Issue 38
Winter 2000

Xcell journal

THE AUTHORITATIVE JOURNAL FOR PROGRAMMABLE LOGIC USERS

PRODUCTS

New PROMS Simplify FPGA Designs

APPLICATIONS

Creating High-Performance Digital Down Converters

SOFTWARE

New Xilinx Foundation Series ISE Software with Integrated Design Flows

NEWS

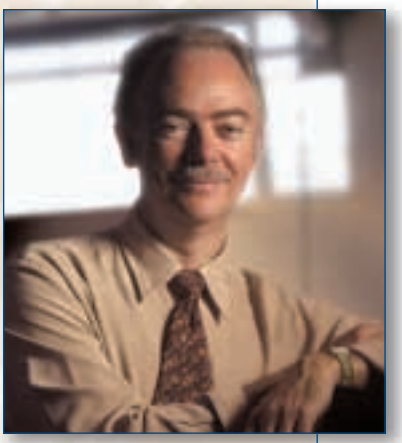
Xilinx Launches Platform-FPGA Initiative

Cover Story

Synopsys Chief Technology Officer Discusses Platform-Base FPGA Design Issues

XILINX®

Business is booming, and Xilinx is growing rapidly...



Xcell journal

Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124-3450
Phone: 408-559-7778
FAX: 408-879-4780

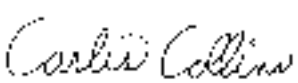
Programmable logic continues to grow faster than other segments of the semiconductor market, and Xilinx continues to grow along with it–there is no end in sight. To keep up with this unprecedented expansion, we are building several new facilities, acquiring new companies, and incorporating the best available complementary technologies.

We are leading the industry, not only with the most advanced device and software technologies but also with the most ambitious plans for future developments. Here are some of our most recent activities:

- Xilinx purchases two new buildings in San Jose. The buildings will provide approximately 200,000 additional square feet and are expected to house up to 700 new Xilinx employees. The purchase of the new buildings is the latest in a string of new construction projects we have undertaken in the last few years. In 1999, we completed construction of a fourth building at our San Jose headquarters, and other projects are also underway at Xilinx locations in Colorado, Ireland, and California.
- Xilinx acquires Visual Software Solutions, Inc. (VSS). Their expertise will help us further extend our software leadership and allow us to deliver a variety of customized tools that facilitate HDL-based design using our new Virtex-II FPGAs, thus improving your time-to-market. Included in the acquisition are the VSS HDL Bencher™ and StateCAD™ design tools.
- Xilinx acquires RocketChips, a leading developer of ultra-high-speed CMOS mixed-signal transceivers serving the networking, telecommunications, and enterprise storage markets. The RocketChips gigabit and multi-gigabit serial CMOS transceiver technologies provide solutions for a wide range of serial system architectures, and this technology will be a key feature of our next-generation FPGA families.
- Xilinx acquires Tornado, a full-function formal verification application deploying state-of-the-art circuit equivalence checking techniques. Based on many years of research and development efforts by Veriphia, this new software adds significant value to our advanced development tools. We plan to develop this technology even further and focus it on the Virtex FPGA architectures, in alliance with key EDA partners.
- Xilinx acquires Integral Design, a privately held design services firm headquartered in Dublin, Ireland. The acquisition enhances our professional design services capabilities in the communications and multimedia market segments. Recent advances in FPGA performance and capabilities continue to drive customer needs for additional design resources. Design services enable you to use dedicated designers with experience in Xilinx solutions to augment your own internal expertise and improve your time-to-market.

These developments continue to enhance our capability to offer you the best programmable logic devices, development tools, and services in the industry.

Our current capabilities already give you a significant ease-of-use and time-to-market advantage. As the market expands, costs decrease, and many new applications become possible, thus fueling even more growth. You can see why programmable logic is quickly becoming the technology of choice for many more applications, from low-cost consumer devices to high-performance switching systems-there simply is no faster or easier way to create the systems of the future. And, Xilinx is well prepared to continue leading the way.



Carlis Collins
Editor

Cover Story Page 8

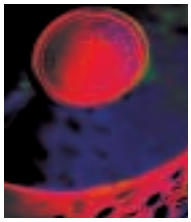
Designing With FPGA Platforms - page 8

The Chief Technology Officer at Synopsys discusses the need for platform-based design in an era of system-on-a-chip FPGAs.

Perspective Page 14

Choosing the ARC User-Configurable Processor - page 14

ARC Cores and Xilinx provide everything you need to develop custom processor applications.



New Products Page 22

New Xilinx Foundation Series ISE Software - page 22

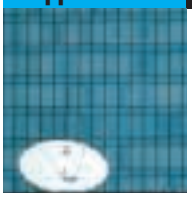
Integrated design flows increase your productivity and accelerate your time to market.

New Products Page 30

New High-Density and Cost-Effective PROMS - page 30

Xilinx announces the addition of the XC17V00 and XC17S00A families to its existing line of PROMS.

Applications Page 48



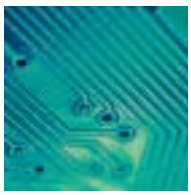
High-Performance Digital Down Converters for FPGAs - page 48

Virtex FPGAs surpass off-the-shelf ASSPs in design flexibility and system integration.

New Products Page 54

Stackable Development Boards - page 54

A new series of prototyping boards to help you quickly test and implement your FPGA designs.



Contents Fall 2000

Xcell journal

For a Free Subscription to the Xcell Journal E-mail your request to: literature@xilinx.com.

Please include:
1. Your full name and mailing address.
2. Your job title.
3. Your e-mail address.
4. Your company name.
5. Is this a new subscription or a renewal?

Platform FPGA- The Future of Logic Design

Xilinx and its partners are building the high-performance technology platform on which the designs of the future will emerge.

By: Wim Roelandts, CEO, Xilinx

The revolution in logic design continues, bringing dramatic performance improvements and new capabilities that help you create the systems of the future, and get them to market faster than ever before. FPGAs were once just interconnect routing and logic gates; then we added dedicated hard cores for memory, clock management, and I/O. Now, FPGAs are becoming the platform on which a combination of complex hard cores and flexible soft cores combine with an abundance of programmable logic gates to give you the best possible performance, along with the ease-of-use and time-to-market advantages for which FPGAs are well known. Plus, we can bring you these advantages at a lower cost than ever before.

Xilinx has entered a number of strategic partnerships and has acquired key technologies for creating the new programmable logic platform. Here is an overview of our recent activities.

The IBM Partnership

Our recent partnership with IBM brings us two immediate and dramatic benefits: the power of the PowerPC™ hard core, and the advanced CMOS manufacturing capability of IBM's state-of-the-art facilities. IBM gets intellectual property (IP) from Xilinx to help reduce defect densities and improve manufacturing productivity. This partnership has far-reaching implications, and gives both companies a significant competitive advantage.

The PowerPC Core

IBM's PowerPC module and CoreConnect™ bus will soon be integrated into Virtex-II FPGAs. With this powerful combination, you can achieve performance that was never possible before, and you can quickly develop unique system-level applications with greater ease. We found that the PowerPC was the most-used processor in high-end designs; our communication and computing customers use the PowerPC because it has good performance, and it has a lot of peripherals and other functions that make it easy to use.

Processors like the PowerPC are often used as logic engines for low speed, very complex logic; they allow you to write detailed programs that perform intricate condition checking and control functions. However, because a processor basically executes one instruction at a time, it's slower than actual gates which can operate in parallel.

Now, in one Platform FPGA, you will have the best of both worlds; you have the dedicated PowerPC processor for complex control applications, and you have programmable logic gates for very-high-speed data paths. The big advantage of having this all on one chip is that you can very quickly move data from the PowerPC processor to on-chip peripherals or custom logic, which may be hard cores, soft cores, or unique designs created with programmable gates. This will give you much higher performance than you get using separate chips, which must pass signals through their slower I/O interfaces.

We are implementing the PowerPC processor and other dedicated functions (such as memory, clock management, multipliers, and I/O interfaces) as hard cores, to give you the best possible performance. We will compliment these hard cores with over 50 soft core peripheral functions. By keeping most of the peripherals as soft cores, you can choose only those functions that you need, and create custom designs with ease.

Advanced CMOS Manufacturing Capability

IBM is one of the most advanced CMOS semiconductor companies in the world, with device manufacturing technologies that are typically a year ahead of most other companies. Our partnership with IBM gives Xilinx access to this manufacturing technology, and a tremendous competitive advantage. To be competitive in our marketplace we have to push manufacturing technology to the limits. By using the most advanced manufacturing processes, we can reduce the size and cost of transistors, which enables us to continue building bigger and bigger FPGAs and reduce the costs of existing devices.

For IBM, FPGAs are the ideal "process drivers" to test and refine their advanced manufacturing processes. Because FPGAs have very regular structures and they allow us to address almost every square micron of space on a chip, it makes them ideal for troubleshooting problem process areas. So, Xilinx gets advanced manufacturing technology and IBM gets devices that help them drive their manufacturing process to maturity. Thus, we can achieve better yields, faster, and that means lower costs.

Xilinx has been the leader in developing programmable logic technology, and we have expanded the market dramatically–today, the PLD business is growing 40% faster than the regular IC business. Our current technologies, bigger densities, higher speeds, and lower costs, are expanding the market much faster than in the past; with IBM, we are pushing it even further.

Gigabit Serial I/O Capability

Many new systems today are requiring much faster data transfer between systems, boards, and devices, due primarily to the ever increasing demand for faster networks. Very high speed (gigabit per second) serial I/O capability promises to solve this difficult problem.

Traditionally, data has been shared by using parallel busses such as PCI (Peripheral Component Interconnect). However, there are inherent limitations with shared busses. To increase the speed of a shared bus, you can either increase the speed of each wire (which is very difficult to do because there are many of them), or you can increase the number of wires (which takes more and more I/O pins). For example, PCI was once just 32 bits wide; now you also have 64-bit PCI–and that's not enough. The problem with this approach to increasing bandwidth is that at some point you reach a level of decreasing return; the extra pins and the need for shared bus protocols limits the performance and makes it prohibitively expensive.

The best solution we have for this bandwidth bottleneck is to use point-to-point connections, over a single pair of wires, operating at very high speeds. Currently, with this technology, you can achieve a data rate of two to three gigabits per second. The big advantage of this method is that you use less wires and less power, and the total amount of data you can move is in fact higher than with a typical parallel bus.

To create a gigabit serial I/O channel, a hard core is needed; you cannot achieve these speeds with soft cores in an FPGA. The hard core does several functions; it receives and transmits the data, and it also recovers the clock (because you can recover the clock from the data, a single pair of wires is all you need for data transfer). The hard core must also serialize and de-serialize the data. By de-serializing the data to a 16- or 32-bit internal bus, the data speed is then reduced by 16 or 32, which an FPGA can easily handle.

With gigabit serial I/O, all the de-serialization is done within the chip. When the work is done, you can serialize the data again and send it out to other devices, over a single pair of pins. This is a very efficient and low cost way to transfer data.

We recently made several announcements regarding our commitment to gigabit serial I/O.

The Conexant™ Partnership

Xilinx recently entered into a strategic development and licensing partnership with Conexant Systems, to integrate their SkyRail™ 3.125 Gbps serial transceiver technology into our next generation Virtex-II FPGAs. This hard core is the fastest core available in CMOS today, and will be available in the second half of 2001 in a select offering of several different Virtex-II devices. In our Virtex-II architecture you will get more than 20 different I/O standards, plus several of these gigabit serial I/O channels.

The high-speed SkyRail transceiver is compliant with industry standards such as Gigabit Ethernet and Fibre Channel in addition to the emerging 10-Gigabit Ethernet (IEEE 802.3ae) standard. By integrating quad transceivers, which are used to create 10-gigabit attachment unit (XAUI) interfaces, a single FPGA can interface to both 10-Gigabit Ethernet and OC-192c. The high-speed transceiver is also compliant with the 2.5 Gbps InfiniBand™ architecture standard being created by the InfiniBand Trade Association.

The RocketChips Acquisition

High-speed serial I/O capability is so important, we decided not to stop at the 3.125 Gbps speed offered by the Conexant core–we are developing the technology further. That's why we recently acquired a company called RocketChips, which is very active in creating high speed serial I/O cores. RocketChips already has a product that is very similar to the Conexant core, and they plan to develop even higher speed cores operating at 5 to 10 Gbps.

RocketChips' gigabit and multi-gigabit serial CMOS transceiver technologies provide solutions for a wide range of serial system architectures in networking, telecommunications, and enterprise storage markets. Their products include serial backplane transceivers (Single and Quad 3.125 Gbps transceivers), telecom transceivers (SONET OC-48 and OC-192), enterprise storage transceivers (Fibre Channel, Ethernet), and networking transceivers (Gigabit Ethernet, 10 Gbps Ethernet, and InfiniBand).

PMC-Sierra Partnership

Xilinx recently announced the availability of POS-PHY™ Level 3 Link Layer and Physical Layer cores. These cores provide solutions for the emerging Packet Over SONET (POS-PHY) applications, and both cores are compatible with the POS-PHY Level 3 interface specified by the SATURN® Development Group. With these cores, broadband system designers can rapidly develop highly functional, scalable, and standards-based equipment to increase the speed of networks up to 2.5 Gbps, and support the exploding growth of IP traffic over SONET/SDH backbones.

Xilinx has also been active in the Optical Internetworking Forum (OIF) and the ATM Forum to drive POS-PHY Level 4 acceptance. And, we are the only FPGA company to demonstrate over 800 Mbps operation, confirming that we can provide the full speed capability to support the 10 Gbps OC-192 draft standard at the OIF (OIF2000.088.2).

Serial Protocol Standards

To use these high speed serial I/O channels effectively, you need well defined protocols and networking standards. Xilinx actively supports all of the emerging standards, including:

- Lightning Data Transport™ (LDT) - A chip-to-chip interconnect that provides much greater bandwidth per I/O channel. It can achieve a bandwidth of 6.4 Gbps per eight-wire link width, and can support up to 32 links.
- InfiniBand™ - This newly designed interconnect system utilizes a 2.5 Gbps wire speed connection with one, four, or twelve wire link widths. Promoted by an association comprising industry leaders such as, Compaq, Dell, HP, IBM, Intel, Microsoft, and Sun Microsystems, InfiniBand intends to deliver a channel based, switched fabric technology.
- XAUI - A quad transceiver utilizing 3.125 Gbps serial links to create a 10 gigabit attachment unit interface (XAUI). Multiple XAUI interfaces can be implemented to allow a single chip to interface to both 10 Gigabit Ethernet and OC-192c.
- Fibre Channel - A high-bandwidth serial standard offering 1.06 Gbps data rates scalable to 2.12 or 4.24 Gbps. It is capable of carrying multiple existing interface command sets, including Internet Protocol (IP), SCSI, IPI, HIPPI-FP, and audio/video.
- Gigabit Ethernet + 10 Gbit Ethernet - This includes devices compliant with the IEEE 802.3 alliance.
- ATM (OC-12, OC-48, OC-192) - This includes support for OC-12 (622 Mbps), OC-48 (2.4 Gbps), and OC-192 (10 Gbps).
- RapidIO™ - A next-generation switched-fabric interconnect architecture for embedded systems that is optimized for both high bandwidth and low latency. Initial implementations are expected to exceed 1.0 Gbps throughput based on clock rates of 250 MHz and higher.

These standards all use the same physical interface, so you can use our hard I/O cores for all of them. Then, we implement the level-2 protocols in programmable logic (soft cores), so you can quickly create designs using any of these standards. This gives you a lot of flexibility and it helps you interface directly to on-site networks.

What Does It Mean?

An FPGA is no longer just gates and routing. Over the years we have added more and more hard cores, such as memory, clock management, and arithmetic functions. Now we are driving the technology a major step further by adding hard CPU cores and high speed serial I/O cores. Combine these dramatic technology advances with our high performance development tools, our unique Internet Reconfigurable Logic capability, our extensive training and support services, our state-of-the-art manufacturing capabilities, and our ongoing partnerships with other industry leaders, and you get a logic design solution that can breathe life into your new designs.

The role of the FPGA is changing; it is becoming a platform on which the combination of soft cores, programmable logic, and hard cores gives you the best possible design solution–the speed of a custom ASIC and the time-to-market advantages of a flexible FPGA.

THE ROLE OF THE FPGA IS CHANGING; IT IS BECOMING A PLATFORM ON WHICH THE COMBINATION OF SOFT CORES, PROGRAMMABLE LOGIC, AND HARD CORES GIVES YOU THE BEST POSSIBLE DESIGN SOLUTION–THE SPEED OF A CUSTOM ASIC AND THE TIME-TO-MARKET ADVANTAGES OF A FLEXIBLE FPGA.

Designing with FPGA Platforms

The Chief Technology Officer at Synopsys discusses the need for platform-based design in the era of system-on-a-chip FPGAs.

by Raul Camposano
Chief Technology Officer, Synopsys, Inc.
raul@synopsys.com

As FPGAs move into million-gate densities, a world of new possibilities and potential applications is opening up for programmable logic devices. And along with these new opportunities comes many new challenges. The pairing of a low-cost, high performance PowerPC processor (and other hard cores), along with the soft cores and programmable logic circuitry in Xilinx Virtex-II™ FPGAs means that you will now be confronting challenges similar to what ASIC designers encountered when they made the transition to system-on-a-chip (SoC) ASICs.

Everyone who participates in the multimillion-gate segment of the FPGA market is wrestling with the same issues: increased complexity, escalating development costs, evolving standards, too few design engineers, increasingly compressed design cycles, and so on. In addition, as complexity increases, the time it takes you to get a product to market becomes dominated more by your design time than by manufacturing considerations, compromising one of the key advantages of FPGAs.

To help address these challenges, there is increasing pressure for designs to share a common architecture or platform, especially those that are targeted to similar applications. A platform is a basic system architec-

ture that is geared towards a specific application, such as cell phone base stations or set-top boxes, among others; it is customized through software and by adding customized logic and IP.

An FPGA platform enables you to differentiate your products by adding customized logic and IP using the tightly integrated FPGA fabric. Platforms are important in the era of multimillion-gate FPGAs because they enable you to focus on adding value through custom IP rather than wasting time and resources by recreating standard components.

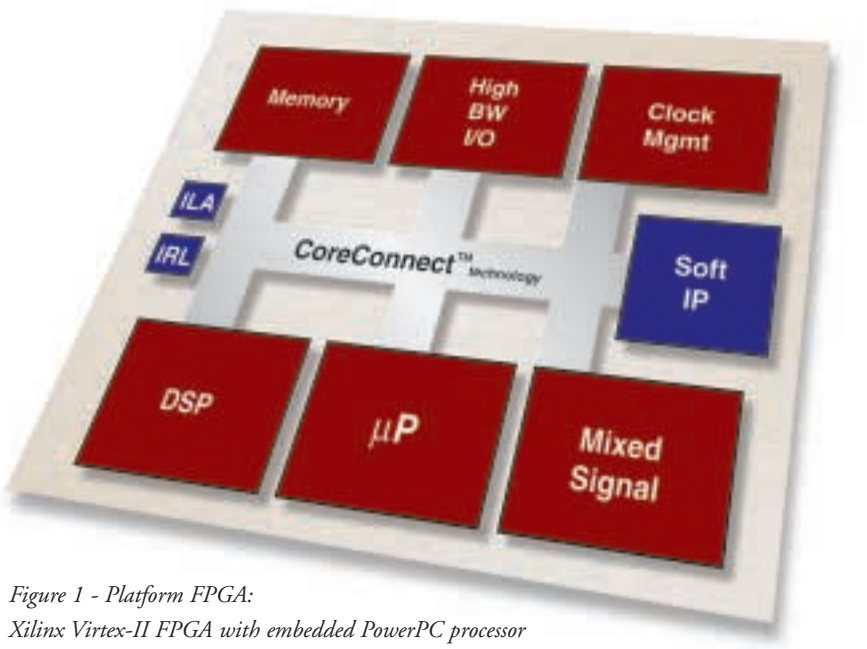


Figure 1 - Platform FPGA:
Xilinx Virtex-II FPGA with embedded PowerPC processor

Platform-Based Design

A central piece of any platform is the embedded processor, such as the IBM PowerPC processor core in the Xilinx Virtex-II platform. A typical platform might also include a bus, DSP, input/output channels, mixed signal functions, memory, and some configurable logic such as shown in Figure 1. FPGA design thus becomes platform design; rather than simply designing with gates, you must now focus on designing entire systems.

For you to effectively exploit a platform by designing at the system level, four primary design considerations must be addressed:

- Hardware design.
- Software design.
- Integration of hardware, software, and IP.
- Verification of the complete system (on a chip).

Synopsys delivers solutions in all four of these areas. The design of hardware has been our traditional domain, and we offer a suite of FPGA synthesis tools for this purpose. FPGA Express™ addresses the push-button, fast turn-around market, while FPGA Compiler II™ addresses more complex designs and compatibility with the ASIC design flow. Looking further into the future, other synthesis technologies such as Synopsys' Physical Synthesis will enable full timing closure for platform FPGAs.

Open SystemC

One of the most difficult aspects of software design involves how to interface software effectively with hardware. Open SystemC, a set of C++ class libraries that enables electronic design at the system level, provides an important tool for designing software and hardware in a common language framework. Based on C and C++ (the languages of choice for most algorithm developers, system architects, and software developers) SystemC also includes all the language elements necessary to effectively address hardware design. In this way, trade-offs between hardware and software can be addressed dynamically, even including reconfiguration in the field.

SystemC helps you create both systems and chips; the suite of tools and methodologies Synopsys has developed around SystemC significantly accelerate the design of elec-

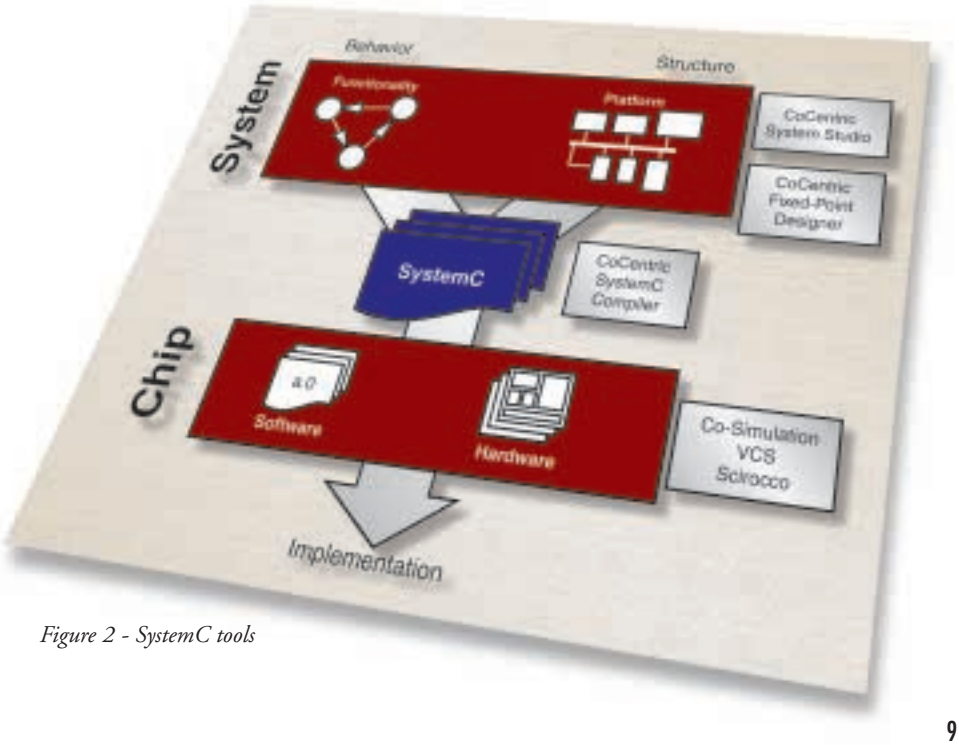


Figure 2 - SystemC tools

tronic systems from concept to implementation (see Figure 2). SystemC follows the community source-licensing model and can be downloaded from the Open SystemC Initiative's website at www.SystemC.org/.

IP Integration

One of the advantages of platform-based design is that it supports the integration of other pieces of proprietary logic and third-party IP. In fact, it is the customized portion of any system-on-a-chip ASIC or platform FPGA that provides the competitive differentiation from one device to the next.

But with the very large number of gates that can now be implemented on a single FPGA, the challenge is to become significantly more productive when creating with these gates. One obvious solution is to leverage existing gates through design reuse. Synopsys has been leading the way in this area and, with its DesignWare® libraries of reusable building blocks and methodology activities, offers several time-saving options for both ASIC and FPGA designers to leverage IP from a variety of sources.

System Verification

The challenge for any system-on-a-chip FPGA is to verify the complete system, including the processor core, and not just the individual blocks that comprise the system. This requires not only a high-speed simulator, but also a complete array of advanced verification tools. In particular, testbench generation, coverage tools, formal verification, a simulation model of the processor and other IP, and static timing analysis tools are essential for platform-based design (see Figure 3).

Static timing analysis illustrates the verification challenges imposed by such a system. Synopsys' PrimeTime® static timing analysis tool can time and analyze a complete chip, offering the multimillion-gate capacity that is required by systems on a chip. It also offers analysis modes that handle the processor core in an effective way.

Conclusion

FPGAs that contain embedded processor cores and application-specific components are creating a need for platform-based design, which requires not only the suite of RTL logic design tools that are already in use today (with the right capacity), but also a comprehensive suite of system-level design tools that will be new to most FPGA designers.

FPGA design has moved beyond the era of simple logic. With the advent of FPGAs that contain an embedded processor core, such as the PowerPC, FPGA designers will soon join their peers in the ASIC world by confronting the challenges of designing entire systems. Synopsys is helping you meet these challenges through the power of its system-level EDA tools optimized for platform-based design.

For more information on all Synopsys products, see www.synopsys.com

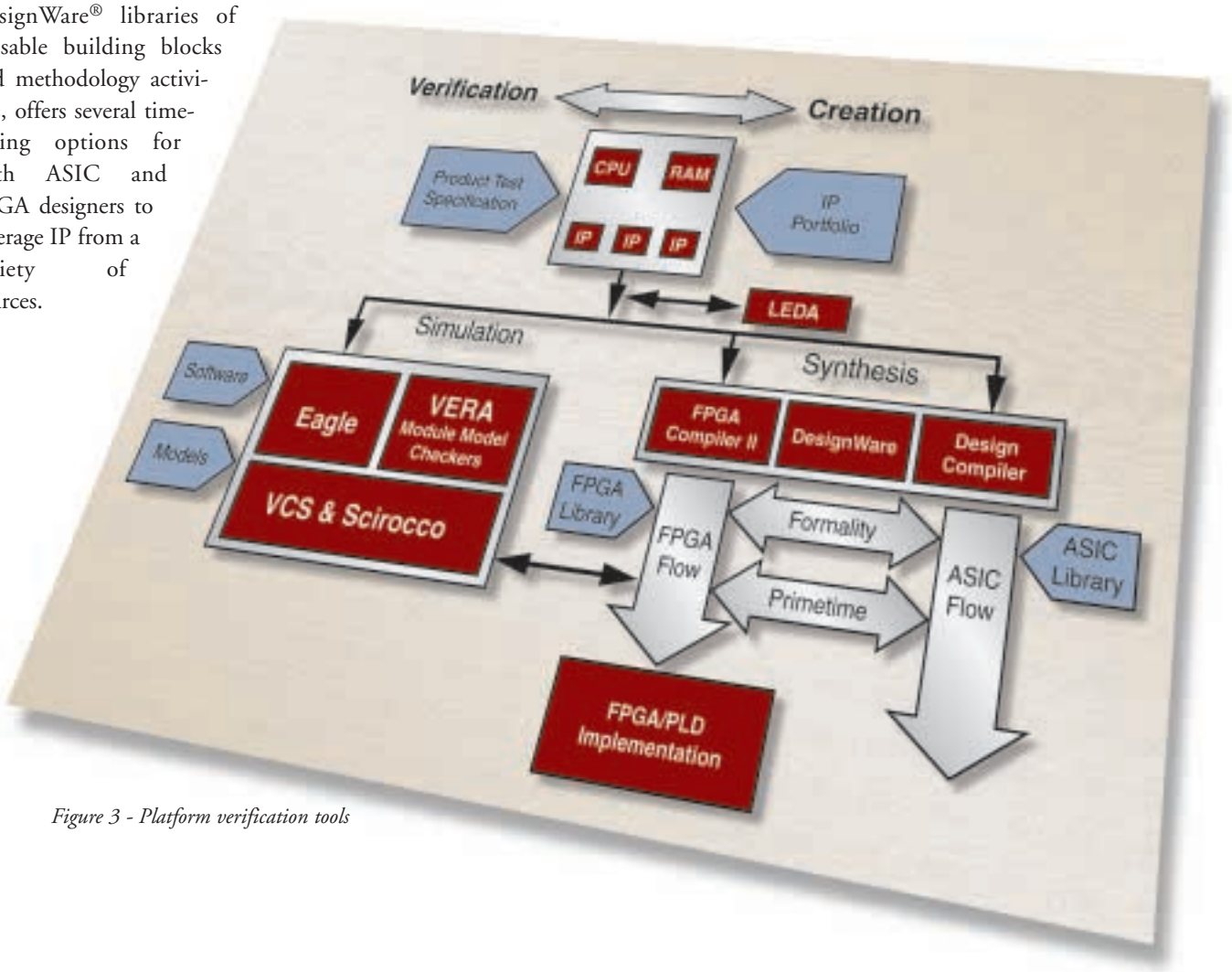


Figure 3 - Platform verification tools

Inferring Multiplexers in FPGA Compiler II and FPGA Express

How to get better results by automatically inferring multiplexers that fully utilize architecture-specific FPGA resources.

by Alan Ma
Senior Corporate Applications Engineer, Synopsys, Inc.
alanma@synopsys.com

In general, multiplexers can be implemented by using Look Up Tables (LUTs). To obtain the best quality of results (QoR), Synopsys FPGA Compiler II™ and FPGA Express™ (FCII/FE) take it one step further by utilizing the built-in multiplexer resources in high-density FPGAs, which produces significantly better results in both area and speed.

