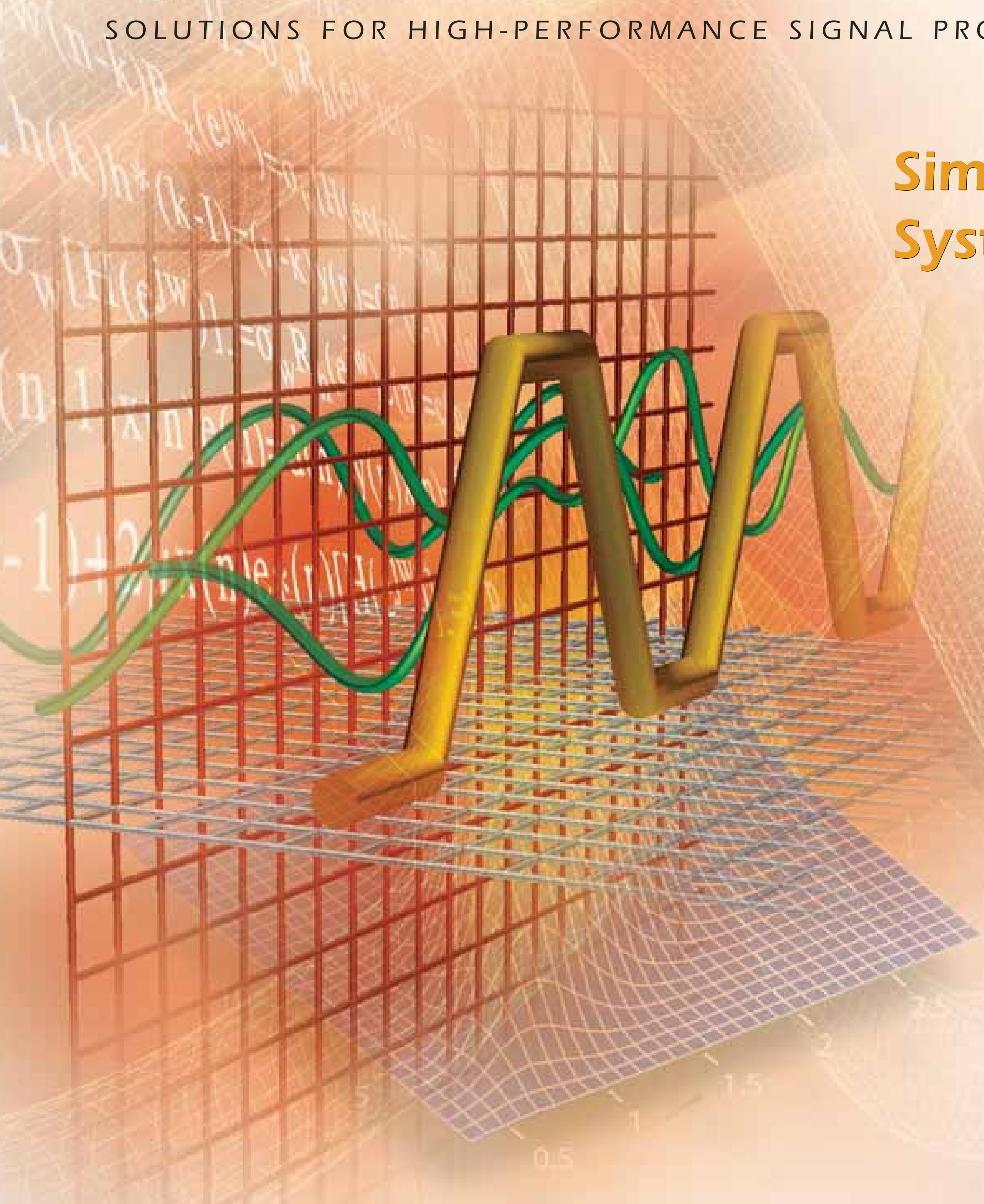
Issue 1
October 2005

DSP magazine

SOLUTIONS FOR HIGH-PERFORMANCE SIGNAL PROCESSING DESIGNS



Simplifying DSP System Designs

INSIDE

FPGA-Based MPEG-4 Codec

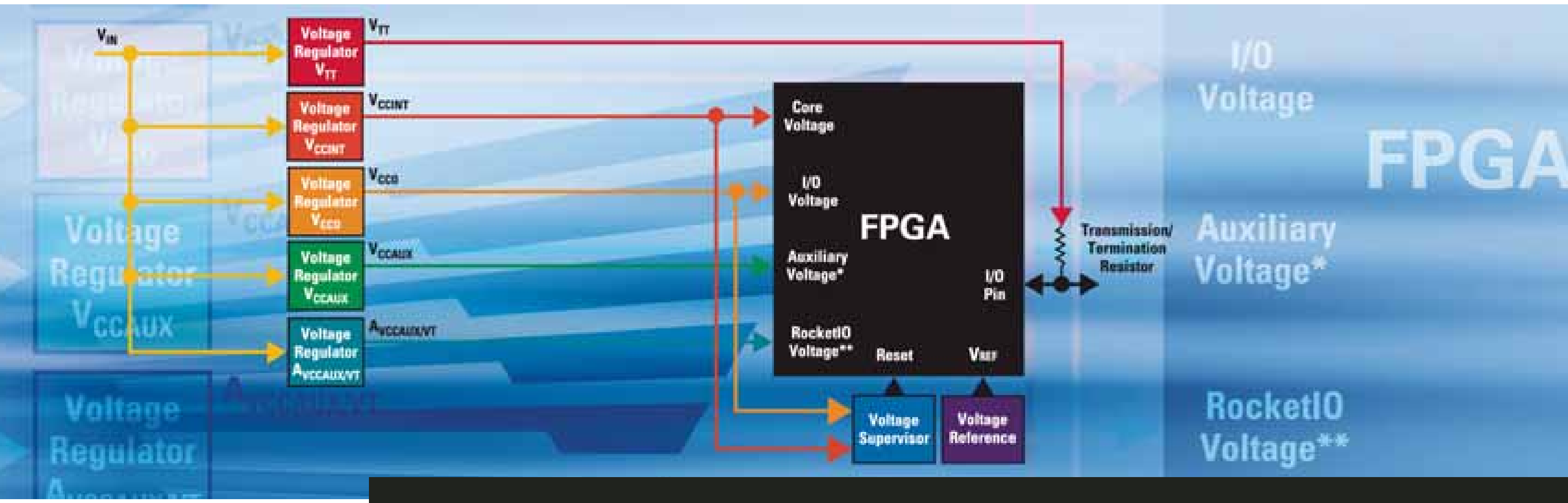
Implementing Matrix Inversions in Fixed-Point Hardware

Designing with the Virtex-4 XtremeDSP Slice

The Design and Implementation of a GPS Receiver Channel

XILINX®

Support Across The Board.™



Power Management Solutions for FPGAs

National Devices supported:

- Voltage Regulators
- Voltage Supervisors
- Voltage References

Xilinx Devices supported:

- Virtex™
- Virtex-E
- Virtex-II
- Virtex-II Pro
- Virtex-4FX, 4LX, 4SX
- Spartan™-II
- Spartan™-IIE
- Spartan-3, 3E, 3L

National Semiconductor
The Sight & Sound of Information

XILINX®

Avnet Electronics Marketing has collaborated with National Semiconductor® and Xilinx® to create a design guide that matches National Semiconductor's broad portfolio of power solutions to the latest releases of FPGAs from Xilinx.

Featuring parametric tables, sample designs and step-by-step directions, this guide is your fast, accurate source for choosing the best National Semiconductor Power Supply Solution for your design. It also provides an overview of the available design tools, including application notes, development software and evaluation kits.

Go to em.avnet.com/powermgtguide to request your copy today.

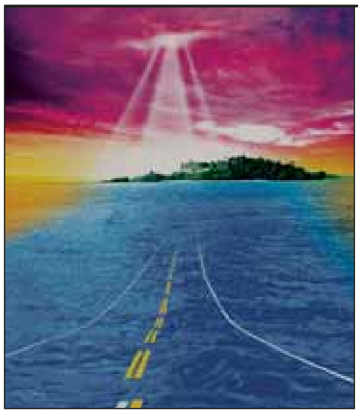
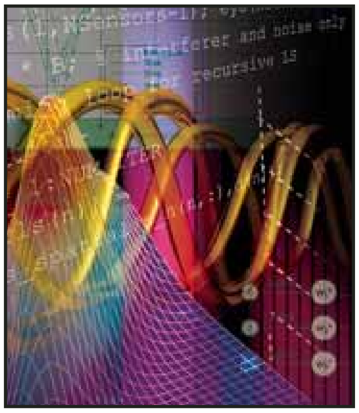
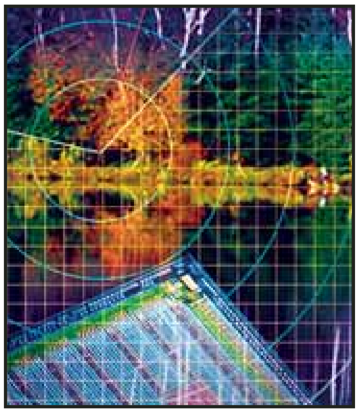
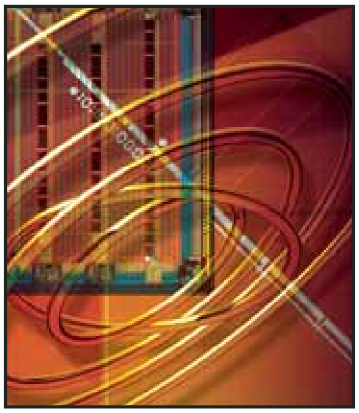
AVNET® electronics marketing

Enabling success from the center of technology™

1 800 332 8638
www.em.avnet.com

NEDA Supplier Authorized Distributor

© Avnet, Inc. 2005. All rights reserved. AVNET is a registered trademark of Avnet, Inc.

CONTENTS

Welcome 4

VIEWPOINT

Setting Industry Direction for High-Performance DSP 5

MULTIMEDIA, VIDEO, and IMAGING

FPGA-Based MPEG-4 Codec 8

Rapid Development of Video/Imaging Systems 10

Encoding High-Resolution Ogg/Theora Video with Reconfigurable FPGAs 13

Implementing DSP Algorithms Using Spartan-3 FPGAs 16

Using FPGAs in Wireless Base Station Designs 20

Accelerated System Performance with APU-Enhanced Processing 24

Alpha Blending Two Data Streams Using a DSP48 DDR Technique 28

DEFENSE SYSTEMS

Implementing Matrix Inversions in Fixed-Point Hardware 32

Integrating MATLAB Algorithms into FPGA Designs 37

Software-Defined Radio: The New Architectural Paradigm 40

Virtex-4 FPGAs for Software Defined Radio 44

DIGITAL COMMUNICATION

Real-Time Analysis of DSP Designs 46

The Design and Implementation of a GPS Receiver Channel 50

GENERAL PURPOSE AND IMPLEMENTATION

Designing Control Circuits for High-Performance DSP Systems 55

Signal Processing Capability with the NuHorizons Spartan-3 59

Designing with the Virtex-4 XtremeDSP Slice 62

Synthesis Tool Strategies 66

CUSTOMER SUCCESS

A Weapon Detection System Built with Xilinx FPGAs 68

EDUCATION

DSP Design Flow – Intermediate Level 72

PRODUCT BRIEFS

Virtex-4 SX 35 XtremeDSP Development Kit for Digital Communication Applications 74

Virtex-II Pro XtremeDSP Development Kit for Digital Communication Applications 76

Virtex-4 DSP Brochure 79

DSP magazine

EDITOR IN CHIEF	Carlis Collins carlis.collins@xilinx.com 408-879-4519
EXECUTIVE EDITOR	Forrest Couch forrest.couch@xilinx.com 408-879-5270
MANAGING EDITOR	Charmaine Cooper Hussain
ONLINE EDITOR	Tom Pyles tom.pyles@xilinx.com 720-652-3883
ART DIRECTOR	Scott Blair
ADVERTISING SALES	Dan Teie 1-800-493-5551

Xcell PUBLICATIONS

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

© 2005 Xilinx, Inc. All rights reserved. XILINX, the Xilinx Logo, and otherdesignated brands included herein are trademarks of Xilinx, Inc. PowerPC is a trademark of IBM, 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.

High-Performance DSP – Vision, Leadership, Commitment

FPGAs are increasingly being used for signal processing applications. They provide the necessary performance and flexibility to tackle many of today's most challenging DSP applications, from MIMO digital communication systems to H.264 encoding to a high-definition broadcast system.

Within such systems, FPGAs are ideally suited for high-performance signal-processing tasks traditionally serviced by an ASIC or ASSP. But you can also use FPGAs to create high-performance DSP engines that boost the performance of your programmable DSP system by performing complementary co-processing functions.

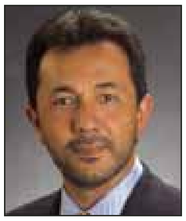
This unique coupling of high performance and flexibility – through exploiting parallelism and hardware reconfiguration – places Xilinx in an ideal position to set the industry direction in the high-performance segment of the DSP market.

Our DSP vision is built on five key pillars:

- **Customer and market focus** – we will create products that meet the needs of our customers and create products in those market segments that are the best fit for our FPGAs.
- **Design methodology** – as most DSP designers don't speak VHDL or Verilog, we will continue to evolve software technologies to support languages that they do speak – like Simulink and MATLAB.
- **Tailored system solutions** – this includes algorithms, tools, services, and devices for focus markets.
- **Ecosystem** – partnerships/alliances with industry leaders like Texas Instruments, The MathWorks, and Xilinx Global Alliance members to deliver total DSP solutions.
- **Awareness** – educating you on how to quickly access FPGAs for signal processing regardless of your background skill set.

This month we are also launching new DSP Roadmaps for the high-performance segment of the DSP market. These roadmaps cover many areas, including digital communications, multimedia video and imaging, defense systems, design tools and methodologies, development platforms, and base IP solutions. The roadmaps demonstrate our continued investment and commitment in solving your current and future signal-processing challenges.

Finally, we are proud to deliver to you the first edition of *DSP Magazine*. Packed with articles demonstrating how you can create optimized DSP designs using FPGAs, this magazine is one of many ways in which we will provide you the knowledge to finish your DSP designs faster. I would like to dedicate this first Xilinx *DSP Magazine* to you, the customer.



Omid Tahernia

Vice President
and General Manager
Xilinx DSP Division

Setting Industry Direction for High-Performance DSP

Xilinx launches new market-focused DSP Roadmaps.

by Jack Elward
Senior Director, Program Management, DSP Division
Xilinx, Inc.
jack.elward@xilinx.com

Have you ever been on a long trip, in somewhat unfamiliar territory, and in search of your next move? You would certainly welcome a map that shows what the road holds in store ahead. Not only is it informative, it can also be reassuring. A good road map will contain enough details about your intended travel path so that you can confidently charge forward or plan for back-ups and alternatives. Of course, sometimes you will want to contact your travel advisor for more details.

Such is the intent of the DSP Roadmap from Xilinx. In publishing the most comprehensive, detailed set of IP, product, and tools plans ever attempted, we intend to shine a floodlight on our next few years of technical releases.

DSP Strategic Pillars

The Xilinx® DSP initiative is based on five strategic pillars:

- Market focus
- Design methodology
- Tailored solutions
- Ecosystem
- Awareness

These pillars are manifested in the DSP Roadmap in the following important ways:

For market focus, we listen to customers and their needs and select high-growth markets where we can add the most value through our products and services. Xilinx target segments include digital communications (both wired and wireless), aeronautics and defense, and MVI (multimedia, video, and imaging). Other DSP markets (such as test and measurement, industrial, and telemetrics) are well served by our current products and their future roadmaps.

Design methodology refers to a growing awareness that traditional users of FPGAs (using VHDL and Verilog) represent only about 10% of the DSP design community. The vast majority of these designers are:

- More familiar with software design tools such as C, C++, and MATLAB,

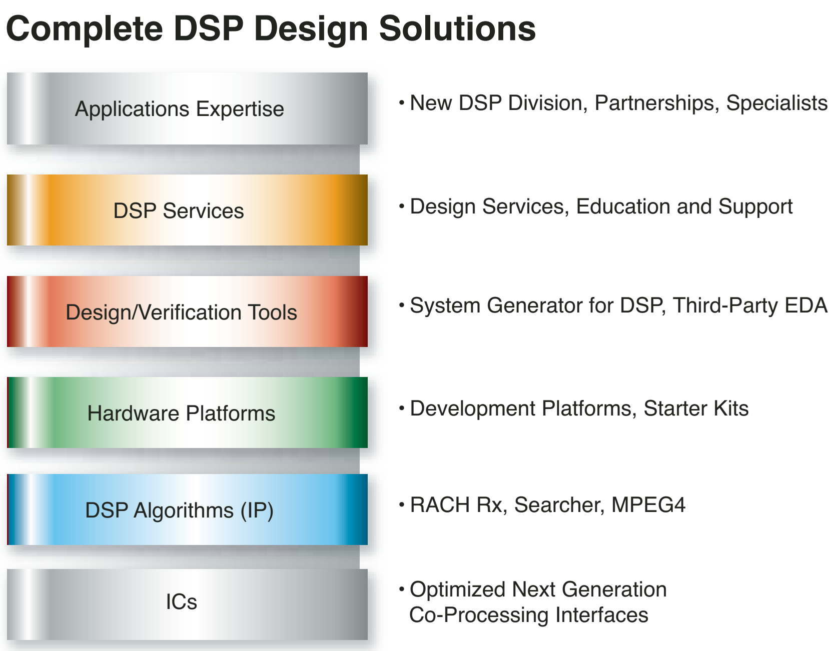

Complete DSP Design Solutions
Applications Expertise
• New DSP Division, Partnerships, Specialists
DSP Services
• Design Services, Education and Support
Design/Verification Tools
• System Generator for DSP, Third-Party EDA
Hardware Platforms
• Development Platforms, Starter Kits
DSP Algorithms (IP)
• RACH Rx, Searcher, MPEG4
ICs
• Optimized Next Generation Co-Processing Interfaces

Figure 1 – Solutions spectum

and a methodology that assumes a robust library of function calls and hardware layer abstraction

- Schooled or experienced in using DSP products from TI, ADI, and Freescale
- In search of higher bandwidth and performance, which can be best delivered through the parallelism of FPGAs
- Concerned with system-level integration, software compatibility and reuse, and rapid prototyping

The tailored solutions strategic pillar is a natural evolution of our traditional building blocks (such as FFT, FIR filters, and other "base blocks") for general DSP applications. There are three clear tines in this fork: IP, tools, and FPGA devices.

In addressing the ecosystem, Xilinx is acknowledging a successful strategy already employed throughout our history. We started off as one of the first fabless semiconductor companies and forged strategic alliances with companies like IBM and TI. Now, with a broad set of IP and tools developed by and offered from third-party vendors, we demonstrate how important it is to go beyond our internal development resources to provide increasingly complete solutions.

Finally, awareness is crucial in affecting the sea change that we desire in positioning Xilinx as a major supplier of DSP solutions. We are clearly positioned and recognized as the world leader in programmable logic, but traditional customer surveys of "DSP supplier awareness" show that we have an uphill climb in the field of entrenched DSP providers such as TI. The roadmaps are a primary vehicle in communicating the expansion of expertise and product offerings, which the recently formed DSP division is capable of delivering.

The DSP Roadmaps cover a broad range of products and services. Figure 1 shows the solution spectrum, ranging from DSP devices to design tools and design services. Tools are inclusive of IP, libraries, boards, and kits.

IP and Solutions

Traditional offerings for DSP designers have been horizontal in nature and apply to market segments. Elements such as FFTs, FIR filters/compilers, encryption, and linear algebra are good examples. The DSP Roadmaps continue to offer enhancements to the functionality and performance, along with forward migration into new generations of FPGA families.

We are also introducing new building blocks to work in conjunction with complex, hard IP embedded into Xilinx FPGA families, such as the PowerPC™ 405 processor and DSP48 blocks. These cores include a floating point co-processor connected to the PowerPC through a dedicated hardware port, and several cores embracing the versatility and inherent performance of the DSP48 slices in their cascaded configuration.

The IP offerings are tailored to meet the needs of specific vertical markets. Therefore, we have created roadmaps to address the following areas: Digital

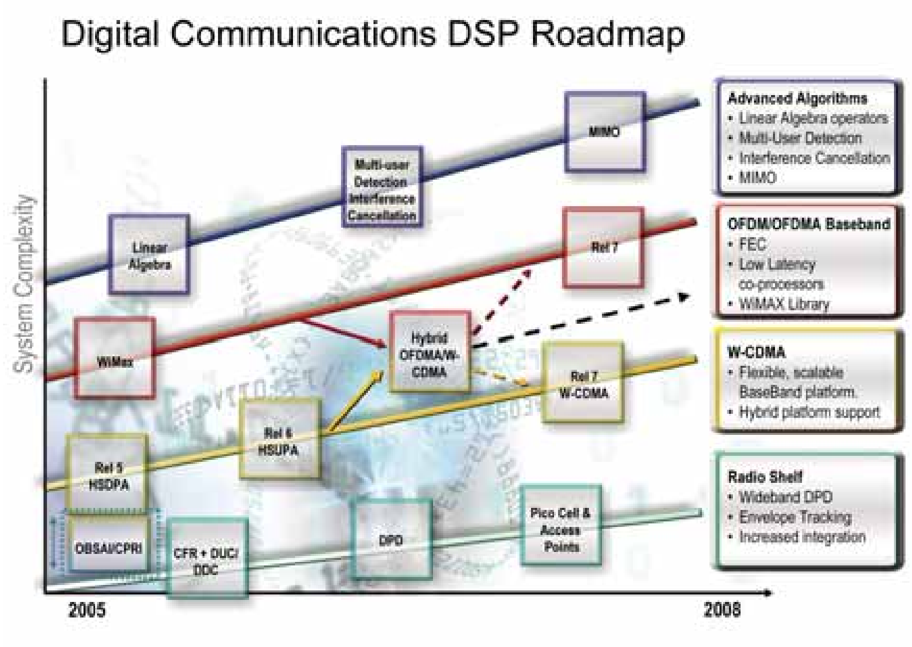


Figure 2 – Digital Communications DSP Roadmap

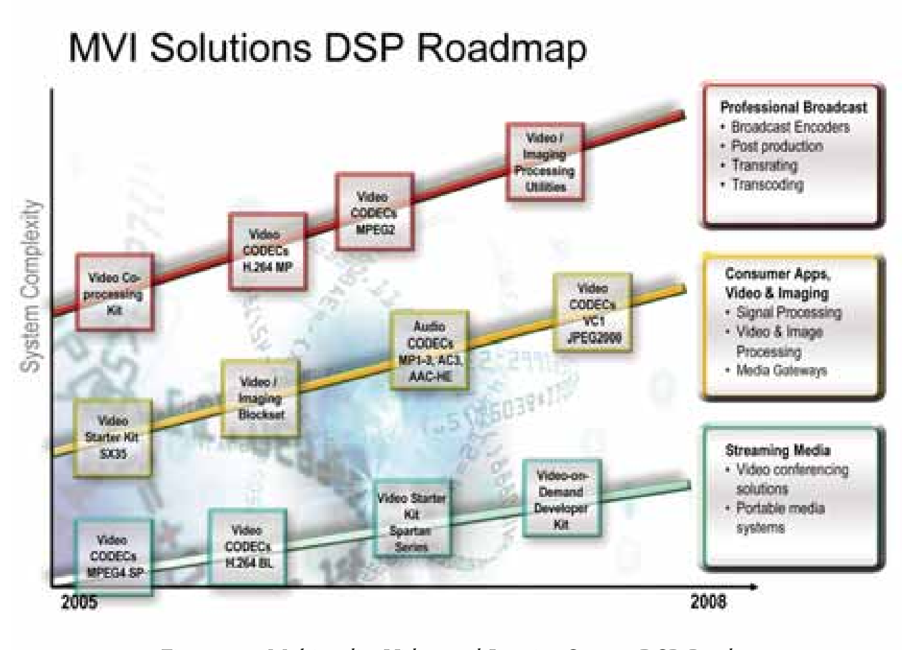


Figure 3 – Multimedia, Video, and Imaging Systems DSP Roadmap

Communication Systems; Multimedia, Video, and Imaging (MVI) Systems; and Defense Systems (represented in Figures 2, 3, and 4, respectively). Each of these roadmaps contains specialized components or solution platforms. This represents the collective expertise of developers, application engineers, and field technical experts in conjunction with invaluable input from customers.

In addition to developing building-block IP, Xilinx is moving toward sets of products intended to provide proof of concept, and in some cases, reference quality designs that can be adopted directly into customer solutions. Examples in the digital communications arena are in the 3GPP and W-CDMA standards in radio-shelf and base-band implementations. New areas of rapidly growing interest are the WiMAX standards and Picocell architectures. Similar solutions are included in each of the other market-focused roadmaps.

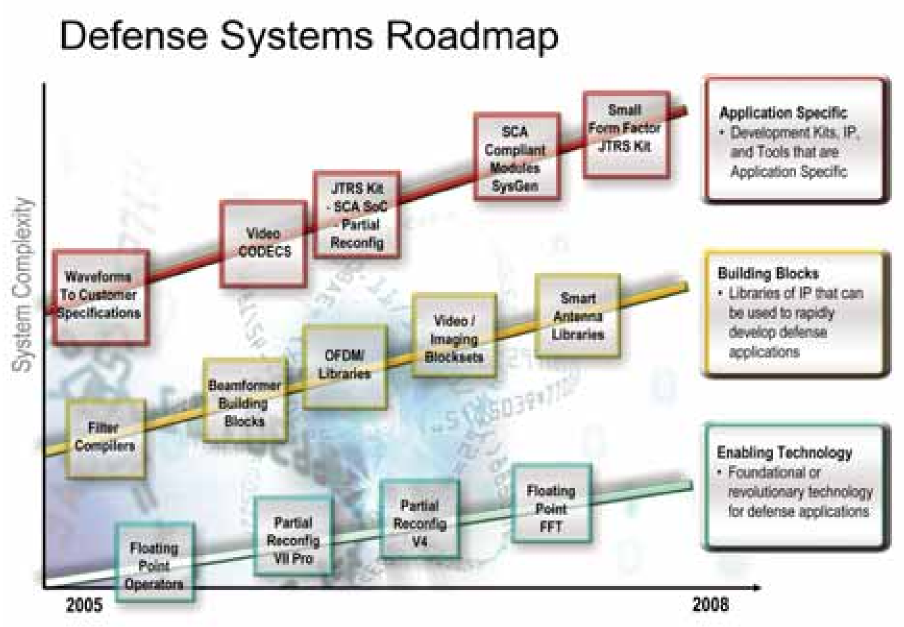

Defense Systems Roadmap
System Complexity
Waveforms To Customer Specifications
Video CODECS
JTRS Kit - SCA SoC Partial Reconfig
SCA Compliant Modules SysGen
Small Form Factor JTRS Kit
Application Specific
• Development Kits, IP, and Tools that are Application Specific
Filter Compilers
Beamformer Building Blocks
OFDM/ Libraries
Video / Imaging Blocksets
Smart Antenna Libraries
Building Blocks
• Libraries of IP that can be used to rapidly develop defense applications
Floating Point Operators
Partial Reconfig VII Pro
Partial Reconfig V4
Floating Point FFT
Enabling Technology
• Foundational or revolutionary technology for defense applications
2005
2008

Figure 4 – Defense Systems DSP Roadmap

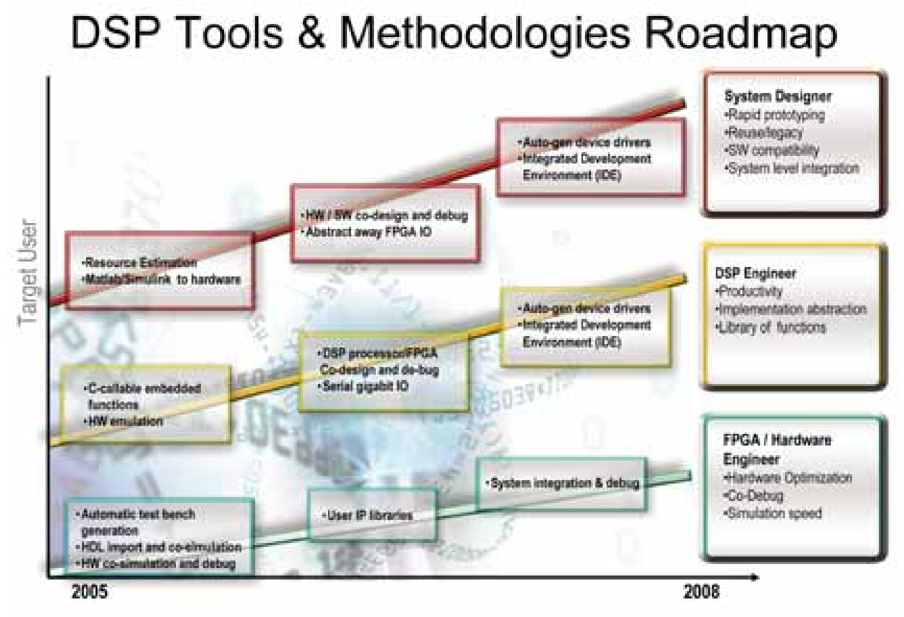

DSP Tools & Methodologies Roadmap
Target User
•Resource Estimation
•Matlab/Simulink to hardware
•HW / SW co-design and debug
•Abstract away FPGA IO
•Auto-gen device drivers
•Integrated Development Environment (IDE)
System Designer
•Rapid prototyping
•Reuse/legacy
•SW compatibility
•System level integration
•C-callable embedded functions
•HW emulation
•DSP processor/FPGA Co-design and de-bug
•Serial gigabit IO
•Auto-gen device drivers
•Integrated Development Environment (IDE)
DSP Engineer
•Productivity
•Implementation abstraction
•Library of functions
•Automatic test bench generation
•HDL import and co-simulation
•HW co-simulation and debug
•User IP libraries
•System integration & debug
FPGA / Hardware Engineer
•Hardware Optimization
•Co-Debug
•Simulation speed
2005
2008

Figure 5 – DSP Tools and Methodolgies Roadmap

Tools

The Tools and Methodologies Roadmap shown in Figure 5 illustrates our desire to address the designer community in three major tiers: traditional Xilinx hardware (FPGA) designers, DSP development engineers, and system designers. The strategy is built on the Xilinx ISE™ software tools suite, but incorporates System Generator for DSP, our embedded development tool suites, and other third-party offerings. If you haven't reviewed this area recently, you will be quite surprised to see the advances in capability and performance that have been introduced and are coming over the next few releases.

Devices

The Spartan™ and Virtex™ FPGA families have continued to evolve and include specific functions that optimize performance and power for specific application areas. The multipliers, DSP48, and embedded processors are examples of content directly aimed at the DSP field. The Virtex-4 generation identified subfamilies that allow focused concentrations of features for cost-optimized delivery. In the roadmap for future devices, you will continue to see this focus played out with additional specialized circuits and building blocks committed to silicon.

Conclusion

The DSP Roadmaps are not intended to be a one-way communication. In presenting our vision of the future, we expect to initiate and share in a dialog with others. We intend to engender discussion and commentary. This is a healthy process of discovery that ultimately leads to better products from Xilinx that help you develop and deliver better products to your customers. We look forward to this dialog and learning between Xilinx and the DSP world.

For more information about our new products and DSP Roadmaps, visit DSP Central at *www.xilinx.com/dsp*.

FPGA-Based MPEG-4 Codec

Using FPGAs to implement complex video codecs goes beyond ASIC prototyping.

by Paul Schumacher
Senior Staff Research Engineer
Xilinx, Inc.
paul.schumacher@xilinx.com

Wilson Chung
Senior Staff Video and Imaging Engineer
Xilinx, Inc.
wilson.chung@xilinx.com

Have you ever wanted to include state-of-the-art video compression in your FPGA design but found it too complex an undertaking? You no longer need to be a video expert to include video compression in your system. Newly released MPEG-4 encoder/decoder cores from Xilinx can help solve your video compression needs.

Video and multimedia systems are becoming increasingly complex, and the availability of low-cost, reliable IP cores for your system is crucial to getting your product to market. In particular, video compression algorithms and standards have become extremely complicated circuits that can take a long time to design and are quite often bottlenecks in getting a system tested and shipped. These MPEG-4 simple profile encoder/decoder cores may just do the trick for your next multimedia system.

Applications

MPEG-4 Part 2 is a recent international video coding standard in a series of such standards: H.261, MPEG-1, MPEG-2, and H.263. It was approved by ISO/IEC as International Standard 14 496-2 (MPEG-4 Part 2) in December 1999. The MPEG-4 Part 2 video codec provides an excellent basis for a number of multimedia applications. The standard provides a set of profiles and levels to allow for a plethora of different application requirements, such as frame size and use of error-resilience tools. Examples of these applications include broadcasting, video editing, teleconferencing, security/surveillance, and consumer electronics applications.

The video coding algorithm used in MPEG-4 Part 2 is an evolution from previous coding standards. The frame data is divided into 16 x 16 macroblocks containing six 8 x 8 blocks for YCbCr 4:2:0 formatted data. Motion estimation with half-pixel resolution is used to efficiently code predicted blocks from the previous frame, while the discrete cosine transform (DCT) provides the residual processing to create a more detailed view of the current frame. Simple profile provides 12 bits of resolution for DCT coefficients with 8 bits per sample for the sampled and reconstructed frame data. Coding efficiency of the MPEG-4 simple profile is better than the previous generation in MPEG-2 across a range of coding bit rates.

A typical multimedia system can use MPEG-4 as the video compression component within a larger system. An example of this is an end-to-end video conferencing system delivering compressed bitstreams between two or more participants. Designations for these sources can modify system requirements, where a key speaker or presenter for a conference may require higher resolution video as well as audio. This type of system can be expanded to video surveillance and security applications, where a display station user may decide to keep a mosaic of all video cameras or focus in on a single camera view for detailed real-time analysis. These applications require that the stream selection is performed at the receiver and is capable of handling real-time viewing specifications.

An FPGA provides an excellent programmable concurrent processing platform that allows for support of varying system requirements while meeting the needs of system throughput. The Xilinx® MPEG-4 decoder core can be built with a scalable, multi-stream interface customized for your application and system requirements, while both the MPEG-4 encoder and decoder are also capable of servicing a user-specified maximum frame size.

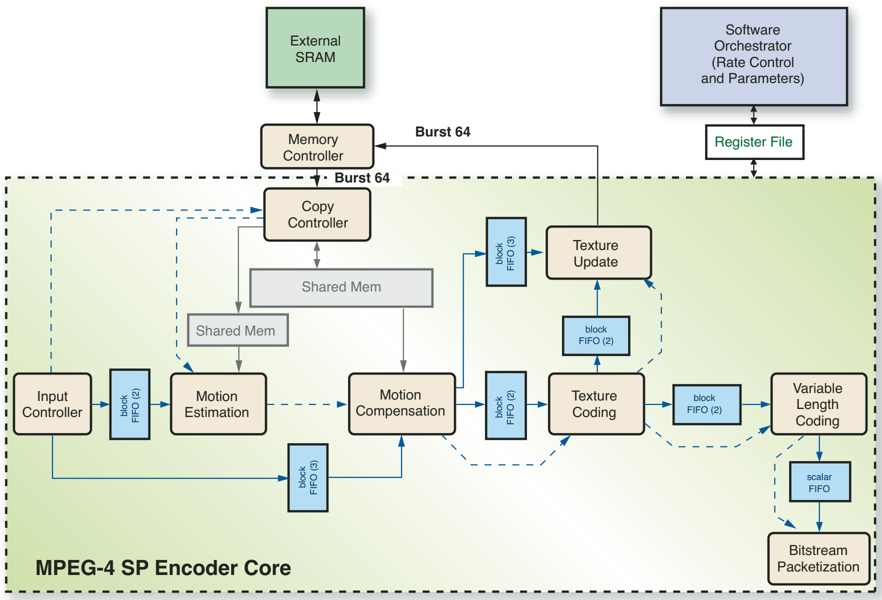


Figure 1 – Block diagram of MPEG-4 Part 2 simple profile encoder core

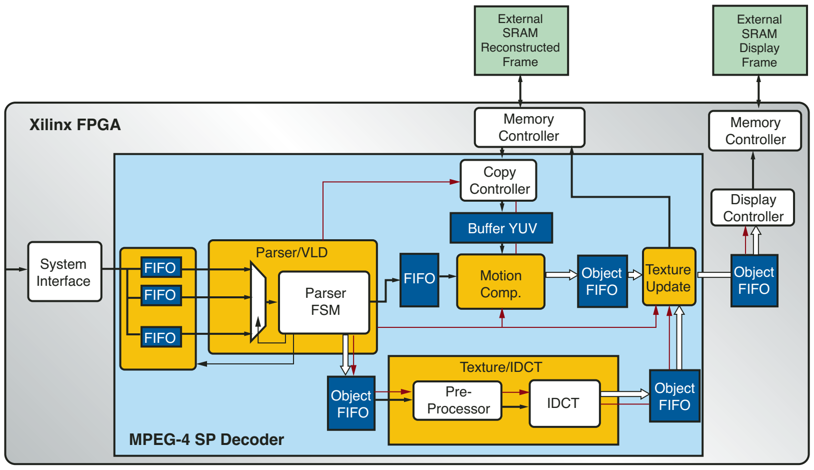


Figure 2 – Block diagram of MPEG-4 Part 2 simple profile decoder core

Architecture

Figures 1 and 2 illustrate the block diagrams for the MPEG-4 simple profile encoder and decoder cores, respectively. Hardware-based, pipelined architectures were used for these implementations, with a host interface provided on the encoder for software-controlled rate control. With an included memory controller, the raw, captured sequence for the encoder and the reconstructed frames for the decoder are stored in an off-chip memory for fast, low-latency access to the pixel data. A simple FIFO interface is provided for communicating the compressed bitstreams, with the decoder custom-built for a user-specified number of bitstreams. A system interface is also included to allow for maximum controllability and observability.

To create scalable multi-stream designs that can meet the needs of different applications, the package provided with the core contains a number of user-specified, compile-time parameters that allow you to customize the encoder and decoder. To create a resource-efficient design, you can also set the maximum supported frame width and height. The compiled design would then include enough memory and registers to support any frame dimensions less than or equal to these two parameters. Other parameters give you complete control over the scalability of the final design and craft a system built exclusively for your application.

Tables 1 and 2 list the FPGA resources for the encoder and decoder cores based on different parameter settings for maximum supported frame size, as well as the number of input bitstreams for the decoder. All of the encoder designs in Table 1 utilize 16 embedded XtremeDSP™ slices, while the decoders in Table 2 utilize 32 embedded XtremeDSP slices. These designs target Virtex™-4 parts, which contain a number of 18 Kb block SelectRAM™ memories as well as embedded XtremeDSP slices. Other compatible FPGA families include Virtex-II, Virtex-II Pro, and Spartan™-3 devices.

Note that the decoder design can automatically instantiate the number of input FIFOs and supporting multiplexing/demultiplexing circuitry based on the number of bitstreams to support. The MPEG-4 encoder is capable of a throughput of approximately 48,000 macroblocks per second, providing enough horsepower to exceed the throughput specifications of simple profile at level 5. Meanwhile, the MPEG-4 decoder design can sustain a throughput of approximately 168,000 macroblocks per second, providing adequate throughput to decode two streams of progressive SDTV (720 x 480 at 60 fps) or 14 streams of CIF resolution. This decoder throughput is more than four times the required throughput for simple profile at level 5.

Conclusion

MPEG-4 simple profile encoder and decoder cores have been designed with unique, scalable, multi-stream capabilities to suit your specific system needs. A number of different applications can take advantage of these cores in a multimedia system, including video conferencing, security, and surveillance, as well as any exciting new consumer application that you have yet to show the world.

High-throughput, pipelined architectures were used for these video designs with enough customizable parameters to create a resource-efficient design exclusive to your application. For more information, visit *www.xilinx.com/dsp.*

The authors would like to acknowledge contributions from Robert Turney, Nick Fedele, Adrian Chirila-Rus, Mark Paluszkiewicz, and Kees Vissers at Xilinx, as well as members at IMEC.

Parameters / Resources: Frame Size	Block RAMs	FPGA Slices	Minimum Clock Rate (MHz)
QCIF @ 15 fps	16	8,051	3.2
CIF @ 30 fps	21	8,309	25.6
4CIF @ 30 fps	30	9,000	100.7

Table 1 – Scalable MPEG-4 Part 2 simple profile encoder core resources

Parameters / Resources: Frame Size	Streams	Block RAMs	FPGA Slices	Minimum Clock Rate (MHz)
QCIF @ 15 fps	1	10	4,332	0.8
	8	17	5,014	6.6
CIF @ 30 fps	1	16	4,558	6.6
	8	23	5,305	52.8
4CIF @ 30 fps	1	26	5,004	26.4
	8	33	5,764	211.2 *

* *Note: Eight streams of 4CIF resolution currently require two instantiations of the decoders.*

Table 2 – Scalable, multi-stream MPEG-4 Part 2 simple profile decoder core resources

Rapid Development of Video/Imaging Systems

Build real-time video and imaging applications quickly and easily with the Xilinx Video Starter Kit.

by Hong-Swee Lim
Senior Manager, DSP Product and Solutions Marketing
Xilinx, Inc.
hong-swee.lim@xilinx.com

Advances in media encoding schemes are enabling a broad array of applications, including digital video recorders (DVRs), network surveillance cameras, medical imaging, digital broadcasting, and streaming set-top boxes. The promise of streaming media presents a series of implementation challenges, especially when processing complex compression algorithms such as MPEG-4 and MPEG-compressed video transcoding. Given the high computational horsepower required for encoding or decoding such complex algorithms, achieving optimal balance of power, performance, and cost is a significant challenge for streaming media devices.

By using FPGAs, you can differentiate your standard-compliant systems from your competitor's products and achieve the optimal balance for your application. With the MPEG-4 compression scheme, for example, it is possible to offload the IDCT (inverse discrete cosine transform) portion of the algorithm from an MPEG processor to an FPGA to increase the processing bandwidth. IDCT (and DCT at the encoder) can be implemented extremely efficiently using FPGAs, and optimized IP cores are readily available to include in MPEG-based designs.

By integrating various IP cores together with the IDCT core, you can develop a low-cost, single-chip solution that increases processing bandwidth and gives higher quality images than your competitor's ASSP-based solution.

To help you accelerate your system design, Xilinx offers the Video Starter Kit (VSK) 4VSX35. The VSK is an all-digital platform for real-time video/image acquisition, processing, and display. It integrates the power of hardware-accelerated processing as well as an embedded PowerPC™ core for the transmission of high-resolution digital video over lower bandwidths, or for processing network protocol stack and control functions.

Xilinx Video Starter Kit 4VSX35

The Xilinx® VSK 4VSX35 allows you to jump-start your high-performance audio, video, and imaging processing designs. At the heart of the VSK are two highly programmable Xilinx FPGAs (XC2VP4 and XC4VSX35), video encoder, video decoder, AC97 CODEC, and a wide range of video interfaces.

Figure 1 illustrates the VSK's primary components, peripherals, and available I/O.

The VSK comprises three major hardware components: a Xilinx ML402-SX35 board; 752 x 480-pixel RGB progressive scan CMOS image-sensor camera with a frame rate as high as 60 frames per second (fps); and video I/O daughtercard (VIODC). The VIODC is connected to the ML402-SX35 board through the Xilinx Generic Interface (XGI), while the CMOS camera is connected to the VIODC through the serial LVDS interface.

The video encoder is a high-speed, video digital-to-analog converter. It has three separate 10-bit-wide input ports that accept data in high- or standard-definition video formats. It also controls the insertion of appropriate synchronization signals; external horizontal, vertical, and blanking signals; or EAV/SAV timing codes for all standards.

The video decoder is a high-quality, single-chip, multi-format video decoder that automatically detects and converts PAL, NTSC, and SECAM standards in the form of composite, S-Video, and component video into a digital ITU-R BT.656 format. The advanced and highly flexible digital output interface enables performance video decoding and conversion in line-locked clock-based systems. This makes the VSK ideally suited for a broad range of applications with diverse video characteristics, including broadcast sources, security and surveillance cameras, and professional video systems. Figure 2 shows a block diagram of the Video Starter Kit.

With the video encoder, video decoder, DVI receiver, DVI transmitter, and camera supporting a two-wire serial I^{2}C-compatible interface, all of these devices can be controlled through an I^{2}C master core located either in the XC4VSX35 or XC2VP4 device.

The flexibility of the VSK architecture makes it suitable as a development platform for a variety of multimedia, video, and imaging applications, which include:

- Medical imaging
- Home media gateways
- Multi-channel digital video recorders
- IP TV set-top boxes
- Video-on-demand servers
- Digital TV
- Digital camera and camcorders
- A/V broadcasts
- Network surveillance cameras

System Generator for DSP v8.1

Converting image processing algorithms to FPGA implementations can be challenging, as the algorithms may be proven in software but not directly linked to the actual implementation. Additionally, it can be difficult to subjectively verify the implementation.

Xilinx System Generator for DSP allows for high-level mathematical verification and converts the heart of the algorithm into ready-to-use HDL, which bridges the gap from the algorithm developer to the FPGA engineer.

Using System Generator and the VSK to develop and implement image-processing algorithms allows for a thoroughly verified and easily executed design. The high-level block diagram allows for easy communication between team members, resulting in less time spent crossing skill boundaries when determining implementation trade-offs.

To accelerate video/imaging system development, Xilinx has developed new System Generator blocks specifically for the VSK, including:

- VIODC interface block
- Multi-port DDR memory controller block
- System-level blocks

With these pre-tested blocks, you can easily build your video/imaging system by just dragging and dropping the blocks within System Generator to construct your system, saving precious time from coding these essential interfacing blocks in HDL.

To be able to handle the enormous video data stream

Figure 1 - Video Starter Kit 4VSX35

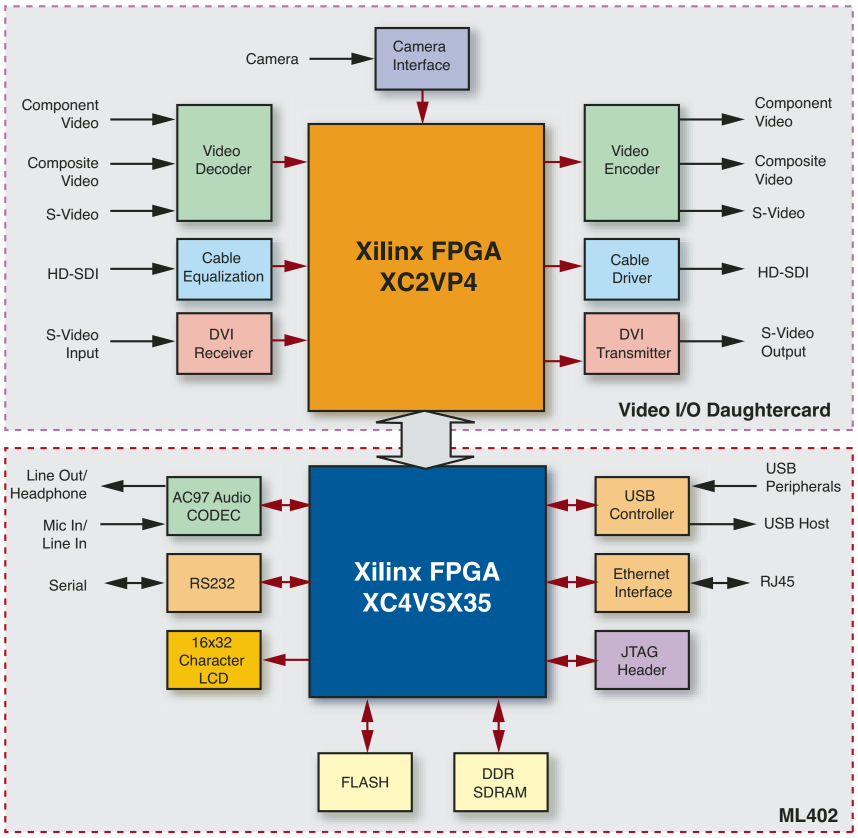


Figure 2 - Block diagram of Video Starter Kit 4VSX35

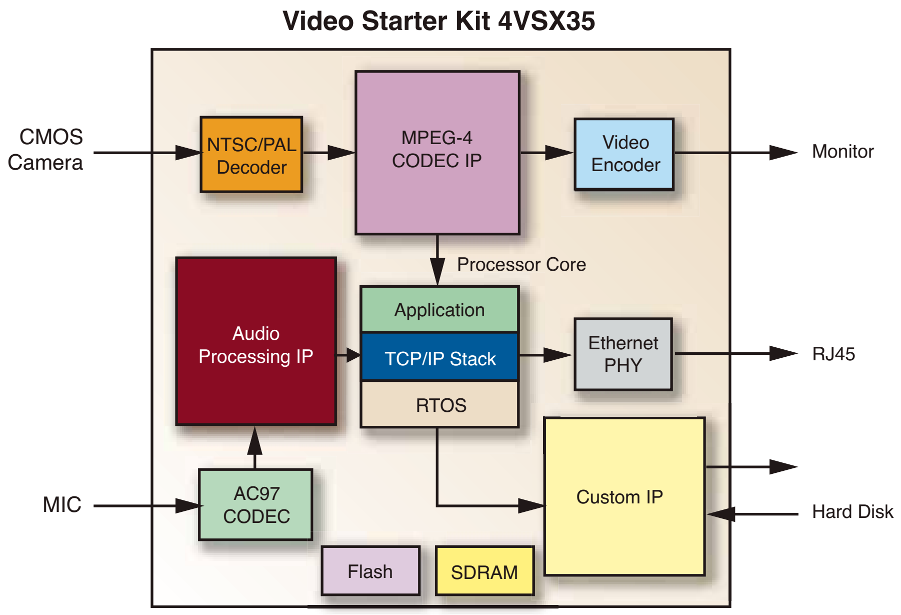


Figure 3 - Network surveillance camera

from the VSK to the PC, another innovative high-speed hardware co-simulation through an Ethernet interface was introduced in System Generator for DSP 8.1. This interface allows high throughput with low latency, which proved to be extremely useful when building video/imaging systems in the System Generator environment.

