HIGH-PERFORMANCE DSP EDITION

Xcell journal

Issue 62
Fourth Quarter 2007

SOLUTIONS FOR A PROGRAMMABLE WORLD

XtremeDSP Solutions: The Sky's the Limit

COVER

Processing Signals from Outer Space with BEE2

INSIDE

XILINX®

www.xilinx.com/xcell/

Support **Across The Board.**™

Add Functionality to Your Prototype Platform

XILINX®

SPARTAN-3

VIRTEX-4

VIRTEX-5

Available EXP Modules

- EXP to P160 Adapter
- EXP Prototype
- Analog Devices EXP Adapter Module
- Video EXP Module
- High-Speed ADC EXP Module
- High-Speed DAC EXP Module
- Interface EXP Module

EXP Baseboards

- Xilinx Spartan-3A DSP Starter Kit
- Xilinx Spartan-3 PCI Express Starter Kit
- Xilinx Virtex-4 FX60 PCI Express Board
- Xilinx Virtex-5 LX Development Kit
- Xilinx Virtex-5 LXT PCI Express Board
- Xilinx Virtex-5 SXT PCI Express Board

Avnet Electronics Marketing introduces the EXP specification – a versatile expansion interface for FPGA development boards that allows designers to add application specific daughter cards and easily connect to FPGA I/Os.

Key EXP Features

- Royalty free, public domain specification
- Full and half card options
- 168 (full module) and 84 (half module) user I/O
- Single-ended and differential pair signaling
- High-performance: > 100 MHz single-ended and > 300 MHz differential

Learn more about EXP at www.em.avnet.com/exp

AVNET® electronics marketing

Avnet Green Initiative

Supplier Authorized Distributor

Enabling success from the center of technology™

800-332-8638
www.em.avnet.com

IS YOUR CURRENT FPGA DESIGN SOLUTION HOLDING YOU BACK?

FPGA Design | Ever feel tied down because your tools didn't support the FPGAs you needed? Ever spend your weekend learning yet another design tool? Maybe it's time you switch to a truly vendor independent FPGA design flow. One that enables you to create the best designs in any FPGA. Mentor's full-featured solution combines design creation, verification, and synthesis into a vendor-neutral, front-to-back FPGA design environment. Only Mentor can offer a comprehensive flow that improves productivity, reduces cost and allows for complete flexibility, enabling you to always choose the right technology for your design. To learn more go to **mentor.com/techpapers** or call us at **800.547.3000.**

DESIGN FOR MANUFACTURING + INTEGRATED SYSTEM DESIGN
ELECTRONIC SYSTEM LEVEL DESIGN + FUNCTIONAL VERIFICATION

Mentor Graphics®

THE EDA TECHNOLOGY LEADER

INNOVATION

Leadership above all…

Xilinx brings your biggest ideas to reality:

Virtex™-5 FPGAs — Highest Performance. Leading the industry in performance and density, the Virtex-5 family offers multiple platforms optimized for logic, serial connectivity, DSP and embedded processing. Plus our Virtex™-5 EasyPath™ FPGAs give you a conversion-free, cost-reduction path for volume production.

Spartan™-3 Generation FPGAs — Lowest Cost. A unique balance of features and price for high-volume applications. Multiple platforms allow you to choose the lowest cost device to fit your specific needs.

CoolRunner™-II CPLDs — Lowest Power. Unbeatable for low-power and handheld applications, CoolRunner-II CPLDs deliver more for the money than any other competitive device.

ISE™ Software — Ease-of-Design. With *Smart*Compile™ technology, users can achieve up to 6X faster runtimes, while preserving timing and implementation. Highest performance in the fastest time — the #1 choice of designers worldwide.

Visit our website today, and find out why Xilinx products are world renowned for leadership . . . *above all*.

Get started quickly with easy-to-use kits

Order online at www.xilinx.com/getstarted

XILINX®

www.xilinx.com

At the Heart of Innovation

A New Era of Signal Processing

It is no accident that Xilinx® FPGAs serve an increasingly vital role in the design and development of today's most demanding digital signal processing (DSP) systems. Superior performance, system-level cost, power efficiency, faster time to market, and unrivaled flexibility are the hallmarks of FPGA-based DSP designs – value propositions that have found increasingly appreciative reception among leaders in markets like the communications industry.

Driven by the global demand for higher quality, higher bandwidth, and inexpensive wired and wireless communications of voice, computer, and video data, the number and complexity of new communications standards has grown exponentially. This is in large part because of the need for interoperability and data exchange across myriad layers of legacy and next-generation networks. Keeping pace with these standards and the extremely critical price/performance/power ratios they pose has been anything but trivial for system vendors. Nonetheless, the flux continues to produce opportunities for industrious innovators willing to tackle the challenge.

In the dynamic markets served by high-performance DSP solutions, the inherent flexibility of the FPGA equates to:

- Faster time to market with leading-edge algorithmic solutions and standards implementations
- Lowered operational costs with easy-to-perform remote adaptation to unforeseen environmental and functional changes
- Reduced capital expenditures due to extended life cycles for existing designs
- Easy migration to keep pace with changing customer demands and market requirements

In This Issue

This high-performance DSP edition of *Xcell Journal* offers several interesting – and in some cases out of this world – applications of high-performance FPGA-based DSPs. For example, the cover story shows how several UC Berkeley astronomers programmed the BEE2 platform using a programming environment that leverages Linux, Xilinx System Generator, and Xilinx EDK for advanced radio telescope applications. Other articles discuss the growing ease of programming an FPGA-based DSP design, interesting system-design challenges using development platforms for FPGA-based image processing applications, analyzing different hardware/software partitioning schemes to boost wireless subsystem performance, designing and verifying the CABAC functionality of an H.264 video encoder with System Generator for DSP, and an efficient implementation of audio algorithms in programmable logic.

As Jeff Bier, president of BDTI, notes, "Today, FPGAs play an increasing role in a wide range of DSP applications. We expect this trend to continue over the next several years." Xilinx sees an ever-broadening horizon for high-performance DSP processing solutions in the demanding digital communications, multimedia video and imaging, and defense systems markets. And Xilinx XtremeDSP™ solutions offer support every step of the way, with advanced DSP-optimized FPGA devices, hardware and software development tools, development kits, and evaluation platforms, as well as comprehensive training and support to ensure your success.

We hope you enjoy this issue.

Forrest Couch
Publisher

Xcell journal

PUBLISHER Forrest Couch
forrest.couch@xilinx.com
408-879-5270

EDITOR Charmaine Cooper Hussain

ART DIRECTOR Scott Blair

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

ADVERTISING SALES Dan Teie
1-800-493-5551

TECHNICAL COORDINATOR Larry Caputo

INTERNATIONAL Piera Or, Asia Pacific
piera.or@xilinx.com

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

Yumi Homura, Japan
yumi.homura@xilinx.com

SUBSCRIPTIONS All Inquiries
www.xcellpublications.com

REPRINT ORDERS 1-800-493-5551

Xcell PUBLICATIONS
www.xilinx.com/xcell/

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

VIEWPOINT

Easy FPGA Development
If current trends continue, FPGAs may become easier to program than DSPs.

FEATURE

Prototyping Image Processing Applications
Development platforms for FPGA-based image processing applications present some interesting system-design challenges.

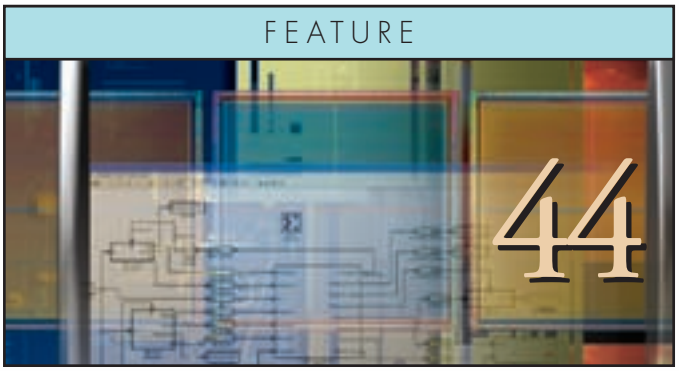
13 **Cover**

Processing Signals from Outer Space with BEE2
UC Berkeley astronomers programmed the BEE2 platform using a programming environment that leverages Linux, Xilinx System Generator, and Xilinx EDK.

FEATURE

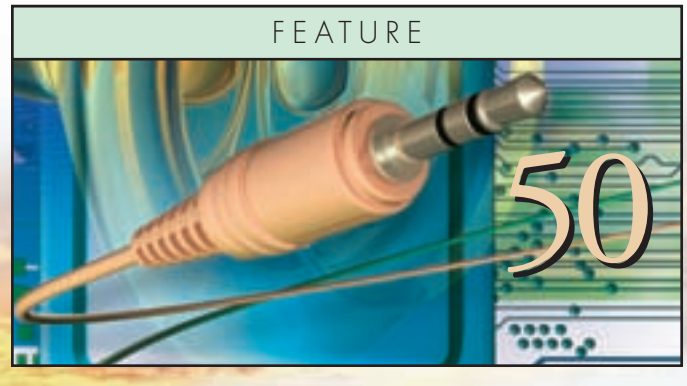
Boosting Wireless Subsystem Performance with FPGA Co-Processing
Analyzing different hardware/software partitioning schemes.

FEATURE



Integrating HDL Design and Verification with System Generator
Designing and verifying the CABAC functionality of an H.264 video encoder is easy with System Generator for DSP.

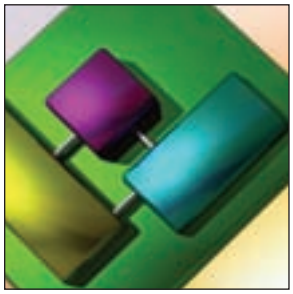
FEATURE

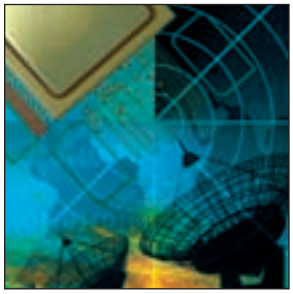


Audio Sample Rate Conversion in FPGAs
An efficient implementation of audio algorithms in programmable logic.

FOURTH QUARTER 2007, ISSUE 62

Xcell journal





VIEWPOINTS

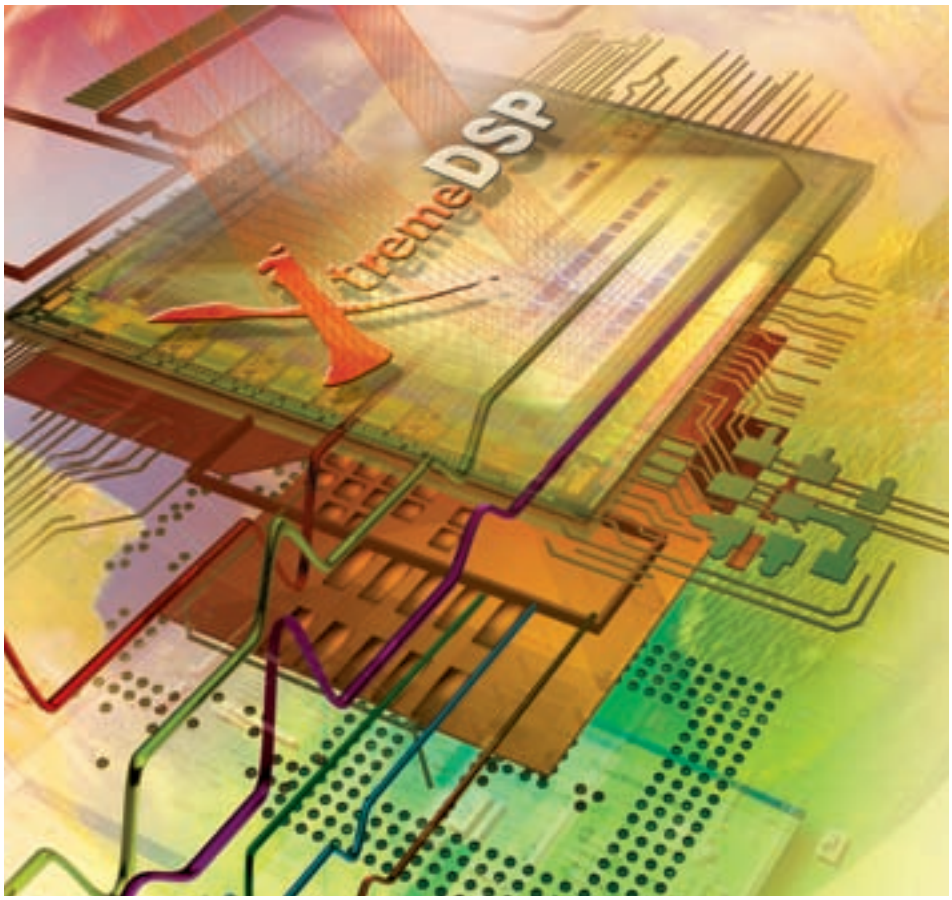
FEATURES

RESOURCES

Savoring the Highs and Lows

High performance and low power distinguish our signal processing solutions.

by Vin Ratford
Vice President and General Manager, PSG Division
Xilinx, Inc.
vin.ratford@xilinx.com



At around the time of our last Xilinx *DSP Magazine* release, I took a much needed (and planned) sabbatical. As I departed, Xilinx was preparing to launch the first family of devices targeted at lower performance signal processing applications. The Spartan™ DSP family debuted with the Spartan-3A DSP 3400 device and soon followed with the Spartan-3A DSP 1800. While watching from sidelines, the rollout looked great and sampling was underway for both devices upon their April 2007 introduction.

However, what really excited me was news that we beat the production release dates for both devices, delivering them a month ahead of schedule. In addition, the crisp execution of the follow-up Spartan-DSP low-power version delivered on a key value that is and will continue to drive high-performance signal processing decisions.

The LP version delivers a power-efficient specification that is an increasingly important value when meeting performance needs. DSP power efficiency refers to the amount of power consumed in performing signal processing calculations. DSP power efficiency measurements can be applied to systems, functions, building blocks, and common operations. The Spartan3A-DSP LP delivers 4.06 GMACs/mW at a maximum speed of 250 MHz when analyzing the common multiply-and-accumulate operation. Overall, the device delivers a 50% static power savings and a 70% savings while in suspend mode compared to non-low-power devices. This complements the dynamic power advantage inherent in the Spartan-3A DSP series given the integration of dedicated DSP circuitry (DSP48A slices).

With that execution theme and power efficiency idea in mind, in this, the high-performance DSP edition of *Xcell Journal*, we dig further into the uses of the XtremeDSP™ solution.

Xcell Journal is honored to again have DSP industry icon Jeff Bier from BDTI expand on why FPGAs excel in signal processing, using independent benchmark results described in his company's report. Also, we welcome *DSP DesignLine* Editor Kenton Williston, who shares his views on what the future may hold for FPGAs and digital signal processing. In addition, we thank all of our contributors in this issue for their outstanding articles and insight in the use of our XtremeDSP solutions in the areas of digital communications, video, and other application areas.

I am excited to be back at work in the Processing Solutions Group and look forward to delivering a host of new solutions to meet your demands. I encourage you to explore the latest advances in XtremeDSP solutions and solicit your input in how we can put this solution to work for you.

Easy FPGA Development

If current trends continue, FPGAs may become easier to program than DSPs.

by Kenton Williston
Site Editor
DSP DesignLine
kentonwilliston@yahoo.com

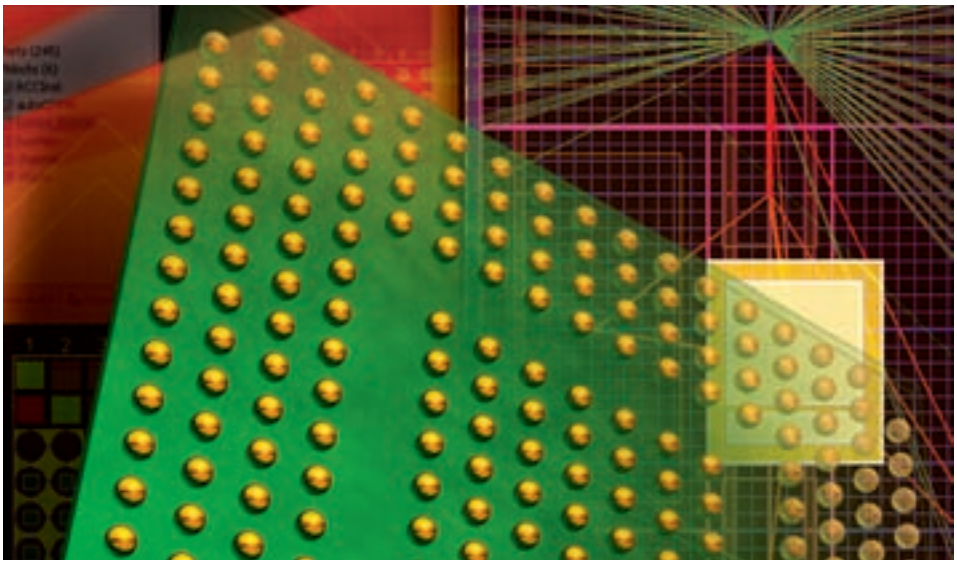
The evolution of FPGAs has paralleled that of DSPs in many ways. In the early days of DSP, programs were written entirely in assembly language. Today, DSP programmers typically start with a high-level language such as C and then optimize the "hot spots" using intrinsics or hand-coded assembly.

Similarly, FPGA designs were once crafted in Verilog or VHDL. Today, FPGA designers often start with a high-level description in an ESL tool, optimizing hot spots using IP blocks or hand-coded HDL.

This transition is a good thing. Five years ago, you couldn't build much of anything in an FPGA without hard-core FPGA skills. This kept FPGAs out of reach for most DSP programmers. Thanks to high-level tools, the average DSP programmer can now make the leap to FPGA design with a reasonable amount of effort – and produce reasonably optimal designs.

Good examples of this trend are the AccelDSP™ synthesis tool and System Generator for DSP tool, both from Xilinx, which integrate into the ubiquitous Simulink environment. These tools allow DSP developers to go straight from algorithm design to FPGA implementation in a familiar tool flow.

FPGAs are also following the path of DSPs in the area of intellectual property. Not long ago, DSP programmers had to write all of their own code. Today, DSP developers can turn to outside sources for "commodity" software. FPGA design has followed a similar path – designers who once had to develop applications on their own now have access to IP blocks ranging from FFTs to video encoders. These IP blocks make development faster and easier, giving DSP designers a way to translate their algorithm knowledge into FPGA designs. For example, in System Generator, designers can use the FFT IP block to try out different FFT variants and explore different trade-offs between performance and resource utilization.



Finally, FPGA families are growing – much in the same way as DSP processor families have grown – to include products that span a broad range of price and performance points. This makes FPGAs suitable for a wider range of applications. For example, Xilinx® products range from the Virtex™-5 FPGA, which is suitable for high-performance DSP applications, to the low-cost Spartan™-3A DSP device.

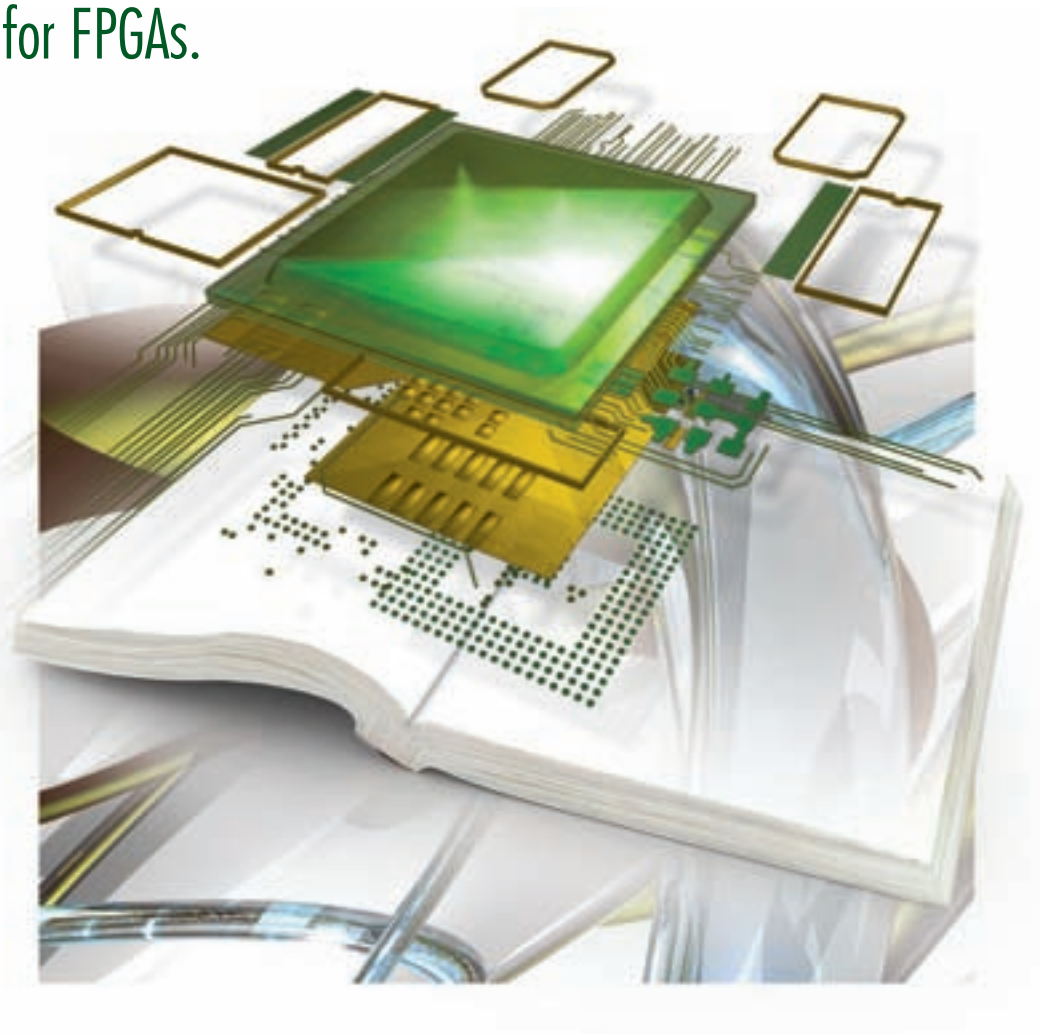
The growing product lines also let designers migrate from one FPGA to another. For example, a design that starts out on a smaller FPGA can migrate to a larger FPGA if the design needs more functionality.

Although the evolution of FPGAs has mimicked that of DSP in many regards, there is one area in which FPGAs are making a dramatic departure. As processors move to smaller process nodes, FPGA designers get more gates to play with. This makes it easier and easier to produce a design with the desired performance.

In contrast, process scaling has reached a point of diminishing returns for traditional single-core processor architectures, forcing vendors to turn to multi-core architectures. Although multi-core processors are very powerful, they are much more difficult to program than single-core architectures. Thus, DSPs are generally becoming harder to program while FPGAs are becoming easier to program. If this trend continues, we may see the day when FPGAs are easier to program than DSPs.

DSP Performance of FPGAs Revealed

Popular report provides independent DSP benchmark results for FPGAs.



by Jeff Bier
President
BDTI – Berkeley Design Technology, Inc.
bdti@bdti.com

FPGAs are increasingly considered for use as processing engines in high-performance DSP applications such as wireless base stations. In these applications, they may compete with DSP processors or work side-by-side with them.

With more choices, system designers need a clear picture of the signal processing performance of high-end FPGAs, both relative to each other and to high-end DSP processors. Unfortunately, the most commonly used performance figures are unreliable, confusing, and often contradictory.

For example, because DSP applications often rely heavily on multiply accumulate (MAC) operations, DSP processor and FPGA vendors sometimes use peak MACs per second as a simple metric for comparing digital signal processing performance. But MAC throughput is a lousy predictor of performance for FPGAs and DSPs alike. Let's examine a few reasons why.

Simple Metrics Fall Short

The MAC performance numbers for FPGAs often assume that the hard-wired DSP elements are operating at their highest possible clock rate. In practice, typical FPGA designs will operate at lower speeds. Plus, using hard-wired elements is not the only way to implement MACs on FPGAs; you can achieve additional MAC throughput using programmable logic resources and distributed arithmetic. This approach can yield higher MAC throughput than using hard-wired elements alone.

Yet another consideration is that typical DSP applications rely on many operations other than MACs. Viterbi decoding, for example, is a key DSP algorithm used in telecommunications applications that makes no use of MACs at all.

Another approach for assessing signal processing performance is to use common DSP functions like FIR filters. But this approach can have drawbacks too. One problem is that each vendor typically uses a different implementation of these functions – perhaps using different data widths, a different algorithm, or different implementation parameters such as latency. This means that results from different vendors are generally not comparable.

Furthermore, small kernel functions typically aren't effective for FPGA benchmarking because the way you would implement a function within a full FPGA application is often quite different from the way you would implement the function alone. (For processors, in contrast, these little benchmarks are usually pretty good at predicting overall DSP application performance.) Benchmarks implemented by processor or FPGA vendors often lack independent verification, making it difficult for engineers to make confident comparisons between devices.

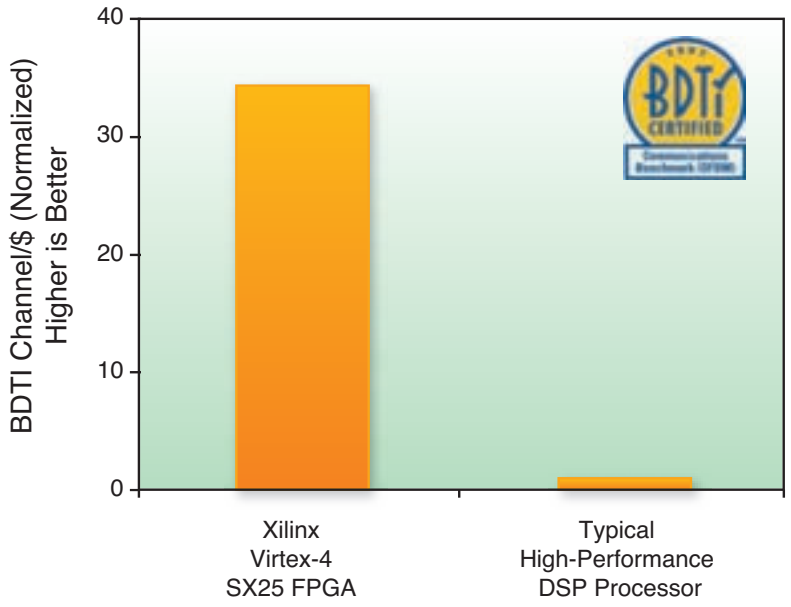


Figure 1 – BDTI Communications Benchmark (OFDM)
BDTI-certified cost-/performance-optimized results

Independent Benchmarks Fill the Gap

BDTI is the most respected source for signal processing benchmarks. Our benchmarks are used by dozens of semiconductor vendors and thousands of chip users to evaluate, compare, and select signal processing engines. BDTI has benchmarked the signal processing performance of processors for nearly 15 years, and has expanded its benchmarking activities to include FPGAs, multi-core chips, and other technologies.

Several years ago BDTI recognized the need for independent, accurate, apples-to-apples performance comparisons between FPGAs and processors targeting DSP applications. To address this need, BDTI developed a new application-oriented benchmark, the BDTI Communications Benchmark (OFDM), based on an orthogonal frequency division multiplexing (OFDM) receiver. This benchmark is designed to be representative of the "baseband" signal processing workloads increasingly found in communications equipment for applications such as DSL, cable modems, and wireless systems. It is suitable for implementation on FPGAs, DSP processors, multi-core chips, and many other signal processing engines, yielding apples-to-apples benchmark results.

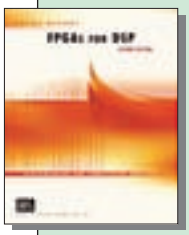
Last year, BDTI used the BDTI Communications Benchmark (OFDM) to evaluate several new high-performance FPGAs and DSP processors. The full results of this analysis are published in our report, "FPGAs for DSP: Second Edition," which has attracted considerable attention among DSP system designers. The report includes two sets of benchmark results: high-capacity results (optimized to support the maximum number of channels per chip) and low-cost results (optimized for lowest cost per channel). Figure 1 shows normalized, low-cost results for a Xilinx® Virtex™-4 SX25 FPGA and a typical high-performance DSP processor.

As Figure 1 demonstrates, BDTI's benchmark results provide a dramatic demonstration of the potential cost advantages of using FPGAs for high-performance DSP applications. The Virtex-4 SX25 device is more cost-effective by more than an order of magnitude over a typical high-performance DSP processor on this benchmark. This information is extremely valuable for system designers, who may have suspected that FPGAs could provide better chip-level cost/performance than DSPs in some applications but were not sure how much better. In the report, BDTI also compares cost/performance results for FPGAs from competing vendors.

Of course, benchmark results alone are not enough to answer the question of whether to use an FPGA in a new system design, or which FPGA to choose. Designers need to understand how their choice of processing engine will affect development flow, implementation effort, and system design. For this reason, BDTI's report explores the qualitative factors that influence the decision of whether to use an FPGA, a DSP, or both, and provides guidance on how to make an informed choice. The report also highlights the key questions that will affect the long-term success of FPGAs in high-end DSP applications, such as FPGA energy efficiency and the effectiveness of new high-level synthesis tools for FPGAs.

Conclusion

BDTI plans to conduct further analysis in these areas and we will continue to evaluate the signal processing capabilities of new FPGAs, DSPs, and massively parallel processors. The competition among signal processing engines is heating up, and BDTI will continue to provide the data and analysis needed to make confident choices.



BDTI's report, "FPGAs for DSP, Second Edition," is the authoritative source for independent FPGA signal processing benchmark results. *Xcell Journal* readers can request detailed report excerpts via e-mail; send your request to *fpga_samples@BDTI.com.*

For more information about "FPGAs for DSP, Second Edition" (including an order form), visit *www.BDTI.com/fpgas2006.* For more information about BDTI, visit *www.BDTI.com.*

Why use FPGAs for signal processing?

Explore the many benefits of using Xilinx high-performance DSPs in your digital communications, multimedia video and imaging, and defense systems applications.

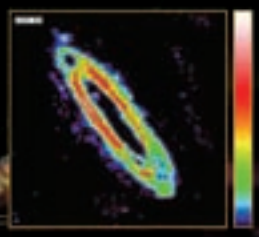
Xilinx offers a complete DSP processing solution, including devices, hardware and software development tools, reference designs, boards and kits, parameterizable algorithms (IP cores), training, and support.

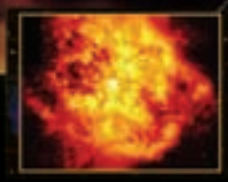
Download our XtremeDSP™ Solutions Selection Guide now and see how quickly you can get started.

www.xilinx.com/publications/prod_mktg/pn0010944-3.pdf

Processing Signals from Outer Space with BEE2

UC Berkeley astronomers programmed the BEE2 platform using a programming environment that leverages Linux, Xilinx System Generator for DSP, and Xilinx EDK.





by Chen Chang
CTO
BEEcube
chen@beecube.com

Bob Brodersen
Professor Emeritus
UC Berkeley
rb@eecs.berkeley.edu

John Wawrzynek
Professor
UC Berkeley
johnw@eecs.berkeley.edu

Dan Werthimer
SETI@home Chief Scientist
Director, Center for Astronomy Signal Processing
UC Berkeley
danw@ssl.berkeley.edu

Kees Vissers
Principal Engineer
Xilinx, Inc.
kees.vissers@xilinx.com

Modern radio telescopes look at the sky in frequency bands ranging from near zero to 11 GHz. After processing, these kinds of telescopes can provide information about very interesting phenomena, like colliding black holes, as shown in Figure 1.