The Process

During elaboration, the process of translating the text-based description of a design to an architecture-independent gate-level representation, FCII/FE infers a generic primitive called MUX_OP when it encounters multiplexers in the Hardware Description Language (HDL). It is during optimization where MUX_OPs are mapped to architecture-specific multiplexer resources. The following sections describe the requirements for MUX_OP to be inferred.

General Implementation

Our research indicates that using architecture-specific multiplexer resources is only beneficial when the number of inputs meets certain requirements. Table 1 illustrates the multiplexer sizes and the primitives FCII/FE utilizes for Xilinx Virtex-II, Virtex, and XC4000 FPGAs (and their derivatives). FCII/FE automatically maps to these hardware resources (primitives) when you follow the recommended coding guidelines.

Coding Guidelines

Synopsys recommends the use of CASE statements to describe multiplexer logic. When the requirements on the number of inputs for the target architecture are met (as shown in Table 1), FCII/FE maps the design to architecture-specific multiplexer resources if at least 75% of all possible cases are specified.

Figure 1 shows an example of an eight-to-one multiplexer in Verilog. Figure 2 illustrates its VHDL equivalent. Note that the control signal sel has three bits so there can be as many as eight possible cases. As a result, at least six (75% of eight) cases need to be specified for multiplexers to be inferred.

Architecture	Min. Inputs	Max. Inputs	Primitives Used
Virtex-II	4	256	LUT, MUXF5, MUXF6
Virtex	4	256	LUT, MUXF5, MUXF6
XC4000	4	256	FMAP, HMAP

Table 1 - Multiplexer size requirements for automatic inference

Using the infer_mux Directive

Figure 3 shows a similar eight-to-one multiplexer with the addition of several arithmetic operators; Figure 4 shows its VHDL counterpart. To allow operator sharing, multiplexers are generally not automatically inferred for CASE statements which contain more than one operator (regardless of the number of cases specified). However, you have the option to override FCII/FE by using the infer_mux directive.

The infer_mux directive forces FCII/FE to infer multiplexers as long as at least 50% of all possible cases are specified. It can be used when:

- The requirements on the number of inputs (as shown in Table 1) are not met.
- The CASE statement contains more than one arithmetic operator.

It is important to understand that FCII/FE generally makes intelligent decisions on multiplexer inference based on the cost of doing so. For example, it may choose not to infer multiplexers, to allow operator sharing for better performance. As a result, QoR is likely to suffer if you override that decision by using infer_mux. Please use this directive with caution.

Conclusion

FPGA Compiler II and FPGA Express take advantage of Xilinx-specific multiplexer resources to deliver the best quality of results. The tools automatically infer multiplexers if the design complies with the coding guidelines and meets the requirements for the target architecture. You also have the option to force multiplexer inference by using the infer_mux directive.

Visit the Synopsys FPGA website at www.synopsys.com/fpga for other information on the latest FPGA synthesis technologies.

```
module mux_8to1   (
a, b, c, d, e, f, sel,
                  mux_out
);

 input                a, b, c, d, e, f;
 input     [2:0]      sel;
 output    [1:0]      mux_out;

 reg  [1:0]       mux_out;

 always @(sel or a or b or c or d or e or f)
case (sel)// synopsys infer_mux
            3'b000    : mux_out = a + b;
            3'b001    : mux_out = a + c;
            3'b010    : mux_out = d - e;
            default   : mux_out = d - f;
endcase
 endmodule
```

Figure 1 - Using CASE statements for multiplexers in Verilog

```
library ieee;
 use ieee.std_logic_1164.all;

 entity mux_8to1 is port  (
                  a, b, c, d, e, f: in std_logic;
sel: in std_logic_vector(2 downto 0);
                  mux_out: out std_logic
);
 end mux_8to1;

 architecture rtl of mux_8to1 is
 begin
process (sel, a, b, c, d, e, f)
begin
case sel is
                  when "000"     => mux_out <= a;
                  when "001"     => mux_out <= b;
                  when "010"     => mux_out <= c;
                  when "011"     => mux_out <= d;
                  when "100"     => mux_out <= e;
                  when others    => mux_out <= f;
end case;
end process;
 end rtl;
```

Figure 2 - Using CASE statements for multiplexers in VHDL

```
module mux_8to1   (
a, b, c, d, e, f, sel,
                  mux_out
);

 input                a, b, c, d, e, f;
 input     [2:0]      sel;
 output               mux_out;

 reg                mux_out;

 always @(sel or a or b or c or d or e or f)
case (sel)
            3'b000    : mux_out = a;
            3'b001    : mux_out = b;
            3'b010    : mux_out = c;
            3'b011    : mux_out = d;
            3'b100    : mux_out = e;
            default   : mux_out = f;
endcase
 endmodule
```

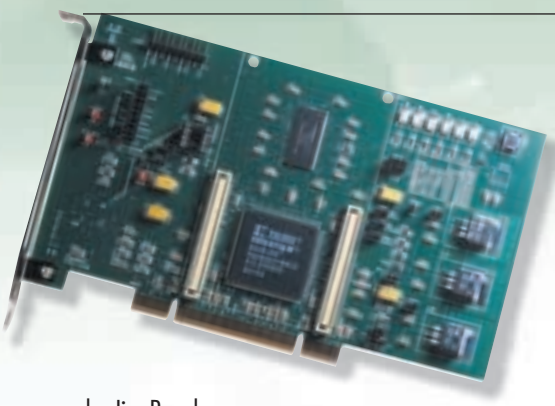
Figure 3 - Using infer_mux for multiplexer inference in Verilog

```
library ieee;
 use ieee.std_logic_1164.all;

 entity mux_8to1 is port  (
            a, b, c, d, e, f: in std_logic;
            sel: in std_logic_vector(2 downto 0);
            mux_out: out std_logic_vector(1 downto 0)
            );
end mux_8to1;

 architecture rtl of mux_8to1 is
 begin
process (sel, a, b, c, d, e, f)
begin
case sel is       -- synopsys infer_mux
            when "000"     => mux_out <= a + b;
            when "001"     => mux_out <= a + c;
            when "010"     => mux_out <= d - e;
            when others    => mux_out <= d - f;
end case;
end process;
 end rtl;
```

Figure 4 - Using infer_mux for multiplexer inference in VHDL



Spartan-II PCI Development Kit

Insight Electronics has introduced a Spartan-II PCI Development Kit to help you jumpstart your next 32-bit PCI design.

by Jim Beneke
Technical Marketing Manager, Insight Electronics
jim_beneke@ins.memec.com

When Xilinx introduced the Spartan™-II FPGA family in January 2000, they not only offered the lowest cost FPGA devices with system-level features, they also enabled programmable logic to effectively replace off-the-shelf ASSP devices for 32-bit PCI applications. Combined with the proven PCI32 LogiCORE™ interface from Xilinx, the Spartan-II PCI solution was the common-sense choice for most PCI designs.

Unfortunately, designers wishing to target a Spartan-II device for a PCI project, were not able to prototype their design with an off-the-shelf PCI platform. Insight Electronics recognized the need for this type of development board, and introduced the Spartan-II PCI Development Kit. The kit includes a Spartan-II prototype board, single-use Spartan PCI32 LogiCORE license, Windows driver development software, one-day (eight hours) of Insight Design Services support, reference designs, Windows-based applications, example Windows 98/NT drivers, source code, and hardware documentation. The demonstration board is based on the 150K-gate Spartan-II FPGA, in a 208-pin plastic quad flat package (PQFP).

Implementing the full initiator/target PCI interface in the FPGA only consumes about ten percent of the logic resources, leaving approximately 135K gates for custom user back ends. Unlike other PCI prototype cards, the Spartan-II PCI board does not contain back-end application circuits to complicate your custom design. Instead, all user I/Os are brought out to expansion connectors for easy access and interfacing. This allows your designs to be quickly implemented, configured, and tested. Figure 1 shows a block diagram of the Spartan-II PCI card included in the kit.

The New Reference Design Center

In addition to the Spartan-II FPGA, the PCI board also includes the new Xilinx XC18V01 in-system programmable configuration PROM. This allows PCI application designs to be quickly downloaded multiple times to the board and saved in non-volatile memory.

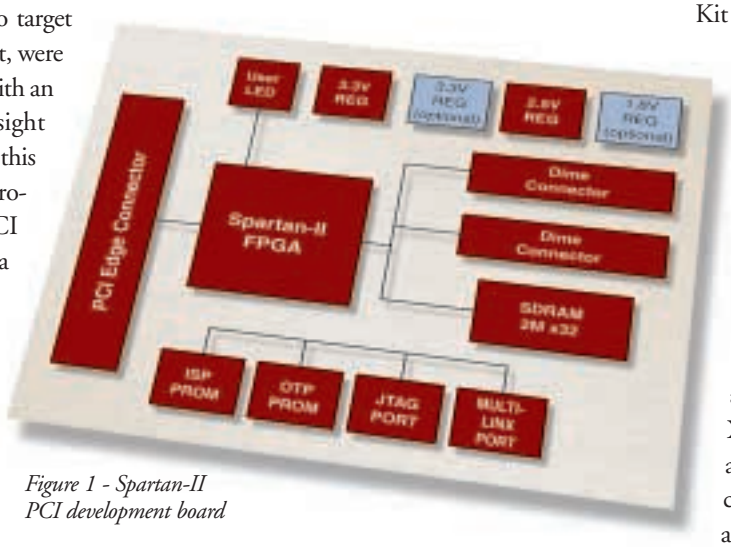


Figure 1 - Spartan-II PCI development board

With this re-configurable feature, Insight is including access to its new Reference Design Center. At the Reference Design Center, owners of the Spartan-II PCI kit can download pre-configured PCI application designs and run them on the demonstration board. Developed by Insight Design Services, these off-the-shelf application designs can be used as is, or can be customized to meet certain application needs. In addition to providing reference design bit streams and their associated source code files, the Reference Design Center also provides example Windows drivers and Windows-based application programs. Both drivers and application programs are provided with C++ source code so you can understand how the examples work.

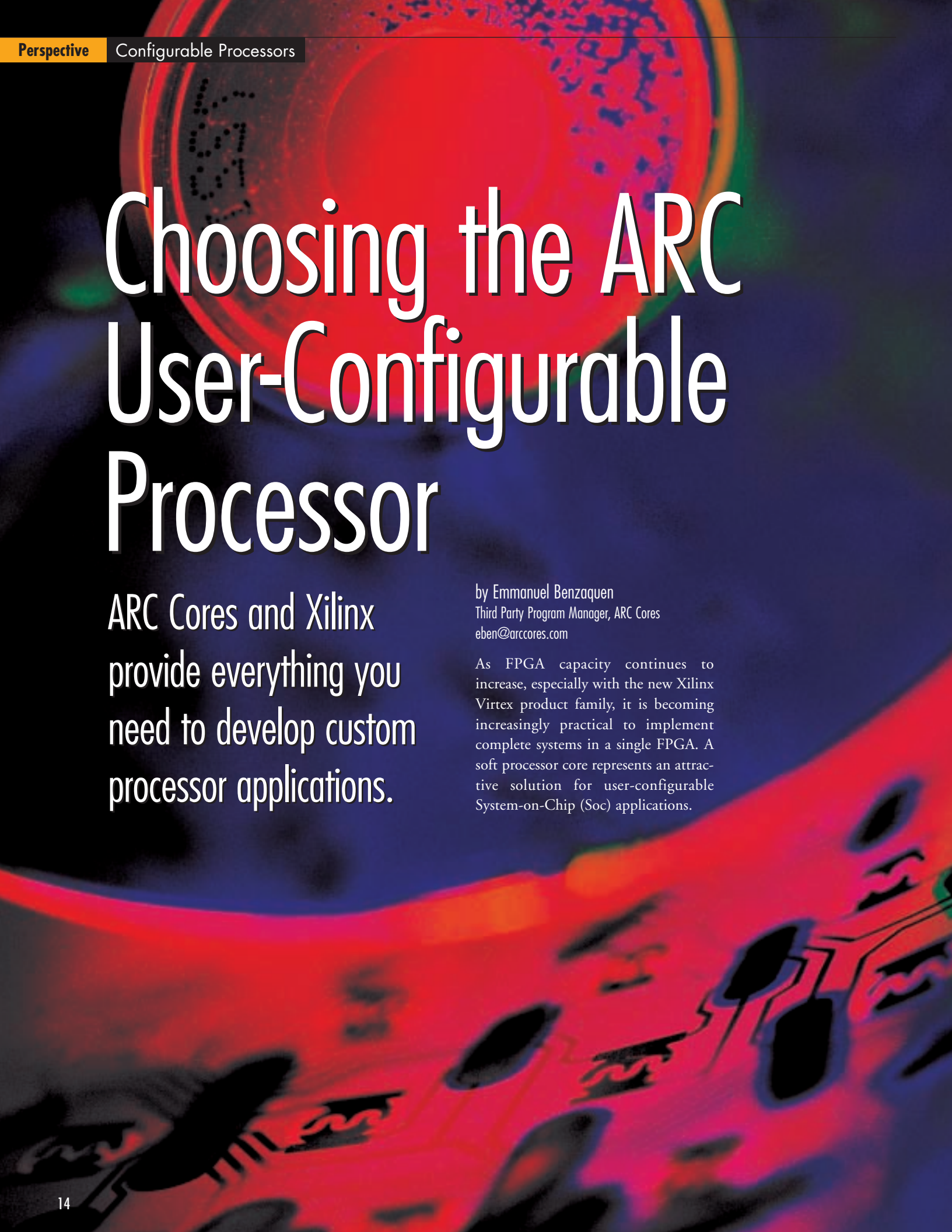
To assist in the development and debugging of Windows device drivers, the Spartan-II PCI Kit includes Compuware's NuMega driver development software. The NuMega package simplifies the task of writing and configuring Windows drivers through a series of GUI windows.

The Xilinx Spartan 32/33 PCI Core

The Insight Spartan-II PCI Development Kit includes the new single-use version of the Xilinx Spartan-only 32-bit, 33 MHz PCI core. The single-use license allows the kit owner to support a single production PCI core implementation. If multiple PCI core solutions are required, then the core license can be upgraded to an unlimited license for a nominal fee. The 32-bit Spartan PCI core is configured and downloaded through the Xilinx PCI Lounge. The downloadable core netlist is fully PCI v2.2 compliant and supports initiator and target functions with zero-wait-state burst operation.

Conclusion

By providing exactly what is needed to complete a PCI design, the Spartan-II PCI Kit meets the demands of both experienced and new designers of programmable logic-based PCI interfaces. Several versions of the Spartan-II PCI Development Kit are available from Insight Electronics. Prices range from $145 for a PCI card only kit, to $3,995 for the complete Spartan-II PCI Development Kit. For more information, go to www.insightelectronics.com/solutions/kits/xilinx/spartan-iipci.html.



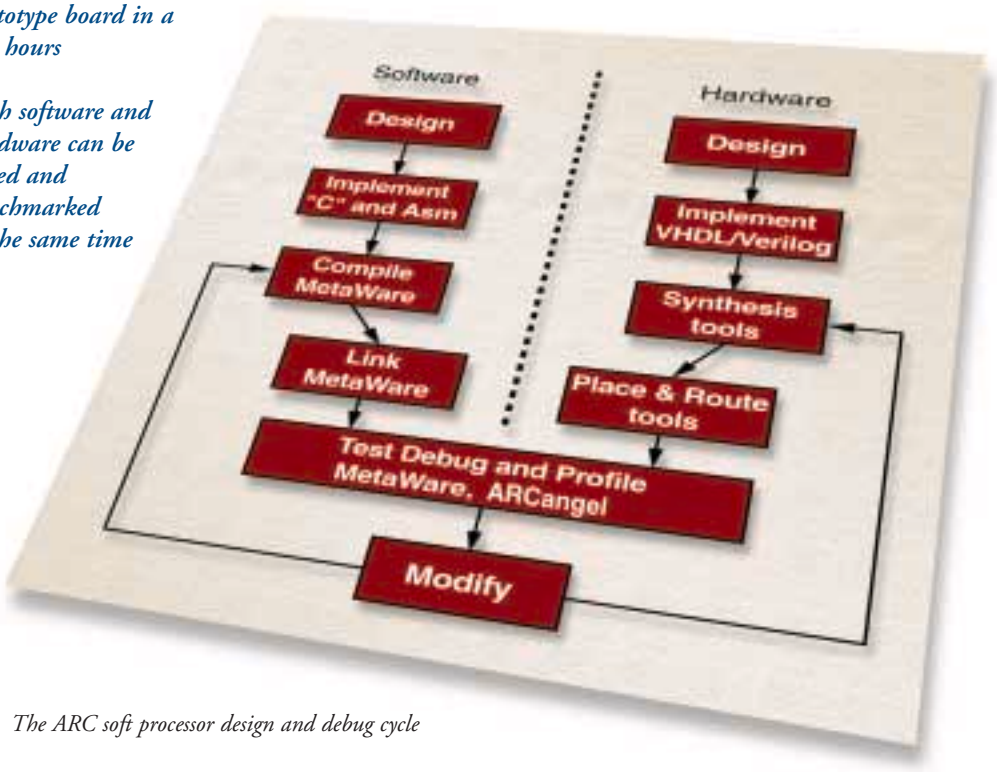
Choosing the ARC User-Configurable Processor

ARC Cores and Xilinx provide everything you need to develop custom processor applications.

by Emmanuel Benzaquen
Third Party Program Manager, ARC Cores
eben@arccores.com

As FPGA capacity continues to increase, especially with the new Xilinx Virtex product family, it is becoming increasingly practical to implement complete systems in a single FPGA. A soft processor core represents an attractive solution for user-configurable System-on-Chip (Soc) applications.

- *An ARC design can be turned from VHDL or Verilog into a configuration that runs on the Xilinx FPGA-based ARCangel prototype board in a few hours*
- *Both software and hardware can be tested and benchmarked at the same time*



The ARC soft processor design and debug cycle

Processor Cores Complement Programmable Logic

Traditionally, an important motivation for adding microprocessors into a design has been that software programmable solutions are easy to change and upgrade. Since FPGAs are, by definition, programmable, you can always upgrade them. System designers know that it is much easier to design and implement certain parts of a system using software, while hardware implementations offer greater performance. For example, you may want to take advantage of a large amount of low-cost software intellectual property (IP) that is available in C or C++ code, for functions such as protocol stacks and modem algorithms. You may also want to implement high-speed co-processor functions in hardware. You can get the best of both worlds when you combine the hardware re-programmability of FPGAs with the software programmability of microprocessors.

The Hard Versus Soft Option

Major FPGA vendors typically provide two different approaches to including processor cores in an FPGA. One approach offers a soft processor core that is provided in a synthesizable HDL format. This processor core is then included in a generic FPGA using the same design process as the rest of the logic. The second approach embeds a specific hard processor core (such as the PowerPC) into the FPGA. The most appropriate choice will depend on the application.

As a general rule, a hard processor core will offer higher clock speeds than a soft core. However, since the hard processor solution will require a specialized FPGA with dedicated processor buses and routing, it will be less flexible than incorporating a soft processor in a generic FPGA. In addition to performance and flexibility trade offs, the choice between a hard or a soft processor will also be influenced by the software applications you wish to run.

Since soft processors are available as synthesizable HDL, they inherently provide more design flexibility than hard processors because you can modify the core interface to fit better into a specific design. Some soft processors provide even greater flexibility by being configurable. A configurable processor core may include a graphical tool that enables certain functions to be included or excluded without having to manually modify HDL source code. As a result, you can create a processor core that is customized for your specific application by using the GUI.

The ARC processor is a perfect example of a soft and user-configurable core available for immediate use in Xilinx FPGAs.

Software Tools

The most important factor that will influence your choice of a soft processor is the software tool set that supports the code that will run on it. SoC designs can have lines of software code running anywhere from 1Kbyte to multi-megabytes. For applications that have only a few Kbytes of code, basic software tools such as an assembler may be sufficient. However once the amount of code starts increasing and becoming more complex, it becomes essential to use a high-level language like C or C++.

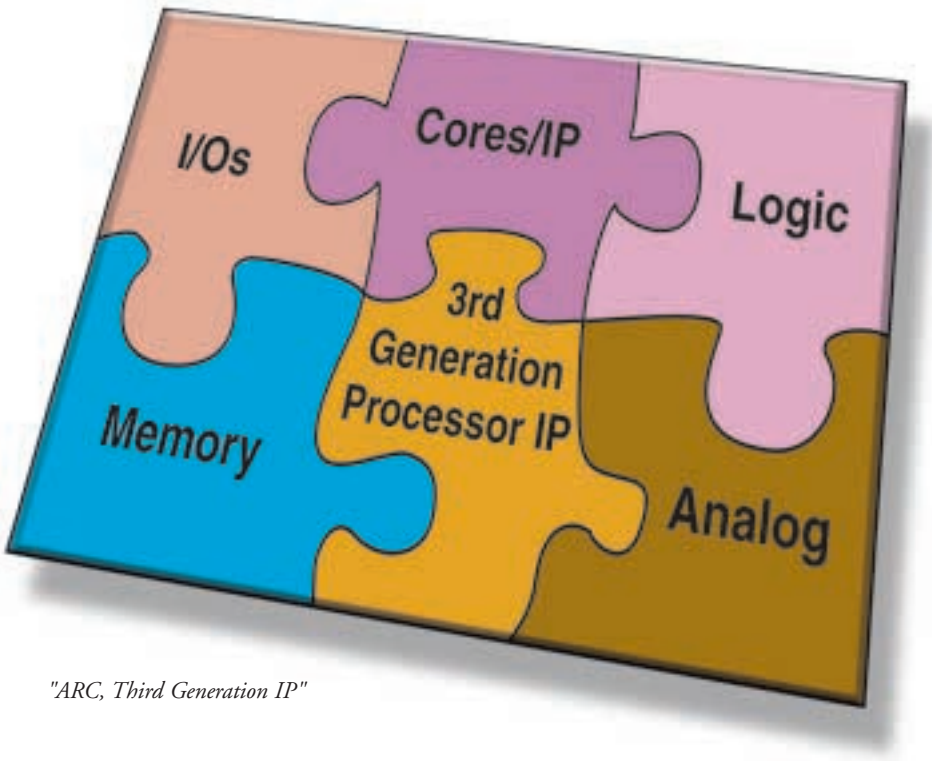
ARC Cores provide a complete set of high-level development tools customized for embedded applications, and offers both DSP (Digital Signal Processing) and general purpose control functions within the same processor architecture. There is no need to learn two different processor architectures and development tool environments.

Integrated Software Environment

Because of the typical complexity of the software code, ARC offers the Metaware development environment. This professional set of software development tools includes a C/C++ compiler, assembler/linker, and the SeeCode™ source-level debugger. Most importantly, it offers you the ability to debug the embedded software running on the processor in the FPGA. It is critical that the core and its host interface include execution control capabilities like breakpoint checking so you can break the program execution or monitor reads and writes to program variables.

As the software content of a design increases, another important factor is the range of applications supported and the available systems software. For example, if a design requires several hundred Kbytes of code along with standard communications software, such as TCP/IP protocols, you can save several months or more of design time by purchasing a real-time operating system (RTOS) that includes prepackaged protocols. ARC supports a large variety of commercially distributed RTOS from leading vendors and is constantly increasing their ease of integration.

In addition to the software tools and applications described above, another critical factor in choosing the ARC core is its level of flexibility. Unlike other configurable processors available today, which sometimes require you to manually "hack" the HDL code, the ARC processor core enables you to easily select special options for configuring the processor. Hacking the HDL code after configuring the processor core might break the core, or even make it incompatible with the software development tools. ARC provides the flexible ARChitect Graphic User Interface (GUI) that can be used to safely create your custom configured processor. This is very helpful when using a soft processor in an FPGA, and it allows you to experiment with different options and configurations within minutes.



"ARC, Third Generation IP"

- *The ARC IP is deeply embedded with the rest of the logic and interface directly with other customer logic function in the Xilinx FPGA*
- *"Gate-hungry" complex system buses and associated logic are no longer needed to reach high-performance because of the tight integration*

Instruction Set Flexibility

The instruction set is one of the most important aspects to consider when choosing a configurable processor. One potential disadvantage of soft processors is that they cannot attain the high clock speeds of a hard processor. For a conventional processor design, the clock speed is essentially the key determinant of performance. The ARC processor changes this equation by offering a configurable instruction set and the ability to add custom instructions. This enables you to accelerate an algorithm by selecting or adding a few appropriate (but powerful) instructions specifically needed for the application that is being executed. Thus, you can get the best of both RISC (Reduced Instruction Set Computer) and CISC (Complex Instruction Set Computer) processor design architectures. This approach provides high performance at lower clock speeds, while still maintaining a software programmable solution.

Instruction extensions are available from ARC and some third parties. Plug-ins can be used and implemented directly in the design. For additional capability, you can also create your own specific instructions. Custom instruction extensions offer you a particularly powerful way to accelerate application performance while retaining programmability. Consider the example of a DES (Digital Encryption Standard) encryption application: by adding specialist bit-permutation, cipher instructions and additional registers to hold the keys, it is possible to greatly accelerate a range of encryption algorithms.

To provide a truly configurable instruction set, it is also important that the number of clock cycles for an instruction extension is configurable. For example, the ARC processor enables the addition of multicycle instructions to the pipeline where desired, and single-cycle operations to proceed in parallel with long latency ones. This is an advantage over architectures that enforce a strict RISC paradigm where

every instruction must execute in a single cycle. Such restrictions may make it impossible to add very powerful, complex instructions that require multiple cycles to execute.

Interaction with Other Logic Functions

The ARC processor can further improve performance by enabling tight integration between the processor core and other logic on the FPGA. Traditional processor cores typically communicate with peripheral hardware via a system bus. To send data to the processor, the peripheral interrupts the processor, which then processes the interrupt using a software routine known as an ISR (Interrupt Service Routine). In addition to supporting this approach, ARC processor enables you to add new core extension registers. If desired, the new registers can be directly accessed by peripheral logic, enabling such devices to communicate with the processor directly. These alternative approaches can improve performance and reduce gate count by eliminating the need to duplicate a complex system bus and its arbitration logic in an FPGA.

It is no longer necessary to pass data via a bus or to interrupt the processor to have it load data from a memory-mapped register. Since the special registers are unique to a particular piece of peripheral logic, there is no need for any decoding or arbitration logic. The firmware simply selects the special purpose registers to communicate with the peripheral.

In addition to providing extension registers, configurable processors like the ARC core can also simplify integration with additional logic by providing multiple buses. This approach enables operations residing on separate buses, such as instruction fetches, load/stores, and communication with peripheral logic. As a result, the bus protocols of each bus can be relatively simple since there is no need to arbitrate between multiple devices attempting to control one bus. The ARC processor has four buses, consisting of instruction and data buses (Harvard architecture), a bus directly into the processor registers (primarily used for debugging), and an auxiliary bus (typically used to connect peripheral logic). The auxiliary bus has a very simple interface that virtually enables peripherals to be connected with just a few wires. This is well suited to FPGAs where there is no actual bus, allowing peripherals to be efficiently connected in a point-to-point manner.