Network Surveillance Camera Application

FPGAs have historically been found in high-end professional broadcast systems and medical imaging equipment. Today FPGAs are also finding their way into high-volume products such as digital video recorders and network surveillance cameras because of their flexibility in handling a broad range of media formats such as MPEG-2, MPEG-4, H.264, and Windows Media. Their extremely high-performance DSP horsepower also makes FPGAs suitable for other challenging video and audio tasks.

Typically, a network surveillance camera product comprises three parts: a camera to convert the real-world image into a video stream; a video decoder for streams compressed into H.264, MPEG-2, or another format; and a video/image processor for de-interlacing, scaling, and noise reduction before packeting the digitized video for transmission over the Internet.

FPGAs can have many areas of responsibility within surveillance cameras, as shown in Figure 3. Bridging between standard chipsets as “glue logic” has always been a strong application of FPGAs, but many more image-processing tasks (such as color-space conversion), IDE (Integrated Drive Electronics) interface, and support for network interfaces (such as IEEE 1394) are now also commonly implemented in low-cost programmable devices.

With high-performance DSP capability inside a network surveillance camera, you can digitize and encode the video stream to be sent over any computer network. You can use a standard Web browser to view live, full-motion video from anywhere on a computer network, including over the Internet. Installation is simplified by using existing LAN wiring or wireless LAN. Features such as intelligent video, e-mail notification, FTP uploads, and local hard-disk storage provide enhanced differentiation and superior capability over analog systems.

The hard-processor core is an IBM PowerPC 405 immersed in a Xilinx Virtex™-II Pro™ FPGA, delivering 600 DMIPS at 400 MHz running MontaVista Linux or Wind River Systems’s VxWorks real-time operating system (RTOS), as well as a network protocol stack to implement these features.

Xilinx also offers the MicroBlaze™ 32-bit RISC processor core, delivering up to 138 DMIPS at 150 MHz and 166 DMIPS at 180 MHz when used in the Virtex-II Pro and Virtex-4 devices, respectively.

Conclusion

Bandwidth is precious; to make the most of it, compression schemes have steadily improved – and new algorithms push the envelope even further. As such, system-processing rates have increased over time, and real-time image processing is an ideal way to meet these requirements while removing memory overhead.

At the same time, Moore’s Law has resulted in low-cost programmable logic devices, such as the new FPGAs, that provide the same functionality and performance previously found only in expensive professional broadcast products.

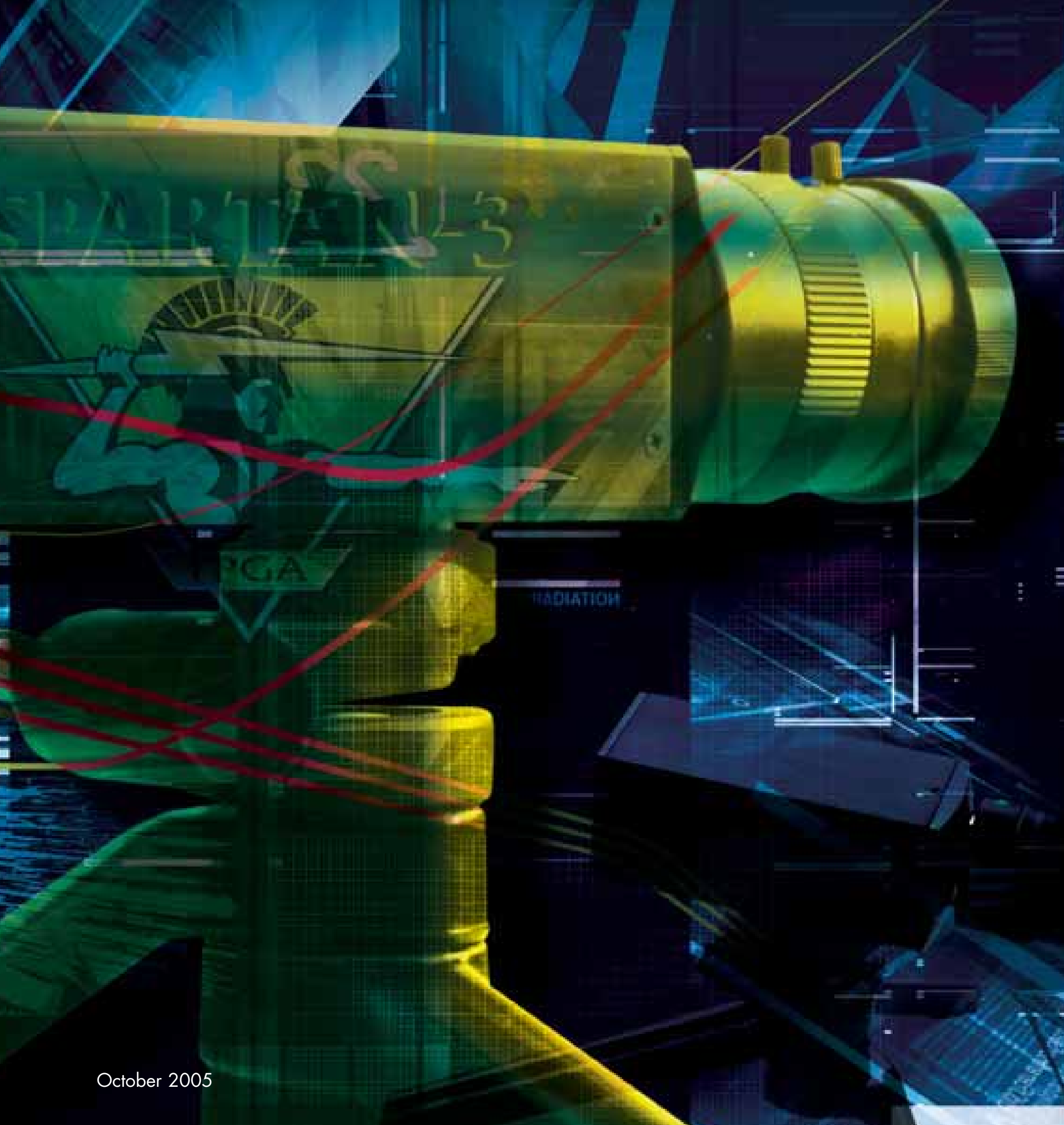
FPGAs provide both professional and consumer digital broadcast OEMs with real-time image processing capabilities that address the system requirements of new and emerging video applications. Compared to other technologies, FPGAs offer an unrivalled flexibility that enables you to get your products to market quickly. Remote field upgradeability means that systems can be shipped now and features, upgrades, or design fixes added later.

The VSK has been architected to reduce implementation risks, time to market, and development costs. By providing hardware and MPEG-4 IP in a pre-tested and integrated platform, you can concentrate on implementing the application-specific video and imaging functionality that is most relevant to your particular product.

For more information, visit *www.xilinx.com/products/design_resources/dsp_central/grouping/index.htm.*

Encoding High-Resolution Ogg/Theora Video with Reconfigurable FPGAs

Once the traditional application area of custom ASICs, modern FPGAs can now handle high-performance video encoding.



by Andrey Filippov
President
Elphel, Inc.
andrey@elphel.com

Much of the Spring 2003 issue of the *Xcell Journal* in which my article about Spartan™-IIE-based Elphel Model 313 cameras appeared ("How to Use Free Software in FPGA Embedded Designs") was dedicated to the Xilinx® Spartan-3 FPGA. I immediately started to think about using these devices in our new generation of Elphel network cameras, but it wasn't until last year that I was finally able to start working with them.

One of the factors that slowed my company's adoption of this new technology was the fact that at first I could not find appropriate software that could handle the devices selected, as it is essential that our end users can modify our products without expensive software development tools. When I visited the Xilinx website in Summer 2004 and found that the current version of the free downloadable WebPACK™ software could handle the XC3S1000 – the largest device available in a small FT256 package – I knew it was the right time to switch to the Spartan-3 device.

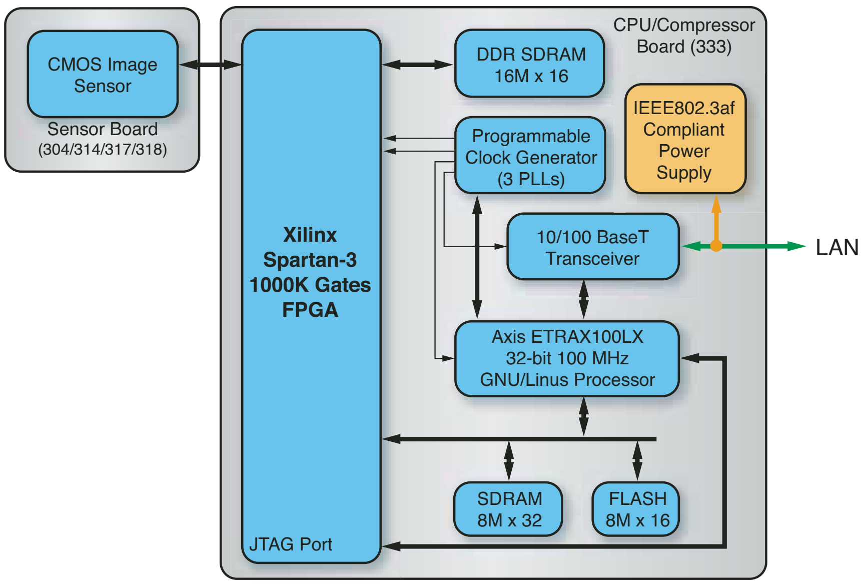

CMOS Image Sensor
Sensor Board (304/314/317/318)
CPU/Compressor Board (333)
Xilinx Spartan-3 1000K Gates FPGA
JTAG Port
DDR SDRAM 16M x 16
Programmable Clock Generator (3 PLLs)
IEEE802.3af Compliant Power Supply
10/100 BaseT Transceiver
LAN
Axis ETRAX100LX 32-bit 100 MHz GNU/Linus Processor
SDRAM 8M x 32
FLASH 8M x 16

Figure 1 – Camera system block diagram

The Camera Hardware

The new Model 333 camera (Figure 1) uses the same Linux-optimized CPU (ETRAX100LX by Axis Communications) as the earlier Model 313, but with increased system memory – 32 MB of SDRAM and 16 MB of Flash. The second major upgrade is the use of 32 MB of DDR SDRAM as a dedicated frame buffer that works in tandem with the FPGA, supplementing its processing power with high capacity and I/O bandwidth.

The Spartan-3 DDR I/O functionality made it possible to increase the memory bandwidth without increasing board size – the complete system still fits on a 1.5 x 3.5-inch four-layer board (see Figure 2). The actual board area is even smaller, as the new one is designed to fit the sealed RJ45 connectors for outdoor applications.

For the camera circuit design, the goals include combining high computational performance with small size (that also simplifies preserving high-speed signal integrity on the PCB) and providing the flexibility for the reconfigurable FPGA on the system level. For the latter, I decided to split the camera circuitry into two boards: one main board and a second containing just a sensor with minimal related components. On the main board the FPGA I/O pins go directly to the inter-board connector, so it is possible to change the pin functions (including polarity) to match the particular sensor boards. A similar solution allowed the earlier Model 313 camera to support different types of sensors (most became available after the board design). It even works in our 11-megapixel Model 323 cameras without any PCB modifications.

Selecting the Video Encoding Technique

After the prototype camera was ready, it took just a couple of weeks to modify the code developed for the Spartan-IIE-based camera and to implement motion JPEG compression. Half of that time was spent trying to figure out how to configure the new FPGA with the generated bitstream. In the camera, JTAG pins of the device are connected directly to the processor I/O pins, so I could not use the software that comes with Xilinx configuration hardware. The JTAG instruction register is six bits wide, not five as it was in the Spartan-IIE devices with which I was familiar. After some trial and error, I figured that out and found that the same code could run at 125 MHz (instead of 90 MHz in the previous model) and used just 36% (not 98% as before) of available slices – plenty of room for more challenging tasks.

Of course, I had some challenging tasks in mind, as motion JPEG is not a really good option for high-resolution/high-frame-rate cameras because the amount of data to be transferred or stored is quite huge. It is a waste of network bandwidth or hard disk space when recording such video streams, as fixed-view cameras in most cases have very little difference between consecutive frames. Something like MPEG-2 could make a difference; that was the standard I was planning to implement in the camera.

But as soon as I got some books on MPEG-2 and started combing through online resources, I found another fundamental difference between MPEG and JPEG – not just that it can use the similarity between consecutive frames. Contrary to JPEG, MPEG-2 requires you to pay licensing fees for using the encoders based on this standard. The fee is small compared to the cost of the hardware, but it still could be a hassle and does not provide freedom for implementation.

It did not take long to find a perfect alternative – Theora, based on the VP3

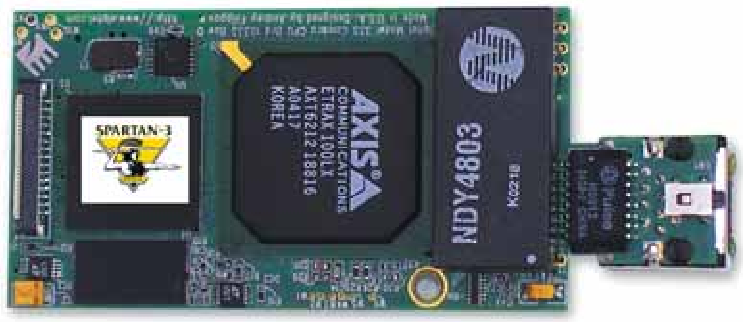


Figure 2 – Camera system board

codec developed by On2 Technologies (*www.on2.com*) and released as open-source software for royalty-free use and modifications (see *www.theora.org/svn.html*).

Theora is an advanced video codec that competes with MPEG-4 and other similar low-bit-rate video compression schemes. It is now supported by the Xiph.org Foundation along with Ogg, the transport layer used with Theora to deliver the video content. The bitstream format is stable enough and supported by multiple players running on different operating systems. Like JPEG and MPEG, it uses a two-dimensional 8 x 8 DCT.

FPGA Implementation

The code for the Elphel Model 333 camera FPGA is written in Verilog HDL (Figure 3). It is designed around the 8-channel SDRAM controller that uses the Spartan-3 DDR capabilities. The structure of the memory accesses and specially organized data mapping both serve the same goal: optimizing memory bandwidth that otherwise would be a system bottleneck.

The rest of the code that currently uses two-thirds of the general FPGA resources (slices) and 20 of 24 block RAM modules includes video compression modules, a sensor, and system interfaces.

A detailed description of the camera code is available, together with the source code, at Sourceforge (*https://sourceforge.net/projects/elphel*).

Conclusion

High-performance reconfigurable FPGAs made it possible to build a fast high-resolution low-bit-rate network camera capable of running 30 fps at a resolution of 1280 x 1024 pixels (12 fps at a resolution of 2048 x 1536). Many of the new features of the Spartan-3 devices proved to be very useful in this design: embedded multipliers for DSP functions, advanced digital clock management, DDR I/O functions, an increased number of global clock networks for the DDR SDRAM controller, and large block RAM modules for the various tables and buffers in the camera.

The free video encoder (Theora) and completely open implementation of the camera (all software and Verilog code is provided under the GNU General Public License) makes the second most important function of Elphel products possible. You can use these cameras not only as finished products but also as universal development platforms – demonstrating the power and flexibility of the Spartan-3 family. It is possible to add your own code, rerun the tools (both for the FPGA code and the C-language camera software), and immediately try the new camera with advanced image processing implemented.

For more information, visit *www.elphel.com*, *https://sourceforge.net/projects/elphel/*, and *www.theora.org*.

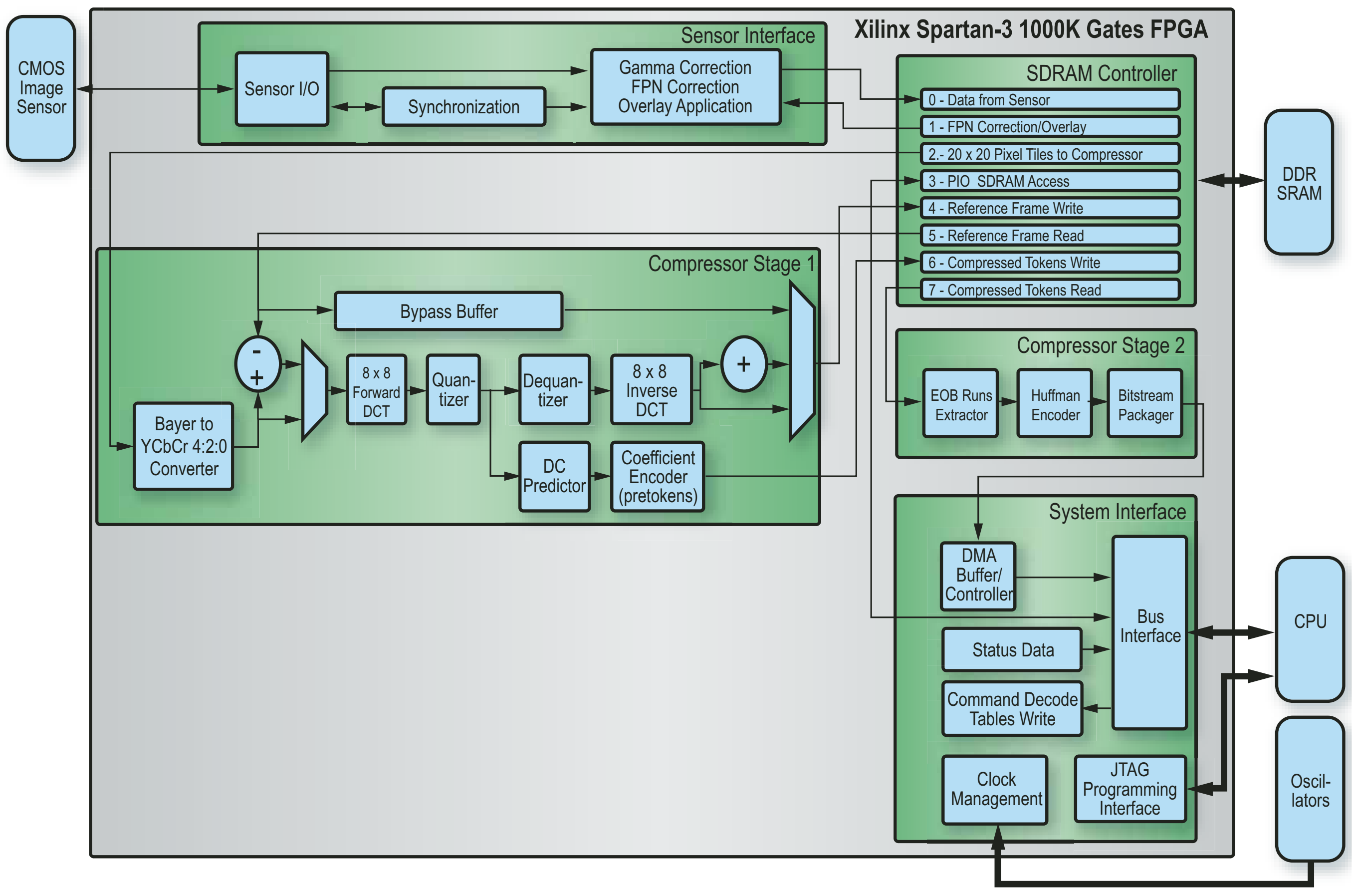
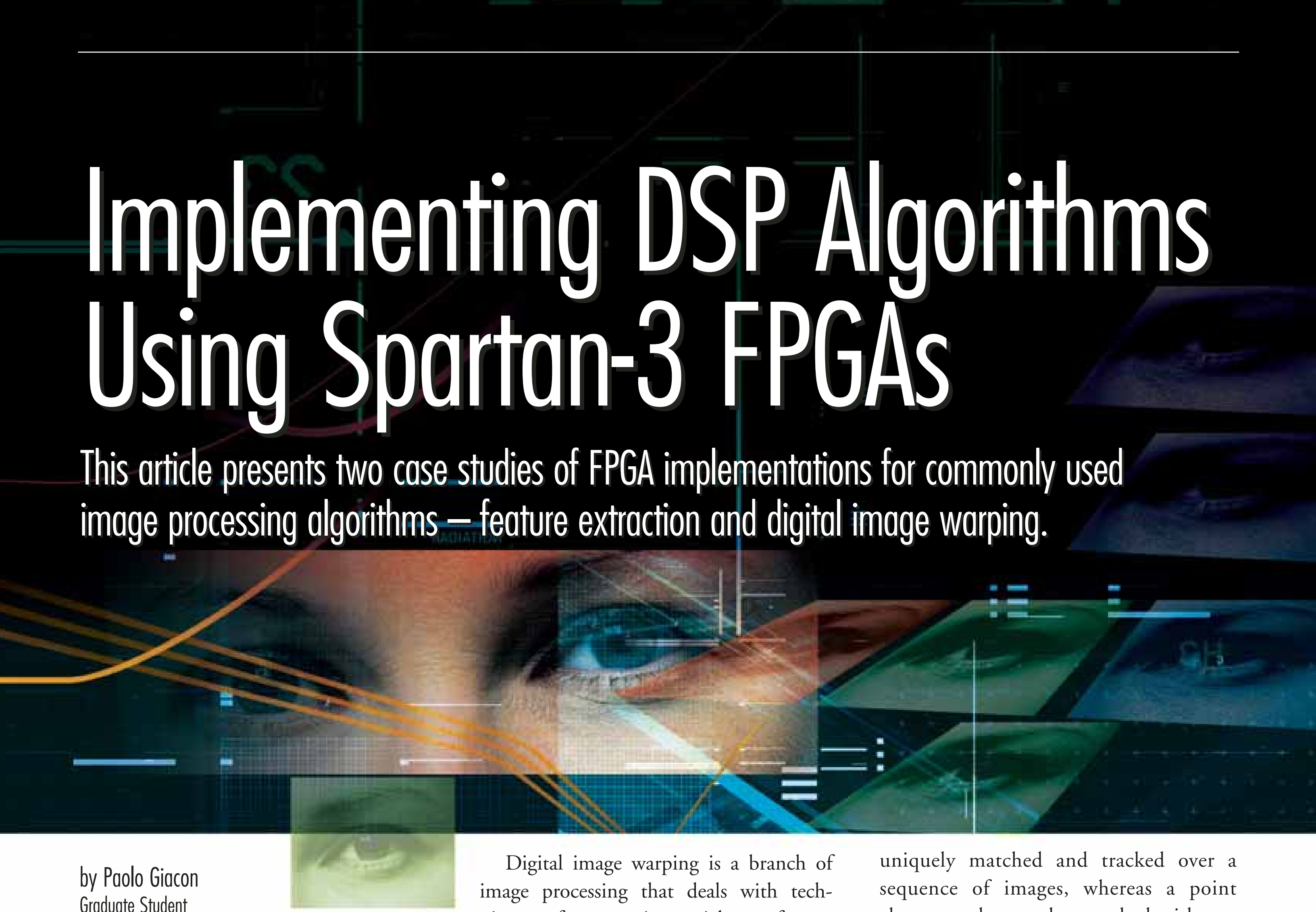


Figure 3 – Block diagram of the FPGA code



Implementing DSP Algorithms Using Spartan-3 FPGAs

This article presents two case studies of FPGA implementations for commonly used image processing algorithms – feature extraction and digital image warping.

by Paolo Giacon
Graduate Student
Università di Verona, Italy
paolo.giacon@students.univr.it

Saul Saggin
Undergraduate Student
Università di Verona, Italy
saul.saggin@students.univr.it

Giovanni Tommasi
Undergraduate Student
Università di Verona, Italy
giovanni.tommasi@students.univr.it

Matteo Busti
Graduate Student
Università di Verona, Italy
matteo.busti@students.univr.it

Computer vision is a branch of artificial intelligence that focuses on equipping computers with the functions typical of human vision. In this discipline, feature tracking is one of the most important pre-processing tasks for several applications, including structure from motion, image registration, and camera motion retrieval. The feature extraction phase is critical because of its computationally intensive nature.

Digital image warping is a branch of image processing that deals with techniques of geometric spatial transformations. Warping images is an important stage in many applications of image analysis, as well as some common applications of computer vision, such as view synthesis, image mosaicing, and video stabilization in a real-time system.

In this article, we'll present an FPGA implementation of these algorithms.

Feature Extraction Theory

In many computer vision tasks we are interested in finding significant feature points – or more exactly, the corners. These points are important because if we measure the displacement between features in a sequence of images seen by the camera, we can recover information both on the structure of the environment and on the motion of the viewer.

Figure 1 shows a set of feature points extracted from an image captured by a camera. Corner points usually show a significant change of the gradient values along the two directions (x and y). These points are of interest because they can be uniquely matched and tracked over a sequence of images, whereas a point along an edge can be matched with any number of other points on the edge in a second image.

The Feature Extraction Algorithm

The algorithm employed to select good features is inspired by Tomasi and Kanade's method, with the Benedetti and Perona approximation, considering the eigenvalues α and β of the image gradient covariance matrix. The gradient covariance matrix is given by:

$$H = \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_x^2 \end{pmatrix}$$

where I_x and I_y denote the image gradients in the x and y directions.

Hence we can classify the structure around each pixel observing the eigenvalues of H:

No structure	: $\alpha \approx \beta \approx 0$
Edge	: $\alpha \approx 0, \beta >> 0$
Corner	: $\alpha >> 0, \beta >> 0$

Figure 1 – Feature points extracted from an image captured by a camera

Using the Benedetti and Perona approximation, we can choose the corners without computing the eigenvalues.

We have realized an algorithm that, compared to the original method, doesn't require any floating-point operations. Although this algorithm can be implemented either in hardware or software, by implementing it in FPGA technology we can achieve real-time performance.

Input:

- 8-bit gray-level image of known size (up to 512 x 512 pixels)
- The expected number of feature points (wf)

Output:

- List of selected features (FL). The type of the output is a 3 x N matrix whose:
  - First row contains the degrees of confidence for each feature in the list
  - Second row contains the x-coordinates of the feature points
  - Third row contains the y-coordinates of the feature points

Semantic of the Algorithm

In order to determine if a pixel (i, j) is a feature point (corner), we followed Tomasi and Kanade's method.

First, we calculate the gradient of the image. Hence the 2 x 2 symmetric matrix G = [a b; b c] is computed, whose entries derive from the gradient values in a patch around the pixel (i, j).

If the minimum eigenvalue of G is greater than a threshold, then the pixel (i, j) is a corner point. The minimum eigenvalue is computed using an approximation to avoid the square root operation that is expensive for hardware implementations.

The corner detection algorithm could be summarized as follows:

The image gradient is computed by mean of convolution of the input image with a predefined mask. The size and the values of this mask depend on the image resolution. A typical size of the mask is 7 x 7.

- For each pixel (i, j) loop:

$$a_{i,j} = \sum_{k}^{N} (I_x^k)^2$$

$$b_{i,j} = \sum_{k}^{N} I_x^k I_y^k$$

$$c_{i,j} = \sum_{k}^{N} (I_y^k)^2$$

where N is the number of pixels in the patch and I_x^k and I_y^k are the components of the gradient at pixel k inside the patch.

- $P_{i,j} = (a - t)(c - t) - b^2$

  where t is a fixed integer parameter.

- If $(P_{i,j} > 0)$ and $(a_{i,j} > t)$, then we retain pixels (i,j)
- Discard any pixel that is not a local maximum of $P_{i,j}$
- End loop
- Sort, in decreasing order, the feature list *FL* based on the degree of confidence values and take only the first *wf* items.

Implementation

With its high-speed embedded multipliers, the Xilinx® Spartan™-3 architecture meets the cost/performance characteristics required by many computer vision systems that could take advantage of this algorithm.

The implementation is divided into four fundamental tasks:

1. Data acquisition. Take in two gradient values along the x and y axis and compute for each pixel three coefficients used by the characteristic polynomial. To store and read the gradient values, we use a buffer (implemented using a Spartan-3 block RAM).
2. Calculation of the characteristic polynomial value. This value is important to sort the features related to the specific pixel. We implemented the multiplications used for the characteristic polynomial calculus employing the embedded multipliers on Spartan-3 devices.
3. Feature sorting. We store computed feature values in block RAM and sort them step by step by using successive comparisons.
4. Enforce minimum distance. This is done to keep a minimum distance between features; otherwise we get clusters of features heaped around the most important ones. This is implemented using block RAMs, building a non-detect area around each most important feature where other features will not be selected.

Spartan-3 Theoretical Performance

The algorithm is developed for gray-level images at different resolutions, up to 512 x 512 at 100 frames per second.

The resources estimated by Xilinx System Generator are:

- 1,576 slices
- 15 block RAMs
- 224 LUTs
- 11 embedded multipliers

The embedded multipliers and extensive memory resources of the Spartan-3 fabric allow for an efficient logic implementation.

Applications of Feature Extraction

Feature extraction is used in the front end for any system employed to solve practical control problems, such as autonomous navigation and systems that could rely on vision to make decisions and provide control. Typical applications include active video surveillance, robotic arms motion, measurement of points and distances, and autonomous guided vehicles.

Image Warping Theory

Digital image warping deals with techniques of geometric spatial transformations.

The pixels in an image are spatially represented by a couple of Cartesian coordinates (x, y). To apply a geometric spatial transformation to the image, it is convenient to switch to homogeneous coordinates, which allow us to express the transformation by a single matrix operation. Usually this is done by adding a third coordinate with value 1 (x, y, 1).

In general, such transformation is represented by a non-singular 3 x 3 matrix H and applied through a matrix-vector multiplication to the pixel homogeneous coordinates:

$$\begin{bmatrix} H_{1,1} & H_{1,2} & H_{1,3} \\ H_{2,1} & H_{2,2} & H_{2,3} \\ H_{3,1} & H_{3,2} & H_{3,3} \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} H_{1,1}x + H_{1,2}y + H_{1,3} \\ H_{2,1}x + H_{2,2}y + H_{2,3} \\ H_{3,1}x + H_{3,2}y + H_{3,3} \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} x'/w' \\ y'/w' \\ 1 \end{bmatrix} \Rightarrow (x'/w', y'/w') \quad (1)$$

The matrix H, called homography or collineation, is defined up to a scale factor (it has 8 degrees of freedom). The transformation is linear in projective (or homogeneous) coordinates, but non-linear in Cartesian coordinates.

The formula implies that to obtain Cartesian coordinates of the resulting pixel we have to perform a division, an operation quite onerous in terms of time and area consumption on an FPGA. For this reason, we considered a class of spatial transformations called "affine transformations" that is a particular specialization of homography. This allows us to avoid the division and obtain good observational results:

$$\begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ 0 & 0 & 1 \end{bmatrix} \bullet \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} A_{1,1}x + A_{1,2}y + A_{1,3} \\ A_{2,1}x + A_{2,2}y + A_{2,3} \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \Rightarrow (x', y') \quad (2)$$

Affine transformations include several planar transformation classes as rotation, translation, scaling, and all possible combinations of these. We can summarize the affine transformation as every planar transformation where the parallelism is preserved. Six parameters are required to define an affine transformation.

Image Warping Algorithms

There are two common ways to warp an image:

- Forward mapping
- Backward mapping

Using forward mapping, the source image is scanned line by line and the pixels are copied to the resulting image, in the position given by the result of the linear system shown in equation (2). This technique is subject to several problems, the most important being the presence of holes in the final image in the case of significant modification of the image (such as rotation or a scaling by a factor greater than 1) (Figure 2).

The backward mapping approach gives better results. Using the inverse transformation A^{-1}, we scan the final image pixel by pixel and transform the coordinates. The result is a pair of non-integer coordinates in the source image. Using a bilinear interpolation of the four pixel values identified in the source image, we can find a value for the final image pixel (see Figure 3).

This technique avoids the problem of holes in the final image, so we adopted it as our solution for the hardware implementation.

Implementation

Software implementations of this algorithm are well-known and widely used in applications where a personal computer or workstation is required. A hardware implementation requires further work to achieve efficiency constraints on an FPGA.

Essentially, the process can be divided in two parts: transformation and interpolation. We implemented the first as a matrix-vector multiplication (2), with four multipliers and four adders. The second is an approximation of the real result of the interpolation: we weighted the four pixel values approximating the results of the transformation with two bits after the binary point. Instead of performing the calculations given by the formula, we used a LUT to obtain the pixel final value, since we divided possible results of the interpolation into a set of discrete values.

Spartan-3 Theoretical Performance

We designed the algorithm using System Generator for DSP, targeting a Spartan-3 device. We generated the HDL code and synthesized it with ISE™ design software, obtaining a resource utilization of:

- 744 slices (1,107 LUTs)
- 164 SRL16
- 4 embedded multipliers

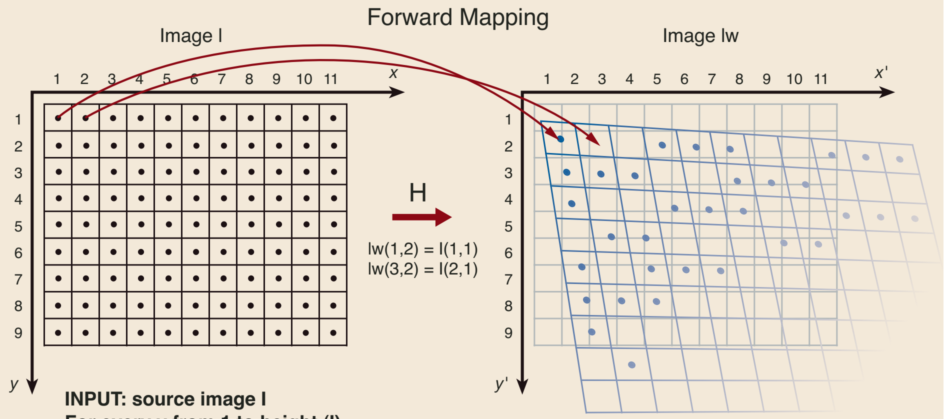
The design can process up to 46 fps (frames per second) with 512 x 512 images. Theoretical results show a boundary of 360+ fps in a Spartan-3-based system.

Applications of Image Warping

Image warping is typically used in many common computer vision applications, such as view synthesis, video stabilization, and image mosaicing.

Image mosaicing deals with the composition of sequence (or collection) of images after aligning all of them respective to a common reference frame. These geometrical transformations can be seen as simple relations between coordinate systems.

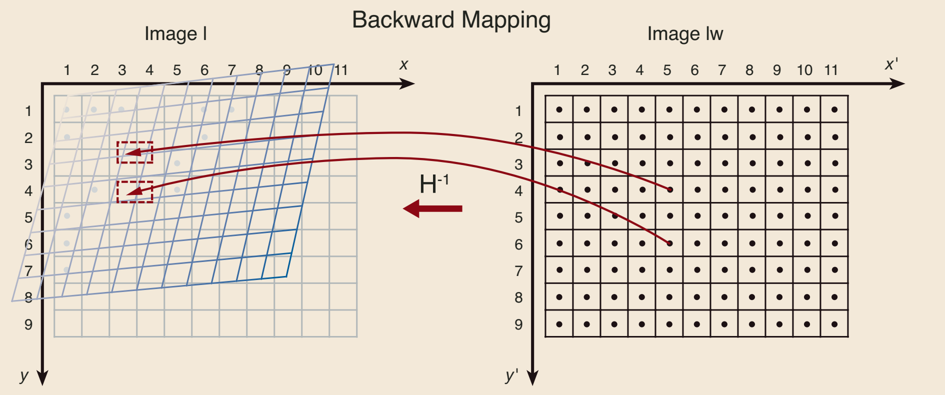
By applying the appropriate transformations through a warping operation and merging the overlapping regions of a warped image, we can construct a single panoramic image covering the entire visible area of the scene. Image mosaicing provides a powerful way to create detailed three-dimensional models and scenes for virtual reality scenarios based on real imagery. It is employed in flight simulators, interactive multi-player games, and medical image systems to construct true scenic panoramas or limited virtual environments.



```
INPUT: source image I
For every y from 1 to height (I)
   For every x from 1 to width(I)
      Calculate x', u = round(x')
      Calculate y', v = round(y')
      If 1<= u <= wodth(Iw) and 1<= v <= height(Iw)
         Copy I(x,y) to Iw(u,v)

OUTPUT: warped image Iw
```

Figure 2 – Forward mapping with a scaling factor greater than one



Iw(4,5) = interpolation of { I(3,2), I(4,2), I(3,3), I(4,3)}
Iw(4,7) = interpolation of { I(3,4), I(4,4), I(3,5), I(4,5)}

```
INPUT: source image I
For every v from 1 to height (Iw)
   For every u from 1 to width(Iw)
      Calculate x
      Calculate y
      Calculate Iw(u,v)as bilinear interpolation the four pixel values:

        $I(\lfloor x \rfloor, \lfloor y \rfloor)$, $I(\lfloor x \rfloor, \lceil y \rceil)$, $I(\lceil x \rceil, \lceil y \rceil)$, $I(\lceil x \rceil, \lfloor y \rfloor)$ such as
```

$$Iw(u,v) = (1-h)\ (1-k)\bullet p00 + h\bullet(1-k)\bullet p10 + (1-h)\bullet k\bullet p01 + h\bullet k\bullet p11$$

where:
$$\begin{cases} h = x - \lfloor x \rfloor, k = y - \lfloor y \rfloor \\ p00 = \mathrm{I}(\lfloor x \rfloor, \lfloor y \rfloor), p10 = \mathrm{I}(\lceil x \rceil, \lfloor y \rfloor), p01 = \mathrm{I}(\lfloor x \rfloor, \lceil y \rceil), p11 = \mathrm{I}(\lceil x \rceil, \lceil y \rceil) \end{cases}$$

OUTPUT: warped image Iw

Figure 3 – Backward mapping with a scaling factor greater than one

Conclusion

The challenge is to design efficient, effective, and reliable vision modules with the highest possible reliability.

Ultimodule, a Xilinx XPERTS partner, and the VIPS Laboratory at the Università di Verona have defined a foundation platform for computer vision using Ultimodule's system-on-module family. The platform provides a stereovision system for real-time extraction of three-dimensional data and a real-time image-processing engine implementing most of the algorithms required when an application relies on vision to make decisions and provide control.

The platform supports applications that require high performance and robust vision analysis, both in qualitative and computational terms (real-time), including active video surveillance, robotic arm motion and control, autonomous vehicle navigation, test and measurement, and hazard detection. The platform provides modules with all required system control logic, memory, and processing hardware, together with the application software. Interconnecting modules allow fast development of a complex architecture.

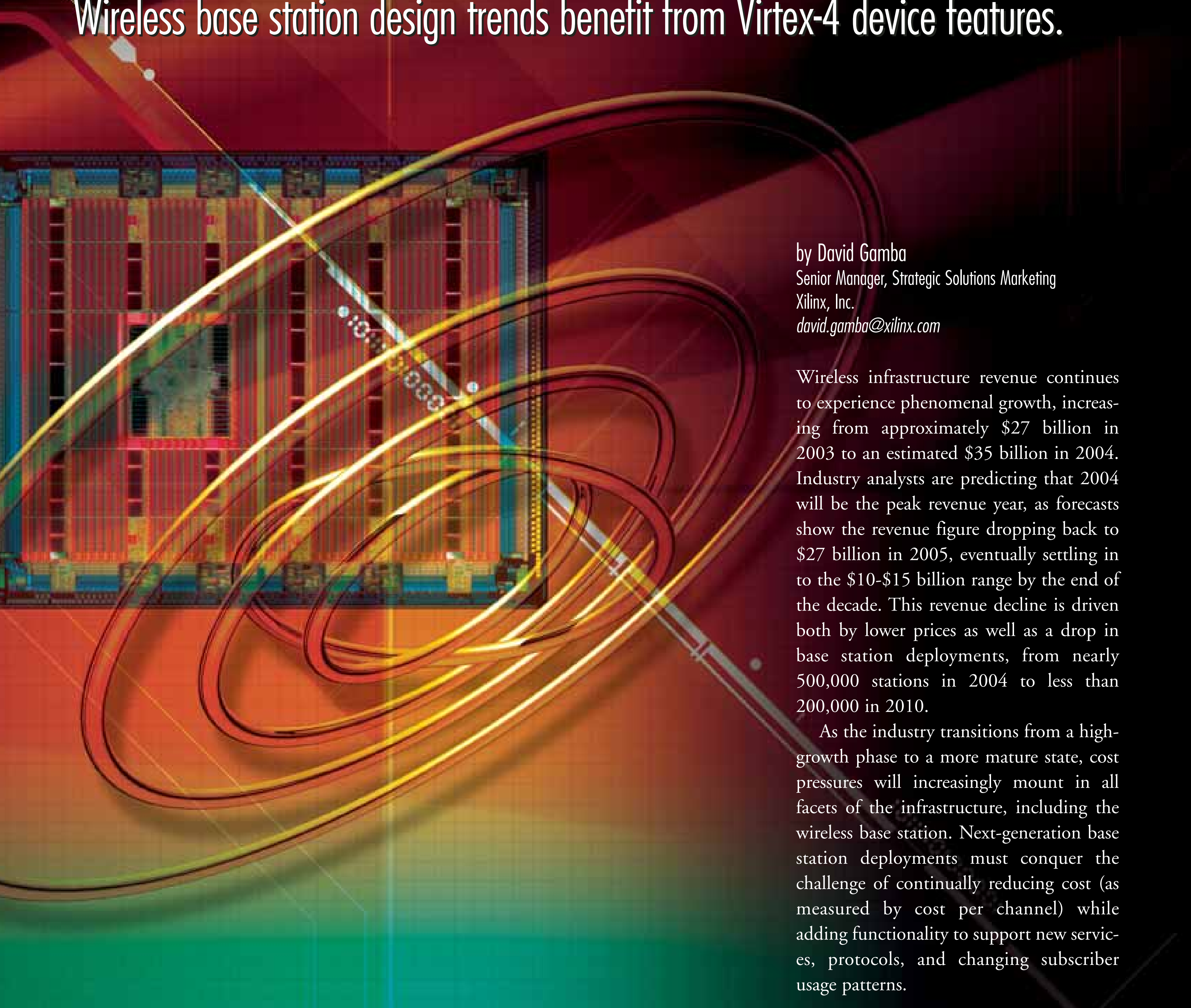
The platform leverages Xilinx Spartan-3 devices, which are an optimal choice for image processing IP cores because of their flexibility, high performance, and DSP-oriented targeting. The Spartan-3 family provides a valid, programmable alternative to ASICs. This characteristic, coupled with its low cost structure, adds considerable value when time to market is crucial.

For more information about feature extraction, you can e-mail the authors at *paolo.giacon@students.univr.it* or *saul.saggin@students.univr.it*. For more information about image warping, you can e-mail *matteo.busti@students.univr.it* or *giovanni.tommasi@students.univr.it*.

We are grateful for the support from our advisor, Professor Murino, in the Vision, Image Processing, and Sound (VIPS) Laboratory in the Dipartimento di Informatica at the Università di Verona, and contributions from Marco Monguzzi, Roberto Marzotto, and Alessandro Negrente.

Using FPGAs in Wireless Base Station Designs

Wireless base station design trends benefit from Virtex-4 device features.



by David Gamba
Senior Manager, Strategic Solutions Marketing
Xilinx, Inc.
david.gamba@xilinx.com

Wireless infrastructure revenue continues to experience phenomenal growth, increasing from approximately $27 billion in 2003 to an estimated $35 billion in 2004. Industry analysts are predicting that 2004 will be the peak revenue year, as forecasts show the revenue figure dropping back to $27 billion in 2005, eventually settling in to the $10-$15 billion range by the end of the decade. This revenue decline is driven both by lower prices as well as a drop in base station deployments, from nearly 500,000 stations in 2004 to less than 200,000 in 2010.

As the industry transitions from a high-growth phase to a more mature state, cost pressures will increasingly mount in all facets of the infrastructure, including the wireless base station. Next-generation base station deployments must conquer the challenge of continually reducing cost (as measured by cost per channel) while adding functionality to support new services, protocols, and changing subscriber usage patterns.

To begin addressing this challenge, wireless base station designs are shifting from ASIC technology to more readily available off-the-shelf components such as FPGAs. This shift is driven both by declining annual base station unit volumes as well as FPGA technology improvements that increase processing power and enable a much lower cost per channel.

The migration to FPGAs is not just an attempt to reduce costs and create a common platform to achieve commoditization – it is also being driven by time-to-market pressures, along with the need to make in-field upgrades of base station deployments. This shift away from ASICs has enabled significant new design opportunities for Xilinx® Virtex-4™ devices to fill the void.

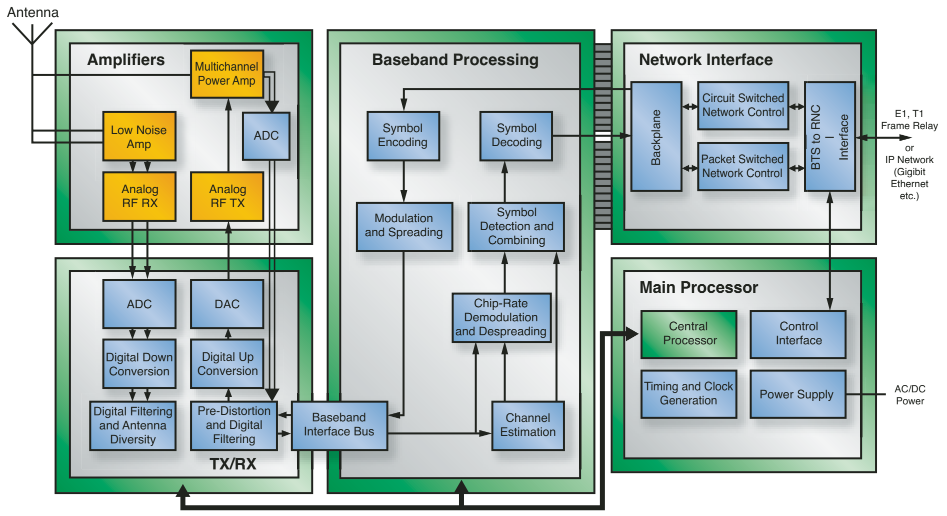


Figure 1 – Wireless base station module block diagram

Wireless Base Station Module Building Blocks

Inside a wireless base station are fairly distinct module blocks performing different functions, such as radio, baseband processing, transport network interfacing, and control (Figure 1). Traditional base station designs used ASICs – along with DSPs and other discrete components – to implement these various architectural features and functions.