These telescopes were historically constructed out of gigantic single-dish antennas such as the Arecibo telescope. However, because of the prohibitive cost of steel for constructing telescopes, Arecibo has remained the single largest collecting area telescope in the world for more than 50 years.

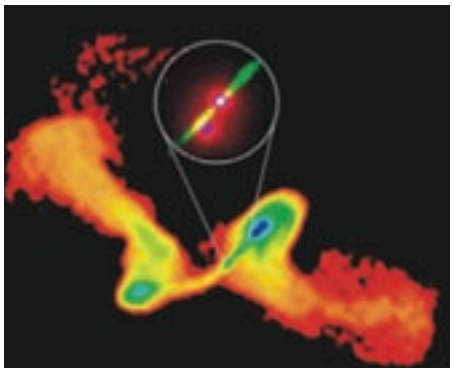


Figure 1 – Information about black holes gathered from a radio telescope image.

Since the invention of digital signal processing, radio telescope design using a large collecting area has moved to large arrays (hundreds to thousands) of small-diameter (6-12 m) antennas, where the steel cost can be balanced with the electronics cost. The antennas are physically spread across large geographic areas, providing extremely long and various length baselines and better angular resolution. The Allen Telescope Array near Hat Creek in Northern Californiia is an example of this modern kind of telescope. In this article, we'll focus on the signal processing with FPGAs employed in these telescopes by the SETI Institute and the Radio Astronomy Laboratory at UC Berkeley.

Large-N Small-D Antenna Array Correlator

Antenna arrays pose significant signal processing challenges. To form proper images, all signals from the antennas must be correlated with each other, thus requiring a computing complexity of $O(N^2)$. To achieve a 1-square-kilometer collecting area requires more than 8,000 antennas, each 12 meters in diameter.

To correlate the whole 11 GHz bandwidth, the computational requirement is on the order of 1,000 peta-operations per second, which is more than 3,000 times the capability of the current fastest supercomputer. Clearly, only a novel computing approach can achieve this computational throughput at a reasonable cost.

Furthermore, radio telescopes are typically designed to operate for more than 30 years. It is not necessary to invest in all of the electronics up-front to meet the bandwidth requirements; rather, the electronics can be upgraded every several years, gradually increasing the total observable bandwidth. This approach can achieve the best price/performance ratio overall by leveraging the exponentially decreasing cost of semiconductor technology.

Today's practice often involves specialized electronics with ad-hoc software. In this article, we'll explore the opportunity to leverage BEE2 systems and programming environments that comprise racks of reliable hardware and commodity routers. The back-end computers and the display part of the system can be commodity computers. Modern FPGA-based systems can cover the signal processing needs between the gigahertz range analog-to-digital converters (ADCs) and the back-end computers. These FPGA systems can be very reliable and cost-effective.

BEE2 System

Each compute module in the BEE2 system comprises five Xilinx Virtex™-II Pro 70 FPGA chips directly connected to four DDR2 240-pin DRAM DIMMs, with a maximum capacity of 4 GB per FPGA. The design organizes the four DIMMs into four independent DRAM channels, each running at 200 MHz (400 DDR) with a 72-bit data interface. Therefore, the peak aggregate memory bandwidth is 12.8 Gbps per FPGA. Each module uses four FPGAs for computation and one for control. The control FPGA runs Linux OS on the embedded PowerPC 405 to manage the compute processes on each user FPGA, as well as monitoring the overall module operation.

The four user FPGAs are directly connected on a 2D grid, with each link providing more than 40 Gbps of data throughput, as shown in Figure 2. The four down links from the control FPGA to each of the computing FPGAs provide as much as 20 Gbps per link. All off-module connections use the MGTs on the FPGA, four channel-bonded into 10 Gigabit Base-CX4 Ethernet interfaces. There are a total of 18 such CX4 interfaces on each BEE2 module – two from the control FPGA and four from each of the four compute FPGAs – for a total 180 Gbps full duplex bandwidth. For applications that require high bi-section bandwidth for random communication among many compute modules, the BEE2 system can be directly connected to commercial 10 Gigabit Ethernet switches. In addition, a 10 or 100 Base-T Ethernet

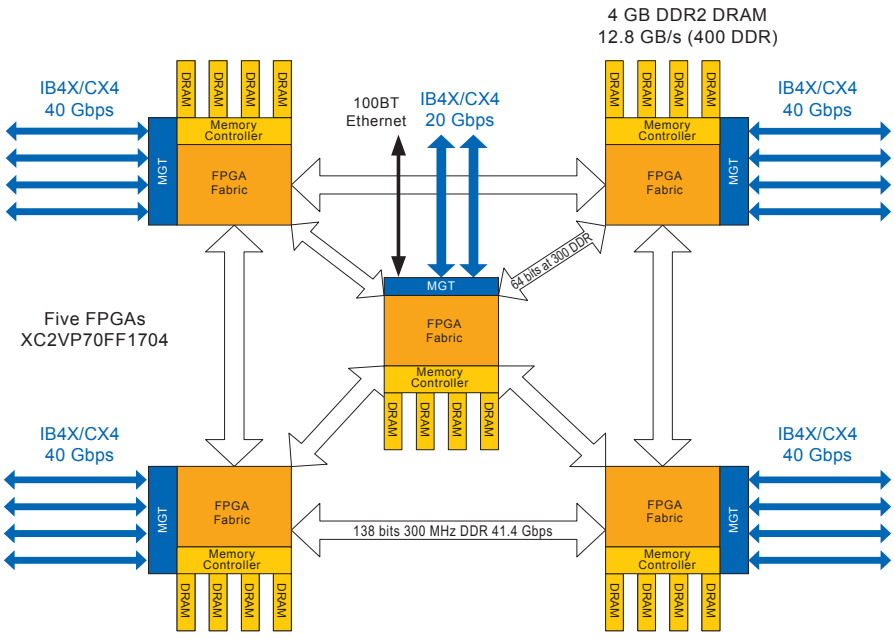


Figure 2 – Compute node connectivity

interface on the control FPGA provides an out-of-band communication network for the user interface, low-speed system control, monitoring, and data archiving.

BEE2 DSP Programming Environment

In the past decade, block diagram-based algorithm description methods have been gaining popularity, especially in the DSP domain. Software simulation environments, such as The MathWorks Simulink, provide parallel data flow execution models that match the nature of DSP data flow processing. With a rich set of domain-specific, high-level block libraries, algorithm designers can quickly construct complex DSP and communication systems. Xilinx® System Generator for DSP tools extend the modeling and simulation capability of Simulink to a direct mapping of core DSP algorithms to FPGA implementations.

However, most real-world DSP designs demand much more than just the core algorithms. With the latest generations of Xilinx FPGAs, many system-level components can now be integrated directly on a single FPGA, such as network interfaces, embedded microprocessors, memory, and I/O devices. Along with these hardware subsystems, an FPGA design is no longer just hardware design but rather a design that contains hardware and software. The software includes complex software OS and applications.

The BEE2 DSP programming environment, called BEE Platform Studio (BPS) and shown in Figure 3, provides an integrated design environment in Simulink that abstracts away the complexity of advanced FPGA hardware/software co-design. This environment was specifically created for BEE2 platforms. It enables algorithm designers to focus on core DSP algorithms, while it automatically generates the code and bit files for the complicated hardware and software subsystems. It is built on top of existing Xilinx tool flows. Where Xilinx System Generator provides an excellent block set for mapping DSP algorithms, the Xilinx Embedded Developers Kit (EDK) provides microprocessor and system integration, and ISE™ software provides the back end for logic synthesis, place and route, and bit generation for the hardware.

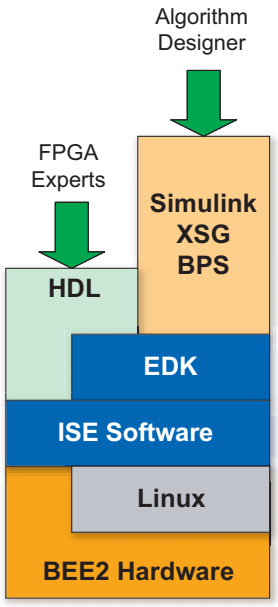


Figure 3 – BEE2 Platform Studio design abstractions

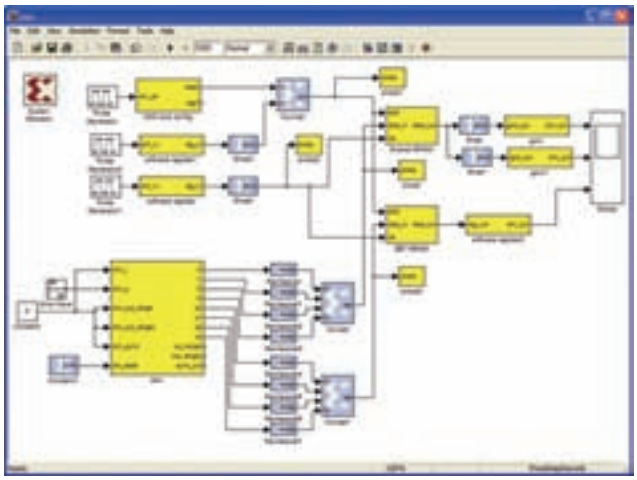


Figure 4 – BPS design example

A typical design in the BPS design environment starts with the core algorithm design in Simulink with Xilinx System Generator for DSP. From the end-user perspective, Simulink designs only exist in an idealized sandbox with the synchronous data flow execution model; all connections outside the core algorithm are virtually mapped through BPS interface block sets. A typical design is shown in Figure 4. The BPS block sets are created by FPGA experts to replace generic XSG "gateways" as interfaces between the core algorithm design in Simulink and the system-level devices.

A processor core, either in the form of a hard core (PowerPC 405) or soft core (MicroBlaze™ processor), is implicitly included in all BPS designs. The processor core can communicate with the user XSG design through software registers, FIFO, or shared memory. Users can specify the desired communication method by selecting the corresponding BPS blocks in Simulink. All external network, I/O, and memory devices are abstracted into Simulink data sources or sinks, with a simple FIFO abstraction.

For each of the supported FPGA board platforms, BPS framework provides a base system package as a complete Xilinx Platform Studio (XPS) project, and the corresponding Simulink BPS block set for all the external devices. Each base-system package includes the essential system device IP cores, initial hardware system configuration, and available software packages. The back-end implementation files for the user-selected external devices are dynamically generated by the BPS tools on top of the base package. These are then combined and linked with all necessary hardware connections and software device drivers.

Algorithm Overview

Radio astronomy typically observes phenomena that happened long ago and far away; hence the radio waves reaching Earth are essentially plain waves. When observing the same phenomenon with two or more radio telescopes that are physically separate from each other, each antenna receives the same wave fronts but at different times, because of the varying incision angles at each physical antenna location. The basic idea of correlation-based radio astronomy imaging is to reconstruct the wave front of interest by correlating received radio signals at different locations, and therefore at different delay values.

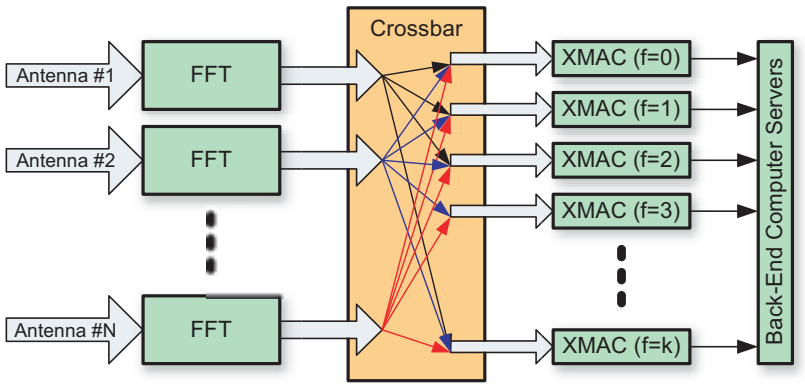


Figure 5 – N-antenna frequency division FX correlator overview

For a large number of antennas, the FX correlator scheme is typically used because of its computational efficiency. Each antenna signal is first transformed into the frequency domain through an FFT (fast Fourier transform). Next, the signals of several antennas are multiplied and accumulated (MAC) for each of the frequency channels independently. This basic correlator system is shown in Figure 5. The number of MAC computations is one per input sample – independent of the total number of frequency channels. The FFT computation in the FX correlator grows as $N \log_2 (N)$ with respect to the total number of antennas, while the MAC calculations grow as $\frac{1}{2} N(N \pm 1)$.

The first step of the correlation algorithm digitizes the analog signals from the antenna, then frequency transforms the digital signals into the frequency domain, which is also referred to as the F-engine. After I/Q digitization at 1 GSPS, a digital down conversion (DDC) block is used to tune each digital signal input to the band of interest, followed by a poly-phase filter bank (PFB) for pre-filtering and FFT. Because the signal-to-noise ratio (SNR) of the input signal is below unity, the output of the PFB is quantized into lower precision fix-point numbers (4-bit real and 4-bit imaginary in this case) such that both the overall correlation computation and network bandwidth utilization are more efficient. Finally, the packet formatter collects a number of frequency data for each bin into a single packet, with a time stamp to be transmitted through the 10 Gigabit Ethernet interface.

One of the most efficient X-engine implementation schemes uses a tiled structure to cross-correlate data packets from each antenna on a linear delay chain, as shown in Figure 6. In an *N*-antenna system, each X-engine contains exactly *N* delay elements; each is *m* data samples deep, corresponding to the *m* data samples in each packet. *N/2* number of MAC units compute the cross-correlations, with one dedicated MAC for auto-correlations. A multiplexer on each MAC unit chooses one of its inputs between the first and last data packet on the delay chain.

The data packets leave the packet receive buffer in sequential antenna order, shifted along the delay chain and correlated on the MAC units. The results are shifted left to the DRAM controller for long-term accumulations.

All $\frac{1}{2} N(N \pm 1)$ correlations are computed in exactly *N* time steps (each time step is *m* clock cycles) for the particular frequency channel in the data packets. The tile-based X-engine design makes it easy to partition a single X-engine across multiple physical FPGA chips. Multiple adjacent tiles can be grouped into each FPGA, and the data stream can simply flow from one FPGA to another. When correlating dual polarization antennas, each MAC unit computes full Stokes parameters in parallel; hence it requires four complex multiplies per clock cycle, or equivalently 16 real multiplies and eight additions. Finally, the four complex terms (eight real terms) are accumulated *m* times before shifting out to DRAM for long-term accumulation.

Scalability

Traditional implementation of FX correlators use direct wired backplanes and cables to construct the crossbar connections from the F-engine to the X-engine, which requires the whole hardware system to be globally synchronous, limiting the scalability of the system. In this new implementation, the time stamp on each frequency data packet effectively decouples the absolute sample time from the computing hardware clocks, and all X-engine computations can be processed when the relevant packets have arrived. Therefore, commer-

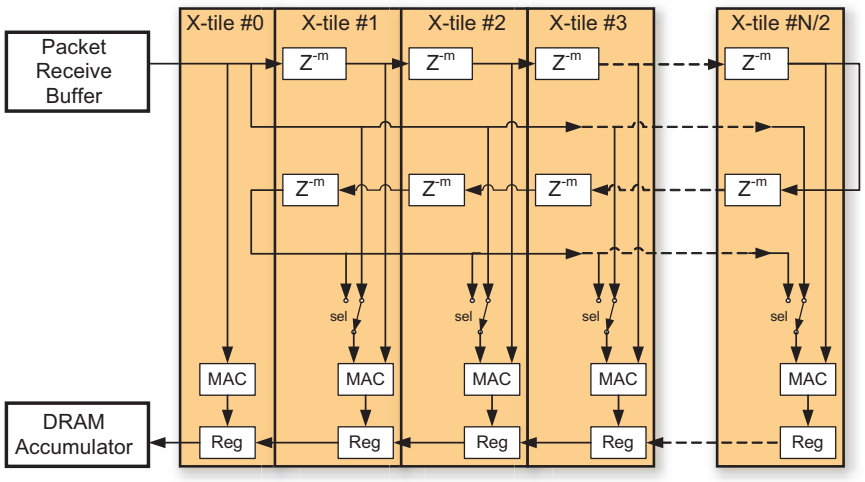


Figure 6 – Linear tile-based X-engine (simplified view)

cial network switches with varying packet latency can be used as virtual crossbar instead of hardwired backplanes.

An LTX implementation can fit a 256-antenna X-engine design in a single BEE2 FPGA running at 250 MHz. The total logic slice utilization including global control and memory interfaces is kept at around 80% of the maximum slices to allow sufficient room for high-clock rate routing. A 1,024-antenna X-engine can be implemented on a single BEE2 board by chaining the four user FPGA chips, each implementing one-fourth of the LTX tiles. The input buffers need to be implemented on the center control FPGA using external DRAM. At 256 KB per data packets per frequency bin, 4,096 frequency bins worth of data can be stored on a single 1 GB DIMM, allowing a maximum of more than 2 seconds delay variation from F-engines through the 10 Gigabit Ethernet switch.

The 10 Gigabit Ethernet switch bandwidth grows linearly with the number of antennas in the system. Because the number of X-engines and F-engines is the same, the crossbar switch essentially redistributes the output data packets from F-engines frequency bin-wise to each of the corresponding X-engines. In a given time step, F-engine output packets can be individually transmitted to unique X-engines on a one-to-one basis. Round-robin rotation of frequency bins from each antenna ensure against long-term congestions on the crossbar switch. As all F-engines are intrinsically synchronized to the same sampling clock used on the ADC boards, even temporary packet congestions should be rare.

Conclusion

So far, we have deployed several correlators using this solution, including a 200 MHz bandwidth, 32-anternna correlator at the Allan Telescope Array using four BEE2 modules. Several scientists have used a 16-antenna version in Green Bank, West Virginia, and in other radio astronomy projects throughout the world.

Because we designed the whole correlator using the BEE Platform Studio environment, the user effort of porting the design to future FPGA hardware platforms, such as the upcoming BEE3 system using Xilinx Virtex-5 FPGAs, is minimized to simply recompiling the new hardware platform from the Simulink design. The Virtex-5 FPGA can achieve more than four times the compute throughput at half the price, so rapid design migration is critical to achieving the goal of reaching a collecting area of 1 square kilometer with more than 8,000 antennas.

After more than six years of research at UC Berkeley, the BEE2-related software and hardware development has been commercialized via a startup company – BEECube Inc. – to further support a wider range of applications, from high-performance DSP to other emerging bioinformatics applications. For more information on the programming environment and future hardware system development, please contact Chen Chang at *chen@beecube.com.*

Acknowledgements

The BEE2 project's radio astronomy application development is in collaboration with the SETI@Home and Serendip (Search for Extraterrestrial Radio Emissions from Nearby Developed Intelligent Populations) projects at the UC Berkeley Space Science Laboratory (Dan Werthimer, Aaron Parsons, Henry Chen), as well as the UC Berkeley Radio Astronomy Laboratory (Melvyn Wright, Dave MacMahon, Matt Dexter, Don Backer). Xilinx has generously donated FPGAs, software tools, and engineering assistance.

Many thanks to all the hard work by peer students and staff members of the BEE2 team: Pierre-Yves Droz, Greg Gibeling, Nan Zhou, Yury Markovskiy, Zohair Hyder, Adam Megacz, Alexander Krasnov, Hayden So, Kevin Camera, Brian Richards, Dan Burke, Ken Lutz, and Susan Mellers. The BEE2 project is funded by the GSRC and C2S2 Focus centers, part of the FCRP, a Semiconductor Research Corporation program, National Science Foundation grant numbers CNS-0551739 and CNS-0403427, and the BWRC and its sponsor companies.

Prototyping Applications with the Spartan-3A DSP Starter Platform

The new Spartan-3A DSP Starter Platform includes expansion capabilities for low-cost and easy-to-use application development.

by Jim Beneke
Vice President, Global Technical Marketing
Avnet
jim.beneke@avnet.com

Have you ever had a great idea for the next killer FPGA application but struggled to find the right prototyping hardware that would allow you to prove out your concept? You're not alone. Finding the perfect development platform is not easy. Boards are often designed more for demonstration than development, leaving designers with little flexibility and limited access to FPGA I/O pins.

With the recent introduction of the Xilinx® Spartan™-3A DSP FPGA family, Xilinx has created a unique, low-cost, prototype-friendly starter platform to ease your application development. Through the EXP expansion interface included on the board, you can add application-specific daughtercards to customize the board's feature set for your prototype needs.

In this article, I'll review the EXP standard, showing you why it's FPGA-friendly and able to meet your most demanding expansion needs. We'll look at several of the EXP modules currently available and see how they can be used to easily create video, embedded, and communications processing applications around the Spartan-3A DSP. You'll see how the EXP specification, the Spartan-3A DSP Starter Platform, and the EXP add-on modules can combine to quickly and cost-effectively provide you with a useful prototype system.

The EXP Expansion Standard

The new Spartan-3A DSP Starter Platform (*www.xilinx.com/s3adspstarter*) includes a standard set of features as shown in Figure 1. Parallel flash and SPI memory are provided for configuration, while DDR2 memory is available for high-performance mass storage. A Gigabit Ethernet PHY and serial port support standard communications links, while clocks, switches, LEDs, and some general-purpose user I/O round out the board interfaces. The remaining FPGA user I/O pins – 168 in total – route to two connectors. These connectors are configured to meet the EXP expansion slot standard, which was designed specifically for FPGA development boards.

Most industry-standard buses, such as PCI, PMC, PCMCIA, or PC-104, implement an address and data bus structure, which is ideal for processor-based systems but limiting for the wide range of FPGA applications. Instead, FPGA development boards require a more generic, universal I/O structure where you can define I/Os as they are needed. Therefore, the EXP specification provides a great deal of flexibility by limiting the number of fixed signal definitions, allowing a more free-form I/O assignment as determined by your end application.

The EXP specification defines a 120-pin connector, with 84 user I/Os and a mix of power and grounds. The standard EXP configuration uses two connectors in a full EXP module configuration for a total of 168 user I/Os. Half EXP module formats are also available, using just a single EXP connector. Two half EXP modules can be connected to a full EXP-configured baseboard.

The Spartan-3A DSP FPGA SelectIO™ interface supports many popular single-ended and differential standards. Table 1 shows the number of pins available at each EXP connector, along with a breakdown of the single-ended and differential pairs sup-

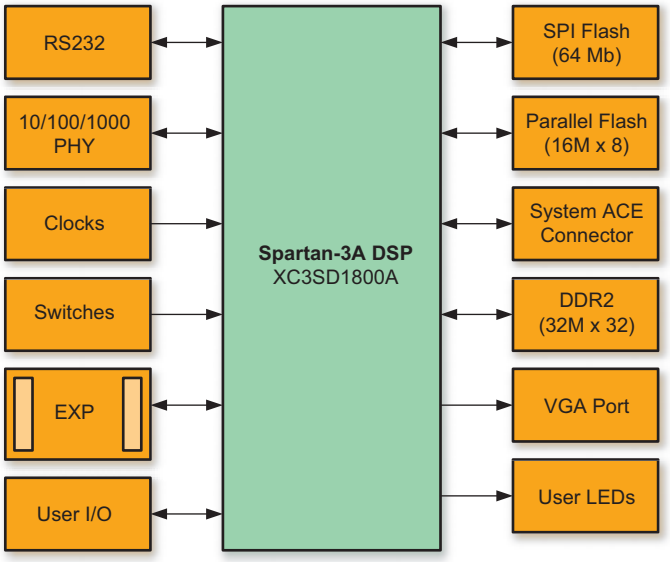


Figure 1 – Spartan-3A DSP Starter Platform block diagram

Signal Category	Pins Per Connector	Pins Per Dual EXP Slot
Single-Ended I/O	34	68
Single-Ended Clocks	2	4
Differential I/O Pairs (22)	44	88
Differential Clock Input Pair (1)	2	4
Differential Clock Output Pair (1)	2	4
2.5V pins (333 mA per pin)	12	24
3.3V pins (333 mA per pin)	12	24
Grounds	12	24
Total	120	240

Table 1 – EXP connector I/O assignments

ported. Each EXP connector can support up to 84 single-ended I/O signals or a mix of single-ended and differential pairs. Providing as many as 24 differential pairs per EXP connector, 48 pairs total, is essential for certain high-bandwidth LVDS interfaces used in video and communications applications. The Spartan-3A DSP Starter Platform board provides a user-selectable I/O voltage jumper for each EXP connector. This allows each EXP connector to be configured for either 2.5V or 3.3V signaling.

As you can see, the EXP slot included on the Spartan-3A DSP Starter Platform opens the door to a wide range of add-on application modules and custom user interfaces. Avnet has created a set of off-the-shelf EXP modules that are interchangeable between any EXP-enabled baseboard, including the Spartan-3A DSP Starter Platform. Let's explore some of these modules to see how they customize the starter platform and create powerful prototype systems that leverage Spartan-3A DSP features.

Video Applications

The Spartan-3A DSP FPGA is ideal for cost-sensitive DSP algorithmic and co-processing applications. Video and image processing, especially the video surveillance and video security markets, are good fits for this FPGA family. To help jump-start applications in this area, Avnet has created the Video EXP module. The Video EXP offers a full compliment of professional/consumer (or "prosumer")-level front-end video functions. With support for DVI, component, composite, S-video, image sensor, and VGA inputs, plus DVI, VGA, and LCD panel video outputs, the Video EXP addresses a broad range of video processing applications.

Figure 2 shows an example video over Ethernet application that can be built from the combined Video EXP and Spartan-3A DSP baseboard. The Spartan-3A DSP SelectIO interface can handle the LVDS interfaces to the image sensors and LCD flat panel. The 250 MHz DSP48A slices and larger enhanced block RAM enable the DSP-intensive image processing pipe. Functions such as the Bayer filter, color-space conversion, chroma sub-sampling, and MPEG-4 video compression can all be implemented and run in real time inside the FPGA. A 32-bit MicroBlaze™ processor helps manage the processing pipe as well as the Ethernet interface.

Wireless Communications Systems

The increasing demand for Internet access has lead to the explosive growth of wireless communication systems that provide a continuous Internet connection. WiMAX is one wireless access technology that is gaining in popularity and has significant market potential. Based on the IEEE 802.16e-2005 standard, WiMAX supports high-speed Internet

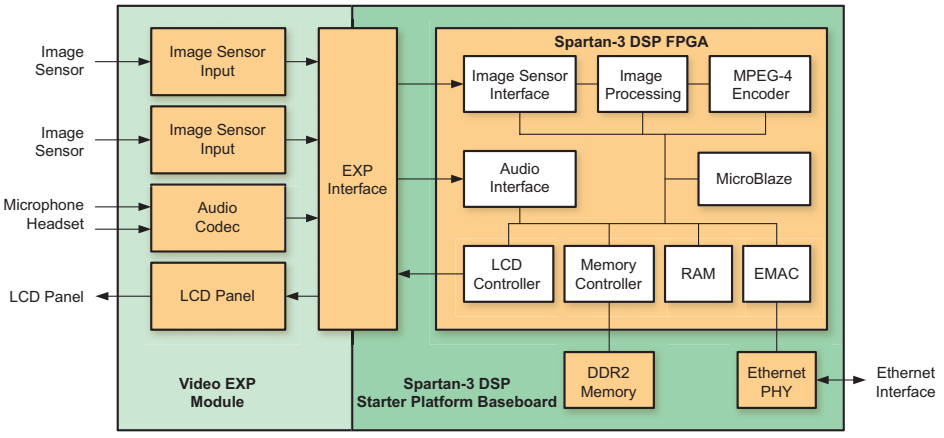


Figure 2 – Video over Ethernet application example

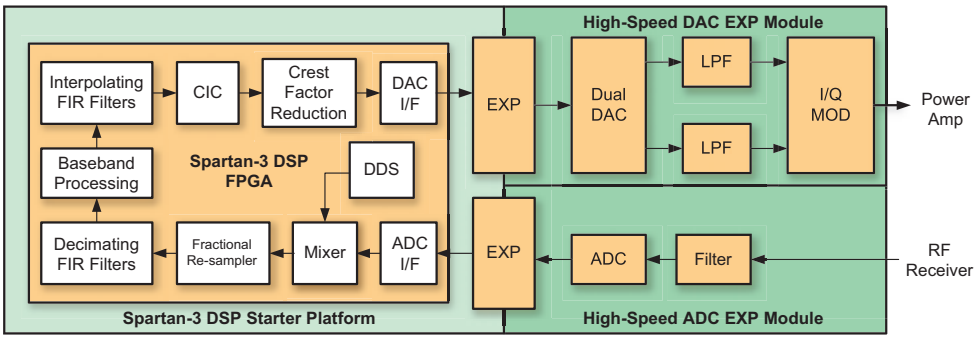


Figure 3 – Example wireless communication system

access using advanced signal processing schemes such as orthogonal frequency-division multiple access (OFDMA) and multiple input multiple output (MIMO) technology. The Spartan-3A DSP is a perfect solution for supporting these high-bandwidth, processing-intensive applications that require flexibility and fast time to market.

To support wireless communications prototyping, Avnet and Texas Instruments have developed two EXP modules that complement the digital IF processing capabilities of the Spartan-3A DSP Starter Platform. A High Speed ADC EXP module and a High Speed DAC EXP module support 12-bit, 500 MSPS analog-to-digital conversion and 16-bit, 1 GSPS digital-to-analog conversion. As half modules, each EXP requires just one EXP connector; thus any combination of two half EXP modules is supported by the starter platform baseboard.

Figure 3 is an example direct conversion wireless communications system that you can prototype with the Spartan-3A DSP Starter Platform, the High-Speed DAC EXP and the High-Speed ADC EXP. The enhanced DSP48A slices inside the Spartan-3A DSP make it ideal for implementing digital front-end processing. Digital up conversion (DUC), comprising two polyphase interpolating FIR filters, a CIC filter, and a crest factor reduction block, can be implemented inside the Spartan-3A DSP FPGA. The output drives the high-speed interpolating DACs on the EXP module, which in turn drives the RF power amplifier for transmission. On the receive side, the IF signal can be directly sampled with the 500 MSPS ADC and passed to the FPGA for digital down conversion (DDC).