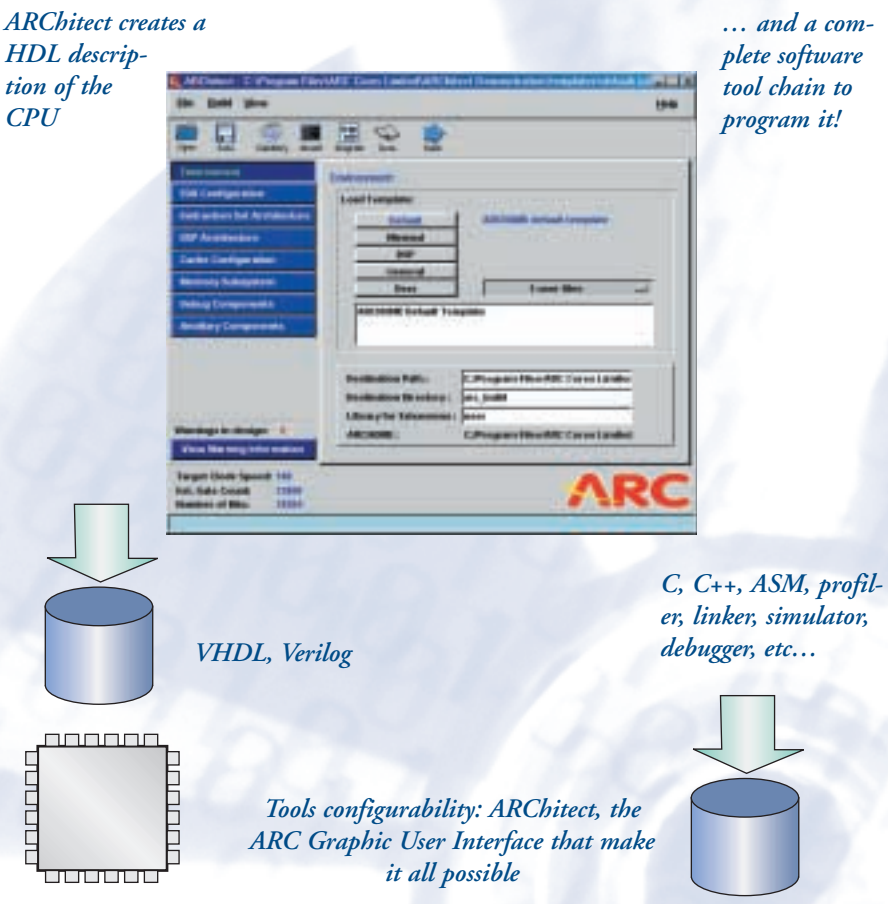
Tool Configurability

Any processor that offers a high degree of configurability must also offer equally configurable software tools and a debugging environment that work in coordination. It is of no use to add new instructions to the processor if there is no way of telling the compiler and assembler about them so that actual software programs can take advantage of them. In a similar vein, the compiler must let you specify which instructions will be present in the processor, as well as be able to take advantage of features such as multipliers or barrel shifters when they are included. In fact, software tool configurability is one of the greatest challenge in providing a truly configurable processor solution.

ARC and Xilinx are responding to this challenge by offering a complete "plug and play" solution to FPGA designers. In addition, the ARC tools suite allows you to enhance the original configurations offered in a simple manner.

Conclusion

Soft processor cores give you the ability to include processors in standard FPGAs. Configurable cores can help you achieve higher performance at lower clock rates through instruction extension and peripheral logic integration. ARC and Xilinx offer the perfect combination of a configurable core with powerful extensions and third party "plug-ins," in addition to a complete development environment and operating system support, ready to use with Xilinx FPGA technology.



Re-thinking Your Verification Strategies for Multimillion-gate FPGAs.

How do you alter your verification techniques to meet today's high gate count requirements? It depends on your background and experience.

by Thomas D. Tessier
President, t2design Incorporated
tomt@hdl-design.com

FPGA verification is essential for successful on-time product delivery, and today's million-gate FPGAs require you to rethink your old verification strategies. Many engineers continue to use simulator-specific approaches for verification; the simulation tools are primarily used for module testing, while the lab is used for system-level integration. This approach requires the engineer to manually stimulate signals and view the resulting waveform responses. Because this process is time consuming, error prone, and difficult to repeat, engineers often spend minimal time in simulation and quickly move to debugging in the lab. Multimillion-gate FPGAs implement functions far too complex to rely on this ad-hoc method.

Designers are choosing million-gate FPGAs because they are fast enough and large enough to handle the design complexity that was previously achievable only with an ASIC. When ASIC engineers begin to use high density FPGAs, they take their verification approaches with them. Those who use a validation process with robust tools and a complete self-checking testbench environment find that continuing to use their familiar testing approaches now causes them to loose valuable design cycle time. ASIC

Designers can benefit from a carefully defined and executed verification plan that takes FPGA reprogramability into consideration. Time that was once well spent in exhaustive verification at the RTL level with an ASIC, now becomes costly for a high density FPGA.

What is Verification?

Verification is not synonymous with simulation. It is a strategy to make sure all parts of the system conform to the specification document; simulation is a tool used in the verification effort. The basic components of verification are shown in Figure 1.

Specification

A detailed and complete specification is essential for producing working products, on schedule. The specification document is the foundation of the verification plan, and describes the features to be implemented, under what conditions they occur, and what their expected outputs should be. This documentation should not determine implementation–that is left to the experience of the RTL designers.

Verification Plan

RTL engineers and verification engineers share the responsibility for implementing the test plan. The level of test granularity (or detail) is outlined at: transactions, protocol, interfaces and timing. Essential functions are identified. A determination of the number of testbenches needed, their complexity, and test module dependencies is made.

Any discrepancies in design implementation versus testbench results should be referred back to the specification for clarification. This is not a new concept but often overlooked in the rush to produce a product. When all elements described within the test plan are checked off, the verification effort has been completed to the required level of confidence. To optimize your verification effort the following list offers examples of the type of information you need to identify:

- External interfaces
  - Stimulus and response
  - Transaction level, such as Read vs. Write operations
  - Timing requirements
- HDL models available to assist in testbench development
  - Packaged with proposed Intellectual Property (IP)
- Tools available to the project
  - Simulators
  - Static Analysis
  - Lab-based tools
- Performance Requirements, such as: need 32 block data write @ 66 MHz with a latency of less than 300 ns.

Execution

A verification strategy that best suits your design means breaking out those functions that are essential to simulate and those that can be tested during in-system test. The execution of the Verification Plan requires simulation and in-system test on the target PC board–the final stages of the pyramid.

Verification Simulation

Simulation has two components:

- Dynamic simulation describes behavioral HDL, RTL, and gates.
- Static analysis encompasses Static Timing Analysis (STA), Formal Verification and Signal Integrity Analysis.

In-System Test

During in-system test you have a distinct advantage when using FPGAs over ASICs. An obvious benefit is the ability to reprogram the FPGA until the desired functionality is achieved. You also have an additional advantage with the Xilinx ChipScope Integrated Logic Analyzer which enables you to observe internal nodes of the chip, on your PC board, while running at system speeds.

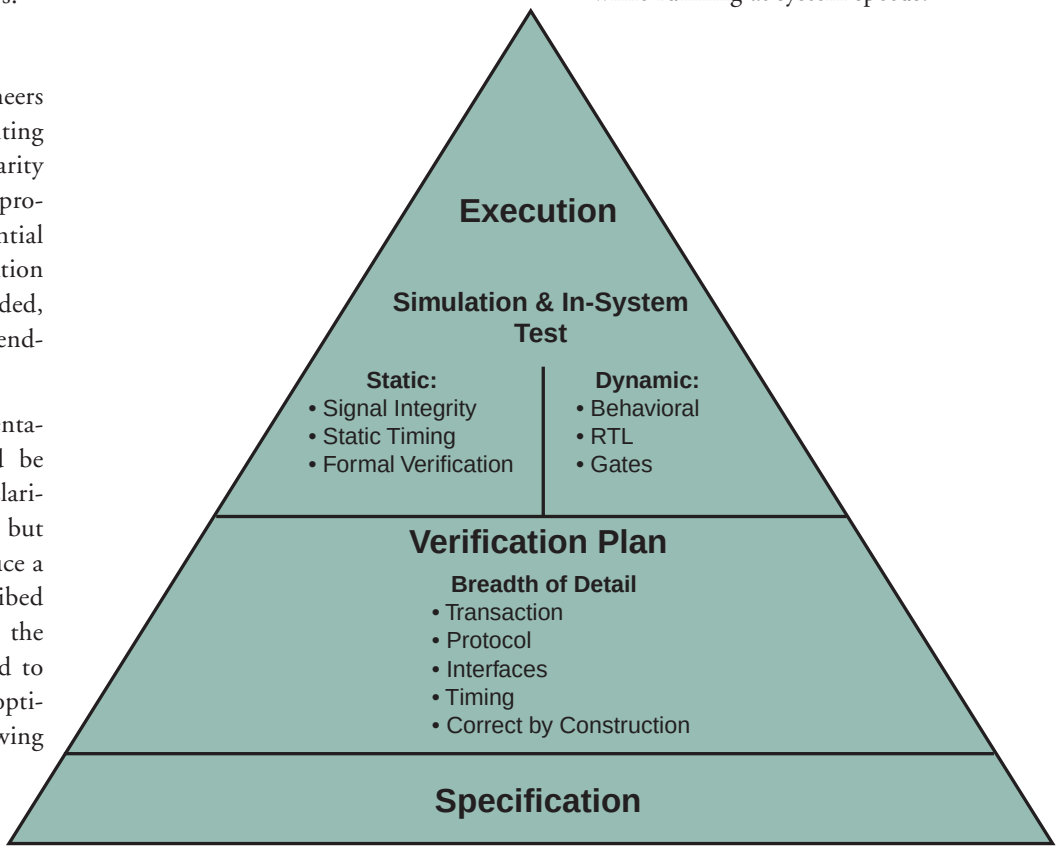


Figure 1 - Verification pyramid

Interaction of Verification and Design Creation

Verification has many interactions with design creation, as shown in Figure 2. To prevent confusion and save time, the design and verification teams must work from the same thorough specification. In addition, the RTL design engineers and verification engineers must share the responsibility for implementing the test plan–testbenches are written to validate the design to the specification, not to verify the design implementation.

Once the executable specification (of the design) and testbench, both written in behavioral HDL, meet the requirements, the design is replaced with RTL code. The RTL is then verified with the system-level testbenches to make sure it meets the written specification conditions. After the RTL is validated it is synthesized and processed by the Place and Route tools. The resulting gates are plugged into the system verification testbenches or formal verification if it is available. This insures the tools have correctly implemented the design. In addition generated gates are run thorough static timing analysis. This step verifies that the system-level timing is met.

System integration is typically referred to as “power on”. This is the time when project teams come up with creative answers to the question "is it working yet?" Projects are ready for in-system test when they have validated RTL code, have been successfully placed and routed, and can create a bit stream to program the FPGA on the physical PCB. At this point it is expected that module-level partitions have been tested for functionality, and that module interfaces are stable and well defined. The design has been simulated, as a chip, at both RTL and gates levels with minimum functionality necessary for power on. The simulation of the chip is often not achieved by FPGA design teams still using simulator specific approaches.

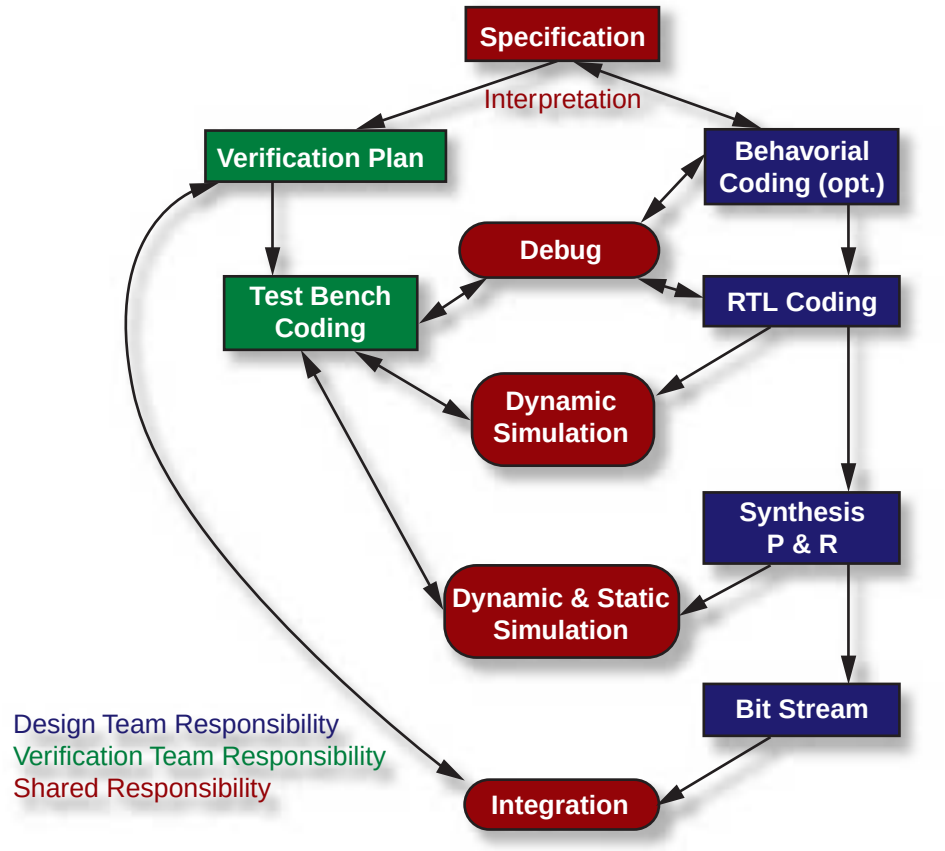


Figure 2 - Interaction of the verification components

Conclusion

A verification strategy combining simulation, static analysis, and in-system testing is key to success with high density FPGAs. You are bombarded with many different choices for verification of a design; to meet time-to-market pressures you need to leverage multiple approaches.

A detailed application note is available to guide you through the verification decision process, including an in-depth case study. It evaluates the design-specific trade-offs of choosing functions that are essential to simulate and those that can be tested during integration. Prepackaged IP testbenches are also evaluated for applicability in the system testbench. The full application note can be found at: www.hdl-design.com.

Foundation ISE – What's In a Name?

Xilinx Integrated Synthesis Environment stirs Design Automation Conference debate.

by Craig N. Willert
Software Marketing Manager, Xilinx
cnw@xilinx.com

The new 3.1i Foundation™ ISE software from Xilinx made its debut at this year's Design Automation Conference (DAC), leaving many with the question "What should ISE stand for?" Xilinx thought the name would speak for itself–Foundation ISE is an Integrated Synthesis Environment. But designers viewing the product for the first time at the DAC show excitedly came up with other ideas of what "ISE" should mean.

- "I" is for Ingenious, Intelligent, Internet-Enabled, Incremental, Innovative, Intriguing, Inspiring, Inventive, Imaginative, Insightful, Intuitive, and Interoperable.
- "S" is for Simple, Speedy, Sensible, State-of-the-art, Smart, Savvy, and Sexy.
- "E" is for Engineered, Easy, Efficient, Empowering (EDA partners), Expedient, Easy-to-Use, Extra-Special, Essential, and Eloquent.

What is ISE?

To understand the basis for the differing opinions, it's necessary to look at the current state of the design process.

Integrated design, synthesis, and implementation tools automatically handle all of the file dependency issues that any designer faces, by answering questions like "What tool do I need to run next," and "Have I re-synthesized all of the modified HDL blocks?" But time and time again, designers are synthesizing their designs with two or more synthesis tools–trying to create the most optimal design implementation from all of the variables.

To simplify this approach, Xilinx has built-in the HDL optimization using Xilinx Synthesis Technology and the FPGA Express HDL synthesis tools from Synopsis. This ensures that every engineer using Xilinx Foundation ISE will have access to at least two HDL synthesis tools that are highly compatible and tightly integrated.

Furthermore, a design "environment" is distinguished by its ability to address all of your needs as a designer, not just a few specific design functions. Foundation ISE provides an environment that ensures a comprehensive, integrated design flow for any programmable logic designer looking for an integrated solution that is capable of delivering world-class results with push-button flows.

Conclusion

The Xilinx 3.1i Foundation ISE software is already being heralded as the industry's best programmable logic design tool. By integrating the HDL design flow, synthesis, and optimization, Xilinx Foundation ISE enables you to spend more time on the creative aspects of programmable logic design. This helps you focus your resources and increase your productivity so you can get to market faster and deliver a more robust product to your customers. Xilinx 3.1i development systems deliver superior push-button, interactive, state-of-the-art design methods.

The 3.1i release will begin shipping to all registered, in-maintenance customers this Spring. To learn more, please visit the Xilinx website at: www.xilinx.com.

Year 2000 Worldwide Xilinx Event Schedules

Year 2000 European Event Schedule

Dec 5	Embedded Computing/Real Time Show Tel Aviv, Isreal

Year 2001 Worldwide Xilinx Event Schedules

Year 2001 North American Event Schedule

Jan 30-31	Portable Design 2001 Santa Clara, CA
Feb 2001	FPGA Conference 2001 Monterey, CA
Feb 13-15	Wireless Symposium 2001 San Jose, CA
March 5-7	Synopsys User's Group 2001 San Jose, CA
April 9-13	Embedded Systems Conference 2001 San Francisco, CA
April 23-26	NAB 2001 Las Vegas, NV
April 30-May 2	FCCM 2001 Rohnert Park, CA
May 8-10	ICASSP 2001 Salt Lake City, UT
May 15-16	Applied Computing Conference 2001 Santa Clara, CA
June 18-20	38th Design Automation Conference Las Vegas, NV
June 20-22	WITI Technology Summit 2001 Santa Clara, CA
June 24-27	ASEE Conference and Expo 2001 Albuquerque, NM

Year 2001 South East Asian Event Schedule

March 26-27	IIC 2001 Shanghai, China
March 29-30	IIC 2001 Beijing, China
April 2-3	IIC 2001 Shenzhen, China

Year 2001 Japanese Event Schedule

Feb 1-2	Electronic Design and Solution Fair Tokyo, Japan

For more information about Xilinx Worldwide Events, please contact one of the following Xilinx team members or see our website at: http://www.xilinx.com/company/events.htm

- North American Shows: Darby Mason-Merchant at: darby@xilinx.com or Jennifer Waibel at: jenn@xilinx.com
- European Shows: Andrea Fionda at: andrea.fionda@xilinx.com or Andrew Stock at: andrew.stock@xilinx.com
- Japanese Shows: Yumi Homura at: yumi.homura@xilinx.com
- SouthEast Asian Shows: Mary Leung at: mary.leung@xilinx.com

Xilinx Foundation Series ISE Software–Delivering the Benefits of HDL Design

Integrated design flows increase your productivity and accelerate your time to market.

by Justine Chen
Product Marketing Manager,
Worldwide Software Marketing, Xilinx
justine.chen@xilinx.com

Karen Fidelak
Product Marketing Manager,
Design Software Division,Xilinx
karen.fidelak@xilinx.com

Teams of software engineers from Synopsys, Synplicity, Model Technology, Visual Software Solutions, and Xilinx, working in close collaboration, have created the ultimate in design automation tools–Xilinx Foundation Series™ ISE (Integrated Synthesis Environment). The Foundation Series ISE software gives you the most advanced design automation tools, in a fully integrated, fast-working environment that increases your productivity and accelerates your time to market.

The Foundation Series ISE software includes:

- Synopsys FPGA Express - HDL synthesis software.
- Synplicity Synplify - HDL synthesis software.
- Model Technology ModelSim - HDL simulator.
- Visual Software Solutions HDL Bencher - Automatic testbench generation tools.
- Visual Software Solutions StateCAD-Automatic State machine generation tools.
- Xilinx XST synthesis technology - For further optimization.
- Xilinx implementation tools - For optimum use of device resources and the fastest place and route times in the industry.

The Keys to Increased Productivity

In the past, most large digital design companies relied on individual point tools, and were less concerned with managing the flow of data between the tools. Solving the problem of connecting point tools came later, and required customized design flows. This need to connect data flows between various point tools led to development of standard information exchange interfaces, such as HDL. But HDLs, including Verilog and VHDL, though useful as industry standards for hardware design, did not deliver a complete solution. For example, various simulation and synthesis tools might interpret and optimize differently, and produce undesirable results.

Today, there's a new focus. As more and more competing companies address the problem of designing a "system on a chip," they see more value in integrated tools that work together seamlessly, than in individual point tools, because tool integration is the key to increased productivity.

Integrated Design Flow Management

Today, you need fast, reliable flows of design information between tools. And, you want to specify common information, just once, for multiple tools; this includes the location of simulation libraries, macro libraries, and timing information. Though a homegrown, customized process for specification of common information can often be automated, updating a single point tool within a flow usually calls for a complete rewrite of setup information. And using various point tools within a design flow often requires creation of additional design data files. That additional design work and processing decreases your productivity, and slows time to market.

The Foundation Series ISE software automatically communicates common information to each tool and eliminates the need to create data file overhead. Unlike homegrown flow automation, an integrated design tool suite is aware of downstream tool requirements. For example, when you want to perform timing simulation after place and route, an integrated tool suite can instruct its place and route tools to produce the timing simulation netlist, so it can be read by the simulator. Today, winners in the race to market are focusing on design automation tools that are integrated (see Figure 1).

Integrated Project Management

Given the large number of source files, control files, and implementation files generated by today's complex, time-pressed design projects, it is not merely desirable, but necessary, to have an automated, integrated software tool that can manage project files. For example, a design project may consist of HDL files, IP cores, netlists, user constraints, or any combination of these. You know it can be difficult to manage the project when

one, or more of these design modules are modified.

The Foundation Series ISE software will manage all modules in the design for you. For example, it knows about all of the HDL code in your design, and it knows when the code has changed; therefore it will know, and can tell you, when HDL-generated netlists must be updated, and processes re-run. Then it will clearly display all design sources and implementation results, and provide easy access to the appropriate editing tool for every source file.

Many HDL compilers, as well as schematic entry tools, require that you specify a device family library up front, to provide appropriate library symbols and components for a given architecture. Additionally, if your design is retargeted to a new device architecture in the middle of your design project, then you must change the project libraries to match the new architecture. The Foundation Series ISE software makes the changes for you. You're left with nothing to do but select the device family, once. Your selection will set the appropriate device libraries for design entry. And automatically pass device information forward to place and route tools.

In the course of a design cycle, it's highly likely a design will be implemented many times. For example, revisions may be made to timing constraints, target device, and place and route options, in pursuit of the best overall design implementation. The Foundation Series ISE software provides revision control by archiving each implementation, along with all design flow control files and design constraint files, for future reference or use. With this information, you can consult or deploy an archived implementation anytime, without recompiling your entire design (see Figure 2).

Integrated Environment for Design Optimization

You usually have some overall design strategy that you are looking to optimize in your design flow. For example, your strategy may place highest priority on fitting the design in the smallest possible device, or on getting the fastest performance. A synthesis tool can be used to optimize the design's performance based on timing requirements, but for the best results, the place and route tools must then receive the same information to complete the design. This can mean setting requirements twice. However, with the Foundation Series ISE software, you only have to define the settings once, so you can optimize your design strategy faster and more reliably.

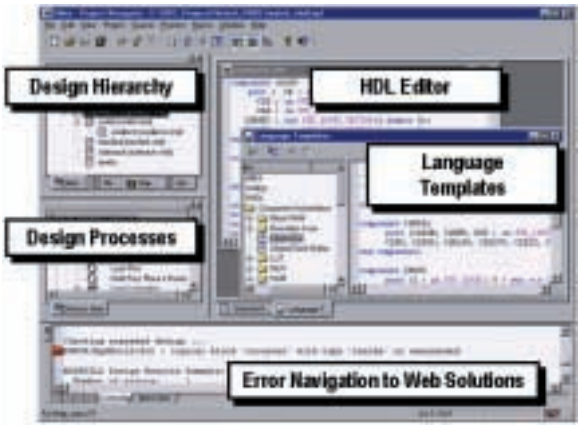


Figure 1 - Foundation Series ISE –well-integrated HDL solution

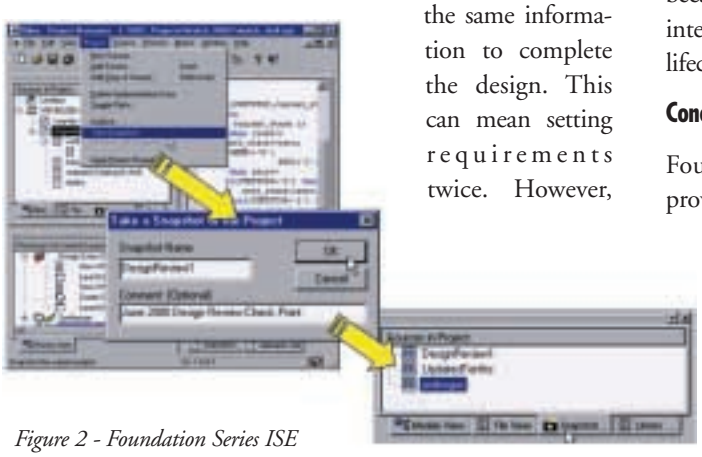


Figure 2 - Foundation Series ISE project snapshots for effective project management

The Foundation Series ISE software ensures that the software tools work well together; the tools must communicate with each other to efficiently transfer design data automatically. What's more, front to back design flow strategies are used, enabling the individual tool's features to be leveraged to their greatest benefits. In a non-integrated environment these communications tasks and decisions are left to you.

Integrated Environment for Collaboration

To facilitate the efficient flow of design data constraints and strategies, it is far more efficient if teams of software developers work in collaboration. An integrated environment makes possible, and enhances, collaborative work, which is critical during the project development phase. However, collaboration presents a new challenge.

Designers, working with an integrated tool, in an integrated environment, depend on software quality. When your in-house designers collaborate with third party partners for example, and use different tools, interoperability problems may occur; you can only hope solutions are available from each tool's vendor.

When you use the Foundation Series ISE software, you are assured of software quality because it has been tested thoroughly for tool interoperability, across the project creation lifecycle.

Conclusion

Foundation Series ISE provides you with a complete HDL design environment. Now you can manage and optimize your design projects, and your engineers can work collaboratively, with confidence in Xilinx quality and technical support.

Learn more about how Xilinx Foundation Series ISE meets your requirements for integrated design automation. See and hear the Xilinx internet presentation, "Xilinx Foundation Series ISE: Delivering the Benefits of HDL Design to Programmable Logic Designers," by going to www.netseminar.com/tbd/tbd.

StateCAD XE for Optimizing State Machine Design

Now you can implement faster, more compact state machines, with ease.

by Andy Bloom
Director of Engineering, Visual Software Solutions
(info@statecad.com)
ambloom@testbench.com

Ricky Escoto
Director of Marketing, Visual Software Solutions
(info@statecad.com)
rescoto@testbench.com

Control logic is usually implemented as finite state machines (FSMs), which usually require you to work through multiple levels of design and optimization, often within tight development schedules. And, as designs grow larger, the complexity of implementing control logic increases correspondingly, forcing you to migrate from schematics to hardware description languages (HDLs). StateCAD® XE automates the state machine development process, saving you a lot of time and trouble.

Manual FSM Design

Until recently, you had to specify control logic manually; you had to draw state diagrams by hand (or with a graphics package), and then manually translate them to schematics or to an HDL. Timing and logic problems identified during simulation resulted in modifications to the original design, which then needed to be re-verified, step-by-step.

This approach tends to be slow, repetitive, and error-prone. Translation errors invariably creep in and require substantial effort to eliminate.

Hardware Description Languages (HDLs) allow more logic to be specified and maintained with less effort, and they can be synthesized in numerous ways. You can control how synthesis operates, allowing you to create your design in the manner best suited to your target application.

The way an HDL is structured dramatically impacts the speed, area, and power consumption of the synthesized device. When doing finite state machine design, the best results can only be achieved by careful consideration of the resources available, and by having the flexibility to experiment with different alternatives.

Automated FSM Design Using StateCAD XE

A quicker way to implement state machines optimized for Xilinx devices is to use the Xilinx ISE software, which includes StateCAD XE. This tool allows you to draw complex state diagrams, choose design specific optimizations, and generate synthesizable VHDL, Verilog, or Abel-HDL. StateCAD allows you to change optimizations (including state assignment mode, registering output, and signal loading), then reproduce the HDL automatically.

One advantage of automatic state machine translation is the ability to change optimizations and regenerate code in seconds. By trying different code styles, state assignment modes, and optimizations, you can find which combination yields the optimal solution for your design.

State Machine Example

By comparing implementations of a simple state machine, we can see the impact on state machine design. The small state machine in Figure 1 will be implemented with both registered and combinatorial outputs, illustrating the impact of output optimization on implementation:

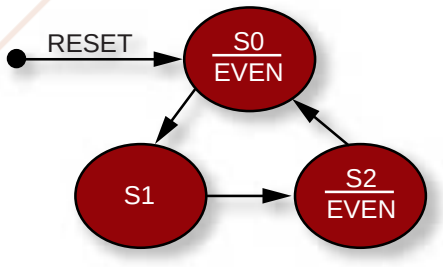


Figure 1 - Example state machine

Output Optimization

Outputs can be optimized for speed (registered) or for area (combinatorial decode). Combinatorial decoded outputs become active by decoding state registers (Moore) or by decoding state registers and inputs (Mealy). Registered outputs are calculated prior to the active edge of the clock, and typically improve speed because a level of propagation delay is removed, but usually require more area than combinatorial implementations. Registered outputs are insensitive to input glitches or to multiple state bit changes.