This design approach is rapidly giving way to more cost-effective and flexible designs that use FPGAs. With lower costs and increased flexibility, product delivery is accelerated and inventory control is much more manageable, avoiding some of the multi-million dollar inventory obsolescence issues that base station manufacturers have faced with ASIC solutions fabricated to support the 3G launch.

Standardizing the Wireless Base Station

Another significant step taken by the wireless industry is the launch of industry organizations focused on standardizing the non-differentiated features inside a base station. The most notable development for Xilinx is the migration to a standardized high-speed serial interconnect solution between the different base station module blocks, such as the Open Base Station Architecture Initiative (OBSAI) Reference Point 3 (RP3) and Common Public Radio Interface (CPRI) interconnects for baseband and radio module connectivity.

Many leading base station manufacturers are members of these organizations and are rapidly preparing to adopt one of these two standard interconnect solutions in their upcoming design implementations. Xilinx is fully prepared to support these standards, and has both OBSAI and CPRI IP solutions and reference designs available for implementing in Virtex-II Pro™, Virtex-II Pro X, and Virtex-4 FX FPGA devices, using the integrated RocketIO™ multi-gigabit tranceivers (MGTs) in association with the logic building blocks.

Extending Current Design Lifecycles

Standardization is the first step towards the commoditization of base station design and will eventually lead to a phasing out of ASICs from wireless base stations. In the interim, companies are inserting discrete devices next to their current ASICs to support new functionality that cannot be added in a timely or cost-effective manner to the current design.

For instance, the Third Generation Partnership Project (3GPP), which is a collaboration agreement between several telecommunications bodies, is actively creating additional standards for the wireless industry. 3GPP has added a high-speed downlink packet access (HSDPA) feature as a new Universal Mobile Telecommunications System (UMTS) requirement in its latest baseband processing specification, Release 5, for Wideband Code Division Multiple Access (W-CDMA).

ASICs in current base stations do not support this new variant for UMTS. This creates a hole in the service offerings for UMTS, which forecasters are predicting will represent approximately 80% of the wireless traffic in the next few years. This deficiency must be addressed before future field deployments, and it can be – without exceeding the system power budget – by using a Virtex-4 LX device next to the ASIC, implementing HSDPA using the available Xilinx HSDPA IP offering.

Next-Generation Base Station Designs

But adding external devices to patch design holes created by existing ASIC designs limitations is purely a stopgap solution. Future base station designs must be able to quickly adapt to changes in subscriber traffic patterns, as well as support the upcoming convergence of new services and emerging cellular technologies such as W-CDMA, TD-SCDMA, EDGE, 1xEV-DO, and WiMAX.

As shown in Figure 2, the amount of cellular technologies is expected to continue to proliferate, leading base stations down the path of having to support many more technologies. Current issues such as

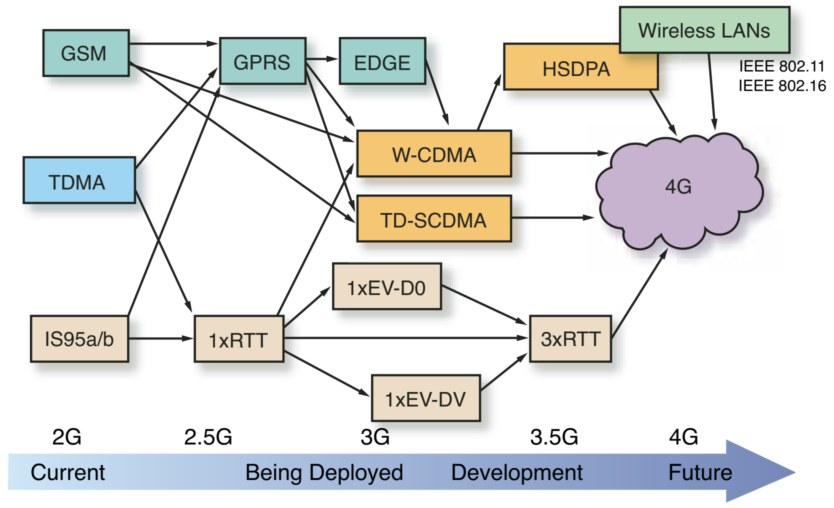


Figure 2 – Mobile technology roadmap

multi-user detection and antenna selection will be augmented by new technical challenges, such as channel provisioning and base station tuning, that will need to be resolved appropriately to reduce a service provider's customer turnover. The fundamental expectation to receive the same high-quality wireless service wherever a customer roams must be completely addressed.

These customer expectations would benefit from substantial flexibility in the base station. Fortunately, many of the baseband processing functions and radio module functions are well suited for implementation in Virtex-4 devices, taking advantage of the integrated XtremeDSP™ slices in the product architecture.

For instance, quite a few baseband processing tasks – such as call initiation and set-up and multi-path signal detection and monitoring – are heavily based on mathematical algorithms. You can very efficiently implement these algorithms by using the integrated multiplier capabilities available in Virtex-4 devices, along with the readily available intellectual property components such as the Random Access Channel (RACH), Searcher, and 3G Turbo Convolutional Codecs (3GTCC) that Xilinx has implemented as reference designs to demonstrate these capabilities.

The integrated DSP capability in the Virtex-4 SX device enables a very low power implementation of these functions. Radio functions can be expanded by using a Virtex-4 SX device to enable more channel support.

Several enabling pieces of intellectual property targeted at radio functions, such as digital pre-distortion (DPD), crest factor reduction (CFR), and digital up/down conversion (DUC/DDC), are supported by the Virtex-4 SX device. Not only does this help increase the number of channels supported in a base station, but it also helps reduce the cost per channel. Table 1 gives an overview of the different capabilities offered by Xilinx baseband and radio module IP offerings.

Xilinx Baseband Intellectual Property Offerings	
IP Offering	Application
HSDPA	Increases downlink data transmission rate to a peak of 14.4 Mbps
RACH	Receiver path preamble detection (specified by W-CDMA)
Searcher	Multi-path delay estimate for each subscriber
3G TCC	Forward error correction
Xilinx Radio Intellectual Property Offerings	
IP Offering	Application
DPD	Signal conditioning to enable use of lower cost RF power amplifiers
CFR	Signal amplitude conditioning to enable increased RF power amplifier efficiency
DUC	Baseband signal modulation for digital-to-analog converter input
DDC	Receiver signal modulation for analog-to-digital converter input

Table 1 – Xilinx baseband and radio IP offerings

System Generator for DSP Development Tool

Xilinx complements its Virtex-4 product offerings with the System Generator for DSP tool. This is a complete integrated DSP design environment that simplifies the development, debug, and verification of high-performance DSP designs targeting wireless base stations. This tool also helps designers interface with complementary general-purpose and DSP processors used in wireless base station designs.

System Generator for DSP provides high-level abstractions that are automatically compiled into Virtex-4 devices at the push of a button, with no loss in performance over designs implemented in lower-level languages such as VHDL. System Generator is part of the XtremeDSP solution, which combines state-of-the-art FPGAs, design tools, intellectual property cores, and design and education services.

Conclusion

To learn more about the key markets and end applications of Xilinx wireless solutions, visit *www.xilinx.com/esp/*, or e-mail *3g@xilinx.com*. For more details about Virtex-4 FPGAs, visit *www.xilinx.com/virtex4/*. And for more details on System Generator for DSP or other pieces of the Xilinx DSP solution, visit *www.xilinx.com/dsp/*.

DAVINCI™

Simple, affordable, high-performance video processing in any format, on any device, in any seat in the house. That's the DaVinci Effect.

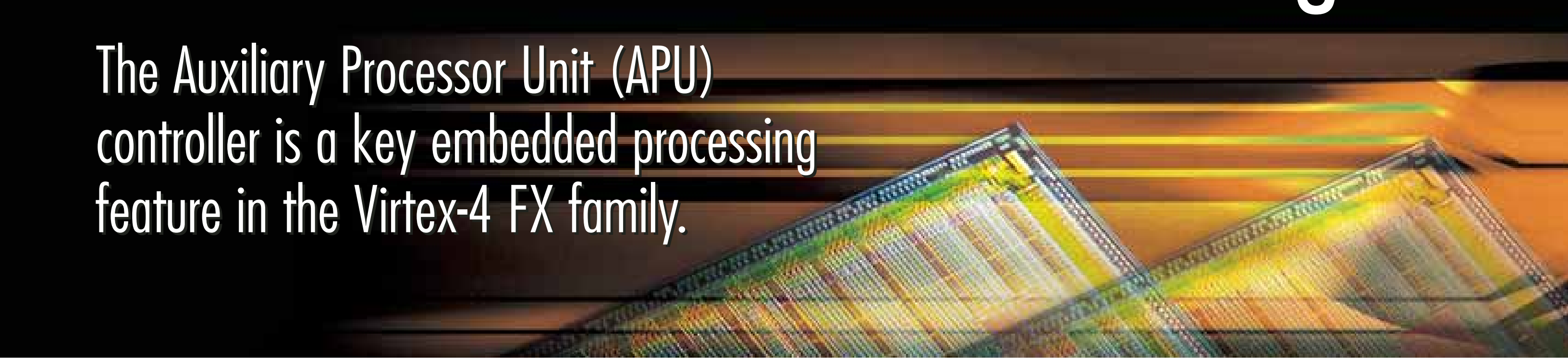
DaVinci™ technology is a DSP-based system solution tailored for digital video applications that provides optimized software, development tools, integrated silicon, and support to simplify design and stimulate innovation in less time. It consists of:

- **DaVinci Optimized Software:** Interoperable, optimized, off-the-shelf digital video and audio codecs, protocols, and user interfaces leveraging integrated accelerators, published APIs, and application specific frameworks that utilize a variety of real-time operating systems for rapid implementation
- **DaVinci Development Tools:** Complete development kits, reference designs, and comprehensive ARM/DSP system-level IDEs to speed design
- **DaVinci Integrated Silicon:** Scalable, programmable DSP-based system-on-chip solutions tailored for digital video applications
- **DaVinci Support/Ecosystem:** System integrators, hardware and software providers, as well as TI and third party comprehensive video system expertise

When your video is better than live, that's the DaVinci Effect. See live action like you've never imagined: closer, clearer and crisper. DaVinci™ technology from Texas Instruments allows a high-performance, video processing platform to stream directly to a handheld device, an on-board system in your car or your home entertainment center. And because of its DSP-based programmability, you can create unique, feature-rich devices optimized with specific applications in mind and get them to market quickly. The applicability is greater, the design process is faster, and the time to begin is now. For a technical brief, go to **www.thedavincieffect.com.**

DaVinci, Technology for Innovators and the red/black banner are trademarks of Texas Instruments. 1140A0 © 2005 TI

Technology for Innovators™ TEXAS INSTRUMENTS



Accelerated System Performance with APU-Enhanced Processing

The Auxiliary Processor Unit (APU) controller is a key embedded processing feature in the Virtex-4 FX family.

by Ahmad Ansari
Senior Staff Systems Architect
Xilinx, Inc.
ahmad.ansari@xilinx.com

Peter Ryser
Manager, Systems Engineering
Xilinx, Inc.
peter.ryser@xilinx.com

Dan Isaacs
Director, APD Embedded Marketing
Xilinx, Inc.
dan.isaacs@xilinx.com

The APU controller provides a flexible high-bandwidth interface between the reconfigurable logic in the FPGA fabric and the pipeline of the integrated IBM™ PowerPC™ 405 CPU. Fabric co-processor modules (FCM) implemented in the FPGA fabric are connected to the embedded PowerPC processor through the APU controller interface to enable user-defined configurable hardware accelerators. These hardware accelerator functions operate as extensions to the PowerPC 405, thereby offloading the CPU from demanding computational tasks.

APU Instructions

The APU controller allows you to extend the native PowerPC 405 instruction set with custom instructions that are executed by the soft FCM; the primary capabilities are shown in Figure 1. This provides a more efficient integration between an application-specific function and the processor pipeline than is possible using a memory-mapped coprocessor and shared bus implementation.

The instructions supported by the APU are classified into three main categories:

- User-defined instructions (UDI)
- PowerPC floating-point instructions
- APU load/store instructions

The UDIs are programmed into the controller either dynamically through the PowerPC 405 device control register (DCR) or statically when the FPGA is configured through its bitstream. The APU controller allows you to optimize your system architecture by decoding instructions either internally or in the FCM.

The floating-point unit (FPU) is an example of an FCM. The PowerPC floating-point instruction set is decoded in the APU controller, whereas the computational functionality is implemented in the FPGA fabric. To support FPUs with different complexities, the APU controller allows you to select subgroups of the PowerPC floating-point instructions. These instructions are executed in the FCM while other subgroups of instructions are either computed through software FPU emulation or ignored completely. This fine-tuning optimizes FPGA resources while accelerating the most critical calculations with dedicated logic.

The APU controller also decodes high-performance load and store instructions between the processor data cache or system memory and the FPGA fabric. A single instruction transfers up to 16 bytes of data – four times greater than a load or store instruction for one of the general purpose registers (GPR) in the processor itself. Thus, this capability creates a low-latency and high-bandwidth data path to and from the FCM.

APU Controller Operation

Figure 2 identifies the key modules of the APU controller and the 405 CPU in relation to the FCM soft coprocessor module implemented in FPGA logic. To explain the operation of the APU controller and the processor interactions related to the execution units in soft logic, we can trace the step-by-step sequence of events that occur when an instruction is fetched from cache or memory.

Once the instruction reaches the decode stage, it is simultaneously presented to both the CPU and APU decode blocks. If the instruction is detected as a CPU instruction, the CPU will continue to execute the instruction as it would normally. Otherwise, within the same cycle, the CPU

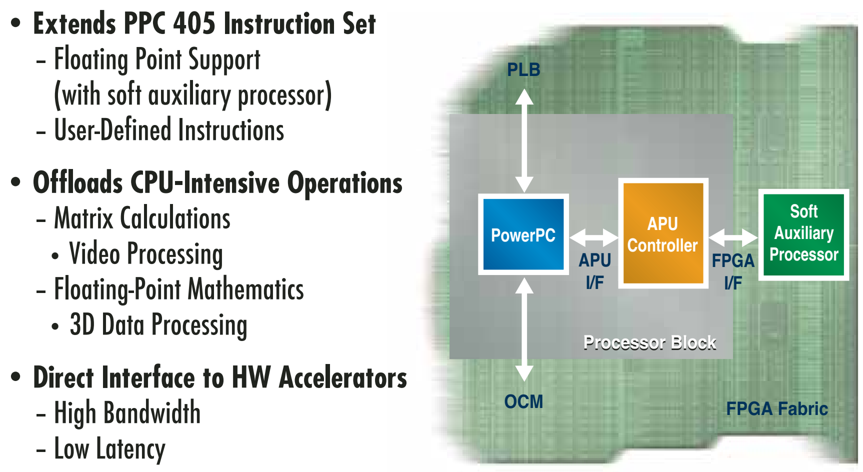

• Extends PPC 405 Instruction Set
– Floating Point Support (with soft auxiliary processor)
– User-Defined Instructions
• Offloads CPU-Intensive Operations
– Matrix Calculations
• Video Processing
– Floating-Point Mathematics
• 3D Data Processing
• Direct Interface to HW Accelerators
– High Bandwidth
– Low Latency
PLB
PowerPC
APU I/F
APU Controller
FPGA I/F
Soft Auxiliary Processor
Processor Block
OCM
FPGA Fabric

Figure 1 – APU expanded processing capabilities

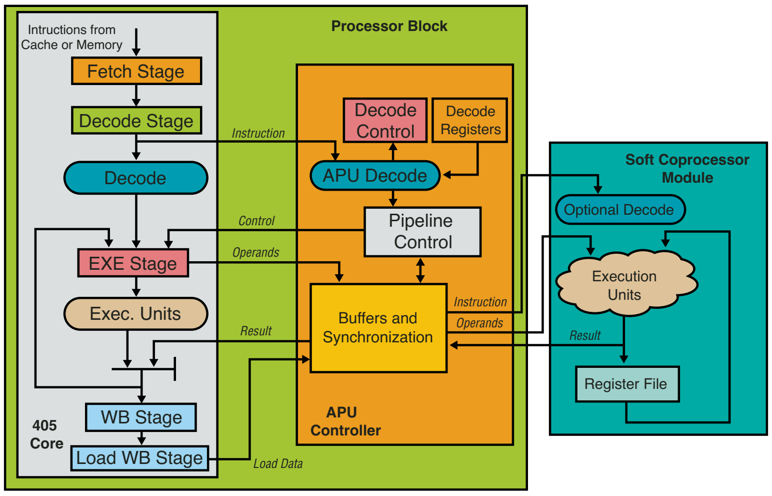


Figure 2 – APU controller processing operative block diagram

will look for a response from the APU controller. If the APU controller recognizes the instruction, it will provide the necessary information back to the CPU.

If the APU controller does not respond within that same cycle, an invalid instruction exception will be generated by the CPU. If the instruction is a valid and recognized instruction, the necessary operands are fetched from the processor and passed to the FCM for processing.

Because the PowerPC processor and the FCM reside in two separate clock domains, synchronization modules of the APU controller manage the clock frequency difference. This allows the FCM to operate at a slower frequency than the processor. In this instance, the APU controller would receive the resultant data from the coprocessor and at the proper execution time send the data back to the processor. The APU controller knows in advance, based on instruction type, if or when it will get the result.

Autonomous and Non-Autonomous Instructions

Two major categories of instructions exist: autonomous and non-autonomous. For autonomous instructions, the CPU continues issuing instructions and does not stall while the FCM is operating on an instruction. This overlap of execution allows you to achieve high performance through techniques such as software pipelining.

On the other hand, during the synchronized execution, the CPU pipeline stalls while the FCM is operating on an instruction. This feature allows you to implement synchronization semantics to pace the software execution with the hardware FCM latency.

Non-autonomous instruction types are further divided into blocking and non-blocking. If blocking, asynchronous exceptions or interrupts are blocked until the FCM instruction completes. Otherwise, if non-blocking, the exception or interrupt is taken and the FCM is flushed.

Software Description

Software engineers can access the FCM from within assembler or C code. On one side, Xilinx has enabled the GCC compiler (which is contained in the Embedded Development Kit) to generate code that uses an FCM floating-point unit to calculate floating-point operations. Furthermore, assembler mnemonics are available for UDIs and the pre-defined load/store instructions, enabling you to place hardware-accelerated functions into the regular program flow. For the ultimate level of flexibility, you can define your own instructions designed specifically for the hardware functionality of the FCM.

You can easily use the pre-defined load/store instructions through high-level C macros. For example, in an application where the FCM is used to convert pixel data into the frequency domain, 8 pixels of 16 bits are transferred from main memory to an FCM register with a simple program:

```
unsigned short pixel_row[8]; // 8 pixels,
each pixel has a size of 16 bits

lqfcm(0, pixel_row); // transfer a row of
pixels to FCM register zero
```

The quadword load operation maintains cache coherency as the data is moved through the cache, if caching is enabled for the corresponding address space.

The FCM operation on the pixel data can start on an explicit command; for example, a UDI. However, for many applications the operation starts immediately after the FCM hardware detects the completion of the load instruction.

The latter approach has many advantages:

- Simple software – A load operation moves the data from the memory to the FCM and starts the operation. A subsequent store instruction retrieves the result of the operation and stores it back to main memory.

- High data transfer rates – Quadword load and store operations take just a few cycles to complete. A single operation moves 16 bytes within that timeframe.
- Low latency – FCM load operations are simple to use. The processor completes the operation in a single cycle.

The principle of the RISC architecture uses a number of simple instructions on data stored in general-purpose registers (GPR) to compute complex operations. User-defined instructions fall into this category but take the concept a step further in that the system architect defines the complexity of the operation on data stored in GPRs and FCM registers (FCR). Again, from a software point of view, the engineer codes user-defined instructions through C macros. GCC recognizes mnemonics such as udi0fcm as a user-defined operation of the general form:

```
udi0fcm<FCRT5/RT5>,<FCRA5/RA5/imm>,
<FCRB5/RB5/imm>
```

The target of the operation is either a GPR or an FCR. The operands are either GPRs, FCRs, immediate values, or a combination. As you can see, the semantics are not defined by the instruction and depend on your intentions and the implementation in the FCM.

This code sequence demonstrates the use of a user-defined instruction as an example of a complex add operation:

```
struct complex {
        int r, i;       // 32 bit integer for real
        and imaginary parts
};
complex a, b, r;
ldfcm(0, &a);       // load complex number a
into FCM register 0
ldfcm(1, &b);       // load complex number b
into FCM register 1
udi0fcm(2, 1, 0);// udi0fcm computes r = a
+ b, where r is stored in FCM register 2
stdfcm(&r, 2);      // store complex result
from FCM register 2 to variable r
```

To increase the readability of the code, you can redefine the user-defined instruction with regular C preprocessor constructs. Instead of using the udi0fcm() macro, you can redefine it to a more comprehensible complex_add() macro with `#define complex_add(r, a, b) udi0fcm(r, a, b)` and change the listing to call complex_add(2, 1, 0) instead of udi0fcm(2, 1, 0).

Therefore, system architects can partition their tasks into hardware- and software-executed pieces that are efficiently and precisely interfaced to one another through the use of the APU controller. This partitioning can be done statically during the initial system configuration or dynamically during the program execution. Using the direct processor/FPGA coupling presented by the APU controller and its high throughput interfaces, hardware/software synchronization is greatly simplified and performance significantly improved.

Accelerating System Performance

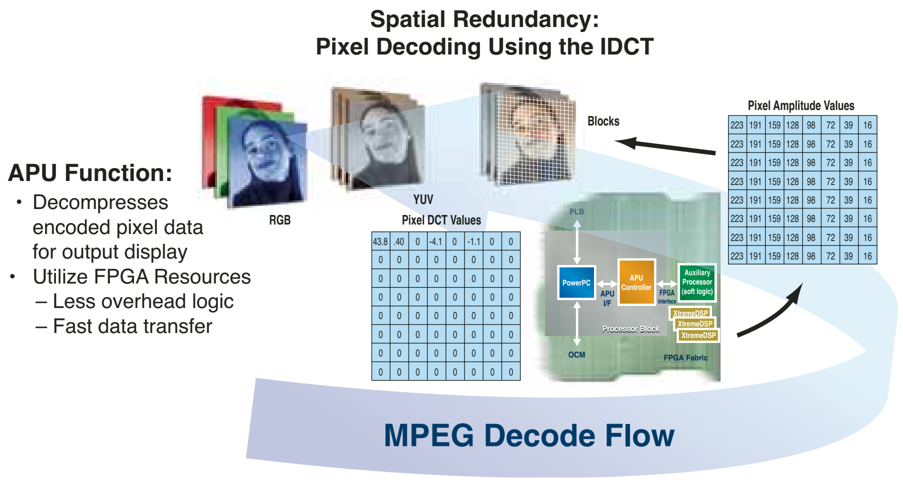
The following examples showcase key advantages the APU provides based on two different scenarios. The first scenario is essentially a benchmarking comparison of a finite impulse response (FIR) filter using a soft FPU core, implemented as an FCM attached directly to the APU controller (as compared to software emulation used to calculate the filter function). The second scenario implements a two-dimensional inverse discrete cosine transform (2D-IDCT) typically used as one of the processing blocks in MPEG-2 video decompression, again compared to emulating the 2D-IDCT function in software.

The two use cases are different in that the FPU implements a set of registers in the FPGA fabric upon which the FPU instructions operate. The 2D-IDCT only requires load and store operations, while the functionality of the operation on the data stream is fixed. In either case the operations are complex enough to justify offloading into the FPGA fabric.

Thus, the combination of using the APU and FPGA hardware acceleration clearly provides a significant performance advantage over software emulation – or the conventional method involving the processor and processor local bus architecture with a soft co-processing function.

FIR Filter

The implementation of floating-point calculations in hardware yields an improvement by a factor of 20 over software emulation. Connecting the FPU as an FCM to the APU controller provides performance improvement because the latency to access the floating-point registers is reduced and dedicated load and store instructions move the operands and results between the FPU registers and the system memory.



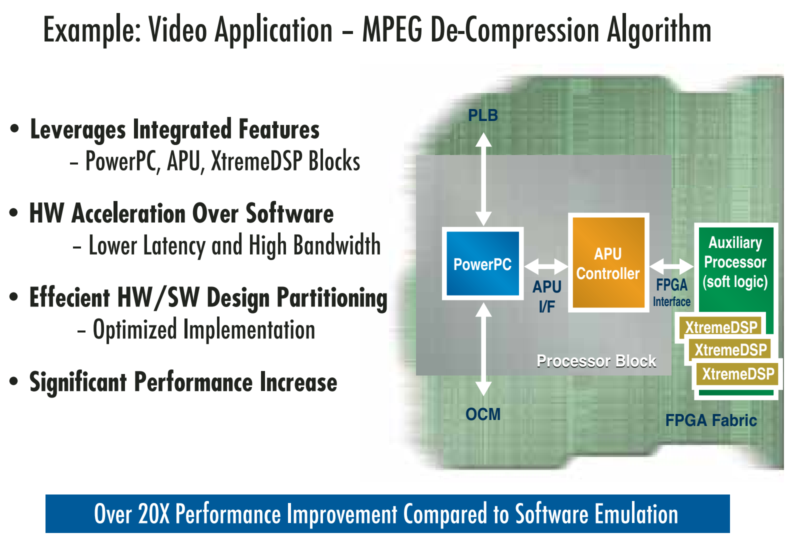
Pixel DCT Values

43.8	.40	0	-4.1	0	-1.1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Pixel Amplitude Values

223	191	159	128	98	72	39	16
223	191	159	128	98	72	39	16
223	191	159	128	98	72	39	16
223	191	159	128	98	72	39	16
223	191	159	128	98	72	39	16
223	191	159	128	98	72	39	16
223	191	159	128	98	72	39	16
223	191	159	128	98	72	39	16

Figure 3 – Utilizing APU to decode pixel data for display output


Example: Video Application – MPEG De-Compression Algorithm

- **Leverages Integrated Features**
  - PowerPC, APU, XtremeDSP Blocks
- **HW Acceleration Over Software**
  - Lower Latency and High Bandwidth
- **Effecient HW/SW Design Partitioning**
  - Optimized Implementation
- **Significant Performance Increase**

Over 20X Performance Improvement Compared to Software Emulation

Figure 4 – Accelerated system performance with APU

2D-IDCT

The 2D-IDCT transforms a block of 8 x 8 data points from the frequency domain into pixel information. A high-level diagram depicting the pixel decode by the APU controller, along with advantages, is shown in Figure 3. In this example, each data point has a resolution of 12 bits and is represented as a 16-bit integer value. The data structure is defined where each row of 8 pixels consumes 16 bytes. This is an ideal size that allows optimal use of the FCM load and store instructions described earlier. In other words, eight FCM quadword load instructions are needed to load a data block into the 2D-IDCT hardware. Eight FCM quadword store instructions are sufficient to copy the pixel data back into the system memory.

The calculation of the 2D-IDCT in the FCM starts immediately after the first load, and the pixel data is available shortly after the last load operation. As shown in Figure 4, the 2D-IDCT makes uses of the new XtremeDSP™ slices in the Virtex-4 architecture that offer multiply-and-accumulate functionality.

A software-only implementation of a 2D-IDCT takes 11 multiplies and 29 additions together with a number of 32-bit load and store operations, while the hardware-accelerated version takes 8 load and 8 store operations. The reduced number of operations results in a speed-up of 20X in favor a 2D-IDCT FCM attached through the APU controller.

By comparison, if you connect the 2D-IDCT hardware block to the processor local bus, as it is done conventionally, the system performance will be reduced. This increased latency is mainly caused by the bus arbitration overhead and the large number of 32-bit load and store instructions. This is illustrated schematically in Figure 5.

Conclusion

The low-latency and high-bandwidth fabric coprocessor module interface of the APU controller enables you to accelerate algorithms through the use of dedicated hardware. Where operations are complex enough to justify the offloading into the FPGA fabric, or when acceleration of a specific algorithm is desired to achieve optimal performance, the combination of the APU controller and FPGA hardware acceleration provides a definitive performance advantage over software emulation or the conventional method of attaching coprocessors to the processor memory bus.

Generating the accelerated functions called by user-defined instructions is easily performed through GUI-based wizards. This functionality will be included in subsequent releases of the powerful Embedded Development Kit or Platform Studio.

If you are more comfortable working at the source code or assembly level, the APU controller allows you to define your own instructions written specifically for the hardware functionality of the FCM, or you can easily use the pre-defined load/store instructions through high-level C macros.

The APU controller provides a close coupling between the PowerPC processor and the FPGA fabric. This opens up an entire range of applications that can immediately benefit customers by achieving increases in system performance that were previously unattainable.

For additional details on the APU controller in Virtex-4-FX devices, including detailed descriptions and timing waveforms, refer to the Virtex-4 PowerPC 405 Processor Block Reference Guide at *www.xilinx.com/bvdocs/userguides/ug018.pdf.*

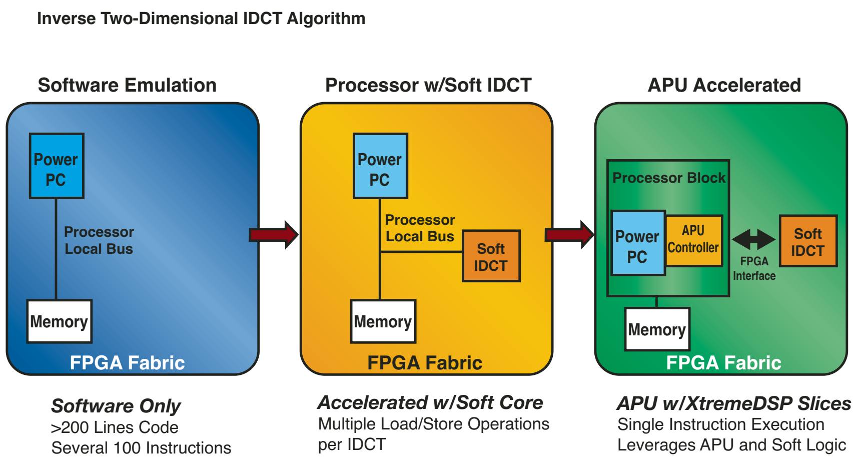
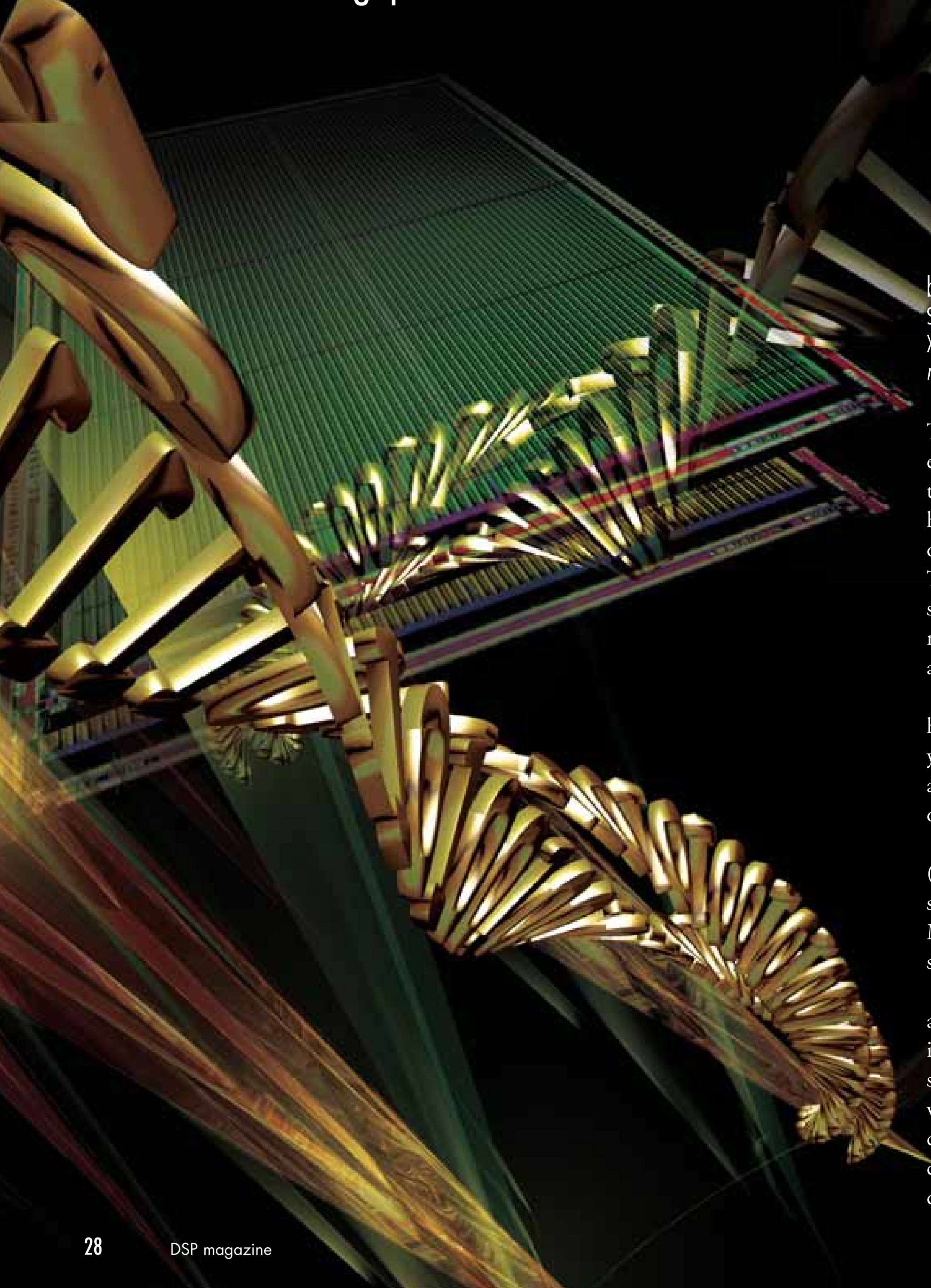


Figure 5 – Comparison of implementation models for 2D-IDCT

Alpha Blending Two Data Streams Using a DSP48 DDR Technique

Achieve full throughput of the DSP48 slice with a double-data-rate technique.



by Reed Tidwell
Sr. Staff Applications Engineer
Xilinx, Inc.
reed.tidwell@xilinx.com

The XtremeDSP™ system feature, embodied as the DSP48 slice primitive in the Xilinx® Virtex-4™ architecture, is a high-performance computing element operating at an industry-leading 500 MHz. The design of the Virtex-4 infrastructure supports this rate, with Xesium clock technology, Smart RAM, and LUTs configured as shift registers.

Many applications, however, do not have data rates of 500 MHz. So how can you harness the full computing performance of the DSP48 slice with data streams of lower rates?

The answer is to use a double-data-rate (DDR) technique through the DSP48 slice. The DSP48 slice, operating at 500 MHz, can multiplex between two data streams, each operating at 250 MHz.

One application of this technique is alpha blending of video data. Alpha blending refers to the combination of two streams of video data according to a weighting factor, called alpha. In this article, we'll explain the techniques and design considerations for applying DDR to two data streams through a single DSP48 slice.

All Virtex-4 devices have DSP48 slices, although the SX family contains the largest number (an industry-high 512) and the highest concentration of DSP48 slices to logic elements, making it ideal for math-intensive applications ...

Virtex-4 DSP48

The DSP system elements of Virtex-4 FPGAs are dedicated, diffused silicon with dedicated, high-speed routing. Each is configurable as an 18 x 18-bit multiplier; a multiplier followed by a 48-bit accumulator (MACC); or a multiplier followed by an adder/subtracter. Built-in pipeline stages provide enhanced performance for 500 MHz throughput – 35% higher than for competing technologies.

All Virtex-4 devices have DSP48 slices, although the SX family contains the largest number (an industry-high 512) and the highest concentration of DSP48 slices to logic elements, making it ideal for math-intensive applications such as image processing.

A triple-oxide 90 nm process makes the DSP48 slice very power-efficient. Architectural features, including built-in pipeline registers, accumulator, and cascade logic nearly eliminate the use of general-purpose routing and logic resources for DSP functions, and further reduce power. This slashes DSP power consumption to a fraction when compared to Virtex-II Pro™ devices.

DDR with Two Data Streams

DDR, in this context, refers to multiplexing two input data streams into one stream at twice the rate, interleaving (in time) the data from each stream (Figure 1). Figure 1 also shows the reverse operation, creating two parallel resultant streams after processing.

You can drive the DSP48 slice inputs at the fast 500 MHz clock rate from CLB flip-flops; CLB LUTs configured as shift registers (SRL16); or directly from block RAM. Block RAM, configured as a FIFO using the built-in FIFO support, also supports the 500 MHz clock rate.

Design Considerations

Dealing with data at 500 MHz requires great care; you should observe strict pipelining with registers on the outputs of each math or logic stage. The DSP48 slice provides optional pipeline registers on the input ports, on the multiplier output, and on the output port from the adder/subtracter/accumulator. Block RAM also has an optional output register for efficient pipelining when interfaced to the DSP48 slice.

Where you are using CLBs, place only minimal levels of logic between registers to provide maximum speed. For DDR operation, only a 2:1 mux (a single LUT level) is required between pipeline stages. Whether you are interfacing to the DSP48 slice with memory or CLBs, placing connected 500 MHz elements in close proximity minimizes connection lengths in the general routing matrix.

DDR requires the DSP48 slice to operate at double the frequency of the input data streams. You can use a DCM to provide a phase-aligned double-frequency clock using the CLK 2X output.

Another aspect of inserting DDR data through a section of pipeline is ensuring that data passes cleanly between clock domains. This may require adding extra registers clocked with the double-frequency clock at the output of the double-pumped section, to synchronize the data with the original clock. The rule of thumb is that in order to insert a double-pumped section cleanly into a single-pumped pipeline, there must be an even number of register delays in the double-pumped section.

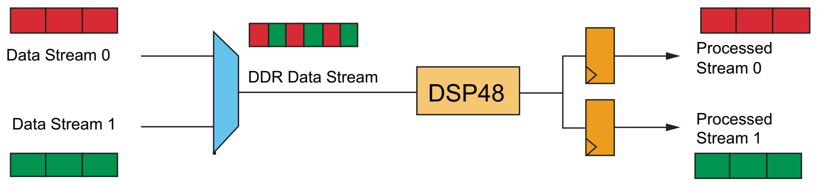


Figure 1 – DSP48 DDR

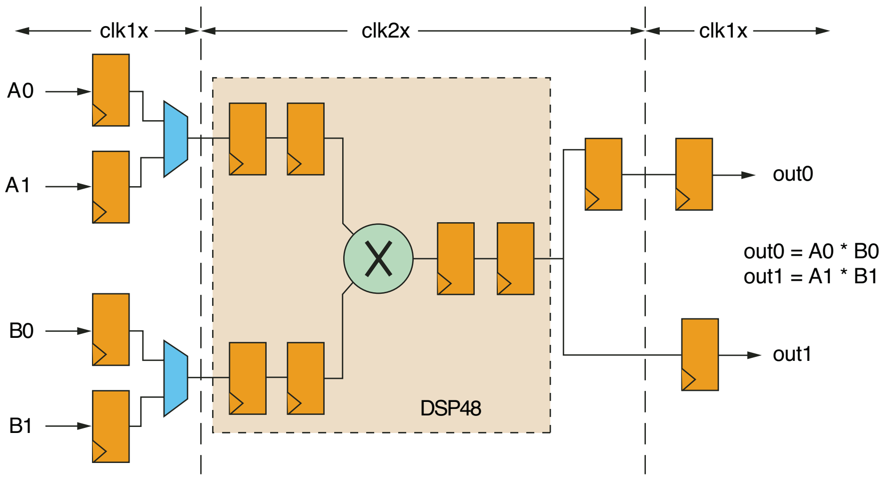


Figure 2 – Two-stream multiply through DSP48 slice

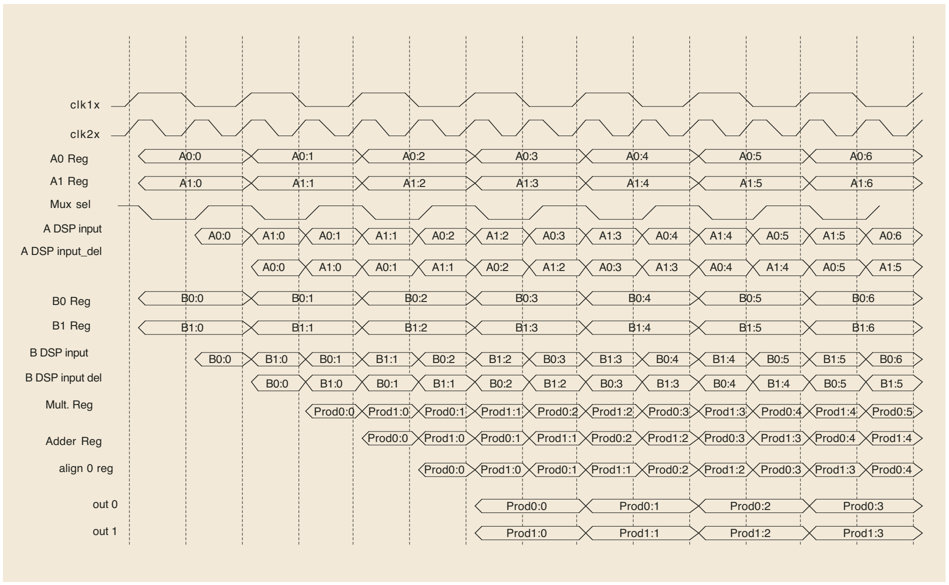


Figure 3 – Timing of two-stream multiply

Implementation

Several configuration options exist for implementing DDR functionality. Figure 2 shows a straightforward implementation.

In Figure 2, stream 0 consists of A0 and B0 inputs. We multiply them together and output as out0. Likewise, stream 1 consists of inputs A1 and B1 multiplied together and output as out1. There are two clock domains: the clk1x domain, at the nominal data stream frequency, and the clk2x domain, at twice the nominal frequency.

Figure 2 shows two registers after the multiplier. The second is the accumulation register, even though we do not use accumulation in this configuration. The register, however, is still required to achieve the full, pipelined performance. We use two sets of registers on the inputs of the DSP to make the total delay through the DSP48 slice an even number (four) for easier alignment of the output data with clk1x. These registers are "free" because they are built into the DSP48 slice, and using them reduces the need for alignment registers external to the DSP48 slice. The extra pipeline register on out0 compensates for taking stream 0 into the DSP one clk2x cycle before stream 1. As seen from the timing diagram in Figure 3, this is required to re-align the stream 0 data back into the clk1x domain.

Note that the input mux select, mux_sel, is essentially the inverse of clk1x. It is important, however, to generate this signal from a register based on clk2x (rather than deriving it from clk1x) to avoid hold-time violations on the receiving registers.

At the transitions between clock domains, the data have only one clk2x period to set up. This is the reason to have no logical operations between registers in the two domains. The placement of the first registers in the clk1x domain is more critical than other registers in the same domain.

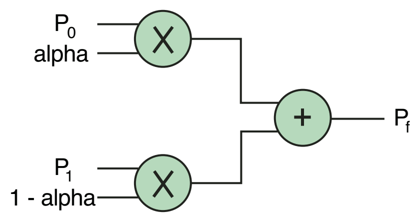


Figure 4 – Alpha blend formula in graphical terms

Alpha Blending

Alpha blending of video streams is a method of blending two images into a single combined image, such as fading between two images, overlaying anti-aliased or semi-transparent graphics over an image, or making a transition band between two images on a split-screen or wipe. Alpha is a weighting factor defining the percentage of each image in the combined output picture. For two input pixels

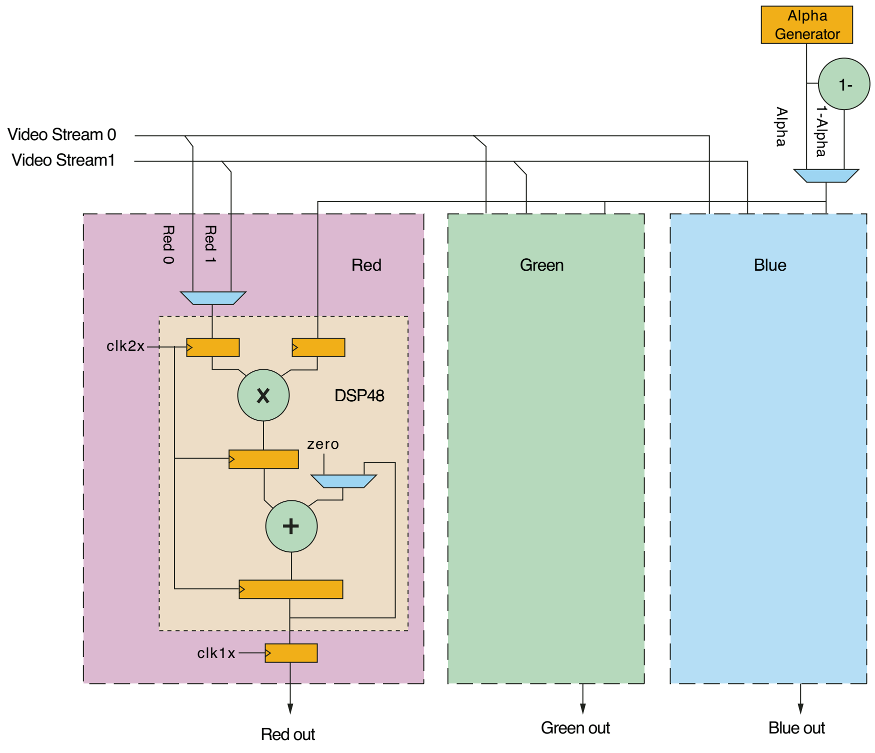


Figure 5 – Alpha blend on three-component video

You can efficiently use the high-performance of Virtex-4 devices with DSP48 slices by processing multiple data streams in a time-multiplexed fashion.

(P0, P1, and a blend factor, α, where 0 <=α < =1.0), the output pixel Pf will be:

$P_f = \alpha P_0 + (1-\alpha)P_1$ (see Figure 4)

This operation is performed separately for each component: red, green, and blue.

A pixel rate of 250 MHz or less is sufficient for all standard and high-definition video rates, and common Video Electronics Standards Association (VESA) standards as high as 1600 x 1200 at 85 Hz. Therefore, one DSP48 slice can perform the multiply and add on one component, and a set of three slices can alpha blend the three components from each of two video streams, as shown in Figure 5. The operations must be performed identically and in parallel on each of the three components.

There are several ways to implement alpha blending depending on the nature of the video streams and how alpha is generated. Figure 6 shows a basic implementation with two video streams alternating as one multiplier input. The other multiplier input alternates between alpha and 1- alpha.

The operating mode of the adder alternates between add zero (pass through) mode and add output (accumulate) mode. The DSP48 slice output register contains the result of the Video0 * alpha multiply during one clock cycle, and the final result (Video1 * (1 – alpha) + Video0 * alpha) on the alternate clock. Figure 7 shows the timing for this configuration.

The align registers on the inputs of the DSP are used to make the total delay through the DSP48 slice an even number (four), as explained in the previous example. The final output register for blend loads new data to every other DSP clock to register the blend results at the original pixel rate.