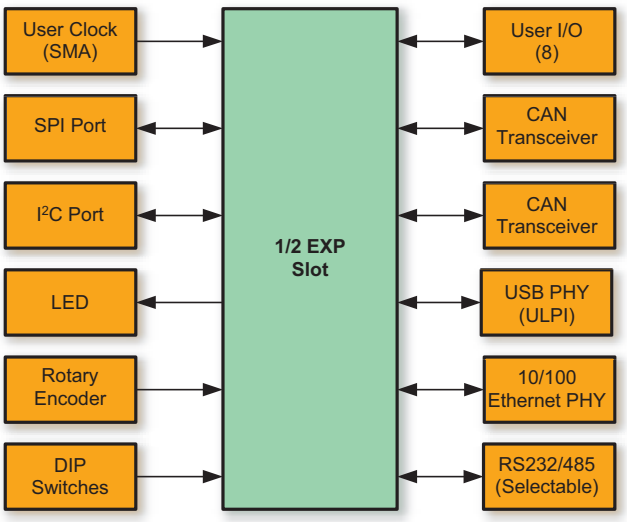


Figure 4 – Interface EXP module block diagram

Embedded Processing Platforms

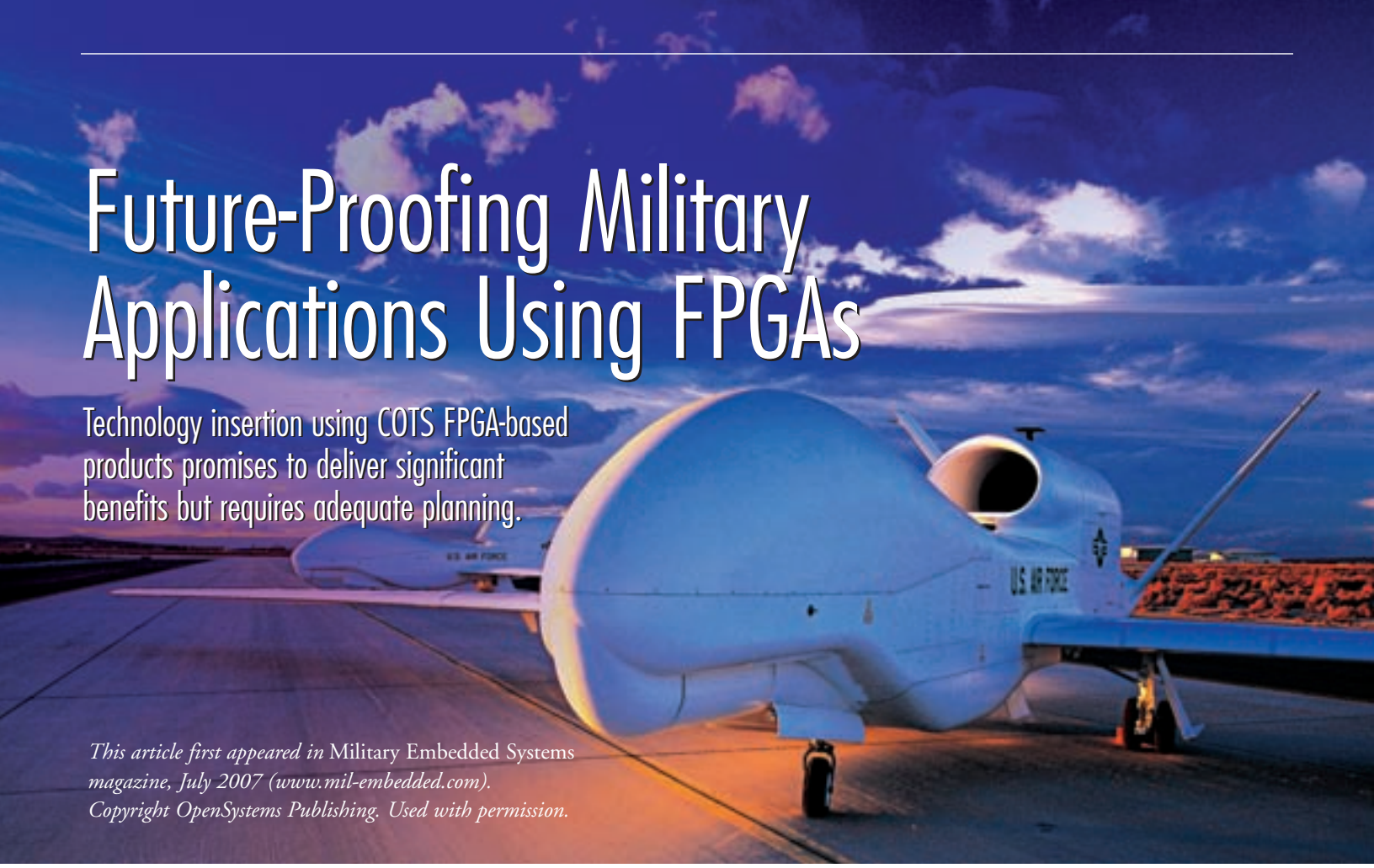
Although the Spartan-3A DSP FPGA family has features that make it well suited for high-performance DSP applications, it is also a capable choice for embedded processing solutions. The enhanced on-chip block RAM makes the Spartan-3A DSP an especially good fit for the soft-core MicroBlaze embedded processor. To better address MicroBlaze processor-based embedded applications, you can add the Interface EXP half module as defined in Figure 4. The Interface EXP supports many of the common embedded processing interfaces found in processor-based systems. Because the Interface EXP is a half-module format, you still have the option of adding a second half EXP module. When combined with soft cores inside the Spartan-3A DSP FPGA, you have ultimate flexibility and control over your design. Xilinx provides IP cores for all of the interfaces, including USB, CAN, Ethernet, SPI, I²C, and UART.

Conclusion

The Xilinx Spartan-3A DSP Starter Platform is a great way to explore applications around the Spartan-3A DSP FPGA family. The EXP slot included on the board sets this kit apart, enabling customization that addresses video processing, communications, embedded systems, and a wide range of other applications. The available EXP add-on modules allow you to quickly build up real-world prototypes and turn your concepts into reality.

Should a specific function not exist in an off-the-shelf form, you can easily create your own with a custom-designed EXP card. Avnet also provides a prototype EXP module that is great for accessing all of the I/O signals on standard .1" headers.

Avnet will continue to introduce new EXP modules that will further the reach and capability of this starter platform as well as other EXP-enabled baseboards. For a complete listing of EXP modules and the EXP specification (available for downloading), visit *www.em.avnet.com/exp*.



Future-Proofing Military Applications Using FPGAs

Technology insertion using COTS FPGA-based products promises to deliver significant benefits but requires adequate planning.

This article first appeared in Military Embedded Systems *magazine, July 2007 (www.mil-embedded.com).*

by Mark Littlefield
Product Marketing Manager
Curtiss-Wright Controls Embedded Computing
mark.littlefield@curtisswright.com

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

Over the past decade, the concept of "technology insertion" (the incremental upgrade of key technology components in a deployed or soon-to-be deployed system) has become a sort of mantra for the aerospace and defense (A&D) community, providing fielded systems with the latest, most advanced technology and the least delay. The rapid pace of technological innovation – and subsequent obsolescence – has made technology insertion critical for the success of multiyear A&D development, test, and deployment projects.

Unmanned aerial vehicles (UAVs), radar, and signals intelligence (SIGINT) are examples of sophisticated platforms that have benefited from technology insertion. To fully understand the challenges of technology insertion, including complexity and cost, it's useful to consider what is involved in successfully using this approach for upgrading legacy systems. Using FPGAs in a reconfigurable computing application provides a good example of the benefits and challenges entailed in making technology insertion work. This example also enables us to examine some common issues associated with technology insertion, how COTS vendors can best help their customers address those problems, and how the overall technology insertion problem can be significantly eased by proper planning.

Using FPGAs in Technology Insertion

The basic problem domains involved in technology insertion can be roughly categorized as:

- How the equipment physically connects (hardware)
- How the operating environment, system libraries and utilities, drivers, and middleware provide an infrastructure for applications (system software)

Using FPGA technology in an embedded multicomputing system provides an interesting case example because it straddles both the hardware and system software domains. FPGAs fall into the hardware domain because they are physically integrated to hardware elements; developers must work very closely with these elements when designing an application.

FPGAs present a software challenge as well, because they must be programmed and often incorporate vendor or third-party supplied blocks or IP. In addition, a general-purpose processor using system library calls typically configures the FPGAs and issues the commands they use to communicate with other processors and devices in the system. Such systems are often complex and heterogeneous, as shown in Figure 1.

For technology insertion, the ideal scenario is a plug replacement module that requires no hardware or software changes. Although this is uncommon for FPGA-based computing products, this scenario can also often be undesirable, as state-of-the-art computing platforms have new, more advanced features that system inte-

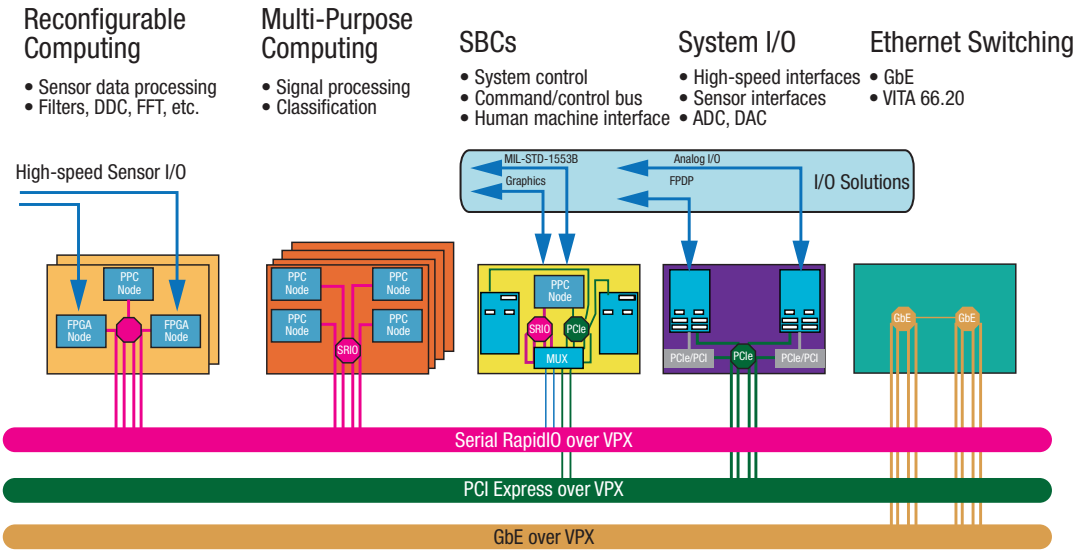


Figure 1 – High-performance embedded signal and image processing systems are often made up of numerous heterogeneous computer and I/O components connected by multiple communications fabrics.

grators can use to their advantage. Thus, the goal for system integrators must be to maximize the benefits of technology insertion while minimizing impact on the existing system. The degree of flexibility in the hardware and software domains can be directly affected by early design choices and products offered by COTS vendors.

The simplest approach for a COTS vendor to ease technology insertion hardware problems is to follow a common hardware model from platform generation to platform generation and to minimize connector pin changes. Industry board and module standards such as VME, VPX-REDI (VITA 46/48), PMC (VITA 32), and XMC (VITA 42) address a large part of this problem by specifying fixed form factors and pin definitions for board I/O such as buses or switched fabric interconnects. Vendors can extend this model by maintaining pin footprints across product generations for such common interfaces as serial ports and Ethernet. The same also holds true for FPGAs. A common, or at least similar, I/O footprint minimizes the need for radical redesigns during a technology insertion project.

The Importance of Software

Although FPGA development is very tightly linked to the target component and platform hardware design, there is a lack of standardized or industry-accepted tools and frameworks to abstract this linkage. The result is that software can typically have an even greater effect on a technology insertion program than hardware.

The software challenges can be somewhat mitigated on general-purpose processors by using off-the-shelf operating systems such as VxWorks or Linux, supported with full-featured BSPs, communications middleware, and application frameworks. For COTS vendors of FPGA products, the challenge is to provide development tools to integrators to ease technology insertion while maintaining the performance demanded by application developers.

A common approach is for COTS vendors to develop a standard "wrapper" or gasket that essentially represents the static infrastructure, such as interfaces to ADCs, memories, Ethernet, and so on. The blocks and infrastructure should be designed to support a set of commonly implemented use cases in the most efficient manner possible so as to minimize the size of the wrapper. Ideally, such an infrastructure would represent only 5 to 10 percent of the total die size of the FPGA.

Software Support

No less important than the ease of integrating existing application code into a new FPGA platform for a technology insertion project is the integration of the FPGA-based application into a larger multicomputer system. General-purpose computing elements are often closely tied to the system's FPGAs by performing various command and control functions such as DMA engine control. Subsequently, the ease of technology insertion can be directly affected by how well a vendor can maintain APIs from one product generation to the next, which in turn is often linked to the level of abstraction in the API. The more abstraction, the less likely it is that the API will change between product generations. One of the key tasks of an external processor is the command and control of data movement, both within the FPGA and between the FPGA and other general-purpose processing nodes.

Making Technology Insertion Real

Several COTS vendors offer common hardware strategies to aid technology insertion. One example is Curtiss-Wright's latest generation of VPX-based board products. The three primary computing platforms in this product line – a dual 8641-based SBC, a quad 8641-based DSP engine, and the dual Xilinx® Virtex™-5 FPGA-based CHAMP-FX2 board – all share a set of common hardware design elements and were specifically designed with common pin footprints for common I/O elements. (See Figures 2 and 3.) For instance, the

FPGA board not only shares the PowerPC node design elements with the two other VPX boards, but closely resembles its preceding VME version in its I/O and memory configuration.

Figure 2 – High-performance FPGA families, such as the Xilinx Virtex-5 family, have common elements for logic, DSP, memory, and I/O, with a common package strategy enabling technology insertion at the component level.

The design of the VPX FPGA board and its VME predecessor took a balanced approach between the twin goals of maximum technology insertion value and minimum insertion effort. Curtiss-Wright's Continuum FXtools design kit provides a highly optimized set of IP blocks focused on peripheral I/O and memories, with only a lightweight, scalable switching block and a few additional utility blocks to ease application integration. By providing a minimal but highly optimized infrastructure, FXtools abstracts those common I/O and memory objects without sacrificing performance. Designing with these IP blocks and adopting basic use cases can help ease future technology insertion.

To ease both the integration and technology insertion of FPGA-based computing elements, the board's Continuum IPC communications middleware was extended to directly control DMA-based data-transfers involving the FPGA node. The IPC software enables application developers to create named buffer and data transfer objects that are globally visible to the system. Thus, any processor in the IPC system can create named buffers, attach to already created named buffers, and create data transfer objects between buffers without having to resort to code-manipulating, complex memory map translations or DMA command packet creation.

Avoiding Technology Insertion Headaches

Although hardware and system software commonality across product generations is an important element in planning a successful technology insertion project, the application developer and system integrator can themselves play an important role in defining how difficult a future technology insertion project may be. For example, if the application developer bypasses vendor or OS APIs to directly manipulate hardware, the resulting code will be significantly less portable to future hardware platforms. However, application developers or system integrators can structure their application/system in a number of ways to minimize technology insertion headaches.

The first rule of designing for later technology insertion should be, "Don't fight the system software or OS." Most software frameworks, such as a BSP, utility library, or communications middleware like Continuum IPC are designed with one or more basic use cases or design patterns in mind. One of the easiest ways for developers to introduce problems in a later technology insertion project is to bend the software framework to match their favorite design pattern. Doing so virtually ensures that later developers will have to bend things in a different way simply to make it work.

The same holds true for FPGA IP developers. Although implementation details and interfaces may shift from one product generation to the next, the basic design patterns usually do not. Using vendor-supplied IP blocks and data flows in the manner that the vendor originally intended helps minimize the amount of recoding needed for a technology insertion project.

Another beneficial technique that can be employed both for software and FPGA IP is to analyze the vendor-supplied APIs and block interfaces, identifying those interfaces that are likely to shift in future products and creating an abstraction layer or "shim" between the interfaces and the application code. A helpful clue is that if an API or block interface closely mirrors the underlying hardware, it is likely to shift between product generations.

Although such layers can add unnecessary code and performance delays to a system, the performance impact is often minimal, while the benefits for a later technology insertion project can be enormous. Changing a shim is much easier than changing multiple instances of an API call or IP block instantiation. It's important not to overuse this approach, however, as adding shims or abstraction layers complicates designs and can sometimes make debugging more difficult.

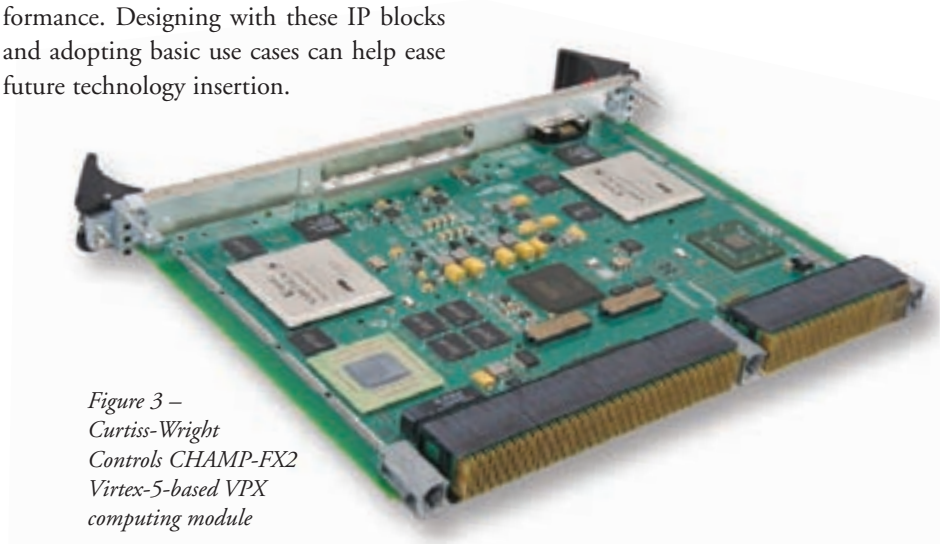


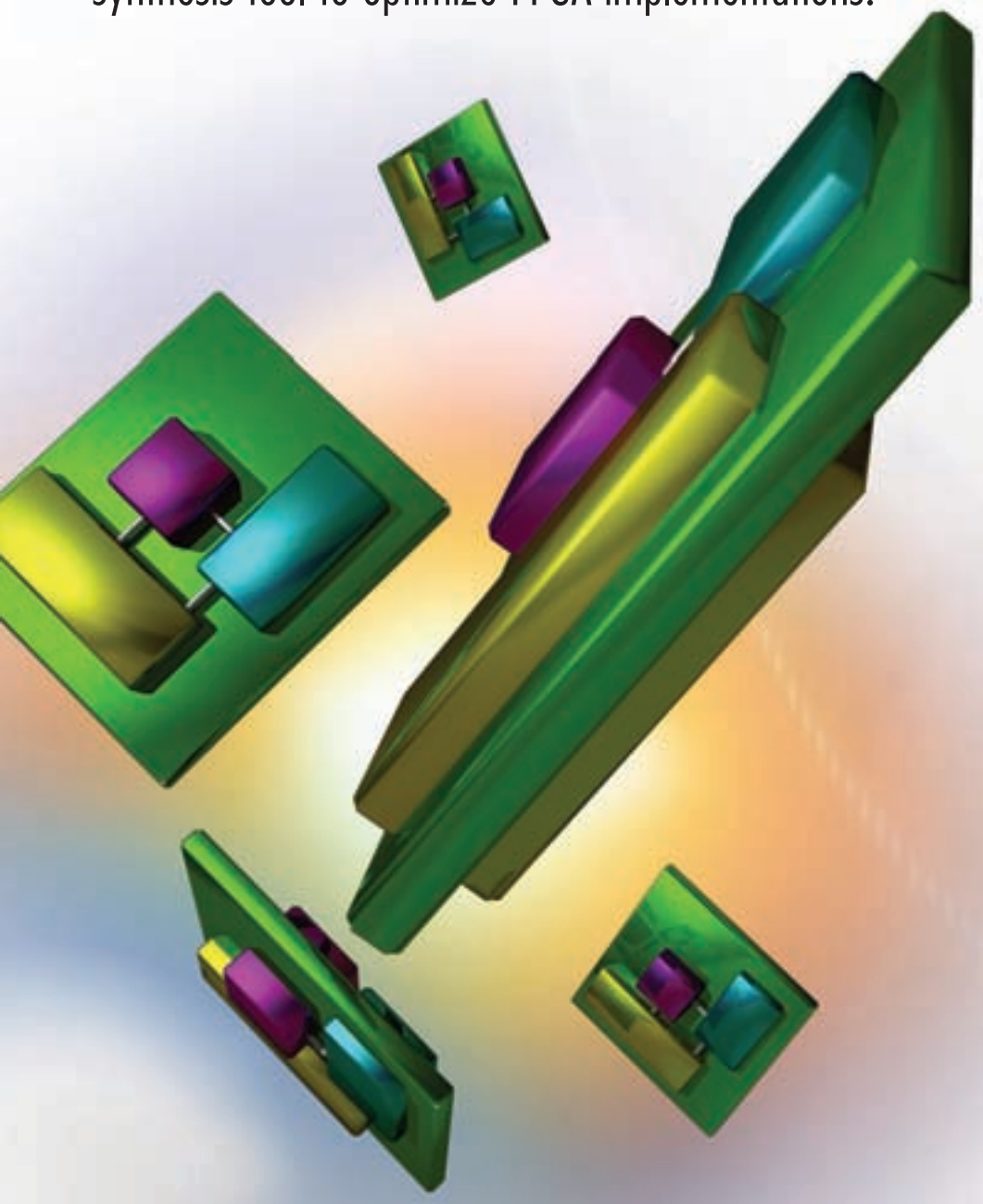
Figure 3 – Curtiss-Wright Controls CHAMP-FX2 Virtex-5-based VPX computing module

Conclusion

Technology insertion can live up to its promise of future-proofing, but only if it is adequately planned for up front. Most COTS vendors have developed successive generations of products with this in mind. To capture this value, system integrators need to work closely with such vendors early in the design cycle.

Using MATLAB to Create IP for System Generator for DSP

You can use MATLAB M-files with the AccelChip synthesis tool to optimize FPGA implementations.



by Tim Vanevenhoven
DSP Tools Product Manager
Xilinx, Inc.
tim.vanevenhoven@xilinx.com

DSP systems are best described by using a combination of both graphical- and language-based methods. The MathWorks, an industry leader in DSP modeling software, caters to this dichotomy by providing a cycle-accurate graphical design environment called Simulink and a mathematical modeling language called MATLAB.

Simulink is well suited for the "system" aspects of DSP design, including control and synchronization of data flow to and from interfaces and memories. Simulink also provides a rich set of pre-defined DSP algorithms in the form of block sets that you can use to construct DSP systems. Simulink is not always the most effective environment for modeling proprietary algorithms, however. It unnecessarily burdens designers with cycle-accurate considerations and forces low-level arithmetic operations and array accesses to be constructed from graphical block sets rather than concise textual expressions.

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

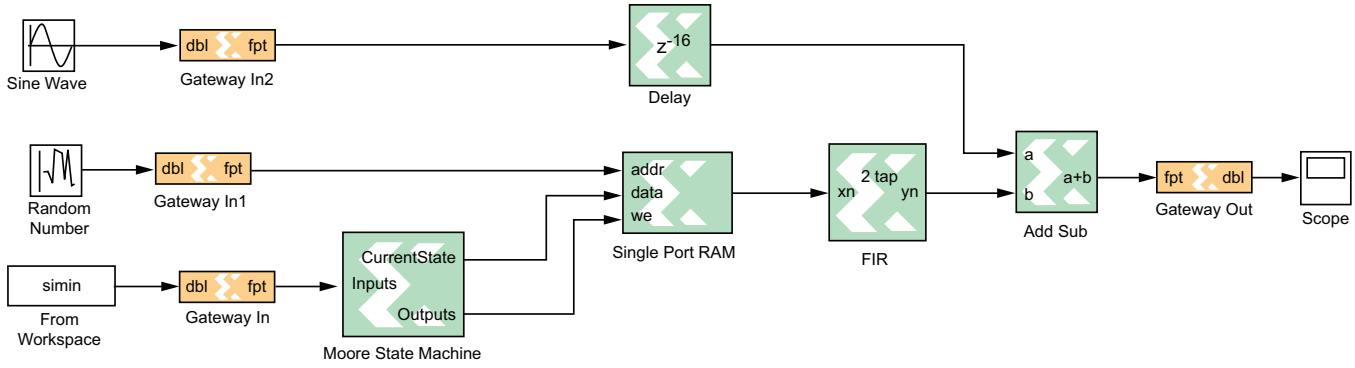


Figure 1 – Xilinx System Generator diagram showing system control and synchronization logic

Simulink unifies both modeling environments with an embedded MATLAB block, which allows MATLAB models to simulate within Simulink and compile into C code through Real-Time Workshop for processor-based DSP hardware implementations.

Xilinx® System Generator for DSP is well established as a productive tool for creating DSP designs in FPGAs. With a graphical environment based on Simulink and a pre-defined block set of Xilinx DSP cores, System Generator for DSP meets the needs of both system architects who need to integrate the components of a complete design and hardware designers who need to optimize implementations. System Generator for DSP, however, lacks support for a design flow based on MATLAB.

The Xilinx AccelDSP™ synthesis tool was developed specifically for algorithm developers and DSP architects who have embraced language-based DSP algorithm modeling. With the AccelDSP synthesis tool, algorithm developers can use their floating-point MATLAB M-files to perform stimulus creation, algorithm evaluation, and results post-processing.

DSP Hardware Systems

System Generator for DSP is well suited for modeling the DSP system, which comprises not only the core DSP algorithm but synchronous interfaces to external buses, memory read/write accesses, system data synchronization, and overall system control. System Generator for DSP provides control-oriented blocks such as the MicroBlaze™ processor as well as individual register, delay, and memory blocks for implementing the synchronous components of a DSP system (Figure 1).

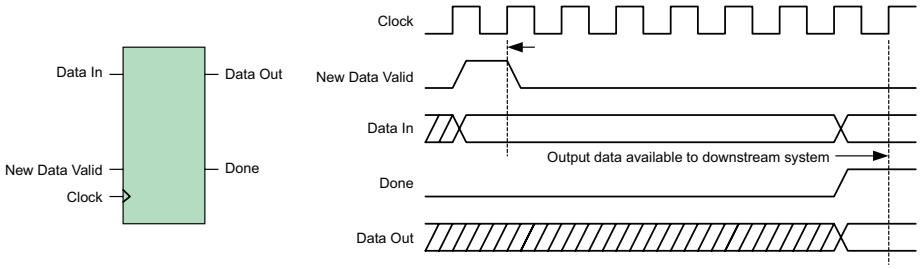


Figure 2 – AccelDSP handshaking interface

Custom DSP Algorithms

The heart of any DSP system is the algorithm. Algorithms distinguish themselves from systems in that a resulting output is a function of a given set of inputs without a sense of clock or hardware. This is described by the simple equation:

$$y = f(x)$$

You can execute an algorithm defined in MATLAB on an FPGA, DSP processor, or software; each will have a different sense of cycle accuracy.

The unique characteristics of an algorithm offer two distinct advantages. First, algorithm developers are completely unencumbered by hardware implementation details and can focus solely on functional behavior. This is exactly why an estimated 90% of the algorithms used today in DSP originate as MATLAB models, even when a design flow dictates that they be re-implemented at a later point in time as Simulink or System Generator for DSP diagrams. Calculating the fast Fourier transform (FFT) of a 4 x 1,024 matrix of data is possible with a simple MATLAB statement, without concern for radix, scaling, buffering, or synchronization of valid signals, as shown here:

```
y = fft(data,1024);
```

Second, when modeling an algorithm, a given set of outputs corresponds to a given set of inputs; therefore, synchronization issues do not need to be addressed in the generated hardware. This makes an algorithm inherently "schedule-able" through a tool like the AccelDSP synthesis tool. Hardware requirements may dictate the need to use multiple clock cycles to compute an output, as in the case of a resource-shared MAC FIR filter, but this operation lends itself nicely to the AccelDSP synthesis tool's automated flow. The introduction of a simple hardware handshaking interface enables easy integration into the overall system, as shown in Figure 2.

MATLAB operators such as matrix transposition allow the MATLAB code to be compact and easily readable. And complex operations like matrix inversion are done using MATLAB's extensive linear algebra capabilities.

Using the AccelDSP and System Generator Tools Together

The AcceDSP synthesis tool enables System Generator for DSP to support both DSP system and algorithm modeling methods by generating System Generator IP blocks based on floating-point MATLAB models. This results in a design flow for FPGAs that is similar to the functionality obtained through the use of the embedded MATLAB block (Figure 3). You can capture the system aspects of the design using the Xilinx DSP block set and capture the algorithm using floating-point MATLAB. The System Generator IP blocks created by the AccelDSP synthesis tool are fixed-point and cycle-accurate.

Kalman Filter Example

Let's take an advanced algorithm written in MATLAB, use the AccelDSP synthesis tool to synthesize the design, and then integrate the algorithm into a System Generator model. A Kalman filter is a special class of recursive, adaptive filters that is well suited to combining multiple noisy signals into a clearer signal. Kalman filters start with 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. The filter then uses measured signals (such as the signature of the aircraft returned to the radar receiver) to periodically correct the prediction.

The following is a MATLAB M-file for a 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 forward and the code that corrects itself.

```
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_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';
```

Common operators like addition and subtraction operate on arrays like A or P_cap without the need to write loops, which would be required in languages like C. Two-dimensional arrays are automatically multiplied as matrices 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 performed using MATLAB's extensive linear algebra capabilities. Although such an algorithm can be constructed as a block diagram, doing so will obscure the algorithm structure so readily apparent in MATLAB.

With the AccelDSP synthesis tool, you can take complex MATLAB toolbox and built-in functions (such as the matrix inverse used in the Kalman filter example) directly to hardware using the AccelWare IP tool kits. These tool kits offer multiple matrix inverse methods. Core selection will depend on the size, structure, and values of the matrix. Returning to the Kalman filter example, the most suitable approach is to use the AccelWare QR matrix inverse core. AccelWare cores are generated based on MATLAB syntax and are available in a variety of implementations that let users optimize designs for speed, area, power, or noise. This small code edit would be required to use the AccelWare function:

```
% K = P_cap_est * inv(P_cap_est+R);

K = P_cap_est *
accel_qr_inverse (P_cap_est+R);
```

Synthesizing MATLAB with the AccelDSP Synthesis Tool

With the AccelDSP synthesis tool, you can perform a floating-point simulation to establish a baseline. The design is then converted into a fixed-point representation so that the fixed-point effects can be simulated. An array of features helps analyze these effects and fixed-point design options like saturation and rounding modes. Bit growth is automatically propagated through the design in a manner that allows for user-controlled overrides. This algorithmic design exploration process helps you obtain the ideal quantization that minimizes bit widths while managing overflows/underflows, allowing early trade-offs of silicon area versus performance metrics.

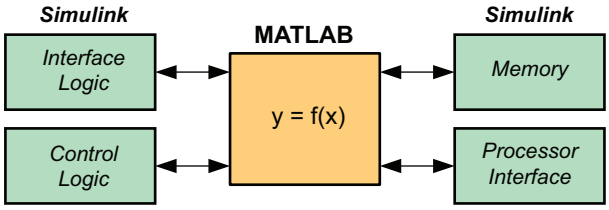


Figure 3 – DSP system block diagram

DSP Synthesis Directive	Effect on Generated Hardware
Rolling/Unrolling of For Loops	Improves input sampling rate by reducing throughput
Expansion of Vector and Matrix Additions and Multiplications	Improves input sampling rate by reducing throughput
RAM/ROM Memory Mapping of Arrays	Improves FPGA utilization by mapping arrays into dedicated Xilinx block RAM resources
Pipeline Insertion	Improves input sampling rate by improving clock frequency performance
Shift Register Mapping	Improves FPGA utilization by mapping shift register logic into SRL16s

Table 1 – Synthesis directives to control the generated hardware

Once suitable quantizations have been determined, the next step with the AccelDSP synthesis tool is to generate RTL for the target Xilinx device. You can dictate hardware intent through the use of the synthesis directives listed in Table 1.

Using these directives constitutes hardware-based design exploration that allows the design team to further improve quality of results. In synthesizing the RTL, the AccelDSP synthesis tool evaluates and schedules the entire algorithm, performing boundary optimization when possible.