Design Results

In Table 1 you can see the registered design has outputs that change at the same time as the state bits, and are stable between clocks. The output delay time is the clock to output delay of the register. All decoding necessary for the output occurs before the clock, at the same time as the decoding for the next state. The decode time is effectively "buried" in the state decode time, producing a faster design.

In comparison, the combinatorial design requires time to decode the state bits, yielding a slower implementation. The advantage for the combinatorial design is the smaller area: 5 logic elements compared to 8 for the registered design.

REGISTERED OUTPUTS

```
PROCESS (sreg, RESET) BEGIN
   next_EVEN <= '0'; next_sreg<=S0;
   IF ( RESET='1' ) THEN
      next_sreg<=S0; next_EVEN<='1';
   ELSE
      CASE sreg IS
         WHEN S0 =>
next_sreg<=S1;
         WHEN S1
=> next_sreg<=S2; next_EVEN<='1';
         WHEN S2 => next_sreg<=S0;
next_EVEN<='1';
      END CASE;
   END IF;
END PROCESS;
```

COMBINATORIAL OUTPUTS

```
PROCESS (sreg, RESET) BEGIN
   EVEN <= '0'; next_sreg<=S0;
   IF ( RESET='1' ) THEN
      next_sreg<=S0; EVEN<='1';
   ELSE
      CASE sreg IS
         WHEN S0 => next_sreg<=S1;
EVEN<='1';
         WHEN S1 => next_sreg<=S2;
         WHEN S2 => next_sreg<=S0;
EVEN<='1';
      END CASE;
   END IF;
```

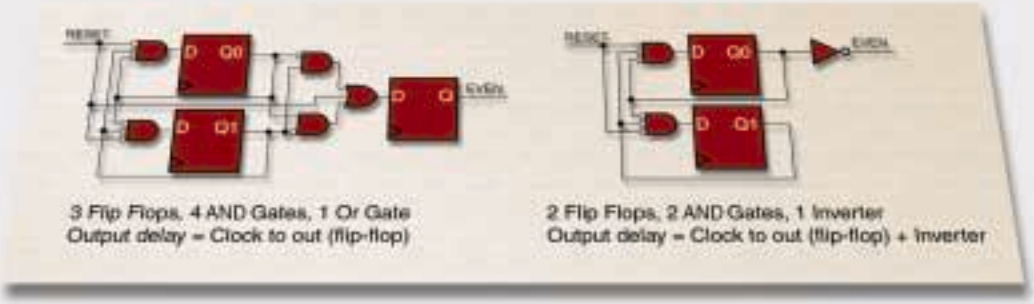


Table 1 - Comparison of output styles

Additional StateCAD Benefits

StateCAD provides additional benefits to Xilinx customers:

- By automating the complete state machine development process, the Xilinx ISE software and StateCAD eliminate manual coding, translation errors, stale documentation, and logic bugs.
- StateCAD includes wizards tailored for designing concurrent state machines and associated logic. State diagrams can include states, transitions, Mealy and Moore outputs, resets, counters, shifters, multiplexers, and much more. No HDL knowledge is required to specify control flow.
- StateCAD exhaustively analyzes state diagrams for inconsistencies, automatically identifying more than 200 problems, such as stuck-at-states, conflicting outputs, and non-deterministic control flow.
- StateCAD includes a built-in simulator called StateBench, for behavioral verification and identification of problems at the state diagram level.
- StateCAD automatically translates state diagrams to synthesizable VHDL and Verilog. Optimizations include one-hot state assignment, registered outputs, and prioritized transitions.
- StateCAD is fully integrated within the Xilinx ISE software, and produces HDL optimized for Xilinx devices, guaranteeing you the best possible results.
- StateCAD can import FSMs created with previous releases of the Xilinx Foundation Series software.

Conclusion

Using StateCAD XE you can quickly implement state machines optimized for Xilinx devices. As design parameters change, just select a new set of optimizations, then regenerate code suited for the new requirements.

StateCAD XE is available at no charge to Xilinx customers, and is included with the Xilinx ISE software or can be downloaded from www.xilinx.com (download StateCAD from the WebPack BackPack section).

HDL Bencher XE for Fast Behavioral FPGA Verification

Now you can develop complete, timing constrained VHDL and Verilog testbenches in minutes.

by Andy Bloom
Director of Engineering, Visual Software Solutions
(info@testbench.com)
ambloom@testbench.com

Ricky Escoto
Director of Marketing, Visual Software Solutions
(info@testbench.com)
rescoto@testbench.com

Validating FPGAs can require substantial effort, unless you have a high order software tool like HDL Bencher XE. The usual process requires you to write many testbenches, simulate them, check the results, and log all failures. To adequately test a design involves verifying all the possible cycle types available to the device, and may require several hundred test cases. And, as your design is revised, port definitions may change, making your existing testbenches obsolete, which results in unnecessary effort to update the HDL source code and the accompanying testbench.

To simplify FPGA and CPLD testing, Xilinx now includes Visual Software Solutions' HDL Bencher XE in the Foundation ISE and WebPack ISE design tools. No knowledge of HDL or scripting is required.

HDL Bencher Overview

HDL Bencher accepts any HDL design, and then lets you select the unit under test (UUT), specify stimulus and response (using the pattern wizard and the WaveTable™ spreadsheet-based interface), and then export a complete, self-checking testbench automatically. If your HDL source contains external dependencies, HDL Bencher prompts you to compile them locally so that the whole design can be simulated.

HDL Bencher lets you manipulate waveforms in the same way you manipulate a spreadsheet. You can cut, insert, and paste rows (signals) or columns (time regions) with ease, and HDL Bencher automatically readjusts timing.

Interactive Simulation

The testbenches are automatically updated when the HDL source changes, eliminating stale test cases. To facilitate design retargeting, HDL Bencher allows testbenches to be moved between VHDL and Verilog with one simple command. When compilation errors are found during simulation, HDL Bencher links the error reported to the offending line in the HDL source.

Create Self-Checking Testbenches

The testbenches include component instantiations, generic specifications, stimulus, output check procedures, and assertions. You can create "golden models" for regression testing and future design validation; mismatches in expected and actual output values are flagged automatically. All the necessary timing constraints are faithfully represented in the resulting testbench.

Verify Timing

By adding timing constraints, you can generate VHDL or Verilog testbenches for post-synthesis verification. Synthesized netlists differ from behavioral HDL because data types are remapped, I/O modes are changed, unused signals are dropped, and generics are flattened. HDL Bencher automatically remaps behavioral testbenches to simulate with synthesized netlists.

Demonstration Design

As an example, the following HDL code is used as input into HDL Bencher:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;
entity counter is
    Port (
      CLK,RESET,CE : in std_logic;
      T : out std_logic;
      COUNT : inout integer range 0 to 7 := 0);
end counter;
architecture behavioral of counter is
begin
   process (CLK, RESET, CE, COUNT)
   begin
      if RESET='1' then
         COUNT <= 0;
      elsif CLK='1' and CLK'event then
         if CE='1' then COUNT <= COUNT + 1;
         else COUNT <= COUNT;
         end if;
      end if;
      if COUNT=1 then T<='1'; else T<='0'; end if;
   end process;
end behavioral;
```

Within the Xilinx ISE software, you start by selecting the HDL file from the source window, then you choose HDL Bencher from the process window. The design is automatically imported, and you are given the opportunity to select worst-case global timing parameters:

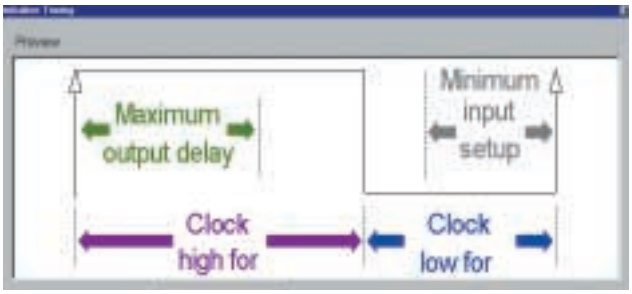


Figure 1 - Initial timing dialog

Figure 2 - Stimulus for the design example

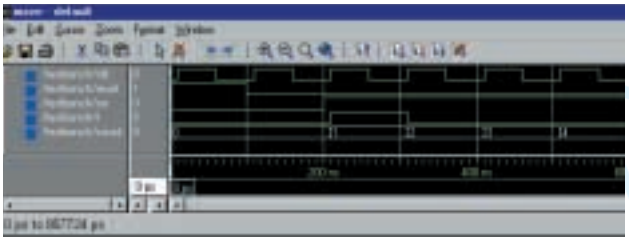


Figure 3 - ModelSim running the testbench and design

A waveform is created next (Figure 1), which includes all the signals for the unit under test (UUT). Individual waveforms are then modified directly on the screen by clicking on the signals to show the expected behavior, or by using the built-in pattern generator.

Next, HDL Bencher automatically exports a self-checking testbench. The testbench includes all stimulus (Figure 2), output assertions, timing constraints, and check routines needed to verify the operation of the design. The testbench is added to the ISE project, then auto-simulated through the Xilinx ISE software and ModelSim (Figure 3).

An advanced version of HDL Bencher is now available which automatically back-annotates the expected response into the waveform. If no expected response was specified, HDL Bencher back annotates the response obtained by ModelSim. Otherwise, expected and actual responses are compared, and discrepancies are highlighted.

Once your design is synthesized, its behavioral testbench may be incompatible with the resulting VHDL netlist generated during the post-route process. In this case, the resulting netlist uses std_logic_vector instead of integers. To make the synthesized netlist simulate, you would switch back to HDL Bencher, re-associate the waveform with the synthesized netlist, and re-export the testbench. Finally you would switch back to the ISE software and re-simulate.

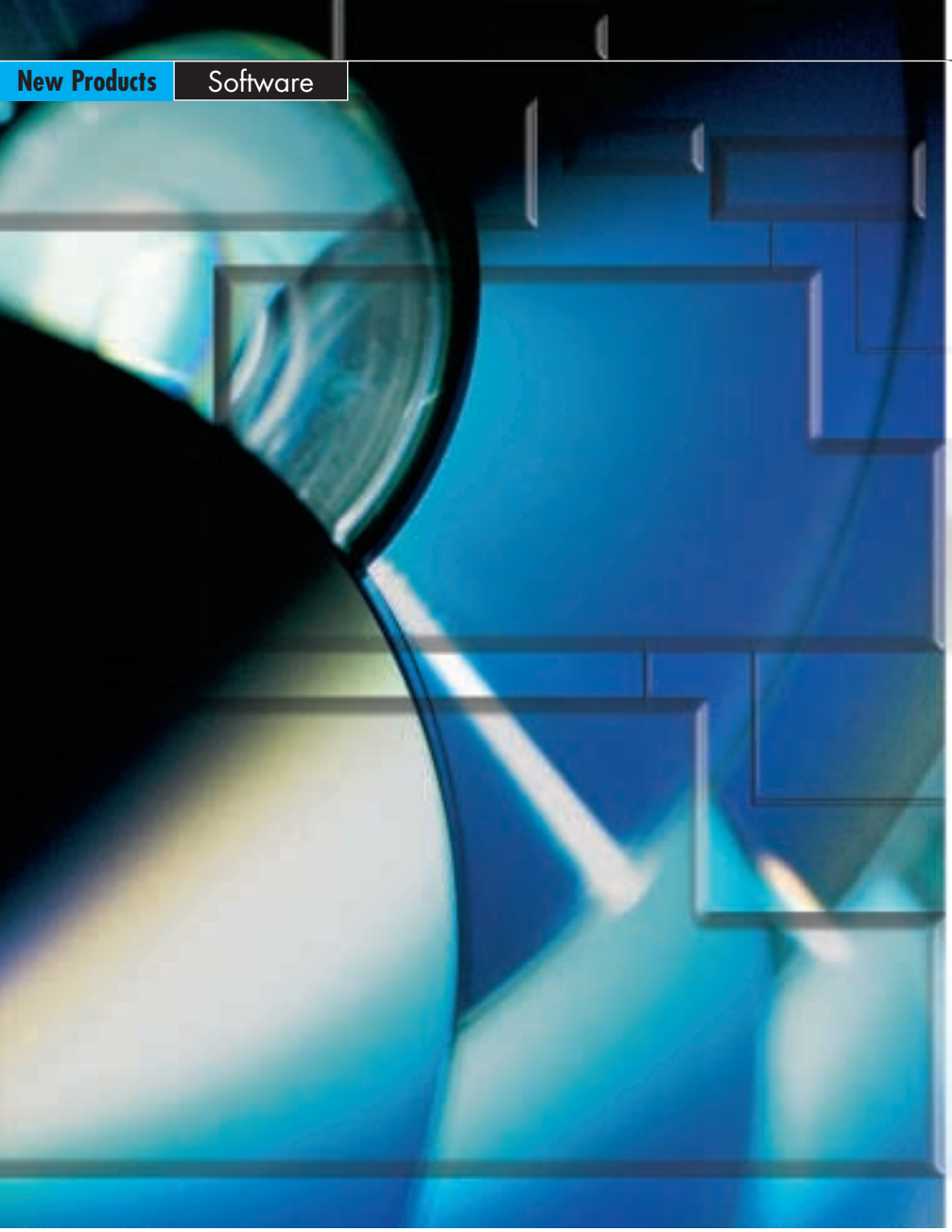
The Resulting Testbench

The exported testbench in this example is 183 lines of code, and took under 1 minute to create and simulate. The following portions of the testbench highlight some of the aspects of automatic testbench generation:

```
Automatically Commented
        — VHDL TestBench created by
        — Visual Software Solution's HDL
          Bencher 2.00
Libraries Extracted
    LIBRARY IEEE;
    USE IEEE.std_logic_1164.all;
Log File Created
    FILE RESULTS: TEXT IS OUT
"results.txt";
Components Instantiated
    COMPONENT counter
      PORT (
        CLK : in std_logic;
              :
        COUNT : inout integer RANGE 0 TO 7
Test Signals Defined
    SIGNAL CLK : std_logic;
              :
    SIGNAL COUNT : integer RANGE 0 TO 7;
Instantiates Unit Under Test
    UUT : counter  PORT MAP (
      CLK => CLK,
              :
      COUNT => COUNT
Clock Process Created
    BEGIN
      CLOCK_LOOP : LOOP
      CLK <= transport '0';
      WAIT FOR 10 ns;
      CLK <= transport '1';
Creates Check Procedures
    PROCEDURE CHECK_COUNT(
        NEXT_COUNT : INTEGER
Reports Errors In Expected Values
      IF (COUNT /= NEXT_COUNT) THEN
        write(TX_LOC,string'("Error at
time="));
Applies Input Stimulus
      RESET <= transport '1';
      CE <= transport '0';
Validates Timing
      WAIT FOR 100 ns; — Time=820 ns
Verifies Outputs
      CHECK_COUNT(7,820); — 7
Reports Success/Failure
    ASSERT (FALSE) REPORT
        "Simulation successful. No prob-
        lems detected. "
Draw Expected Behavior
```

Conclusion

With HDL Bencher you can verify the operation of VHDL and Verilog designs in minutes; no HDL scripting is needed. The resulting testbenches are self-checking, and are compatible with the Xilinx ISE software. HDL Bencher XE is available at no charge to all Xilinx customers, and is included with the ISE software or can be downloaded from www.xilinx.com (download the HDL Bencher "BackPack" from the WebPack section).



Guided Design Using BLIS

With Block Level Incremental Synthesis (BLIS), your design implementation times will improve dramatically.

by Karen Fidelak
Technical Marketing Engineer, Xilinx
karen.fidelak@xilinx.com

Incremental design changes (due to ECOs, specification changes, and repeated design iterations) can cause significant delays if you have to synthesize and place and route your entire design after each change. Ideally, your synthesis and place-and-route software tools should recognize where changes have been made in your overall design and recompile just those portions that have changed. That's what you get with BLIS, a unique synthesis and place-and-route capability, developed by Synopsys for Xilinx, that provides a guided synthesis methodology. Used in conjunction with Xilinx High-Level Floorplanning, BLIS provides the most robust incremental design capability ever offered.

BLIS, a part of the Synopsys FPGA Express/FPGA Compiler II v3.4 software (FE/FCII), is now available in the Xilinx ISE 3.2i development tools.

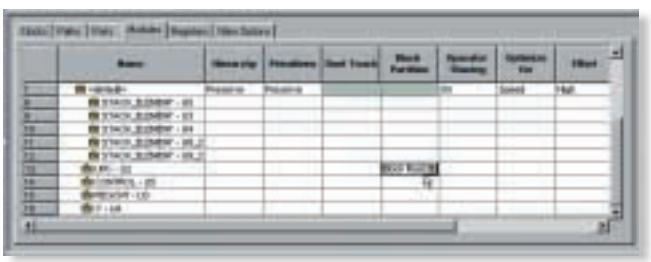


Figure 1 - Constraint Editor, specifying Block Roots

Block Level Incremental Synthesis

As you make design changes, BLIS recognizes "blocks" of the design which have been changed at the source, and intelligently synthesizes only those portions of the design. In this flow, a block is defined as a module/entity and any hierarchy tree beneath it. To enable BLIS, you choose blocks in your design that you want to denote as "Block Roots" through the FE/FCII Constraint Editor GUI or scripting language, as shown in Figure 1.

A Block Root is a block which is intelligently updated by FE/FCII in incremental synthesis runs, and has the following characteristics:

- A separate netlist is created by FE/FCII for each Block Root.
- Only those Block Roots whose corresponding source has been modified are re-synthesized.
- The Block Root has hard boundaries around it–no optimization occurs with neighboring modules.

The Advantages of BLIS

There are two main advantages to using this type of incremental flow.

- Runtime for both synthesis and place-and-route will be improved because only the modified portion of your design will be re-synthesized and re-netlisted. The remainder of the design will remain unchanged and the netlists for the unchanged portions of the design will not be rewritten. Because the netlists of the unchanged portions of the design remain untouched, you are assured that all net and instance names in that part of your design are identical to earlier runs.
- Timing predictability will be improved because the "Guide" function of the place-and-route tools, which relies on matched component names from run to run, will have a higher success rate.

Benchmarks

We compared the results of incremental design flows using BLIS against the more traditional methodology of re-synthesizing and re-routing the entire design. With the BLIS flow, incremental changes are made to a small number of design blocks (Block Roots). With the traditional flow incremental changes are made to the same design blocks, however they are not specified as Block Roots.

After our example design synthesis was completed, the design was placed and routed using the Guide feature of the Xilinx implementation tools, which allow you to specify an existing placed-and-routed design to be used as a "Guide" when implementing a design. The existing placed-and-routed design was used as a template when re-implementing the design. Any portions of the design which existed in both the "Guide" design and the new modified design (determined by matching net and component names) were placed in the same location in the new implementation as they were in the "Guide" design. New or changed logic was implemented around existing, "Guided" logic.

Runtime Improvements

Runtime improvements of up to 50% (with an average of 47%) were observed when using BLIS with Xilinx Guided Place-and-Route in an incremental design flow;

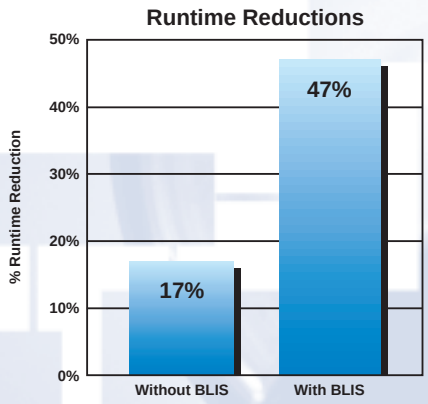


Figure 2 - BLIS runtime reductions

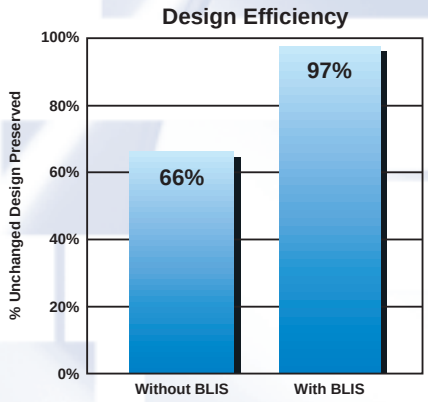


Figure 3 - BLIS design efficiency

Figure 2 shows averaged design results. Because FE/FCII does not re-elaborate or re-optimize unchanged blocks of the design, synthesis runtime was reduced.

Implementation runtime was improved due to increased design component matching during guided placement and increased signal matching during routing. Additionally, the synthesis tool does not rewrite the EDIF netlists for the unchanged blocks, further reducing runtime, because no file re-translation is needed.

Guide Improvements

When a design is placed-and-routed using the Guide feature, the success of the Guide can be determined by the "Design Components Matched" statistics available in the Place-and-Route report. The higher the percentage of matched components, the closer the incremental design is to the original results, leading to better predictability of timing and placement results.

When using the BLIS incremental design flow, Guide success rates reached levels of at least 95%, and averaged 97%. When BLIS was not used to guide the design, component and route matching was as low as 52%, as shown in Figure 3.

The improvements when using BLIS can be attributed to the increase in net and component name matches between the original placed-and-routed design and the incrementally modified version of the design. Because unchanged blocks of the design are not re-synthesized, the netlists are untouched and thus remain identical to the original version. (Even if there are no logic changes in the source, re-synthesizing a block can lead to net and component names being changed in the final netlist.)

Conclusion

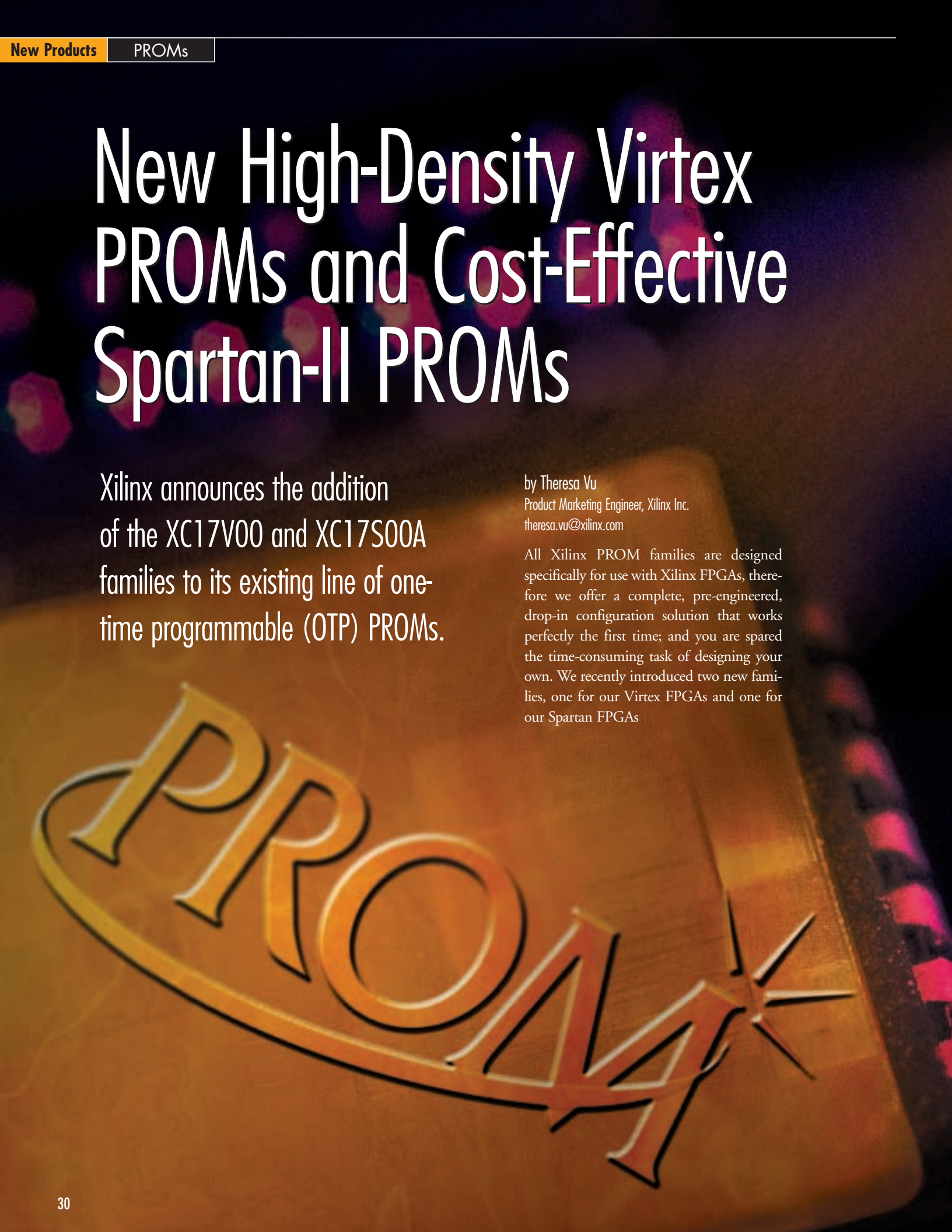
When utilizing FE/FCII Block Level Incremental Synthesis in a Xilinx guided design, runtimes as well as timing and placement consistency exhibit significant improvements over a more traditional design flow. These enhancements help you achieve a higher level of productivity by allowing you to synthesize and implement incremental design changes, with a significantly reduced runtime, while preserving the unchanged portions of your design. This new design flexibility allows you to realize the productivity necessary to complete large or small FPGA designs faster.

New High-Density Virtex PROMs and Cost-Effective Spartan-II PROMs

Xilinx announces the addition of the XC17V00 and XC17S00A families to its existing line of one-time programmable (OTP) PROMs.

by Theresa Vu
Product Marketing Engineer, Xilinx Inc.
theresa.vu@xilinx.com

All Xilinx PROM families are designed specifically for use with Xilinx FPGAs, therefore we offer a complete, pre-engineered, drop-in configuration solution that works perfectly the first time; and you are spared the time-consuming task of designing your own. We recently introduced two new families, one for our Virtex FPGAs and one for our Spartan FPGAs



Virtex Configuration PROMs

Our low-cost XC17V00 PROMs support Virtex and Virtex-E FPGAs, up to 3.2 million system gates, and are offered in 1-Mb to 16-Mb densities. The available packages are shown in Table 1.

The 16-Mb 17V16 PROM, a four-fold increase in maximum bit density, extends the Xilinx leadership in configuration memories and provides a one-chip configuration solution for our entire line of Virtex FPGAs.

Key Features

The XC17V00 serial/parallel PROM family is based on our proven, OTP architecture that provides a stable, low-cost, highly-reliable one-chip configuration solution with the following features:

- 1-Mb to 16-Mb densities.
- Simple, fast, serial FPGA interface that requires only one user I/O pin.
- Parallel configuration up to 264 Mbps (17V16 and 17V08 only).
- Available in SOIC, VOIC, VQFP, and PLCC packages.
- Low-power CMOS floating gate process.
- Programming support by leading programmer manufacturers.
- Cascadable for storing longer or multiple bitstreams.

Device	Density	8-pin VOIC	20-pin SOIC	20-pin PLCC	44-pin PLCC	44-pin VQFP
XC17V01	1.6Mb	✔	✔	✔		
XC17V02	2Mb			✔	✔	✔
XC17V04	4Mb			✔	✔	✔
XC17V08	8Mb				✔	✔
XC17V16	16Mb				✔	✔

Table 1 - Virtex PROM packages

XCV300E	✔ ✔ ✔ ✔ ✔ ✔ ✔ ✔
XCV400E	✔ ✔ ✔ ✔ ✔ ✔
XCV405E	✔ ✔ ✔ ✔
XCV600E	✔ ✔ ✔ ✔
XCV812E	✔ ✔
XCV1000E	✔ ✔
XCV1600E	✔ ✔
XCV2000E	✔
XCV2600E	✔
XCV3200E	✔

Table 2 - Number of Virtex-E FPGAs configurable by one 16Mb PROM

FPGA	PROM Solution	8-pin PDIP	8-pin VOIC	20-pin SOIC	44-pin VQFP
XC2S15	XC17S15A	✔	✔	✔	
XC2S30	XC17S30A	✔	✔	✔	
XC2S50	XC17S50A	✔	✔	✔	
XC2S100	XC17S100A	✔	✔	✔	
XC2S150	XC17S150A	✔	✔	✔	
XC2S200	XC17S200A	✔	✔		✔

Table 3 - Spartan-II PROM packages and device compatibility

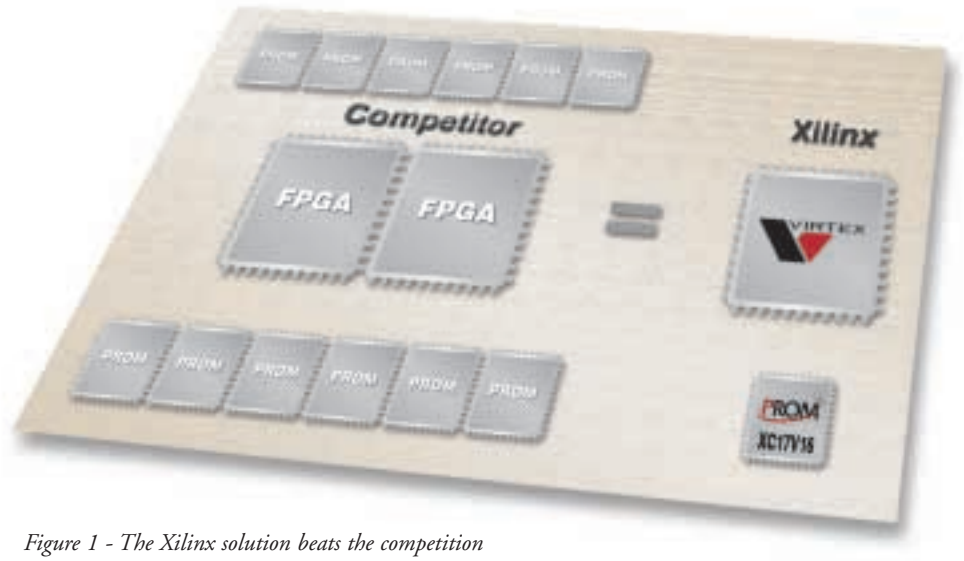


Figure 1 - The Xilinx solution beats the competition

The Most Cost-effective Solution

The new XC17V00 family also offers significant savings in board space, design time, and cost. Using one 17V16 to configure the new 3.2 million system-gate Virtex XCV3200E FPGA requires less than one fourth the board space of any previous Xilinx configuration PROM solution. To get the equivalent functionality from our nearest competitor would require 14 chips and more than 2x the board space, as illustrated in Figure 1.

Xilinx process expertise has also allowed us to use smaller packages, further reducing the need for board space.

Configuration of Multiple FPGAs

The XC17V16 can also be used to configure multiple, daisy-chained FPGAs. This allows you to store configuration data for up to eight FPGAs in a single PROM, as illustrated in Table 2.

Spartan-II Configuration PROMs

Our XC17S00A PROM Family provides a high-performance, low-cost configuration solution, optimized for use with Spartan-II FPGAs. This family offers a dedicated PROM for each gate density in the Spartan-II family for ease-of-selection and guaranteed compatibility, as shown in Table 3. This family also offers extended availability of the smallest package offered by Xilinx, the 8-pin VOIC.

Key Features

- Simple, fast, serial Spartan FPGA interface that requires only one user I/O pin.
- Available in DIP, VOIC, SOIC, and VQFP packages.
- Advanced, low-cost CMOS process.
- Programming support by leading programmer manufacturers.

Conclusion

With the new XC17V00 and XC17S00A PROMS, there is no easier, faster, or less expensive way to configure Xilinx FPGAs. For more information see: www.xilinx.com.

Create Efficient FIR Filters Using Virtex and Spartan FPGAs

The Virtex and Spartan-II LUTs, configured as shift registers combined with Xilinx True Dual-Port™ RAM, give you a very compact, flexible, and area-efficient FIR filter design platform.

by Rotem Gazit
Design Engineer, MystiCom LTD.
rotemg@mysticom.com

A Finite Impulse Response (FIR) filter works by multiplying a vector of the most recent N data samples by a vector of coefficients and summing the elements of the resulting vector. In every cycle the filter receives a new sample of data and shifts out the oldest sample. FIR filters are very common in FPGA-based Digital Signal Processing applications.

The design concept described here is suitable for systems with relatively low input rates (0.5 to 8 MHz), which require a FIR filter implementation with hundreds of taps; this is common in modem and demodulation applications.

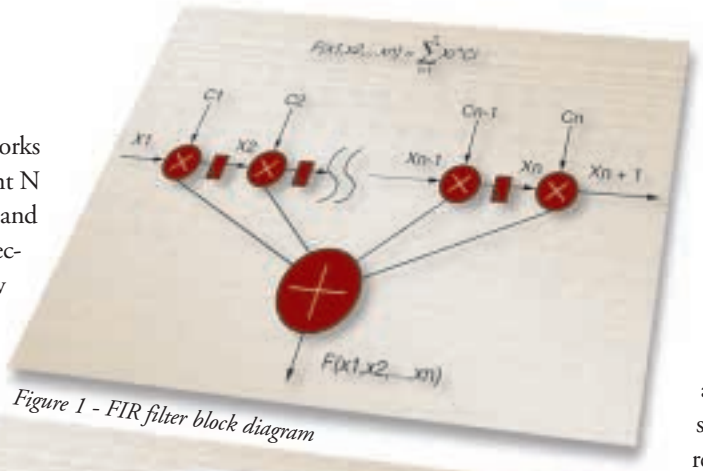


Figure 1 - FIR filter block diagram

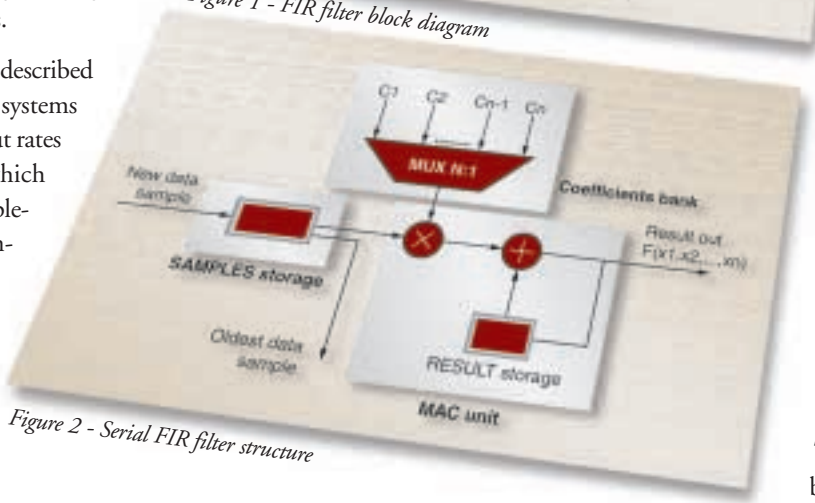


Figure 2 - Serial FIR filter structure

FIR Filter Design Concepts

By examining the FIR block diagram in Figure 1, you can see that if the filter is implemented in a straight forward manner, a multiplier will be required for every filter tap (N multipliers for an N-tap filter). In addition, an adder with N inputs will be needed to sum all multipliers outputs. However, if the data input rate is slower than the performance capability of the FPGA, the filter can be implemented much more efficiently.

Serial FIR Filters

Assuming that the performance capability of the FPGA is M times faster than the data input rate, we will examine the case where M is ≥ N (where N is the required number of filter taps).

To implement a serial N-tap filter uses only one multiplier, a 2-input adder, and storage for the partial results and the filter input samples. The input sample storage holds the last N input samples. For every new sample entering the filter, N multiply operations will be performed, each multiplying the filter coefficient by the respective input sample.

The result of each multiply operation is added to the partial result storage to produce a new partial result. This newly calculated partial result is then saved in the partial result storage by replacing the previous partial result. After N such multiply and add operations, the partial result storage content is driven out of the filter. The partial result storage content is then cleared to begin processing a new data sample. A block diagram of serial FIR filter structure is shown in Figure 2.

The hardware responsible for the combination of multiplying, adding, and storing is called a MAC (Multiply Accumulate) unit. Due to the serial nature of the filter, the MAC will operate on M taps of the filter. In the case where N is greater than M, several serial filters can be chained together. The oldest data sample leaving the first filter in the chain is used as the new data sample in the next filter, and so on. The results of all the chain filters must be added together.

Implementing a Serial FIR Filter

You can implement Serial FIR filters very efficiently in Virtex and Spartan-II devices. The design can be divided into three separate units: the coefficients bank, the MAC unit, and the input sample storage.

Coefficients Bank

The Virtex block RAM can be used to hold the filter coefficients. No multiplexer is needed; all you need is a simple cyclic counter used as an address generator. In systems where a host DSP or an adaptation mechanism is present, the block RAM can be configured as a dual port RAM, enabling the coefficients to be dynamically changed during the normal filter operation.

MAC Unit

The MAC unit consists of an adder, a multiplier, and result storage. Careful design of the adder and multiplier is very important for area efficiency.

Theoretically, the result of a 2^x tap filter, which has 2^y bits on every input data and 2^z bits on every coefficient, will be $2^{(x+y+z)}$ bits wide. In real world applications however, the number of bits in the result is usually much smaller because the least significant bits of the result are usually ignored in the final result, after processing. It is very important to throw away those unnecessary bits as early as possible in the data processing (in the MAC multiplier and adder).

An example MAC implementation is shown in Figure 3.

Input Samples Storage Unit

The input data storage unit can be implemented very efficiently in Virtex devices using the LUTs as shift registers. Each MAC, operating on M taps of the filter, requires an input data storage of M-1 stage delay line. During the first M-1 cycles, the delay line output is driven both to the MAC and back to the delay line input.

In the Mth cycle, the delay line output is driven only to the MAC, and the new input data sample enters the delay line. If several filters are chained together, then the

Throwing away bits in the MAC can sometimes lead to different results than you get from throwing away the bits from the final result; a thorough discussion of the effect of such an operation on the filter performance is beyond the scope of this article.