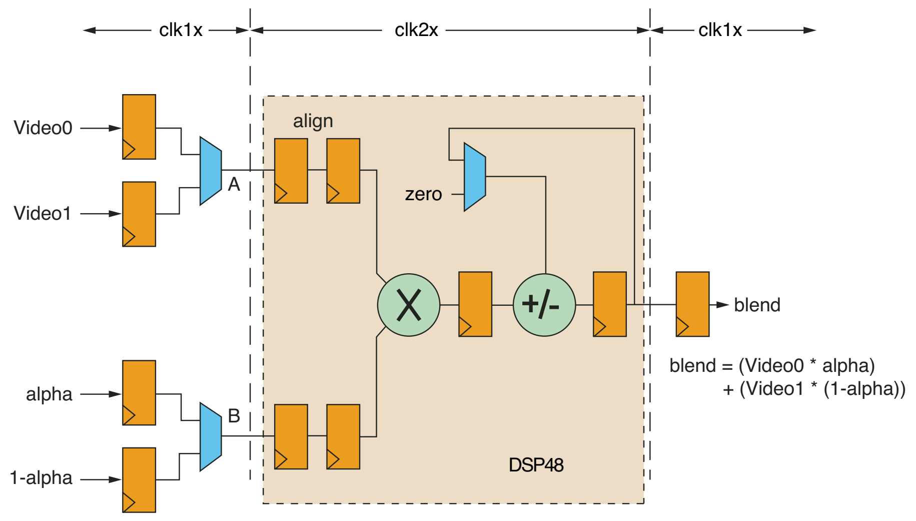


Figure 6 – Alpha blend implementation (one component)

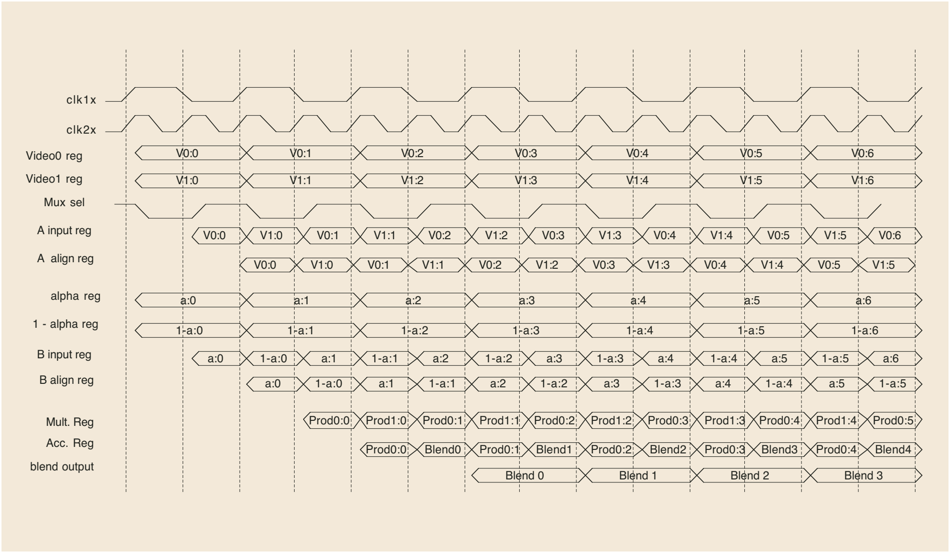

clk1x
clk2x
Video0 reg
Video1 reg
Mux sel
A input reg
A align reg
alpha reg
1 - alpha reg
B input reg
B align reg
Mult. Reg
Acc. Reg
blend output
V0:0 V0:1 V0:2 V0:3 V0:4 V0:5 V0:6
V1:0 V1:1 V1:2 V1:3 V1:4 V1:5 V1:6
a:0 a:1 a:2 a:3 a:4 a:5 a:6
1-a:0 1-a:1 1-a:2 1-a:3 1-a:4 1-a:5 1-a:6
Prod0:0 Prod1:0 Prod0:1 Prod1:1 Prod0:2 Prod1:2 Prod0:3 Prod1:3 Prod0:4 Prod1:4 Prod0:5
Prod0:0 Blend0 Prod0:1 Blend1 Prod0:2 Blend2 Prod0:3 Blend3 Prod0:4 Blend4
Blend 0 Blend 1 Blend 2 Blend 3

Figure 7 – Alpha blend timing

Conclusion

You can efficiently use the high-performance of Virtex-4 devices with DSP48 slices by processing multiple data streams in a time-multiplexed fashion. With careful design, a single DSP48 can perform multiply operations on two independent data streams, operating at 250 MHz each.

Alpha blending of video streams, as outlined in this article, is one example of processing two data streams through a single DSP48 slice. This capability complements the DSP features of Virtex-4 FPGAs – including built-in pipelining and cascading, integrated 48-bit accumulator, and an abundance of DSP48 slices in the SX family – to make Virtex-4 devices the ideal DSP platform.

For details about the DSP48 slice, refer to the "Virtex-4 FPGA Handbook," Chapter 10, or the "XtremeDSP Design Considerations User Guide" at *www.xilinx.com/bvdocs/userguides/ug073.pdf.*

Implementing Matrix Inversions in Fixed-Point Hardware

Our method is based on a synthesizable QR-decomposition MATLAB model and the AccelChip DSP Synthesis tool.

s_p + s_i + DetNoise;
ones(NSensors,1);
s*Wc; % broadside array output
blocking matrix
[eye(NSensors-1); zeros(1,NSensors-1)] -
[zeros(1,NSensors-1); eye(NSensors-1)];
* B; % interferer and noise only
for recursive LS
2.5
2

by Ramon Uribe
Sr. Principal IP Development Engineer
AccelChip Inc.
ramon.uribe@accelchip.com

Tom Cesear
Chief Scientist
AccelChip Inc.
tom.cesear@accelchip.com

Matrix inversion is an important operation in many state-of-the-art DSP algorithms and implementations, including radar, sonar, and multiple antenna systems for communications. A common component of these algorithms is a beamformer or spatial filter, whose function is to steer (in some optimal fashion) the response of an array of sensors for the reception of signal sources.

When using the least-squares (LS) criterion, the computation of optimum weights is based on the solution of a system of linear equations known as the deterministic normal equation. This is shown in the equation:

$$R_x w = p$$

Here, w is a vector of beamformer weights, which can be obtained with inversion of the correlation matrix R_x as shown in the equation:

$$w = R_x^{-1} p$$

From a numerical point of view, the best approach to matrix inversion is to not do it explicitly, whenever possible. Instead, it is better to solve the system of equations using an adequate solution technique.

Traditionally, implementations like this have been done with general-purpose DSP devices using floating-point arithmetic to minimize round-off error. A disadvantage of these implementations, however, is the limited processing power because of the small number of floating-point processing units commonly available per device. An appealing alternative for implementation is to use the Xilinx® Virtex™-4 FPGA family, which offers large amounts of parallelism. One complication with these silicon fabrics is that they are tailored for fixed-point arithmetic, and implementation in these is inherently challenging because of sensitivity to round-off error.

In this article, we'll present an efficient methodology that enables the implementation of algorithms involving matrix-inversion operations in hardware with fixed-point arithmetic. This methodology includes three essential steps to follow in the development process:

- Capturing the DSP algorithm description in the MATLAB language
- Definition of the fixed-point parameters directly coupled to the MATLAB algorithm description
- Automated generation of a hardware implementation that is bit-accurate to the fixed-point arithmetic model and that meets area/speed requirements for a particular application

Using this methodology, you can fully exploit the benefits of the processing power offered by implementations in FPGA or ASIC fixed-point hardware.

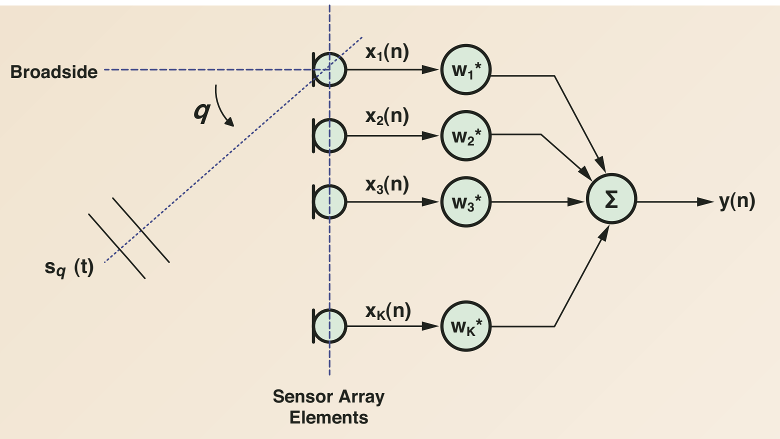


Figure 1 – Narrowband beamformer

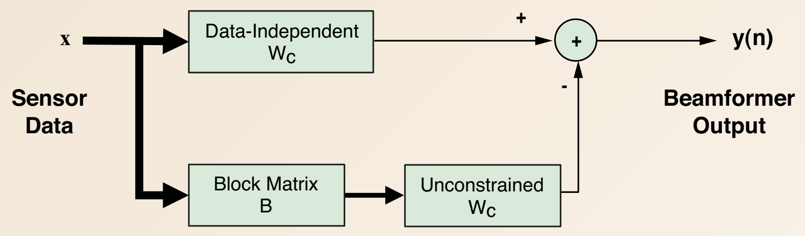


Figure 2 – Generalized sidelobe canceller (GSC)

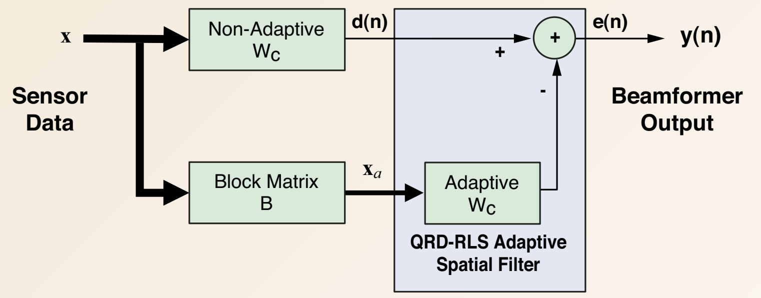


Figure 3 – Adaptive GSC beamformer

Beamforming and Matrix Inversion

Figure 1 shows a basic narrowband beamformer with K sensor elements arranged in a uniform linear array (ULA); this also shows a signal source $s_q(t)$ impinging on the array at an angle of incidence q. The K beamformer weights (w_1, w_2, ..., w_K) are used to linearly combine the array data observation samples ($x_1(n)$, $x_2(n)$, ..., $x_K(n)$). These are set to "steer" the response of the array for optimum reception. The output of the beamformer is the scalar y(n).

A generalized sidelobe canceller (GSC) is a special beamformer structure that allows the use of unconstrained optimization methods in the design of the optimum beamformer weights. The structure of the GSC is shown in Figure 2. To find the optimum weights w_a using the LS criterion, the following deterministic normal equation must be solved:

$$R_x w_a = b$$

Here, R_x is the correlation matrix of the input to the unconstrained section of the GSC and the vector b is the cross-correlation of the input X_a and the ideal response.

One effective technique for the solution of this equation is the recursive least-squares (RLS) approximation with QR decomposition of the input data matrix. This technique finds the solution without explicit inversion of a matrix and avoids constructing the correlation matrix, explicitly reducing the dynamic range requirements of signals involved in the computations.

Figure 3 shows the diagram of an adaptive GSC beamformer that uses a QRD-RLS algorithm for a recursive solution of the normal equation.

Example Beamformer Model for Hardware Synthesis

The GSC beamformer MATLAB model we used for the design includes the following features:

- A ULA array of four sensor elements
- A narrowband input signal of interest, impinging at an angle of 0°
- A narrowband interfering signal impinging at an angle of 10°, with the same amplitude as the signal of interest
- Uncorrelated white noise to model receiver noise at a level of -20 dB relative to the signal of interest

The GSC MATLAB model consists of three parts. A top-level script generates signals and displays results to analyze the performance the beamformer. The script invokes the QRD-RLS algorithm function in a streaming fashion to perform interference cancellation. Figure 4 shows an excerpt of this script.

The second part of the model is a synthesizable QRD-RLS algorithm function, qrd_rls_spatial(), which performs optimum cancellation of the interferer signal. This function is shown in Figure 5.

The last part of the GSC model is the synthesizable function that rotates arrays of values to perform orthogonal Givens rotations (givens_rotation). This rotation function can be automatically generated by the AccelChip DSP Synthesis tool as part of the Advanced Math Toolkit of synthesizable models. This function is shown in Figure 6.

The analysis and visualization of the performance of the GSC model is shown with plots in Figure 7. The signal of interest is shown on the top plot.

The middle plot shows the signal from the data-independent portion of the GSC. This signal clearly shows the distortion caused by the interferer signal impinging on the sensor array.

The bottom plot shows the output of the GSC after effective cancellation

```
...
% combine signal, interference and noise
s = s_p + s_i + DetNoise;
Wc = ones(NSensors,1);
d = s*Wc; % broadside array output
% blocking matrix
B = [eye(NSensors-1); zeros(1,NSensors-1)] -
- [zeros(1,NSensors-1); eye(NSensors-1)];
s_n = s * B; % interferer and noise only
% streaming loop for recursive LS
computation
for n = 1:NUM_ITER
   [e_rls(n)] =
qrd_rls_spatial(s_n(n,:),d(n));
end
...
```

Figure 4 – GSC MATLAB top-level script

```
function [e] = qrd_rls_spatial(xa,d)
% QRD-RLS for adaptive spatial filtering
M  = 3;
persistent R p
if isempty(R)
   R = zeros(1,M*M);
   R(1:M:end) = 0.01;
   p = zeros(1,M);
end
Ci = 1.0;
lambda = sqrt(0.99);
% update R matrix and p vector
for row = 1:M
   xvec = lambda*[R(1:M) p(1) Ci];
   yvec = [xa d 0];
   xvec_rot,yvec_rot]         =
givens_rotation(xvec,yvec);
   R = [R(M+1:end) xvec_rot(1:M)];
   xa = [yvec_rot(2:M) 0];
   p = [p(2:end) xvec_rot(M+1)];
   d = yvec_rot(M+1);
   Ci = xvec_rot(end);
end
e = Ci*d; % recursive error estimate
```

Figure 5 – Synthesizable QRD-RLS function

```
function [v, w] = givens_rotation( x, y )
% Givens rotation
r_sqr = x(1)^2 + y(1)^2;
r_inv = invsqrt_001(r_sqr);
sin_phi = y(1)*r_inv;
cos_phi = x(1)*r_inv;
v = x*cos_phi + y*sin_phi;
w = y*cos_phi - x*sin_phi;
```

Figure 6 – Givens rotation function

of the interferer done by the QRD-RLS algorithm function.

Figure 8 shows beampatterns of the GSC. The top plot shows the beampattern of the data-independent portion of the GSC. This shows that the interferer signal impinging at 10° suffers an attenuation of only 2 dB approximately relative to that of the desired signal at 0°; this small attenuation is what causes the distortion in the received signal from the broadside.

The middle plot shows the overall GSC beampattern. The improvement in the cancellation of the interfering signal can be seen with the larger attenuation at 10°. This is what accounts for the cancellation of the interferer signal obtained at the output of the GSC.

The bottom plot is a zoomed view of the overall GSC beampattern to highlight the attenuation achieved around 10°.

Definition of the Fixed-Point Model

The starting point of the design is the original "golden" reference MATLAB model of the GSC. The next step is to define a fully parameterized fixed-point arithmetic model. This model is directly coupled to the original MATLAB model to maintain lockstep with this golden reference. There are two critical aspects for efficiency in this step:

- The ability to intuitively associate fixed-point parameters with variables in the MATLAB algorithm description
- The ability to quickly evaluate the effects of the fixed-point arithmetic on the overall performance of the algorithm

Defining a fully parameterized fixed-point arithmetic model is an iterative process. In the case of the GSC with QRD-RLS, the numerical performance of the implicit matrix inversion operation is measured by the attenuation shown in the overall beampattern. We evaluated several input bit-widths with the intermediate variables sized to avoid overflows – the effect on the attenuation in the beampattern is shown in Figure 9.

We selected a 16-bit implementation that achieves an almost ideal interference rejection, as shown in Figure 9.

Generation of Hardware Implementation

The final step in our methodology is to generate the hardware implementation. There are two critical aspects to achieve efficiency:

- The ability to automatically generate an implementation that is bit-accurate against the fixed-point model of the DSP algorithm
- The ability to tailor the hardware architecture of the implementation to meet area/speed requirements

The generation of a suitable hardware implementation is also done iteratively to balance resource utilization and speed of operation. For the QRD-RLS algorithm, there are two points where the area/speed of the implementation can be affected:

- Controlling the degree of resource sharing of the givens_rotation function
- The rotation of row elements in the Givens rotation function can be achieved with different computation styles, including Newton-Raphson (using multipliers) and CORDIC (multiplier-less) micro-architectures

We performed iterations of this step exploring different combinations of resource utilization and speed of operation of the QRD-RLS function. The results of RTL synthesis, summarized in Table 1, are for a Xilinx Virtex-4 XC4VSX55 target device.

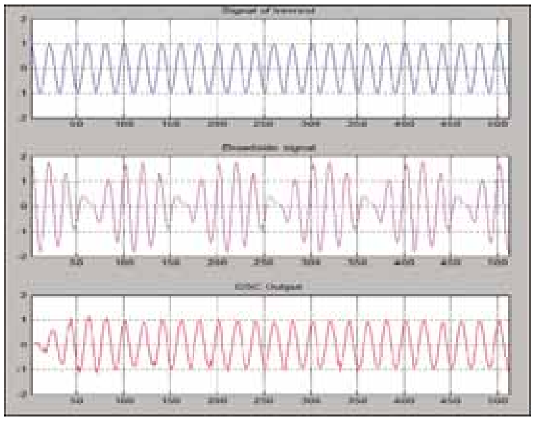


Figure 7 – GSC time-domain signals

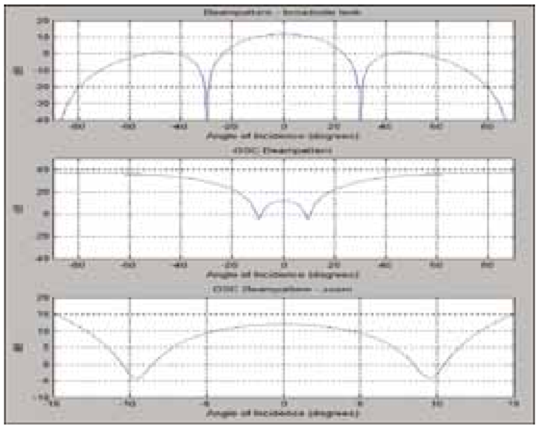


Figure 8 – GSC beampatterns

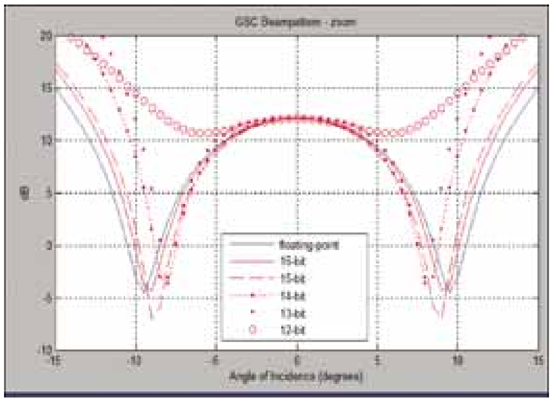


Figure 9 – GSC beampattern in fixed point

The results indicate a small decrease in resource utilization (LUTs) with sequential implementations. With a goal of maximum speed of operation and a minimum use of hardware multipliers, the CORDIC parallel implementation was picked for the place and route of the netlist from RTL synthesis. The results of the implementation using the Xilinx ISE™ software mapped to the same target device as during synthesis are shown in Table 2.

The hardware implementation results show that the QRD-RLS function can be implemented in 12% of the logic resources of a XC4VSX55 device with a sustainable data rate of 1.7 megasamples per second.

Conclusion

You can create an efficient hardware implementation of DSP algorithms in Xilinx FPGAs using matrix inversion operations with fixed-point arithmetic. The efficiency with which you can implement these algorithms is based on the use of the AccelChip DSP Synthesis tool to enable a high level of automation. The results show the effectiveness of this methodology in the implementation of a challenging algorithm in fixed-point arithmetic hardware.

To get more information about AccelChip's solutions, visit *www.accelchip.com*. For a list of other technical papers from AccelChip, visit *www.accelchip.com/papers*.

This article is based on the paper, "Efficient Methodology for Implementation of Matrix Inversion in Fixed-Point Hardware," presented at the 2005 GSPx Conference.

Metric	Multiplier Parallel	Multiplier Sequential	CORDIC Parallel	CORDIC Sequential
LUTs	5%	4%	10%	9%
DSP48s	51	19	1	1
Sustainable Data Rate	1.9 MSPS	0.8 MSPS	1.8 MSPS	0.18 MSPS

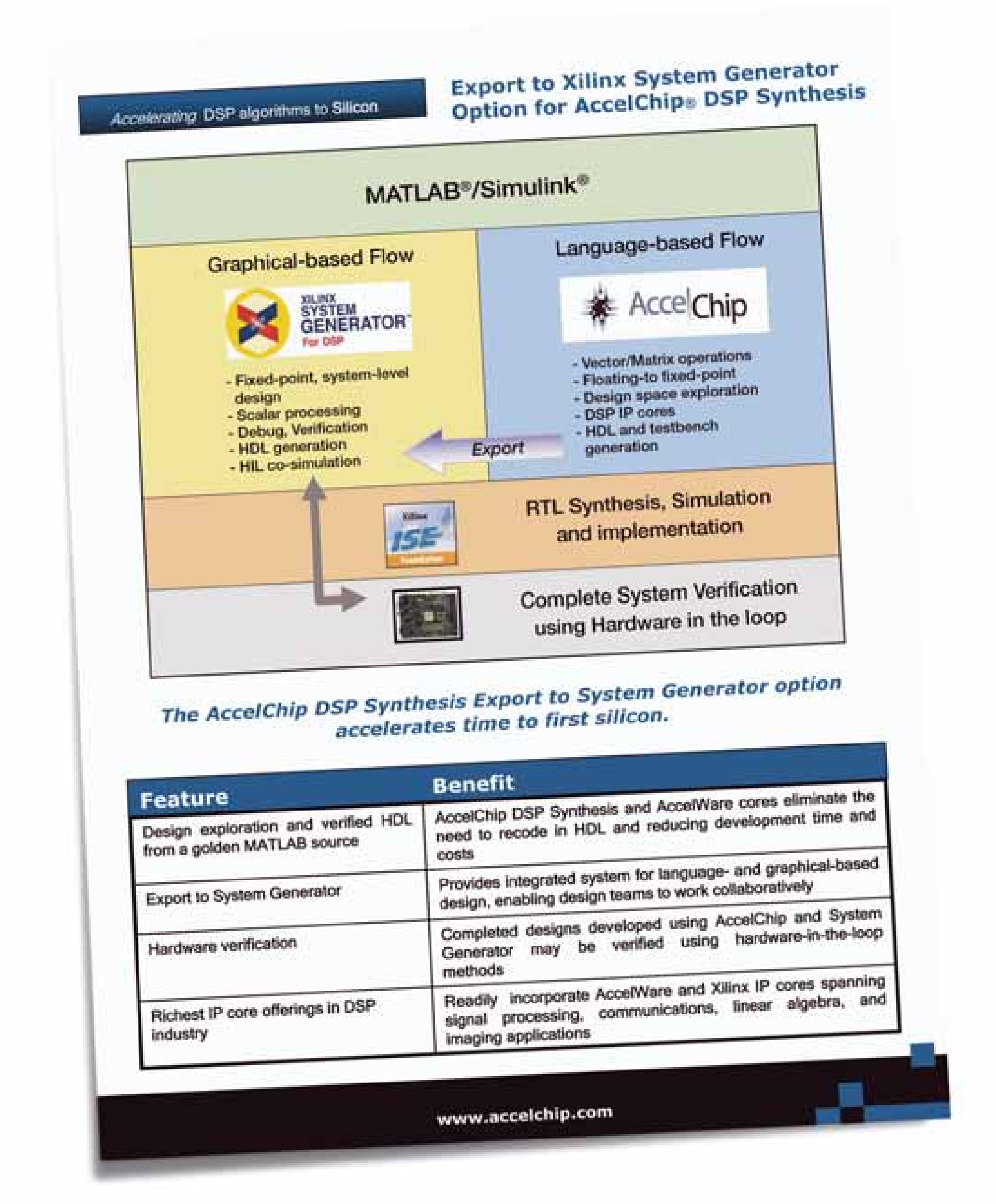
Table 1 – RTL synthesis results

Occupied Slices	3076 (12%)
DSP48s	1
Sustainable Data Rate	1.7 MSPS

Table 2 – Implementation results

STOP OVER-THE-WALL ENGINEERING.

(FINALLY, ALGORITHM DEVELOPERS, DSP SYSTEM ARCHITECTS AND HARDWARE DESIGNERS ARE ALL ON THE SAME PAGE)



Accelerating DSP algorithms to Silicon

Export to Xilinx System Generator Option for AccelChip® DSP Synthesis

MATLAB®/Simulink®

Graphical-based Flow

XILINX SYSTEM GENERATOR™ For DSP

- Fixed-point, system-level design
- Scalar processing
- Debug, Verification
- HDL generation
- HIL co-simulation

Language-based Flow

AccelChip

- Vector/Matrix operations
- Floating-to fixed-point
- Design space exploration
- DSP IP cores
- HDL and testbench generation

Export

RTL Synthesis, Simulation and implementation

Complete System Verification using Hardware in the loop

The AccelChip DSP Synthesis Export to System Generator option accelerates time to first silicon.

Feature	Benefit
Design exploration and verified HDL from a golden MATLAB source	AccelChip DSP Synthesis and AccelWare cores eliminate the need to recode in HDL and reducing development time and costs
Export to System Generator	Provides integrated system for language- and graphical-based design, enabling design teams to work collaboratively
Hardware verification	Completed designs developed using AccelChip and System Generator may be verified using hardware-in-the-loop methods
Richest IP core offerings in DSP industry	Readily incorporate AccelWare and Xilinx IP cores spanning signal processing, communications, linear algebra, and imaging applications

www.accelchip.com

Company partnerships don't usually generate a lot of interest, but a partnership that yields team-building technology does. For example, the industry's first DSP design flow from a mixed MATLAB/Simulink environment to a verified FPGA. It's based on a tight integration between AccelChip DSP Synthesis and Xilinx System Generator for DSP that accelerates development of signal processing systems. Work more collaboratively and get designs to market faster by eliminating the need to recapture in HDL. Readily incorporate AccelWare and Xilinx IP cores and verify the entire system using hardware-in-the-loop simulation, further speeding development. How's that for teamwork? **Stop your Over-the-Wall engineering. Download free whitepapers at accelchip.com/papers or call (408) 943-0700.**

XILINX® AccelChip

Integrating MATLAB Algorithms into FPGA Designs

The integration of AccelChip DSP Synthesis and Xilinx System Generator for DSP provides a seamless path from MATLAB/Simulink to verified FPGA-based DSP systems.

by Eric Cigan
Product Marketing Manager
AccelChip Inc.
eric.cigan@accelchip.com

Narinder Lall
Senior DSP Marketing Manager
Xilinx, Inc.
narinder.lall@xilinx.com

There are two kinds of people in electronic design: those who think in terms of words and those who think in terms of pictures. This dichotomy is most apparent in the world of DSP. Some designers prefer to develop algorithms in language form, while others choose to draw out block diagrams showing data flows. Until now, different sets of tools were required for each method – but why should you have to choose?

Xilinx® System Generator for DSP is well established as a productive tool for creating DSP designs using graphical methods. With a visual programming environment that leverages The MathWorks Simulink tool and its predefined Xilinx block set of DSP functions, System Generator meets the needs of both system architects (to integrate design components) and hardware designers (to optimize implementations). (For more details, see "Implementing DSP Algorithms in FPGAs" in the Winter 2004 issue of the *Xcell Journal*.)

Many DSP algorithm developers have found that the MATLAB language best meets their preferred development style. With more than 1,000 built-in functions, as well as toolbox extensions for signal processing, communications, and wavelet processing, MATLAB offers a rich environment for algorithm development and debugging.

In addition to the IP functions provided in MATLAB, the MATLAB language is uniquely adept with vector- and array-based waveform data at the core of algorithms in applications such as wireless communications, radar/infrared tracking, and image processing.

The AccelChip DSP Synthesis tool was developed specifically for algorithm developers and DSP architects who have embraced a language-based flow. With AccelChip, you begin with MATLAB M-files to perform stimulus creation, algorithm evaluation, and post-processing. You can load the M-files into the AccelChip tool; they become the "golden source" for a design flow that ultimately produces optimized implementations in Xilinx FPGAs.

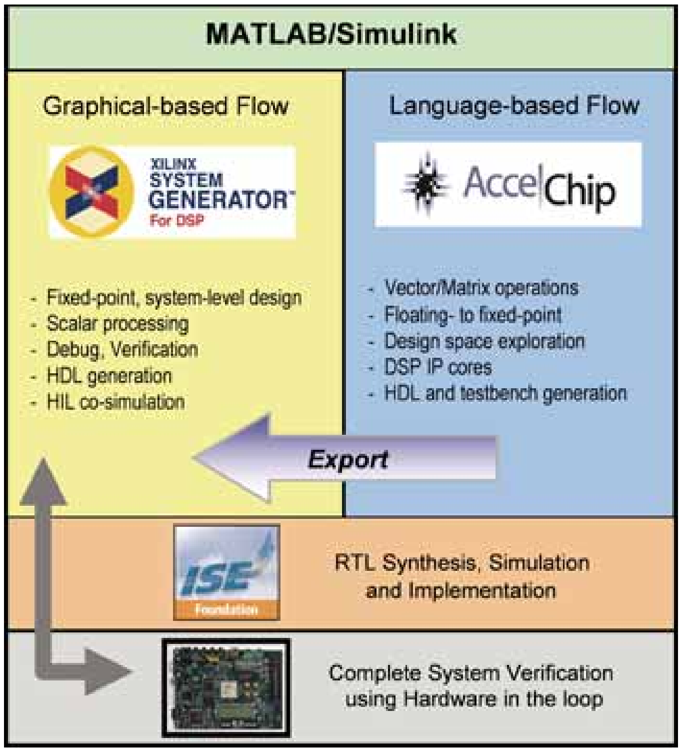

MATLAB/Simulink
Graphical-based Flow
XILINX SYSTEM GENERATOR For DSP
- Fixed-point, system-level design
- Scalar processing
- Debug, Verification
- HDL generation
- HIL co-simulation
Language-based Flow
AccelChip
- Vector/Matrix operations
- Floating- to fixed-point
- Design space exploration
- DSP IP cores
- HDL and testbench generation
Export
ISE Foundation
RTL Synthesis, Simulation and Implementation
Complete System Verification using Hardware in the loop

Figure 1 – System Generator/AccelChip interface

Unifying Words and Pictures

In the past, design teams looked on System Generator and AccelChip DSP Synthesis as mutually exclusive design tool options, but wished for access to the best aspects of each tool. In response, AccelChip and the DSP division at Xilinx collaborated in an effort to combine AccelChip's tools with System Generator. The result allows you to mix language-based algorithm design in MATLAB and graphical block-oriented design in Simulink in a novel unified electronic-system-level (ESL) design environment (Figure 1).

AccelChip's DSP Synthesis tool augments System Generator by providing a seamless integration path for algorithm developers, enabling the rapid creation of IP blocks, directly from M-files, that enhance the Xilinx block set in System Generator. In addition, AccelChip has optional AccelWare toolkits that complement System Generator with additional IP cores optimized for Xilinx cores. AccelWare toolkits include mathematical building blocks, signal processing, communications, and advanced math to implement linear algebra functions.

Kalman Filter Design Example

To illustrate this approach, let's take an advanced algorithm written in MATLAB, use AccelChip to synthesize the design, and then integrate it into a System Generator model. Our example is a Kalman filter – a recursive, adaptive filter well-suited to combining multiple noisy signals into a clearer signal (for details on the topic, see Arthur Gelb's book, "Applied Optimal Estimation").

Kalman filters embed a mathematical model of an object – such as a commercial aircraft being tracked by ground-based radar – and use the model to predict future behavior. Kalman filters then use measured signals (such as the signature of the aircraft returned to the radar receiver) to periodically correct the prediction.

```
function [S] = simple_kalman(A)
    DIM = size(A,2);
    persistent p P_cap
    if isempty(P_cap)
        P_cap = [8 0 0; 0 8 0; 0 0 8];
        p     = ones(DIM,1)/2;
    end;
    I = eye(DIM);
    R = [128 0 0;0 128 0; 0 0 128];

    % estimate step:
    %p_est     = p;
    P_cap_est = P_cap+I;

    % correction step:
    K     = P_cap_est * inv(P_cap_est+R);
    p     = p + K * ( A' - p );
    P_cap = (I - K)*P_cap_est;
    S     = p';
```

Figure 2 – Kalman filter example M-file

Figure 2 shows the MATLAB M-file describing the Kalman filter. The algorithm defines matrices R and I that describe the statistics of the measured signal and the predicted behavior. The last nine lines of the algorithm are the code that predicts and corrects the estimate.

This algorithm illustrates the flexibility and conciseness of the MATLAB language. Common operators such as addition and subtraction operate on variables like the two-dimensional arrays A or P_cap without having to write loops, as you would in languages like C. Multiplication of two-dimensional arrays is automatically performed as matrix multiplication without any special annotation. MATLAB operators such as matrix transposition allow the MATLAB code to be compact and easily readable. And complex operations like matrix inversion are completed using MATLAB's extensive linear algebra capabilities. Although such an algorithm could be constructed as a block diagram, doing so would obscure the algorithm structure so readily apparent in MATLAB.

With AccelChip, a first step in synthesizing a complete algorithm is to generate any major cores that are referenced – in this case, the matrix inverse indicated by the function call inv(P_cap_est+R). But you can implement a matrix inverse in many ways; the choice of which method to use depends on the size, structure, and values of the matrix.

Using the matrix inverse IP core from the AccelWare toolkit, you can choose from micro-architectures designed for different applications. These micro-architectures can be optimized for speed, area, power, or noise. In this case, the most suitable approach is to use the AccelWare QR matrix inverse core.

Synthesizing RTL with AccelChip

With the MATLAB M-file loaded into AccelChip, the next step is to simulate the floating-point design to establish a baseline. You would then use AccelChip to convert the design to fixed-point math, verifying it in MATLAB as shown in Figure 3. AccelChip offers an array of tools to help you trim bits from the design and verify fixed-point design effects like saturation and rounding. AccelChip aids in this process by propagating bit growth throughout the design and letting you use directives to set constraints on bit width. This algorithmic design exploration allows you to attain the ideal quantization that minimizes bit widths while managing overflows or underflows, allowing early trade-offs of silicon area versus performance metrics.

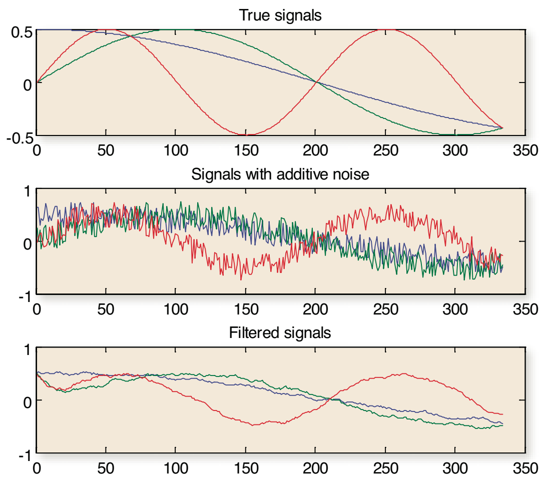


Figure 3 – The Kalman filter constructs estimates of the three signals based on input signals corrupted by noise.

Once you have attained suitable quantization, the next step is to generate RTL for your target Xilinx device. At this point, you can use the AccelChip GUI to set constraints on the design using the following design directives to achieve further optimizations:

- Rolling/unrolling of FOR loops
- Expansion of vector and matrix additions and multiplications
- RAM/ROM memory mapping of vectors and two-dimensional arrays
- Pipeline insertion
- Shift-register mapping

Using these directives constitutes hardware-based design exploration, allowing the design team to further improve quality of results. In synthesizing the RTL, AccelChip evaluates the entire design and schedules the entire algorithm, performing necessary boundary optimization in the process.

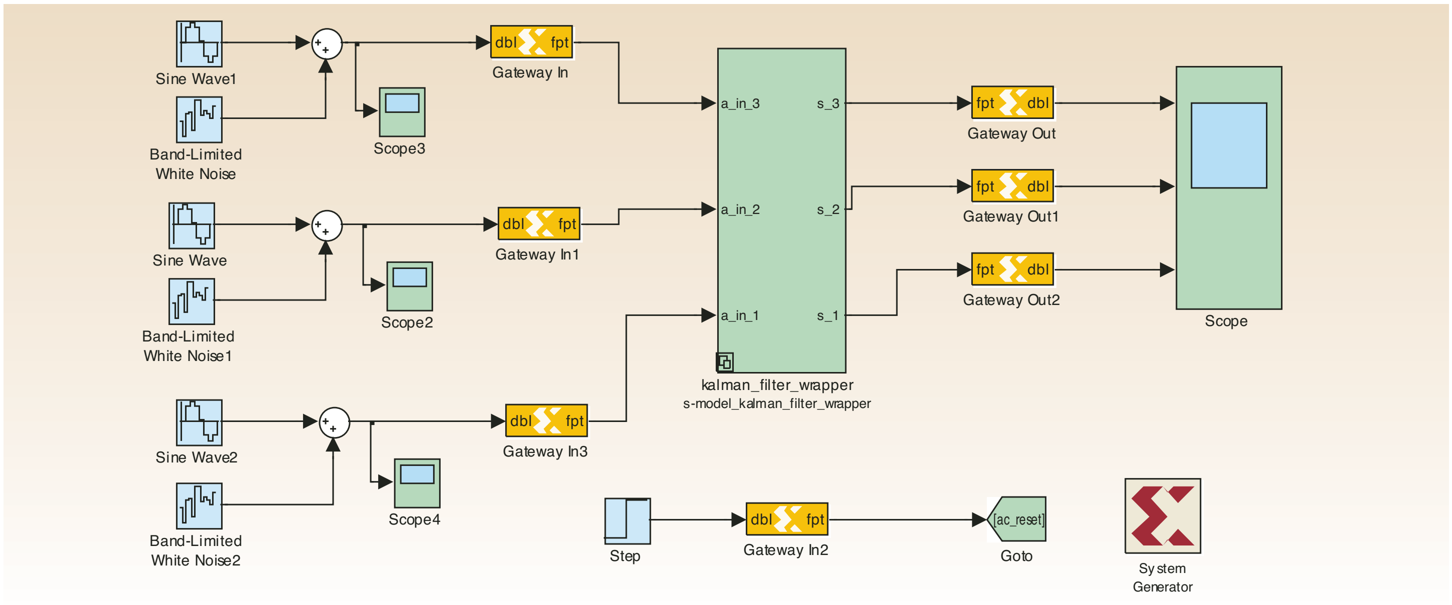


Figure 4 – AccelChip exports a cycle-accurate System Generator block that supports both simulation and RTL code generation.

Throughout this flow, AccelChip maintains a uniform verification environment through a self-checking test bench; the input/output vectors that were generated when verifying the fixed-point MATLAB design are used to verify the generated RTL. The RTL verification step also gives AccelChip the information necessary to compute the throughput and latency of the Kalman filter. This is essential information to assess whether the design meets specifications and is critical for achieving cycle-accurate simulation.

Exporting from AccelChip to System Generator

Although RTL verification is a key step in the design flow, designers want to see algorithms running in hardware. System Generator's hardware-in-the-loop co-simulation interfaces make this a push-button flow, allowing you to bring the full power of MATLAB and Simulink analysis functions to hardware verification.

Now that you have run RTL verification in AccelChip, you are ready to export the AccelChip design to System Generator by going to the "Export" pull-down menu in the AccelChip GUI and selecting "System Generator." AccelChip then generates a cycle-accurate System Generator block that supports both simulation and RTL code generation.

At this point, you transition the design flow to System Generator, where a new block for the Kalman filter is available in the Simulink library browser. You need only select the Kalman filter block and drag it into the destination model to incorporate the AccelChip-generated Kalman filter into a System Generator design.

Figure 4 shows the resulting diagram for the Kalman filter. In the center of the System Generator diagram is the Kalman filter, and around it are the gateway blocks needed for a System Generator design.

Once the AccelChip-generated block is included in the System Generator design, you can perform a complete, system-level simulation of cycle-accurate, bit-true models to verify that the system meets specifications. You can use AccelChip-generated blocks for System Generator in conjunction with the Xilinx block set. Once this system-level verification step is completed, the next step in the System Generator flow is to move on to design implementation. The "Generate" step in System Generator compiles the design into hardware.

Verification Options

All design files generated by AccelChip, including exported System Generator files, are verified back to the original "golden" source MATLAB M-file. AccelChip's verification approach is based on the generation of a test bench from the MATLAB source – this test bench is applied at the RTL level within AccelChip and can be applied in System Generator to verify the correctness of the design.

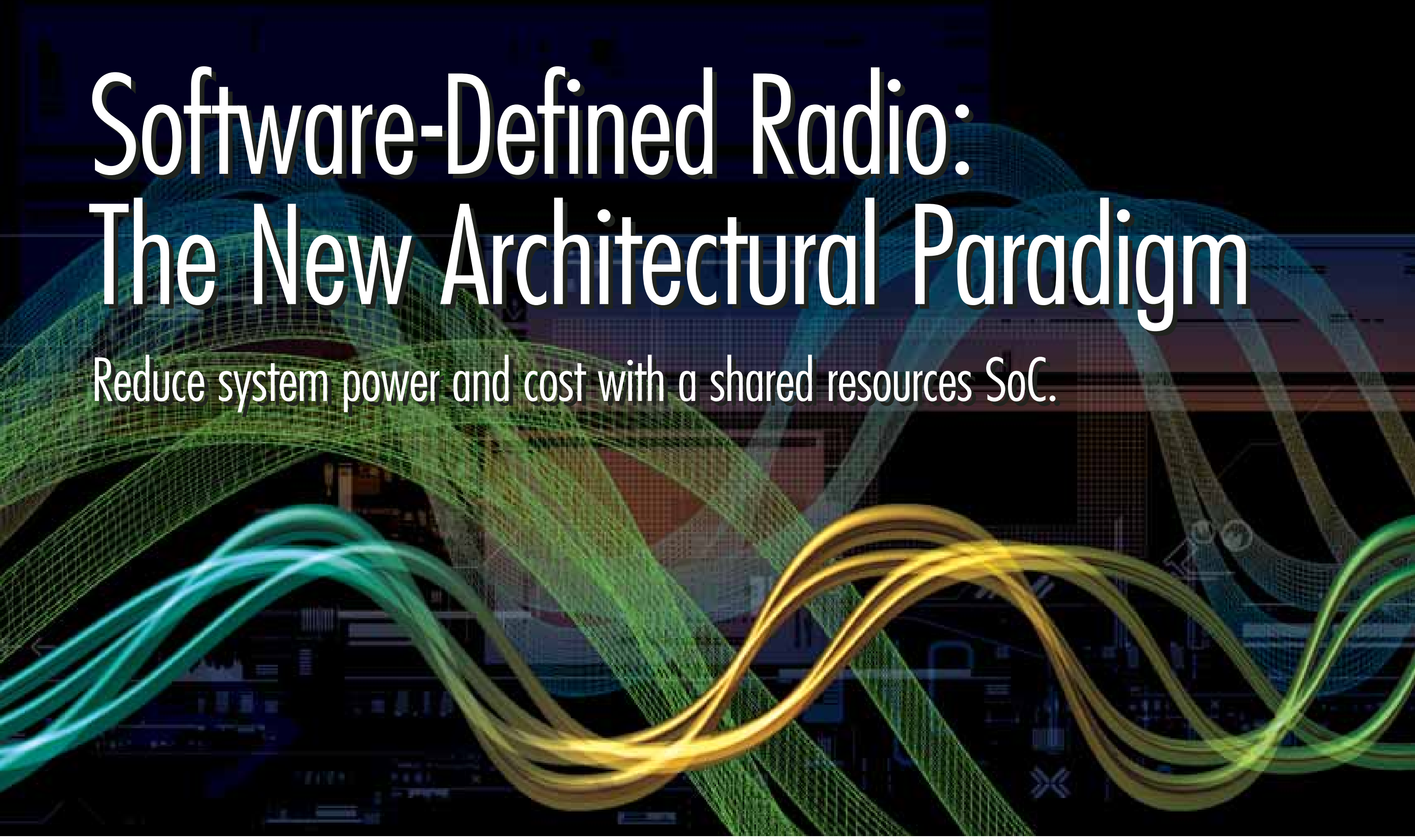
Once verified in the System Generator environment, you can verify the AccelChip-generated block using System Generator's supported methods – including HDL co-simulation and hardware-in-the-loop – to accelerate hardware-level simulation 10 to 100 times.

Conclusion

The integration of AccelChip with Xilinx System Generator is the first solution to unite MATLAB-based algorithmic synthesis favored by algorithm developers with the graphical design flow used by system architects and hardware designers. It uses the rich MATLAB language and its companion toolboxes to create System Generator IP blocks of complex DSP algorithms.

By using these tools together, design teams can employ the most productive means of modeling hardware for implementation, fully involving algorithm developers in the FPGA design process and completing higher quality designs more quickly.

For more information on AccelChip and its interface to Xilinx System Generator, visit *www.accelchip.com*. For more information on System Generator, visit *www.xilinx.com/products/design_resources/dsp_central/grouping/index.htm*.



Software-Defined Radio: The New Architectural Paradigm

Reduce system power and cost with a shared resources SoC.

by Manuel Uhm
Senior Marketing Manager
Xilinx, Inc.
manuel.uhm@xilinx.com

Software-defined radios (SDRs) have already become a reality in the defense industry through programs such as the Joint Tactical Radio System (JTRS). Because SDRs are already being deployed, why would a new architectural paradigm be of interest? The reason is simple: the power consumption and cost of first-generation SDRs is generally too high for widespread deployment.

Despite Moore's Law, incremental process improvements are not sufficient to reach the power and cost restrictions for small-form-factor, handheld, and manpack radios, such as those required for JTRS Cluster 5. In addition, architectural changes are required to reach the strenuous requirements.

Using partially reconfigurable platform FPGAs as an SDR system-on-chip (SoC) addresses both of these issues by decreasing the number of DSP components in an SDR while still providing the necessary functionality. In this article, I'll discuss how to apply this revolutionary technology to an SDR black-side modem.

Architectural Improvement #1: SCA-Enabled SDR SoC

Current SDR modem architectures, such as that of JTRS Cluster 1, utilize a discrete general-purpose processor (GPP) to manage the application and control infrastructure, known as the Software Communications Architecture (SCA) Operating Environment (OE). The GPP is coupled with an FPGA for channelization and wideband waveform processing and a DSP for narrowband waveform processing.

The SCA OE comprises a POSIX-compliant real-time operating system (RTOS) for scheduling and memory protection, a CORBA (Common Object Request Broker Architecture) ORB for message passing, and an SCA Core Framework (CF) for loading and tearing down waveforms.

With the availability of platform FPGAs such as Xilinx® Virtex™-II Pro and Virtex-4 FX devices that incorporate a hard-core PowerPC™ 405 GPP, it is now possible to run the entire SCA OE in the same FPGA that is required in the modem for channelization and wideband waveform processing. Because the 405 has a memory management unit (MMU), it is possible to run a full RTOS and ensure memory protection, unlike a soft-core GPP. This results in an SCA-enabled SDR SoC.

I/O accounts for the most power consumption in an SDR modem; thus it is important to reduce the I/O count as much as possible. Removing the discrete GPP from the modem does just this, thereby having the primary benefit of reducing the power consumption of the modem. Furthermore, the 405 hard-core is power-efficient, providing 600 DMIPS at 0.4W.

Secondary benefits include potentially lower cost by the removal of the discrete GPP and a reduced signal processing footprint, which can result in a smaller form factor or better thermal dissipation through superior board layout.