Throughout this flow, the AccelDSP tool maintains a uniform verification environment through the use of 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 AccelDSP synthesis tool will also report the throughput and latency of the Kalman filter, which is essential to gauge whether the design meets specifications and to produce a cycle-accurate System Generator model.

Figure 4 – "Generate System Generator" command in the AccelDSP synthesis tool

Generating a System Generator Model

Upon successful completion of RTL verification, the design is now ready to generate a System Generator model by clicking on the "Generate System Generator" icon, as shown in Figure 4. The AccelDSP tool generates a System Generator IP block that supports both simulation and RTL code generation.

At this point, the design flow transitions to System Generator for DSP, where the new block for the Kalman filter is available in the Simulink library browser. You only need to select the Kalman filter block and drag it into the destination model to incorporate the AccelDSP-generated Kalman filter into a System Generator design, illustrated in Figure 5.

Conclusion

Custom DSP algorithms are best modeled mathematically using MATLAB, while complete systems are best modeled as cycle-accurate using Simulink. The marriage of these two modeling domains provides an efficient means to design DSP systems into FPGAs by allowing the use of the rich MATLAB language, including the built-in and toolbox functions, to create System Generator IP blocks of complex DSP algorithms.

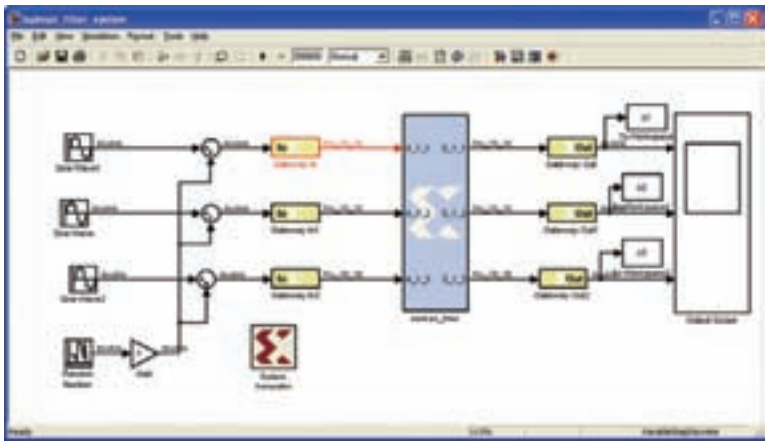


Figure 5 – Kalman filter added to a System Generator design

TAKE THE NEXT STEP (Digital Edition: www.xcellpublications.com/subscribe/)

- See an online demo of the System Generator and AccelDSP tools.
- Evaluate or purchase:
  - The AccelDSP synthesis tool.
  - System Generator for DSP.

Automatic IP Block Selection with IP-Explorer Technology

You can use the AccelDSP synthesis tool to select from various macro-architectures based on your system requirements.

by Tom Hill
System Generator Product Manager
Xilinx, Inc.
tom.hill@xilinx.com

Developing DSP algorithms for silicon requires careful selection of IP blocks and their macro-architecture specifications in the context of the target application. For example, three methods are commonly used to implement basic trigonometric functions in hardware:

1. Bipartite tables – a good choice when input values require relatively small word lengths.
2. Linearly interpolated look-up tables (LUTs) – best for slightly larger input word lengths.
3. CORDIC – ideal for applications that require large word lengths when performance is not an issue. If performance is important, you can implement the CORDIC algorithm using parallelism and pipeline stages.

The graph in Figure 1 compares the area of these methods over a range of input word lengths.

Knowing how to build and when to use each of these cores is a complex task, but necessary to achieve an optimal hardware solution.

Introducing IP-Explorer

The Xilinx® AccelDSP™ synthesis tool with IP-Explorer technology eliminates the trial-and-error process when using IP blocks by allowing the tool to select from various macro-architectures. The AccelWare™ DSP IP tool kit provides a set of IP generators that produce architecture-specific, synthesizable MATLAB for common built-in functions such as sine, cosine, log, and divide. These hardware architectures, developed by a team of DSP hardware design experts, are characterized by area and performance for each supported FPGA technology using more than 6,000 designs.

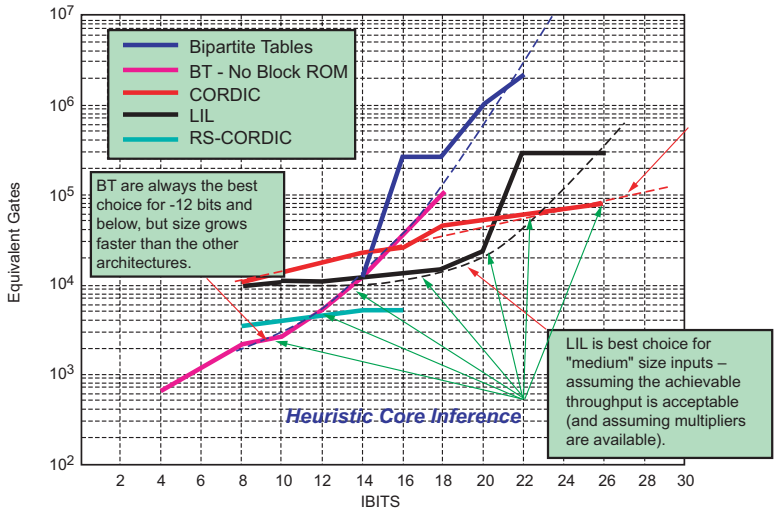


Figure 1 – Area versus input word length

During the DSP synthesis process, IP-Explorer analyzes each use of these functions against the overall system area and performance requirements to determine the most optimal hardware solution. Once determined, the tool automatically inserts the appropriate IP core into the design (Figure 2).

Designing at a Higher Level

IP-Explorer raises the abstraction level of DSP hardware design significantly closer to that of DSP algorithm development. Basic MATLAB language primitives form the foundation of this abstraction pyramid (Figure 3) by providing the ability to model any function or algorithm at a low level. IP-Explorer elevates this abstraction level significantly by targeting MATLAB's standard library of pre-configured DSP building blocks and functions directly into hardware.

Design Example

To illustrate the effect that IP-Explorer has on a design, consider the example shown in Figure 4: a Euler-to-quaternion angle conversion algorithm that performs a sine and cosine operation on three input angles. In this example, each input has a word length of 12 bits.

Let's compare a typical approach – designing a single implementation of the trigonometric function based on the CORDIC method and instantiated multiple times – versus IP-Explorer. Table 1 shows the area and performance results for both approaches.

IP-Explorer selects a bipartite table approach because the input bit widths are reasonably small. A secondary benefit is that this approach leverages the built-in DSP blocks of the targeted FPGA device. This would not have been the optimal choice for an ASIC. Not only has the task of designing hardware for the sine and cosine functions been eliminated, but this approach offers a 33% area savings and a 12x performance advantage.

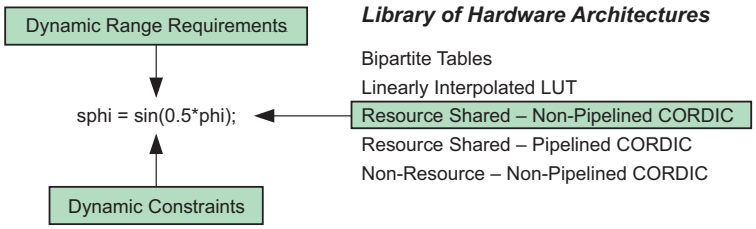


Figure 2 – Design context-based automatic IP insertion

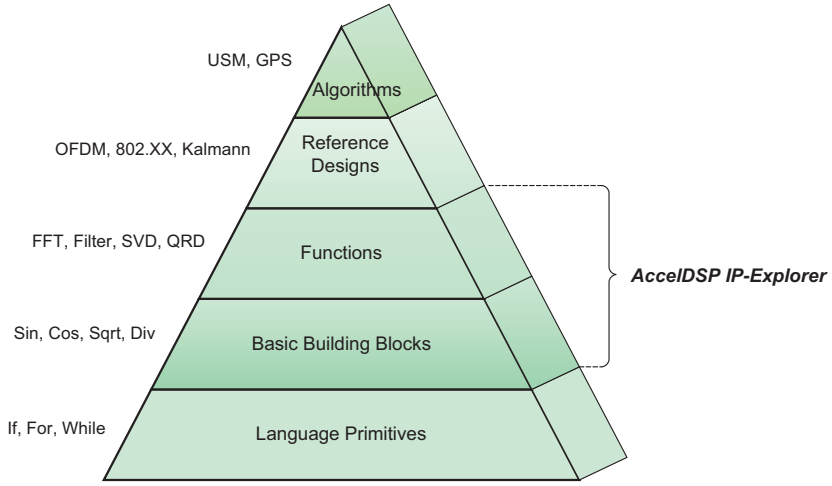


Figure 3 – DSP design abstraction pyramid

```
function [q] = euler2quat(phi,theta,psi)

sphi = sin(0.5*phi);
cphi = cos(0.5*phi);
stheta = sin(0.5*theta);
chteta = cos(0.5*theta);
spsi = sin(0.5*psi);
cpsi = cos(0.5*psi);

q(1) = sphi*stheta*spsi+cphi+ctheta+cpsi;
q(2) = sphi*ctheta*cpsi-cphi*stheta*spsi;
q(3) = cphi*stheta*cpsi+sphi*ctheta*spsi;
q(4) = cphi*ctheta*spsi-sphi*stheta*cpsi;
```

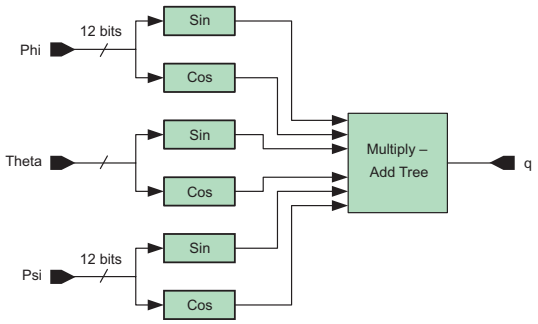


Figure 4 – Euler-to-quaternion angle converter block diagram

Sine/Cosine Architecture	Area (LUTs)	Performance
Six CORDIC Architectures	2,670	8.4 MSPS
Selected by IP-Explorer	1,287	100 MSPS

Table 1 – Area and performance results for CORDIC and IP-Explorer-selected approaches

Conclusion

IP-Explorer represents a truly unique advancement in the development of DSP hardware. By providing a seamless path to hardware for key DSP building blocks, hardware design has moved significantly closer to the abstraction level enjoyed by algorithm developers using MATLAB.

TAKE THE NEXT STEP
(Digital Edition: www.xcellpublications.com/subscribe/)

- Learn more about the AccelDSP synthesis tool.
- Listen to the free recorded lecture, "AccelDSP Jump Start Modules."

inrevium
by Tokyo Electron Device Ltd.

A new lineup! Releasing
the SX50T DSP Platform version

DSP

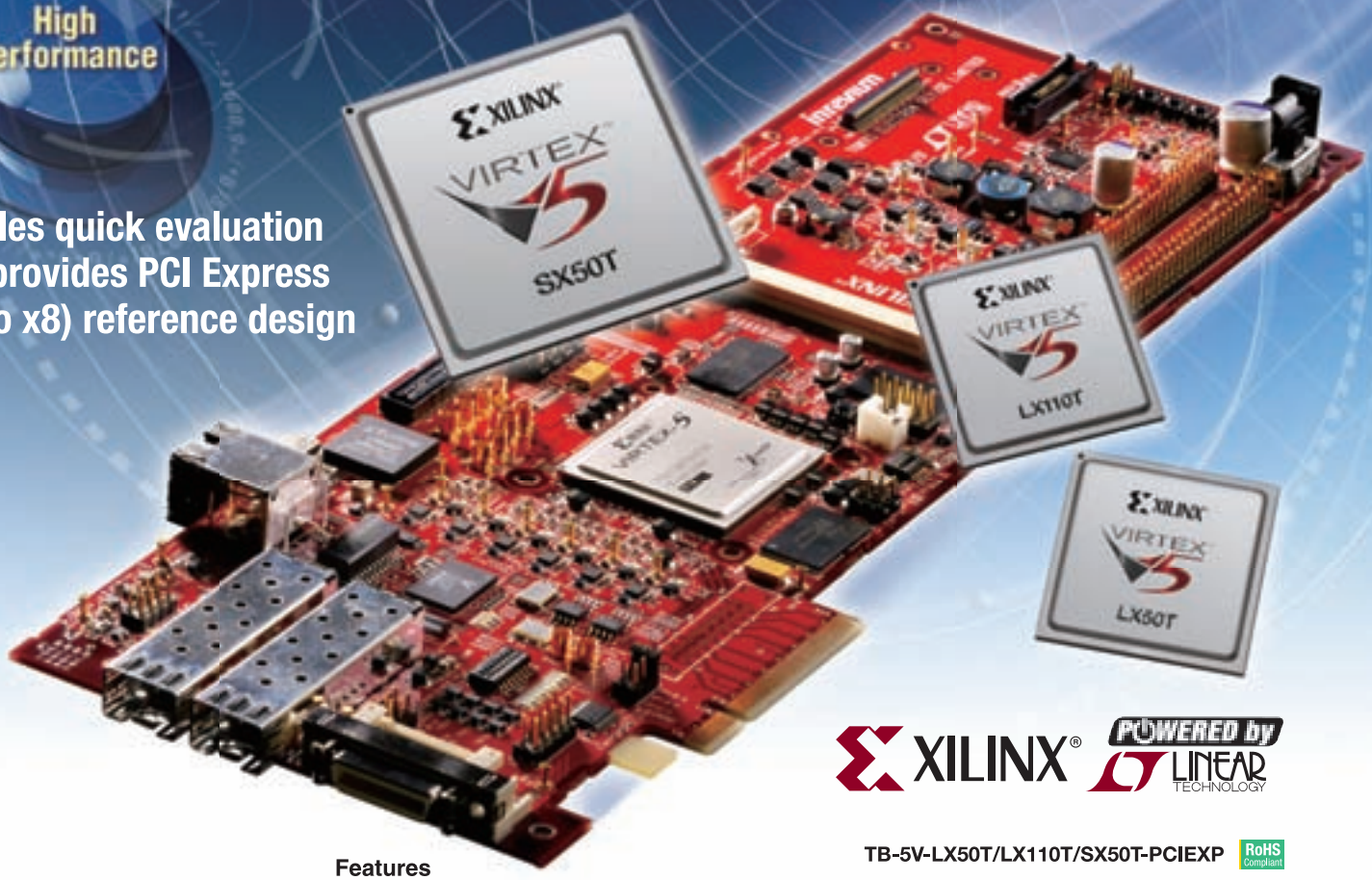
Supports various types of
high-speed interfaces

PCI Express Verified Platform

Flexible

High Performance

Enables quick evaluation
and provides PCI Express
(up to x8) reference design



XILINX® POWERED by LINEAR TECHNOLOGY

TB-5V-LX50T/LX110T/SX50T-PCIEXP RoHS Compliant

Features

Xilinx Device
XC5VLX50T/LX110T/SX50T-1FF1136

Interfaces
PCI Express 8-Lane
Camera Link Rx Connector x1 (Base Configuration)
SFP x2ch

Power supply
LTM4600/LTC3773/LTC3026

Memory
DDR2 SDRAM (512Mb/data bus16bit): 1chip
DDR2 SO-DIMM: 1pc

For processor (MicroBlaze™)
10/100/1000Mbps Ethernet
MICTOR Connector x1 (For Debugger)
RS232C Port x1

Others
MMCX Connector x2 (GTP x2ch)
Samtec Connector x1 (For LX110T, GTP x4ch)
50 I/O Pin Header x2ch (For LX110T)
Connector for Option Board x1 (For LX110T x3)
Push switch x4/ DIP switch x8/ LED x8

TOKYO ELECTRON DEVICE LIMITED

Headquarters: 1, Higashikata-cho, Tsuzuki-ku, Yokohama-City,
Kanagawa 224-0045, Japan
Phone: +81-45-474-7028
E-mail: psd-sales@teldevice.co.jp

US office: 2953 Bunker Hill Lane, Suite 300Santa Clara, CA 95054, USA
Phone: +1-408-919-4772

URL http://www.inrevium.jp/eng/

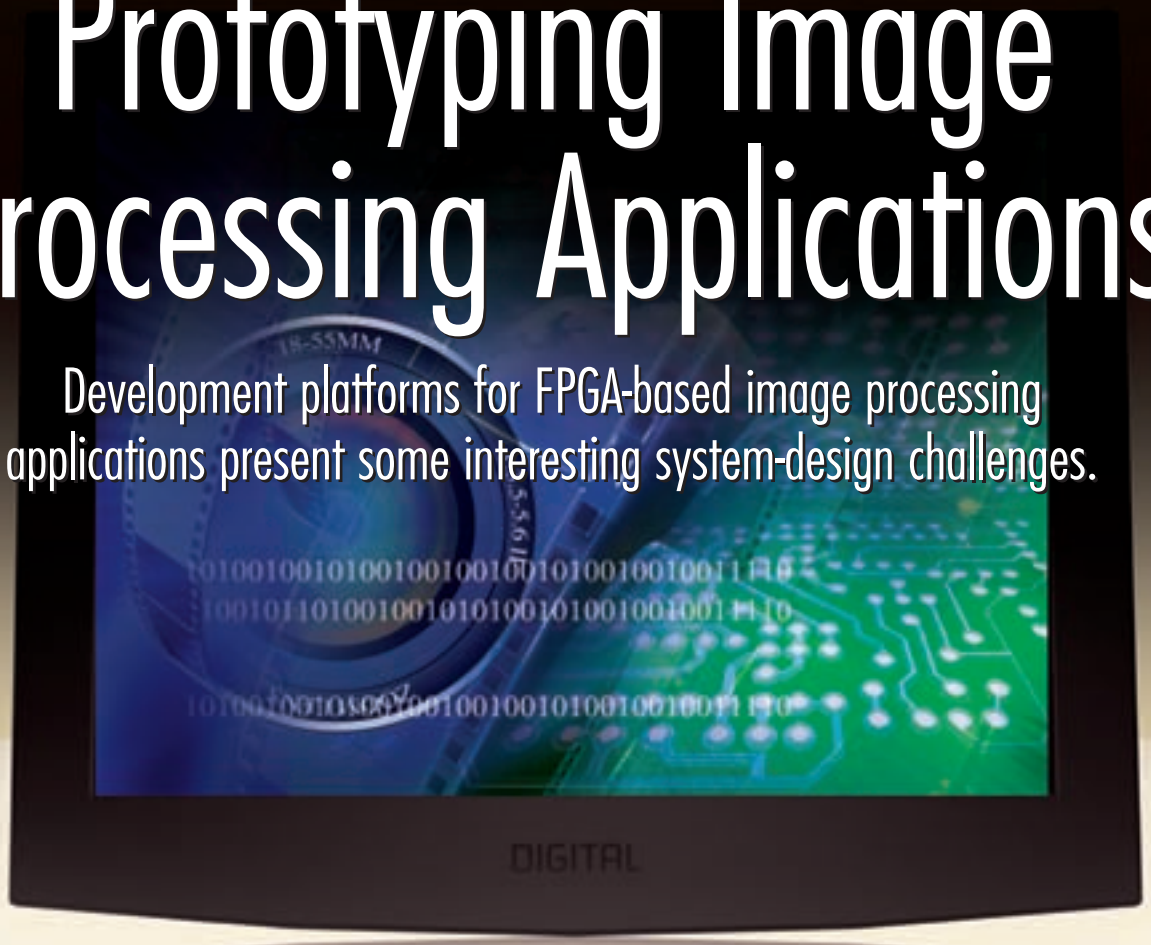
Worldwide Distributors:

Nu Horizons Electronics Corp.
Phone: +1-888-747-NUHO (6846)
URL http://www.nuhorizons.com

HiTech Global Design & Distribution, LLC
Phone: +1-408-781-8043
URL http://www.hitechglobal.com

Prototyping Image Processing Applications

Development platforms for FPGA-based image processing applications present some interesting system-design challenges.



by Mike Hodson
Managing Director
Image Processing Techniques Ltd.
mike.hodson@imageproc.com

FPGA devices have been used for many years in the design of professional video systems. The consumer electronics marketplace, however, has traditionally been an area where FPGAs have not made much of an impact, primarily because of device cost.

Recently, goods such as high-resolution flat-panel displays, MPEG-2/MPEG-4 encoders and decoders, portable video players, and digital cameras have grown in popularity. Such items require significant image processing functionality. A few years ago, custom silicon would have been the only sensible choice, but with the introduction of next-generation devices such as the Xilinx® Spartan™-3A DSP and Virtex™-5 families, the price/performance balance of FPGAs versus ASIC or ASSP devices has shifted, leading to significant growth in the use of FPGAs for such applications.

Using FPGAs for image processing applications presents system designers with some interesting problems. The analysis and manipulation of video images is inherently a high-bandwidth process, while software simulations do not always provide engineers with enough information to establish the performance of their algorithms because they are not in real time. For this reason, there is a growing demand for FPGA hardware prototyping systems targeted specifically at the requirements of image processing engineers – development platforms that provide the real-time I/O and memory interface capabilities necessary to prototype today's most demanding signal processing applications, as illustrated in Figure 1.

In this article, I'll look at the requirements and availability of FPGA development platforms for image processing applications.

Required Elements

When considering the design of an FPGA platform for image processing algorithm development, it is useful to review the principal elements that are common to the most popular image analysis techniques, and what hardware resources are needed to support these elements.

High-Speed I/O

Getting real-time video signals into and out of an image processor is obviously critical to the performance of the system. Typically, image data is input and output synchronously through a proprietary or industry-standard interface such as DVI, HDMI, SDI, analog component, S-video, or composite. The images may alternatively be transferred asynchronously through a processor or backplane bus (PCI, VME) directly into frame stores.

The latest generation of serial bus interfaces, such as PCI Express, have sufficient

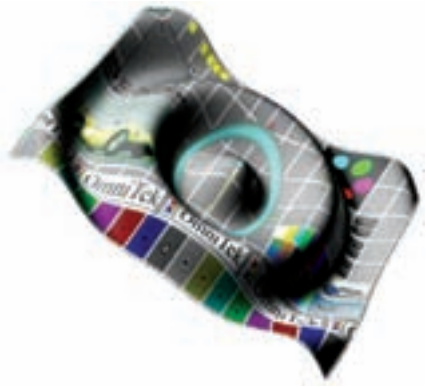


Figure 1 – Real-time 3D image manipulation using FPGA technology

bandwidth for multiple streams of uncompressed high-definition video to be transferred in real-time, while Virtex-5 devices, with their on-chip PCI Express interface and multiple MGTs, provide an excellent range of I/O capabilities for video interfacing.

Frame Stores

The frame stores used for temporary storage of image data are at the core of many image processing functions. They serve many purposes, such as synchronizing multiple image sources, image re-sizing and manipulation, motion or object detection and tracking, noise reduction, and de-interlacing.

The storage of multiple video frames typically requires a large quantity of memory that is accessed sequentially. For this reason, single- or double-data-rate SDRAM is normally employed. However, in applications where image transformations include perspective, rotation, or non-linear "warps" of the source image, the frame store architecture is typically based on very high speed synchronous static RAMs to give low-latency random access operation (rather than linear data bursts from SDRAM).

Filtering and Interpolation

Applications that involve re-sampling of the source image data also need to interpolate or filter the image to avoid aliasing problems. Efficient implementation of filters is therefore important. Spartan-3A DSP and Virtex-5 devices are useful here, as the architecture of their DSP blocks is ideally suited to the efficient implementation of all kinds of filters.

Delay Elements

Image processing algorithms typically require a wide range of delay elements. For delays of a few pixels – to equalize pipeline processing latency or for polyphase filter taps – the SRL16 blocks in Xilinx FPGAs are highly efficient. Block RAMs, with their dual-port architecture, are ideal for line delays, which are needed where the vertical columns of data within 2D raster-scan images are processed.

Look-Up Tables

Adjustments to the dynamic range of image data and non-linear transfer functions can be efficiently implemented using look-up tables. How such tables are implemented – and how they are referred to – depends on the number of inputs they have. Look-up tables (LUTs) are said to be "1D" where the table output value depends on only one input value, "2D" where the output depends on two inputs, and "3D" where it depends on three inputs.

The 1D and 2D cases can usually be implemented in block RAM (depending on the size of the input vectors). However, to obtain sufficient precision in 3D LUTs, it is often necessary to use external memories such as high-speed synchronous static RAMs. DRAMs are not usually suitable because LUT operations are rarely burst-oriented. Instead, look-up address values tend to be random in nature, which makes DRAM usage inefficient.

Color-Space Conversion

A very common image processing operation is the conversion of image data between different color spaces. RGB, YCbCr, XYZ, and xvYCC are all in widespread use at the moment, so there is frequently a need to convert between these different formats. Conversion typically uses a 3 x 3 matrix multiplier array. Such arrays can be implemented very easily using Xilinx DSP blocks.

Video Mixing and Keying

Many image processing applications require multiple image sources to be mixed or cross-faded, or one image to be overlaid on top of another using a "key" or alpha channel as a guide. It is quite possible to perform these operations using a simple switch between sources; however, this does not allow for variable transparency and may also result in out-of-band components in the output image. For this reason, professional mixers and keyers use proper cross-fader logic built from multiplier pairs and key signals with "soft" low-pass filtered edges. Ideally, FPGA-based systems should implement similar logic.

Software Control and Analysis

Virtually all image processing systems require some form of software-programmed control system, either to configure the system correctly or to analyze the data and present the results in an efficient manner. These control systems may range from a simple soft- or hard-IP core processor embedded in the FPGA (such as the PowerPC embedded in Virtex FPGAs) all the way up to the latest Intel Core 2 Quad systems running Microsoft Windows Vista, which have tremendous processing capability. With such high-performance processors, system designers now have the choice to split the image processing tasks between hardware and software; hence the importance of implementing a high-bandwidth data pipe between the two.

Resulting Platform Architecture

The ideal platform for image processing algorithm development will provide most (if not all) of these processing elements.

Although many of the required elements are already supported by the Virtex-5 family of FPGAs, some items (such as large frame stores and dedicated video I/O) require external components. Hosting the whole system in a PC-based environment is also an ideal way to integrate Xilinx ISE™ software with any proprietary user application.

Figure 2 shows the block diagram of a new FPGA prototyping system that has been developed by Image Processing Techniques (IPT) in conjunction with Xilinx. Called the Advanced Video

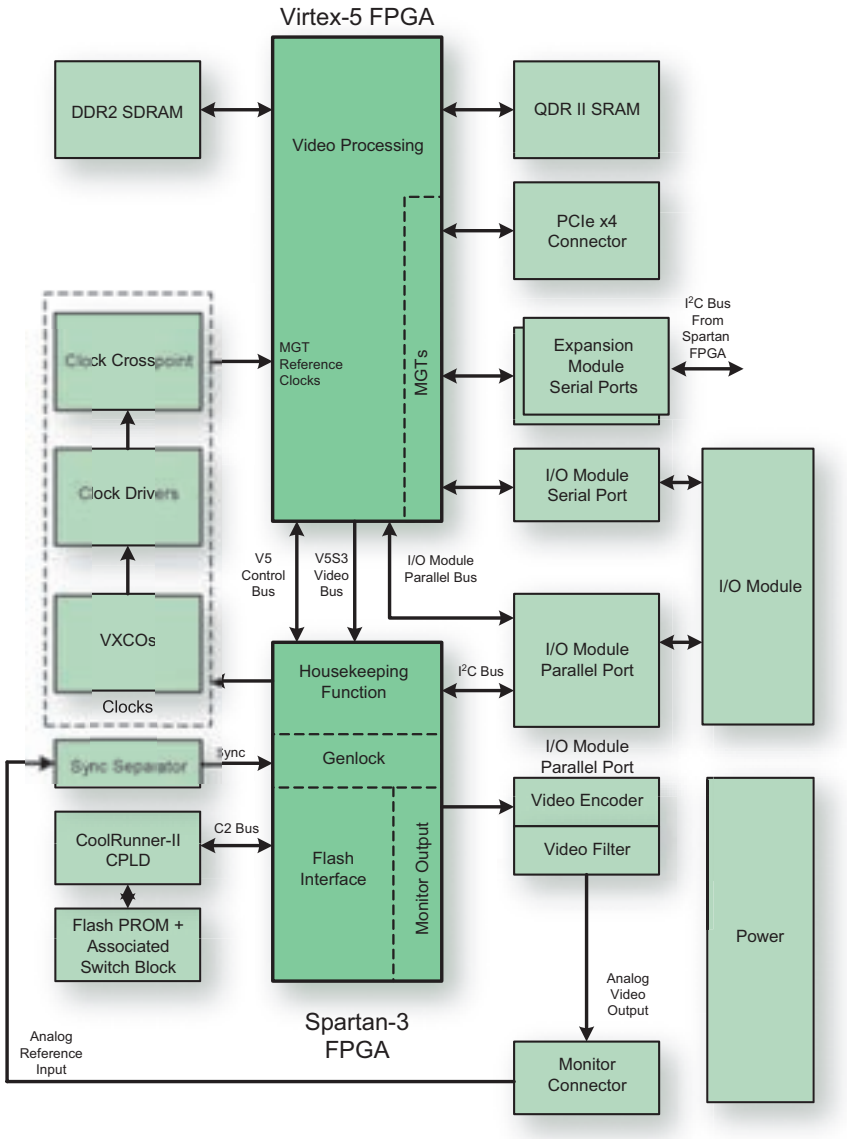


Figure 2 – AVDP block diagram

Figure 3 – The Advanced Video Development Platform

Development Platform (AVDP), the system has been specifically designed to assist engineers with the development of a wide range of image processing algorithms.

The top of the AVDP is shown in Figure 3. It is designed around a Xilinx Virtex-5 FPGA, leveraging its advantages of multiple MGTs for high-speed I/O and an architecture that is well suited to the implementation of filters and color-space conversion arrays. The board also features a PCI Express x4 connector, providing an efficient interface to a PC for software control and analysis; up to 1 GB of high-speed DDR2 SDRAM for bulk video storage; and up to 4 x 36 MB of high-speed QDR II SRAM.

The AVDP system is available with a range of Virtex-5 devices installed, from the LX50T all the way up to the DSP-intensive SX95T, and is provided together with a range of specialist video IP in VHDL source code, software API, device drivers, and a debugging application to help designers develop their image processing applications.

Conclusion

The AVDP board, developed by IPT in conjunction with Xilinx, represents the state-of-the-art in FPGA development systems and addresses the requirements of platforms for image processing algorithm development: flexible video I/O, high-speed memory, and high-performance data acquisition and control.

The key applications that the system is designed to accelerate include:

- Codec design and development: MPEG-2, MPEG-4, JPEG-2000
- Video standards conversion
- Video de-interlacing and resizing algorithms for flat-panel displays
- Broadcast graphics: character generators, still-stores, keyers, and mixers
- Real-time image manipulation for video special effects
- Robotic vision processing and algorithm development

For more information, please visit *www.imageproc.com.*

The Virtex-5 SXT Option for High-Performance Digital Signal Processing

Understanding the demand for high-performance DSP.

by Brent Przybus
Sr. Product Marketing Manager,
Advanced Products Division
Xilinx, Inc.
brent.przybus@xilinx.com

The thirst for data in the information age is driving digital convergence. Triple-play, medical imaging, image processing, defense, aerospace, security, and encryption require higher performance digital signal processing (DSP), putting a strain on traditional solutions, technologies, and techniques.