```
////////////////////////////////////////////////////////////////////
// Name:mac
//_____________________________
// Target device:
//_____________________________
// Module description:
//_____________________________
// MAC of 16 bit coefficient by 5 bit input data_sample.
// the result is 22 bits wide
//
// Parent:
//_____________________________
// filter_top
//
// childrens:
//_____________________________
//mac_adder.v ,mac_multiplier.
////////////////////////////////////////////////////////////////////

module mac (coefficient,data_sample,rst,clk,enable,new_data,out);

input [15:0] coefficient;     //filter coefficient coming from coefficient storage
input [4:0] data_sample;  //filter data_samplescoming from samples storage
input clk,enable,rst;
input new_data;           //indicates a new data sample. new_data goes high for one cycle
                          //every 64 clocks, 3 clocks after the new data arrives
                          //Because of MAC pipeline.

output [21:0] out;       // MAC output.
reg [21:0]  out;         // MAC output changes whenever a new data is being processed.

wire [16:0] mul_out;    // mac_multiplier output.
wire [21:0] add_out;    // mac_adder output.

reg [21:0] add_out_d;    // sampled mac_adder output.
reg [16:0] mul_out_d;    // sampled mac_multiplier output.


mac_multiplier mac_multiplier(.coefficient(coefficient),.data_sample(data_sample),.mul_out(mul_out) );

always @(posedge clk or negedge rst) // sample the multiplier output
begin                 // to improve timing
   if (!rst)
   mul_out_d <= #2 17'b0;
   else
   mul_out_d <= #2 mul_out;
end

mac_adder mac_adder(.adder_out(add_out),.adder_in_0(mul_out_d),.adder_in_1(add_out_d) );

always @(posedge clk or negedge rst) // sample the adder output
begin                                          // this is the "RESULT storege"
   if (!rst)
   add_out_d <= #2 22'b0;
   else
      if (new_data)   // clear accumulator for new data processing
         add_out_d <= #2 22'b0;
   else
         add_out_d <= #2 add_out;
end


always @(posedge clk or negedge rst) // MAC output changes only when a new data arrives
begin
   if (!rst)
   out <= #2 22'b0;
   else if (enable & new_data)
         out <= #2 add_out;
end

endmodule
```

Figure 3 - An example MAC implementation

```
////////////////////////////////////////////////////////////////////
// Name:delay_line
//-----------------------------------
// Target device:
//-----------------------------------
// Module description:
//-----------------------------------
// delay line of 63 delays x 5 bit.
// the oldest sample is delayed for 64-clock cycle before driven to the next
// delay line in the chain
//
// Parent:
//-----------------------------------
// filter_top
//
// Childrens:
//-----------------------------------
//shift5x63.v shift63.v
////////////////////////////////////////////////////////////////////

module delay_line (new_data_sample,clk,rst,enable, new_data, mac_data,next_mac_data);

input [4:0] new_data_sample;        // new_data sample
input clk,enable,rst;
input new_data;                     // new_data is active every 64 cycle for one cycle ->
                                    // SR mux control (input from it's output OR new_data_sample)

output [4:0] mac_data;              // data for MAC
output [4:0] next_mac_data;         // data for next MAC in chain
reg  [4:0] next_mac_data;           // Hold next_mac_data back for one MAC cycle
                                    // (64 clock cycles)

wire [4:0] mac_data;

shift5x63 shift5x63(.din(new_data ? new_data_sample : mac_data) ,
          .clk(clk),.enable(enable),
          .dout(mac_data)
         );

// Hold next_mac_data back for one MAC cycle (64 clock cycles)
always @(posedge clk or negedge rst)
begin
   if (!rst)
   next_mac_data <= 5'b0;
   else if (new_data & enable)
   next_mac_data <= mac_data;
end

endmodule

module shift5x63 (din, clk,enable, dout);
input [4:0] din;
input clk,enable;
output [4:0] dout;
shift63 bit0(.din(din[0]), .clk(clk),.enable(enable), .dout(dout[0]));
shift63 bit1(.din(din[1]), .clk(clk),.enable(enable), .dout(dout[1]));
shift63 bit2(.din(din[2]), .clk(clk),.enable(enable), .dout(dout[2]));
shift63 bit3(.din(din[3]), .clk(clk),.enable(enable), .dout(dout[3]));
shift63 bit4(.din(din[4]), .clk(clk),.enable(enable), .dout(dout[4]));
endmodule