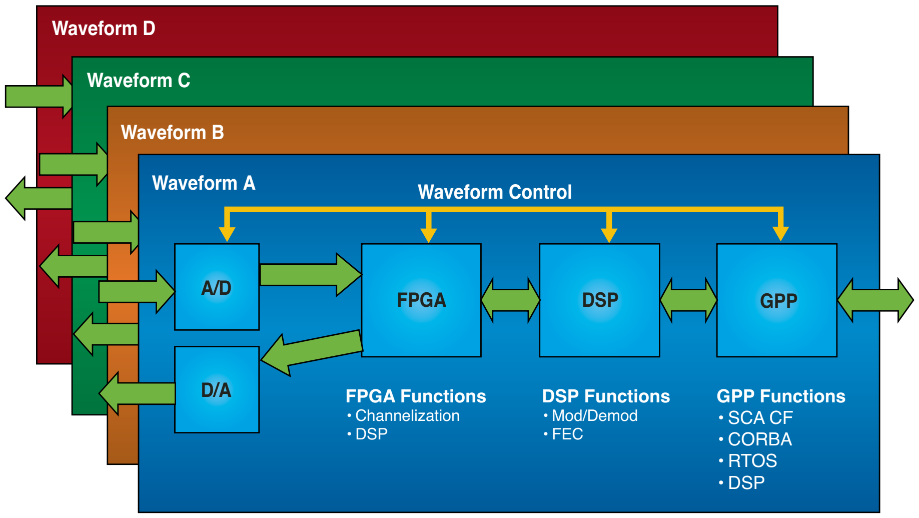


Figure 1 – Current SCA-enabled SDR modems use a dedicated set of signal processing hardware for each channel. The more channels the SDR must support, the more hardware it contains. This has a direct impact on power consumption and cost.

Architectural Improvement #2: Shared Resources

The current architecture for implementing an SDR modem is known as a dedicated resources model. It is called dedicated resources because a set of processing resources is dedicated to a radio channel (where each channel is capable of running a waveform, such as the Single Channel Ground and Airborne Radio System [SINCGARS]). In this case, the processing resources consist of an A/D, D/A, FPGA, DSP, and GPP. To implement an N-channel radio, N sets of processing resources are required. This is illustrated in Figure 1 for a four-channel SDR modem supporting an SCA CF.

From a functional perspective, this architecture is sufficient for radios with less power and size-constrained environments, such as Cluster 1 vehicular radios. However, it is an inefficient usage of the available processing resources, resulting in excess power consumption and cost. For example, the signal processing parts for all channels of the radio must be selected for the worst-case scenario; the processing resources must be able to support the largest waveform (WNW, the Wideband Networking Waveform) such that if only a small narrowband waveform like SINCGARS is instantiated, most of that channel's processing resources are not utilized. This has a significant impact on driving up the cost of the modem. Obviously, the problem gets exacerbated as you scale the model further. JTRS AMF (the JTRS Cluster for Airborne, Maritime, and Fixed installations) requires some radios to support eight channels. This also has an impact on power consumption.

A more efficient architecture for an SDR modem is referred to as a shared resources model. Unlike a dedicated resources model, this architecture offers the capability to support multiple waveforms across a single set of processing resources, allowing for much more efficient usage of the resources. The number of waveforms that can be supported is a function of the size of the waveform and the size of the available processing resources. Figure 2 illustrates how a multi-channel SCA-enabled SDR modem could be implemented using this architecture. In this instance, the signal processing part count has decreased from 20 to 4 components. Hence, implementation of these architectural advantages can result in a production cost and power consumption that is two to three times lower than the dedicated resources model.

You will also notice that the FPGA is capable of doing all the heavy digital signal processing. The embedded GPP is also a natural fit for the light signal processing, such as synchronization loop control, as well as the upper protocol layers such as link and network layers.

It is also worth noting that both architectures illustrated here are using 100% commercially available components. In the shared resources model, the SoC FPGA can be a mid- to large-sized Virtex-II Pro or Virtex-4 FX FPGA with an embedded PowerPC core.

Partial Reconfiguration: The Enabling Technology

The technology that enables the shared resources model is partial reconfiguration of the FPGA. Partial reconfiguration enables an application or component, such as a waveform or waveform component, to

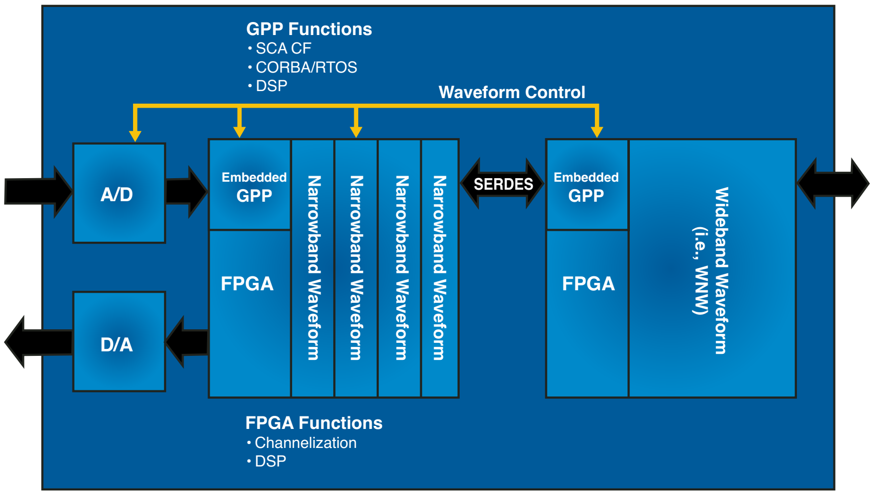


Figure 2 – A multi-channel SCA-enabled SDR modem architecture using a shared resources model. In this example, 5 channels supporting 1 wideband waveform and 4 narrowband waveforms have been implemented, while reducing the signal processing part count from 20 to 4 components compared to a dedicated resources model.

be dynamically loaded or unloaded in a portion of the device while other portions are either being used by other applications or going unused. This allows support for multiple independent applications concurrently in a single FPGA, which is somewhat analogous to dynamic task switching of a GPP. Without this capability, it would be necessary to reconfigure the entire FPGA to support a different application, which would result in the loss of all previous applications.

For example, if an FPGA was configured to support a SATURN comm link, it would have to be fully reconfigured to support an HAVEQUICK comm link, therefore resulting in the loss of the SATURN link regardless of how much leftover logic there was in the FPGA. Clearly this is unacceptable for a radio. Furthermore, partial reconfiguration enables an adaptive waveform to be supported in a smaller FPGA because the FPGA can dynamically load and switch between waveform components, rather than having all possible waveform components loaded at runtime.

Three basic elements are required to support partial reconfiguration in an FPGA:

1. An FPGA that inherently supports partial reconfiguration, such as the Xilinx Virtex family. The Virtex family is frame-reconfigurable, meaning that individual frames of logic within the device can be dynamically reconfigured independent of the rest of the frames. In Virtex-II Pro devices, a frame consists of a column, while in Virtex-4 FPGAs, a frame consists of a 16 x 1 configurable logic block (CLB) "tile."

2. At least a basic controller must be available to dynamically manage the reconfiguration of the FPGA. This could be an embedded GPP, a soft-core GPP (such as the Xilinx MicroBlaze™ processor core), or an external GPP connected to the FPGA. In the shared resources model example, the same embedded GPP that is running the SCA OE is also managing the partial reconfiguration of the FPGAs.

3. Partial reconfiguration software development tools that support the development of applications, restricted to boundaries complying with the hardware architecture of the FPGA.

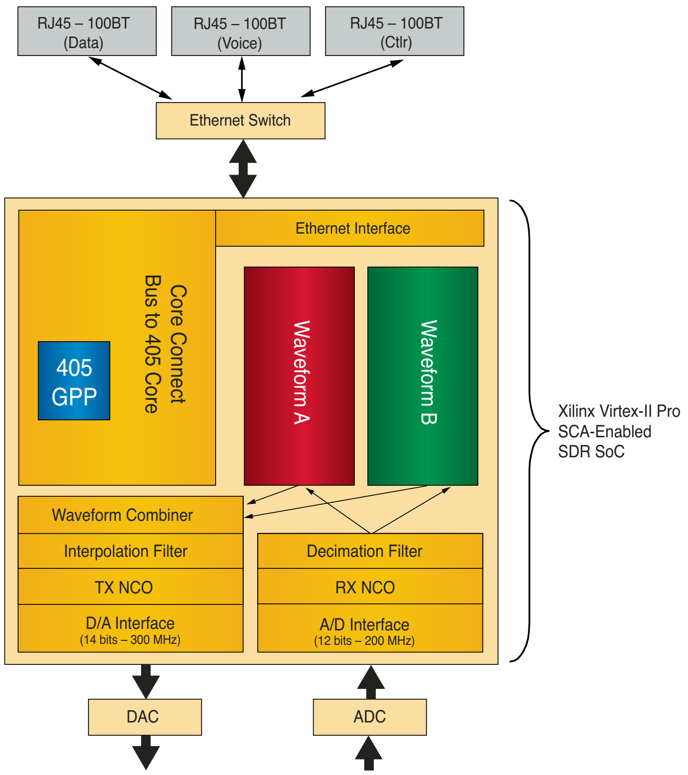


Figure 3 – Floorplan of a Xilinx Virtex-II Pro-based SCA-enabled SDR SoC suporting shared resources through partial reconfiguration. The yellow areas represent static infrastructure and do not change from waveform to waveform, whereas the rest of the FPGA is available for partially reconfigured applications.

Although elements #1 and #2 have been around for some time, it is only recently that software development tools enabling partial reconfiguration design have become available. Standard tools from Xilinx are available by request for Virtex-II Pro today and for Virtex-4 in the fourth quarter of 2005.

Implementing the Architecture

The shared resources model using an SCA-enabled SoC has been proven to work today in a COTS (commercial-off-the-shelf) Xilinx Virtex-II Pro-based SDR modem from ISR Technologies. The demonstration system uses two modems, each supporting two independent applications: a narrowband, 256 Kbps waveform supporting a comm link between two VoIP phones; and a wideband, 1,024 Kbps waveform supporting a streaming video link between two laptops.

Using a COTS SCA CF from the Communications Research Centre, the video link can be instantiated and torn down while maintaining the comm link – and vice versa.

More details on the demonstration can be found in the December 2004 JTRS JPO Technology Awareness Bulletin, published by the JTRS Joint Program Office at *http://jtrs.army.mil/sections/technicalinformation/fset_technical.html.*

Figure 3 illustrates the floorplan of the Virtex-II Pro device in each of the modems. The yellow areas represent static infrastructure, as they do not change regardless of the waveforms being supported. This includes the digital down and up converter, internal shared buses (the Core Connect bus for the embedded PowerPC 405) and the interfaces to external devices, such as the A/D and D/A. The two applications (waveform A and B) run independently in the partially reconfigurable region in the right-hand side of the device. If necessary, a larger waveform or more smaller waveforms could run in the same space.

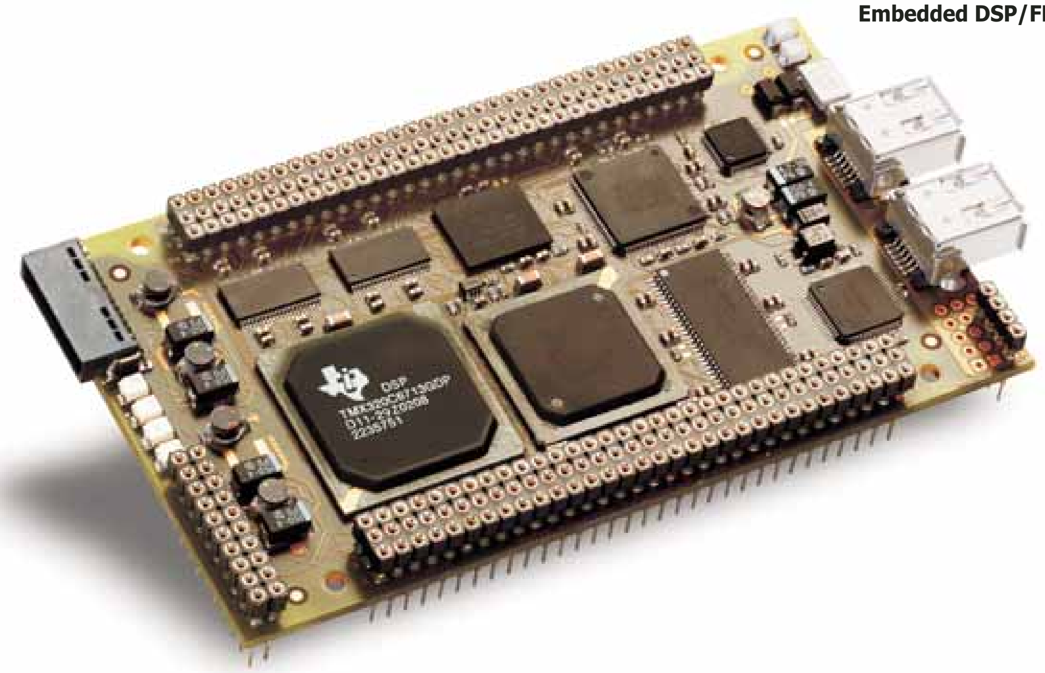
Conclusion

Power consumption and cost are issues that are preventing widespread deployment of SDRs today, particularly in size, weight, power, and cost-constrained environments. Architectures that incorporate SCA-enabled SDR SoCs and a shared resources model can help to address these issues by providing the most efficient SDR modem implementation, thereby driving down the power and cost.

TRAQUAIR
www.traquair.com

The Compact DSP & FPGA Solution

micro-line C6713Compact
Embedded DSP/FPGA board



Virtex-II FPGA
250k, 500k or 1MGates

TMS320C6713 DSP
Up to 2400MIPS/1800 MFLOPS

IEEE 1394 FireWire
400MBit/sec Communications
IIDC DCAM Video Framecaputre

Optional Data Storge
HD or Compact FLASH
FAT32 Filesystem Support

Optional Ethernet
10/100BaseT,
TCP/IP, UDP, ICMP, IGMP, Telnet,
HTTP, SMTP, POP3, FTP ,
Embedded Web Server

Optional Analog I/O
12/14/16-bit multi-channel A/D/A
via FPGA I/O Pins or DSP EMIF

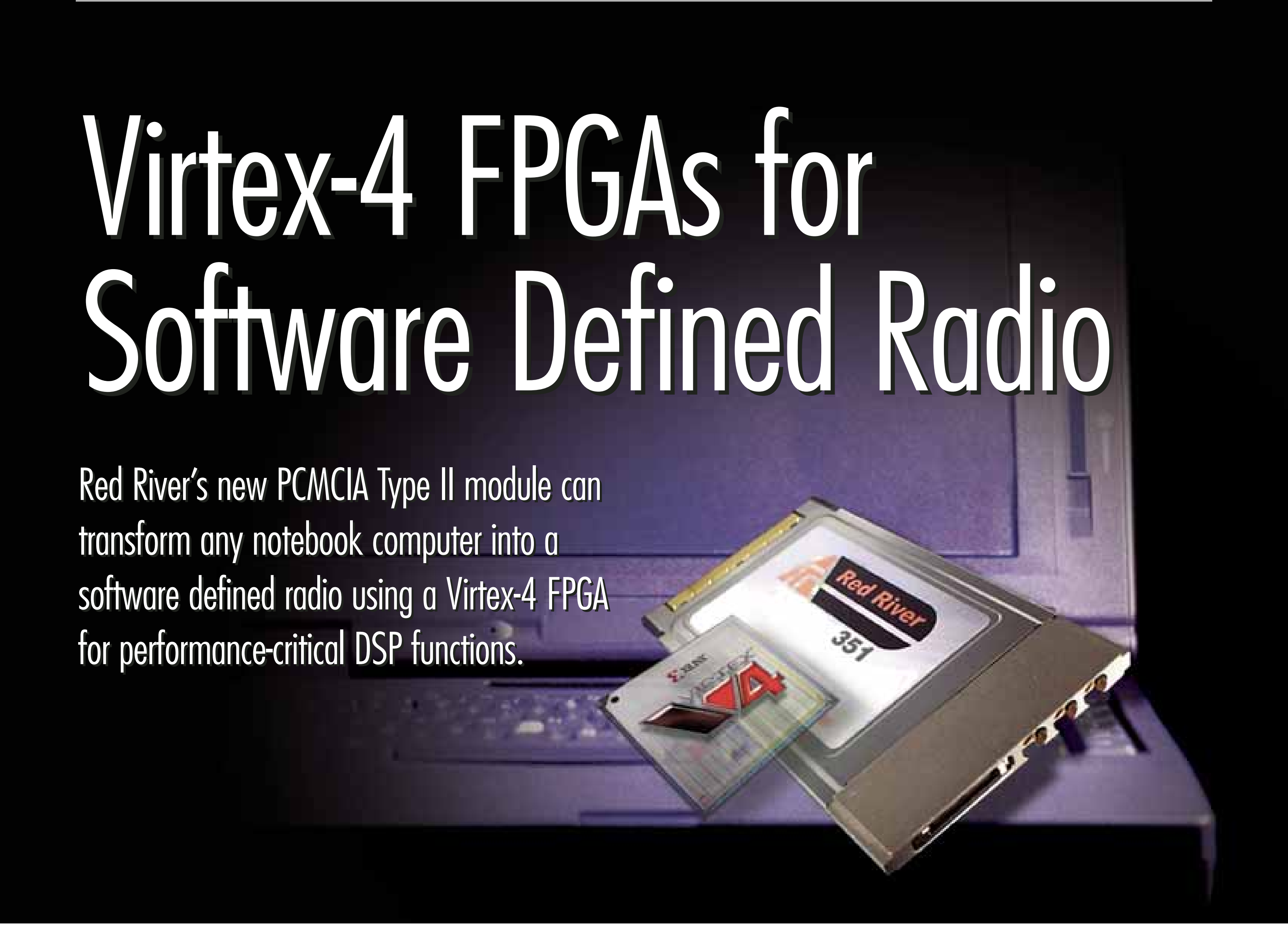
The micro-line C6713Compact is a high performance single board DSP/FPGA solution, offering exceptional capabilities and flexibility.

Measuring only 67 x 120mm, it combines a Xilinx Virtex-II FPGA with Texas Instruments' most powerful floating point DSP processor, the TMS320C6713, as well as with up to 64MBytes of SDRAM, 8MBytes of FLASH ROM, FireWire, and optional Ethernet communications and analog I/O.

It is suitable for stand-alone operation or as a mezzanine daughter card, and has extensive digital I/O capabilities for easy integration with the on-board FPGA, DSP and FireWire resources.

Traquair Data Systems, Inc. Tel: 607-266-6000 Email: sales@traquair.com Web: www.traquair.com

Virtex-4 FPGAs for Software Defined Radio



Red River's new PCMCIA Type II module can transform any notebook computer into a software defined radio using a Virtex-4 FPGA for performance-critical DSP functions.

by Ken Sienski
President
Red River
sienski@red-river.com

Established in 1996, Red River specializes in high-performance signal processing and data communication solutions for the embedded systems market, especially software defined radio applications.

Our main challenge in serving the software defined radio market is to have a hardware platform that meets the demands of multiple configurations. Some customers are looking for a complete, pre-built radio solution; others are looking to add custom features to a radio platform. These disparate requirements place great demands on us to find a common programmable silicon solution that meets both needs.

The Xilinx® Virtex-4™ FPGA family allows us to do exactly that – provide different customer solutions at the lowest cost. Advanced features such as FIFO logic, embedded PowerPC™, RocketIO™ transceivers, and Ethernet MAC, as well as advanced power and packaging technology, makes Virtex-4 devices a perfect choice for us.

Model 351 (Pocket Change)

Our next-generation product, the Model 351, or "Pocket Change," transforms any portable computer into a high-performance multi-channel software defined radio transceiver. The Pocket Change CardBus PC Card accepts two analog input signals through MMCX coaxial connectors on the outside edge of the card. The receiver input is AC-coupled to a 14-bit (80 MSPS) A/D converter. The transmitter output is supplied through a 14-bit (100 MSPS) D/A converter. Most of the digital logic is supplied using a Virtex-4 FPGA device.

When we began developing the Model 351, we investigated various offerings on the market and finally decided to use Virtex-4 FPGAs. The Virtex-4 FPGA family provides the flexibility and features that support both our needs and the requirements of our customers.

The Model 351 design comprises a Virtex-4 FPGA connected to an A/D converter, a D/A converter, and a dedicated PCI bus controller (for the CardBus interface to the host computer) (Figure 1). Although it is targeted at our traditional software defined radio customers, the Model 351 is also suitable for signal acquisition or generation, signal intelligence collection, transceiver modem algorithm prototyping, frequency hop signal generation, or portable signal recorder/playback applications.

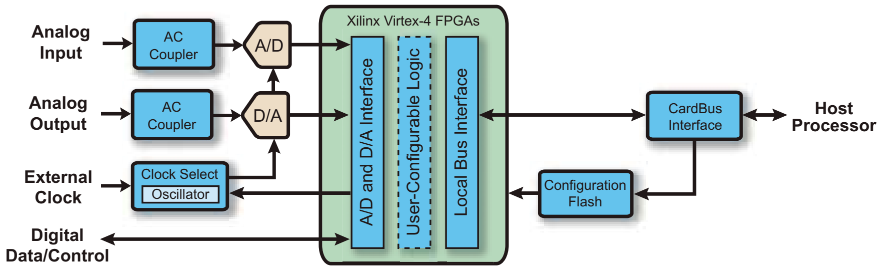


Figure 1 – Model 351 block diagram

Customization and Flexibility

Initially we considered using dedicated digital upconverter/downconverter chips to implement the Model 351 transceiver function. However, many of our customers prefer the flexibility of inserting custom functions into their designs. The customization requirement pushed us to use programmable technology.

By selecting a leading programmable logic architecture, we can address the customization needs of a broad set of customers. Xilinx ISE™ development software provides our customers a familiar design environment to embed custom DSP functions in the uncommitted logic of the Virtex-4 FPGA.

Another benefit from using Virtex-4 FPGAs is that we can offer multiple products using one common hardware platform. This has helped reduce hardware development time and simplify inventory management.

Power and Space Efficiency

One of the challenges in CardBus PC Card development is to select a device that meets the PCMCIA functional specification and the tight power restriction of 3.3W. We were impressed with the power efficiency of the Virtex-4 family, as it consumes half the power of comparable logic solutions.

Virtex-4 FPGAs give us significant features and performance while still meeting the tight power budget of our design. In addition, PCMCIA imposes severe height restrictions in order to fit into the Type II module form factor. The Virtex-4 FF668 package offering is one of the few FPGA packages that meet the height requirements.

Advanced Features and Performance

One key requirement for a software defined radio application is high-performance DSP capability. The performance requirement is driven by the need to support multiple signal channels in real time.

Virtex-4 FPGAs are capable of performing multi-channel digital upconversion and downconversion across the entire Model 351 analog bandwidth. The Virtex-4 device can also perform Fast Fourier Transforms (FFTs) for spectral analysis of incoming signal data.

The Virtex-4 FPGA provides the "heavy lifting" to process digital information between the host computer and the A/D or D/A converter. The signal processing power comes directly from the SX platform. Virtex-4 devices can achieve high-DSP performance by taking advantage of massive parallelism within each FPGA. For math-intensive algorithms (like DUC/DDC applications in a software defined radio), the high number of DSP slices – multiply/add/accumulate engines – that can run up to 500 MHz provides the kind of performance only previously available in fixed ASIC technology.

Our designs also make extensive use of the internal block memories in the FPGA to provide multi-queue FIFO capabilities. The FIFOs are used to buffer data between the A/D or D/A converters and the local bus for DMA operations, providing performance-intensive processing without involving the host CPU in memory transfers. This gives our products the ability to flexibly handle digital radio data without completely consuming the CPU performance of the host computer. With the highest-performance internal block RAM and unique integrated FIFO logic, Virtex-4 FPGAs give us the FIFO quantity and performance that we need to keep up with the bandwidth of the analog components and host interface.

Three Platforms Satisfy Multiple Requirements

The three Virtex-4 platforms (LX, SX, and FX) give us unique capabilities for several upcoming products. For customers wanting to add custom logic functionality, we use the LX platform. LX offers the choice of many different gate densities within the same package footprint, allowing us to use the same base design to support many different customer needs.

We have some designs that necessitate tremendous additional DSP capability for math-intensive processing, including signal modulation and demodulation. For these applications, we see the SX platform as a natural fit. SX devices give us by far the largest amount of DSP performance.

For some of our other designs, we are implementing the advanced system-level block functionality of the FX platform – PowerPC running VxWorks, RocketIO transceivers for optical and PCI Express interfacing, and gigabit Ethernet MAC cores. Because Virtex-4 devices give us three platforms to choose from, we can offer different capabilities across our product line.

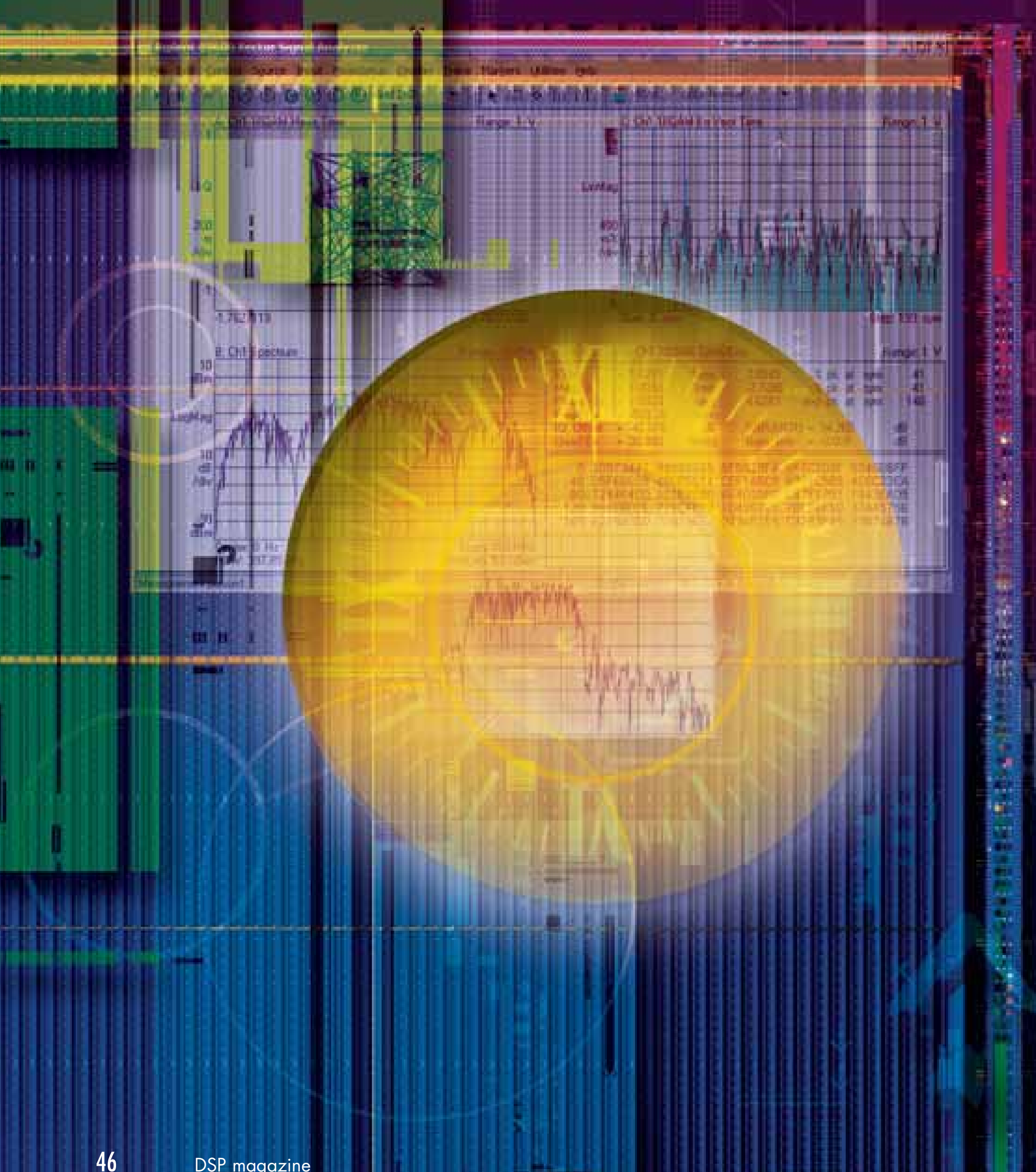
Conclusion

Software defined radio products must address a broad application space, which presents a challenge when selecting component features. The three Virtex-4 platforms give us the feature choice and performance that we require to field a family of solutions for both fixed and mobile installations.

The upcoming Model 351 demonstrates cutting-edge capabilities in an extremely small, power-efficient module that operates in a standard notebook computer. Visit *www.red-river.com* for more information about the Model 351 and other Red River products.

Real-Time Analysis of DSP Designs

Agilent combines the FPGA Dynamic Probe and digital VSA.



by Scott Ferguson
Factory Application Engineer, Logic Analyzers
Agilent Technologies, Inc.
sferguson@agilent.com

As FPGAs become a viable option for high-performance signal processing in the digital communications design space (cellular base stations, satellite communications, and radar), analysis and debug tools must include new techniques to help you get the most optimal performance in your circuits in the least amount of time.

Although signal analysis tools that connect to simulation and RF analog signals are available, it's important to be able to measure signal quality (frequency spectrum, I-Q constellation, and error vector magnitude [EVM]) in the sub-circuits of your FPGA. Thus, Agilent has linked its 89601A Vector Signal Analysis (VSA) software with its line of logic analyzer products (1680, 1690, and 16900 families) to create a digital VSA tool. This tool, when combined with the Xilinx® ChipScope™ Pro Agilent Trace Core, allows you to perform signal analysis anywhere inside your FPGA design quickly and easily.

In this article, we'll show how this combination of tools works – and how it can help you get the most from your Xilinx-based DSP circuits.

Digital VSA

VSA uses Fast Fourier Transform (FFT)-based data processing to provide a combination of time- and frequency-domain displays and measurements. Figure 1 shows a typical VSA display. Although the display is extremely flexible and configurable, the main components include the I-Q constellation plot (upper left), magnitude spectrum (lower left), error vector (upper right), and measurements (lower right). The EVM is displayed in the measurements section. This single value is a key indicator of the quality of the modulated signal.

EVM is computed by extracting I-Q symbols from the captured data; the symbols are the grid points in the constellation defined by the QPSK, QAM, or other modulation scheme. Once extracted from the measured signal, the symbol sequence is used to create an ideal (theoretically perfect) signal known as the "reference" signal. Each measured signal is compared to the reference signal, and the difference is known as an error vector. (The error can contain both I and Q, or magnitude and phase components). The individual error vectors for a single capture are combined to make a single EVM measurement.

Although this analysis software was originally created to analyze analog RF signals, it was developed in a hardware-independent, PC-based software package. Because Agilent logic analyzers are also PC-based, it was easy to extend the VSA software to link to the logic analyzers.

Digital baseband and IF signals are representations of analog signals. Rather than using an instrument that digitizes a signal to enable FFT analysis (like an RF signal analyzer), the signal is digital from the start. These digital versions of analog signals can be displayed in a logic analyzer in a chart-style waveform, which resembles an oscilloscope display (as in Figure 2).

As you can see, when the bus is synchronously sampled and the sample rate meets the Nyquist requirements, the logic analyzer captures a sufficiently accurate version of the "once-was" or "will-soon-be" analog signal.

FPGA Dynamic Probe

The FPGA Dynamic Probe, working with the ChipScope Pro analyzer, can provide access to any part of a DSP design without recompiling. In Figure 3, a simplified digital radio transmitter design is connected to the Agilent Trace Core 2 (ATC2). This core is a switching MUX incorporated into the design using the ChipScope Pro Core Inserter, typically post-synthesis. During core insertion, you select the internal nets to connect to the trace core, and the physical pads to which you will connect the MUX output. These pads are then routed on the circuit board to a logic analyzer probe.

The logic analyzer controls the FPGA through JTAG (downloading the bit file and selecting banks). When you select a new bank, the logic analyzer automatically reconfigures itself to match the names of the nets now connected to the probe.

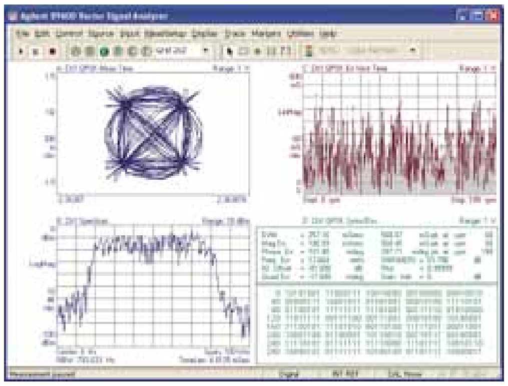


Figure 1 – VSA display

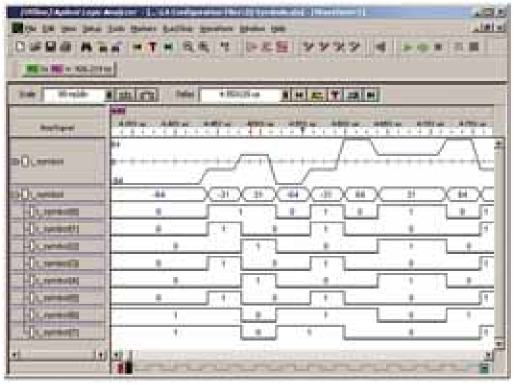


Figure 2 – Chart display of digital bus

Design Example – QAM16 Modulator

With help from our local Xilinx DSP specialist FAE, we created a demo that fits into a small Virtex™-II part (XC2V250-FG256) using Xilinx System Generator for DSP. This tool makes creating DSP designs quick and easy. The design (shown in the block diagram in Figure 3) contains a 25 MHz symbol encoder; a root-raised cosine filter with 24 taps and 4X interpolation (the output running at 100 MHz); and an IF modulation stage with a 25 MHz local oscillator.

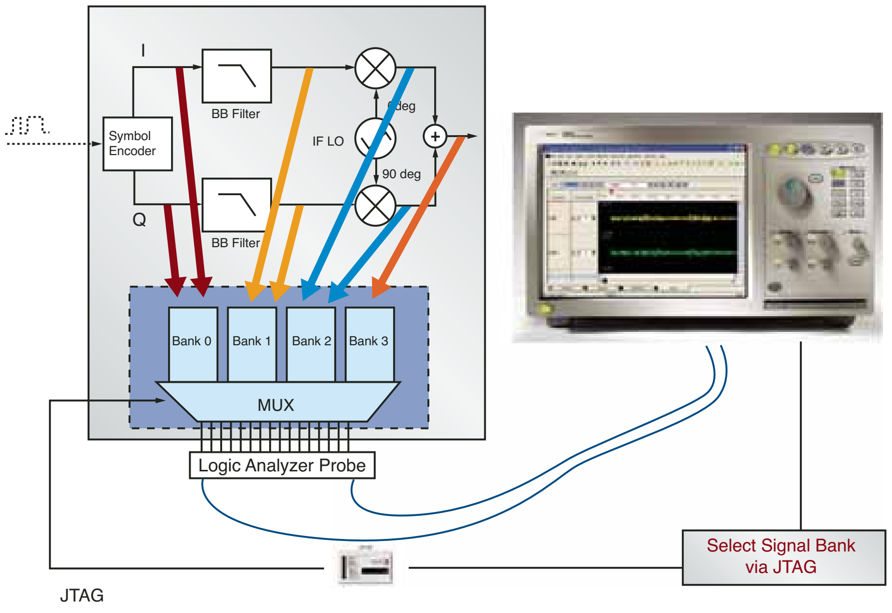


Figure 3 – FPGA Dynamic Probe

Integrating the ATC2 Core in a System Generator Design

After compiling this design into VHDL, we inserted the ATC2 core. To make the signal names more logical on the logic analyzer display, we did some hand-editing of the VHDL. (You could avoid this step by carefully choosing net names in the System Generator.) We then connected most of the interesting nets as output ports from the top-level object to make the net names short enough to fit on the logic analyzer screen.

When connecting nets to output ports solely for use with the FPGA Dynamic Probe, a good trick is to use the "keep" attribute in the VHDL. Because you don't add the ATC2 core to the design until after synthesis, many nets would otherwise be optimized out because they're not connected to anything. In VHDL, the syntax to use the "keep" attribute looks like this:

```
attribute keep : string;

attribute keep of i_symbol:
signal is "true";

attribute keep of q_symbol:
signal is "true";
```

We created an ATC2 core with four banks, each with 48 signals. Using the ATC2 core's 2X TDM option (time-slicing two signals at a time on each pad), this requires only 25 package pads on the FPGA (one for a clock and 24 for data). This gives us access to 192 signals. Actually, we only need to view about 92 signals:

- I-Q symbols, 8 bits each (16)
- I-Q filter output, 24 bits each (48)
- IF local oscillator sine and cosine, 2 bits each (4)
- Combined IF signal (24)

The output of the RRC filter with 24-bit I and Q signals was the largest requirement, defining the number of pins required. If 24 pins were not available, you could drop the least significant bits, losing some dynamic range but still being able to view the signals.

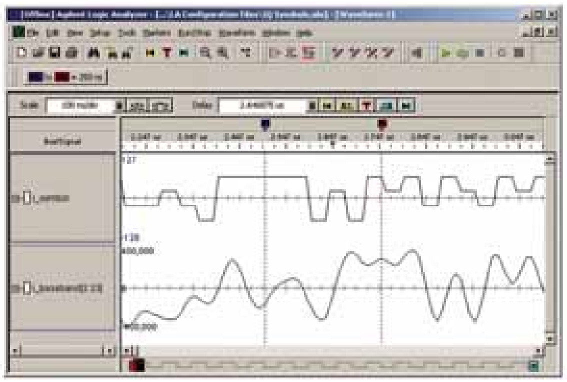


Figure 4 – Filter group delay measurement

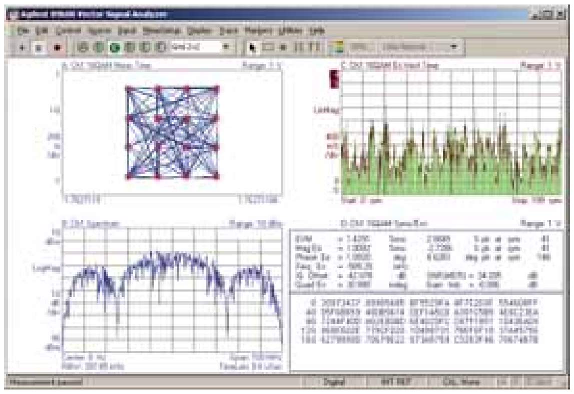


Figure 5 – Unfiltered QAM16 symbols

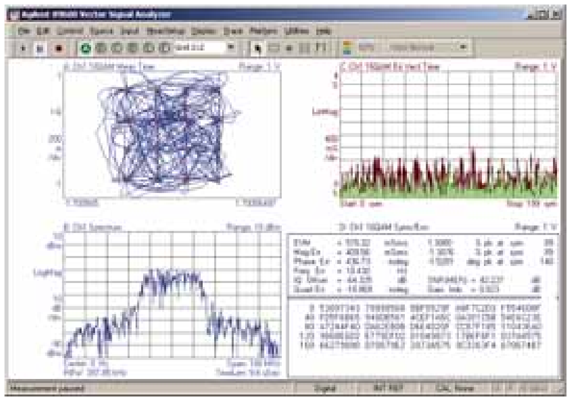


Figure 6 – Filtered IQ baseband data

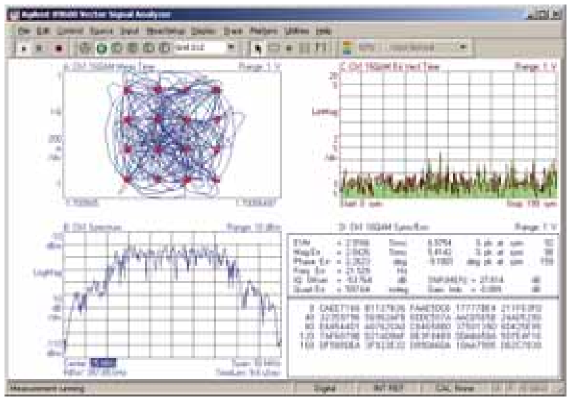


Figure 7 – Digital IF signal

Time-Domain, Logic, and VSA Measurements

The logic analyzer uses synchronous sampling (or "state mode") to capture the output of the ATC2 core. This means that data is sampled on each edge of the ATC2's output clock. Our design has two clock rates in the circuit – 25 MHz for the symbol data before the RRC filter and 100 MHz for all parts after the filter. Because the ATC2 core supports only one clock per core, two options exist for debug:

- Using two cores, one for each clock rate
- Using one core with the faster clock rate and over-sampling the 25 MHz bus

Because the two clocks are correlated – and one is an integer multiple of the other – you can just over-sample the slower bus. If over-sampling is not desirable, the logic analyzer can use a setup that stores every fourth sample, thereby capturing the 25 MHz bus accurately with one sample per 25 MHz clock.

With the extra signals available in the MUX, we were able to double-probe some of the interesting signals. For example, in bank 0 we have the I and Q symbols before the filter, and also the I component after the RRC filter. This means we can do some time-domain analysis in the logic analyzer to measure group delay in the filter, as in Figure 4. Two markers indicate a common signal feature: a wide, flat top and the marker measurement display showing an interval of 250 ns.

After probing the interesting parts of the circuit, we performed vector signal analysis on the signals and measured the quality of our RRC filter and IF modulation stages.

Looking at the QAM16 I-Q symbols before they were filtered (as shown in Figure 5), you can see the 16-point QAM constellation (upper left graph).

With one point per symbol, the lines between the constellation points are straight. The frequency spectrum (in the lower left graph) is centered at 0 Hz and has a 25 MHz pass-band with power in adjacent channels. Adjacent channel power is undesirable in the RF signal, of course, which is the reason for the baseband filter.

By selecting a different bank in the ATC2 core (controlled by the logic analyzer), you can perform analysis on the IQ signal after the baseband filter, as seen in Figure 6. Now the spectrum has sidebands removed, and the measurement display (in the lower right quadrant) shows an EVM of 0.5%. The next time your RF team complains of errors in the baseband design, you can point to this measurement (with which they are quite familiar), and prove that it's not your filter's fault.

In many digital radio designs, this IQ signal would now be converted to analog. However, we performed the IF modulation digitally inside the same FPGA. Switching banks in the FPGA Dynamic Probe gives us access to the digital IF (again, without another synthesis and place and route step), as shown in Figure 7. Note that the spectrum and I-Q constellation are roughly the same, only now centered about 25 MHz. The EVM is a little bit higher, indicating that you may want to use a higher quality local oscillator or another filter stage.

Conclusion

Xilinx System Generator and ChipScope Pro analyzer, combined with the Agilent logic analyzer and Agilent VSA software, allow you to perform real-time in-depth analysis on digital baseband and IF signals inside your Xilinx FPGA. This will save you time and eliminate doubts about the difference between simulation and real hardware. It can also help you communicate with your colleagues on the RF design team, enabling you to speak their language and use the same analysis software regardless of signal format (analog, digital, baseband, or RF).

For more information about these applications, visit *www.agilent.com/find/logic-sw-apps*, or contact your Agilent representative.

Xcell journal

THE AUTHORITATIVE JOURNAL FOR PROGRAMMABLE LOGIC USERS

Would you like to write for Xcell Publications?

It's easier than you think.

We recently launched the Xcell Publishing Alliance to help you publish your technical ideas. We can help you – from concept research and development, through planning and implementation, all the way to publication and marketing.

Submit articles for our Web-based *Xcell Online* or our printed *Xcell Journal* and we will assign an editor and a graphics artist to work with you to make your work look as good as possible. Submit your book concepts and we will bring our partnership with Elsevier, the largest English language publisher in the world, and our broad industry resources to assist you in planning, research, writing, editing, and marketing.

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

Forrest Couch

Executive Editor, Xcell Publications

xcell@xilinx.com

Xcell PUBLICATIONS

The Design and Implementation of a GPS Receiver Channel

An FPGA plus DSP creates a powerful combination.

by Dick Benson
Consulting Applications Engineer
The Mathworks Inc.
dbenson@mathworks.com

In today's competitive environment, with product lifetimes now measured in months, getting it right the first time takes on new importance. Designs are increasingly complex and often comprise hybrid technologies including RF, high-speed signal processing (50-200 megasamples per second [MSPS]), as well as lower speed signal processing and control.

More often than not, it is unclear at the outset of the design process where the optimal positions of the technology boundaries will be. System designers and implementers often make educated guesses as to the partitioning. Only near the end of the design process will they know if their guess was accurate, and that is obviously the worst time to discover faults. The concept of model-based design addresses this as well as other design challenges.

Model-Based Design

Having one set of cost-effective integrated tools that you can use to design, verify, partition, and automatically generate code for both FPGAs and DSPs is now a reality. The Mathworks calls this process "model-based design."

The concept is quite simple. First, you create a functional implementation independent model of the system. This is an "executable specification," a model that forms the basis of all that is to follow. Once you have verified the model to achieve your system objectives, you can incorporate further detail, such as adding fixed-point effects, RF/ADC non-idealities, and partitioning the design between high-speed fixed-point hardware (an FPGA) and lower speed hardware (a DSP). At every step of the process, you verify that the model achieves the performance goals. The final step is to use the automatic code generation capability to flawlessly implement the model on hardware.

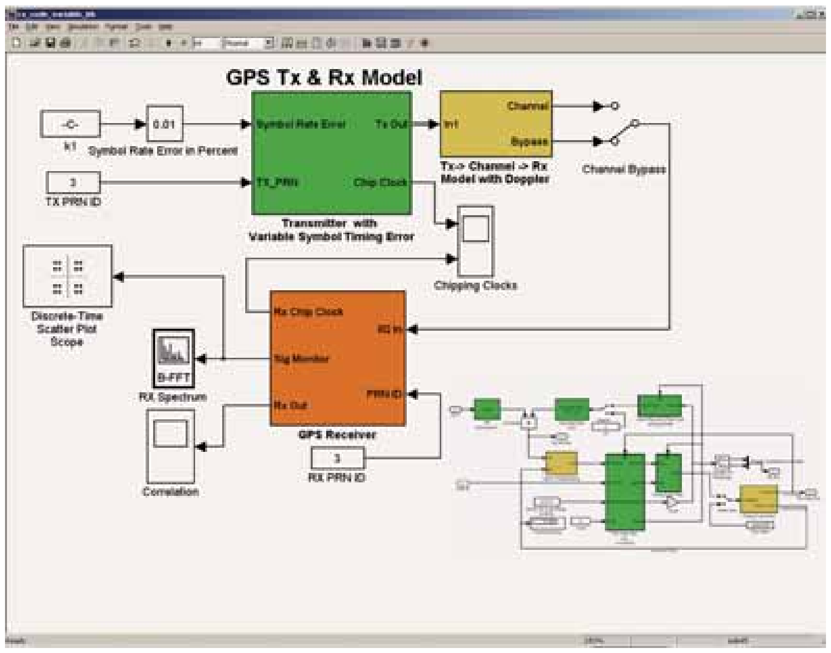


Figure 1 – A top-level view of a GPS system, including a transmitter with timing error, channel model with Doppler, and a receiver with timing and Doppler de-rotation loops. The model contains numerous levels of hierarchy. A glimpse down one level into the receiver is shown in the lower right.

GPS Receiver

The GPS system has been fully operational with 24 satellites in its constellation since 1994. It is used by millions of people, both civilian and military, every day. The fundamental concept of using code-division multiple access (CDMA) for time-delay measurement (yielding range) while allowing all satellites to share the same carrier frequency (1.57542 GHz for civilian access) has not changed in the past 30 years.

Each GPS satellite uses a unique 1,023-chip orthogonal code (Gold code) to spread the low-speed binary phase shift keying (BPSK, 20 ms per bit) navigation data bitstream. The chipping clock rate is 1.023 MHz, and therefore the sequence of 1,023 chips repeats every millisecond.

The GPS receiver generates a local copy of the same Gold code, which is then cross-correlated with the incoming signal. When the receiver code phase aligns with the incoming signal code phase, there is a +30 dB improvement in the SNR (10*log10(1023)), and the BPSK navigation bitstream can then be easily detected. Roughly speaking, you can use the local code time offset, where the correlation is maximized to estimate the time difference between the received signal and the signal transmitted by the satellite. The range is directly proportional to time; therefore, with four satellite ranges, and knowing the positions of the satellites, you can navigate.