To understand this, you need only explore the technology advances and resulting increases in algorithm complexity that strain existing solutions. Figure 1 illustrates this point with Shannon's limit, the theoretical maximum information transfer rate of a communications channel at given noise level. As Figure 1 shows, traditional general-purpose processors (GPP) and DSP solutions are unable to efficiently address this limit.

The alternative is to implement digital signal processing algorithms using FPGA technology, enabling performance increases up to an order of magnitude over traditional solutions. The Xilinx® Virtex™-5 SXT device family has been designed to address this alternative FPGA approach, providing a unique ratio of resources to efficiently enable the highest performance digital signal processing possible.

Traditional DSP Versus FPGA DSP

Break up even the most complex DSP algorithm to its atomic parts and you'll find multipliers, adders, and some delay element. By combining these basic elements, you can implement even the most complex DSP algorithms.

Traditional DSP solutions combine sequential control logic with a number of arithmetic logic units (ALUs). The ALUs provide the multiply add capability, while the sequential control logic enables code interpretation and control. Bandwidth is determined by the frequency of the system and the number of ALUs that can run in parallel. A solution that runs at a base fre-

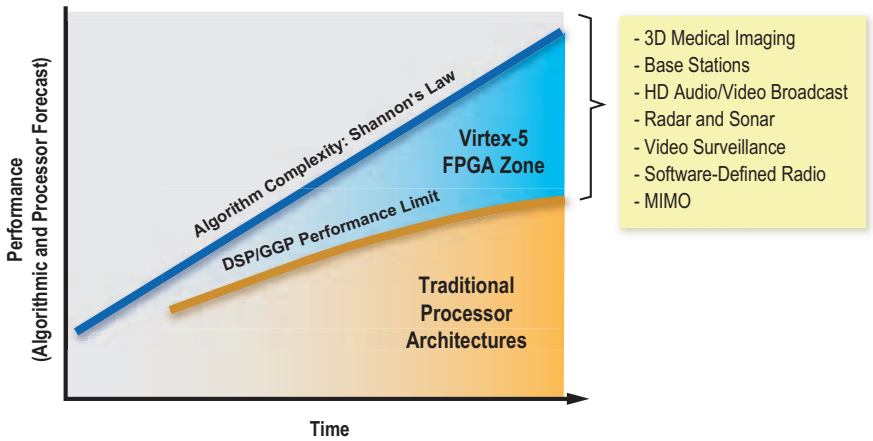


Figure 1 – Digital signal processing performance requirements over time

quency of 2 GHz with eight parallel ALUs will provide 16 GMACs (16 billion multiply accumulates per second) DSP bandwidth.

FPGAs provide user-defined control through programmable logic or a hardened embedded processor. The primary performance enabler is the variable number of DSP slices that provide the multiply add capability (see Figure 2). Bandwidth is determined by the base frequency and the number of DSP slices. The Virtex-5 SXT device runs at up to 550 MHz with as many as 640 DSP slices, providing up to 352 GMACs DSP bandwidth. The Virtex-5 SXT solution provides as much as 40x the number of DSP resources at a given frequency compared to the most advanced traditional DSP solution.

The Virtex-5 SXT High-Performance DSP Solution

The key to achieving this higher DSP bandwidth in Virtex-5 SXT FPGAs is a well-architected solution that achieves 550 MHz performance and reduces system bottlenecks through dedicated resources. The keys to implementing this complete solution are a focus on the following:

- The DSP48E utilizes a systolic implementation technique that allows each of the 640 self-contained DSP48E slices in Virtex-5 devices to operate alone or in combination, using identical interconnects that exist between each of the slices (see Figure 3).

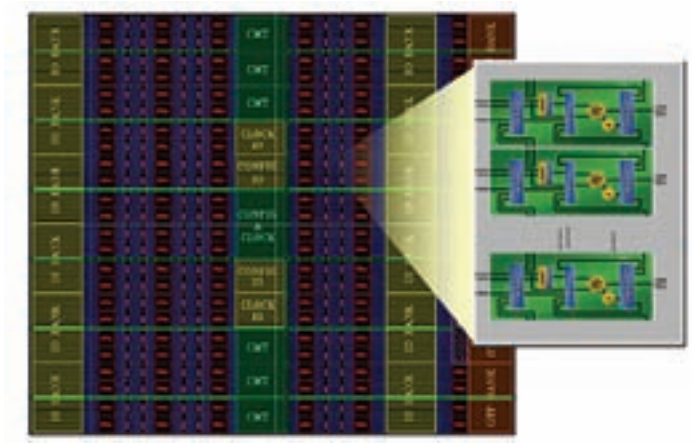


Figure 2 – Virtex-5 SX95T FPGA and DSP48E slice column

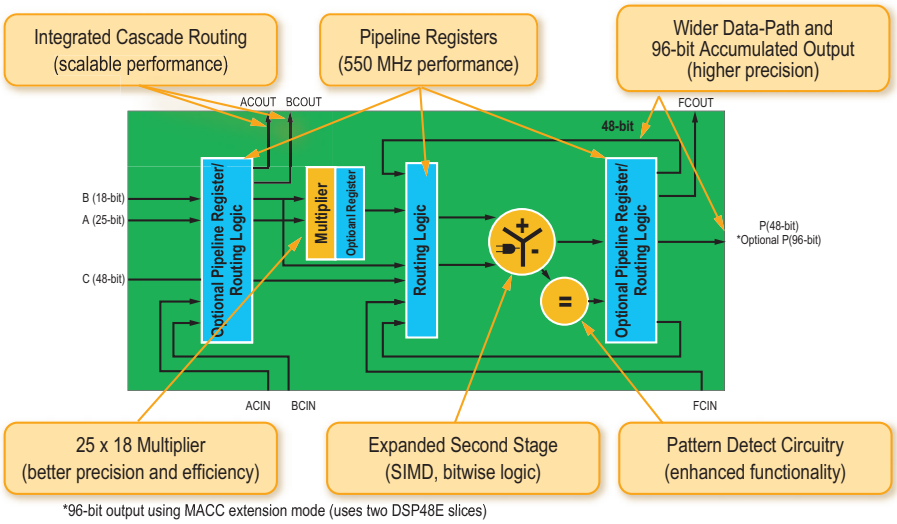


Figure 3 – Single DSP48E slice

- The DSP48E slices in Virtex-5 devices have been optimized for 65-nm process geometry and use advanced MathIP from Arithmetica to achieve maximum performance.
- The entire system – DSP48E slices, processor system, programmable logic, and memory – are all designed to work together using an external system clock operating up to 550 MHz. The system performance is tied together, controllable and sustainable.

Using the Virtex-5 SXT Solution

There are trade-offs between a traditional DSP solution and one using Virtex-5 SXT FPGAs. Traditional DSP leverages a well-established code base that runs on the sequential control logic or processor. This solution enables the reuse of existing code and provides an easy-to-use programming model.

FPGAs provide greater single-chip performance that is flexible and scalable, but requires implementation using a new set of tools or FPGA design techniques. Depending on the application, performance required, or history of the solution, one may be better than the other. Often a combination of solutions is best. To illustrate possible implementations, let's look at three different challenges, possible solutions, and the benefits of these solutions:

- **Example 1:** Legacy system utilizing a complete library of algorithms and implementation techniques that is running out of performance.
- **Plan for Example 1:** Identify DSP performance bottleneck and implement this portion of the design in a Virtex-5 SXT FPGA. Accelerate the design and achieve the required performance.
- **Benefits for Example 1:** This solution allows engineers to maintain legacy code for flexibility, ease of use, and compliance, while at the same time increasing system performance. One of the smaller Virtex-5 SXT devices may be ideal for this for co-processing, or pre-processing acceleration to a tradi-

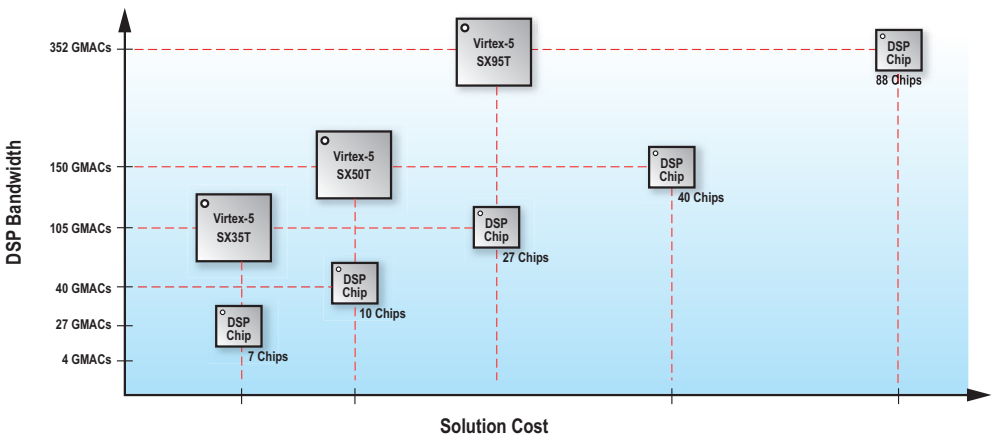


Figure 4 – Comparing performance using multiple DSP chips to a single FPGA DSP solution

tional DSP. Communication between the traditional DSP solution and the Virtex-5 SXT device is accomplished using SelectIO™ technology or using the GTP high-speed serial transceivers. Depending on the complexity of the design and algorithms implemented in the FPGA, outside technical resources may work to facilitate the implementation.

Our next example requires a significant increase in processing performance.

- **Example 2:** Performance requirements necessitate a completely new implementation to achieve much higher performance.
- **Plan for Example 2:** Re-implement the entire design into one of the larger Virtex-5 SXT devices. Use the programmable logic, I/O, and internal block RAM to implement equivalent control functionality. Use the DSP slices to implement higher speed DSP algorithms, leveraging massively parallel design techniques enabled by the systolic architecture. Overcome the new design methodology by using System Generator for DSP or the AccelDSP™ synthesis tool from Xilinx.
- **Benefits for Example 2:** Achieve orders of magnitude higher performance in a single-chip solution. Reduce board space and BOM cost compared to a traditional solution implemented using multiple DSP chips (see Figure 4).

The last example involves a redesign of a current FPGA system to achieve higher performance DSP:

- **Example 3:** Use the DSP slices available in any of the Virtex-5 FPGAs to implement high-performance DSP algorithms in addition to a standard FPGA implementation. As more DSP resources are required, move to a pin-compatible Virtex-5 FPGA platform.
- **Plan for Example 3:** Redesign an existing implementation while leveraging DSP resources, reference designs, and design techniques for Virtex-5 SXT devices.
- **Benefits for Example 3:** Higher performance and an integrated solution, which leverages the scalability and parallelism possible using DSP slices available in any of the Virtex-5 FPGA devices and platforms.

Although the DSP slices available in the Virtex-5 FPGA family enable fast and flexible digital signal processing implementations, you must take into consideration the whole solution. High-speed data plane processing requires not only high-performance DSP, but also fast input, output, and memory to store data and coefficients.

The Virtex-5 SXT device is the only FPGA that combines high-speed multi-gigabit serial transceivers for streaming data into and out of the FPGA, with a unique ratio of block RAM and distributed RAM perfect for sample data and coefficient storage. Finally, the SelectIO interface, with ChipSync™ technology, enables easy and efficient memory interfaces that simplify printed circuit board layouts. Combine this with a flexible programmable technology that enables adaptability to changing standards, and a shorter design time to meet aggressive time-to-market as well as time-in-market requirements.

For cost reduction, the Virtex-5 SXT family is available in an EasyPath™ program that allows up to a 40% cost reduction without losing the use of DSP48E slices or multi-gigabit transceivers (MGTs).

Conclusion

As digital convergence drives digital signal processing performance beyond what traditional solutions can address, explore an FPGA solution. Whether accelerating a traditional DSP solution, embarking on a new high-performance design, or rethinking your current usage of FPGAs, Xilinx has the right technology, tools, IP, and reference designs for you to succeed.

TAKE THE NEXT STEP (Digital Edition: www.xcellpublications.com/subscribe/)

These resources are available today and can help accelerate your design performance regardless of your use model.

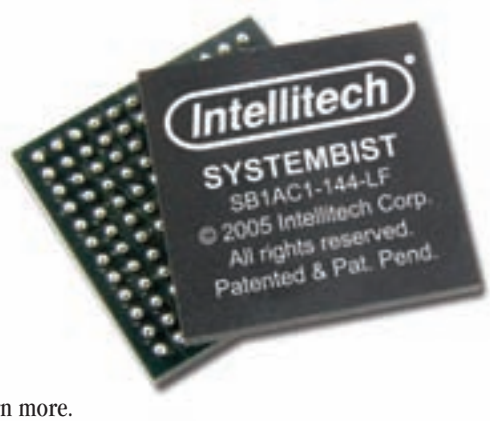
- Buy the Virtex-5 SXT ML506 DSP general-purpose evaluation and development platform, which includes a variety of on-board memories and industry-standard connectivity interfaces.
- View an online demo-on-demand and learn how to accelerate FPGA designs with the AccelDSP synthesis tool and System Generator for DSP.

Great Test, Less Filling

SystemBIST™ enables FPGA Configuration that is less filling for your PCB area, less filling for your BOM budget and less filling for your prototype schedule. All the things you want less of and more of the things you do want for your PCB – like embedded JTAG tests and CPLD reconfiguration.

Typical FPGA configuration devices blindly "throw bits" at your FPGAs at power-up. SystemBIST is different – so different it has three US patents granted and more pending. SystemBIST's associated software tools enable you to develop a complex power-up FPGA strategy and validate it. Using an interactive GUI, you determine what SystemBIST does in the event of a failure, what to program into the FPGA when that daughterboard is missing, or which FPGA bitstreams should be locked from further updates. You can easily add PCB 1149.1/JTAG tests to lower your down-stream production costs and enable in-the-field self-test. Some capabilities:

- User defined FPGA configuration/CPLD re-configuration
- Run Anytime-Anywhere embedded JTAG tests
- Add new FPGA designs to your products in the field
- "Failsafe" configuration – in the field FPGA updates without risk
- Small memory footprint offers lowest cost per bit FPGA configuration
- Smaller PCB real-estate, lower parts cost compared to other methods
- Industry proven software tools enable you to get-it-right before you embed
- FLASH memory locking and fast re-programming
- New: At-speed DDR and RocketIO™ MGT tests for V4/V2

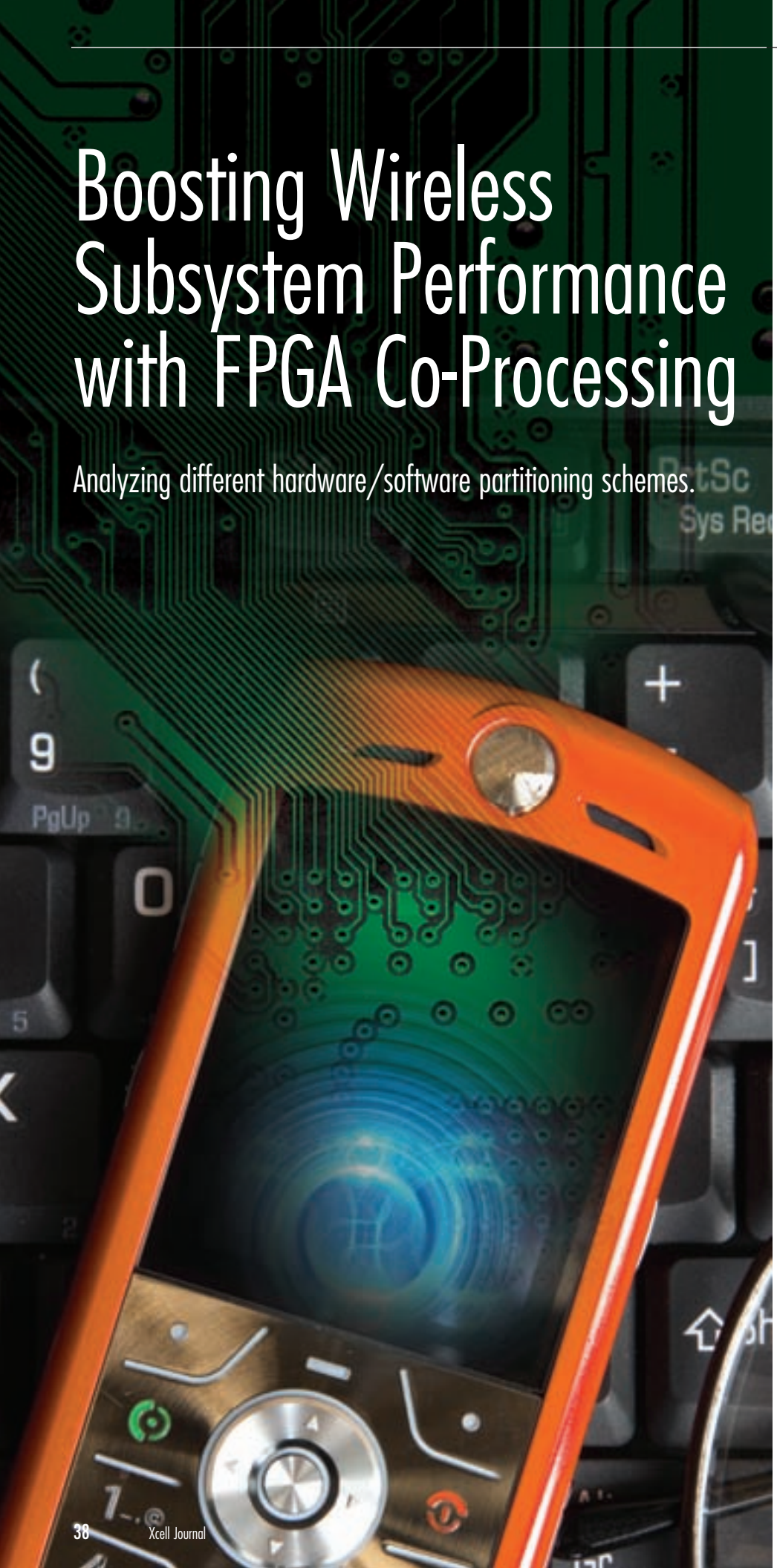


If your design team is using PROMS, CPLD & FLASH or CPU and in-house software to configure FPGAs please visit our website at http://www.intellitech.com/xcell.asp to learn more.

Intellitech®

Boosting Wireless Subsystem Performance with FPGA Co-Processing

Analyzing different hardware/software partitioning schemes.



by Dave Nicklin
Vertical Marketing and Partnerships, Wireless
Xilinx, Inc.
dave.nicklin@xilinx.com

Tom Hill
System Generator Product Manager
Xilinx, Inc.
tom.hill@xilinx.com

You can realize significant improvements in the performance of signal processing functions in wireless systems. How? By taking advantage of the flexibility of FPGA fabric and the embedded DSP blocks in current FPGA architectures for operations that can benefit from parallelism.

Common examples of operations found in wireless applications include finite impulse response (FIR) filtering, fast Fourier transforms (FFTs), digital down and up conversion, and forward error correction (FEC) blocks. Xilinx® Virtex™-4 and Virtex-5 architectures provide as many as 512 parallel embedded DSP multipliers, capable of running in excess of 500 MHz, to provide a peak DSP performance of 256 GMACs.

By offloading operations that require high-speed parallel processing onto the FPGA and leaving operations that require high-speed serial processing on the processor, you can optimize overall system performance and cost while lowering system requirements.

Sub-System Partitioning Choices

The FPGA can be used with a DSP processor, serving either as an independent pre-processor (or sometimes post-processor) device, or as a co-processor. In a pre-processing architecture, the FPGA sits directly in the data path and is responsible for processing the signals to a point when they can be efficiently and cost-effectively handed off to a DSP processor for further lower-rate processing.

In co-processing architectures, the FPGA sits alongside the DSP, which offloads specific algorithmic functions to the FPGA to be processed at significantly higher speeds than what is possible in a DSP processor alone. The results are passed back to the

DSP or sent to other devices for further processing, transmission, or storage (Figure 1).

The choice of pre-processing, post-processing, or co-processing is often governed by the timing margins needed to move data between the processor and FPGA and how that impinges on the overall latency. Although a co-processing solution is the topology most often considered by designers – primarily because the DSP is in more direct control of the data hand-off process – this may not always be the best overall strategy.

Consider, for example, the latest specifications for 3G LTE, in which the transmission time interval (TTI) has been reduced to 1 ms, down from 2 ms for HSDPA and 10 ms for WCDMA. This essentially requires that data be processed from the receiver and through to the output of the MAC layer in less than 1,000 µsec.

Figure 2 shows that using an SRIO port on the DSP running at 3.125 Gbps, with 8b/10b encoding and a 200-bit overhead for the Turbo decode function, results in a DSP-to-FPGA transfer delay of 230 µsec (that is, nearly a quarter of the TTI period just to transfer the data). Taking into account other expected delays, the Turbo codec performance required to meet these system timings is a very demanding 75.8 Mbps for 50 users.

Using an FPGA to process the Turbo codecs as a largely independent post-processor not only removes DSP latency but saves time because there's no need to transfer the data at a high bandwidth between the DSP and FPGA. This reduces the throughput rate of the Turbo decoder down to 47 Mbps, a decrease that allows you to use more cost-effective devices and have reduced system power dissipation.

Another consideration is whether to use soft- or hard-embedded processor IP on the Xilinx FPGA to offload some of the system processing tasks, which in turn offers the possibility of additional cost, power, and footprint reduction benefits. Given such a wide range of signal processing resources, complex functions such as those found in baseband processing can be more optimally partitioned between the DSP processor, the FPGA-configurable logic blocks (CLBs), embedded FPGA DSP blocks, and FPGA embedded processor. Xilinx offers two types of embedded processors: the MicroBlaze™ soft-core processor – often used for system control – and the higher performance PowerPC hard-core embedded processor for more complex tasks.

FPGA-embedded processors provide an opportunity to consolidate all non-critical operations into software running on the embedded processors, minimizing the total amount of hardware resources required for the overall system.

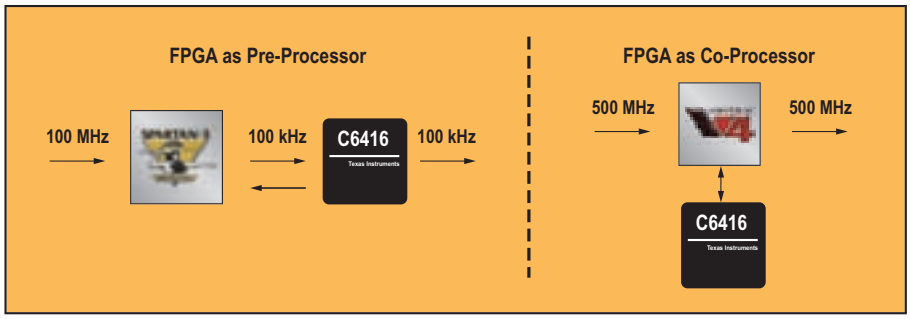
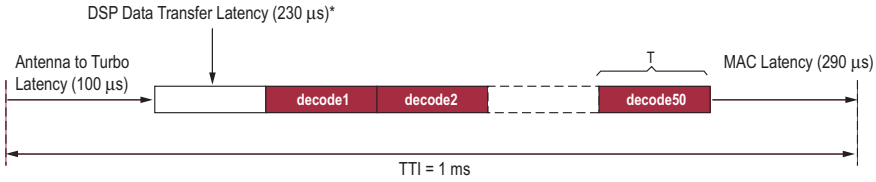


Figure 1 – FPGA as a pre-processor and co-processor solution



Implications for Turbo Decode (50-user example):
Processing bandwidth available for Turbo Decode = 380 µs
Throughput rate necessary = (576 bits × 50 users) / 380 µs = **75.8 Mbps**
Throughput rate *without* DSP transfer latency = (576 bits × 50 users) / 610 µs = *47 Mbps*

Data-transfer latency directly impacts cost and power dissipation.

Figure 2 – LTE example of co-processing data-transfer latency issues

The Importance of Software and IP

A critical issue is how to unlock all of this potential capability. You need to consider both the software needed to abstract the complexity of the problem and the availability of IP, focused on key areas where FPGAs can provide an optimal solution.

Xilinx is dedicated to developing industry-leading tools and ecosystems that allow you to produce efficient FPGA implementations from a higher level of abstraction than what HDL tools such as MATLAB models and C code can provide. With development tools like Xilinx System Generator for DSP and the AccelDSP™ synthesis tool, you can go from algorithm to silicon as seamlessly as possible.

There is also an increasingly important ecosystem of tool providers whose products take development up to the electronic system level (ESL) through C/C++-to-gate design flows. ESL design tools are aimed at providing an integrated system-level approach to the production and integration of hardware-accelerated functions and the control code for processors controlling these functions.

No single high-level language or software tool is suitable for all of the different elements found in today's complex systems. The choice of language and design flow is

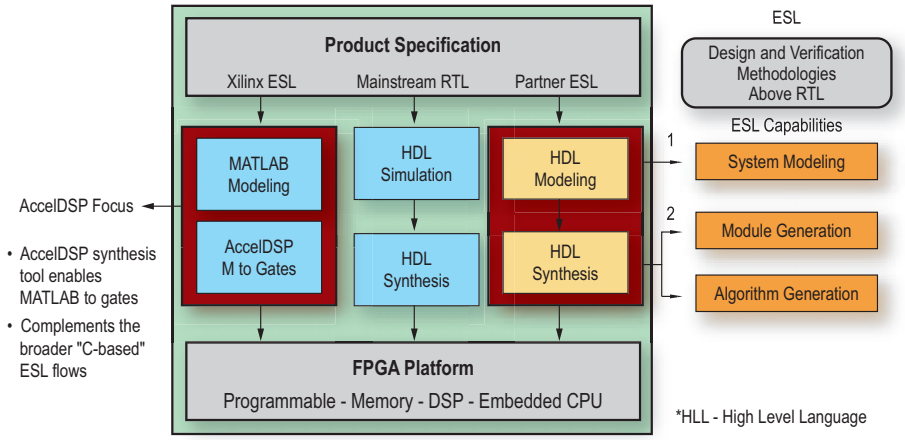


Figure 3 – System level-to-FPGA design flows

governed by the customer and sometimes the individual engineer. Xilinx has therefore developed a comprehensive set of integration features in order to meet customer needs and provide the most optimal design environments (see Figure 3).

Conclusion

Xilinx is also making significant investments to provide a comprehensive suite of high-value IP, boards, and reference designs that cover many critical areas in radio card and baseband applications. These include FFT/iFFT, modulation, digital up and down conversion, and crest factor reduction.

An example of this focused approach has been the development of industry-leading high-performance FEC functions, such as Turbo encoders and decoders, optimized for specific wireless standards and FPGA architectures. As we demonstrated in our analysis of 3G LTE latency and Turbo decoder throughput requirements, hardware acceleration of FEC functions and their impact on system architecture is an increasingly important necessity in modern wireless equipment design.

Although some specialist DSP processors integrate such functions as embedded blocks over time, it can take many months from when the parameters of an FEC function for a new wireless standard are set before the resulting embedded acceleration block appears in silicon. Once embedded, there are still challenges remaining, and occasions when not all of the functionality in the embedded blocks work as required. In the meantime, standards have quickly evolved to include new requirements that cannot be supported by the fixed embedded blocks.

Because of these situations, designers require flexibility. They are looking for the ability to swiftly develop and deploy complex baseband functions such as FEC and then adapt them based on feedback from field trials and evolving standardization efforts. Perhaps they wish to add their own proprietary IP to differentiate their solution in the marketplace. It is in these circumstances that designers should not simply consider the current solution portfolio from a supplier, but also investigate how easily the solutions can be adapted and what level of support and tools the supplier can provide.

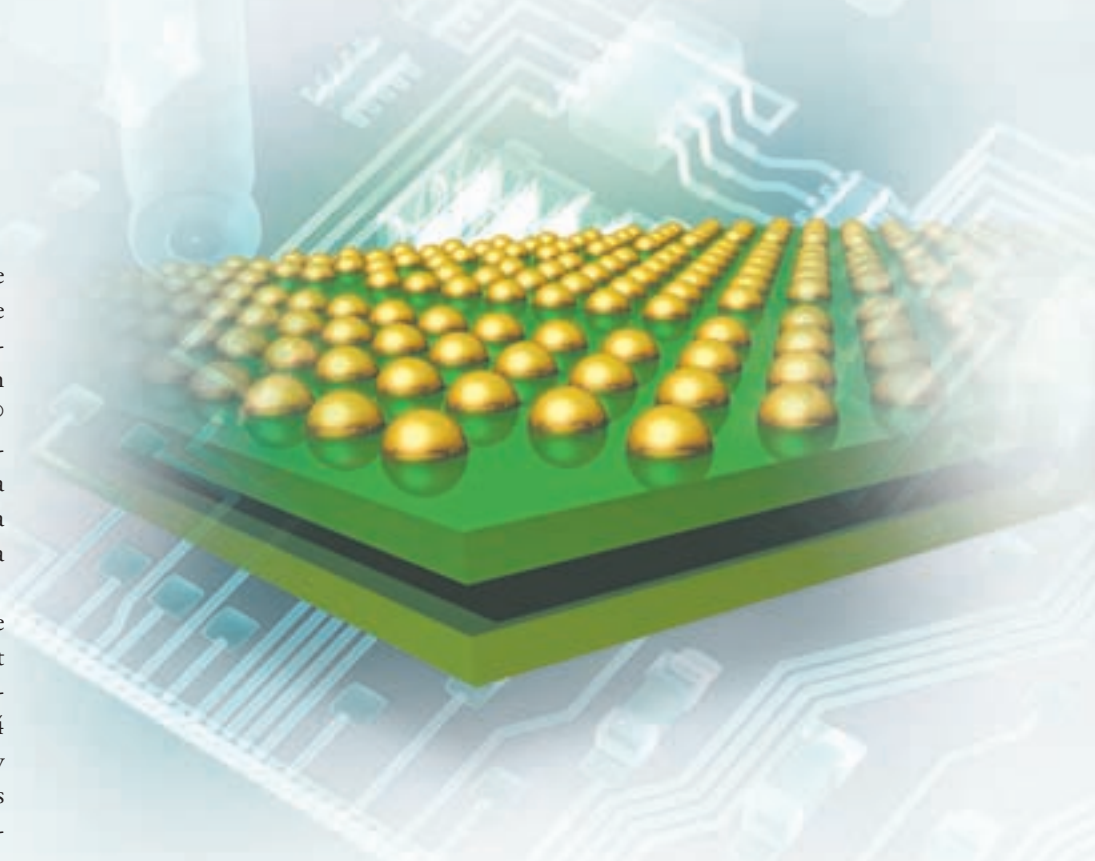
TAKE THE NEXT STEP (Digital Edition: www.xcellpublications.com/subscribe/)

- Get more details on Xilinx DSP tool flows and our ecosystem partners.
- See the complete range of solutions available to wireless systems design engineers.

Selecting the Right Memory Controller for Real-Time Applications

You can use performance modeling to compare the MPMC memory controller and CoreConnect bus for high-bandwidth DSP applications.

by Deepak Shankar
Founder and CEO
Mirabilis Design Inc.
deepak.shankar@mirabilisdesign.com



Imaging and networking applications require large data movement that must be available to the application on a specific time schedule. The embedded processor and custom co-processors in the fabric of Xilinx® Virtex™-4 and Virtex-5 FPGAs provide significant processing capacity to handle data processing algorithms. This means that data moving to and from external DRAM has a significant impact on response time.