module shift63 (din, clk,enable, dout);
input din, clk,enable;
output dout;                        //Synplify automatically infers
                                    //SRL16 for shift register with no reset
reg [62:0] shifter;
always @(posedge clk)
begin
   if (enable)
   begin
   shifter[62:0] <= {shifter[61:0],din} ;
   end
end
assign dout = shifter[62] ;
endmodule
```

Figure 4 - An example of a LUT SRL16-based delay line implementation

delay line output needs to be held for M cycles before it is driven as an input to the next filter in the chain. Sometimes (depending on the available resources inside the device) it is better to implement the delay line using block RAM configured is a simple FIFO.

An example of a LUT SRL16-based delay line implementation is shown in Figure 4. A diagram of the complete serial FIR filter is shown in Figure 5.

Conclusion

FIR filters with many hundreds of taps can be implemented easily even in the smallest members of the Virtex and Spartan-II FPGA families. By taking advantage of the Virtex and Spartan-II architecture, you can implement FIR filters very efficiently.

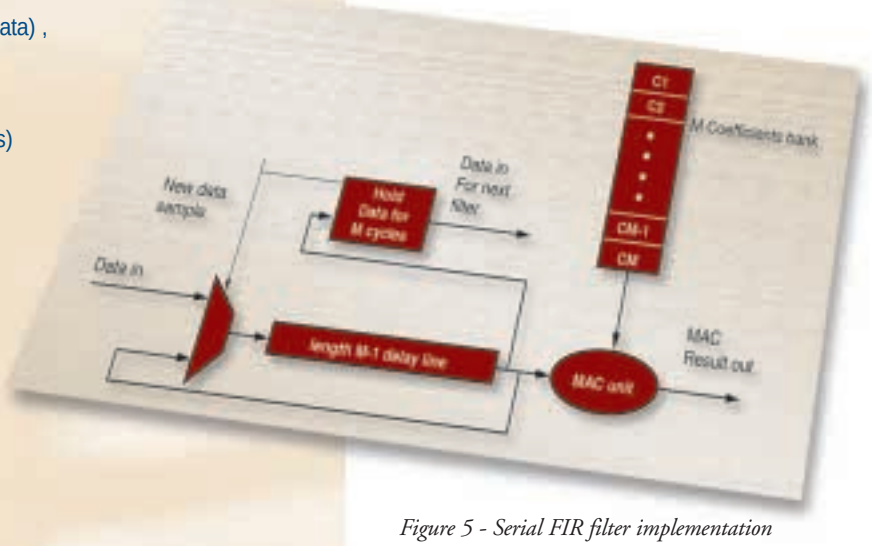


Figure 5 - Serial FIR filter implementation

About MystiCom

Founded in 1997, MystiCom is dedicated to providing DSP and mixed-signal VLSI cores for high-speed communications. The company's first product line implements the physical layer (PHY) for Local Area Networks (LANs) using Fast Ethernet and Gigabit Ethernet protocols. MystiCom is headquartered in Netanya, Israel, and has marketing and customer support offices in Mountain View, Calif. Additional information can be found at www.mysticom.com.

LogiCORE PCI Module Is a Key Element in Voice over IP Applications

Silicon & Software Systems provides an elegant solution to Nortel Networks, using a Xilinx LogiCORE PCI module implemented in a Spartan-XL FPGA.

by Dara Hurley
Director, Hardware Systems Division
Silicon & Software Systems
dara.hurley@s3group.com

Voice over Internet Protocol (VoIP) offers companies and consumers enormous potential cost savings compared to traditional switched telephone networks. This emerging technology is enabling low-cost international telephony and remote teleworking (also known as telecommuting).

Nortel Networks, a pioneer in VoIP, has employed a LogiCORE™ PCI module in a Xilinx Spartan™-XL FPGA to improve service. The system architecture is shown in Figure 1. The system is based on the popular PC architecture. The network adapter receives data (containing compressed speech) from the Internet and passes the data to the DSP compression/decompression engine via the PCI bus. The digital speech is then routed back through a time-switched FPGA to the telecommunications network.

In the other direction, digital speech is routed from the telecommunications network via two Xilinx FPGAs to the DSP engine. An x86 CPU controls the system. The DSP engine uses a system memory, which is connected to the CPU local bus. The CPU provides IP packet processing, and data is transferred from the system memory to the network adapter using DMA on the PCI bus.

"Critical to the system performance is the PCI implementation," said Eugene Garvin development manager of Nortel Networks "the DSP bus operates at a much slower speed than the PCI bus, so the realization of the bus must be optimized."

Initially, Nortel used a simple approach: Data transfer was routed through the CPU and

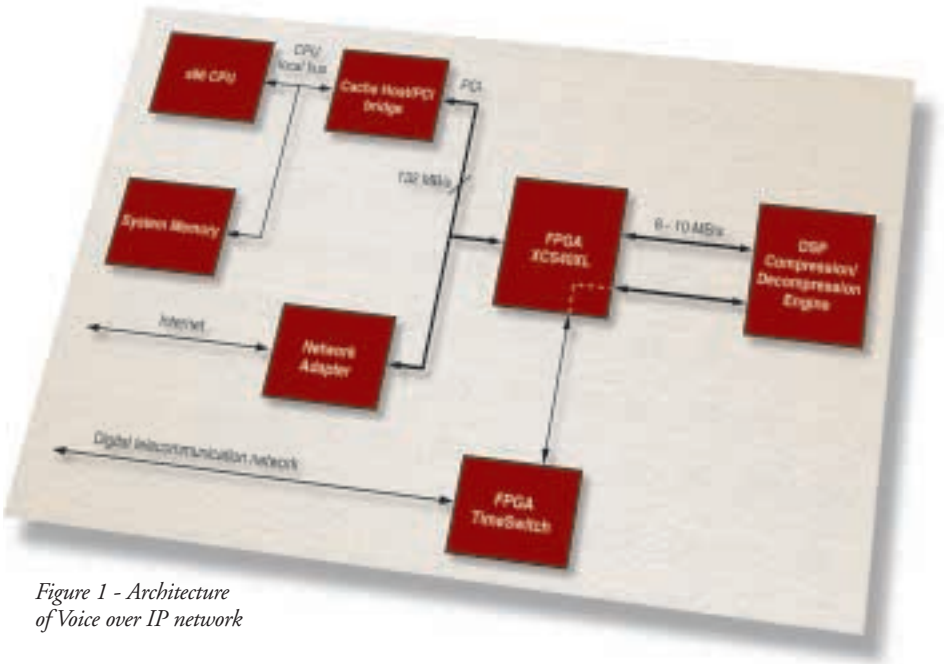


Figure 1 - Architecture of Voice over IP network

wait states were inserted in the PCI transactions to compensate for the different data rates. This approach, however, consumed too much of the PCI throughput, Garvin stated.

This is where Silicon & Software Systems (S3) stepped in and provided a more elegant solution. Silicon & Software Systems designed a DMA controller and FIFO data buffer, and integrated these along with a Xilinx LogiCORE PCI module into a Spartan-XL FPGA device (XCS40XL). The device also contained an interface between the time-division multiplexed (TDM) speech data from the telecommunications network and the data presented to the DSP compression/decompression engine. The designers used Spartan SelectRAM™ memory to create the dual-ported RAM-based FIFO buffer.

"Use of DMA and data buffering over the PCI bus has freed up the processor to do other necessary tasks," Garvin reported. "It also makes better use of the available PCI throughput, because it uses zero-wait-state burst transfers instead of one-word-delayed transactions."

About Silicon & Software Systems

Based in Dublin, Ireland, Silicon & Software Systems (S3) is a world-class electronic design service company. Since its inception 1986, Silicon & Software Systems has experienced rapid growth, penetrating markets throughout Europe, the US, the Middle East, and Eastern Europe. In the 1999 Data Quest survey, Silicon & Software Systems emerged as one of Europe's leading electronic design facilities. The company employs over 300 design engineers specializing in ICs, software, and hardware systems for the communication infrastructure, digital consumer, and wireless markets. To learn more about Silicon & Software Systems, go to: www.s3group.com.

Creating Finite State Machines

Using True Dual-Port Fully Synchronous SelectRAM Blocks

Create very dense, high-performance, highly efficient designs that require no logic resources.

by Edgard Garcia
Senior Engineer, Multi Video Designs
edgard.garcia@mvd-fpga.com

The latest Virtex, Virtex-E, and Spartan-II FPGA families offer a broad range of unique features, including block RAM, that give you dramatic speed and density improvements. The dedicated RAM blocks allow you to build fast and dense bidirectional data buffers and FIFOs, with built-in data width conversion. This RAM can also be used to implement very fast and efficient sequencers and Finite State Machines (FSMs), which frees your logic gates for other tasks.

A well known approach to building sequencers consists of a ROM-based design with output registers. The same method can apply to the Virtex 4K-bit block SelectRAM™ which incorporates output registers. You can use a single 4K-bit RAM block as a 512 x 8 clocked ROM, to implement a very fast FSM working at more than 150 MHz, and it uses no CLBS. You can implement the following, for example:

- 16 states + 4 additional outputs and 5 inputs + Enable and Synchronous Reset.
- 32 states + 3 additional outputs and 4 inputs + Enable and Synchronous Reset.

Design Example

The following example shows how to implement FSMs or sequencers with a single 4K-bit block SelectRAM. The same method can easily be expanded to more complex designs that could require two or more blocks.

Synchronous FSMs and sequencers have some important characteristics in common:

- They are clocked by a single clock.
- A feedback path allows you to (partially) define what the next step will be.
- They may need a clock enable to suspend the operations.
- They must have a reset to go back to a predefined state.

Figure 1 shows a typical FSM or sequencer logic diagram.

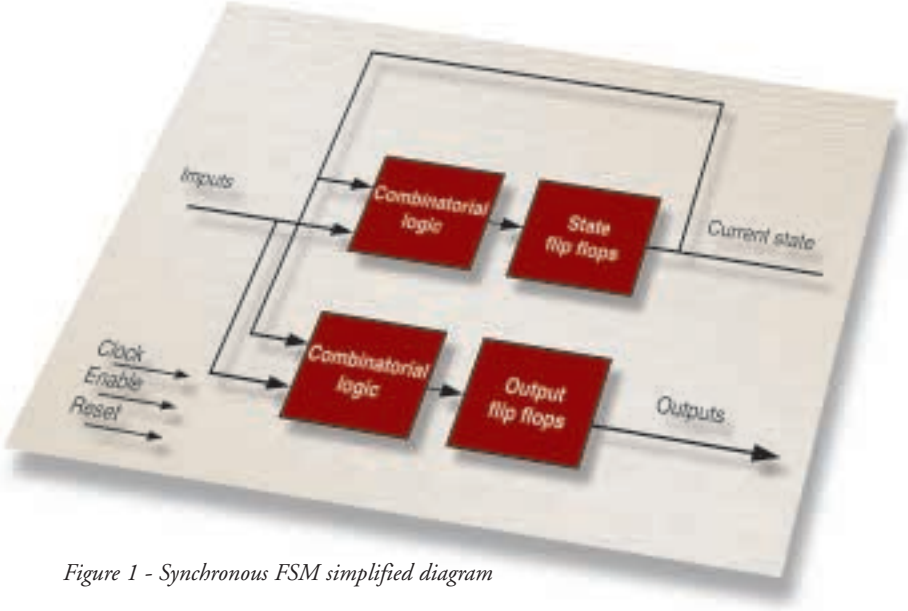


Figure 1 - Synchronous FSM simplified diagram

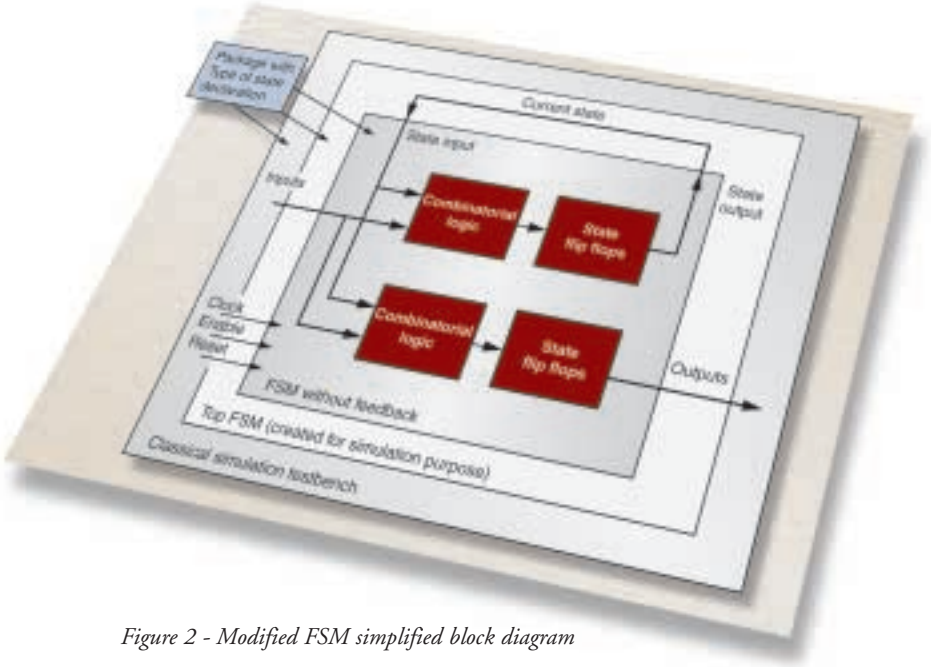


Figure 2 - Modified FSM simplified block diagram

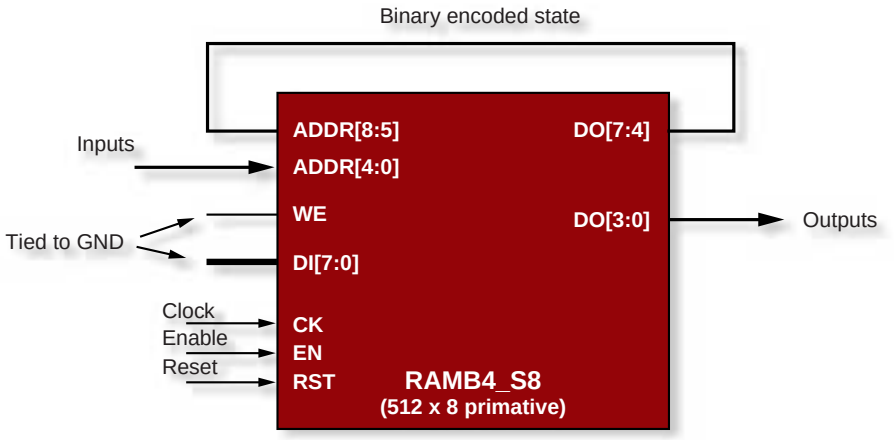


Figure 3 - Using fully synchronous RAM blocks for FSM implementation

Simulation

Converting the FSM behavior to a truth table can be a very tedious and time consuming task; debugging and modifying the design can turn into a nightmare.

The method used here takes advantage of the modern design entry, simulation, synthesis, and implementation tools, combining their complementary respective power. The goal is to use a VHDL simulator to automatically generate the constraint file for initialization of the block SelectRAM used as ROM.

One of the problems encountered when simulating a design with VHDL, is that an output can't be forced to any logic level. An easy way to avoid this inconvenience consists of breaking the feedback loop, which allows you to enter patterns into the inputs, including current and illegal states. Figure 2 illustrates one way of designing the FSM VHDL code so it can be simulated more easily. A top-level file can provide the feedback for a classical simulation of the FSM.

Optimization

If the number of inputs (including binary encoding states feedback) is nine or less, and the number of outputs (including binary encoding states) is eight or less, a single 4K-bit block SelectRAM can be applied by using structural VHDL with the scheme shown in Figure 3.

RAM Initialization

By initializing the contents of the memory with the appropriate values, the behavior of any synchronous FSM can be reproduced. The initialization of the RAM block is done by an NCF (constraint) file that will be used by synthesis and Xilinx implementation tools. A very easy way to initialize the memory with the correct values is to make an automatic generation of the NCF file, by using a VHDL simulator and another testbench.

Consider the behavioral VHDL code of the FSM without feedback. A simple 9-bit pseudo counter (generated by a testbench) can provide all the 512 possible states of the inputs, including illegal states. Each associated result (state output and FSM outputs) can

thus be converted to a text file obeying the NCF file format. See Figure 4.

Resources Required

Table 1 summarizes some examples of typical FSMs or sequencers in terms of performance and logic resources. All these designs can be implemented with a single RAM block, but the same method can easily expanded to more complex functions, providing similar improvement. As you can see, a RAM-based FSM is much faster and uses no logic resources.

True Dual-Port RAM Advantages

Block SelectRAM provides true dual port capability; a single location can be read at the same time by the two ports. Therefore, a single block can be used to implement two identical synchronous FSMs, with separate inputs, synchronous reset, and clock enable; you can also implement separate clocks, if needed. Figure 5 shows the architecture of a dual FSM using a single dual-port block SelectRAM. Table 2 shows a dual FSM implementation comparison.

Conclusion

The Virtex architecture provides powerful features and flexibility, allowing you to create very dense and high performance designs by taking advantage of the innovative features such as block SelectRAM. By combining the power of the Xilinx architecture and implementation tools, with the associated VHDL synthesizers and simulators, your design productivity is greatly improved, and complex designs can be easily implemented or modified.

For more information, please e-mail: Edgard.garcia@mvd-fpga.com

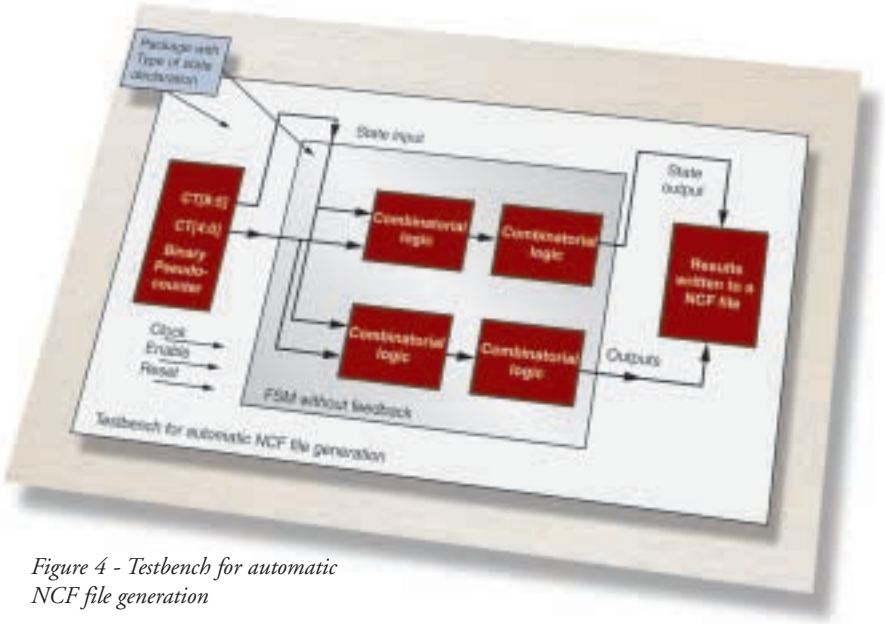


Figure 4 - Testbench for automatic NCF file generation

Implementation mode	Number of FFs	Number of Slices	RAM Blocks	Speed*
One Hot Encoding	20/35	30...60	–	80...110Mhz
Binary Encoding	8/8	25...50	–	70...100Mhz
SelectRAM block (binary encoding)	0	0	1	150 Mhz

• Results for a single 16 (or 32) state synchronous FSM with Enable, synchronous Reset, 5 (4) inputs and 4 (3) outputs.

Table 1 - Single FSM implementation comparisons

Implementation mode	Number of FFs	Number of Slices	RAM Blocks	Speed*
One Hot Encoding	40/70	60...120	–	80...110Mhz
Binary Encoding	16/16	50...100	–	70...100Mhz
SelectRAM block (binary encoding)	0	0	1	140 Mhz

*Dual 16 (or 32) states synchronous FSM with Enable, synchronous Reset, 5 (4) inputs and 4 (3) outputs.

Table 2 - Dual FSM implementation comparisons

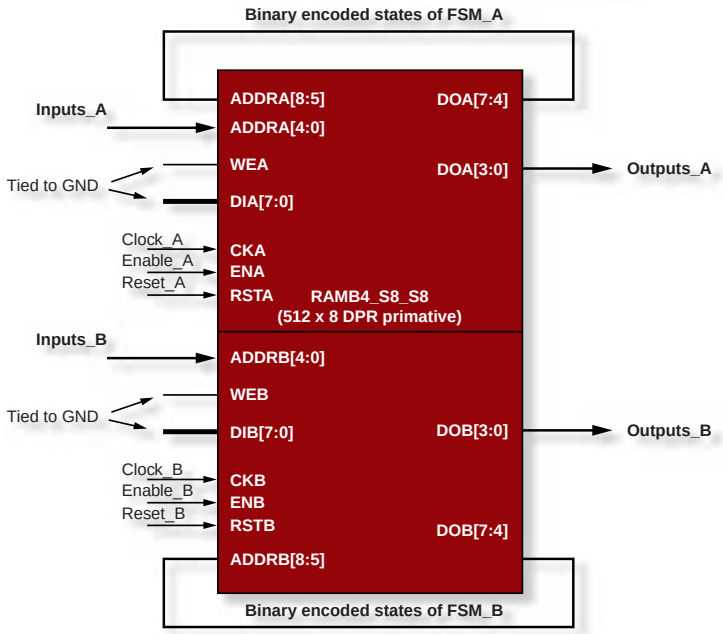


Figure 5 - Dual FSM implementation

About Multi Video Designs

MVD is a design and training center, specializing in Xilinx FPGA/CPLD designs and Hardware Description Languages. Consulting services and on site classes are offered in France, neighboring countries, and South America, in French, Spanish, and Portuguese. More information on our activity as well as the source code of some examples are available on our website, at: www.mvdfpga.com/training/VHDL_examples.htm

Design a Low-Power SMBus System Using CoolRunner™ CPLDs

The ultra low-power CoolRunner CPLD is the ideal choice for SMBus systems, because you can easily configure it to suit your specific needs.

by John Hubbard
Applications Engineer, Xilinx, Inc.
john.hubbard@xilinx.com

Low-power devices use the System Management Bus (SMBus) protocol to communicate with components and peripherals. SMBus is a compatible derivative of the I²C two-wire serial bus protocol and can therefore reside in the same device. In addition to the I²C features, SMBus enhances systems designed for power management tasks. Because SMBus is often used in handheld devices containing "smart" batteries, CoolRunner CPLDs are the perfect solution for implementing the low-power SMBus components designed into these batteries.

A system that uses the SMBus protocol can pass information between components without the need for individual control lines. This passed information can contain manufacturer information, model numbers, part numbers, system status, control parameters, errors, and so on. SMBus is so flexible you can even add or remove components during system operation. It also has the ability to determine arbitration in multi-master systems, and if a newly inserted device has the ability to check packets of data for errors.

Functionality

The SMBus implementation for the CoolRunner CPLD consists of a microcontroller or microprocessor interface and an SMBus master/slave controller as shown in Figure 1. It implements the following features:

- Microcontroller interface.
- Master or slave operation.
- Multi-master operation.
- Host operation.
- Automatic mode switching between master and slave.
- Calling address identification.
- START and STOP signal generation and detection.
- Repeated START signal generation.
- Acknowledge bit generation and detection.
- Bus busy detection.
- From 10 to 100 kHz operation.
- Optional signal SMBSUS# for system suspend mode.
- Optional signal SMBALERT# for slave interrupt request.
- Packet Error Code (PEC) using 8th polynomial Cyclic Redundancy Check (CRC-8) methods.
- Automatic determination of PEC capable devices.
- Compliant with System Management Bus Specification Rev. 1.1. (Note: the new SMBUS 2.0 Specification was posted Aug. 3. See http://www.smbus.org/index.html.)
- Software selectable SMBus acknowledge bit.
- Arbitration.

A more detailed description of the SMBus controller is shown in Figure 2. You can easily modify the microcontroller interface to adapt the design to any microcontroller of your choice.

This design was created in VHDL using Xilinx WebPACK™ ISE (Integrated Synthesis Environment). It was verified using ModelSim™ XE (Xilinx Edition) simulation software.

Conclusion

You can get started with your own CoolRunner CPLD SMBus design by visiting the Xilinx website at www.xilinx.com/xapp/xapp353.pdf. Download (for free) the following:

- Complete detailed application notes.
- Complete VHDL source code.
- VHDL test benches.

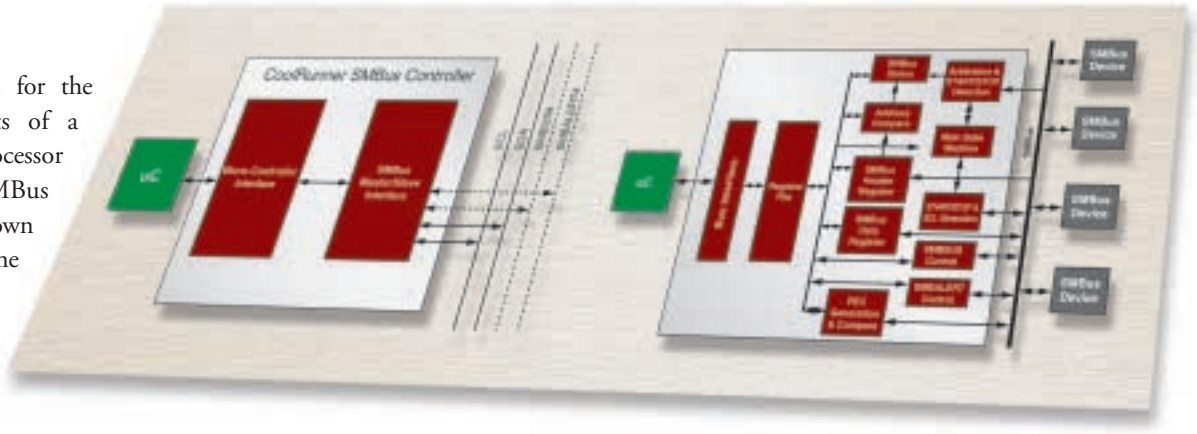


Figure 1 - Basic SMBus controller functions

Figure 2 - SMBus controller detailed block diagram

CoolRunner Power-Saving Tips and Tricks

These techniques can lower your CoolRunner power consumption by 40%.

by Frank Wirtz
Staff Applications Engineer, Xilinx Inc.
frankw@xilinx.com

With the advent of Fast Zero Power technology and CoolRunner CPLDs, you can now create portable, high-performance, low-power, programmable devices, effortlessly. And, with some additional effort, you can reduce your power consumption by as much as 40%. However, to accomplish ultra low power reductions, you must first understand the mechanics of CPLD logic generation.

Xilinx has published a new application note, "Low Power Tips for CPLD Design" (XAPP346), that describes design techniques you can use to further reduce power consumption in CoolRunner CPLDs, which are already the lowest power-consuming CPLDs in the world. Here are some highlights from that application note.

Tips and Tricks for Reducing Power

The CoolRunner XPLA architecture gives you a flexible logic allocation tool that allows you to decrease power consumption by placing your logic in optimum locations. To use this tool, you need to understand the basic architecture of the device, and you need to know how the fixed geometry of the device determines both the speed-sensitive paths and the power-sensitive paths.

The following design implementation techniques are just a sampling of the information you can use to slash power consumption to a minimum:

Terminate!

You must properly terminate all inputs to a CMOS buffer. A single floating pin can result in an increase of quiescent current by 13mA. Slow input transitions will also cause unnecessary power use. Test data shows that input buffer power consump-

tion doubles if input rise time increases from 800ps to 5ns per input.

Congregate!

You can see how your design is implemented by reviewing your fitter report, and then adjust the fit to constrain your high frequency signals to a single logic block. This will decrease the distribution of high-speed nets and further decrease power consumption.

Modulate!

The application note details special clock considerations and explains how asynchronous clocking can provide low power benefits. Typically, asynchronous clocking increases power consumption. Modulation in this instance refers to only applying a clock signal to a register when it is required. Many designs have registers that infrequently change state, yet the clock signal is continually present and applied to the register. While asynchronous design techniques are usually discouraged, they do provide designers with additional flexibility when low power (or sometimes high speed) characteristics are required.

As an example of this technique, consider a counter circuit. In the case of a binary counter, not all of the registers change state on each significant clock edge. Designers can use a high speed clock for the LSBs of the counter, and then use a prescaled clock for the higher order bits, so the total amount of power required by the clock buffers is decreased.

Mixed Voltage Interfacing

When interfacing devices that have different VCC levels, consider the impact caused by under driving a CMOS input. Because a CMOS input buffer is comprised of at least two primary transistors, a P-channel pull up and an N-channel pull down, there exists a region of input voltage where both transistors are slightly on, and current flows from VCC to GND through these buffers. This causes power to be wasted, and since the output of this buffer may also be in the linear region, it can cause problems because other internal devices depend upon the output voltage of the buffer for driving their inputs.

In some cases, mixed voltage interfacing is necessary. Slight modifications to differing VCCs can drastically reduce power consumption in these instances. For example, the XPLA3 devices may be powered at 3.3V +10%, and 5V devices may be powered at 5V -10%. This changes the differential between Voh and Vih by 800mV per input, and will significantly reduce wasted power. However, examine the data sheet to ensure safe and reliable operating conditions.

Default System Conditions

Attention to default system operating conditions may provide an insight into ways you can further decrease power consumption. As an example, a CPLD may be interfaced to a CMOS microcontroller with programmable (polarity sensitive) interrupts. If it is necessary to interface a 3.3V CPLD to a 5V interrupt, system power can be saved by programming the microcontroller interrupt such that the system operates with the interrupt level normally low. This decreases the amount of time that the interrupt is active (high) which will reduce the overall amount of power consumed when under driving a CMOS input.

The Effects of Implementation Style

Implementation style affects power consumption. For example, consider how different types of counters are implemented; binary, Grey, and LFSR counters are created in different ways and require different amounts of resources. Keep in mind that a minimal number of changing signals will always deliver the lowest dynamic power solution.

Binary Counters

Typical binary counters will have their outputs changing state at a rate of:

$$\text{Percent of bits toggling} = \frac{2^{n+1} - 2}{n2^n} X\ 100$$

Where the number of bits of the counter = n

So, a typical 8-bit binary counter would have approximately 25% of its bits changing state for any single clock edge.

LFSR Counters

LFSR (Linear Feedback Shift Register) counters are wonderful solutions for FPGA users who need to keep look-up table fan-in to a minimum. However, because of the internal hardwired feedback of CPLDs, this type of counter consumes much more power than the other counter examples described here.

For example, an 8-bit LFSR counter has approximately 50% of its bits changing on average for any single clock edge. In comparison, an 8-bit binary counter changes at a 25% bit rate.

Grey Code Counters

Because of their characteristic step pattern of a single changing bit, Grey code counters offer designers the lowest power consumption of these three counter methods. The average bit change rate for an 8-bit Grey code counter is approximately 13% as defined by the equation:

$$\text{Percent of bits toggling} = \frac{1}{n} X\ 100$$

The Grey code design implementation is the most difficult, however, because next-state information must be coded for each count value.

The Bottom Line

Take full advantage of the CoolRunner low power benefits by downloading the free "Low Power Tips for CPLD Design" application note (in PDF format) from the Xilinx website at: www.xilinx.com/xapp/xapp346.pdf.

Creating a Low Power Serial Peripheral Interface

How to implement communications between microprocessors and peripherals, using the Xilinx CoolRunner XPLA3 CPLDs.

by Anita Schreiber
Staff Applications Engineer, Xilinx
anita.schreiber@xilinx.com

The CoolRunner implementation of a Serial Peripheral Interface (SPI) Master described here can be used to add an SPI controller to microprocessors or microcontrollers that do not provide this interface. It will permit direct inter-processor communication and communication with numerous commercially available peripherals.

Serial Peripheral Interface Protocol

SPI is a full-duplex, synchronous, serial data link. A single SPI device is configured as a master; all other SPI devices on the SPI bus are configured as slaves.

The SPI bus consists of four wires:

- Serial Clock (SCK) - Driven by the SPI master and regulates the flow of data bits. The SPI specification allows a selection of clock polarity and a choice of two fundamentally different clocking protocols on an 8-bit oriented data transfer.
- Master Out Slave In (MOSI) - Data output from the SPI Master and input to the SPI Slaves.
- Master In Slave Out (MISO) - Data input to the SPI Master and output from the selected SPI Slave. Only one selected slave device can drive data out from its MISO pin.
- Slave Select (SS) - Selects a particular slave via hardware control. Slave devices that are not selected do not interfere with SPI bus activities. The SS control line can be used as an input to the SPI master indicating a multiple-master bus contention (SS_IN). If the SS signal to the master is asserted, it indicates that some other device on the bus is attempting to be a master and address this device as a slave. Assertion of SS automatically disables SPI output drivers in the master device if more than one device attempts to become master.