Like all CDMA systems, the design of a GPS receiver presents some formidable challenges. The satellites are traveling at close to 7,000 km per hour, giving rise to a Doppler shift ranging from -6000 to +6000 Hz. They are 20,200 km away, running low-power (50W) transmitters. Signals at the terrestrial receiver input are typically 20 dB or more below the noise level. And the local receiver clock is never in step with the ultra-precise Cesium clocks in the satellites, so Doppler and symbol timing recovery loops need to be implemented as well as acquisition logic.

An Executable Specification

Figure 1 shows a top-level view of a Simulink model. It contains the transmitter, channel, receiver, and measurement visualization subsystems. This model has numerous levels of hierarchy – a glimpse under the hood of the receiver is shown in the inset. The transmitter model allows timing errors to be introduced while the channel model includes Doppler shift. These impairments test the receiver timing recovery and carrier tracking loops.

Once the simulation meets the required performance goals, it becomes an executable specification. In theory, the model implements the GPS physical layer according to written specifications that you can find in the public domain.

Verify the Model and Partition

Studies have shown that defects are most often introduced at the beginning of a design. Before adding more detail to the model, it is prudent to verify that it truly implements a GPS receiver. The MATLAB language is becoming increasingly popular in test and measurement applications. The Instrument Control Toolbox option for MATLAB can communicate with virtually any instrument that has a hardware interface. Beyond this, several test and measurement vendors have integrated MATLAB into their instruments. Anritsu is one such company; their Signature spectrum analyzer can capture data into MATLAB with a single mouse click.

An antenna and low-noise pre-amplifier were connected to the Signature analyzer, and tuned to 1.57542 GHz. We recorded approximately one second of I/Q format data, which was then available in the MATLAB workspace. Note that since the satellite signals are more than 20 dB below the noise, it is not immediately obvious that a useable data set has been captured. We used a separate Simulink model implementing a simplified GPS receiver (no tracking loops) to confirm that satellite signals were present in the data. The transmitter portion of the original model was then replaced with a Simulink library block to provide actual satellite data for testing the receiver model.

Next, the model is partitioned into a portion that will reside in the FPGA and a portion that will reside in a floating-point DSP.

The incoming I/Q data at the 8 MSPS rate is first passed through a root-raised cosine FIR filter. Naturally, this higher speed processing is best suited for the FPGA. The filtered signal is then down-sampled by a factor of two, and after the Doppler de-rotation, feeds three cross-correlators: early, prompt, and late, which refer to the local de-spreading code phase driving the respective correlators. The numerically controlled oscillators for both the Doppler and the local de-spreading code are also in the FPGA partition. Because the de-spreading sequence repeats every millisecond, the outputs of the three correlators are only of interest at this one-millisecond rate, which is easily handled by a DSP.

After the receiver model is working using floating-point arithmetic, the next step is to define the fixed-point attributes that will be required for the FPGA partition.

Simulink models can accommodate arbitrary precision fixed-point representations of signals. The FPGA partition includes these fixed-point constraints. As before, numerous levels of hierarchy exist in this model, and Figure 2 represents the top level. Notice that the partitioning reveals a feedback control system between the DSP and the FPGA.

In review, the 8 MHz I/Q satellite signal input is processed by the FPGA producing low-rate (1 kHz) correlator outputs, which are then processed by the DSP. Using these signals, the DSP in turn implements the proportional-integral-derivative controllers for both the Doppler and timing recovery loops. The two controller outputs are fed back into the FPGA. The real captured satellite data is again used as a source to test the partitioning and the chosen fixed-point data constraints. Figure 2 shows the satellite data source (picture), the FPGA partition (yellow), and the DSP partition (green).

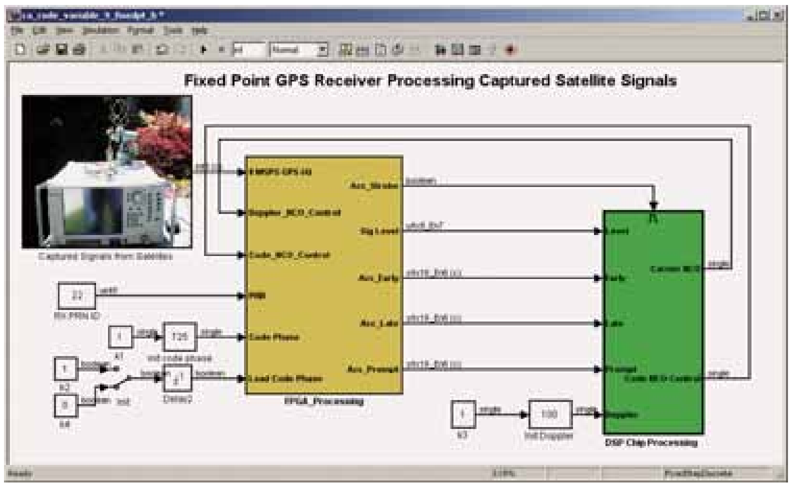


Figure 2 – The receiver portion of the model has now been partitioned with the high-speed fixed-point portion in yellow, and the lower speed single precision floating-point portion in green. RF from actual GPS satellites is captured with a spectrum analyzer. This is now the data source for verifying the model with real-world data.

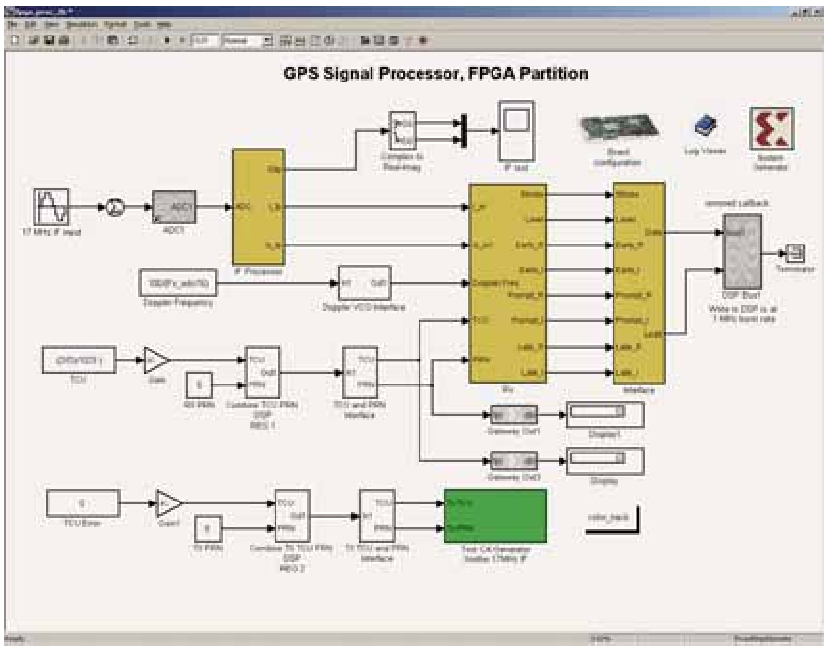


Figure 3 – The fixed-point partition in Figure 2 is now implemented using blocks from the Xilinx System Generator library. A digital down converter (IF processor) has been added along with a 64 MSPS ADC to form a front end for the base-band receiver subsystem. The correlation data from the receiver is time-division multiplexed and buffered with a FIFO before being fed to the DSP partition. All of the VHDL (or Verilog) required to implement this is automatically generated using Xilinx System Generator.

Implement the FPGA Partition

It is now relatively easy to transition the fixed-point receiver subsystem in Figure 2 to one using blocks from the Xilinx® System Generator for DSP library. The transition to

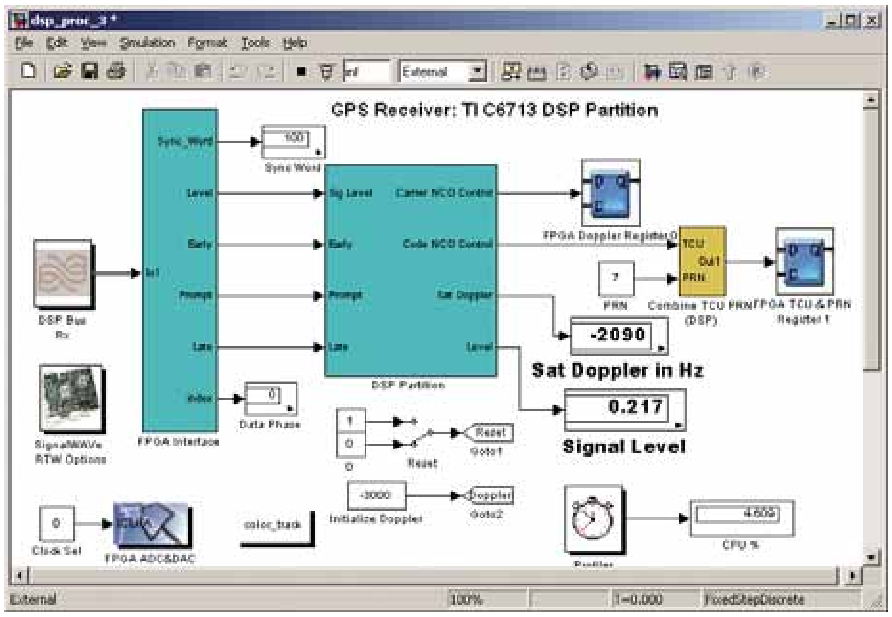


Figure 4 – The floating-point portion of the receiver is implemented using a TI C6713 DSP. The data from the FPGA is de-multiplexed into the signals required to implement both acquisition and tracking of the CDMA satellite signal. The control signals back to the FPGA pass through hardware "gateways" that are specific to the Lyrtech SignalWave hardware. The C code to implement this partition is automatically generated using The Mathworks' Real-Time Workshop. Once the bitstream and binary are loaded to the hardware, the block diagram becomes a user interface, allowing you to change parameters on the fly and have dynamically updating readouts, including time-history scopes.

the FPGA is easy if Simulink blocks that functionally match those in the System Generator library were used.

Figure 3 shows the FPGA portion of the design. Hardware-specific gateway blocks pass signals between the FPGA and the DSP. The output signals from the FPGA include three complex correlation signals (early, prompt, late), a signal-level estimate for AGC, and a synch word. These eight values are time-division multiplexed into a single data stream and fed through a 32-bit gateway block back to the DSP. Control signals for the timing and Doppler tracking loops come through 32-bit gateway blocks to the FPGA, along with other ancillary control signals such as satellite selection.

When running the hardware in real time, the signal input to this GPS receiver comes from an analog front-end down converter with an IF of 17 MHz, not a spectrum analyzer. Therefore, a digital down converter (DDC) is also needed in the FPGA. The DDC takes this 17 MHz real-band pass signal sampled at 64 MSPS and translates it to a base-band I/Q signal running at 8 MSPS.

The high-level signal processing functions (DDS, CIC, FIR, FIFO, TDM) from the Xilinx System Generator library make it easy to implement this portion of the design. Once the simulation is verified, a click of the mouse automatically generates the circa 350 VHDL files (750 kB of ASCII) required to implement the design. After this point we are in the standard Xilinx ISE™ design flow.

Implement the DSP Partition

The top-level view of the DSP portion is shown in Figure 4. The time-division multiplexed signals from the FPGA arrive through the gray gateway block on the left of the model. They are then de-multiplexed into the early/prompt/late correlation signals and level required by the timing recovery and Doppler controllers implemented in the DSP partition. The task of acquiring the GPS is implemented in the DSP using Simulink's event-driven option, StateFlow. Operations such as square root and arc tangent are required. Although these operations are possible with a CORDIC in the FPGA, they are even easier to do in the floating-point DSP.

Because the signals from the FPGA arrive every millisecond, the processing is light duty for the DSP. The numerical readout in Figure 4 indicates that only 4.6% of the available DSP horsepower is being consumed. That said, it should be noted that this example implements one "channel" of a GPS receiver in that only one satellite can be received. A typical GPS receiver incorporates six to ten channels and the loading of the DSP will increase in proportion to the number of channels.

The C code for the DSP is automatically generated using the Real-Time Workshop option in the Simulink environment. Once this is complete, a click of the mouse downloads both the bitstream for the FPGA and the binary for the DSP to the Lyrtech SignalWave hardware. The SignalWave has a Virtex™-II Pro XC2V3000 FPGA 64 MSPS ADC and DAC, as well as audio and video CODECs.

When the real-time processing is started in the hardware, the Simulink block diagram then becomes a GUI that allows you to seamlessly interact with the processing. Scopes and numerical readouts are the primary real-time display options. You can also change the state of switches, the values of constants, and multiplier gains to interact with the design without stopping or introducing gaps in the real-time processing.

Conclusion

This operational GPS receiver was developed using tools from Xilinx, The Mathworks, and Lyrtech. Not a single line of code was handwritten for either the FPGA or the DSP. It took approximately six weeks to create – from scratch – a receiver producing the navigation data bitstream from RF satellite input signals.

This model-based design example has been privately presented to several GPS design groups. Feedback indicates that accomplishing what we have shown in this article typically takes these designers more than a year.

If you are designing and implementing complex signal processing systems for real-time hardware, you cannot afford to be without these tools.

For more information, visit *www.mathworks.com*, *www.xilinx.com/systemgenerator_dsp*, and *www.lyrtech.com*.

320,000,000 MILES, 380,000 SIMULATIONS
AND ZERO TEST FLIGHTS LATER.
THAT'S MODEL-BASED DESIGN.

After simulating the final descent of the Mars Rovers under thousands of atmospheric disturbances, the engineering team developed and verified a fully redundant retro firing system to ensure a safe touchdown. The result—two successful autonomous landings that went exactly as simulated. To learn more, go to mathworks.com/mbd

MATLAB®
&SIMULINK®

The MathWorks
Accelerating the pace of engineering and science

©2005 The MathWorks, Inc.

Designing Control Circuits for High-Performance DSP Systems

These simple techniques could save you days of work.

by Narinder Lall
Sr. DSP Marketing Manager
Xilinx, Inc.
narinder.lall@xilinx.com

Brad Taylor
System Generator Applications Manager
Xilinx, Inc.
brad.taylor@xilinx.com

FPGAs have made significant strides as engines for implementing high-performance signal processing functions, whether for ASIC replacement or performance acceleration in the signal processing chain with DSP processors. Although much has been written about how to use FPGAs as signal processors, not much has been published about building control circuits within such systems.

There are perhaps two key decisions to make when implementing control circuits for FPGA-based DSP systems:

- Should the control circuit be implemented in hardware or developed as a software algorithm?
- What building blocks are available to make the development of the control circuit as efficient and painless as possible?

Software or Hardware?

In this first stage, you can make tradeoffs between algorithms that are implemented in hardware, and those that are better implemented in software using a soft microprocessor (Xilinx® PicoBlaze™ and MicroBlaze™ processors) or hard embedded microprocessor (PowerPC™ 405). Table 1 shows the tradeoffs between hardware- and software-based approaches.

A number of attributes need to be considered when making tradeoffs between these approaches. These include:

- **Algorithm complexity.** You can easily implement simple algorithms (like those that do not need many lines of C-code) in both software and hardware. Although no absolute measure exists to correlate how many lines of C-code represent one slice, a good rule of thumb is that one line equals 1 to 10 slices. When algorithm complexity rises, implementing and testing the algorithm in hardware becomes more challenging. You can more easily implement complex algorithms in lines of C code on a microprocessor, which is the preferred route most designers choose.
- **Need for an RTOS.** If an RTOS is a mandatory piece of the control algorithm, this again favors a software approach that exploits the use of the hard embedded PowerPC on Virtex™-II Pro or Virtex™-4 FX FPGAs, or on external microprocessors. RTOS support for these microprocessors currently includes support from Wind River and MontaVista.
- **Communication with the host.** Communication with a host processor will often – but not always – require a bus architecture of some kind. In this instance, a microprocessor such as the

MicroBlaze processor or PowerPC processor is ideal, as both support bus architectures such as the OPB. Hardware-based host communication using state machines, although possible, can be somewhat more cumbersome.

- **Speed of decisions.** If you specify speed of decisions as clocks per decision, then for decisions that are needed quickly it is obvious that a hardware circuit will be preferred, if not required. For decisions that can be made in hundreds or thousands of clock ticks, software-based algorithms will be sufficiently capable to handle this level of performance.

- **Need for floating point.** Although floating point is largely tangential to control functions, cases do exist where systems employ floating point for control. One example is the calculation of filter coefficients. In sonar systems that require matrices to be inverted, floating-point control is often preferred, as it is often easier to develop with. Floating-point control is also preferred when control precision is high and the algorithms are not available in fixed point.

Once you've decided on hardware or software, you have access to a number of building blocks. Each one is particularly suited to different types of control tasks.

Types of Control Tasks and Possible Circuits

Many different types of control tasks exist. For this article, we have chosen to focus on the following types of problems, which are commonly found in signal processing systems.

- Hardware-Based
  - – Data-Driven Multiplexing
  - – Implementing Finite State Machines (FSMs)
  - – Sample Rate Control
  - – Sequencing – Pattern Generation
- Software-Based
  - – Implementing Low-Rate Control Algorithms
  - – High Complexity Control of Physical Layer Data Paths (MAC layer) (outside the scope of this article)

	Clocks Per Decision	Algorithm Complexity	RTOS	Floating Point	Communicate With Host
Hardware-Based Control	1-10	Simple	No	No	Difficult
Software-Based Control	100-100000	Complex	Yes	Sometimes	Easy

Table 1 – Hardware/software tradeoffs

Table 2 shows a summary of the tools and types of typical control tasks. These are not hard-and-fast recommendations – merely some suggestions for some of the better options. As Xilinx System Generator for DSP is the tool of choice for modeling and designing DSP systems onto FPGAs, in this article we'll also provide some examples using free demos contained within System Generator that demonstrate the use of the control circuit.

Task 1: Data-Driven Multiplexing

An example of a control task that does not require monitoring of the current state is data-driven multiplexing, in which data is monitored and tests are performed on that data. The results of those tests determine the output of the control circuit. Figure 1 shows an example of data-driven multiplexing. Here the function is determined by a simple MATLAB function called xlmax. Input y is selected unless x > y, in which case input x is selected.

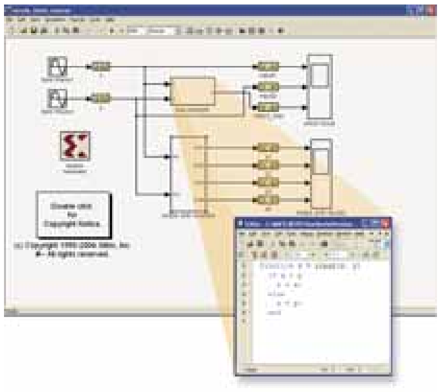


Figure 1 – Data-driven muxing using m-code

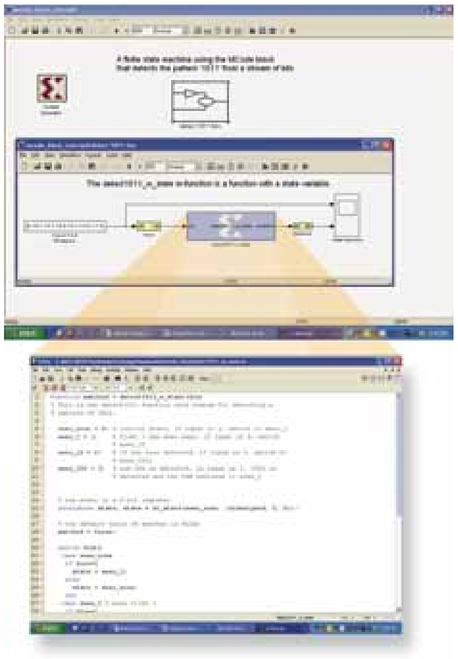
Task 2: Implementing FSMs

Finite state machines are used when decisions must be made based on the current "stored" state of the input(s). For hardware-based high-performance DSP systems, it is not uncommon to see circuits where monitoring of state(s) is performed every clock tick.

Although you can implement them in many ways, perhaps the most common ways to implement FSMs within a System Generator design are through m-code CASE statements (popular with algorithm

	Class of Control Problem				
Toolkit	Sequencing	State Machine	Data-Driven Muxing	Sample Rate Control	Low-Rate Control Algorithm
Pattern Generation Components (ROMs, Expressions, Comparators, Counters, Delays)	X				
M-code		X	X		
Comparators/Muxes			X		
FIFOs/Clock Enables/Up/ Down Conversion				X	
PicoBlaze					X
MicroBlaze/PowerPC 405					X

Table 2 – Types of control problems and tool kits available in System Generator


A finite state machine using the MCode block that detects the pattern 1011 from a stream of bits

The detect1011_w_state m-function is a function with a state variable.

Figure 2 – Implementing an FSM in System Generator

developers) and writing HDL (preferred by hardware engineers). HDL can be easily incorporated into System Generator designs using a black box and co-simulated using ModelSim if necessary.

Figure 2 illustrates how easily you can implement an FSM in System Generator using the m-code block that pulls in a MATLAB script contained in the file detect1011_w_state. The purpose of this script is to detect a 1011 pattern from a signal passed through from the MATLAB workspace.

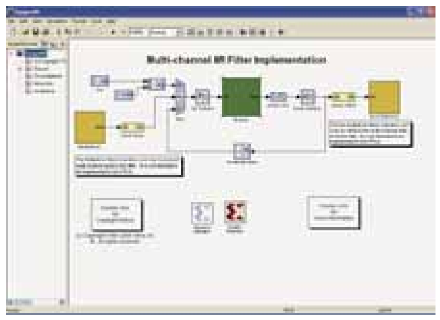


Figure 3 – Sample rate control

Task 3: Sample Rate Control

In high-performance DSP systems, samples often arrive into a system or a piece of a system at a different rate than that of the FPGA clock. We recommend that engineers therefore learn techniques for performing sample rate control. Possible solutions within System Generator that facilitate the design of sample rate control circuits include up/down sampling, clock domains, FIFOs, and clock enables.

Figure 3 demonstrates how you can implement multiple IIR filters using a single time-shared second-order section (biquad). Specifically, 15 distinct IIR filters, each consisting of 4 cascaded biquads, are realized in a "folded" architecture that uses a single hardware biquad. Hardware folding is a technique to time-multiplex many algorithm operations onto a single functional unit (adder, multiplier). For low-sample-rate applications like audio and control, the required silicon area can be significantly reduced by time-sharing the hardware resources.

This design uses a number of control circuits, including a count-limited counter feeding into a two-input mux (that selects between the serial data and the feedback path) and up-sample and down-sample blocks (that control the data rate through the biquad).

Task 4: Sequencing (Pattern Generation)

Sequencing problems usually involve the need for a periodic control pattern that is predictable and not necessarily dependent on the current "stored" state. A common solution for sequencing is to use a simple pattern generator. You can build a pattern generator using building blocks like counters, comparators, delays, ROMs, or the logic expression block within the System Generator block set. The beauty of this underutilized technique lies in its simplicity – yet many designers often opt for more complex, unnecessary state machines.

An example of a pattern generator is contained in the biquad block in Figure 3. The address generator within the biquad block (not shown) generates all of the addresses of the RAMs and ROMs, as well as the write-enable signal for the single-port RAM in the folded biquad module.

Task 5: Low-Rate Control Algorithms

Implementing low-rate algorithms using a Xilinx microprocessor is becoming increasingly common. With a choice of three mainstream processors – the PicoBlaze 8-bit processor, MicroBlaze 32-bit processor, and embedded IBM PowerPC 405 32-bit processor – you have the ability to scale depending on the task at hand. Common tasks that often necessitate the need for on-chip processors include calculating filter coefficients, scheduling tasks, detecting packets (such as in an FEC receiver), and RTOS implementation.

Figure 4 shows a simple control circuit built using the PicoBlaze microprocessor. This example forms the receive path of a 16-QAM demodulator that performs adaptive channel equalization and carrier recovery on a QAM input source. An

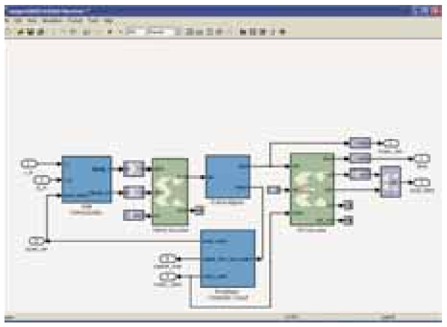


Figure 4 – QAM packet detection using the PicoBlaze microprocessor

attached synchronization marker (ASM) applied by the transmitter is stripped from the demodulated data before concatenated FEC is applied. The PicoBlaze microcontroller controls the RS decoder, maintains frame alignment of the received packets, and performs periodic adjustments of the de-mapping QAM-16 quadrant reference.

Conclusion

When implementing control circuits for high-performance FPGA-based DSP systems, you have access to a number of building blocks within System Generator to make this an easier task. Tables 1 and 2 list some of the tradeoffs to consider and summarize possible control solutions.

All of these designs – and many more – are included within the Xilinx System Generator tool, which retails for $995. You can also try the tool free for 60 days by downloading the evaluation version at *www.xilinx.com/systemgenerator_dsp.*

NU HORIZONS SIGNAL PROCESSING VIRTUALAB™

http://www.techonline.com/community/38647

Lab Summary

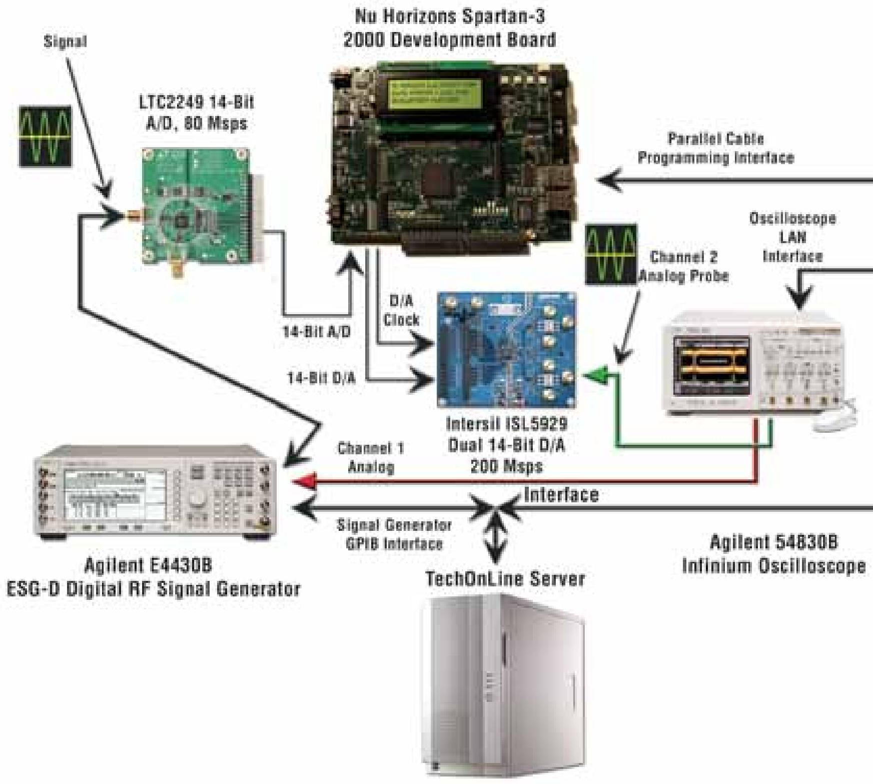
The lab is designed to allow the user to create and download a DSP application, provide signal insertion and measure output waveforms. Advanced algorithms can be developed and complex measurements can be preformed via a full complement of test equipment connected to the VirtuaLab environment. Both a Signal / Pattern Generator and High Speed Oscilloscope are connected to the A/D and D/A modules and each piece of test equipment is placed in the Host Mode so the user can remotely manage the equipments front panel controls. Settings and scripts can be easily saved for re-use on a future session. Real time access to silicon provides accurate measurements, real results and confidence in the findings. The signal insertion and output measurement capabilities enable designers to validate their algorithms.

Nu Horizons VirtuaLab Key Points:

- Enabled by collaboration from a number of industry leaders including: Nu Horizons, Xilinx, TechOnLine, Linear Technology, Intersil, and Mathworks
- Designed to support high performance DSP applications
- First VirtuaLab to incorporate entire signal chain including both A/Ds and D/As
- First VirtuaLab to incorporate Simulink from Mathworks
- Includes test equipment to provide signal insertion and measure output waveforms
- Xilinx ISE Foundation Development Environment
- Xilinx System Generator for DSP

Reference Labs:

- Reference DSP designs are provided in the VirtuaLab environment, allowing the design engineer to evaluate the Spartan-3 2000 FPGA in a pre-verified environment with ease. Reference designs include:
  - Existing Sysgen Tutorial - which introduces the design engineer to other features such as: ChipScope, HDL Cosim, Hardware Cosim and PicoBlaze. Simple FFT with a 256 Tap FIR filter with interpolation by 3. Ease of use and reasonably high performance allow the design engineer to evaluate the tool interface as well as the hardware. The filter design will be provided to the design engineer using the FDA Tool to generate the coefficients, allowing the design engineer the ability to modify the coefficients and view the results. Simulation can be run in Sysgen, as well as run live in hardware.
  - Equalized 16-QAM demodulator including the Adaptive filter. The receiver architecture provides subsystems that demonstrate adaptive channel equalization and carrier tracking on a random QAM data source.



NU HORIZONS ELECTRONICS CORP

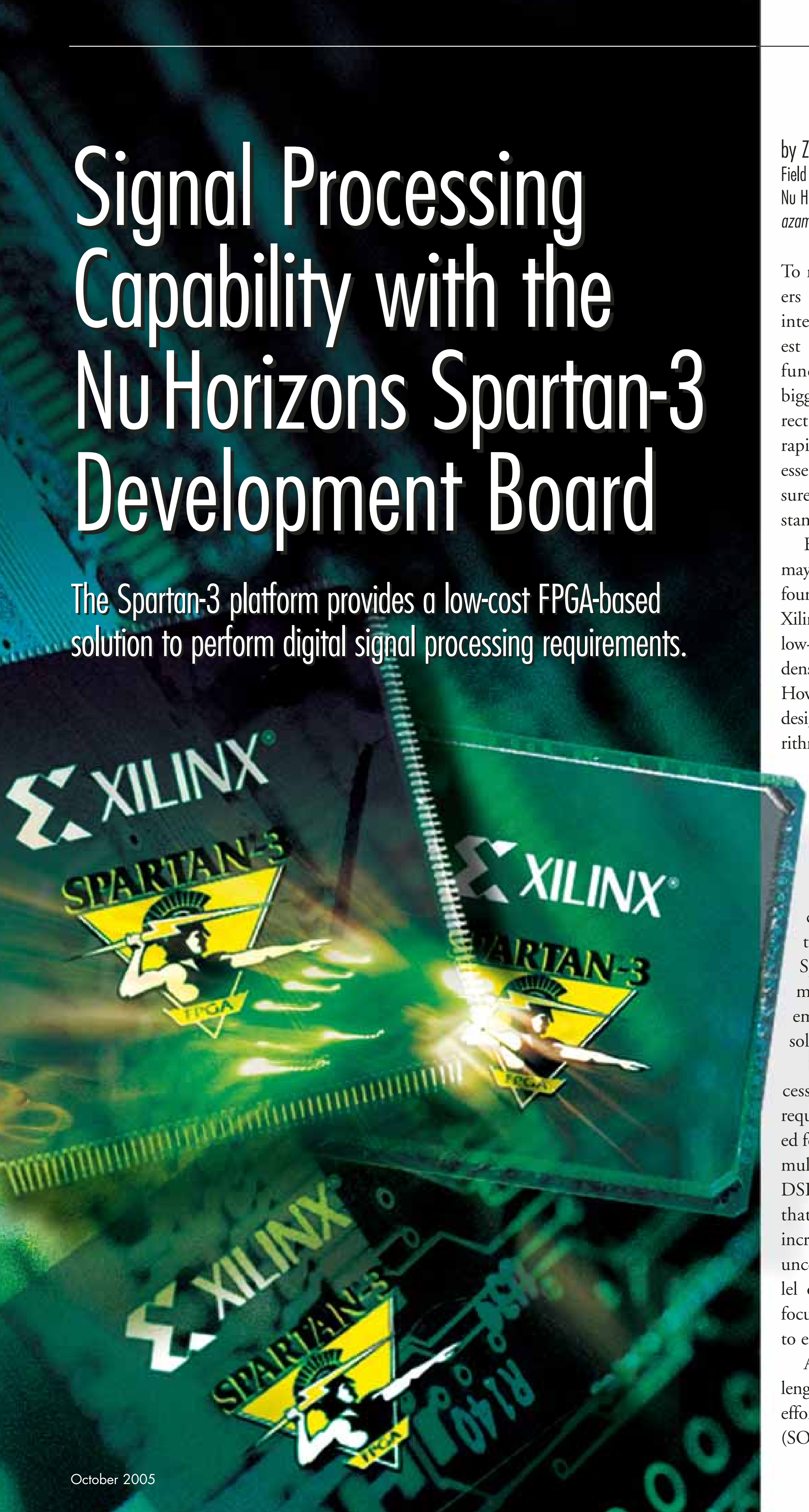
1-888-747-NUHO

www.nuhorizons.com

http://www.techonline.com/community/38647

Signal Processing Capability with the Nu Horizons Spartan-3 Development Board

The Spartan-3 platform provides a low-cost FPGA-based solution to perform digital signal processing requirements.



by Zulfiqar Ali Zamindar
Field Application Engineer
Nu Horizons Electronics Corp.
azamindar@nuhorizons.com

To meet their system design goals, designers today must prototype a new idea and integrate its product features into the lowest cost silicon, complete with versatile functionality. Specifically, two of the biggest challenges are to get the design correct in the first place and to fix a problem rapidly. Immediate design solutions are essential for meeting time-to-market pressures, keeping up with changing industry standards, and prototyping quickly.

Even after the design is completed, a need may exist for field upgradeability if a bug is found or a new functionality is available. The Xilinx® Spartan™ FPGA has been a great low-cost programmable platform for low-density control logic and system interfaces. However, when it comes to signal processing, designers traditionally purchase a fixed algorithm standard product for high-speed multiply and accumulate (MAC) functions. This requirement demands additional design resources, verification time, system components, and more board space.

With the explosion of the wireless communication market and proliferation of low-cost 90 nm processes, Spartan-3 devices – with plenty of logic, memory, and as many as 104 18 x 18 embedded hard multipliers – are the ideal solution for many signal processing needs.

The challenge for today's digital processing systems is their large memory requirements and the very fast MACs needed for rapid mathematical operations. Every multimedia system contains an external DSP processor and memory component that reduces system performance and increases component costs. Once you have uncovered a solution that integrates a parallel or semi-parallel system, you can then focus on improving the overall DSP design to eliminate performance bottlenecks.

As a result of digital processing challenges, many companies have focused their efforts on developing the system-on-chip (SOC) concept by adding feature sets to bring additional functionality to a single piece of silicon. Customized ASICs have become very costly solutions in today's competitive landscape. Traditional DSP processors are capable of carrying out high-speed MAC operations, but have bandwidth limitations. FPGA technology has made tremendous progress in recent years by increasing a large number of intellectual properties to reduce the cost of silicon development in various markets. This is accomplished by optimizing architectures, using leading process technologies, and adding IP cores.

Some of the typical applications for digital signal processing are digital cameras, phones, 3G wireless, video conferencing systems, and high-definition digital televisions. Having a signal processing capability inside an FPGA is the perfect design innovation – the stepping stone to system-on-chip in an FPGA without the high cost of complex customized chip development.

System Generator for DSP

Xilinx expanded its features in the Spartan-3 FPGA by adding embedded multipliers in the architecture. This technological innovation is similar to embedded block memory, clock management, and multiple standards for high-speed I/O circuits, all standard characteristics of the Xilinx Spartan-3 and Virtex™-II Pro families.

Time to market remains critical for companies developing both system hardware and software. With Xilinx System Generator (SysGen), you can simultaneously create behavioral-level hardware blocks and simulate the entire system with just a few tool clicks. The design environment allows you to create block-based systems like digital QAM modulators for software-defined radio, finite impulse response (FIR) filters, image processing functions, mathematical operators, A/D and delta-sigma D/A conversion, and all-in-one silicon for widely used applications.

Using System Generator within the Simulink design environment from The MathWorks, you have unrestricted access to many blocks. You can select both Xilinx and third-party blocks, drag and drop to the Simulink work space, connect, and simulate a system within minutes. The Xilinx System Generator block is used to select various implementation options such as FPGA device, package, speed, system clock, synthesis options, and HDL. The need to write HDL is eliminated, as the tool creates the proper language for you; however, it can write HDL if you prefer.

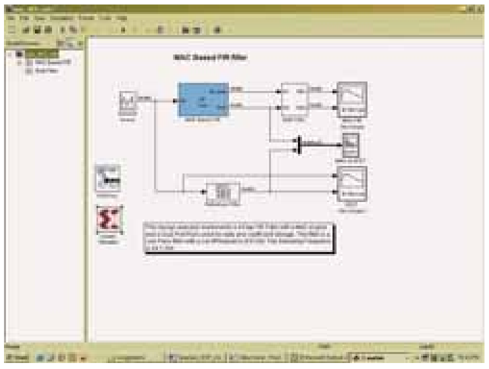


Figure 1 – MAC FIR filter block diagram in System Generator

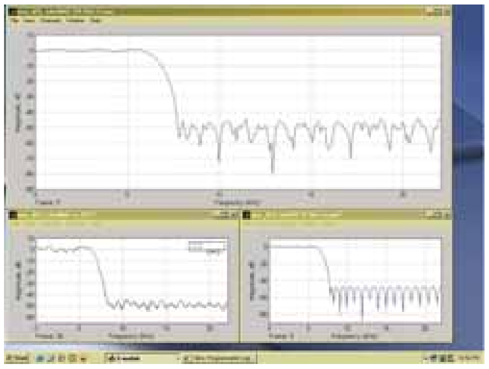
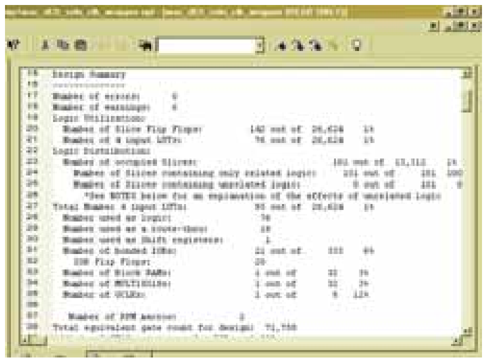


Figure 2 – Simulation view of FIR filter



```
Design Summary
--------------
Number of errors:      0
Number of warnings:    4
Logic Utilization:
  Number of Slice Flip Flops:        142 out of  26,624    1%
  Number of 4 input LUTs:             76 out of  26,624    1%
Logic Distribution:
  Number of occupied Slices:                        101 out of  13,312    1%
    Number of Slices containing only related logic:    101 out of    101  100%
    Number of Slices containing unrelated logic:         0 out of    101    0%
      *See NOTES below for an explanation of the effects of unrelated logic
Total Number 4 input LUTs:            93 out of  26,624    1%
  Number used as logic:               76
  Number used as a route-thru:        16
  Number used as Shift registers:      1
  Number of bonded IOBs:              21 out of    333    6%
    IOB Flip Flops:                   20
  Number of Block RAMs:                1 out of     32    3%
  Number of MULT18X18s:                1 out of     32    3%
  Number of GCLKs:                     1 out of      8   12%

   Number of RPM macros:          1
Total equivalent gate count for design:  71,758
```

Figure 3 – Implementation summary

Simulink also enables you to integrate blocks from many different libraries. Commonly used DSP block sets include math functions, signal management, a variety of filters, transforms, encoders, decoders, and linear feedback shift registers. For example, you can create a fast Fourier transform (FFT) or FIR core with the easy-to-use GUI in System Generator for DSP, customize the core as per your application, and run the Xilinx ISE™ tool in the background to build your signal processor system. This flow can synthesize, place, route, and generate hardware configuration files.

The Simulink environment allows you to verify the functionality of each block or subsystem created with scopes and graphs to view images or observe data.

Digital filters are among the most significant components in digital signal processing applications. The function of a filter is to eliminate undesirable parts of the signal (random noise) or to extract signals in a particular frequency range. Basic FIR filters are used extensively in video broadcasting and wireless communications. A mathematical expression of a basic FIR filter is:

$$Y(n) = \text{SUM}\ h(k) * x(n-k);\ \ k=0 \text{ to } k=N-1$$

It consists of an input sample, output sample, and coefficients. Imagine "x" is a continuous stream of input signal and "y" is a resulting filtered stream of output signal. The "n" and "k" in the equation correspond to a particular instant in time, so to compute "y (n)" at time "n," a group of input samples at "n" different points in time are required, or numerically x (n), x (n-1), x (n-2) ... x (n-k). A group of "n" input samples are multiplied by "n" different coefficients and summed together to form a result "y (n)."

This design example implements a 43-tap FIR filter with a MAC engine and a dual-port RAM used for data and coefficient storage. The filter is a low-pass filter with a cut-off frequency of 6 KHz. The sampling frequency is 44.1 KHz.

Figure 1 represents the model of the filter. The model has a coefficient width of 12, a coefficient binary point value of 12, a data width value of 10, a data binary point value of 8, and a sampling frequency of 44.1 KHz. Scope shots of the filtered output are shown in Figure 2. An implementation summary displaying the use of RAM and multiplier resources is shown in Figure 3. This design achieved ~ 125 MHz performance in a -4 speed grade of the Spartan-3 device.

The NuHorizons Spartan-3 Board

Xilinx, its distributors, and third-party companies offer several boards for prototyping or emulating a DSP-based system. A low-cost prototyping platform from Nu Horizons Electronics Corp. is the Spartan-3 development board (HW-AFX-SP3-2000-DB) (Figure 4). The board comprises these elements (Figure 5):

- Xilinx XC3S2000-4FG676 Spartan-3 device
- XCF08 Flash PROM for configuration
- 4 x 24-character LCD display
- Graphical LCD interface
- 64 Mb SDRAM (2-1 Mb x 16 x 4)
- 32 Mb Flash 2 Mb x 16
- 50 MHz clock oscillator
- PLL clock multiplier
- CAN 2.0B transceiver
- RS232 interface
- PS2 interface
- Audio CODEC
- Two-channel A/D and D/A converter
- 10/100 Ethernet MAC
- 10/100 Ethernet PHY
- Flash memory interface
- SDRAM memory interface
- LED/push buttons
- JTAG configuration header for programming
- 16-bit LVDS I/F with clock and control
- Test point headers for debugging

This fully loaded Spartan-3 development board is priced at $449, which includes all of the features necessary to prototype a DSP-based system design. The board comes with a user's manual, power supply, documents, and design files. The option of getting the board bundled with ISE Foundation™ software, System Generator, and ChipScope™ Pro software is also available at an additional cost.

Besides DSP designs, the Spartan-3 platform is a great tool to implement many other Xilinx reference designs. Several designs written by Nu Horizons' field

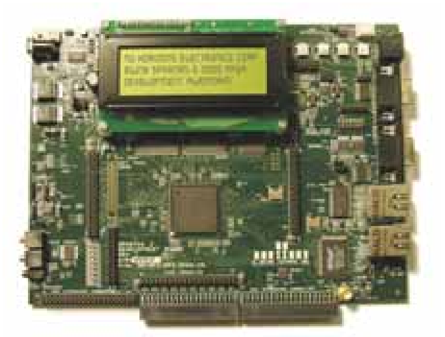


Figure 4 – Spartan-3 board

application engineers cover topics such as memory controllers, embedded processors, hardware-in-the-loop with digital signal processing, and system monitor design using ADC and DAC on the board. ADC and DAC are very powerful attributes of our low-cost board, and two of the many competitive board features. The Spartan-3 platform can be expanded with the add-on ADC board from Linear Technology.

Conclusion

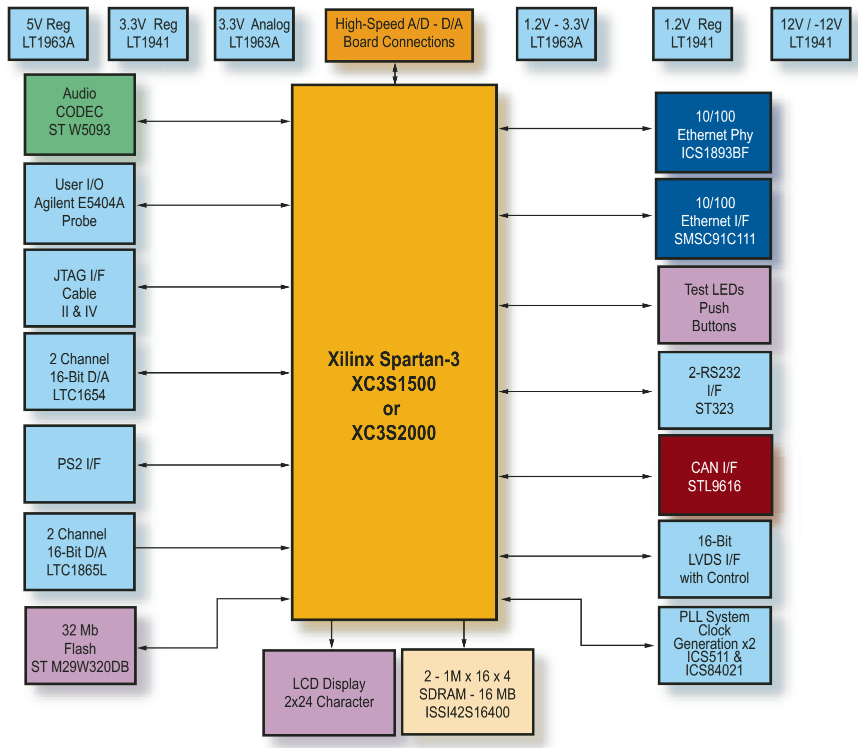
With fast multipliers and lower cost FPGAs, engineers now have the ideal solution to their signal and image processing requirements. With tools like System Generator, anyone can implement a powerful parallel or semi-parallel customized DSP system-on-chip design within days.

The Spartan-3 board from Nu Horizons is a perfect solution for prototyping signal processing using Spartan-3 FPGAs. The board has all of the interfaces necessary to create designs for wireless and digital imaging applications if you want to:

- Integrate your logic and signal processing capability on a single chip
- Prototype a configurable DSP system-on-chip
- Reduce the cost of conventional serially implemented external signal processors
- Improve system performance

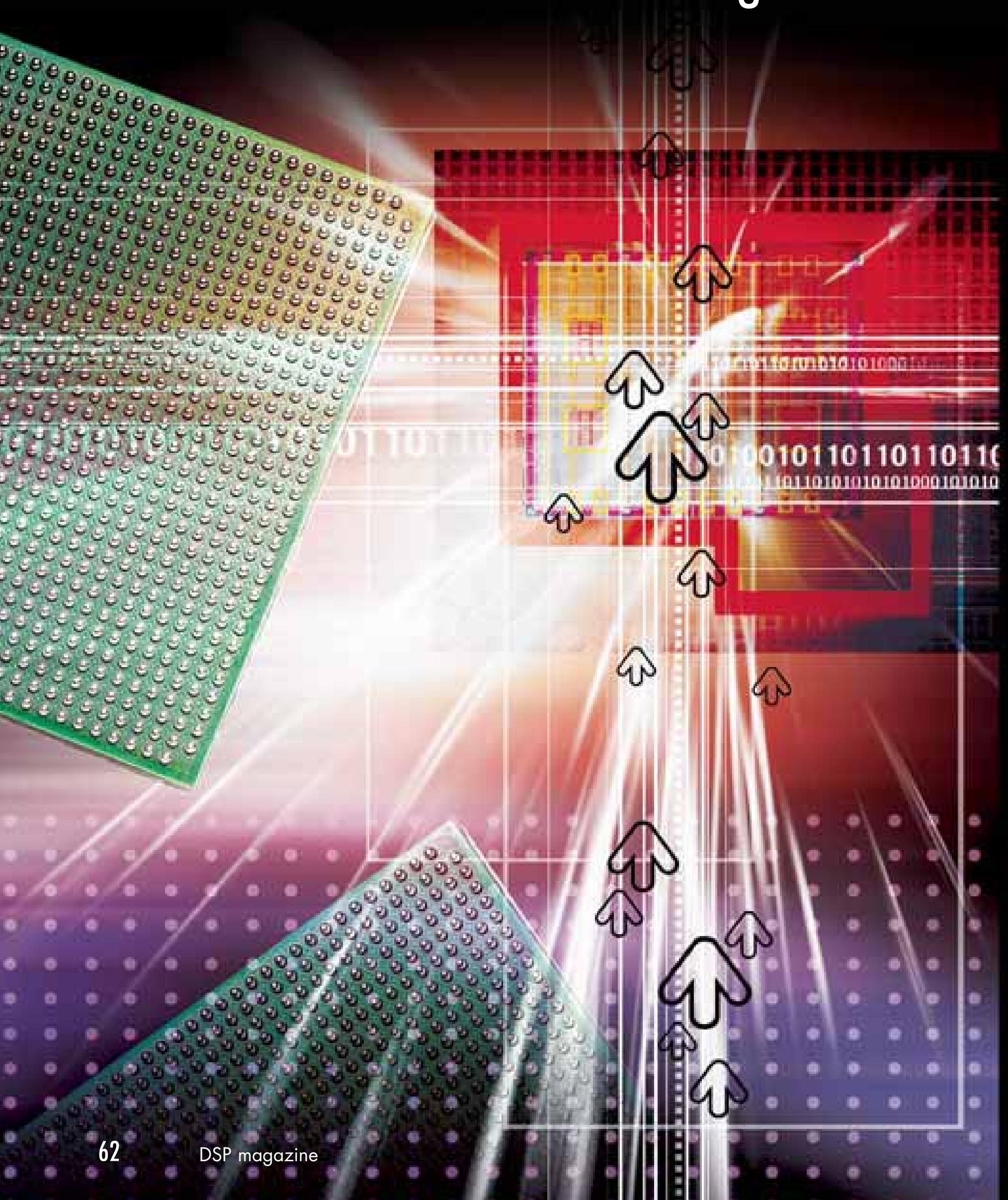
Nu Horizons is also in the process of releasing a Virtex-4 platform board for customers requiring higher density logic, more memory, and hard MACs running up to 500 MSPS for high-performance interfaces to video and imaging applications.