In this article, I'll show how you can use performance analysis to evaluate the impact on task response time using different memory access techniques on the Virtex-4 FPGA. We conducted an evaluation by constructing a virtual prototype in Mirabilis Design's VisualSim, a performance modeling and architecture exploration solution.

An add-on library to VisualSim called the VisualSim Xilinx FPGA Modeling Toolkit provides components that represent common hard and soft IP available for the Virtex-4 FPGA. You can simply drag components from this library, instantiate them in a graphical block diagram editor, and modify parameters to represent the specific implementation under study. To learn more about developing virtual prototypes, see the article in Third Quarter 2006 issue of *Xcell Journal*, "Accelerating Architecture Exploration for FPGA Selection and System Design."

We used a data-dependent DSP algorithm for the evaluation. Our exploration looked at processor stalls, cache hit ratios, I/O throughput, and latency per task. These virtual prototypes provided the flexibility to experiment with different architecture configurations and select the solution that best met our requirements. Figure 1 shows an example of a virtual prototype developed in VisualSim using components from the VisualSim Xilinx FPGA Modeling Toolkit. This example shows a model containing the e405, CoreConnect bus, SDRAM, DMA, and other interface cores.

Project Overview

There are a number of techniques for I/O processing to minimize the cycle count for data and instruction access from external SDRAM. The Xilinx multi-port multi-channel (MPMC) memory controller is an extremely efficient technique for interfacing the processor and key high-speed devices to SDRAM. Another popular method is to use a Xilinx CoreConnect bus with multiple masters, including the processor instruction/data caches and key high-speed devices connected by a slave port to the SDRAM.

To investigate the similarities and differences between the two approaches, we constructed models of both configurations using the VisualSim Xilinx FPGA Modeling Toolkit. Figure 1 shows the virtual prototype and Figure 2 is the timing diagram for request, acknowledge, read, and write access on the CoreConnect bus.

Design Considerations

We used the same configuration to explore the Xilinx MPMC memory controller and CoreConnect bus. The instruction and data channels of the PowerPC e405, Ethernet, and PCI interface were connected as masters; the SDRAM is a slave. The Ethernet and PCI data are burst to SDRAM using dedicated CDMAC engines. The masters and slaves were maintained at constant speed and size in both virtual prototypes. Table 1 shows the instruction and data cache statistics for the virtual prototype model in Figure 1.

The design considerations were:

- What is driving the design – overall performance or a combination of power and performance?
- Do I need one or two e405 PowerPCs for my core tasks?
- If the e405 PowerPC is running at 400 MHz, what might be the best MPMC or CoreConnect bus clock ratio?
- Does my design have equal read and write memory activity, multiple reads for one write, or multiple writes for one read?
- What is the effect of SDRAM speed on the processor/memory controller/bus clock ratio?
- What is the effect of SDRAM speed on added application access to memory?

Analysis

We ran the simulation on a 1.6-GHz Microsoft Windows XP (SP2 and Standard Edition) with 512 Mb of cache. We simulated 3.0 ms of real time and the VisualSim simulation took 28 seconds of wall clock time to finish. The model was constructed

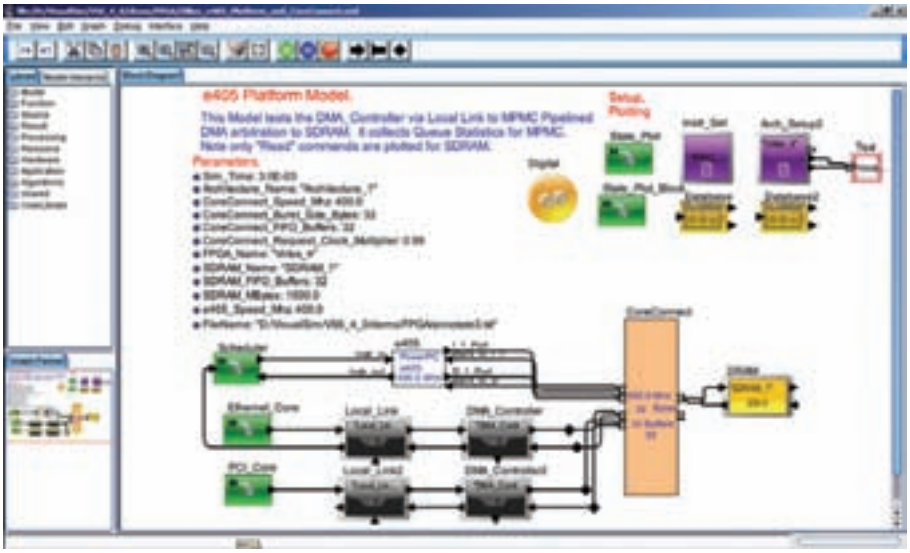


Figure 1 – VisualSim model of the Xilinx Virtex-4 platform: e405 with CoreConnect bus model

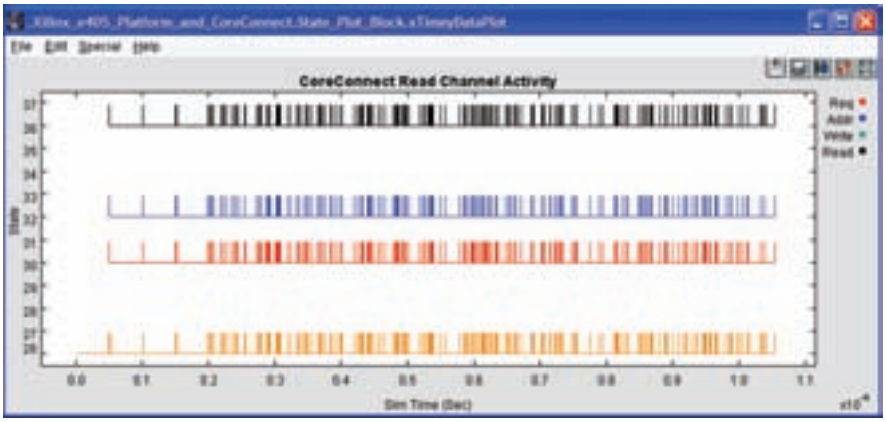


Figure 2 – CoreConnect bus activity with SDRAM at bottom

Statistic Name	CoreConnect		MPMC	
	Instruction Cache	Data Cache	Instruction Cache	Data Cache
Hit_Ratio_Max	95.05	96.23	94.98	97.37
Hit_Ratio_Mean	90.51	90.62	94.98	97.37
Throughput_MIPs_Max	74.45	1.69	7.44	0.16
Throughput_MIPs_Mean	7.44	0.17	7.44	0.16
Utilization_Pct_Max	18.61	0.42	1.86	0.04
Utilization_Pct_Mean	1.86	0.04	1.86	0.04

Table 1 – Instruction and data cache statistics

and validated in four days using standard elements in the VisualSim library.

Using the 200-MHz MPMC, the end-to-end latency for the execution of the application benchmark was 87.190 µs, while the 400-MHz CoreConnect was 86.42 µs. Both matched our real-time threshold.

We found that the MPMC offered uniform latency cycles for cache to SDRAM accesses, whereas CoreConnect was more variable in cache to SDRAM access times. The lowest cycle count was achieved with the CoreConnect bus, but its average count was significantly higher. At the lower clock rate, CoreConnect performance deteriorated significantly. Except for eight out of the 33 tasks, the MPMC Memory Controller typically finished its tasks faster than the CoreConnect bus.

Table 3 shows the e405 processor stalls statistics. The CoreConnect caused a peak processor stall of 19.64%; this caused excessive latency for certain tasks. When the CoreConnect was stalled, the impact on the application latency was higher than with the MPMC. This also confirmed our finding that the CoreConnect bus had a higher overall latency and processor stalling percentage. Increasing network traffic did not cause any noticeable increase in MPMC latency, although Ethernet traffic increases did cause the overall CoreConnect latency to increase.

The MPMC memory controller had a significantly better average hit ratio (94% versus 90.51%) for the I-cache (instruction). At certain times during the simulation, the CoreConnect bus did get to a 94% hit ratio, but the duration was very short. This indicated that the MPMC arbitrated SDRAM requests better than the CoreConnect bus.

Task Name	Cycles in Processor	
	MPMC Memory Controller	CoreConnect Bus
DFT	108	109
DFT	735	742
DFT	716	712
CS_Weighting	448	523
IR	850	881
Q_Taylor_Weighting	439	523
CS_Weighting	437	523
IR	842	934
Q_Taylor_Weighting	463	492
CS_Weighting	460	549
IR	832	879
Q_Taylor_Weighting	459	515
CS_Weighting	466	523
IR	851	871
Q_Taylor_Weighting	486	502
CS_Weighting	521	513
IR	834	879
Q_Taylor_Weighting	533	482
CS_Weighting	518	518
IR	1,565	1,649
DFT	763	743
DFT	2,671	2,671
DFT	762	748

Table 2 – Task latency in cycles for key tasks in the application benchmark software

Statistic Name	CoreConnect	MPMC
Stall_Time_Pct_Max	19.64	1.99
Stall_Time_Pct_Mean	1.96	1.99
Task_Delay_Max	6.823E-06	6.750E-06
Task_Delay_Mean	2.372E-06	2.374E-06

Table 3 – Latency and stall activity statistics

Looking at the number of threads (33) and the kilobytes per thread (1.8), we could see that having a 64-KB cache for the PowerPC could reduce SDRAM latency and achieve a 100% hit ratio for the I-cache. The analysis could be extended to include the impact of multiline prefetch and duration of loops in the application code. Table 2 shows the latency to execute the individual tasks within the application benchmark for both architectures.

Conclusion

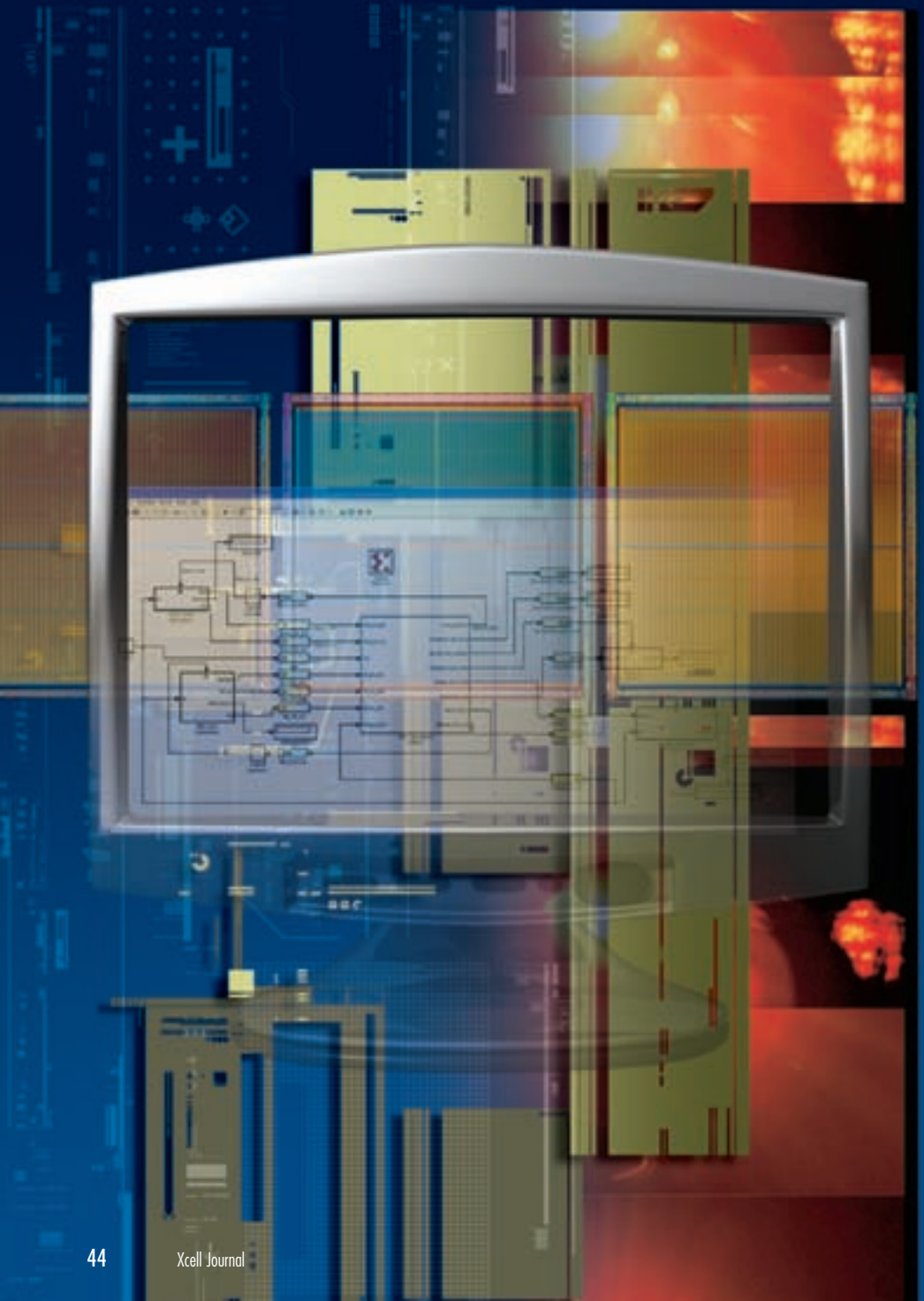
Virtual prototypes and performance modeling allowed us to explore a number of scenarios and clock configurations. The statistics generated provided full visibility into the operation of the FPGA design. Moreover, the fully validated VisualSim FPGA Modeling Toolkit components allowed us to save time without compromising overall accuracy. The toolkit has accurate models and extensive design statistics for Xilinx FPGAs, thus making relative comparisons between different configurations possible.

Mirabilis Design provides a performance and architecture analysis solution for the Xilinx Virtex FPGA family. Our VisualSim graphical environment contains a series of building blocks that emulate the performance characteristics of common Xilinx IP and cores. Using this library of building blocks, you can construct a specification-level simulation model of a proposed system containing multiple FPGA and on-chip resources such as memories, sensors, and buses. Model construction is a process of connecting icons that represent the IP in a graphical editor.

For details on Mirabilis and VisualSim, visit *www.mirabilisdesign.com*. To experience virtual prototyping, try running pre-built simulation models at *www.mirabilisdesign.com/WebPages/mdi_demonstration/mdi_demonstration.htm*.

Integrating HDL Design and Verification with System Generator

Designing and verifying the CABAC functionality of an H.264 video encoder is easy with System Generator for DSP.



by Justin Delva
Staff DSP Engineer
Xilinx, Inc.
justin.delva@xilinx.com

Ben Chan
Senior Software Engineer
Xilinx, Inc.
ben.chan@xilinx.com

Shay Seng
Engineering Manager
Xilinx, Inc.
shay.seng@xilinx.com

Xilinx® System Generator for DSP is a MATLAB Simulink block set that facilitates system design. Targeting Xilinx FPGAs within the familiar MATLAB environment, System Generator for DSP gives you the ability to functionally simulate a design and use the MATLAB environment to verify bit- and cycle-true models against the golden reference results. These reference results can be produced either externally or inside the MATLAB environment, and you can target a Xilinx FPGA hardware platform all from within MATLAB.

System Generator complements HDL design tasks by providing an easily configured test bench for both functional simulation and hardware verification. You can simulate your HDL code within MATLAB by using the built-in interface to HDL simulators like ModelSim. A System Generator for DSP test bench platform built around the HDL code provides a powerful yet fast simulation environment that interacts seamlessly with ModelSim. Setting up this environment is easy.

You can use this same environment to test your HDL code running in real hardware without any modifications. The hardware co-simulation system uses pre-supported FPGA platforms such as the Xilinx ML506 board for performing either a Simulink-controlled stepped clock hardware run or a real-time data-burst run.

Systematic Design and Verification of a CABAC Module

The H.264/AVC video encoder is the product of years of collaborative efforts that resulted in a standard with good video quality at substantially lower bit rates than previous standards. A reference C source code called the H.264/AVC Joint Model (JM) is available to developers. You can use this source code as a starting point for the functionality implemented in HDL.

Context-adaptive binary arithmetic coding (CABAC) is part of the H.264 video standard. The functionality of the CABAC module is manually translated in HDL using standard communication primitives. The primitives mostly used in this verification are FIFO interfaces. The original JM source code is also used for generating test vector files for the module.

We have also built a test environment that uses the JM model to generate the input stimuli for the CABAC HDL module and verify the results of the output of the HDL with the results produced by the JM reference model. This is a significant improvement over traditional HDL ad-hoc test benches.

Module verification is a three-step process comprising:

1. Functional HDL simulation. MATLAB verification with input and output test vectors feeds the ModelSim simulation and compares results, respectively.
2. Functional hardware verification. An intermediate step for working out any bugs not caught during functional HDL simulations. This stage uses a System Generator for DSP-controlled single-step clocking. The input and output test vectors are extracted in files from the original execution of the JM source model. Special care is taken to build file interfaces in System Generator for DSP.
3. Real-time hardware verification. Using the input test vectors of the functional HDL simulation, the design is tested in hardware at the targeted input rate and clocking frequency. The output of the hardware is captured into MATLAB and compared with the output test vectors.

Functional HDL Simulation

In this step, we integrate the ModelSim simulation with the System Generator for DSP simulation, as shown in Figure 1.

To simulate the HDL in System Generator for DSP, a black box of the top VHDL entity is created by inserting converters at the boundaries, also shown in Figure 1. These converters translate the ModelSim unknown "X" states to zeros in the Simulink simulation. The test bench setup of the total simulation is shown in Figure 2.

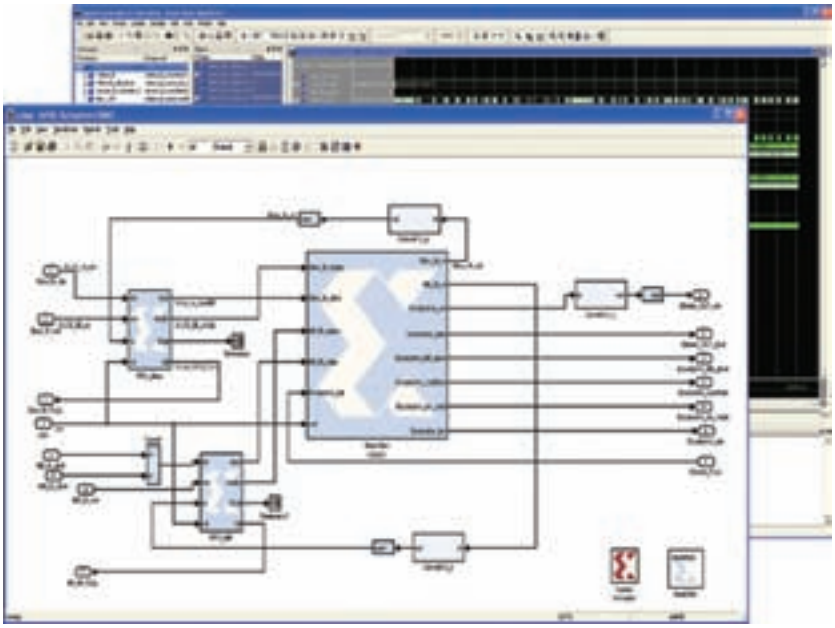


Figure 1 – Black box of CABAC HDL and ModelSim co-simulation

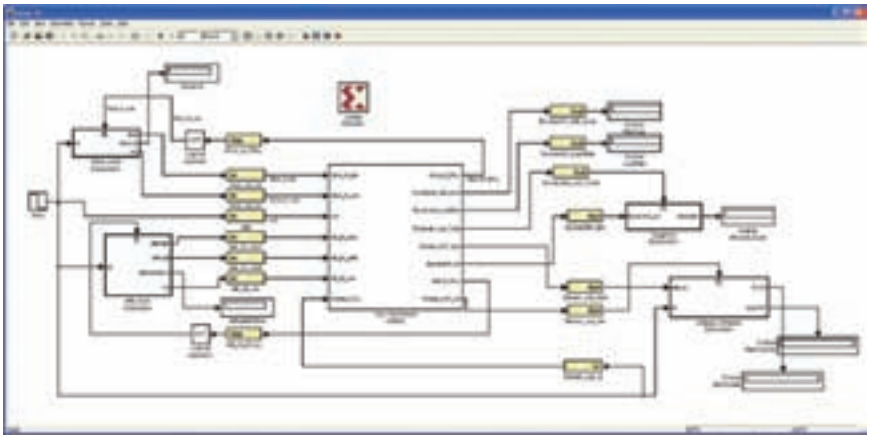


Figure 2 – Test bench setup for HDL simulation of the CABAC design

In Figure 2, the "Slice_input subsystem" block and "MB_input subsystem" block use special interface code to read from files containing stimuli created by the JM source code. The "Output_compare" subsystem is a special block that compares the results of the simulation with the original test vector results from the JM source code. This simulation is done on a single-step basis.

Functional Simulation with Hardware Using the ChipScope Analyzer

The next step is to use a similar environment where the complete HDL module is mapped onto hardware, in this case using an ML506 board with Ethernet and JTAG connections. The Ethernet connection is used to provide and read the stimuli, while the JTAG port is used to connect the ChipScope analyzer. This provides the same user experience, but now the HDL is completely implemented in hardware. The system setup is shown in Figure 3.

The advantage of this setup is that you can abstract completely from the details of the specific interfaces. You do not need to learn how the Ethernet connection is used to provide the input stimuli or read the output from the CABAC block. The special gateway blocks shown in Figure 3 abstract completely from these details.

Real-Time Hardware Verification

Previous simulations and executions provide a good environment for the detailed execution of the block at the cycle level. In a complete design, you often want to use large test sets that are representative of real-world test cases. The single cycle or step interface is not suited for this style of verification.

Using the same hardware setup with an ML506 board, verification now occurs with large data sets provided through a novel MATLAB interface called M-HWcosim. M-HWcosim is an API that transfers data to hardware from a MATLAB script M-file. Now the MATLAB scripting environment is used to provide all of the data to the actual CABAC block running in the hardware.

The implementation of FIFOs with flow control allows asynchronous communication between the computer running MATLAB and the hardware running the CABAC block at full speed. This environment abstracts the details of this interface and has been critical in verifying large data sets of the CABAC module. For details about this environment, see the white paper, "Using System Generator for Systematic HDL Design, Verification, and Validation," on *www.xilinx.com*.

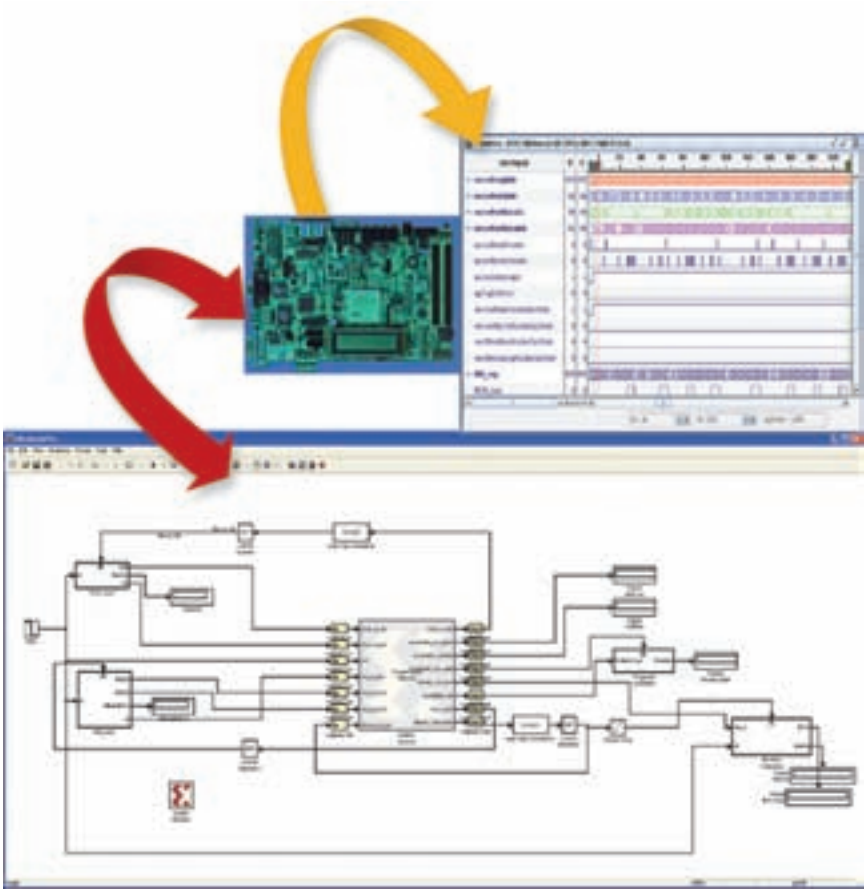


Figure 3 – Execution of the CABAC design in hardware, with Ethernet interface for stimuli and JTAG for the ChipScope analyzer.

Conclusion

In a complete system design, the verification is often as much work as the actual design. The design of the CABAC block in H.264 leverages the JM source code model for generating test vectors from a high-level language. Here the HDL design verification is integrated with System Generator for DSP and MATLAB. Furthermore, the integration with complete boards that can run the CABAC block at speed is a major improvement over ad-hoc environments. This substantially reduces the time needed to build a verification environment, and therefore allows you to focus on the actual block at hand.

TAKE THE NEXT STEP (Digital Edition: www.xcellpublications.com/subscribe/)

- Familiarize yourself with System Generator for DSP by taking the free online recorded lecture, "System Generator, Getting Started Training."
- Download the System Generator for DSP User Guide.
- Order the Xilinx H.264 CABAC core.

Accelerating System Development Cycles with the Radar Blockset Library

A radar blockset library offers components selectable from the generic Simulink environment for complex system designs using FPGAs.

by Meena Das
Contract Engineer
Electronics and Radar Development Establishment (LRDE), DRDO
dmeenasatish@yahoo.co.in

Karthik Kabbinahitlu Subrahmanya
Contract Engineer
Electronics and Radar Development Establishment (LRDE), DRDO
karthik.urimajalu@gmail.com

Taniza Roy
Scientist "C"
Electronics and Radar Development Establishment (LRDE), DRDO
tanizaz@yahoo.com

Gnana Michael Prakasam
Scientist "F"
Electronics and Radar Development Establishment (LRDE), DRDO
lgmprakasam@yahoo.com

Designers often struggle with time-consuming development cycles during complex system realization using FPGAs. We have designed a radar blockset library in The MathWorks Simulink environment that addresses this challenge by providing pre-synthesized components.

Our radar blockset library components directly generate programming code for Xilinx® FPGAs on a target board. Thus, the blocks lead to a single design solution rather than having to use multiple blocks for dedicated functions.

For applications such as complex radar systems, a single design solution speeds up development activity and keeps designers from having to design from scratch. To highlight this feature, we'll describe how a typical radar blockset library component works in a radar receiver design.

Figure 1 – Radar blockset library

You can use these library components just as you would use Xilinx System Generator components. The blocks are optimized for FPGA resource utilization and are pre-synthesized and compiled in the radar blockset library. Each component in the library is equipped with the required parameterization block and help menus for configuration and proper usage, just like System Generator block sets.

The first version of the radar blockset library comprises a quadrature demodulator for generating complex video from real samples, various radar interface blocks, a sensitivity time control (STC) block, and a signal processing block (Figure 1). Designers widely need these components when designing complex systems.

Our team designed the radar blockset library and its components using VHDL and the System Generator tool. It is exclusively intended for implementation on Xilinx FPGAs.

Integrated Library Components

Let's explain the principles on which we designed each of the components of the radar blockset library.

Quadrature Demodulator

The quadrature demodulator disintegrates real input sampled signals into in-phase and quadrature (I and Q) components. It then demodulates the intermediate frequency-to-baseband signal.

You can use this component as a part of your receiver design and generate the bit file directly from the design model to program the FPGA.

STC Block

Sensitivity time control is a gain control method for radar receivers. Gain control adjusts the sensitivity of the receiver gain as a function of range, thereby regulating the intensity of the returns. STC causes receiver gain to vary with range in such a way that receiver output is less dependent on range, thus reducing the effect of clutter on receiver saturation. STC helps in increasing the dynamic range of the receiver and prevents receiver saturation by selecting suitable STC laws in real time. Even though complex logarithmic calculations are required to realize this function, this library component aids designers because they can use the library components in a new design model through simple drag-and-drop operation.

Radar Interface Block

The interface block comprises various components such as a global positioning system (GPS) receiver interface block, Manchester code inversion protocol interface block, controlled receiver interface block, and RS232 standard interface block.

GPS Receiver Interface Block

This module interfaces a system with GPS receivers. It expects the receiver to work according to the Trimble Standard Interface Protocol (TSIP). The component was initially designed to interface a radar system with a GPS receiver, but you can use it wherever a GPS receiver interface is required with the same protocol.

The initialization, request, and report packets are stored in FPGA memory as per TSIP. Initialization packets set the GPS parameters when the GPS is initialized. These packets decide the GPS's mode of operation, the filters to be used, and the I/O port status. After initialization is complete, the request packets are used for inquiring about the health and status of the GPS. Report packets handle all of the inquiries made by the system to the GPS receiver.

CMI Protocol Interface Block

This module serves as an interface between the beam steering unit (BSU) and another radar subsystem. The BSU message comprises a header word followed by descriptive words. The number of words that will follow is indicated in the header word. Standard CMI protocol is used for coding these messages. This module provides an interface solution for any system based on the CMI protocol.

Controlled Receiver Interface Block

This module interfaces systems to other systems by providing a control signal to initiate the reception process, according to the RS232 standard for radar applications. The block expects a reply based on a control signal within a fixed duration, which is decided by the next occurrence of a control signal.

RS232 Standard Interface Block

This component is designed to support communication controller implementation according to the RS232 standard for serial communication. It supports both transmission and reception processes. You can configure the component with a suitable baud rate for communication from the component GUI.

Signal Processing Block

This block comprises various signal processing functions such as pulse compression and Doppler filter processing. The pulse compression block involves the transmission of a long coded pulse and the processing of the received echo to obtain a relatively narrow pulse. The Doppler filter block extracts Doppler information from radar echo signals.

Interactive Component Help

Selecting the component provides the first level of information about it, similar to how other blocks are described in Simulink. Additional detailed information is provided as a help file in Simulink (Figure 2). These help files support designers in component configuration.

Figure 2 – Interactive component help

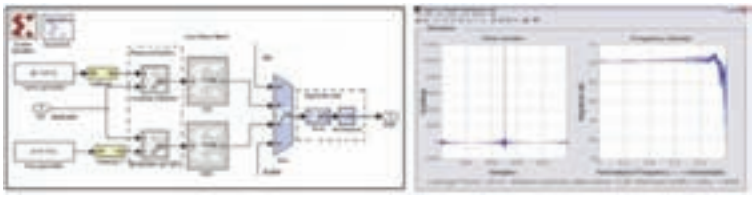


Figure 3 – Design model for the quadrature demodulator component, with its pulse compression result

Radar Receiver Example

The quadrature demodulator component module forms an important building block of modern receivers for the demodulation of intermediate frequency (IF) signals and generation of video signals. This component expects a sampling frequency four times the intermediate frequency. The sampled input signal at baseband frequency forms the real-time input to the demodulator component. In addition to this, the component expects the corresponding I and Q multiplication coefficients for configuring the library component.

The design of the quadrature demodulator component comprises block designs for functions such as filtering, quadrature component extraction, sign extension, decimation, and I and Q signal multiplexing (Figure 3). Selecting the appropriate filter in the digital domain for the extraction of the I and Q components is crucial in the design.