The SCK, MOSI, and MISO pins of all SPI devices on the SPI bus are connected together in parallel.

CoolRunner SPI Master Implementation

This SPI master design supports the following features:

- Microcontroller interface.
- Multi-master bus contention detection and interrupt.
- Eight external slave selects.
- Four transfer protocols available with selectable clock polarity and clock phase.
- SPI transfer complete interrupt.
- Four different bit rates available for SCK.

A high-level block diagram is shown in Figure 1. The microcontroller (µC) interface is a VHDL module that you can easily modify to support other microcontrollers.

The Address Decode/Bus Interface logic interprets the bus cycles of the microcontroller and performs the read/write operations to the Register File.

The Register File is the interface between the µC and the SPI master logic, and allows the µC to configure and control the operation of the SPI master. Status of the current transfer is provided to the µC via a status register in the Register File. Registers are also included to contain the µC data to be transmitted on the SPI bus and data received from the SPI bus. The SPI Control State Machine controls the shifting and loading of SPI data in the SPI shift registers, and the generation of the slave select signals. The SCK clock logic generates an internal SCK based on the settings in the control register for clock phase, division, and polarity.

Conclusion

CoolRunner CPLDs operate at the lowest standby power (<100µA) of any CPLD available today, and they are an ideal programmable logic solution for providing interface controllers in portable or power sensitive applications. See www.xilinx.com/apps/epld.htm#CoolRunner for an SPI reference design which contains a detailed application note (XAPP348), VHDL source code, and VHDL testbenches.

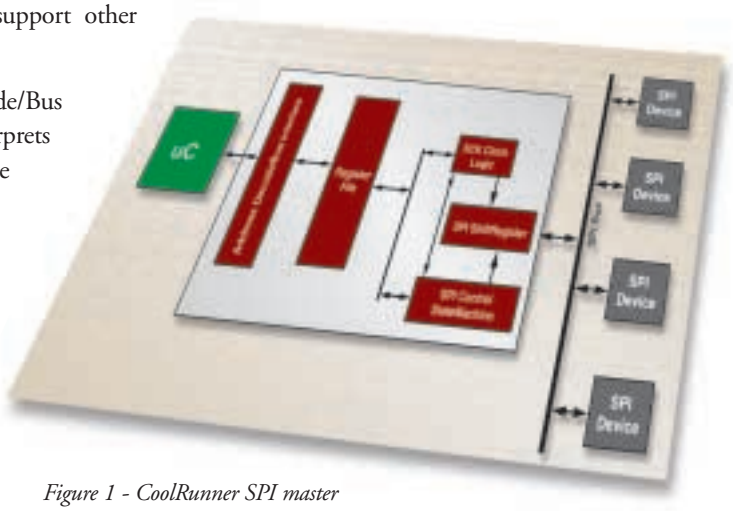


Figure 1 - CoolRunner SPI master

CoolRunner CPLDs Beat the Heat

The disadvantages of high power components...

by Steve Prokosch
CoolRunner Product Marketing, Xilinx Inc.
sprokosch@xilinx.com

Conventional thinking assumes that high performance requires high power consumption, but the Xilinx XPLA3 CoolRunner CPLD family defies convention. In both ultra-low standby and total current consumption modes, Xilinx XPLA3 complex programmable logic devices consume less power than any other CPLDs in the world.

By consuming the least amount of power, CoolRunner CPLDs radiate the least amount of heat. Devices that emit excessive heat can cause serious problems, such as:

- Higher FIT (Failures In Time) rates.
- Intermittent field failures.
- Higher cabinet or enclosure design and manufacturing costs.
- Increased risk of EMI/RFI leakage (caused by extra cooling vents).
- Mechanical stress to package parts.
- Printed circuit board layout concessions.
- Design compromises that affect the overall size and appeal of the end product.

The High Costs of High-Power Components

In addition to radiating destructive heat, devices that consume high amounts of power also generate extra costs. For instance, when you buy power supplies, the more power output you need to run the system, the higher the cost. Many designers accept the high costs of power supplies-without ever questioning the efficacy and efficiency of the devices demanding all that power.

Furthermore, a device that consumes a lot of power may require the addition of another physical component, such as a larger cooling fan or a heat sink. Adding to the BOM (bill of materials) can be expensive. Besides the cost of the component itself, additional expenditures of time and money can be incurred:

- Locating and ordering the component.
- Shipping and delivery costs and delays.
- Controlling inventory.
- Dealing with the availability of the physical device.

Availability can be a real budget-buster; if that one single component cannot be obtained or delivered in a timely fashion, the entire system cannot be shipped to the customer. Assembly costs continue to rise until all components are delivered, installed, and shipped. Such a delay can turn a company's bottom line upside down.

The CoolRunner Reliability Advantage

Heat plays an enormous role in determining the reliability of your designs in the field. Because most semiconductor devices are tested for hot temperature operating life (HTOL), it is easy to compare how long a product would last under high temperature conditions.

Consider how well a CoolRunner XPLA3 256-macrocell device performs in a Thin Quad Flat Package. The TQFP has the worst thermal characteristics of any available CPLD package. A worst-case analysis shows a CoolRunner XPLA3 256-macrocell device would run for 107 years at a constant 145 degrees Celsius!

As shown in Table 1, all other CPLD products, even in their low-power modes, radiate more heat than CoolRunner CPLDs. (For more information on how these measurements were obtained, see the Xilinx Thermal Emissions Web page at: www.xilinx.com/products/cpldsolutions/techtopic/thermalimg.htm.)

Conclusion

Xilinx CoolRunner XPLA3 CPLDs are designed for high-performance, low-power products such as portable PCs, PDAs, and handheld wireless devices where heat can be a critical factor in form, fit, and function. If heat is the problem, CoolRunner CPLDs are the solution.

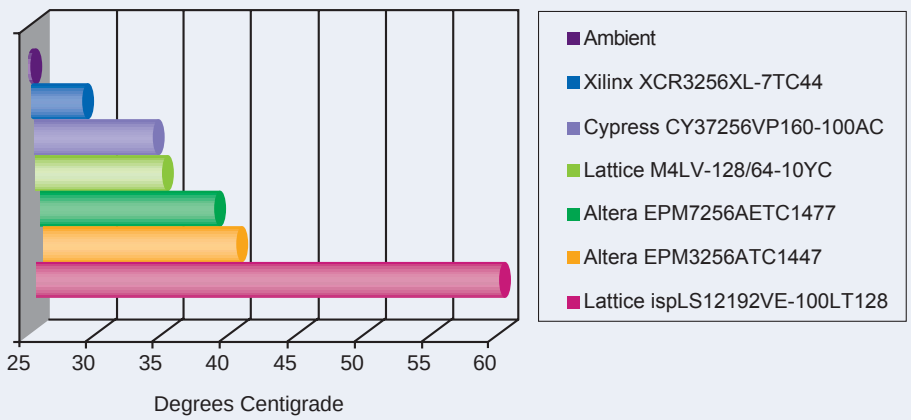
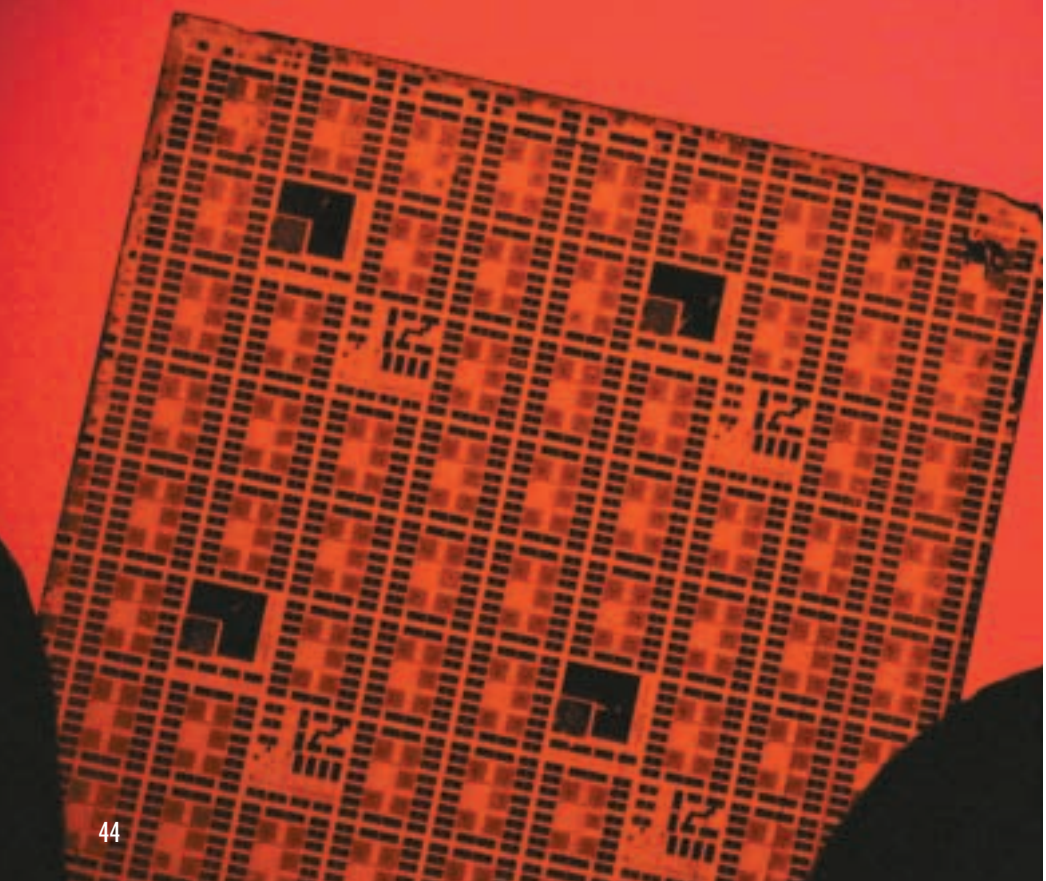


Table 1 - CPLD thermal emissions

FPGAs — The Solution to Ultra-Deep Sub-Micron Design

With Xilinx FPGAs you can focus on your design function without being concerned with the tricky physical design issues caused by today's ultra-deep sub-micron device geometries.



by Austin Lesea
Principal Engineer, Xilinx
austin@xilinx.com

Sudip Nag
Manager Implementation Tools Engineering, Xilinx
sudip@xilinx.com

Hitesh Patel,
Manager Alliance EDA Marketing, Xilinx
hitesh@xilinx.com

As device geometries decrease from 0.25µ to 0.13µ, the problems of substrate coupling, ground bounce, and interconnect crosstalk increase dramatically. ASIC designers who want to take advantage of these new device technologies are faced with a difficult task; they must create designs that are both logically correct and reliable within the specified environmental extremes. As device geometries decrease, it becomes much more difficult to produce reliable designs.

The FPGA Solution

The FPGA solution (consisting of both devices and software) assures you of a reliable design, because FPGAs are composed of a consistent architecture that has been tested over a long period of time, under real world conditions. Thus, FPGAs are guaranteed to operate exactly as specified, with very predictable interconnect delays, so you can spend more time on design optimization.

Substrate Coupling

Substrate coupling is a serious problem for either an ASIC or an FPGA. To guarantee performance, both ASIC and FPGA IC designers must use the best tools and models possible to accurately simulate the device, and have the means to automatically verify the design once it is ready for mask making.

Xilinx has developed proprietary automated techniques similar to those used by EDA vendor CadMOS (used in their SeismIC™ and PacifIC™ products for traditional DSM ASIC designs). We use these tools to automatically model and verify all of our FPGAs, which isolates you from this issue.

Substrate Bounce

Substrate bounce is caused by the switching of fast, high-current I/O transistors. It can cause double-clocking, as well as indeterminate and invalid logic states. The substrate bounce effects on Xilinx FPGAs are modeled precisely, and our designs are guaranteed to provide adequate isolation. For all of our FPGAs, Xilinx models and minimizes substrate noise in the design prior to making the device masks for fabrication.

Interconnect Crosstalk

Interconnect crosstalk between the tiny wires, on multiple metal layers, now requires a 3D field solver for extraction; anything less is not accurate enough to completely model the interconnections on the chip. Models must take into account the potential for crosstalk induced delay so that any possible user circuit will behave predictably and reliably, regardless of process and silicon variation. Xilinx IC designers perform extensive interconnect modeling using field modeling to ensure that our FPGAs do not have crosstalk problems.

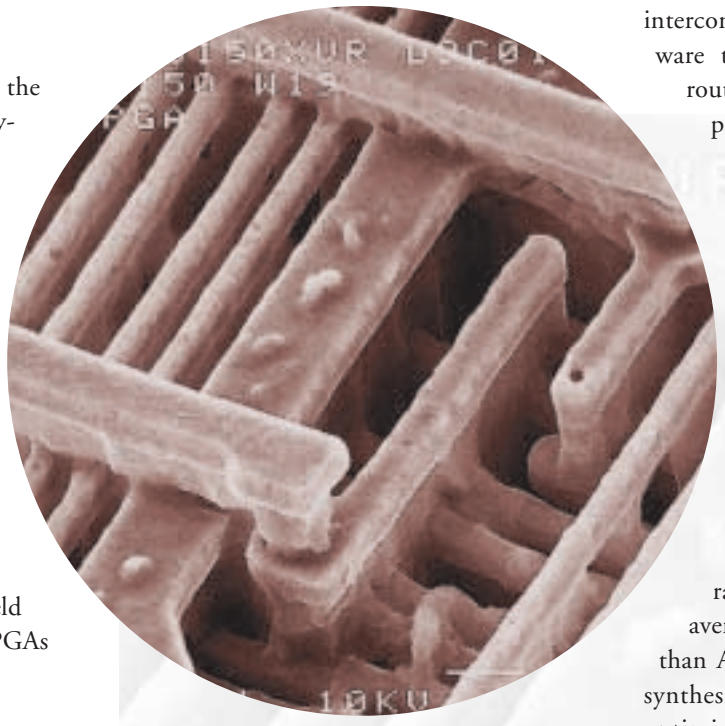
Device Fabrication

IC Designers must also model devices carefully to avoid yield problems, speed grading issues, and design failures. Xilinx manufactures millions of devices for each FPGA family, and the manufacturing process utilizes device monitors, test structures, and other process related structures that are measured on every wafer. The models are incrementally refined so that all process corners are modeled. The use of highly accurate device models enables Xilinx IC designers to rigorously verify and characterize all of our devices.

The combination of extensive and accurate modeling, simulations, and verification requires hundreds of engineer-years. Xilinx has made the investment and pre-engineered all of our devices so you don't have to worry about whether the silicon will work. ASIC IC designers must also expend the same effort, but often the expense is too high and resources are inadequate, resulting in designs that are not reliable.

The Software Solution

Using the current ultra-deep sub-micron design rules, interconnect delays account for approximately 75% of the path delays. Therefore, design modeling and timing closure become significant factors, and development tools are critical.



For FPGAs, interconnect delays have always been a significant portion of path delays because of the existence of switches in the routing paths. However, FPGAs have specific, fully characterized routing resources and, therefore, accurate delay models can be achieved using exhaustive empirical methods and functional analysis.

FPGA software tools are extremely mature in the area of handling interconnect delays. Therefore the ultra-DSM technology does not pose a new problem for FPGA place and route tools. Specifically, the delay estimation methods are mature and sophisticated resulting in accurate delay prediction. The placement and routing tools are smart enough to determine when to update the timing/slack information dynamically, based on changing interconnect delays during layout.

The newer Virtex and Spartan-II generation FPGAs, are co-developed by the Xilinx software and hardware teams. This process naturally results in a highly predictable architecture, and predictable interconnect delays. This allows the software to make correct placement and routing decisions early in the design process, even during the synthesis phase where there is maximum flexibility to influence the design performance.

The quality of the synthesis wireload models plays a key role in the timing predictability after synthesis. Advancements in FPGA synthesis technology (such as improved wire delay estimation by synthesis-driven placement tools) enable highly accurate timing predictability, and is on average 20% to 25% more accurate than ASIC technology. In addition, re-synthesis capabilities for critical path optimization reduce the number of design iterations for faster time to timing closure.

Conclusion

With FPGAs, you can now implement multi-million gate designs without being plagued by the DSM problems inherent in ASICs. Our advanced FPGA architectures shelter you from physical design issues such as crosstalk and ground bounce, and the latest synthesis and implementation software delivers timing predictability early in the design flow, giving you a significant reduction in design closure time, and a shorter time to market.

Implementing a Histogram for Image Processing Applications

Use the Virtex™ or Spartan™-II True Dual-Port™ RAM and DLLs to create a real-time histogram.

by Edgard Garcia
Engineer, Multi Video Designs
edgard.garcia@mvd-fpga.com

Image processing is key for many automated industrial inspection applications. However, even the most sophisticated algorithms can't extract the right information if the image contents are not available in a convenient format. By using a histogram, you can ensure that the image content can be easily processed.

What is a Histogram?

For each possible pixel value, the histogram algorithm counts the number of times the value was encountered in the current image. For example, the histogram of an 8-bit-per-pixel image will contain 256 values (2^8), each one representing the number of pixels found at this value. This allows a microprocessor or DSP to quickly get the profile of the image, and take the appropriate decisions, by analyzing just those 256 precomputed values. You can do this easily, in real time and at low cost, in a Virtex or Spartan-II FPGA.

A Basic Hardware Implementation

For an 8-bit-per-pixel image, 256 different values are possible for each pixel, so 256 16-bit counters would be necessary to complete the real time histogram. However, only one of the 256 counters will be active at each valid pixel clock (only one value will be updated). Therefore, the registers of the 256 x 16 bit counters can be replaced by a memory array, such as a 4K-bit block SelectRAM™ organized as 256 x 16 (RAMB4_S16).

A 16-bit incrementer will allow you to update the RAM contents during a Read-Modify-Write operation, where the video data inputs are used as the address of the memory block. Figure 1 shows the block diagram of a basic hardware implementation.

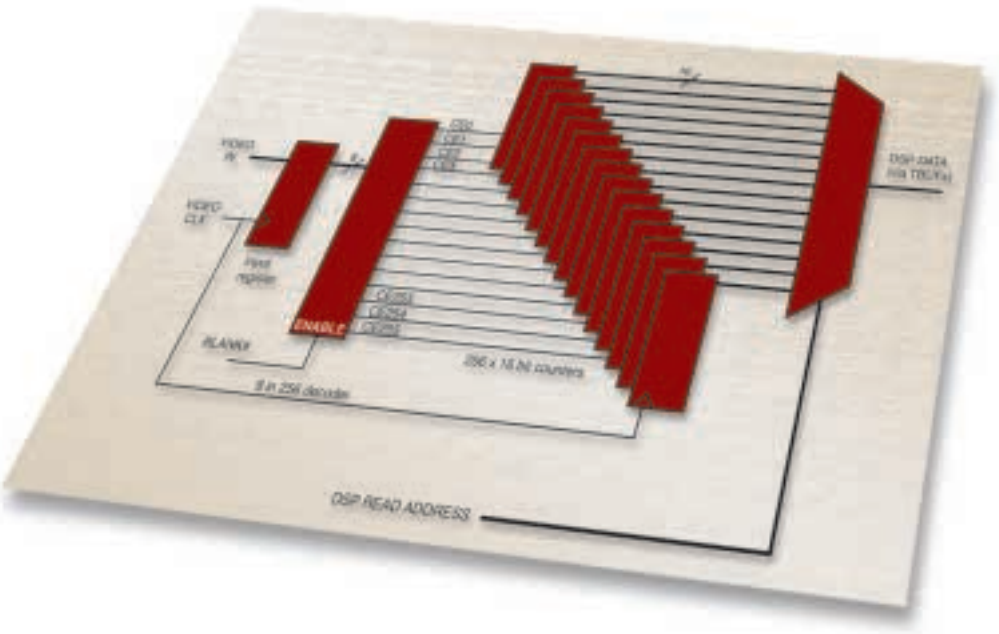


Figure 1 - Basic hardware implementation

Optimized Implementation

Each memory cycle can be either a Read or a Write, so we need to divide each pixel clock cycle in two sub-cycles: a Read cycle for getting the current value, and a Write cycle for updating (+1) the memory content. You can do this easily, using a CLKDLL to recover a clock at twice the frequency of the video clock (CLK2X), and to create an image of this clock shifted by 90° to validate Read and Write cycles. Figure 2 shows the detailed diagram of an optimized implementation.

During horizontal and vertical retrace, pixel values must be discarded. This is done with no additional logic, by connecting the BLANKING# signal to the ENA input of the memory block. Figure 3 illustrates the timing of the operations.

The DSP or microprocessor can directly read the result of the histogram by using the B port of the same block SelectRAM (configured as a RAMB4_S16_S16). A multiplexer is not needed because the two ports (A and B) each have dedicated inputs.

Resources and Performance

Here are the logic resources required for implementing the histogram algorithm:

- 1 x CLKDLL + BUFG
- 1 x RAMB4_S16_S16
- 1 x 16-bit INCREMENTER (8 slices)

For a Virtex -6 or Spartan-II -6 device, Fpix = 50 MHz.

Conclusion

By taking advantage of the high level features of the Virtex and Spartan-II FPGA architectures, you can greatly increase the speed and reduce the cost of your designs. For more information about how to implement the histogram algorithm, e-mail: edgard.garcia@mvd-fpga.com

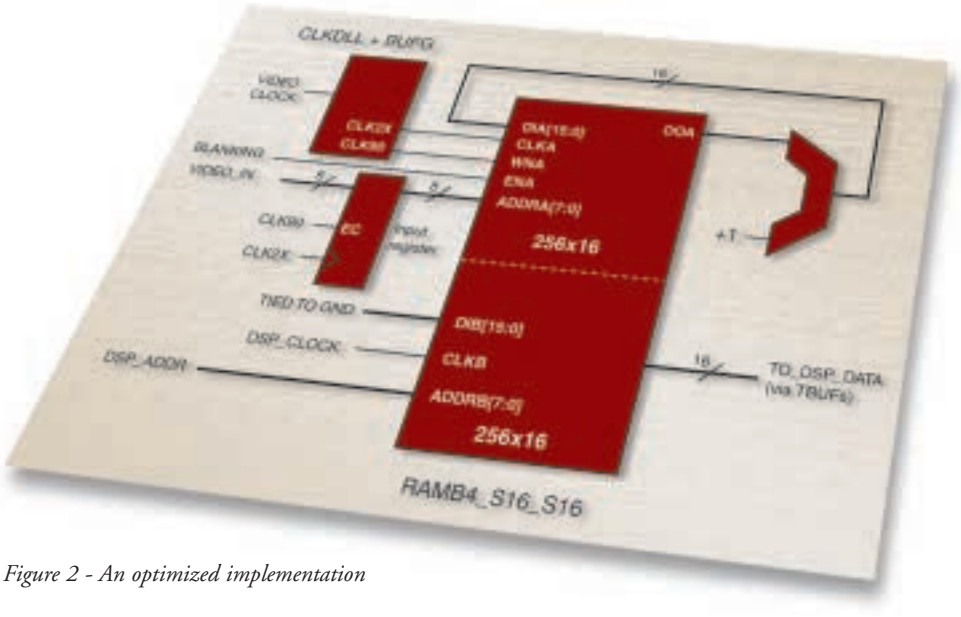


Figure 2 - An optimized implementation

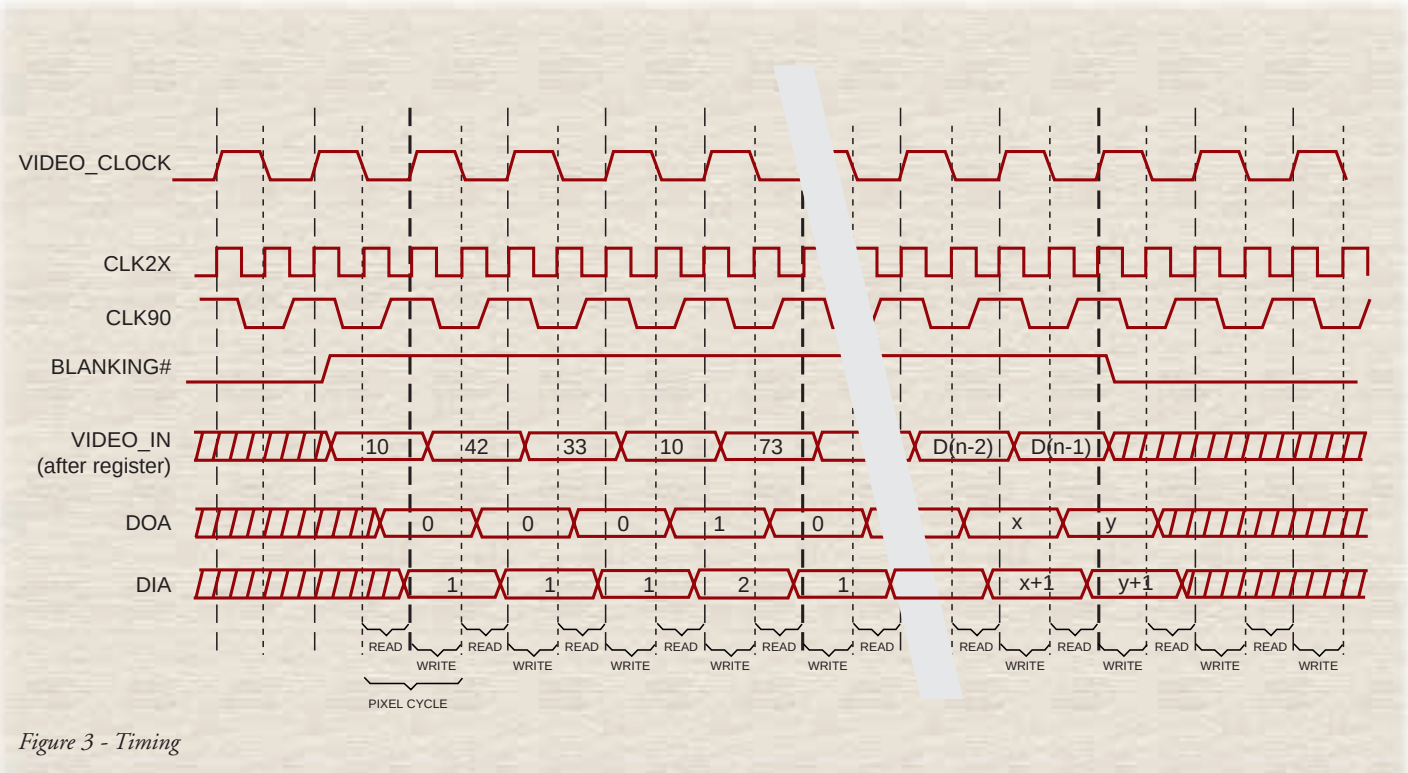
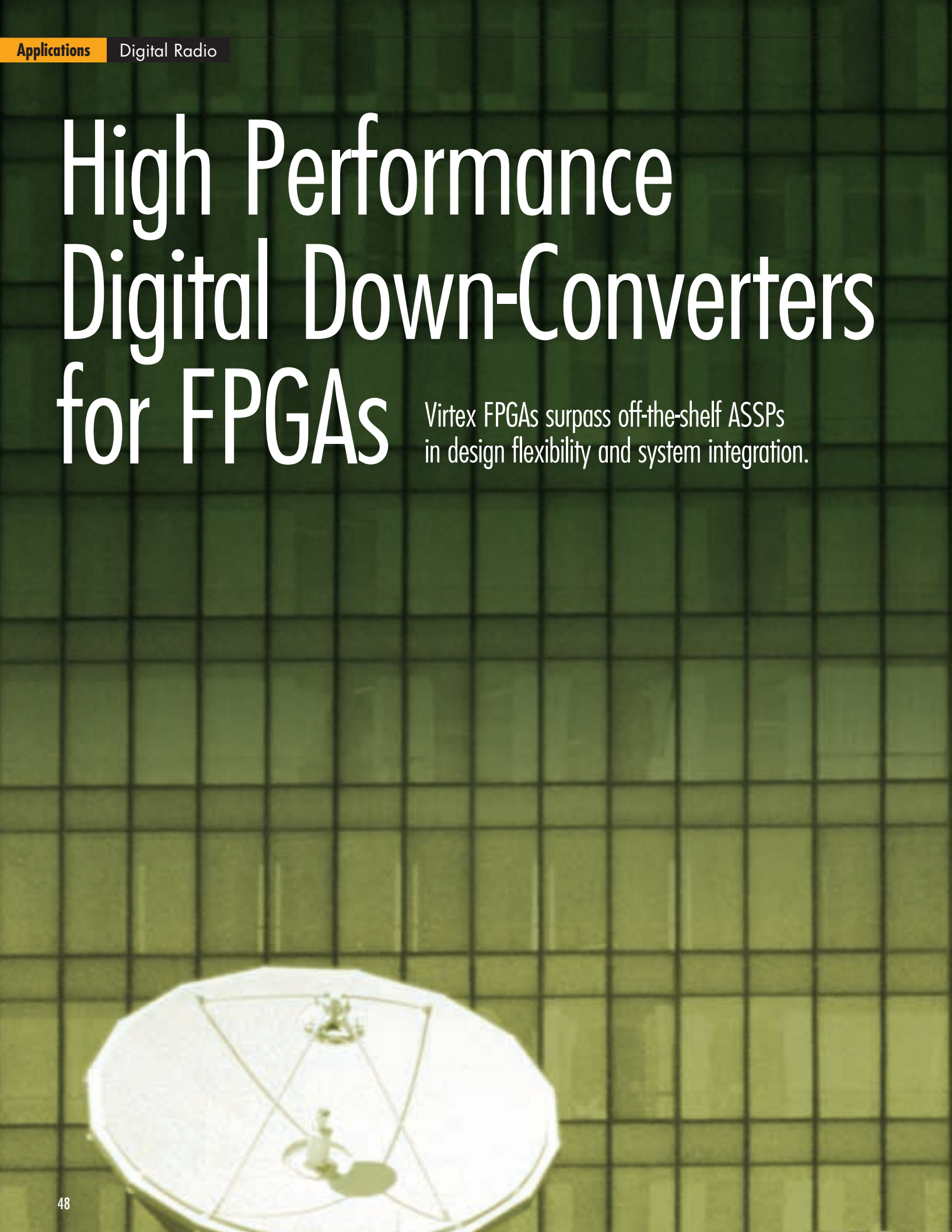


Figure 3 - Timing

High Performance Digital Down-Converters for FPGAs

Virtex FPGAs surpass off-the-shelf ASSPs in design flexibility and system integration.



by Ray Andraka
President, Andraka Consulting Group, Inc
ray@andraka.com

Digital down-converters (DDC) are a key component for digital radio. The DDC performs the critical frequency translation needed to recover the information from a digitized modulated signal.

Thanks to the high-level of interest in digital radio, the market for DDC devices is soaring. Typically, a designer will select an off-the-shelf application-specific-standard-part (ASSP) for this task. Although the costs of these parts have fallen precipitously in the face of market demand, ASSPs don't offer the design flexibility or integration attainable in an FPGA.

ASSP vendors are stuck with the challenge of creating a one-size-fits-all design, and end users are stuck with fitting the device to their needs–often paying for features or performance they don't need or want. DDCs implemented in FPGAs, however, can compete with ASSPs by offering the additional benefits of customizability and higher integration.

A down-converter consists of a numerically controlled digital oscillator, a mixer (shown as a pair of multipliers), and a low pass filter, as shown in Figure 1. The band-limited output from the filter allows us to reduce the sample rate by decimating. The design is fairly straightforward, although we must to pay attention to the fidelity of the digital sinusoid-sine and cosine waveforms produced by the numerically controlled oscillator. We must also consider the quality of the filters, if we are to have acceptable noise performance. (We must keep the design from adding so much noise to the incoming modulated signal that we can't reliably detect it. How much noise is acceptable depends on the application.)

Some digital radio applications have fairly high sample rates, which can make the design more challenging. With careful design, however, modern FPGAs can handle data as fast as any commercially available analog-to-digital converter can supply it. The advantage of using an FPGA is that it allows us to customize the DDC to exactly match our application. Furthermore, with an FPGA implementation, we can put the DDC and any post-processing in the same chip. Post-processing is usually some form of demodulator.

The Oscillator

In terms of system performance, the critical component in digital down-conversion is the numerically controlled oscillator (NCO). This component generates a sampled digital sinusoid, which when mixed with the incoming signal, shifts the signal's spectrum. In other words, if we multiply (mix) a signal with a sine wave, we get a frequency translation or "shift" of the spectral image. The amount of translation is equal to the frequency of the "carrier" sine wave.

Insufficient precision or accuracy in the sinusoid leads to degraded signal-to-noise ratios and to spurious spectral artifacts, either of which can swamp the incoming signal. Attention to the quantization that leads to these noise terms is essential for the proper design of an NCO. In our implementation model, our NCO consists of a phase accumulator frequency synthesizer and a phase angle-to-wave shape conversion. The phase angle-to-wave shape conversion circuit may be any one of several possible designs.

Frequency Synthesizer

The frequency synthesizer is simply an accumulator used to integrate a phase increment value. If we interpret the MSB (most significant bit) of the accumulator as having a weight of π then the accumulator represents the fractional portion of the accumulated phase angle. Phase accumulator frequency synthesis is discussed in detail in Xcell Journal #31 in an article by Austin Lesea (www.xilinx.com/xcell/xl31/xl31_32.pdf).

Using a phase accumulator offers several advantages over other methods:

- The synthesized frequency need not have an integer relationship to the sample clock, because modulo arithmetic preserves the fractional part of the accumulated phase on an overflow. This lets us set the local oscillator to an arbitrary frequency without changing the sample rate.
- The phase increment value does not have to be a constant. By dynamically changing the increment value, we can easily modulate the phase or frequency of the generated signal.
- Because 2^N represents a full phase revolution, this generator interfaces nicely with look-up tables for wave shape conversion. Nothing in the phase accumulator design will impair the noise performance of the NCO; reducing word width only restricts the frequencies that can be synthesized.

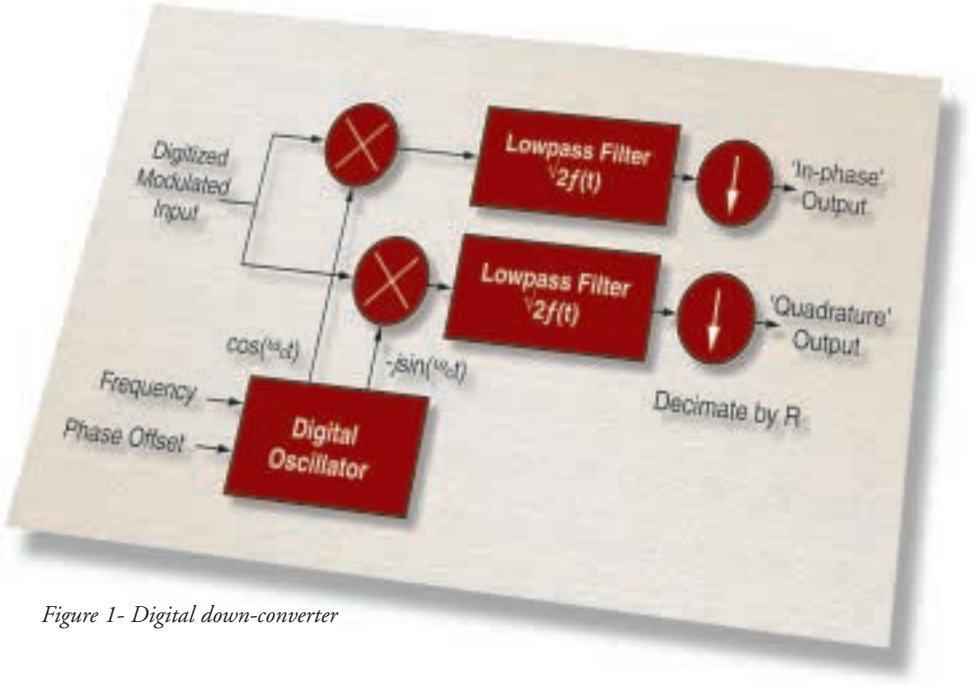


Figure 1- Digital down-converter

Noise is generated by an imperfect rendition of the sinusoid at the output of the NCO. That noise can be phase errors (angular distortions) or amplitude errors. The phase accumulator generates only a phase angle, so there is no amplitude error. Errors caused by quantization of the phase increment can cause a frequency error, but not a changing phase error.

Waveform Synthesis

The phase accumulator produces a "wrapped" phase angle that must be converted to a sampled complex sinusoid. The accuracy of the conversion directly affects the noise performance of the DDC. The noise introduced by the NCO is caused by amplitude and phase errors, which manifest themselves as reduced signal-to-noise-ratio (SNR) and degraded spurious free dynamic range (SFDR) respectively. Each additional bit of phase improves the SFDR by about 6dB and extra amplitude resolution adds to the SNR by about 6dB.

The most obvious conversion circuit is a simple lookup table of sine values by phase angle, which is addressed directly by the phase accumulator. The phase resolution determines the depth of the table, while the amplitude precision determines the width. To keep the size of the table reasonable without sacrificing frequency resolution, we must truncate the phase accumulator output, using only the MSBs at the cost of degrading the SFDR. The size of a table grows exponentially with phase resolution, so for even moderate SFDR requirements, the table becomes larger than what we would like to use in an FPGA.

Simple amplitude and phase symmetry allows us to reduce the table size by a factor of 4 by reusing the first quadrant data for the other quadrants. The same table is used for the both sine and cosine values, so if clock cycles per sample permit, the same ROM can be read twice per sample. In Virtex devices, you can use the dual-port feature of the block RAM to simultaneously obtain both the sine and cosine values from a shared ROM. Large ROMs in FPGAs are expensive in terms of resources used so, for phase resolutions of more than 8 to 10 bits, other methods should be used.

The large ROMs can be avoided by algorithmically generating the sine and cosine on the fly. While that sounds difficult, there is a simple shift-add algorithm based on vector rotation called CORDIC (COordinate Rotation DIgital Computer) that makes this task fairly easy in hardware. (See www.andraka.com/cordic.htm for details on CORDIC.) The algorithm simultaneously generates a sine and cosine value by rotating a unit vector from the "I" axis to the desired phase angle using a series of successively smaller elemental rotations. The angles of those elemental rotations are specifically selected for a shift-and-add implementation. The "I" (real or in-phase) and "Q" (imaginary or quadrature) components of the rotated vector are proportional to the cosine and sine of the phase angle respectively.

The Mixer

The function of the mixer is to multiply the incoming signal by the locally generated sinusoid to shift the spectrum of the signal. A straightforward implementation uses two multipliers, one each for the sine and the cosine. The multipliers produced by the CORE Generator tool can easily be used for this application.