For all of the designs and related documentation for the Spartan-3 board, visit the Nu Horizons website at *www.nuhorizons.com/sp3/*.


5V Reg LT1963A
3.3V Reg LT1941
3.3V Analog LT1963A
High-Speed A/D - D/A Board Connections
1.2V - 3.3V LT1963A
1.2V Reg LT1941
12V / -12V LT1941
Audio CODEC ST W5093
User I/O Agilent E5404A Probe
JTAG I/F Cable II & IV
2 Channel 16-Bit D/A LTC1654
PS2 I/F
2 Channel 16-Bit D/A LTC1865L
32 Mb Flash ST M29W320DB
Xilinx Spartan-3 XC3S1500 or XC3S2000
LCD Display 2x24 Character
2 - 1M x 16 x 4 SDRAM - 16 MB ISSI42S16400
10/100 Ethernet Phy ICS1893BF
10/100 Ethernet I/F SMSC91C111
Test LEDs Push Buttons
2-RS232 I/F ST323
CAN I/F STL9616
16-Bit LVDS I/F with Control
PLL System Clock Generation x2 ICS511 & ICS84021
Figure 5 – Spartan-3 XC3S1500 /2000 board block diagram

Designing with the Virtex-4 XtremeDSP Slice

Harness the full capabilities of the XtremeDSP slice in filter design.



by Niall Battson
DSP Applications Engineer
Xilinx, Inc.
niall.battson@xilinx.com

With the introduction of Xilinx® Virtex-4™ FPGAs in September 2004, the world of DSP design witnessed a dramatic leap in programmable logic DSP: higher performance, lower cost, lower power, and maximum flexibility.

At the same time this phenomenon asks DSP hardware engineers to change their traditional way of designing and embrace a different approach. These great improvements have been made possible by the XtremeDSP™ slice.

The XtremeDSP Slice

The XtremeDSP slice (also referred to as the DSP48) is a high-performance multiplier and arithmetic unit with great flexibility that can form the building block of many DSP algorithms implemented in FPGAs. A detailed diagram of the DSP48 structure is shown in Figure 1.

The XtremeDSP slice comprises four main sections:

- I/O registers
- 18 x 18 signed multiplier
- Three-input adder/subtractor
- Op-mode multiplexers

The I/O registers ensure a maximum clock performance of 500 MHz in the fastest speed grade device (400 MHz in the slowest speed grade), also ensuring support for higher sample rates. The dynamic op-mode multiplexers are key to the functionality of the structure; they are responsible for the DSP48's great flexibility. For example, in a simple MACC engine, you set the X and Y MUX to multiply and select the feedback path from the registered output P as the Z MUX input to the arithmetic unit.

In the Virtex-4 architecture, XtremeDSP slices are arranged in columns. The most important aspect about the column is the cascade logic and routing between each block, which exists on both the input and output stages of each slice. This dedicated routing enables a number of filters and other functions to be built entirely within the XtremeDSP slice, thus removing the need for signals to be routed through the FPGA interconnect or logic fabric.

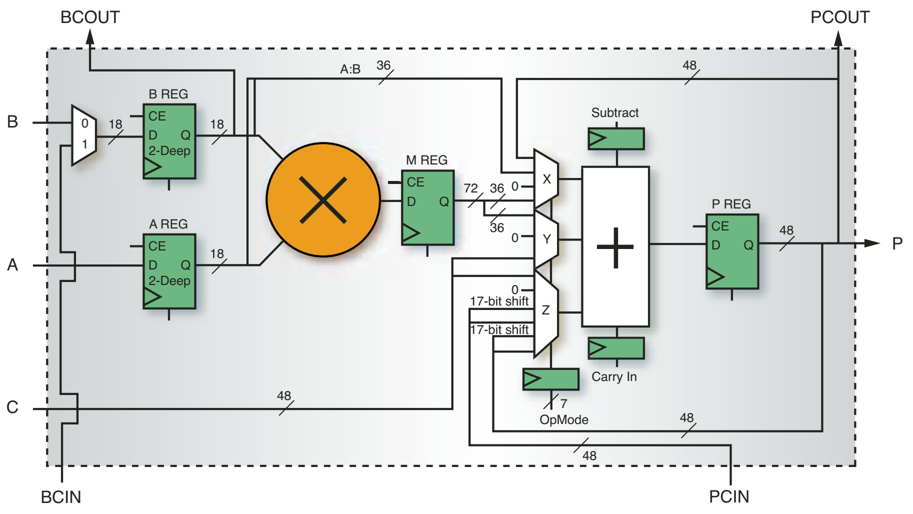


Figure 1 – Simplified diagram of the XtremeDSP slice

However, you must take this adder-chain configuration into account when designing functions that exploit the XtremeDSP slice. Herein lies the fundamental change in the approach to filter design. The simple, traditional adder-tree approach limited the performance and extensibility of a given filter implementation. By using adder-chain-style implementations, these limitations are lifted and the huge benefits Virtex-4 FPGAs offer are possible.

The embedded nature of the XtremeDSP slice has also had a radical impact on reducing the power consumed by high-speed multiply and add functions. Figure 2 illustrates this dramatic reduction, showing that the dynamic power consumption is 1/17 of Virtex-II Pro™ devices with a specification of 2.9 mW/100 MHz. As a designer, you should migrate as much functionality into these embedded functions as possible.

Filter Techniques

During the last ten years, hardware and FPGA designers have created a wide variety of filter architectures to efficiently exploit the building blocks that the current generation of technology offers. With the introduction of Virtex-4 FPGAs and the XtremeDSP slice, filter implementations must change to most efficiently exploit this latest FPGA offering. Filters are prolific in DSP designs and nearly always form the starting point for analyzing an architecture.

The general FIR filter equation is a summation of products (also known as an inner product) defined in the equation:

$$y_n = \sum_{i=0}^{N-1} x_{n-i}\, h_i$$

In this equation, a set of N coefficients is multiplied by N respective data samples, and the results are summed to form an individual result. The values of the coefficients determine the characteristics of the filter: low-pass, band-pass, or high-pass.

The Semi-Parallel FIR Filter

Even within the filter world, you can implement a wide variety of filters. The key parameters that tell us which FIR filter implementation we will construct are:

- Number of coefficients (N)
- Sample rate (Fs)

Let's examine a particular filter structure to demonstrate the key design techniques that can help you maximize the benefits of Virtex-4 devices. Our filter has 20 coefficients and a sample rate of 74.25 MHz.

As noted earlier, the maximum capable clock speed of the XtremeDSP slice is 400 MHz in the slowest speed grade (-10). Therefore, we have a total of five clock cycles to perform the required 20 multiply and adds to form the result.

This equation determines how many multipliers to use for a particular semi-parallel architecture:

Number of Multipliers =
(Maximum Input Sample Rate x
Number of Coefficients) / Clock Speed

For our example, the required number of multipliers will be four. Once we have determined the required number of multipliers, there is an extendable architecture using the XtremeDSP slices that can serve as the basis for the filter.

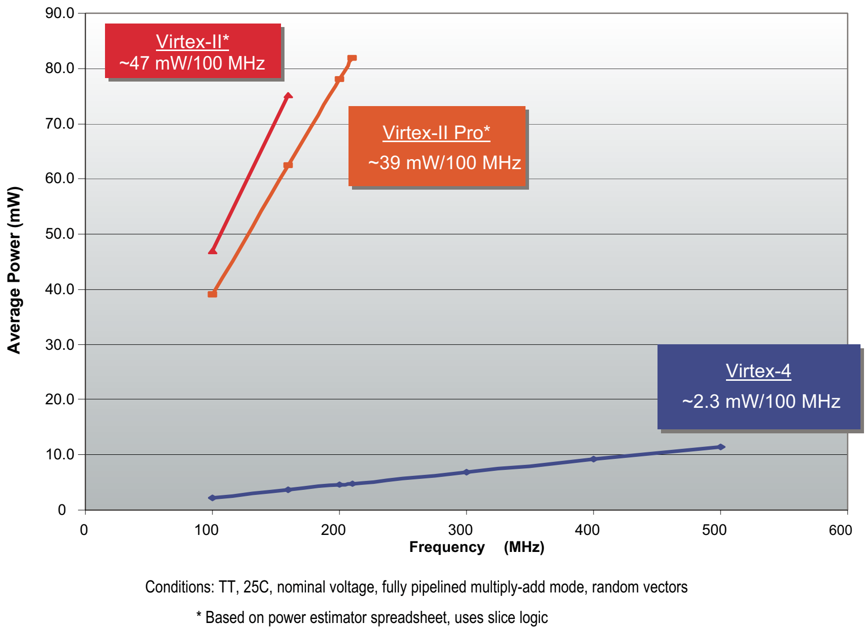


Figure 2 – Dynamic power consumption of the XtremeDSP slice

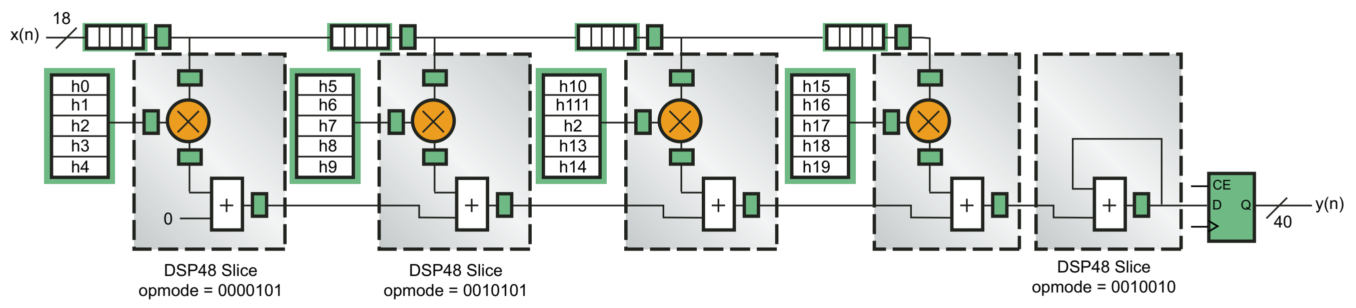


Figure 3 – The four-multiplier semi-parallel systolic FIR filter

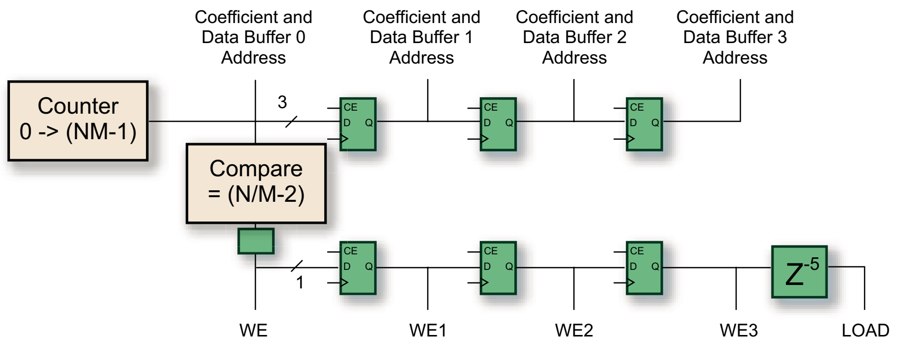

Coefficient and Data Buffer 0 Address
Coefficient and Data Buffer 1 Address
Coefficient and Data Buffer 2 Address
Coefficient and Data Buffer 3 Address
Counter 0 -> (NM-1)
3
Compare = (N/M-2)
1
Z^{-5}
WE
WE1
WE2
WE3
LOAD

Figure 4 – Control logic for the four-multiplier semi-parallel FIR filter

XtremeDSP arithmetic units are designed to be chained together easily and efficiently thanks to dedicated routing between slices. Figure 3 illustrates how the four XtremeDSP multiply and add elements are cascaded together to form the main part of the filter.

It is critical to highlight the usage of the adder chain here rather than the more traditional adder tree. The adder chain has a profound impact on the control logic required for the filter, as well as its efficiency, because of the mapping to the XtremeDSP slice.

Continuing to analyze the filter structure, an extra XtremeDSP slice is required to perform the accumulation of the partial results, thus creating the final result. A new result is created every five clock cycles. This means that for every five cycles the accumulation must be reset to the first inner product of the next result. This reset (or load) is achieved by changing the op-mode value of the XtremeDSP slice for a single cycle, from 0010010 to 0010000 (this is just a single bit change). At the same time, the capture register is enabled and the final result stored on the output.

The Control Logic

The control is the most important and complicated aspect of semi-parallel FIR filters; getting it right is crucial to filter operation. Because the XtremeDSP slice is most efficiently used in adder chains, memory addressing is necessary to provide the delay for each multiply-add element that the adder chain causes. Figure 4 illustrates the control logic required to create memory addressing.

The counter creates the fundamental zero through four count. This is then delayed by one cycle by the use of a register in the control path. Each successive delay is used to address both the coefficient memory and the data buffer – and their respective multiply-add elements. Hence, a single delay is required for the second multiply-add element, two delays for the third multiply-add element, and so on. Note that this is extensible control logic for M number of multipliers.

Figure 4 also shows write enable sequencing. A relational operator is required to determine when the count limited counter resets its count. This signal is high for one clock cycle every five cycles, reflecting the input and output data rates. The clock enable signal is delayed by a single register just like the coefficient address; each delayed version of the signal is tied to the respective section of the filter.

The filter and control logic are extremely cascadable. The address for each SRL16E data buffer and coefficient memory pair are a delayed version of the previous elements' address, and are identical.

The performance and resource utilization for our filter is specified in Table 1. In the table, you can see how logic slice utilization dramatically drops when using the XtremeDSP slice. Clock frequency performance approximately doubles over Virtex-II Pro FPGAs.

Four-Multiplier 20-Tap Semi-Parallel FIR Filter 18-Bit Data, 18-Bit Coefficients	Virtex-4 (-11)	Virtex-II Pro (-7)
Logic Slices	108	309
XtremeDSP Slice	5	
Embedded Multipliers		7
Performance (Sample Rate)	90 MHz	77 MHz
Performance (Clock Frequency)	450 MHz	231 MHz

Table 1 – Resource utilization and performance of four-multiplier 20-tap semi-parallel FIR filter

Three Important Design Points

This new filter architecture, along with Virtex-4 devices and the XtremeDSP slice, addresses the demanding needs of current and future DSP designs. However, it is only one filter in an extremely large array of possible implementations, not to mention other DSP functions such as IIRs, FFTs, and DCTs.

Knowing this, you can take away three very important design questions that will enable you to exploit the XtremeDSP slice and Virtex-4 device as designed.

1. *Is the design running as fast as possible?*

   The fastest speed grade (-12) should run at 500 MHz. If your design is running at 50 MHz, you've got the room to reduce your resource utilization by increasing performance (and reducing cost) by making more efficient use of the FPGA resources. The faster a particular function operates, the smaller it becomes. Our semi-parallel FIR filter, for example, used five XtremeDSP slices running at 375 MHz instead of 20 XtremeDSP slices running at 74.25 MHz.

2. *Are there any XtremeDSP slices left?*

   If you are not using them all up, you can probably add some functionality. This can lead to logic slice reduction and lower power consumption.

3. *Are you using adder chains instead of adder trees?*

   DSP algorithms must aim to exploit adder chain-based implementations wherever possible, as this will lead to the best utilization of the XtremeDSP slice. Such implementations will result in performance gains, power reduction, and logic slice reduction.

Conclusion

For more information, see the XtremeDSP Slice Design Considerations User Guide, which provides in-depth details on other filter implementations and DSP functions, at *www.xilinx.com/bvdocs/userguides/ug073.pdf.* There are also other HDL and System Generator for DSP reference designs to get you started.

Let Xilinx help you get your message out to thousands of programmable logic users worldwide.

That's right ... by advertising your product or service in the Xilinx *Xcell Journal*, you'll reach more than 70,000 engineers, designers, and engineering managers worldwide.

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

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

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

Join the other leaders in our industry and advertise in the Xcell Journal!

XILINX®

Synthesis Tool Strategies

Set up your designs in Synplify for performance improvement and area savings.

by Steve Pereira
Technical Marketing
Synplicity, Inc
stevep@synplicity.com

You can benefit greatly from a proper synthesis strategy. Such strategies include knowing the final target architecture, knowing what coding problems could arise, and understanding what performance the periphery will require. You should also understand how to use IP. Are models available? Is cost an issue? Your initial setup can greatly affect productivity and help you achieve quicker peripheral timing closure.

In this article, I'll describe a known good strategy while using Synplify Pro tools.

Best Practices

Setting up your design correctly can result in huge performance increases or reductions in area. The following checklist describes the best practices to use when setting up your design.

1. Include any CoreGen EDIFs or timing models for black boxes. If you use black-box IP in the design, ensure that all EDIF, NGC netlists, or timing models are provided. It is essential that the Synplify Pro tool knows the timing requirements into and out of the box so that surrounding logic can be altered to reduce or remove criticality.

   If the design has an ngc file, use the ngc2edn (provided in the Xilinx® ISE™ /bin/ directory) converter utility to produce an edn file for synthesis.

2. Ensure that the device is correct and size the design. Ensuring that the wireload models are correct for synthesis can greatly affect the resulting logic. The selection of wireload models is simply a matter of selecting the correct device and speed grade. If synthesis is performed on a different device to the final implementation, sub-optimal results are quite likely.

   Selecting the correct device will also provide Synplify Pro with the accurate number of resources. One example of the importance of this is with block-select RAM mapping. Synplify orders all of the RAMs from biggest to smallest, and then starts to map the largest RAMs until there are no block RAMs left. The rest are placed into distributed RAM. If the wrong device is selected, sub-optimal mapping will occur.

   Sizing the design also has a significant impact. For example, if the design uses 80% of the device, the wireload models are correct for the design. If the design consumes only a small percentage of the total resources (for example, synthesizing just a part of the design for verification), the wireload models will be inaccurate. To resolve this problem, you can assign the logic to an area group. Please see the Synplify Pro documentation for instructions on how to do this.

3. Provide accurate clock constraints. Under- or over-constraining results in reduced performance. Do not over-constrain by more than 15%. For maximum performance, ensure that there is 10% negative slack on the critical clock. This ensures that critical paths are squeezed. The Fmax field on the front panel is fine for a quick run, but do not use it if you need maximum performance. Put unrelated clocks in separate clock groups in the Synplify Pro .sdc file. If your clocks are in the same group, the Synplify Pro tool works out the worst-case setup time for the clock-to-clock paths.

Figure 1 shows a timing diagram for two clocks that are in the same clock group. Synplify rolls the clocks forward until they match up again. The tool then calculates the minimum setup time between the clocks, in this case 10 ns.

If the clocks are unrelated, there may be several hundred clock periods before the clocks match up again. This may result in the worst-case setup time being very small (100 ps). You can check the setup time in the clock relationships table in the log file. If the setup time is too short, it is best to re-constrain the clocks so that they are more related.

4. Specify timing exceptions. Provide all timing exceptions, such as false and multicycle paths, to the Synplify Pro tool. With this information, the tool can ignore these paths and concentrate on the real critical paths.

5. Constrain I/Os. If the design has I/O timing constraints, it is likely that the critical path is through the I/O block (IOB). The Synplify Pro tool sees these paths as the most critical and tries to optimize them. Usually, I/O paths physically cannot be optimized any further; as they are the most critical, the Synplify Pro tool stops optimizing the rest of the design.

A new switch has been added to the Synplify Pro 7.3 release called "use clock period for unconstrained I/O." When enabled, the tool does not include any unconstrained I/O paths in timing optimizations.

6. Keep code generic. Keeping code generic and not locked down to a particular architecture can aid design, reuse, and portability. For example, with the DSP48 block in the Virtex™-4 architecture, you can specify generic code to implement a DSP function. The tool will map to that component(s) when possible and reduce uncertainty regarding how the function was mapped. If DSP blocks are generated, timing is better known and can speed up debug – and time to market. You can specify the register configuration and code in the opcode to drive the DSP block configuration, all with generic code.

If for some reason you wish to use a different device family, such as moving from Virtex-4 FPGAs to Virtex-II Pro FPGAs, the porting process itself should be seamless, but you may feel uncertain about implementation and logic levels.

Good Switch Settings

- Retiming and pipelining. Enabling retiming and pipelining options can improve your design performance by as much as 50%. Retiming attributes such as syn_allow_retiming let you refine your constraints by surgically applying retiming to a single register. Synplicity recommends that you enable both of these switches.

- Resource sharing. Always turn this switch on. The behavior of this switch was changed in Synplify 8.0. With the switch on, timing-driven resource sharing occurs. Non-critical logic will have resource sharing, but critical logic will not share resources (for performance reasons).

- FSM Compiler extracts and optimizes FSMs based on the number of states:
  - 2 to 4: sequential
  - 5 to 40: one-hot
  - More than 40: gray

  Synplicity also recommends enabling the FSM Compiler and setting default enumerated encoding to the "default" value for VHDL designs.

- FSM Explorer timing-driven state encoding. The Synplify Pro tool automatically selects the best encoding for the specified timing. This switch is design-dependent. If the critical path starts or ends at a state machine, turn the switch on.

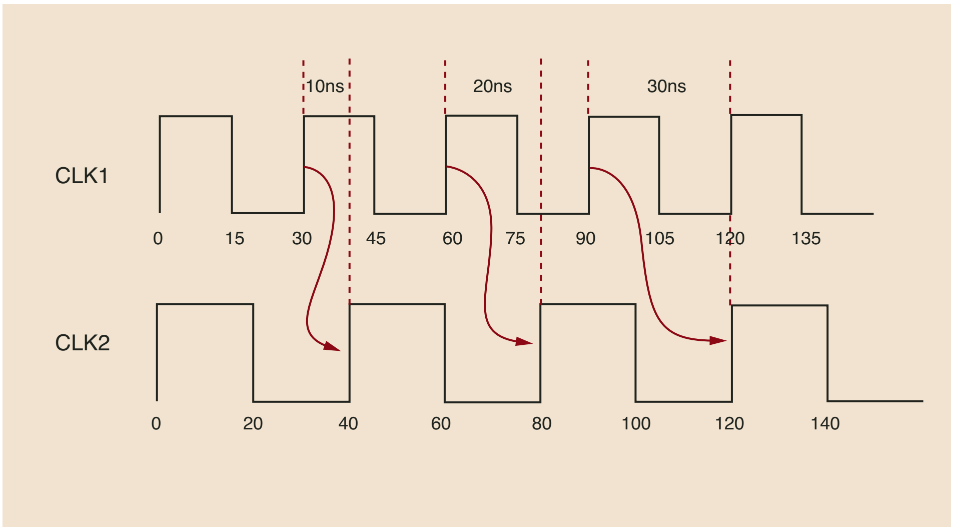


Figure 1 – Clock rolling for clocks in the same domain

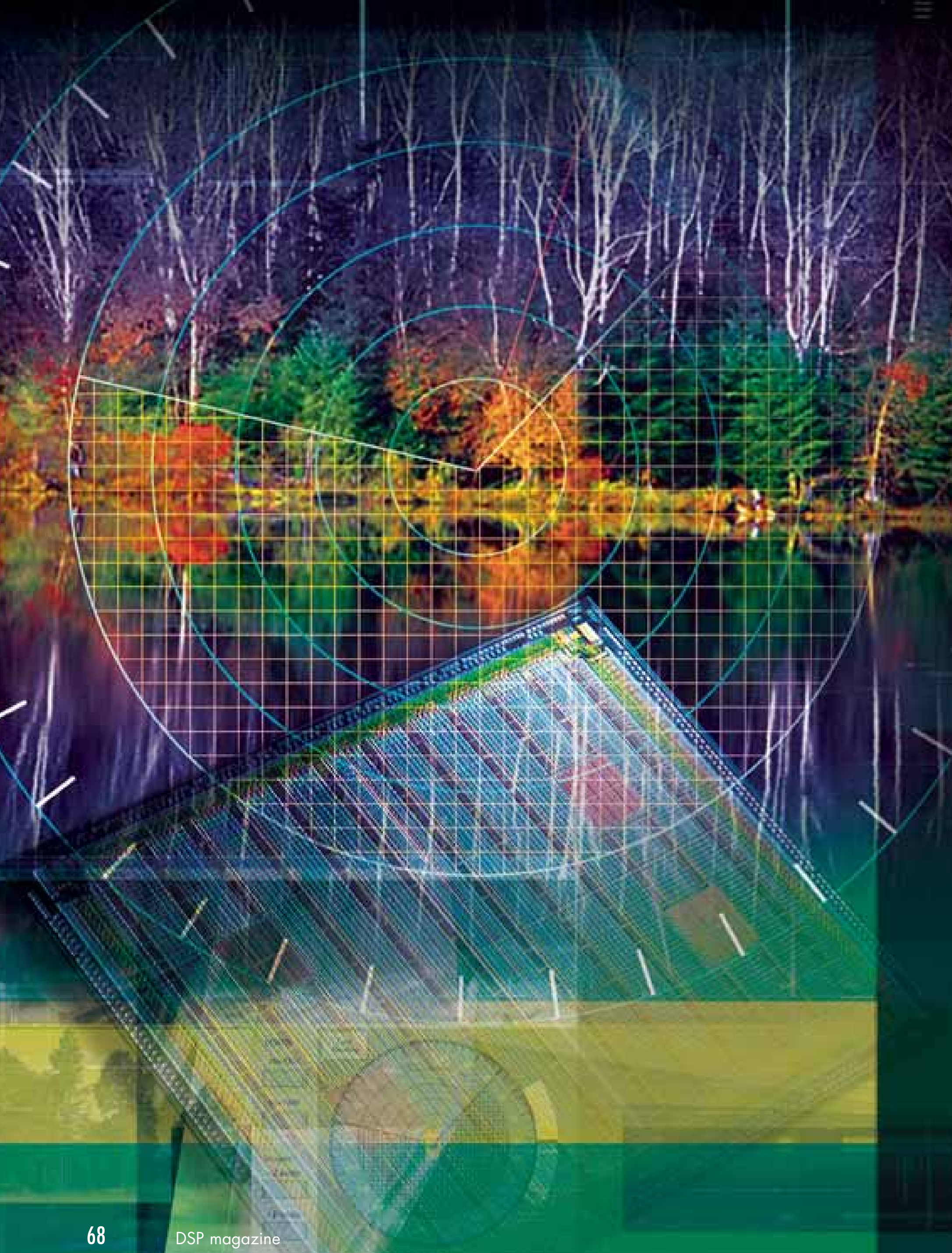
Conclusion

Because synthesis tools are operating at higher levels of abstraction, synthesis optimizations can have a dramatic impact on design performance. After parsing through the HDL behavioral source code and extracting known functions (arithmetic functions, multiplexers, and memories), synthesis tools then map these functions on the target architecture features.

The tools trade off area and performance based on design constraints and tools settings; these influence the use of optimizations such as replication, merging, re-timing, and pipelining. As a result, the right tools settings in synthesis can greatly increase productivity and time to market.

A Weapon Detection System Built with Xilinx FPGAs

Building a low-cost, low-power, lightweight, highly reliable weapon detection system.



by Randall Smith
Senior Electrical Engineer
Radiance Technologies, Inc.
rsmith@radiancetech.com

David Ellgen
Senior Electrical Engineer
Radiance Technologies, Inc.

The need to instantly detect the location and type of enemy gunfire is becoming more urgent than ever. Troops engaged in humanitarian, peacekeeping, or wartime operations are particularly vulnerable to fire from snipers who may operate among the civilian population with their tacit support.

The possibility of attacks in urban settings where firing locations are abundant, escape is easy, and echoes from closely spaced buildings make it difficult to identify the source of a shot mean that detection and location is a priority. With the escalation of sniper fire, myriad engagements in the arena of small-arms fire, and the increased use of rocket-propelled grenades (RPGs) comes the impetus behind the development of an infrared (IR) technology-based sniper-detection system. One of these systems is the Weapons Watch System (WWS), a sniper-detection system that Radiance Technologies Inc. is providing to the U.S. Armed Forces.

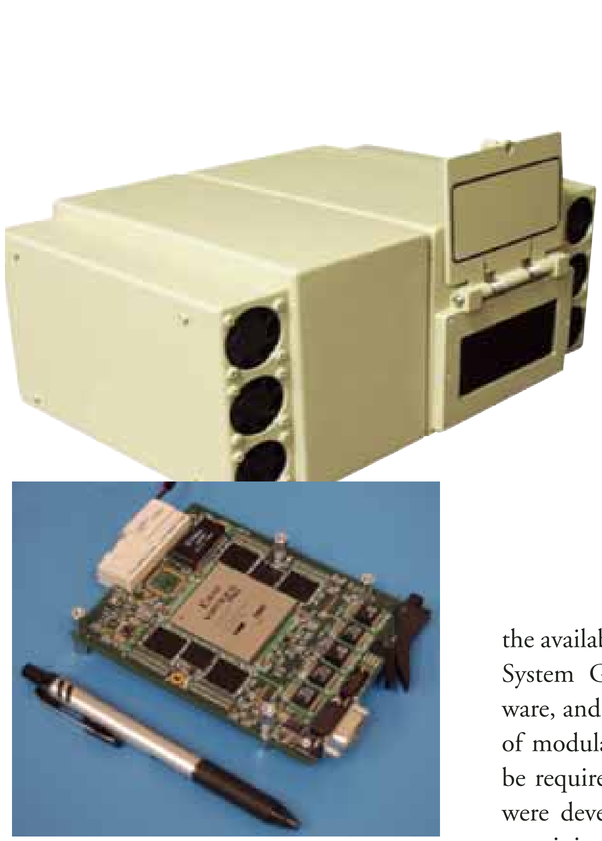


Figure 1 – System chassis and a single FPGA board

WWS provides the ability to detect weapon fire; determine azimuth, elevation, and range measurements; and identify the type of fire using high-performance image processing from IR sensors that detect the heat plume of a muzzle flash. This detection technology is deployed at fixed sites utilizing a commercial-off-the-shelf (COTS) system based on eight tightly coupled Power PC™ processors running concurrently to provide real-time image processing.

However, there is also a need for sniper-detection systems to be mounted on mobile platforms such as ground vehicles, helicopters, and unmanned aerial vehicles – or possibly worn by individual soldiers. This necessitated a miniaturized version of this technology, which prompted the next generation of the system combining all of this processing capability into a single FPGA.

Radiance met the requirements for a low-cost, lightweight, low-power, next-generation WWS with the use of the Xilinx® Virtex™-II Pro 2VP100 FPGA as the main processing engine. We chose the Virtex-II Pro device over competing FPGAs because of the hard-core processor, high-speed I/O, and wide array of IP cores available within the FPGA, as well as the support software. Additionally, Xilinx development tools allowed this complex processing system to be consolidated into a single FPGA, as pictured in Figure 1.

Parallel System Development

System development speed was a high priority for our design team. The system design was broken into several logical sections that best fit the available staff and design groups. Xilinx System Generator for DSP, ISE™ software, and EDK are well suited to this form of modularizing. Key modules that would be required for the development of others were developed first. Frameworks for the remaining modules were created to allow each engineer to develop their section in high-level MATLAB, ISE, or EDK environments.

An example of this technique was the development of a standard clocking and DCM resource module that could be used as either a black box for Simulink, PCORE for EDK, or used directly in ISE design tools. The design required the use of three separate clocking frequencies, with multiple phases of each frequency used in sections of the design. The first problem the team encountered was that EDK and System Generator do not directly support differential clock inputs.

The LVDS input primitives were instantiated directly in the VHDL code. When this code was imported into an EDK PCORE, a simple modification to the associated MPD file was all that was required to allow the design to properly compile. Clocking requirements for the high-speed I/Os, DCMs, and BuffG allocation were also explicitly defined in the clocking module and its associated UCF file. The development of this clocking module allowed our developers to consistently use defined clocking resources in any of the Xilinx design environments.

Algorithm Analysis, Development, and Testing

At first glance, we assumed that the design would require multiple FPGAs to replicate the high processing throughput of the eight processors in the original system. One real challenge for the team was to implement proven existing C code that ran on the COTS Power PCs into a System Generator bitstream model.

To better understand the timing requirements, we tested each section of the existing algorithms using the Xilinx ML300 prototyping board. Using System Generator

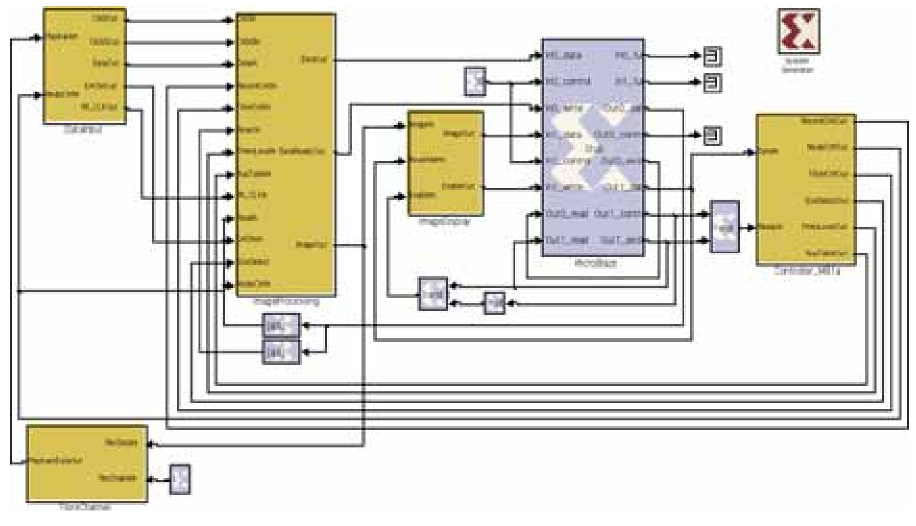


Figure 2 – High-level System Generator model

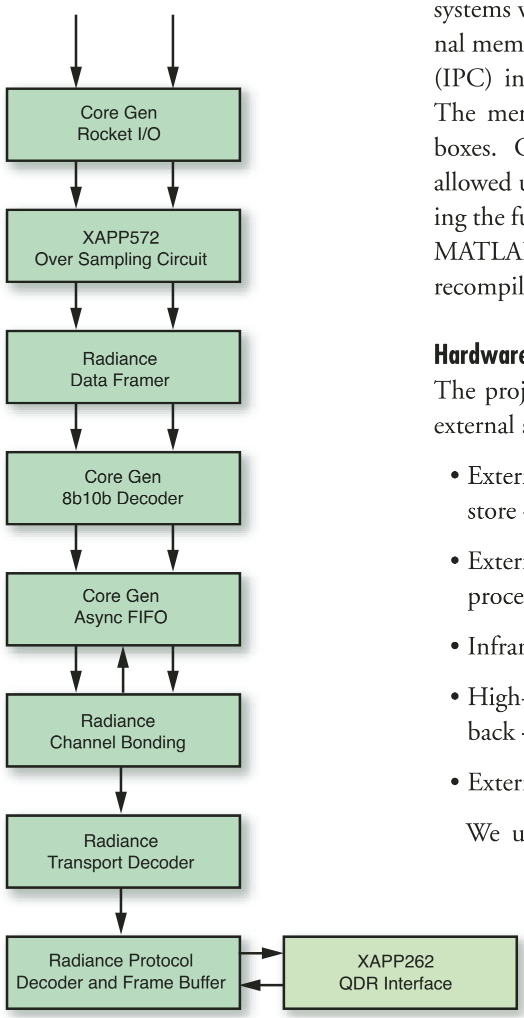


Figure 3 – Design approach

models to understand the processing, we employed several methodologies to test the algorithm's timing and resource utilization. We developed drivers to stimulate the algorithms and used System Generator's hardware-in-the-loop testing to quickly analyze the results.

As a result of the up-front testing, we determined that the entire design code could be placed into a single FPGA, which dramatically reduced the size, cost, and complexity of the system. The high-level model of the System Generator core is shown in Figure 2.

We used the System Generator's black-box module to encode modules that were better expressed in interfacing VHDL code. The WWS design required the use of several black-box modules that interfaced with other subsystems. One of these subsystems was a high-bandwidth QDR external memory for the image processing chain (IPC) in our System Generator PCORE. The memory is connected through black boxes. Configurable subsystem modules allowed us to use simulation code, replicating the functionality of the black box in the MATLAB environment without having to recompile the model.

Hardware Integration and Testing

The project required interfacing to several external subsystems, including:

- External memory for program and data store – DDR2
- External memory for high-bandwidth processing – QDR
- Infrared camera interface – HotLink
- High-speed data recording and playback – Fibre Channel
- External master clocks – Differential

We used ISE design tools to test and implement these interfaces. ISE software and the ChipScope™ Pro analyzer provided a rapid methodology for subsystem development and testing. Individual projects were set up to create cores that could be "wrapped" for importing into EDK or System Generator. By using this approach, our hardware engineers were able to realize these subsystem interfaces with minimal system integration time because they were completely tested using the ChipScope Pro analyzer to ensure system compliance. Several subsystem interfaces had extremely tight timing constraints; in many cases we used Xilinx Floorplanner software to manually place and route the components.

Meeting the physical size constraint of the WWS design necessitated that we eliminate as many discrete functions as possible. The Radiance team leveraged several Xilinx application notes to accomplish this. One obstacle was the video input to the processing engine, which comprised two 400 MHz serial streams. This normally requires the use of a dedicated PHY and a parallel interface to the processor. With the use of the high-speed I/O in the 2VP100 FPGA, as well as several Xilinx IP solutions, the design team was able to completely eliminate the need for an external PHY. The Radiance Technologies design approach to this is shown in Figure 3.

The team decided early on in the project that DDR2 would be the best memory choice for use with the embedded

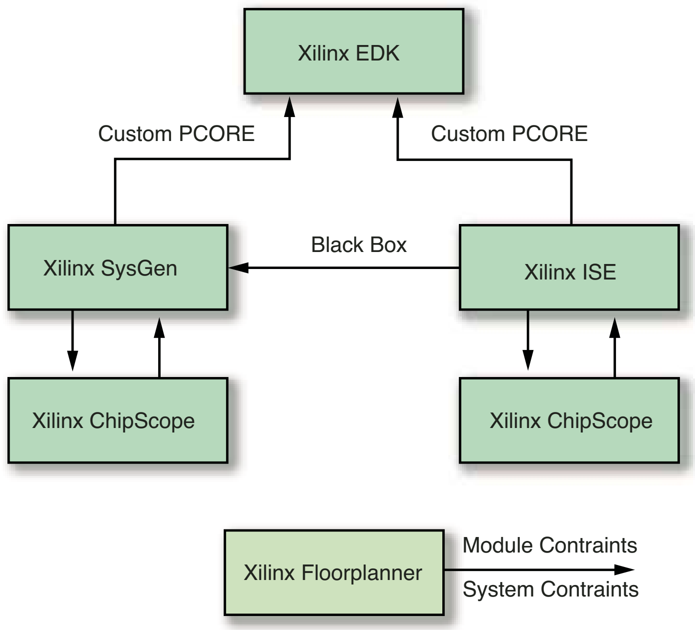


Figure 4 – High-level system layout

The Radiance engineering team exceeded the requirements of the next-generation WWS and accomplished the task on an accelerated schedule by leveraging the benefits of all Xilinx development tools – both hardware and software.

Power PCs. Because Xilinx lacked an available IP core for the selected memory part, and because of the short time-to-working-product schedule, Radiance decided to use Xilinx Design Services to provide the memory interface. The interface was delivered and integrated into the system in a timely manner.

FPGA System Integration Using EDK

The WWS team chose EDK to be the overall integration tool, allowing for quick generation of the download bitstream. The FPGA design was implemented using two Power PCs, three MicroBlaze™ processors, and several PicoBlaze™ processors integrated with the System Generator developed PCORE. The block diagram shown in Figure 4 details the integration methodology using EDK as the top layer.

The C programs executed on the Power PCs and MicroBlaze processors using a "roll-your-own" operating system. The two MicroBlaze processors provided an interface to the System Generator-rendered IPC. (During the design, the need to have two MicroBlaze processors connected to the IPC became apparent.)

A simple workaround for the System Generator limitation of only having one MicroBlaze processor per core is to design the core's single MicroBlaze interface to have multiple FSL buses. You can then connect the additional FSL links to other MicroBlaze processors using the add/edit cores function in EDK.

Two MicroBlaze processors perform dedicated processing, such as communications with the IPC, while the third served a Java-enabled website to display system diagnostics. The server uses a memory file system (MFS) library to maintain the website.

One problem we encountered was the 64 KB contiguous block RAM limit imposed on the OPB bus. Because the Web server required larger address space, we solved this problem by working with the Xilinx factory and field engineers to create "make files" that distributed the code and data over multiple block RAMs. This approach allowed us to create programs that were up to 256 KB in size running on the MicroBlaze processor.

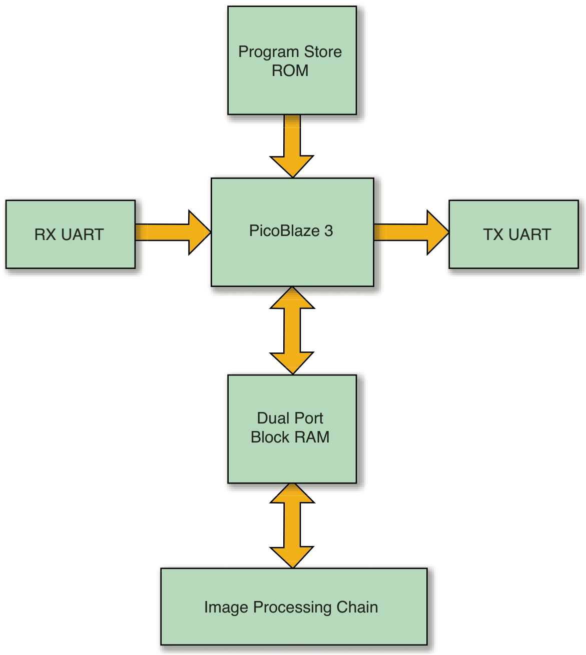


Figure 5 – Typical PicoBlaze layout

We implemented several PicoBlaze processors in the design to allow the IPC running in the fabric to communicate with external subsystems using standard serial protocols through transmit and receive UARTs. Figure 5 shows a typical connectivity of the PicoBlaze processors. These subsystems could then inject low-level input data, such as GPS position updates, directly into the metadata of the video stream.

Conclusion

The Radiance engineering team exceeded the requirements of the next-generation WWS and accomplished the task on an accelerated schedule by leveraging the benefits of all Xilinx development tools – both hardware and software. Using System Generator for the modeling and implementation of complex algorithms – versus hand-coding VHDL – significantly reduced system design and development time, saving several man years of work.

ISE and EDK design tools provided the framework that allowed a very complex system COTS to be placed on a single FPGA. By acquiring the Xilinx ML300 prototype boards, we were able to work out design issues such as critical timing constraints. Another advantage was the ability to determine maximum system loading early in the process.

Employing EDK as the main system integration tool is a viable option to consider, especially when several processors are employed in the design. Exploiting all Xilinx development tools, especially the System Generator-MATLAB programming model, will promote rapid understanding and development of complex system designs.

Radiance Technologies is a 100% employee-owned small business specializing in the application of emerging technologies to deliver government and commercial solutions. Radiance Technologies ranks #214 on the 2004 Inc. Magazine's 500, and ninth among defense contractors on the list.

DSP Design Flow

DSP10000-7-ILT (v1.0)

Course Specification

Course Description

The DSP Design Flow course provides the advanced tools and expertise you need to develop advanced, low-cost DSP designs. This intermediate course in implementing DSP functions focuses on learning how to use System Generator for DSP, as well as HDL design flow, CORE Generator™ software, design implementation tools, and hardware-in-the-loop verification. Through hands-on exercises, you will implement a design from algorithm concept to hardware verification by using Xilinx FPGA capabilities.

Level – Intermediate
Course Duration – 3 days
Price – $1500 USD or 15 Training Credits
Course Part Number – DSP10000-7-ILT
Who Should Attend? – System engineers/designers, logic designers, and experienced hardware engineers who are implementing DSP algorithms using MathWorks MATLAB and Simulink and using Xilinx System Generator for DSP
Prerequisites

- Fundamentals of MATLAB/Simulink and Xilinx FPGAs
- Basics of digital signal processing theory for functions, such as FIR (Finite Impulse Response) filters, oscillators and mixers, and FFT (Fast Fourier Transform) algorithms

Software Tools

- ISE™ 7.1i SP3
- System Generator for DSP
- EDK 7.1i SP2
- ChipScope™ Pro 7.1.3
- Mentor Graphics ModelSim
- The MathWorks MATLAB

After completing this comprehensive training, you will have the necessary skills to:

- Describe the different design flows for implementing DSP functions, with a large focus on System Generator
- Identify Xilinx FPGA capabilities and know how to implement a design from algorithm concept to hardware simulation
- Implement a design from start to finish by using System Generator
- Perform hardware-in-the-loop and HDL co-simulations and improve productivity
- Integrate the ChipScope™ Pro block in a design and analyze the design
- Develop a hardware co-simulation model using System Generator Board Description Builder
- Integrate a System Generator design as a peripheral in a MicroBlaze™ processor-based system

Course Outline

Note: Target architectures include Virtex™-4, Virtex-II Pro, and Spartan™-3E FPGAs.

Day 1: DSP Design Implementation Tools

- Introduction
- DSP Design Flows in FPGAs
- **Lab 1:** Creating a 12 x 8 MAC Using VHDL
- **Lab 2:** Creating a 12 x 8 MAC Using the Xilinx Architecture Wizard
- **Lab 3:** Creating a 12 x 8 MAC Using the Xilinx System Generator

Day 2: Digital Signal Processing Functions

- Digital Filtering
- **Lab 4:** Designing a FIR Filter
- HDL Co-Simulation
- **Lab 5:** MAC FIR Filter Verification Using Simultaneous Co-Simulations
- Looking Under the Hood
- **Lab 6:** Looking Under the Hood
- Controlling the System
- **Lab 7:** Controlling the System

Day 3: Digital Signal Processing Functions

- Multi-Rate Systems
- **Lab 8:** Designing a MAC-Based FIR Using the DSP48 Slice
- Advanced Features
- **Lab 9:** Designing Using the PicoBlaze Microcontroller
- **Lab 10:** Creating Parametric Designs
- **Lab 11:** Integrating the ChipScope Pro Analyzer
- **Lab 12:** A System Generator Design as an XPS Peripheral

Lab Descriptions

This lab-intensive class gives you hands-on experience by using System Generator for DSP to visualize, simulate, verify, and implement DSP algorithms in Xilinx FPGAs. The labs start at a descriptive level and build on each other. You should expect each successive lesson's challenges to increase. In addition, the labs included in the Advanced Features module provide you experience with other tools such as the ChipScope Pro analyzer and the Embedded Development Kit. System Generator for DSP 7.1 features are identified, including hardware and software co-simulation verification.