The radar blockset component model from Simulink can be directly used as a System Generator model to realize the quadrature demodulator. Configuring the component for the new design is as easy as dragging and dropping the component from the radar blockset library. The functional design model expects the required inputs in the proper format for hardware realization. Like any other System Generator model, the component will generate code for Xilinx FPGAs, which you can then download to your target systems. This process is much faster than designing with discrete models.

Let's explain how you could use the component for a radar receiver design. Figure 1 shows a design model for a quadrature demodulator component in a typical radar receiver test case. The component inputs are taken as echo samples of linear frequency modulated (LFM) code at 20 MHz, an IF of 5 MHz, and a bandwidth of 2.5 MHz. The I and Q multiplication coefficients are provided as [0 -1 0 1] and [-1 01 0], respectively.

We evaluated the model for logic and timing in the simulation environment by using ModelSim software, resulting in the generation of the required I and Q components. To check and validate component output, it is pulse-compressed with complex conjugate LFM code at baseband frequency (Figure 3). As the output of the component is I and Q signal-demodulated at baseband frequency, the output of the pulse compression shows a perfect peak; thus the function of the component is validated.

Conclusion

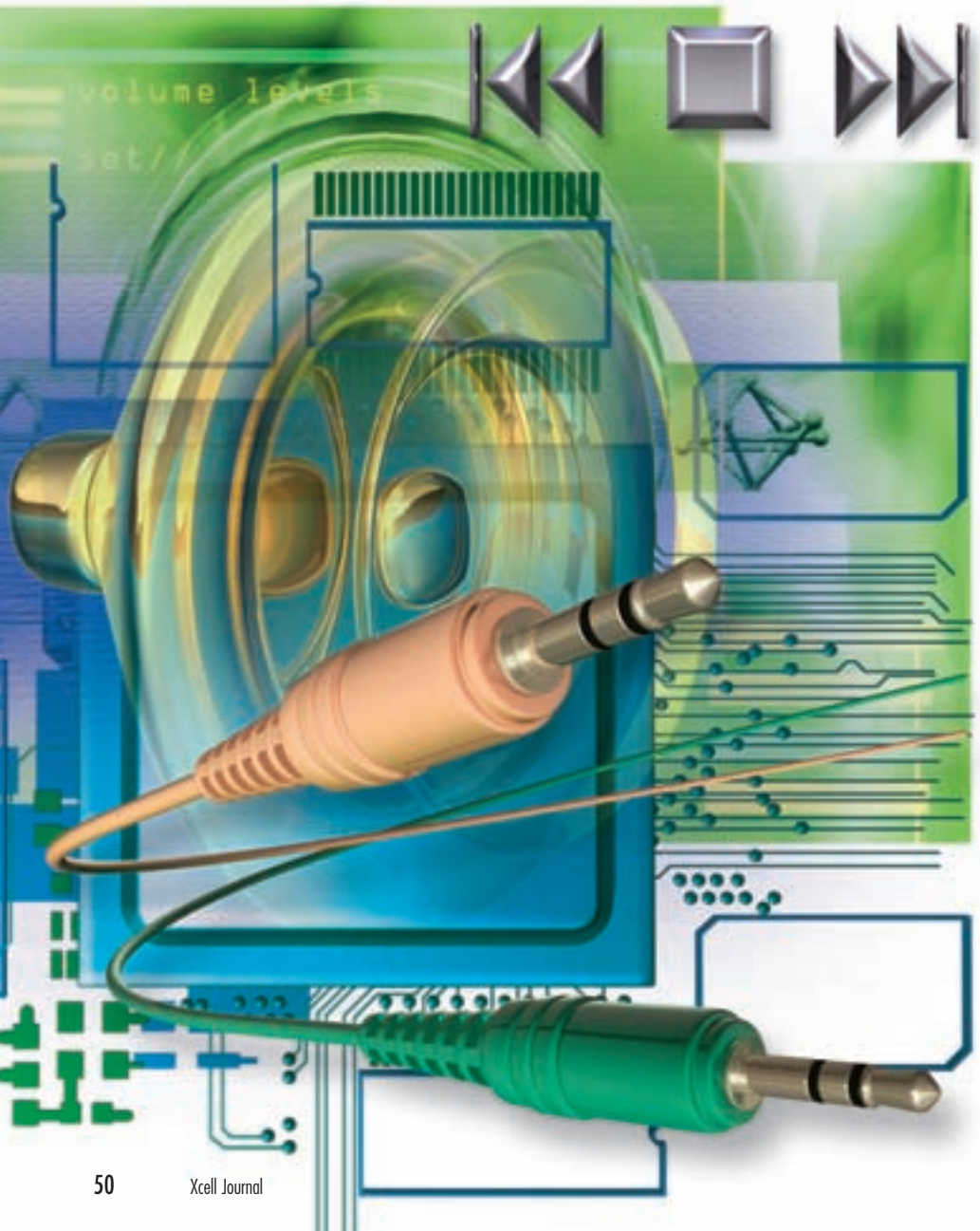
You can configure generic library components in The MathWorks Simulink environment for any real-time application without altering other programming-level aspects. These blocks lead to a single design solution rather than having to use multiple blocks for dedicated functions.

Most of the design modules use IP core elements during implementation to reduce propagation delays. These customized cores deliver high levels of performance and area efficiency, resulting in the efficient implementation of logic by using fewer FPGA resources in less time.

We wish to acknowledge Soumitra Das and Sandeep Kulkarni from Avnet Pvt. Ltd. for their support in writing this article.

Audio Sample Rate Conversion in FPGAs

An efficient implementation of audio algorithms in programmable logic.



by Philipp Jacobsohn
Field Applications Engineer
Synplicity Deutschland GmbH
philipp@synplicity.com

Derek Palmer
DSP Marketing Manager
Xilinx, Inc.
derek.palmer@xilinx.com

Today, even low-cost FPGAs provide far more computing power than DSPs. Current FPGAs have dedicated multipliers and even DSP multiply/accumulate (MAC) blocks that enable signals to be processed with clock speeds in excess of 550 MHz.

Until now, however, these capabilities were rarely needed in audio signal processing. A serial implementation of an audio algorithm working in the kilohertz range uses exactly the same resources required for processing signals in the three-digit megahertz range.

Consequently, programmable logic components such as PLDs or FPGAs are rarely used for processing low-frequency signals. After all, the parallel processing of mathematical operations in hardware is of no benefit when compared to an implementation based on classical DSPs; the sampling rates are so low that most serial DSP implementations are more than adequate. In fact, audio applications are characterized by such a high number of multiplications that they previously could

only be implemented using very large FPGAs. So audio applications with low sampling frequencies were implemented more efficiently using a DSP than a large FPGA – at a lower cost and with proven software support.

More recently, Synplify DSP, a synthesis tool from Synplicity, allows you to efficiently map even algorithms with large numbers of multiplications and a low sampling rate onto specialized DSP blocks in FPGAs. The tool is based on the popular MATLAB and Simulink tools from The MathWorks.

Algorithms are defined using a special block set or description in the proprietary "M" scripting language and are later translated into an RTL hardware description language. The block set allows both single-rate and multi-rate implementations. It not only generates VHDL and Verilog code but also handles tasks such as fixed-point quantization, pipelining, loop unrolling, and connects to block sets from the Simulink development environment for simulation (see Figure 1).

Application Example: Sampling Rate Conversion

Let's use a sampling rate converter for audio frequencies as a practical example. This converter can convert a signal from one sampling rate to another with minimal impact on the signal. Such converters are required to process signals with differing sampling rates.

For example, compact discs are sampled at 44.1 kHz, while digital audio tape is usually sampled at 48 kHz. But with data format conversion, playing the source data with the new sampling rate is not sufficient. Playing compact disc material at the sampling rates used for digital audio tape would cause distortions. Thus, the sampling rate must be converted.

When processing audio signals, many sampling frequencies are used: 44.1 kHz, 48 kHz, 96 kHz, and 192 kHz are common. During conversion, you must take care to maintain signal integrity in the audible range between 0 and 20 kHz. Changing the information contained in the signal should be kept to a minimum to limit degradation in audio quality (Figure 2).

Not surprisingly, the implementation of a sampling rate converter for audio frequencies raises two issues in the FPGA:

1. The algorithm issue:
   a. Highest possible signal-to-noise ratio
   b. Minimum possible change in the information carried by the original signal
   c. Efficient description of the algorithm, as the resources consumed in the FPGA are highly dependent on the quality of the description
   d. Quantization
2. The implementation issue:
   a. Logically correct implementation of the algorithm
   b. FPGA resource constraints
   c. Speed-optimized implementations
   d. Latency

The conversion requires a high clock speed because the implementation depends on adequate oversampling of the signal being converted. The difference between the FPGA system clock frequency and the signal frequencies being converted must be correspondingly high.

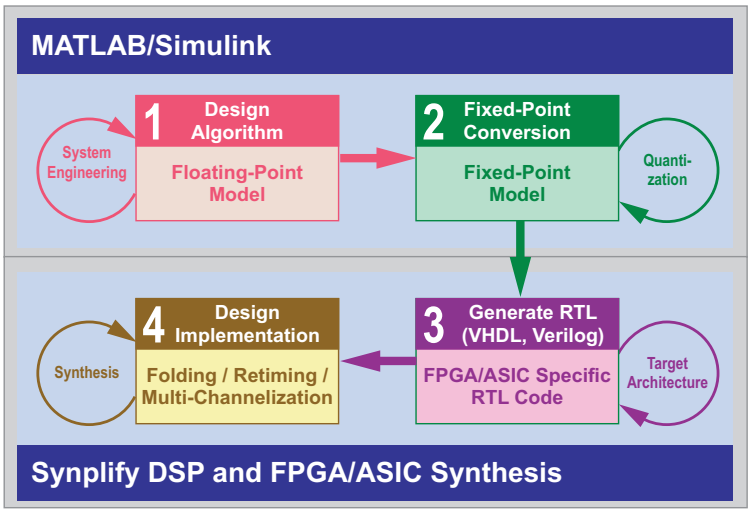


Figure 1 – The model is implemented, quantified, and verified in MATLAB/Simulink. The Synplify DSP tool converts the model into RTL code. The code can be optimized for space or speed.

Figure 2 – Modules from the Synplify DSP block set and Simulink FDA tool implement the sampling rate converter. Simulink block set elements perform verification.

The FDA tool helps generate and verify FIR and IIR filters of any kind. It is part of Simulink's signal processing toolbox, which Synplify DSP uses to implement filter structures.

For CD-quality audio signals, the signal-to-noise ratio must also be at least 100 dB. Professional applications even require audio signals of >120 dB. Other low-frequency signals (such as control electronics algorithms) are far less demanding than audio signals when it comes to signal quality.

The Algorithm

A polyphase FIR filter structure converts the sample rates (asynchronous resampling). The algorithm comprises two steps. In the first step, frequencies are oversampled. The second step – linear interpolation – is required to generate a different frequency from a given frequency. The two frequencies are asynchronous to each other.

Resampling the signal in a single step would require far more resources because the filter would be far more complex. This type of implementation would result in several million multiplications. Such a description is inefficient and should be avoided. If a linear interpolation is implemented for the second step, the resulting structures are far simpler (Figure 3).

Efficiently described oversampling (the first step) is the only way to achieve a resource-saving FPGA implementation. The number of computations required will drop dramatically if this part of the circuit is implemented in several cascaded stages rather than in a single computing step.

When implementing the algorithm, you must decide on the target architecture that will perform the computation (DSP or FPGA). Unlike digital signal processors that have a fixed architecture, FPGAs can implement any architecture. They are ultimately limited, however, by the size of the device when implementing large numbers of individual multiplications.

The number of multipliers required increases with the tap of the filter. Each tap results in the use of a DSP block or multiplier. When cascading resampling stages, each filter must perform functions that are far less complex. In theory, an optimal filter implementation would result from as many individual stages as possible.

The mathematical deduction of how to reduce computing operations has been described extensively in technical literature. Practical results show that while it is necessary to cascade filter stages, the number of cascades must be limited. If you introduce too many cascaded stages, you could exceed the available resources to implement the design. If an FPGA is used as the target architecture, two stages has proved to be optimal.

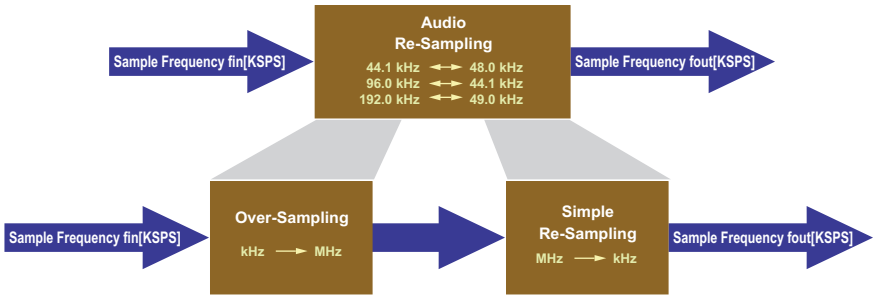


Figure 3 – The sample-rate converter is implemented in two steps (one, oversampling; two, linear interpolation) to improve efficiency.

The entire circuit comprises two relatively simple filters for oversampling and a simple linear interpolator. This structure can be efficiently mapped onto an FPGA.

The Implementation

You can implement the circuit in Simulink using the Synplify DSP block set and Simulink's filter design and analysis (FDA) tool. The FDA tool helps generate and verify FIR and IIR filters of any kind. It is part of Simulink's signal processing toolbox, which Synplify DSP uses to implement filter structures.

All circuit components from the Synplify DSP block set or the FDA tool, which are defined between a PortIN and a PortOUT description, generate VHDL or Verilog code. FFT and SCOPE elements from Simulink block sets conduct spectrum analysis and verification of the dynamic response. These blocks are exclusively used for functional verification, including floating-point to fixed-point conversion effects (quantization). The blocks are not implemented in hardware.

The first part of the algorithm implementation comprises two FIR filters: the first has 512 taps and the second has 64 taps. The RTL code resulting from oversampling therefore contains a total of 576 multiplications, which is why using an FPGA does not appear to be commercially viable. Such a large FPGA would be cost-prohibitive, requiring the largest Xilinx® Virtex™-5 XC5VSX95T device with its 640 DSP48 blocks.

All multiplications that are not mapped onto dedicated hardware structures (DSP blocks) must be built from generic logic resources (LUTs or registers). This results in higher resource requirements as well as a lower maximum clock speed. Dedicated DSP48 blocks are far more efficient multipliers than generic logic cells (Figure 4).

Optimization

Synplify DSP's folding option allows you to minimize the number of multipliers used. Those circuits operating at low sampling

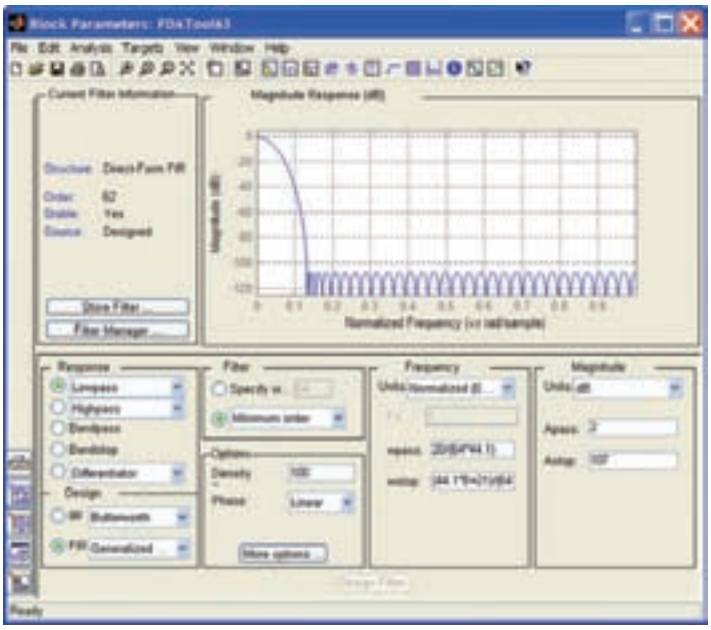


Figure 4 – Implementing filters using Simulink's filter design and analysis (FDA) tool

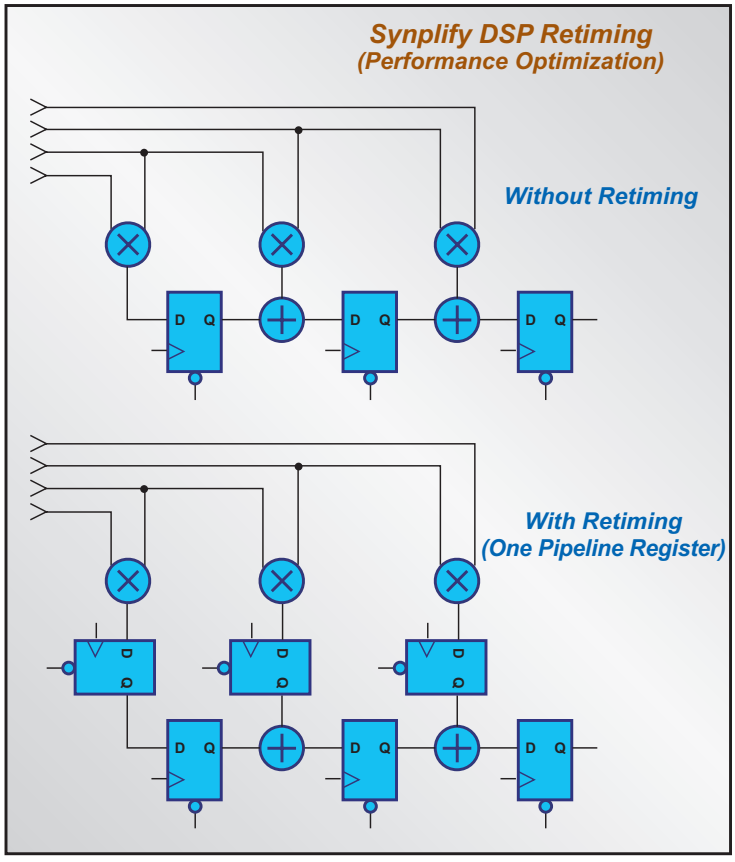


Figure 5 – You can dramatically reduce the FPGA resources required by using the folding feature.

frequencies can particularly benefit from this optimization.

The idea is simple. Normally, one hardware multiplier is used for each multiplication, even when the sampling frequency is in the kilohertz range. However, FPGAs can operate with clock speeds in the triple-digit megahertz range. If the hardware multiplier operates at the system frequency of the FPGA, multiplications can be processed sequentially using a time multiplexing process.

Let's say that the sampling frequency of the circuit is 3 MHz and the FPGA can run at a maximum of 120 MHz. Each hardware multiplier can perform 40 computing operations if the multipliers are run at the system frequency. The necessary hardware is therefore reduced by a factor of 40. This means that a sampling rate converter as described above (or any other circuit using low sampling frequencies) can be "folded" to the point where only very few hardware multipliers are required. Therefore, this converter can also be implemented in the smallest available low-cost FPGA and thus is a real alternative to DSPs.

Of course, it is also possible to offload particularly computationally intensive algorithms from a DSP to an FPGA, thereby reducing processor load. This is particularly useful if your DSP application has exceeded the performance capability and if you have a significant investment in application source code targeted at a specific DSP architecture (Figure 5).

Because the folding feature in Synplify DSP also supports multi-rate systems, you can reduce the number of multipliers required even more than in a system with a single sampling frequency. Oversampling is performed using two FIR filters. These filters run at different sampling frequencies. The filter running at a higher sampling frequency is folded using a folding factor that you specify.

The filter with the lower sampling frequency is folded using a correspondingly higher factor. This factor is obtained by multiplying the difference between the sampling frequencies of the two filters by the folding factor. For example, if the sampling frequency of one filter is 8 times

higher than the sampling frequency of the other filter, the faster filter is folded by a factor of 8 and the slower filter is folded by a factor of 64.

In this way, it is even possible to produce space-optimized circuits running at very high sampling rates that normally cannot be folded. For example, if a system runs at a sampling rate of 200 MHz with a folding factor of 2, the system frequency increases to 400 MHz.

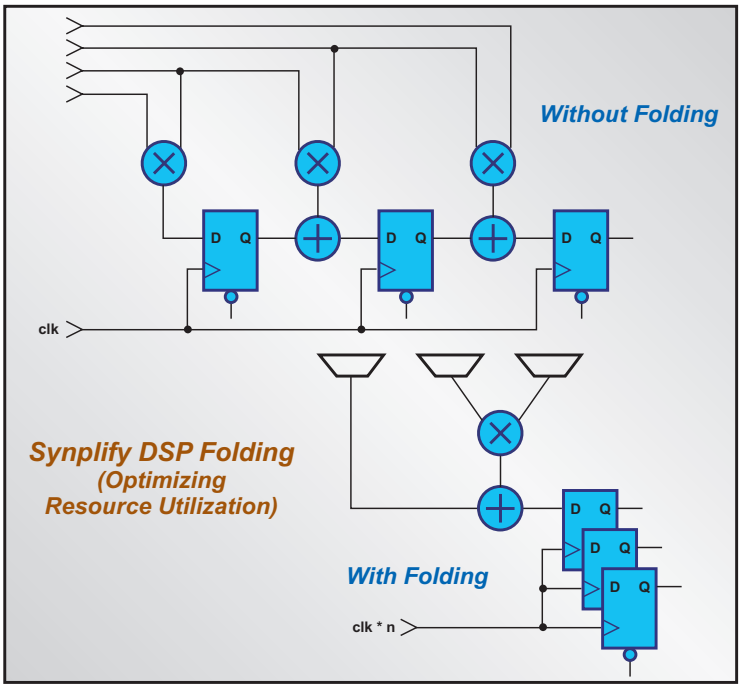


Figure 6 – Using the retiming feature, you can define the maximum latency allowed for the circuit. Synplify DSP then automatically adds pipeline stages until achieving the desired frequency.

Alternatively, you can define a folding factor of 1. Those circuit components running at the highest sampling rate are not folded. However, all circuit components of a multi-rate system running at slower sampling frequencies benefit from folding and space-optimized implementation. You only need to define the folding factor for the system as a whole. Folding is then propagated automatically across all sampling frequencies.

The folding feature can be combined with additional optimization functionality – the retiming feature. If a system does not meet the target frequency requirements, you can add pipeline stages until you achieve the desired rate. This is particularly important for circuits with high folding factors, which need to operate at a correspondingly high system speed.

You can also use retiming for circuits with little or no folding except where the performance limit of the FPGA is reached. Adding pipeline stages allows the number of combinatorial gates between two registers (logic levels) to be reduced, which increases the system clock speed.

When generating the RTL code, the Synplify DSP tool performs a timing analysis that takes the desired sampling frequency, the folding factor, and the target architecture of the FPGA into account. A circuit mapped to a fast Virtex-5 FPGA, for example, can be optimized using fewer pipeline stages than an identical circuit implemented in a slower, low-cost Spartan-3A DSP FPGA.

FPGAs provide large numbers of registers that you can use for this optimization. Unlike multipliers or LUTs (look-up tables), which can be used up rapidly, registers are available in abundance, which means that the system clock speed can be increased significantly with little effort using registers.

Of course, adding pipeline stages increases system latency. By introducing a retiming factor of 8, for example, the result of the computation will appear eight system clock cycles (not sampling frequency cycles) later at the FPGA's output. You must take this into account when embedding a circuit in a system (Figure 6).

It is particularly important to ensure that the optimizations described previously do not impact the original MATLAB model described in Simulink. Verification allows the algorithm to be validated and the impact of quantization effects to be represented. The Synplify DSP software block set allows floating- to fixed-point conversion using either truncation (elimination of irrelevant bits), rounding (in case of underflow), or saturation (in case of overflow). As soon as the simulation shows that the algorithm works as intended, the RTL code can be generated. Optimizing the VHDL or Verilog code may change latency, but not the operation of the circuit.

Conclusion

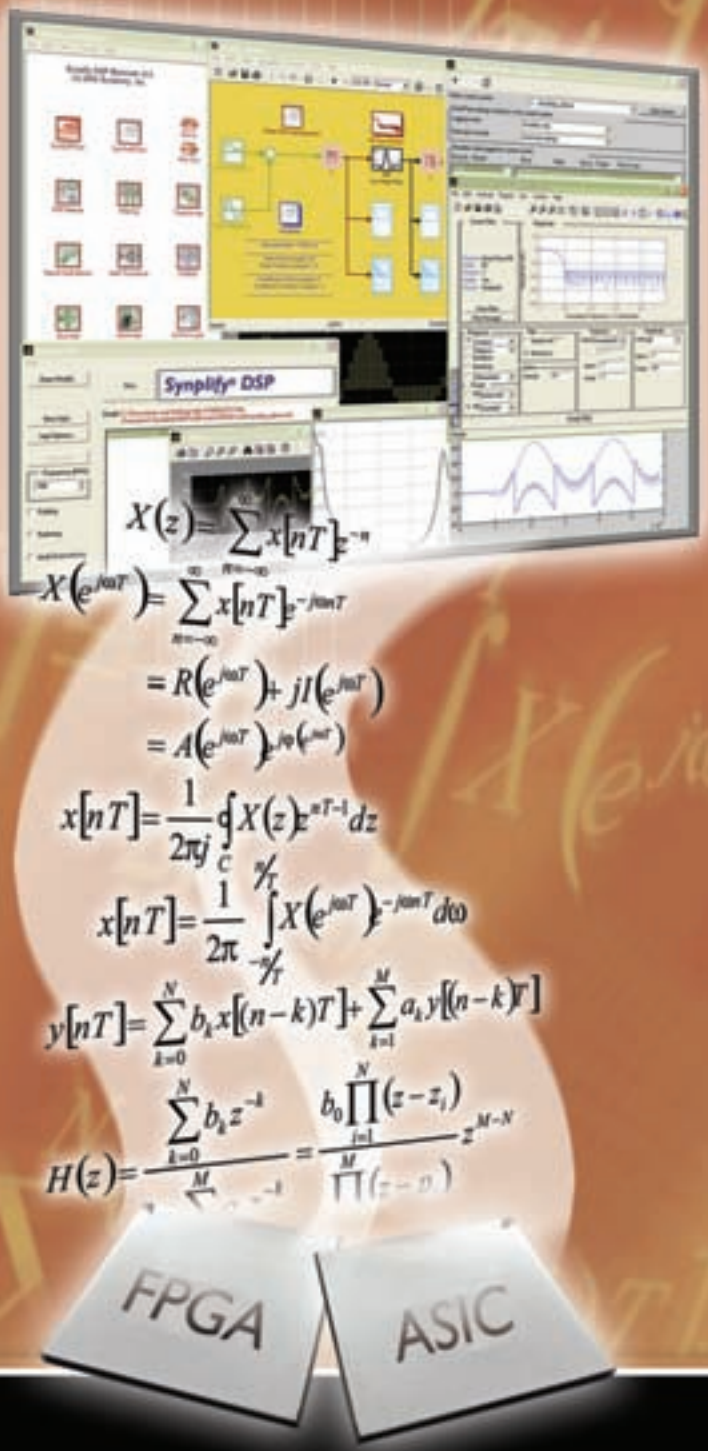
The Synplify DSP tool is based on the industry-standard MATLAB/Simulink software from The MathWorks. A block set provides a library of standard components that you can use to implement complex algorithms. Apart from basic components such as add, gain, and delay, the library contains many complex functions such as FIR or IIR filters and CORDIC algorithms. All features, including the highly complex FFT or Viterbi decoder, can be parameterized as you like. It is also possible to create user-defined libraries or integrate existing VHDL or Verilog code into a Simulink model.

Synplify DSP allows the implementation of both single- and multi-rate systems. Using folding, multi-channelization, or retiming, you can optimize the code for either size or speed. The RTL code generated is always generic, non-encrypted code that is synthesizable using popular tools.

For best results with FPGAs, Synplicity recommends its synthesis product, Synplify Pro. An ASIC variant of the development environment is also available now.

Synplify® DSP

ESL Synthesis Solution



Synplify DSP is an ESL synthesis solution that offers a fast, efficient way to implement DSP algorithms in FPGAs or ASICs. By automating architectural optimizations like pipelining, resource sharing, and multi-channelization, engineers can save months of RTL coding, simplify design capture, speed up verification, and create technology-independent IP.

Synplify DSP Software Uniquely Offers:

- Technology-independent DSP modeling library
- Comprehensive multi-rate and vector math support
- Fixed-point quantization and analysis tools
- Powerful DSP synthesis engine
- Architecturally optimized Verilog and VHDL implementation
- Target the latest ASIC technologies and FPGA devices
- Integrated support for standard ASIC design flows
- Memory extraction for flexible support of 3rd party ASIC memory vendors

For more information on Synplicity's Synplify DSP solution and all of Synplicity's offerings, please visit our website at www.synplicity.com or contact info@synplicity.com

Synplicity®

Simply Better Results


NO TOYS
Real Tools for ASIC Prototyping
Prove your design with high speed FPGA hard-ware logic emulation, that plugs directly into your PC. This 12+ million gate, 8-lane PCIe board, is ready to go: fixed PCIe core, extensive on board memory and eight independent clock networks will get your design running — *fast!* The DN9000K10PCIe-8T features six Virtex-5, LX330 chips with 100% of FPGA resources user-available.
There are no unreliable cables or stacks of overheating boards. The price, with the biggest and baddest FPGAs, is less than $70,000. Why pay more for a "some assembly required" toy? Visit **www.dinigroup.com** to see our complete selection of real high-speed prototyping solutions.
The DiNI Group
1010 Pearl Street, Suite 6 • La Jolla, CA 92037 • (858) 454-3419 • e-mail: sales@dinigroup.com

Featured DSP Application Notes

Jumpstart your DSP design with Xilinx application notes describing specific design examples and methodologies.

Application notes illustrate how to use a Xilinx® product in a specialized way. Xilinx maintains a large database of application notes on numerous topics, including design files, to assist you in your next design. The main Application Notes search page, *www.xilinx.com/apps*, includes different sections on specific market applications.

For application notes on digital communication, multimedia video and imaging, and defense systems, see our "XtremeDSP™ Selection Guide," *www.xilinx.com/publications/prod_mktg/pn0010944-3.pdf*.

Part Number	Title	URL
Design Methodology Application Notes		
XAPP547	PowerPC Processor with Floating Point Unit for Virtex™-4 FX Devices	*www.xilinx.com/bvdocs/appnotes/xapp547.pdf*
Error Correction Application Notes		
XAPP222	Designing Convolutional Interleavers with Virtex Devices	*www.xilinx.com/bvdocs/appnotes/xapp222.pdf*
XAPP383	Single Error Correction and Double Error Detection (SECDED) with CoolRunner™-II CPLDs	*www.xilinx.com/bvdocs/appnotes/xapp383.pdf*
XAPP551	Viterbi Decoder Block Decoding – Trellis Termination and Tail Biting	*www.xilinx.com/bvdocs/appnotes/xapp551.pdf*
XAPP715	Multiple Bit Error Correction	*www.xilinx.com/bvdocs/appnotes/xapp715.pdf*
XAPP948	Hardware Acceleration of 3GPP Turbo Encoder/Decoder BER Measurement Using System Generator	*www.xilinx.com/bvdocs/appnotes/xapp948.pdf*
DSP Processor Application Notes		
XAPP634	Analog Devices TigerSHARC Link	*www.xilinx.com/bvdocs/appnotes/xapp634.pdf*
XAPP635	Interfacing Virtex-II FPGAs with Analog Devices TigerSHARC TS20x DSPs via LVDS Link Ports	*www.xilinx.com/bvdocs/appnotes/xapp635.pdf*
XAPP706	Alpha Blending Two Data Streams Using a DSP48 DDR Technique	*www.xilinx.com/bvdocs/appnotes/xapp706.pdf*
XAPP753	Interfacing Xilinx FPGAs to TI DSP Platforms Using the EMIF	*www.xilinx.com/bvdocs/appnotes/xapp753.pdf*
XAPP919	Video Virtual Socket Architecture	*www.xilinx.com/bvdocs/appnotes/xapp919.pdf*
Transform Application Notes		
XAPP610	Video Compression Using DCT	*www.xilinx.com/bvdocs/appnotes/xapp610.pdf*
XAPP611	Video Compression Using IDCT	*www.xilinx.com/bvdocs/appnotes/xapp611.pdf*
Filter Application Notes		
XAPP212	CDMA Matched Filter Implementation in Virtex Devices	*www.xilinx.com/bvdocs/appnotes/xapp212.pdf*
XAPP219	Transposed Form FIR Filters	*www.xilinx.com/bvdocs/appnotes/xapp219.pdf*

DSP Development Boards and Kits

Xilinx DSP development boards and starter kits provide the fastest path to implementing your systems or algorithms onto FPGAs.