Register Today

Xilinx delivers public and private courses in locations throughout the world. Please contact Xilinx Education Services for more information, to view schedules, or to register online.

Visit **www.xilinx.com/education**, and click on the region where you want to attend a course.

North America, send your inquiries to registrar@xilinx.com, or contact the registrar at 877-XLX-CLAS (877-959-2527). To register online, search by **Keyword** "DSP" in the Training Catalog at https://xilinx.onsaba.net/xilinx.

Europe, send your inquiries to eurotraining@xilinx.com, call +44-870-7350-548, or send a fax to +44-870-7350-620.

Asia Pacific, contact our training providers at: www.xilinx.com/support/training/asia-learning-catalog.htm, send your inquiries to education_ap@xilinx.com, or call: +852-2424-5200.

Japan, see the Japanese training schedule at: www.xilinx.co.jp/support/training/japan-learning-catalog.htm, send your inquiries to education_kk@xilinx.com, or call: +81-3-5321-7772.

You must have your tuition payment information available when you enroll. We accept credit cards (Visa, MasterCard, or American Express) as well as purchase orders and training credits.

© 2005 Xilinx, Inc. All rights reserved. All Xilinx trademarks, registered trademarks, patents, and disclaimers are as listed at **http://www.xilinx.com/legal.htm**.
All other trademarks and registered trademarks are the property of their respective owners. All specifications are subject to change without notice.

LYRTECH
LYRTECH SIGNAL PROCESSING

Infinite possibilities™...

VHS-ADC-V4

HIGH-SPEED MULTICHANNEL ANALOG-TO-DIGITAL VIRTEX-4 CPCI CARD

The VHS-ADC-V4 is a high-speed, multichannel acquisition input/output card. It is equipped with eight phased-synchronous analog-to-digital converters that operate at a maximum rate of 105 MHz. It also comes equipped with a Xilinx Virtex-4 FPGA for high-speed processing. Finally, it comes with a mezzanine connector that allows you to add eight input channels, eight output channels, or SDRAM memory to store data.

YOUR CHALLENGE: TO CREATE STATE-OF-THE-ART EMBEDDED SOLUTIONS. THE **VHS-ADC-V4** LETS YOU DO JUST THAT.

MODULAR AND EXPANDABLE

Interfaces seamlessly with other powerful Lyrtech DSP/FPGA platforms.

STAR CONFIGURATION

Outstanding clock synchronization capabilities, with less than a 1-picosecond delay between channels.

VIRTEX-4 FPGA

The Virtex™-4 based family of platforms provides the most advanced logic, highest performance, highest density, and greatest memory capacity available.

HIGH-BANDWIDTH, MULTI I/O INTERFACES

The only FPGA-based platform with eight onboard inputs expandable to 16. Use the 1-GBps, full-duplex RapidCHANNEL ports to add processing boards such as the SignalMaster Quad or to add a RAID storage system.

MODEL-BASED DESIGN TOOLS

Fully integrated with the System Generator for DSP model-based design tools.

STAND-ALONE OPERATION

Onboard flash memory for the FPGA. Hot-swap capabilities and an I²C external port.

VHS RECORD AND PLAYBACK RAID

Equipped to support a RAID storage system with which you can record data continuously, at up to 200 MBps, for more than five hours.

GRAPHICAL USER INTERFACE

The VHS graphical interface built on the API SDK is ideal to control the VHS-ADC-V4 parameters in a Windows environment.

TYPICAL APPLICATIONS

- Advanced base stations
- Smart antennas, multichannel IF systems
- Channelizers and multiplexers
- Geolocation
- High-speed multichannel recording/playback
- High-speed test and measurement systems
- Radar, phased-array radar, SATCOM
- Medical, ultrasound, and other applications

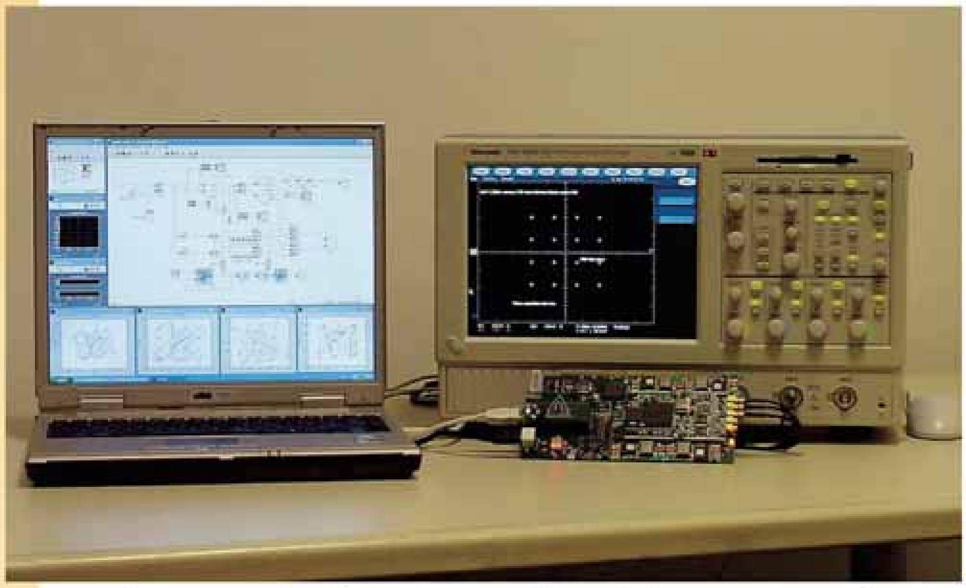
1 888 922-4644 www.lyrtech.com

Virtex™-4 SX 35 XtremeDSP™ Development Kit for Digital Communication Applications

Creating extremely high-performance digital communications signal-processing solutions can present significant challenges in both design complexity and time to market. The XtremeDSP™ Development Platform from Xilinx provides a complete development solution, so your designs will be faster, easier, and earlier to market.

Virtex-4 SX FPGAs feature up to 512, XtremeDSP slices, each capable of running at 500 MHz. This performance makes them the ideal co-processors for your DSP processors and the best way to increase your system performance by several orders of magnitude.

The XtremeDSP Development Platform — together with the Xilinx System Generator for DSP software and Xilinx DSP IP algorithms — provide the ideal development environment for developing Virtex-II Pro based signal-processing designs.


Hardware co-simulation with the XtremeDSP Platform and Xilinx System Generator for DSP

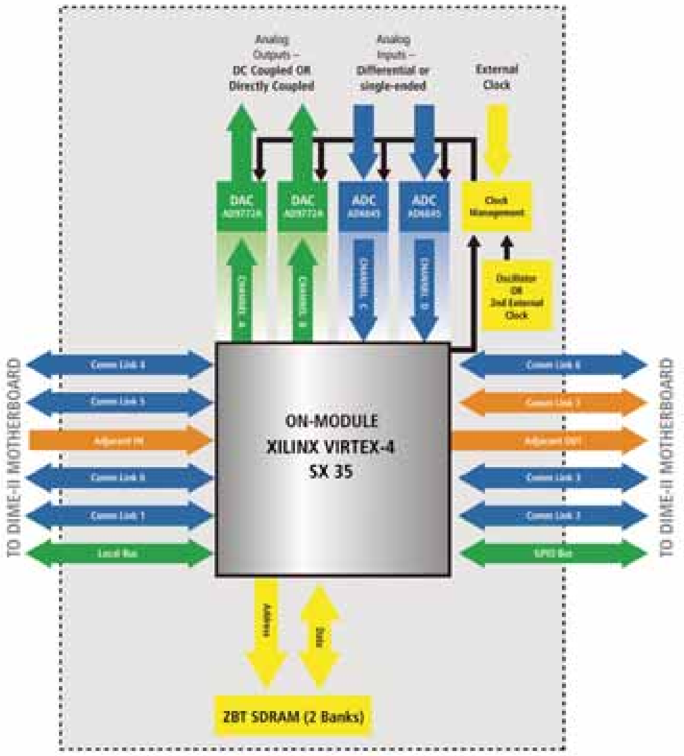
Your Complete Devleopment Platform

Developed with Nallatech, the Virtex-4 SX XtremeDSP Development Platform offers everything you need to create high-performance signal-processing designs more quickly and efficiently.

- **Exceptional Performance** – The dual-channel, high-performance ADCs and DACs, coupled with a user-programmable Virtex-4 SX-10 FPGA, make this platform ideal for implementing high-performance digital communication systems such as Software Defined Radios. The SX 35 FPGA features over 55,000 logic cells, 192 XtremeDSP slices.
- **Ease of Use** – Combining the Xilinx System Generator for DSP software tool and the XtremeDSP Development Kit provides an easy transition to using FPGAs for high-performance signal processing—from algorithm concept to hardware verification. The System Generator tool interfaces with MATLAB®/Simulink® and enables you to perform hardware co-simulation on the XtremeDSP Development Platform via PCI or JTAG. This provides simulation acceleration by an order of magnitude and allows you to debug and verify the design on the FPGA.
- **Comprehensive Support** – Reduce your time to knowledge with the Xilinx DSP Design Flow and DSP Implementation Techniques courses. You can also take advantage of senior DSP support engineer expertise on the Xilinx Hotline.

XILINX

Finish Faster with Xilinx DSP Design Solutions



Dime II module functional diagram

Hardware Platform Specifications

- XtremeDSP development board consisting of a motherboard ("BenONE-Kit Motherboard") populated with a daughter card ("BenADDA DIME-II Module").

BenONE-Kit Motherboard

- Supports the supplied BenADDA DIME-II module only
- Spartan-II™ FPGA for 3.3V/5V PCI or USB interface
- Host interfacing via 3.3V/5V PCI 32-bit/33-MHz or USB v1.1 interfaces
- Status LEDs
- JTAG configuration headers
- User 0.1-inch pitch pin headers connected directly to user programmable FPGA I/O

BenADDA DIME-II module

- Virtex-4 SX 35 user FPGA: XC4VSX35
- Two independent ADC channels: AD6645 ADC (14 bits up to 105 MSPS)
- Two independent DAC channels: AD9772 DAC (14 bits up to 160MSPS)
- Support for external clock, on-board oscillator, and programmable clocks
- Two banks of ZBT-SRAM (133 MHz, 512 Kx32 bits per bank)
- Multiple clocking options: internal and external
- Status LEDs

Also included with the XtremeDSP Platform

- External power supply (US Mains cable with separate UK, European or Australian Mains adapters)
- Wide ranging input (90 - 264Vac), multiple output, power supply, generating +5 Volts @ 5A, and +12 Volts @ 2A, -12 Volts @ 800mA
- USB v1.1-compatible cable, two meters long
- Five MCX-to-BNC cables for connecting to the ADC/DAC and external clock connectors
- PCI back-plate and two screws
- 2x BNC jack-to-jack adapters for use in loop-back configurations
- Large carrying case

XtremeDSP Installation Pack

- Nallatech FUSE Software CD — Enables control and configuration of FPGAs and provides tools to transfer data between the Kit and a host PC via a GUI or a C-based API

Applications

This multi-purpose board can be used for many digital communications applications including:

- Narrow-band systems (QAM demodulation, carrier timing recovery, channel coding)
- Spread-spectrum systems (e.g. chip rate processing, RACH, path profiling, TCC)
- Multi-carrier systems (e.g. OFDM, MIMO, TCC)
- And many more.

Take the Next Step

Purchase your XtremeDSP Platform at **www.xilinx.com/store**. For more information, visit **www.xilinx.com/dsp**. To learn more about the complete Nallatech platform offering, visit **www.nallatech.com**.
Price: $2,495

VIRTEX 4 — XILINX SYSTEM GENERATOR For DSP — NALLATECH The High Performance FPGA Solutions Company

Corporate Headquarters	European Headquarters	Japan	Asia Pacific	Distributed By:
Xilinx, Inc.	Xilinx	Xilinx, K.K.	Xilinx Asia Pacific Pte. Ltd.	
2100 Logic Drive	Citywest Business Campus	Shinjuku Square Tower 18F	No. 3 Changi Business Park Vista, #04-01	
San Jose, CA 95124	Saggart,	6-22-1 Nishi-Shinjuku	Singapore 486051	
Tel: (408) 559-7778	Co. Dublin	Shinjuku-ku, Tokyo	Tel: (65) 6544-8999	
Fax: (408) 559-7114	Ireland	163-1118, Japan	Fax: (65) 6789-8886	
Web: www.xilinx.com	Tel: +353-1-464-0311	Tel: 81-3-5321-7711	RCB no: 20-0312557-M	
	Fax: +353-1-464-0324	Fax: 81-3-5321-7765	Web: www.xilinx.com	
	Web: www.xilinx.com	Web: www.xilinx.co.jp		

FORTUNE 100 BEST COMPANIES TO WORK FOR 2005

XILINX® The Programmable Logic Company℠

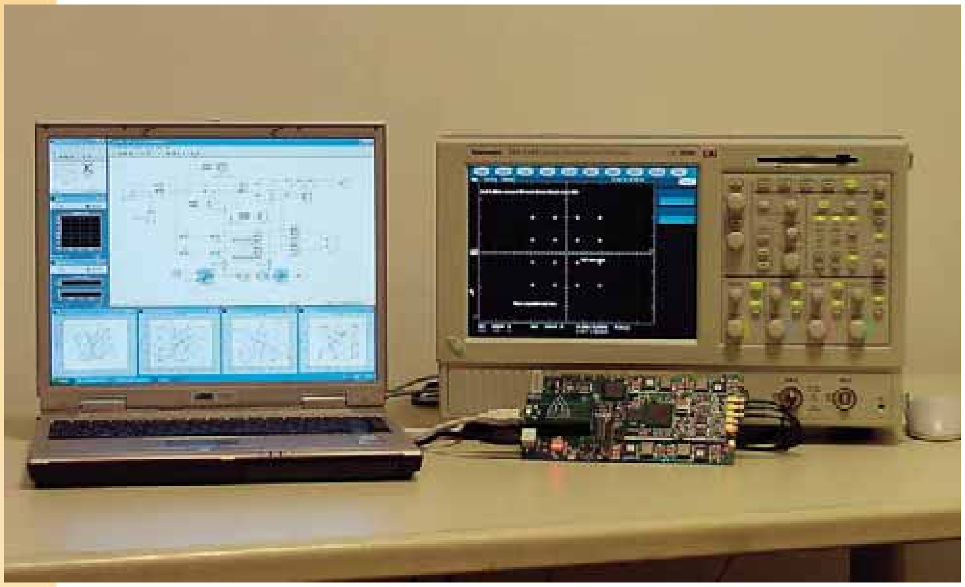
© 2005 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. All other trademarks are the property of their respective owners.

Virtex-II Pro™ XtremeDSP™ Development Kit for Digital Communication Applications

Creating extremely high-performance digital communications signal-processing solutions can present significant challenges in both design complexity and time to market. The XtremeDSP™ Development Platform from Xilinx provides a complete development solution, so your designs will be faster, easier, and earlier to market.

Virtex-II Pro FPGAs feature up to 444 embedded 18x18 multipliers, each capable of running at 300 MHz. This performance makes them the ideal co-processors for your DSP processors and the best way to increase your system performance by several orders of magnitude.

The XtremeDSP Development Platform — together with the Xilinx System Generator for DSP software and Xilinx DSP IP algorithms — provide the ideal development environment for developing Virtex-II Pro based signal-processing designs.

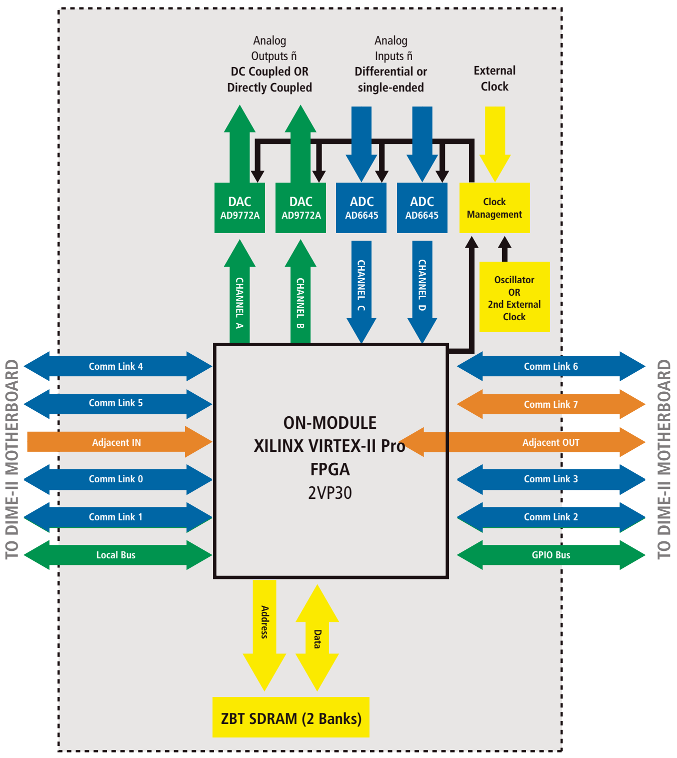

Hardware co-simulation with the XtremeDSP Platform and Xilinx System Generator for DSP

Your Complete Devleopment Platform

Developed with Nallatech, the XtremeDSP Development Platform offers everything you need to create high-performance signal-processing designs more quickly and efficiently.

- **Exceptional Performance –** The dual-channel, high-performance ADCs and DACs, coupled with a user-programmable Virtex-II Pro FPGA, make this platform ideal for implementing high-performance digital communication systems such as Software Defined Radios. The 2VP30 FPGA features over 30,000 logic cells, 136 embedded 18x18 multipliers, and an integrated PowerPC™ 405 processor.
- **Ease of Use –** Combining the Xilinx System Generator for DSP software tool and the XtremeDSP Development Kit provides an easy transition to using FPGAs for high-performance signal processing—from algorithm concept to hardware verification. The System Generator tool interfaces with MATLAB®/Simulink® and enables you to perform hardware co-simulation on the XtremeDSP Development Platform via PCI or JTAG. This provides simulation acceleration by an order of magnitude and allows you to debug and verify the design on the FPGA.
- **Comprehensive Support –** Reduce your time to knowledge with the Xilinx DSP Design Flow and DSP Implementation Techniques courses. You can also take advantage of senior DSP support engineer expertise on the Xilinx Hotline.

Finish Faster with Xilinx DSP Design Solutions



Dime II module functional diagram

Hardware Platform Specifications

- XtremeDSP development board consisting of a motherboard ("BenONE-Kit Motherboard") populated with a daughter card ("BenADDA DIME-II Module").

BenONE-Kit Motherboard

- Supports the supplied BenADDA DIME-II module only
- Spartan-II™ FPGA for 3.3V/5V PCI or USB interface
- Host interfacing via 3.3V/5V PCI 32-bit/33-MHz or USB v1.1 interfaces
- Status LEDs
- JTAG configuration headers
- User 0.1-inch pitch pin headers connected directly to user programmable FPGA I/O

BenADDA DIME-II module

- Virtex-II Pro user FPGA: XC2VP30-5FF1152
- Two independent ADC channels: AD6645 ADC (14 bits up to 105 MSPS)
- Two independent DAC channels: AD9772 DAC (14 bits up to 160MSPS)
- Support for external clock, on-board oscillator, and programmable clocks
- Two banks of ZBT-SRAM (133 MHz, 512 Kx32 bits per bank)
- Multiple clocking options: internal and external
- Status LEDs

Also included with the XtremeDSP Platform

- External power supply (US Mains cable with separate UK, European or Australian Mains adapters)
- Wide ranging input (90 - 264Vac), multiple output, power supply, generating +5 Volts @ 5A, and +12 Volts @ 2A, -12 Volts @ 800mA
- USB v1.1-compatible cable, two meters long
- Five MCX-to-BNC cables for connecting to the ADC/DAC and external clock connectors
- PCI back-plate and two screws
- 2x BNC jack-to-jack adapters for use in loop-back configurations
- Large carrying case

XtremeDSP Installation Pack

- Nallatech FUSE Software CD — Enables control and configuration of FPGAs and provides tools to transfer data between the Kit and a host PC via a GUI or a C-based API

Applications

This multi-purpose board can be used for many digital communications applications including:

- Narrow-band systems (QAM demodulation, carrier timing recovery, channel coding)
- Spread-spectrum systems (e.g. chip rate processing, RACH, path profiling, TCC)
- Multi-carrier systems (e.g. OFDM, MIMO, TCC)
- And many more.

Take the Next Step

Purchase your XtremeDSP Platform at **www.xilinx.com/store**. For more information, visit **www.xilinx.com/dsp**. To learn more about the complete Nallatech platform offering, visit **www.nallatech.com**.
Price: $2,495

VIRTEX-II PRO   XILINX SYSTEM GENERATOR For DSP   NALLATECH The High Performance FPGA Solutions Company

Corporate Headquarters
Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Tel: (408) 559-7778
Fax: (408) 559-7114
Web: www.xilinx.com

European Headquarters
Xilinx
Citywest Business Campus
Saggart,
Co. Dublin
Ireland
Tel: +353-1-464-0311
Fax: +353-1-464-0324
Web: www.xilinx.com

Japan
Xilinx, K.K.
Shinjuku Square Tower 18F
6-22-1 Nishi-Shinjuku
Shinjuku-ku, Tokyo
163-1118, Japan
Tel: 81-3-5321-7711
Fax: 81-3-5321-7765
Web: www.xilinx.co.jp

Asia Pacific
Xilinx Asia Pacific Pte. Ltd.
No. 3 Changi Business Park Vista, #04-01
Singapore 486051
Tel: (65) 6544-8999
Fax: (65) 6789-8886
RCB no: 20-0312557-M
Web: www.xilinx.com

Distributed By:

FORTUNE 2005 100 BEST COMPANIES TO WORK FOR

XILINX® The Programmable Logic Company℠

© 2005 Xilinx Inc. All rights reserved. The Xilinx name is a registered trademark; Virtex-II Pro, Spartan-II, and XtremeDSP are trademarks, and The Programmable Logic Company is a service mark of Xilinx Inc. PowerPC is a trademark of International Business Machines Corp. in the United States, or other countries, or both. All other trademarks are the property of their owners.

Keep up with the flow!

Powerful

- Virtex II Pro FPGA Processing Engine
- Large SDRAM and very fast QDR-II banks

Flexible

- Four Serial Links operating up to 3.125 Gb/s (Optical/Copper)
- FPGA and Flash Programming Utilities for easy configuration

Innovative

- I/O and Pre-processing in a single PMC module
- VxWorks, Windows and Linux Host Software Support
- Parallel Front Panel I/O available using I/O modules

With PMC-FPGA03

I/O and Pre-processing modules

Processing and FPGA - I/O - Data Recording - Bus Analyzers

For more information, please visit
www.vmetro.com or call (281) 584 0728

VMETRO
innovation deployed

Industry's Highest DSP Performance, Now at New Low Cost Points

Xilinx Virtex™-4 FPGAs provide the performance and flexibility to tackle the most demanding DSP applications. These devices are ideally suited for high-performance signal-processing tasks traditionally serviced by an ASIC or ASSP. They enable you to create high-performance DSP engines that boost the performance of your programmable DSP system by performing complementary co-processing functions in applications such as digital communications,video/imaging, and more.

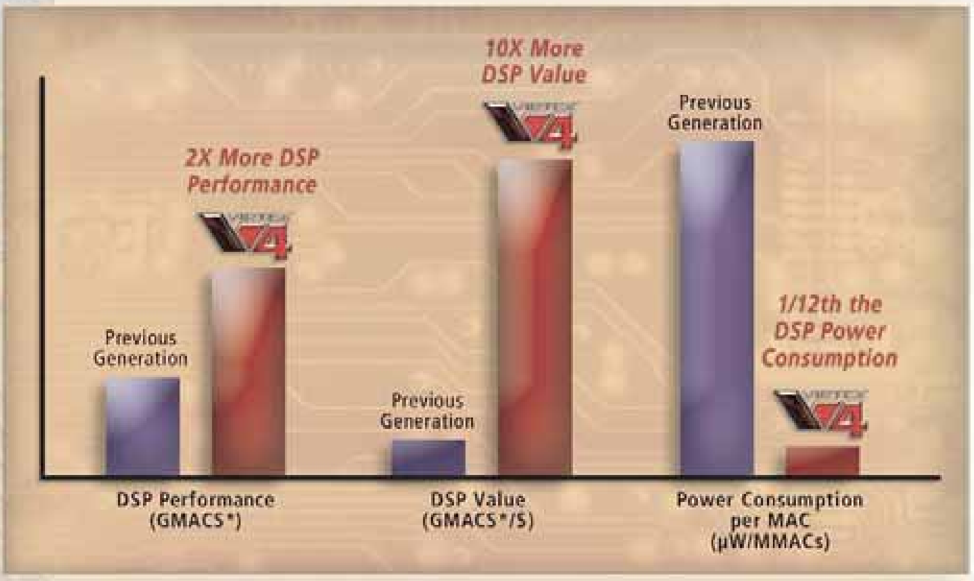
The Virtex-4 FPGA family is the newest and most powerful addition to the Xilinx XtremeDSP™ solution, providing blazing DSP performance with unrivalled economy. With up to 512 XtremeDSP slices operating at 500 MHz, these devices can implement complex tasks such as:

· Hundreds of IF-to-baseband down conversion channels

· 128X chip-rate processing for spread-spectrum systems

· High-definition H.264 and MPEG-4 encode/decode algorithms

The XtremeDSP solution accelerates your products' time-to-market through superior devices, design tools, intellectual property cores, and design services. This gives you the fastest means of designing, verifying, and deploying your DSP algorithms and systems in FPGAs.



XtremeDSP Slice Delivers Maximum Performance and Efficiency

The 500 MHz XtremeDSP slice delivers unmatched versatility, efficiency, and performance.

- Configure each XtremeDSP slice for over 40 DSP functions, such as multiply-accumulate, multiply, addition, and multiplexing
- Reduce DSP power consumption by 92% (23mw/100MHz) and save precious logic resources for other tasks
- Cascade multiple XtremeDSP slices at full system speed to build complex filters and multi-precision functions

Optimized Performance and Cost for Your DSP Applications

All three Virtex-4 platforms offer XtremeDSP capabilities. Choose the device that provides the optimal ratio of DSP performance for your unique application.

- Virtex-4 SX devices offer the most cost-effective implementation of ultra-high-performance DSP functionality, with the highest ratio of XtremeDSP slices — up to 512 slices delivering up to 256 GMACS* performance
- Virtex-4 LX devices offer ample XtremeDSP slices and add more logic, memory, and I/O resources
- Virex-4 FX devices add embedded PowerPC™ processors and RocketIO™ multi-gigabit transceivers

Easiest-to-use Design Solutions for FPGA-based DSP

Xilinx and its partners provide complete solutions for rapid DSP development and implementation.

- Reduce design time with System Generator for DSP
- Implement fast, highly optimized algorithms with a rich DSP IP library
- Bring products to market faster with award-winning technical support and DSP services

* 18x18 bits, 48-bit accumulator

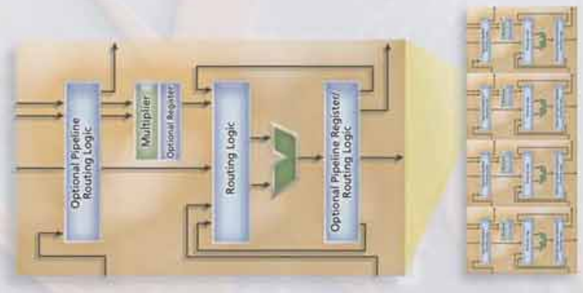
World's Highest-Performance, Best

Versatile 500 MHz XtremeDSP Slices

The Challenge: Implement high-performance DSP algorithms more cost-effectively.

The Virtex-4 Solution: Up to 512 new XtremeDSP slices

- 500 MHz throughput (256 GMACS overall performance) in 4VSX55
- 40+ arithmetic functions
- 1/12th the power compared to previous-generation FPGAs
- Directly cascadeable without loss in speed


Optional Pipeline Routing Logic
Multiplier
Optional Register
Routing Logic
Optional Pipeline Register/ Routing Logic

Digital Communication Systems

Whether you are working with spread-spectrum, multi-carrier, or narrowband communication systems, Virtex-4 FPGAs are the ideal choice.

Example: Wireless Base station

Narrowband
Spread Spectrum (eg. 3GPP2)
Multi-Carrier

Digital Radio Card

Spectrum Channelization

- DDC
- Polyphase Transform
- Matched Filter

Baseband Card

- QAM
- Adaptive EQ
- Carrier/Timing Recovery
- Reed Solomon
- Viterbi
- De/Interleaver

- Chip Rate
- RACH Rx
- Path Profiler
- TCC
- Viterbi

- OFDM
- MIMO
- TCC
- LDPC

Use the new XtremeDSP slices to efficiently implement:

- Digital Radio Functions
- Baseband Functions

XtremeDSP slices are used in the receive path (shown adjacent) and the transmit path for corresponding transmit functions.

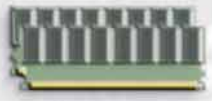
Powerful Serial and Parallel Interfaces

The Challenge: Need to interface to DSP processors, memory, and other systems

The Virtex-4 Solution: Extremely flexible I/O interfaces

DSP Processors & ADCs and DACs

- Serial RapidIO
- EMIF etc.
- LVDS etc.



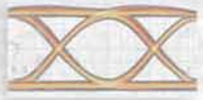
External Memories

DRAM

- DDR2, DDR
- SDRAM, RLDRAM II
- FCRAMII

SRAM

- QDRII, ZBT



System Interfaces

- Serial RapidIO
- PCI Express
- PCI
- HD-SDI
- Aurora
- CPRI, OBSAI

-Value FPGA for Signal Processing

Integrated Hard and Soft Microprocessors

The Challenge: Complex control and RTOS implementation.

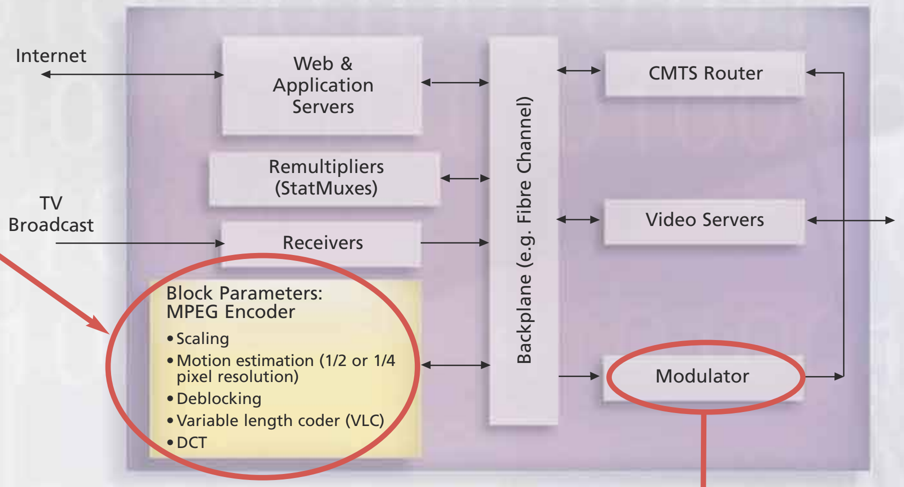
The Virtex-4 Solution: A broad selection of 8- to 32-bit microprocessor systems and operating system support (VxWorks, Integrity, Linux, etc.)

PowerPC
MicroBlaze
PicoBlaze

- Hard 32-bit IBM PowerPC 405 cores in FX platforms for implementing advanced frameworks such as Software Communications Architectures (SCAs) for software-defined radio applications
- Xilinx PicoBlaze™ and MicroBlaze™ soft microprocessors for control circuits

Video/Imaging and Broadcast Systems

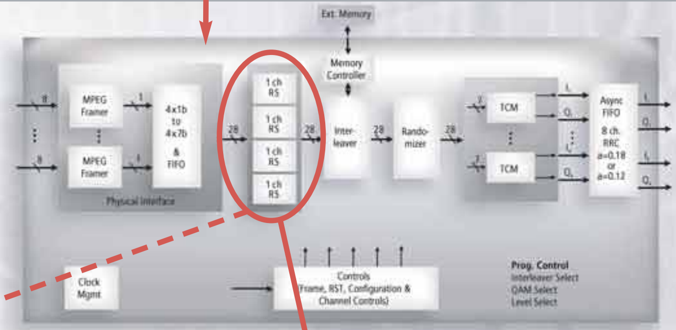
Example: Cable Head-end System

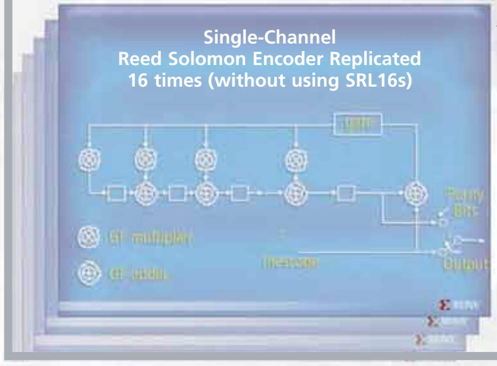

Internet
TV Broadcast
Web & Application Servers
Remultipliers (StatMuxes)
Receivers
Block Parameters: MPEG Encoder
• Scaling
• Motion estimation (1/2 or 1/4 pixel resolution)
• Deblocking
• Variable length coder (VLC)
• DCT
Backplane (e.g. Fibre Channel)
CMTS Router
Video Servers
Modulator

Compact Multi-Channel Designs using SRL16s

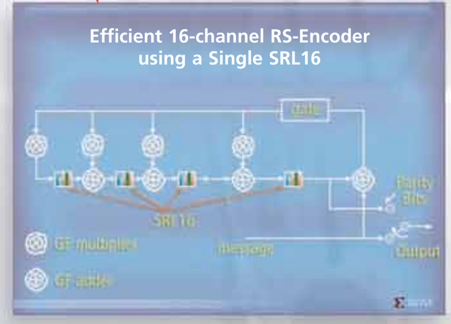
The Challenge: Keep cost and power down for multi-channel signal-processing designs

The Virtex-4 Solution: Unique SRL16s enable you to achieve very high compute density and make efficient use of logic slices.




Single-Channel Reed Solomon Encoder Replicated 16 times (without using SRL16s)
GF multiplier
GF adder
Parity Bits
Output

Using a single SRL16, a 16-channel Reed-Solomon encoder consumes only 10% of the silicon area compared to a design that simply replicates a single channel version 16 times.



Finish Faster with Xilinx DSP Design Solutions

Xilinx System Generator for DSP

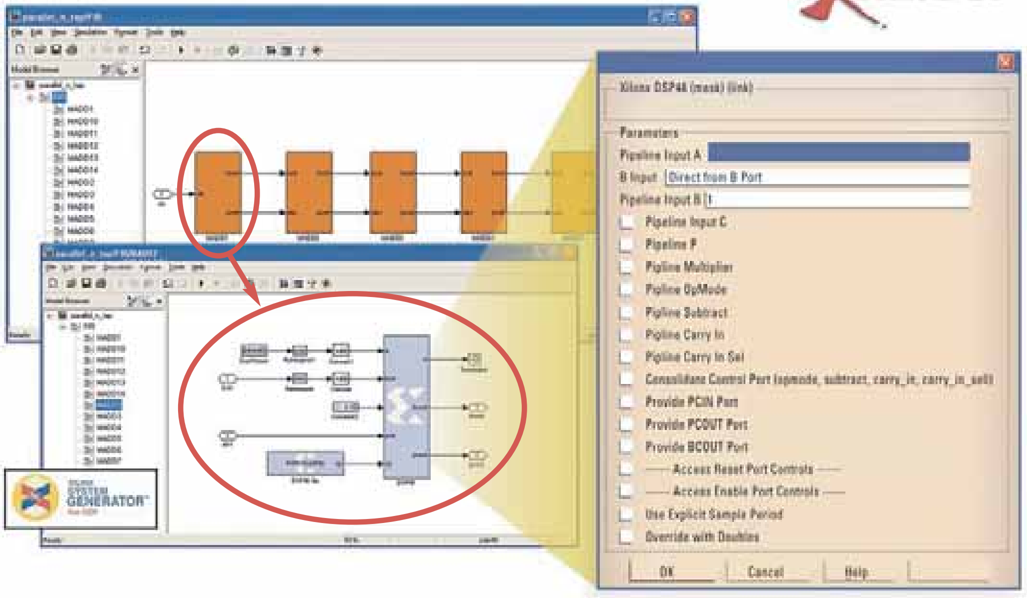
The industry's premier solution for FPGA-based DSP design enables you to:

- Generate high-performance DSP algorithms in FPGAs from a high level executable specification in Simulink®
- Accelerate simulations by orders of magnitude using your target hardware "in the loop" with Simulink or ModelSim®
- Import VHDL and Verilog modules directly into Simulink using a ModelSim co-simulation interface
- Specify state machine and other logic using MATLAB® code that is automatically compiled into RTL HDL

Pre-verified Signal Processing Algorithms as IP Cores

Xilinx and partners provide a range of DSP IP cores optimized for speed and cost:

- FEC: Reed-Solomon, Viterbi, TCCs, and others
- FFTs, Filters, and others
- Math functions: CORDIC, Multiplier, MACs, and others
- Video IP—Compression scaling and others
- Industry-standard DSP connectivity with Serial RapidIO, PCI, and EMIF



Xilinx DSP48 (mask) (link)

Parameters

Pipeline Input A

B Input: Direct from B Port

Pipeline Input B: 1

- Pipeline Input C
- Pipeline P
- Pipline Multiplier
- Pipline OpMode
- Pipline Subtract
- Pipline Carry In
- Pipline Carry In Sel
- Consolidate Control Port (opmode, subtract, carry_in, carry_in_sel)
- Provide PCIN Port
- Provide PCOUT Port
- Provide BCOUT Port
- —— Access Reset Port Controls ——
- —— Access Enable Port Controls ——
- Use Explicit Sample Period
- Override with Doubles

OK | Cancel | Help

DSP Services and Support for Virtex-4 FPGAs

DSP Education Services

Reduce your time-to-knowledge with public, private, and online courses including:

- DSP Design Flow (three-day course)
- DSP Implementation Techniques for Xilinx FPGAs (three-day course)

DSP Support Services

Ensure the success of your DSP project with award-winning technical support, including:

- Industry's best support web site
- Free DSP hotline support
- Platinum DSP hotline support
- Titanium on-site AE support

DSP Design Services

Reduce your project risk by allowing our engineers to help you with:

- System architecting
- FPGA implementation
- IP core modification
- Turnkey system design

Take the Next Step

Learn more about achieving blazing DSP performance with unrivalled economy. Visit us online at **www.xilinx.com/virtex4**

Corporate Headquarters
Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Tel: (408) 559-7778
Fax: (408) 559-7114
Web: www.xilinx.com

European Headquarters
Xilinx, Ltd.
Citywest Business Campus
Saggart,
Co. Dublin
Ireland
Tel: +353-1-464-0311
Fax: +353-1-464-0324
Web: www.xilinx.com

Japan
Xilinx, K.K.
Shinjuku Square Tower 18F
6-22-1 Nishi-Shinjuku
Shinjuku-ku, Tokyo
163-1118, Japan
Tel: 81-3-5321-7711
Fax: 81-3-5321-7765
Web: www.xilinx.co.jp

Asia Pacific
Xilinx, Asia Pacific
Unit 1201, Tower 6, Gateway
9 Canton Road
Tsimshatsui, Kowloon,
Hong Kong
Tel: 852-2-424-5200
Fax: 852-2-494-7159
E-mail: ask-asiapac@xilinx.com

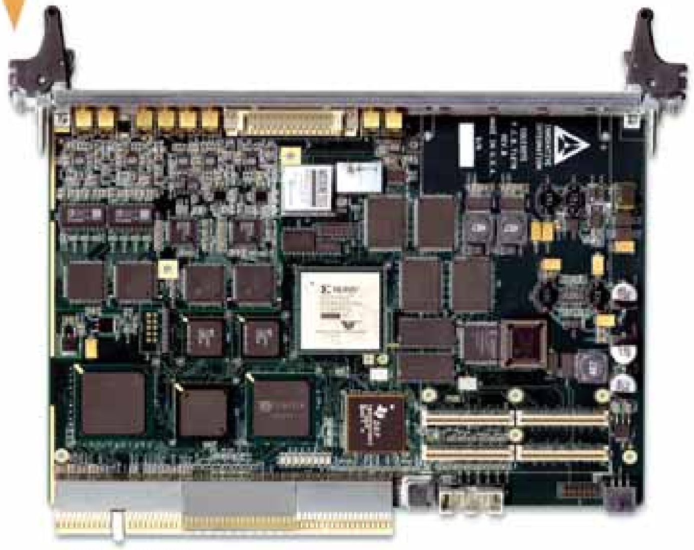
XILINX®
The Programmable Logic Company℠

Earth Friendly

FORTUNE® 2005
100 BEST COMPANIES TO WORK FOR

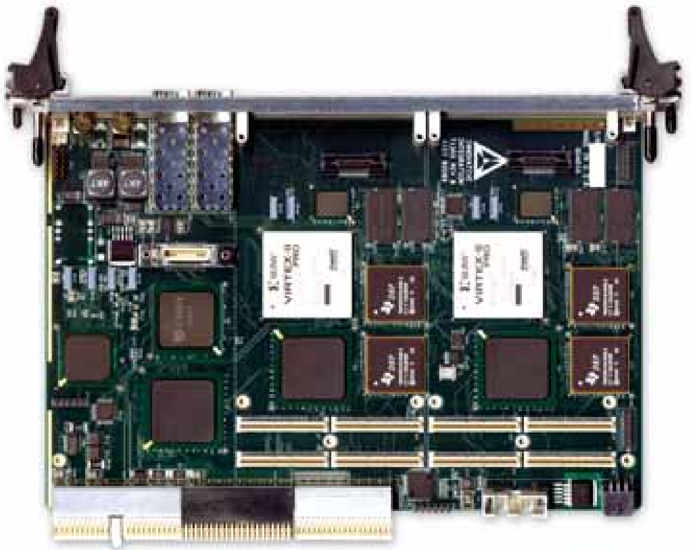
© 2005 Xilinx Inc. All rights reserved. The Xilinx name is a registered trademark; Virtex-4, RocketIO, XtremeDSP, PicoBlaze, and MicroBlaze are trademarks; and The Programmable Logic Company is a service mark of Xilinx Inc. Power PC is a trademark of International Business Machines Corporation in the United States, or other countries, or both. All other trademarks are the property of their owners.

DSP-Based RF Solutions!



Quixote

Quixote
- 1 GHz TMS320C6416 DSP
- 2 to 6 MGATE Virtex-II FPGA
- 32 MB SDRAM, 8 MB ZBT SBSRAM
- AD6645 & AD9764 Converters
- 64/32 bit CompactPCI, 66 MHz, 5V/3.3V
- Complex trigger modes w/HW event log
- PMC Site with Jn4 to FPGA DIO
- PICMG2.17 StarFabric Compliant



Quadia

Quadia
- 1 GHz TMS320C6416 DSP (x4)
- 64MB SDRAM per Processor
- Flexible communication mesh
- 64-bit/66MHz CompactPCI
- Two PMC Sites with Jn4 to FPGA
- External Data Port, up to 12Gbit/s
- StarFabric PICMG2.17 Compliant Port
- Two 4M Gate FPGAs for end-user code
- Up to two 2MB private DDR SBSRAM
- Up to two 128MB private DDR SDRAM
- Up to 512MB Global DDR SDRAM

Complete DSP-Based PMC/XMC RF Solutions

HSSIO
High-Speed Tx/Rx Serial I/O
Dual 1Gbit/sec Full Duplex

WB-NB
16 Channel Digital Receiver
Four A/D 125MHz

UWB
Ultra-Wide Digital Receiver
Dual A/D 210MHz

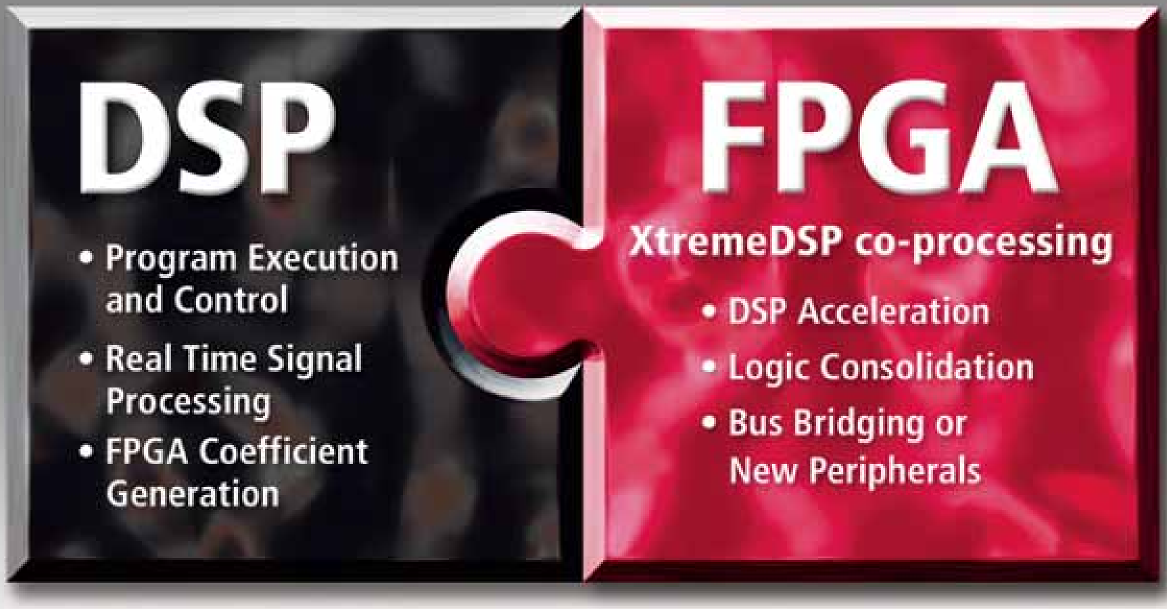
TX
Digital Transmitter
Four 500 MSPS DAC

Free Instant On-Line Pricing!

805.520.3300 phone
www.innovative-dsp.com

Innovative Integration
... real time solutions!

Turbocharge your DSP performance


DSP
• Program Execution and Control
• Real Time Signal Processing
• FPGA Coefficient Generation
FPGA
XtremeDSP co-processing
• DSP Acceleration
• Logic Consolidation
• Bus Bridging or New Peripherals

Achieve high-definition, higher frame rates or multiple video streams

When complimenting a TI DSP, Xilinx XtremeDSP co-processing offers the performance, versatility, and economy for today's high-end video and imaging applications. Whether it's high-definition, motion estimation, video scaling, or any number of compute intensive functions, a Xilinx Virtex-4 or Spartan-3/3E FPGA can boost your DSP performance. XtremeDSP co-processing delivers higher resolution, higher frame rate video processing than a standalone DSP processor, plus the ability to handle multiple video streams.

Reduce power and cost per channel in wireless systems

For implementing custom wireless functions, such as multi-carrier crest factor reduction (CFR), digital pre-distortion (DPD), MIMO and other advanced antenna processing, our FPGAs lower your costs and power per channel. With up to 256 GMAC/s performance, you have the advantage of offloading compute intensive tasks from a TI DSP to a Xilinx FPGA while increasing channel density in your wireless system.

Visit us at *www.xilinx.com/dsp/coprocessing* to learn more about the highest-performance DSP in the industry, and download your FREE evaluation copy of XtremeDSP software.

XILINX®

www.xilinx.com/dsp/coprocessing

FORTUNE® 2005
100 BEST COMPANIES TO WORK FOR

© 2005 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.

PN 0010886