Xilinx, together with distributors and third party partners, offers a complete line of development and expansion boards to help you test and develop designs using Xilinx® devices over a wide range of applications. The following boards and kits are for DSP applications.

Virtex-5 SXT ML506 Evaluation Platform

The ML506 is a feature-rich DSP general-purpose evaluation and development platform (Figure 1). Though economically priced, the ML506 offers you the ability to create DSP-based and high-speed serial designs utilizing Virtex™-5 DSP48E slices and RocketIO™ GTP transceivers. A variety of on-board memories and industry-standard connectivity interfaces add to the ML506's ability to serve as a versatile development platform for embedded applications.

Part Number:
HW-V5-ML506-UNI-G

Price: $1,195

Purchase: Avnet, NuHorizons, local distributor

XtremeDSP Development Platform – Spartan-3A DSP 3400A Edition

Powered by the Spartan™-3A DSP 3400A device and supported by industry-standard peripherals, connectors, and interfaces, the Spartan-3A DSP Development Platform provides a rich feature set that spans a wide range of applications (Figure 2). Designed for use with Xilinx System Generator for DSP, the XtremeDSP™ Development Platform provides a great entry-level environment for developing signal processing designs.

Part number:
HW-SD3400A-DSP-DB-UNI-G

Price: $995

Purchase: Avnet, NuHorizons, local distributor

Figure 1 – Virtex-5 SXT ML506 Evaluation Platform

Figure 2 – XtremeDSP Development Platform

XtremeDSP Starter Platform – Spartan-3A DSP 1800A Edition

The low-cost Spartan-3A DSP Starter Platform (Figure 3) is the ideal way to evaluate your DSP applications on the DSP-optimized Spartan-3A DSP 1800A device. This versatile platform is supported by industry-standard peripherals, connectors, and interfaces. Designed for use with Xilinx System Generator for DSP and ISE™ software development tools, the XtremeDSP Starter Platform provides a great low-cost, entry-level environment for developing signal processing designs.

Figure 3 – XtremeDSP Starter Platform

The Spartan-3A DSP 1800A XtremeDSP Starter Platform gives you instant access to the capabilities of the Spartan-3A DSP family to target signal processing applications in the wireless, automotive, consumer, industrial, medical, defense/aerospace, server, storage, and telecom/datacom markets.

Part Number:
HW-SD1800A-DSP-SB-UNI-G

Price: $295

Purchase: Avnet, NuHorizons, local distributor

The DSP Design Curriculum Path

Xilinx Education Services is ready to help you use DSP solutions in your next design.

By Jannis McReynolds
Sr. Manager, Services Marketing
Xilinx, Inc.
jannis.mcreynolds@xilinx.com

Xilinx® Education Services (XES) programs provide targeted, high-quality education products and services that are designed by experts in programmable logic design and delivered by Xilinx-qualified trainers. XES offers targeted courses at all levels of programmable logic design and creates an engaging learning environment by blending lecture, hands-on labs, interactive discussions, tips, and best practices. By leveraging our global network of Authorized Training Providers (ATP) and online learning systems, we can offer courses in 26 languages at 110 locations around the world.

By becoming proficient with advanced programmable logic design techniques and methodologies, you will be able to take full advantage of all of the DSP capabilities available from today's leading FPGAs. The knowledge you gain will reduce your R&D costs through greater efficiency in the design process, and reduce production costs through the use of smaller devices in a slower speed grade. This expertise can greatly improve your organization's time to market, driving greater market success.

The Xilinx DSP Design Curriculum Path provides a recommended course sequence. You should meet the prerequisites of each class to gain the full benefit of each course. The curriculum path includes courses delivered by Xilinx and additional courses offered by The MathWorks. To view the DSP Design Curriculum Path in its entirety, visit *www.xilinx.com/support/training/cur_paths/atp-dsp.htm.*

The DSP Curriculum Path comprises these courses:

- DSP Design Using System Generator
- Introduction to AccelDSP™ Synthesis Tool
- The MathWorks Simulink for Xilinx
- DSP Implementation Techniques

DSP Design Using System Generator

This course allows you to explore the System Generator for DSP tool and gain the expertise required to develop advanced, low-cost DSP designs. This intermediate course focuses on using System Generator for DSP, design implementation tools, and hardware-in-the-loop verification. Through hands-on exercises, you will implement a design from algorithm concept to hardware verification with Xilinx FPGAs.

Introduction to AccelDSP Synthesis Tool

This course will teach you how to synthesize an algorithm written in the language of MATLAB software into a design that is optimized for a Xilinx FPGA. You'll learn how to make coding changes in MATLAB to improve area and performance, and how to use the floating- to fixed-point and design exploration features of the AccelDSP synthesis tool to achieve maximum results.

The MathWorks Simulink for Xilinx

This course covers the basics of using Simulink, an interactive, graphical environment for modeling and simulating dynamic systems. This course covers all aspects of system modeling with Simulink, including creating a model, simulating the system, and analyzing the results. Advanced simulation concepts and simulations from the command line are also explained. The final section discusses refining models by providing additional functionality with S-functions, block masks, and GUIs for interaction with your system.

DSP Implementation Techniques

This course will show you how to take advantage of Xilinx FPGA architectures to effectively implement DSP algorithms. The techniques also demonstrate which system-level decisions have the greatest impact on the implementation process and product costs.

TAKE THE NEXT STEP (Digital Edition: www.xcellpublications.com/subscribe/)

- Register today for any of these courses.
- View the full DSPDesign Curriculum Path.
- Contact your Xilinx sales representative.

Intellectual Property Offerings

Xilinx DSP Intellectual Property Offerings

The Xilinx XtremeDSP Initiative helps you develop tailored, high-performance DSP solutions.

Xilinx® DSP intellectual property (IP) offerings implement simple algorithms (filters and transforms, for example) with varying implementations that allow you to trade-off algorithmic performance for area and speed. Thus, you can achieve faster time to market by focusing on the algorithms and implementations that differentiate your designs from the competition. Xilinx DSP IP includes both free and for-purchase cores.

Table 1 lists the broad categories of Xilinx DSP IP cores.

Xilinx DSP IP	
Filters	Error Correction
Transforms	Basic Math
Waveform Synthesis	Floating Point
Linear Algebra	Trigonometric Functions

Table 1 – Categories of Xilinx DSP IP cores

What's New

Fast Fourier Transform (FFT)

The FFT v5.0 provides four different architectures, along with system-level fixed-point C-models, and reduces typical implementation times from three to six months to the push of a button.

FFT v5.0 expands the focus on orthogonal frequency-division multiplexing (OFDM) systems, providing reduced area for OFDM multi-channel systems like WiMAX and 3GPP-LTE through increased coverage of the multi-channel architecture and cyclic-prefix insertion.

A typical FFT used in a 2 x 2, three-sector modem has 42% and 54% fewer logic resources in memory usage when comparing version 5.0 to version 4.1. This LogiCORE™ IP was released in ISE™ software 9.2i, CORE Generator software IP update 2.

www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=FFT

FIR Compiler

The FIR filter is one of the most ubiquitous and fundamental building blocks in DSP systems. Version 3.2 of the FIR Compiler expands device support to the lowest cost Spartan™-3A/3E device families. This LogiCORE IP was released in ISE software 9.2i, CORE Generator software IP update 2.

www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=FIR_Compiler

Cascaded Integrator Comb (CIC) Compiler

The CIC Compiler version 1.0 reduces filter implementation time to the push of a button, while also providing you with the ability to make trade-offs between differing hardware implementations of your CIC filter specification.

Along with greater than 50% area savings versus older CIC filter IP offerings from Xilinx, the version 1.0 of the CIC Compiler also comes close to the 450 MHz (-1) and 250 MHz (-4) achievable in Virtex™-5 and Spartan-3A DSP devices. This high-performance capability enables support for the highest performance ADC and DAC technology available, or alternatively supports more channels in a single structure to save area. This LogiCORE IP was released in ISE software 9.2i, CORE Generator software IP update 2.

www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=CIC

Xilinx Wireless Intellectual Property Offerings

Xilinx offers high-performance, cost-effective solutions for wireless networking equipment.

Xilinx wireless intellectual property (IP) offerings are suitable for RF digital front-end (DFE) signal processing, baseband processing (including forward error correction [FEC], FFTs, and adaptive modulation), and advanced interfacing, connectivity, and bridging solutions.

Xilinx Radio Reference Designs

Xilinx has developed a number of reference designs for customers supporting various air interface standards. These designs support Virtex™-4, Virtex-5, and Spartan™ DSP devices and demonstrate how to design efficiently for FPGA architectures leveraging high clock rates and specific architectural features.

Our WCDMA, WiMAX, and TD-SCDMA designs include digital up conversion (DUC), digital down conversion (DDC), automatic gain control (AGC), and crest factor reduction (CFR).

These designs are developed in System Generator, allowing you to test and verify your designs using Xilinx-supplied MATLAB scripts. They are also easily modifiable.

Using these techniques, you can quickly realize efficient digital radio implementations, resulting in lower equipment costs. For more information about the reference designs shown in Table 1, visit *www.xilinx.com/esp/wireless.htm#rf*.

Xilinx Baseband IP

Demonstrating our commitment to delivering optimized system-level solutions for commercial wireless applications, ISE™ software v9.2, IP update 1, includes three new system-level LogiCORE™ IP products for 3GPP release 6.

3GPP Downlink Chip Rate

The 3GPP Downlink Chip Rate LogiCORE solution provides a release 6-compliant, Xilinx FPGA-optimized solution for femtocell, picocell, and macrocell solutions. The architecture has been designed to provide efficient use of FPGA logic, while offering a low-bandwidth interface to an external DSP or microprocessor and reducing system-level overhead using a built-in OCP interface.

The FPGA performs timing-critical operations, which simplifies the software impact with traditional DSP solutions, allowing for an optimum software/hardware balance. The core is fully optimized for speed and area while supporting all frequency division duplex (FDD) channels. This LogiCORE IP was released in ISE software 9.2i, CORE Generator software IP update 1.

www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=DO-DI-DLCR-3GPP

Topic	Resource	Type	Provider
WCDMA/HSPA	WCDMA/HSPA Reference Design (Login Required) Device Architecture: Virtex-4, Virtex-5 FPGAs DUC: Three Carrier DDC: Six Carrier (Receiver Diversity) CFR: Three Carrier (PAPR<6 dB@<11% EVM, >60 dB ACLR)	Reference Design	Xilinx
WiMAX	WiMAX Reference Design (Login Required) Device Architecture: Virtex-4, Virtex-5 FPGAs DUC: One carrier (dynamic switching 3.5, 5, 7, and 10 MHz) DDC: One carrier (dynamic switching 3.5, 5, 7, and 10 MHz) CFR: One carrier (PAPR>1.5 dB@2% EVM) WiMAX Reference Design (Login Required) Device Architecture: Spartan-3A DSP FPGA DUC: One carrier (Dynamic Switching 5 and 10 MHz) DDC: One carrier (Dynamic Switching 5 and 10 MHz)	Reference Design	Xilinx
TD-SCDMA	TD-SCDMA Reference Design (Login Required) Device Architecture: Virtex-4 FPGA DUC: Up to Six Carrier DDC: Up to Six Carrier	Reference Design	Xilinx
Common Digital Radio System (CDRSX)	Common Digital Radio System Development Platform Device Architecture: Virtex-II Pro, Virtex-4 FPGAs	Development Board	Axis Networking

Table 1 – Xilinx radio reference designs

3GPP RACH Preamble Detector

The 3GPP RACH preamble detector is used in WCDMA transmission systems. This LogiCORE IP allows you to scale the solution from femtocell up to macrocell architectures. The LogiCORE IP is also highly cost-effective, with minimal utilization by using streamed correlation calculations and supporting coherent and non-coherent detection methods.

Scalability is maximized through customizable inputs such as search window size, coherent accumulation window size, number of antenna, hardware over sample rate, and I/O quantization. It also offers ease of integration with a suitable DSP or microprocessor through Open Core Protocol (OCP) interfaces. This LogiCORE IP was released in ISE software 9.2i, CORE Generator software IP update 1.

www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=DO-DI-RACH-3GPP

3GPP Searcher

The 3GPP Searcher is used in WCDMA transmission systems to identify the multiple user transmission paths in a 3GPP uplink. This LogiCORE IP offers a compact and scalable solution when used for either picocell/femtocell or macrocell applications, resulting in the lowest cost for the given target application.

The core includes all of the logic required for scramble code generation, correlation, accumulation, and filtering functions in a single co-processing solution, easily integrated with either a DSP or microprocessor using available OCP interfaces. The CORE Generator software GUI allows you to fully customize to your own needs. This LogiCORE IP was released in ISE software 9.2i, CORE Generator software IP update 1.

www.xilinx.com/xlnx/xebiz/designResources/ip_product_details.jsp?key=DO-DI-SEARCHER-3GPP

Conclusion

To access the IP solutions highlighted in this article, visit *www.xilinx.com/ipcenter*.

Tech Tips

Xilinx technical support answers your DSP application questions.

by Jeffrey Harriman
DSP Tools Product Applications Engineer
Xilinx, Inc.
jeffrey.harriman@xilinx.com

These tips offer insight into popular questions from Xilinx® System Generator for DSP users trying to run hardware co-simulation for hardware verification and simulation acceleration.

I hope these tips will give you some guidance next time you're thinking about using System Generator for DSP for hardware co-simulation. Xilinx also provides additional examples in the System Generator for DSP documentation.

Q: How can I use hardware co-simulation with System Generator for DSP to accelerate my simulation?

A: Hardware co-simulation is an excellent tool for increasing the simulation speed of complex System Generator for DSP models. There are several options when setting up a hardware co-simulation that affect how data is transferred between the FPGA hardware and your computer. Depending on your design, one may be more appropriate than another.

In the most common type of hardware co-simulation, you will take your System Generator model "as is" and select your hardware co-simulation target as the compilation target for generation. The Gateway In and Gateway Out blocks become input and output ports on the generated hardware co-simulation block. For this type of setup, "single stepped" simulation mode is the best simulation mode. In this mode, the FPGA hardware is kept in sync with the Simulink simulation by System Generator, which drives the clock along with the data at each time step.

By adding the newly generated hardware co-simulation block to your model, you can now simulate a system-level Simulink model with FPGA hardware running the Xilinx System Generator for DSP portion of your design. You can leave the System Generator blocks from which the hardware co-simulation block was generated and run them side-by-side with hardware to verify that they match. Or you can remove all of the software simulation blocks to give your simulation a speed boost, offloading the System Generator portion of your design completely to hardware for simulation.

Q: Which development board and type of interface is the best for running hardware co-simulation?

A: One of the most important factors affecting simulation speed is the type of interface used to communicate with the board. The available interfaces are JTAG, PCI or PCMCIA, and Ethernet. JTAG is the most flexible option, as it can be used with any board with a JTAG connection to a Xilinx FPGA. However, JTAG typically has the lowest bandwidth. The System Generator board description builder is available to help generate the necessary board support files to add custom boards to the list of compilation targets in System Generator. In the GUI, you can specify board details such as clock rate, JTAG chain information, and board-specific I/O ports (Figure 1).

The Nallatech XtremeDSP™ kit boards have built-in support using PCI, while other companies use a PCMCIA interface for hardware co-simulation. Both of these options will offer increased simulation throughput over JTAG when running hardware co-simulation. These development platforms often include DSP-specific hardware such as ADCs and DACs.

Ethernet offers the fastest hardware co-simulation speeds. The XtremeDSP Spartan™-3A DSP edition supports Ethernet connections at 10 and 100 Mbps, while the ML506 and ML402 can support up to 1 Gbps connection rates. These boards also allow you to run a point-to-point or network-based Ethernet connection. Because network-based Ethernet co-simulation has to share traffic and deal with additional Ethernet protocols, the point-to-point setup has much higher performance.

Many factors can result in some performance overlap of the various interface options:

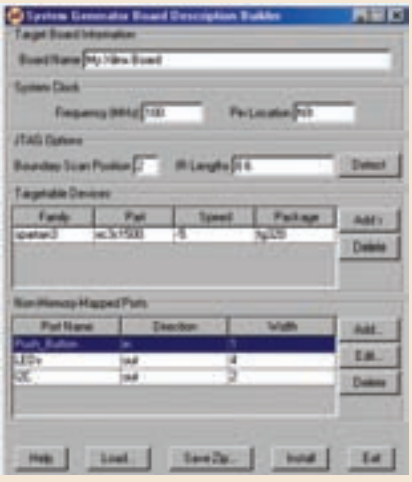


Figure 1 – The System Generator board description builder makes it easy to add hardware co-simulation support for any board with a Xilinx FPGA in the JTAG chain.

data width, the number of ports, and the number of simulation steps, for example. Note that there is some overhead associated with hardware co-simulation – device configuration and initializing the interface – so the more data samples collected from a simulation, the more advantages you have by using hardware co-simulation.

Q: How can free-running mode and shared memories be used with hardware co-simulation in System Generator?

A: The free-running simulation mode has two common applications. The first involves interaction with other hardware on your board. For instance, you may want to process a real-world signal that comes in from an ADC and send the processed signal back out to a DAC to view on an oscilloscope. In this case, you can use non-memory-mapped ports (such as those in the XtremeDSP kit block set) with a free-running hardware co-simulation to connect your design to the hardware-wired FPGA pins instead of communicating with the Simulink simulation.

Another scenario using free-running simulation involves the shared memory blocks in the Xilinx block set. The frame-based nature of the shared memory interface makes it a good fit for frame-based designs such as video processing, where you can leverage the shared memory blocks using large block transfers down to the FPGA for great increases in your hardware co-simulation bandwidth. To use shared memories in free-running mode, they need to be "lockable," which means using additional control signals to request and grant access to the shared memory space.

Because the design running on the FPGA uses the on-board clock in free-running mode, it is important to use data-valid signals throughout the design so that the hardware can be paused when there is no data to consume from the shared memory buffers. For this type of simulation, the Simulink test bench should buffer the data so that it matches the size of the shared memory buffer in hardware. To maintain efficiency on the hardware side, it is important to use memory buffers sized appropriately for your design.

The "Simulink System Period" set in the System Generator token determines how often the memories on the FPGA and CPU are synchronized. By setting this so that a vector is transferred only once each time the buffer is filled up, you will get a more efficient simulation and optimize your shared memories.

For example, in the design in Figure 2, with 4k deep shared memory, the "Simulink System Period" is set to 4,096 times the Simulink sources sample period (the sine wave generator on the left). In this way, after the appropriate number of samples are buffered up in software, a block transfer is performed down to the hardware memory instead of transferring a single sample per simulation step.

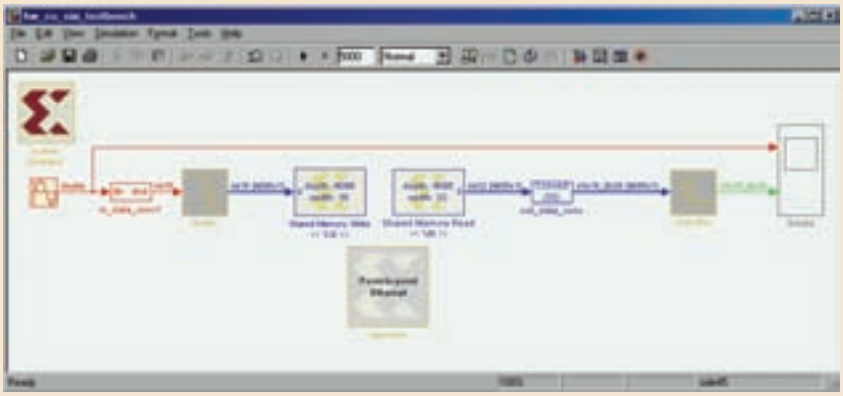


Figure 2 – Shared memories are used to transfer blocks of 4,096 samples between Simulink and the FPGA hardware to increase the hardware co-simulation bandwidth.

Logic analyzers up to 50% off

A special limited-time offer from Agilent.

Agilent portable and modular logic analyzers

- *Increased visibility with FPGA dynamic probe*
- *Customized protocol analysis with Agilent's exclusive packet viewer software*
- *Low-cost embedded PCI Express packet analysis*
- *Pricing starts at $9,450*

Now you can see inside your FPGA designs in a way that will save weeks of development time.

The FPGA dynamic probe, when combined with an Agilent Windows®-based logic analyzer, allows you to access different groups of signals inside your FPGA for debug—without requiring design changes. You'll increase visibility into internal FPGA activity by gaining access up to 128 internal signals with each debug pin.

Our 16800 Series logic analyzers offer you unprecedented price-performance in a portable family, with up to 204 channels, 32 M memory depth and a pattern generator available.

And now for a limited time, you can receive up to 50% off our newest 16901A modular logic analyzer mainframe when you purchase eligible measurement modules. Offer valid February 1, 2007 through August 15, 2007. Reference promotion 5.564.

www.agilent.com/find/logic-offer

Agilent Technologies

Virtual Worlds

Future systems will have to rely on programmability.

by Ivo Bolsens
CTO
Xilinx, Inc.
ivo.bolsens@xilinx.com

The wave of digitization is all around us. None of us has a crystal ball to predict the next killer application. But we will be surrounded by intelligent systems that monitor our health, protect our security, and increase our comfort. Thousands of sensors and actuators will generate and manipulate real-time digital signals to control our personal environment.

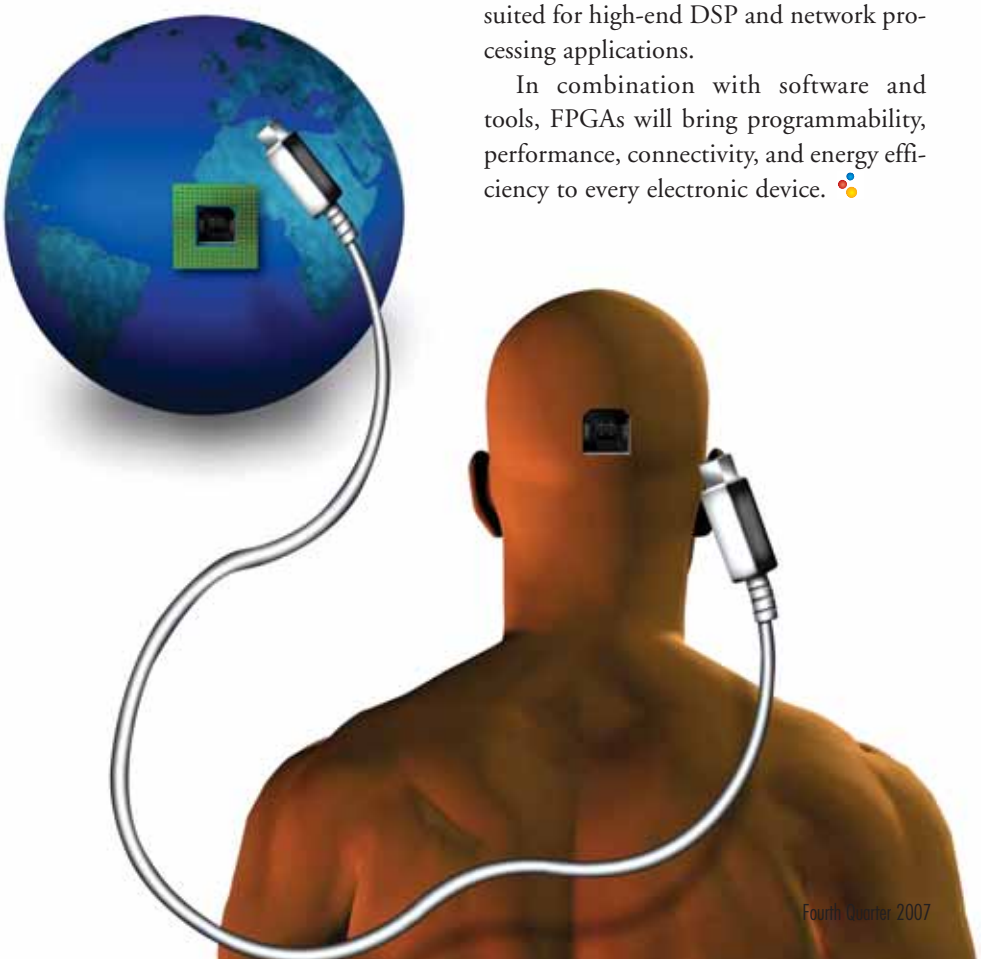
In less than 10 years, we will partly work and live in virtual worlds comprising an online alternate universe several times larger than today's Internet. These virtual worlds – like "Second Life" and "World of Warcraft" – require a massive amount of 3D real-time compute power.

The pace at which these trends are developing is astonishing. In September, for the first time in history, 9,000 IBM employees in Italy organized a strike in the virtual 3D world of Second Life. Driven by massive virtual reality data sets, complex 3D rendering models, and real-time animation, we will witness a data explosion and a need for a digital processing capacity that makes current data centers pale.

Mass Market of One

In the mass market of one, customers will expect standard products to be tailored to their specific user profile and requirements. For example, a programmable remote control will configure itself to deal with an individual's specific home network, comprising video and audio devices, security systems, and domestic appliances. FPGAs are ideal for such products: they are standard, they are programmable, and they offer performance that exceeds any other programmable platform.

Moreover, FPGAs have the advantage of getting increased performance by exploiting both the increase in clock rates as well as the increase of extra available silicon hardware as transistor sizes decrease. Processors are relying on increasing performance by boosting up clock frequency. However, at a time when the available number of transistors is outpacing the capability to use them efficiently, adding more hardware and using parallelism and pipelining to increase performance makes much more sense.



This parallelism allows FPGAs to react instantaneously on interrupts from the outside environment. It is clear that processor architectures are running out of steam with respect to further performance increases. FPGAs still have a very promising road in front of them.

The Case for FPGAs

The explosion of digital data to be processed creates a data-transfer bottleneck for traditional processor architectures. It is getting very hard to move data from one large global data or instruction memory to the processor data path. Increasingly complex caching schemes and pipelining try to deal with this issue. Today's modern processor data paths occupy less than 10% of the silicon die. The rest of the die is covered with caches and controllers to keep up with the data-transfer problem.

In contrast, FPGAs rely on a distributed memory architecture that supports the concept of keeping the data as close as possible to arithmetic and logic, while guaranteeing memory bandwidth orders of magnitude larger than traditional processors. This makes FPGAs especially well suited for high-end DSP and network processing applications.

In combination with software and tools, FPGAs will bring programmability, performance, connectivity, and energy efficiency to every electronic device.

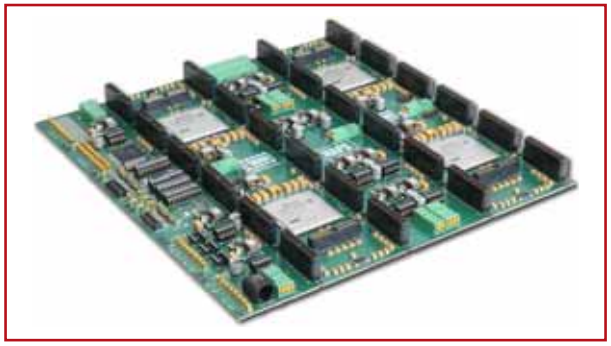
ACCELERATE VERIFICATION

With The Confirma™ ASIC/ASSP Verification Platform From Synplicity®

Reduce verification time from months to days

The Confirma platform is a tightly-integrated, easy to use, and comprehensive at-speed ASIC/ASSP verification solution that dramatically accelerates functional verification of ASICs, ASSPs, and SoC designs.

The three major components of the Confirma Platform are already considered best-in-class tools: the Certify® ASIC verification software implements an existing ASIC design in multiple FPGAs; the HAPS™ High-performance ASIC Prototyping System™ is the execution engine; and the Identify® Pro software, with its revolutionary TotalRecall™ technology, provides full visibility and a seamless interface into industry-standard software simulators for easy debugging. When used together, the tools provide the highest-performance ASIC and ASSP verification platform available today.



HAPS-54 Virtex-5 Board Offers Prototypers 8 Million ASIC Gates

Visit http://www.synplicity.com/products/prototyping_solutions.html to learn more about reducing verification time.

Synplicity®

Simply Better Results

Synplicity
600 West California Avenue
Sunnyvale, CA 94086
Phone: (408) 215-6000
Email: info@synplicity.com

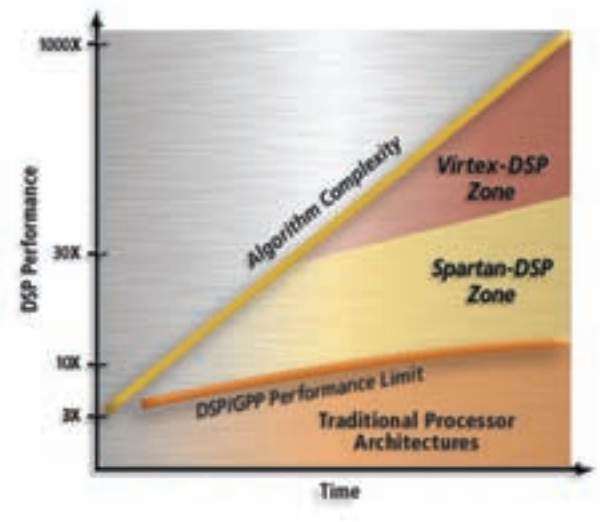
BREAK THROUGH

Now it's your turn to innovate...

XtremeDSP™

Leading the way in high-performance, configurable DSP

Outperform your competition with the high-performance signal processing delivered by XtremeDSP. With the new ***Spartan-DSP*** series, low-cost implementations are now possible without compromising performance. That opens up an exciting world of innovative applications in wireless, video, imaging and more.



Jump start your design with XtremeDSP solutions

- **XtremeDSP Device Portfolio:** *Configurable, scalable, high-performance devices: Spartan-DSP and Virtex-DSP series*
- **XtremeDSP Design Environment:** *Flexible, easy-to-use design flow with Xilinx Signal Processing Libraries, Xilinx System Generator™ for DSP, and AccelDSP™ high-level MATLAB® language-based tool*
- **XtremeDSP Development Tools:** *Application-specific reference designs, development boards, kits and IP blocks*
- **XtremeDSP Support:** *World-class support from ecosystem of partners and dedicated Xilinx teams*

Visit ***www.xilinx.com/dsp*** today, order your FREE evaluation software, and take your next DSP design to the Xtreme.

XILINX®

www.xilinx.com/dsp

Innovation to the Xtreme

PN 2047