If we use CORDIC for the wave shape conversion, however, we can obtain the mixer function for free. The combination of the NCO and the mixer multiplies the incoming signal by $\cos(\omega t)$-$j\sin(\omega t) = e^{-j\omega t}$. Because the NCO and mixer generate a complex phasor, the net effect is to rotate

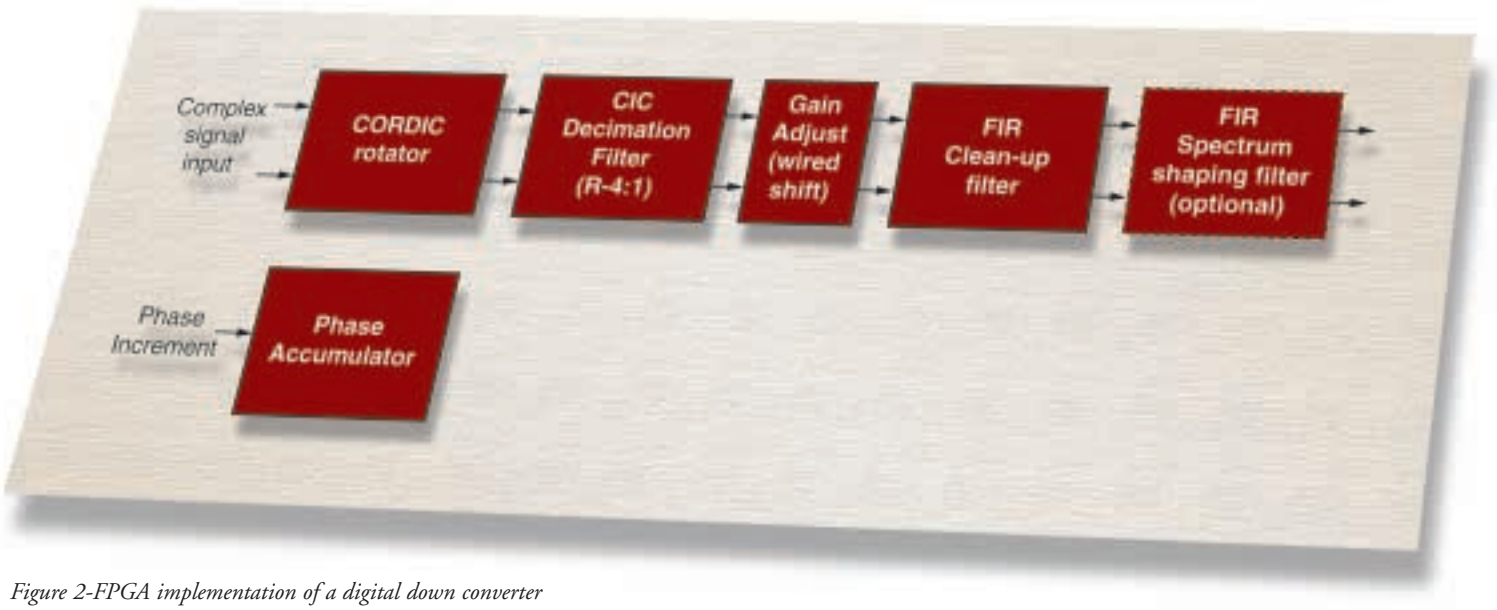


Figure 2-FPGA implementation of a digital down converter

the incoming signal by a constantly changing phase angle. Rather than rotating a unit vector to get I and Q scale values, we can use the CORDIC to directly rotate the input signal. This eliminates the two multipliers and avoids the potential for additional quantization noise.

A more subtle advantage to using CORDIC is that it actually rotates the vector rather than multiplying the components separately. This means it does not add noise to the signal other than the spectral spurs caused by the phase quantization. The CORDIC hardware occupies about the same area as a pair of multipliers with the same input width in the Virtex architecture. Thus, in effect, we have a net area savings about equal to what we would have used for the sine and cosine wave shape conversion. The CORDIC rotator also accepts a complex input, so no additional hardware is needed for applications requiring a complex signal input.

The Filter and Decimator

The mixed signal has to be filtered to isolate the portion of the spectrum containing the signal of interest. The filter typically has to be a narrow-band filter with a fairly high rejection of unwanted spectrum. This translates to an expensive filter if it is done at the input sample rate. Instead, we can use a multi-rate approach in which the signal is first decimated to a much lower sample rate using a less computationally intensive filter. Then the signal is cleaned up with a second more complex filter working at the decimated sample rate.

High Ratio Decimator

A high-ratio decimation can be performed very efficiently using a cascaded integrator-comb (CIC) filter. The CIC filter is a recursive implementation of the "boxcar" or moving average filter. The spectral response of such a filter is the sinc (sinx/x) function. In a CIC filter, the number of effective taps is an integer multiple of the decimation ratio, so the filter nulls alias onto the passband when the spectrum is folded by decimation. If the passband is sufficiently narrow, the rejection of the aliased image is quite good, much better than might be expected otherwise. We can also cascade several sections to lower the amplitude of the side lobes. The passband of this filter does exhibit a pronounced roll-off that usually must be corrected by the clean-up filter. Keeping the passband of the final filter narrow not only improves the alias rejection, but also makes the roll-off compensation easier.

The advantages of using a CIC filter in this implementation are:

- It is a computationally easy filter to realize.
- The same filter structure works for a very wide range of decimation ratios by simply changing the timing of the clock enables on the comb section.
- The filter response referred to the output sample rate is nearly independent of the decimation ratio, so one clean-up filter can be used for all decimation ratios.

The gain of the CIC filter is a function of the decimation ratio. Therefore, a barrel shifter is required after the CIC filter in applications where the decimation ratio has to be changeable without changing the circuit. This is an issue in an ASSP DDC, as it is a one-size-fits-all solution. Most of the time in FPGAs, we can hardwire the shift, or at worst, use a limited barrel shift, because we can customize the DDC for our application.

"Clean-Up" Filter

The output of the CIC filter has a sinc shape, which is not suitable for most applications. A "clean-up" filter can be applied at the CIC output to correct for the passband droop, as well as to achieve the desired cut-off frequency and filter shape. This filter typically decimates by a factor of 2 or 4 to minimize the output sample rate after the passband has been limited and shaped. An application-specific filter response, such as a raised cosine Nyquist filter, can either be combined into the correction filter or be applied at a subsequent filter stage. The clean-up filter is compactly implemented using serial distributed arithmetic (see www.andraka.com/distribu.htm for a tutorial on distributed arithmetic).

Identical filters must be applied to both the I and Q channels. Even using the slowest speed grade Virtex FPGAs, the DDC design described here can be clocked at more than 130 MHz if the design is carefully executed and floor planned. This high potential clock rate permits us to time multiplex the I and Q data through the same filters by interleaving the I and Q samples on a clock-to-clock basis. Thus for very little additional overhead, we can handle both the I and Q data in the same filter. We can also use the same technique to handle several independently tuned channels with a single instance of the DDC design.

An advantage of using an FPGA for the DDC is that we can customize the filter chain to exactly meet our requirements. With an off-the shelf chip, we would have to either fit our requirements to the chips' features or add additional post-processing to modify the output to our needs.

Conclusion

We've briefly discussed implementation of a high performance DDC in an FPGA. If we apply these techniques to a 16-bit DDC with a 64 MS/sec input and a 100 dB SFDR requirement, we come up with a design that occupies about 550 Virtex CLBs (configurable logic blocks). The occupied area is heavily influenced by specific requirements of the application. The cited design, shown in Figure 2, consists of an NCO and mixer implemented as a CORDIC rotator and a programmable decimating filter. The filter is a 4th order CIC filter followed by a 63-tap symmetric Finite Impulse Response (FIR) filter. Backing off on any of the requirements can substantially reduce the area occupied by the DDC. Because we are using an FPGA, we have the luxury of picking the features and performance to match our application. If we were to use an ASSP component, we would have to mold our requirements and design around the capabilities of the selected device.

Digital Image Processing with LogiCOREs

New Color Space Converter LogiCOREs and a Combined Forward/Inverse DCT LogiCORE give you pre-designed blocks that solve difficult design problems.

by David Mann
Multimedia ASVC Marketing, Integrated Silicon Systems
info@iss-dsp.com

The real-time manipulation of high-resolution images (either moving-picture video or still-frame image streams) usually demands custom digital video processing in hardware. But why use a bunch of different ASSPs for common video/image processing tasks-such as Color Space Conversion (RGB to YCrCb or vice versa), or Discrete Cosine Transform, when you can do it all in a Virtex, Virtex-E, or Spartan device? And the performance of these FPGA-optimized "Application-Specific Virtual Components" is very attractive.

There are several standard video processing functions that are common to many vision systems; these systems include video broadcast, machine vision, and image filtering applications. Now, thanks to Integrated Silicon Systems' ASVC technology, the IP cores powering these applications can be implemented in FPGAs, and Xilinx can supply all of the vital links in your customized video compression/decompression chain (as shown in Figure 1).

Color Space Conversion

Color Space Conversion (CSC) is one of the standard image processing techniques; it's a trick that allows you to more-efficiently use the digital image data, associated with a color pixel, by switching color domains. Processing an image in the Red-Green-Blue color space with a set of (R, G, B) values for each and every pixel really isn't very efficient. The RGB representation has a significant downside: although its the natural paradigm for rendering full-color pictures using display technologies that emit mixtures of the three primary colors (such as CRTs, LCDs, LEDs, etc), it is not as efficient as special alternative schemes.

The standard alternative representations use de-correlated components–luminance and chrominance. Thus CSC only comes into play whenever it's time to present a picture to the human visual cortex, or after a real-world image is captured using a scanner or camera followed by processing in the digital domain.

Color Space Converter LogiCOREs

Xilinx presently offers a family of four different Color Space Converter LogiCOREs, as shown in Table 1.

Other useful pre-processing functions such as Gamma Correction are also incorporated in these particular ISS-designed LogiCOREs, so you spend less time developing your CSC design using these solutions.

Here's just a few application areas for the Color Space Conversion family of LogiCOREs:

- Video output conversion to digital RGB.
- Image filtering.
- Machine vision.
- Video and still-image processing.

DCT Engine LogiCORE

Figure 1 shows where one of the RGB2YUV or RGB2YCrCb Color Space Converter LogiCOREs fits into a typical video/image processing flow. This example

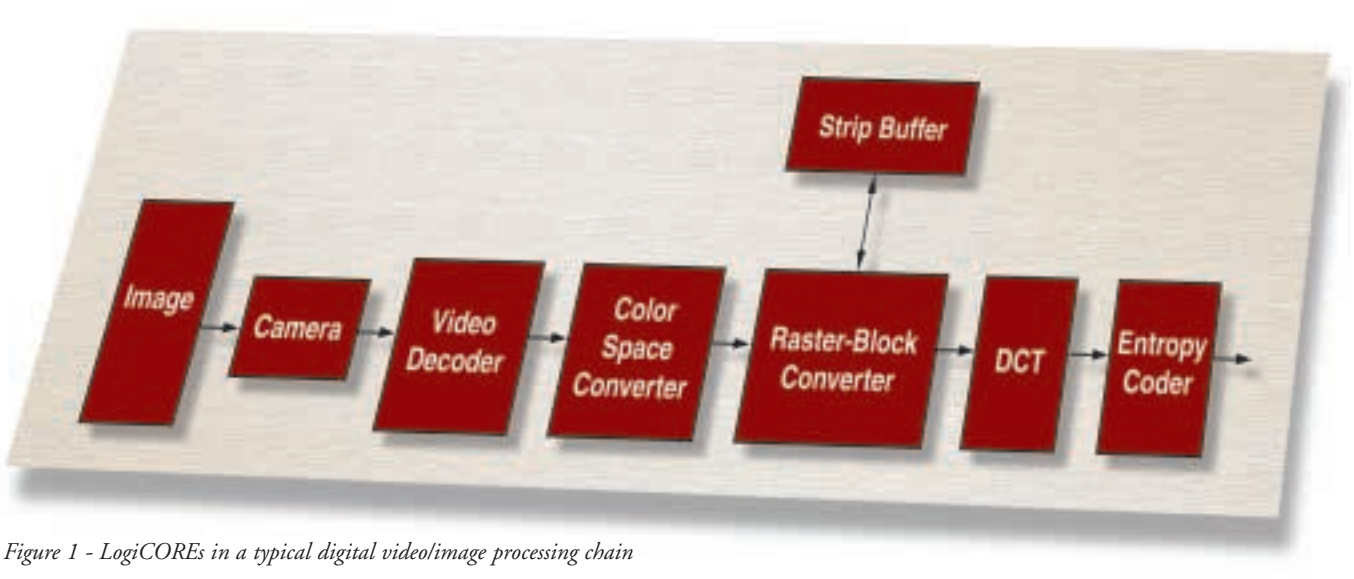


Figure 1 - LogiCOREs in a typical digital video/image processing chain

data path incorporates a Discrete Cosine Transform block–a necessary element in image compression algorithms.

Now, there's a new LogiCORE which combines both Forward DCT and Inverse DCT functions in one, and it's ISO/IEC 10918-1 JPEG compliant. This high-performance DCT/iDCT engine offers 1-symbol/cycle processing power thanks to its fully pipelined architecture. The design is highly-tuned for optimal performance across the various Xilinx FPGA technologies. It requires only 1756 slices in Virtex, 1759 in Virtex-E, or 1728 in Spartan devices; and only 48 IOBs are needed for interfacing.

This design is very efficient in the Xilinx architecture because the 2-D architecture uses row-column decomposition to separate the transform into two distinct 1-D operations. Each operation generates a set of intermediate results that are written into transpose memory. Data is "burst" into the DCT/iDCT core as blocks of 64 values, and the results of the transform are presented in the same format.

When in Forward DCT mode, this LogiCORE takes 8-bit input data words and produces an 11-bit output. In the Inverse mode, the converse is true. You've got 14-bit cosine coefficients, and a 15-bit representation in transpose memory, so there's no need to worry about precision.

LogiCORE	RGB to YUV	YUV to RGB	RGB to YCrCb	YCrCb to RGB
Slices used	230	147	211	186
IOBs used	50	50	49	49
System clock				
Virtex	>75 MHz	>75 MHz	>60 MHz	>75 MHz
Spartan-II	>80 MHz	>65 MHz	>65 MHz	>70 MHz
Virtex-E	>100 MHz	>100 MHz	>90 MHz	>90 MHz
Features used	Carry logic	Carry logic	Carry logic	Carry logic
Precision	10-bit	10-bit	10-bit	10-bit
Datasheet	Yes	Yes	Yes	Yes
Availability	Now	Now	Now	Now

Table 1 - LogiCORE specifications

Using the Combined Forward/Inverse DCT LogiCORE makes it very easy to create your own design, even if you don't have the engineering bandwidth or DCT expertise. And, the Xilinx software tools make it easy.

Designing with LogiCOREs

If you're familiar with HDL-based design and simulation, component instantiation, script-based logic synthesis, and the use of testbenches, then you're all set to design using LogiCOREs. All the LogiCORE modules described here are available under a standard license agreement from Xilinx. You get the code and test vectors, together with installation and instantiation instructions as part of the LogiCORE deliverables.

Conclusion

Digital video/image processing applications can be very difficult to develop. However, the new Xilinx LogiCOREs offer feature-enhanced Color Space Conversion and Forward/Inverse Discrete Cosine Transforms that give you a time-to-market advantage.

There are more LogiCOREs in development for digital video applications, including standalone M-JPEG Codec solutions for Virtex and Virtex-E. Talk to an ISS representative or your local Xilinx FAE about your particular application.

To find out more, or to access the datasheets, visit the Xilinx dedicated IP Center at: www.xilinx.com/ipcenter.

Stackable Development Boards for Spartan-II, Virtex, and Virtex-E FPGAs

A new series of prototyping boards to help you quickly test and implement your FPGA designs.

by Dr. Stefan Schafroth
Hardware and software development engineer,
ErSt Electronic GmbH
stefan.schafroth@erst.ch

To help you increase your productivity and decrease your time to market, we recently designed a new set of development boards, using Spartan-II, Virtex, and Virtex-E FPGAs. Like our original Virtex-based board, the new boards provide all the necessary basic components needed in most of FPGA-based designs. In addition, we incorporated an optional large ZBT RAM to satisfy the needs of modern telecommunication and imaging applications. All I/Os are routed to header connectors where you connect your special purpose interfaces. By stacking several boards you can easily cope with complex designs that exceed the scope of a single FPGA. The boards are fully compatible with their predecessor such that

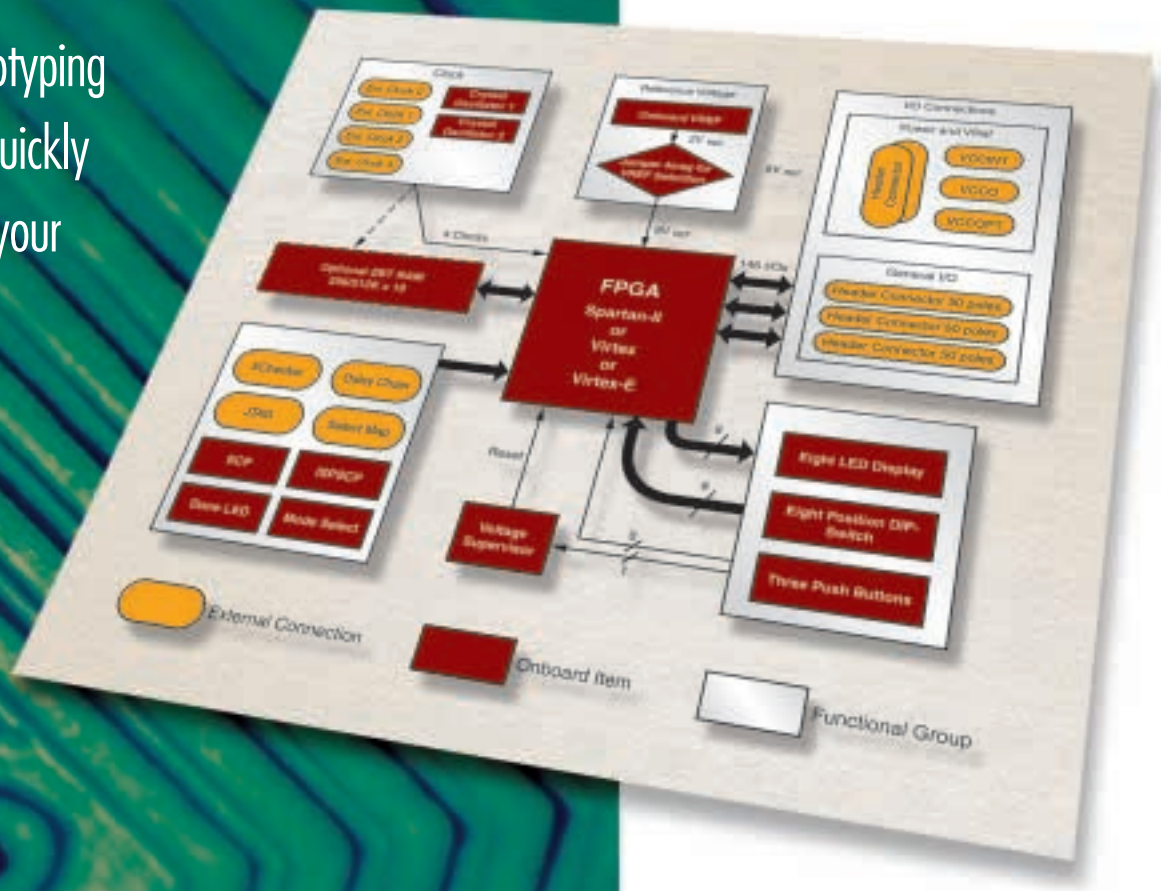


Figure 1 - Functional diagram of the development board modules

you can stack them together and reuse our power module (PWR3) as the supply for the various required supply and reference voltages.

Key Features

Each development board uses either a Spartan-II, Virtex, or Virtex-E FPGA in a PQ-208 package (Spartan-II) or an HQ-240 package (Virtex, Virtex-E). Vital components for a basic system are placed around the FPGA, including two crystal oscillators, three push buttons, eight DIP switches, and nine status LEDs. An optional ZBT RAM helps to support any memory demanding applications.

All configuration modes of the FPGA are supported, and you can provide configuration data either by using serial configuration PROMs (SCPs) sitting in onboard sockets, in-system programmable (ISP) PROMs, or by connecting a Xilinx MultiLINX, XChecker, or JTAG cable. The ISP PROMs and the FPGA form a single JTAG chain. A functional diagram detailing the building blocks of the prototyping boards is shown in Figure 1.

The crystal oscillators are housed in standard DIL-8 or DIL-14 size metal cans plugged into sockets, so you can easily change the frequency. To facilitate the distribution of very fast clocks, we mounted four SMB coaxial connectors, next to the clock pins of the FPGA, which may be terminated with optional resistors to ground. The synchronous clock input of the ZBT RAM is also connected to one of these connectors. Alternatively you can use an FPGA-generated clock driven on an I/O pin.

Push buttons, DIP switches, and LEDs form a user interface that allows you to provide configuration data and monitor display status information from the running system.

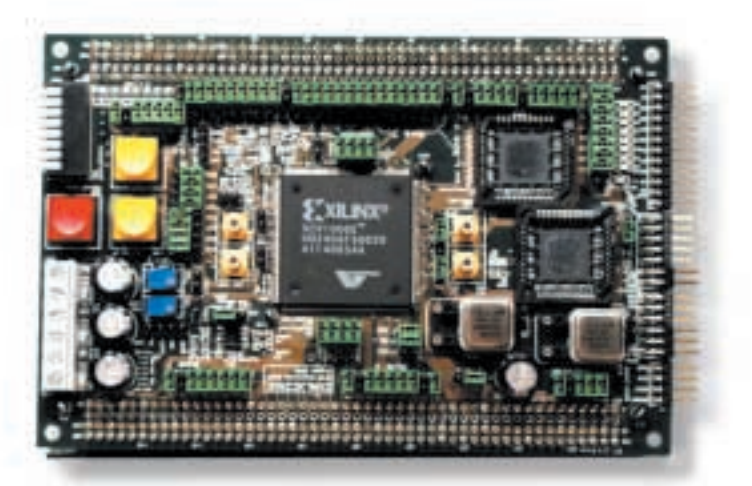


Figure 2 - Top view of the development board module

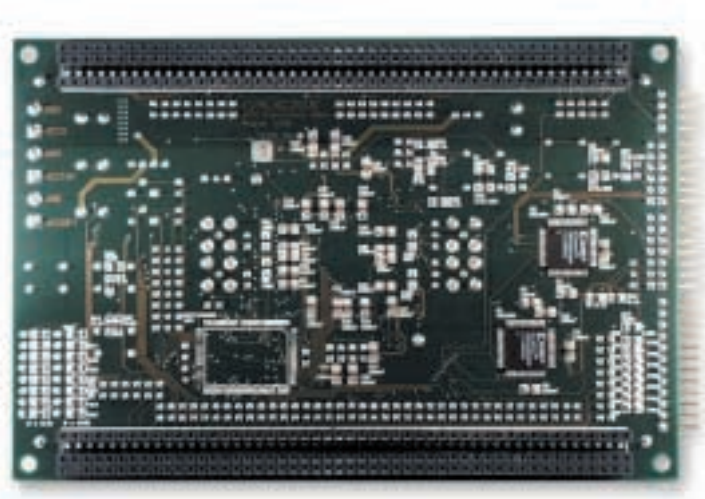


Figure 3 - Bottom view of the development board module

You can configure all eight I/O banks independently of each other, and you can select their VCCO and reference voltages individually with jumpers. Two different reference voltages (derived from the FPGA core voltage) can be generated onboard by means of trim potentiometers. Up to eight reference voltages can be connected from an external source, such as our PWR3 power module.

Figures 2 and 3 show the top and bottom view of a development board module equipped with an XCV1000E FPGA.

Applications

The board is very well suited to:

- Evaluate the larger members of the Spartan-II, Virtex, and Virtex-E FPGA families in the PQ-208 or HQ-240 packages, respectively.
- Experiment with different low voltage I/O standards.
- Implement custom designs using the full power of the Virtex architecture.
- Test algorithms under real-time conditions and watch the signals with a logic analyzer.
- Quickly and easily expand the complexity of the system by stacking several boards.

Conclusion

The EVALXC2S, EVALXCV, and EVALXCVE development board series gives you an ideal platform for evaluating, implementing, testing, and extending custom designs using Spartan-II, Virtex, or Virtex-E devices. Using the optional ZBT RAM you can even implement applications calling for large amounts of memory. You can also easily integrate the board into a larger system. Like their predecessor, the boards can be combined with the PWR3 power module to form a compact unit that runs from a single power supply. This makes it ideal for teaching, seminars, and courses.

For additional information on EVALXC2S/XCV/XCVE see: www.erst.ch, or contact us at info@erst.ch.

Xilinx Global Services

In the race to market, make the most of your time and resources.

by Jannis McReynolds
Business Development Manager
Xilinx, Global Services Division
jannis.mcreynolds@xilinx.com

It's a fact of life–the market waits for no one. Your new product idea won't be nearly as successful if your competitors get to market first. But success is not just about moving quickly; design methodologies are getting more and more complex with each new advance in technology, so you must also move *intelligently*.

Xilinx Global Services is a portfolio of services, and tools designed to keep you on the fast track. From technical support to education to design consulting, Xilinx Global Services compress your learning curve and accelerate your design time. The Xilinx Global Services portfolio consists of Product Services, Education Services, Design Services, and the highly acclaimed support.xilinx.com.

> "Designers face a multitude of challenges in their efforts to stay ahead of the market," said Dave DeMarinis, Business Development Manager, Global Services Division. "We developed Xilinx Global Services to help them work faster and smarter."

It's Okay To Cut In Line

Our Platinum Technical Service, which costs $1,295.00 per seat per year, is a remote technical support service developed to help you reduce the time you spend troubleshooting and accelerate your design schedule–it gives you rapid access to our Senior Application Engineers. When you call or log-on to the website, your call automatically moves to the head of the queue.

In addition, you receive eight Xilinx education credits you can use to further reduce design time.

Of course, our free services already give you access to a wealth of technical support, service packs, and software updates, but our Platinum Technical Service gives you top priority and is always available when you are: in North America between 7 a.m. and 5 p.m. Pacific Standard Time (PST), In Europe, service is available 9 a.m. to 5:30 p.m. Greenwich Mean Time (GMT). Xilinx Platinum Technical Service is a powerful tool to help you get your product to market before your competitors.

"The Xilinx Design Services teams bring a lot to the table — unique expertise, experience with Xilinx tools and development, and the fruits of investments Xilinx continuously makes in R&D," said Dave DeMarinis, the Business Development Manager for Global Services. "With our team, you pay for results, not for a consultant's time."

Move to the Head of the Class

Our Education Services are a flexible, comprehensive collection of courses and venues targeted at keeping you and your designers' technical skills sharp, so you get increased design productivity and therefore reduced time to market.

We offer a broad range of classes at training centers worldwide or we'll bring the classroom to your place of business. We even have portable classrooms complete with computers, software training materials, and instructor set-up, so all you have to do is show up ready to learn.

If you're looking to expand your skills with the ultimate in convenience and cost savings, check out Xilinx e-Learning. "Live" e-Learning is a virtual, real-time classroom which allows you to interact with our instructors and chat with other engineers over the Internet. Live e-Learning is also an excellent way to collaborate, particularly if your design team is spread out around the world, and we offer over 70 live e-Learning modules.

Recorded e-Learning allows you to log-in from anywhere, listen to recorded education sessions and learn at your own pace. The e-Learning modules can be accessed anytime, day or night, and are less expensive than the live e-Learning classes.

All of our Education Services provide excellent, hands-on training, suitable for all skill levels, from the novice to the expert. Classes are led by instructors who are themselves experienced designers. With our flexible and always available courses, you can choose the instructional method that keeps you on the technical cutting edge, without sacrificing your productivity.

Stack the Deck in Your Favor

Our Design Services give you the advantage of experienced engineers, who are dedicated to your design. In the rush to market, it makes good business sense to take advantage of every resource you can, whether in-house or outside your company. Now you can outsource your design to an experienced Xilinx team, made up of best-in-class designers, silicon experts, and software specialists who become your virtual in-house design team.

You can immediately extend your project bandwidth and eliminate ramp-up time by using Xilinx design Services. The benefit is that you're free to focus on tasks of the highest priority to you, and you can feel confident that your programmable logic design is in the most capable hands.

Go Ahead... Ask Us Anything

Our free suport.xilinx.com website is recognized as the industry leader for online solutions to programmable logic design issues; it has received rave reviews from industry insiders.

At support.xilinx.com you'll find:

- The Answers Database, with over 4,000 proven design solutions.
- Problem Solvers, to troubleshoot device configuration, software installation, and JTAG issues.
- A Web support interface, which allows you to open a Platinum priority case.
- Discussion forums through which you can interact with other designers.

As you might expect, support.xilinx.com is available seven days a week, 24 hours a day, 365 days a year, so you can troubleshoot your design when it's convenient for you– anytime, anywhere, support.xiinx.com has your answers.

"When compared against the competition in customer surveys, Xilinx (support.xilinx.com) outscored every other PLD vendor's website in content, accessibility, and ease of use," wrote Gartner Group in their 1999 Corporate Customer Satisfaction Study.

Conclusion

Xilinx Global Services can increase your productivity, reduce your design time, and improve your return on investment when designing programmable logic solutions. The Xilinx Global Services portfolio of education, product, and design services, along with the support.xilinx.com online resources will extend your technical capabilities and accelerate your time to market.

For more information, contact your Xilinx sales representative. You can find the Xilinx sales office nearest you at: http://www.xilinx.com/company/sales/offices.htm.

e-Learning Makes the Grade

Get up to speed on the latest technologies quickly, conveniently, and cost effectively.

by Renne Ricciardi
Business Development Manager,
Xilinx Educational Services
renne.ricciardi@xilinx.com

Despite enormous technical advances since the days of chalkboards and spiral notebooks, traditional instructor-led classroom training is still the best way to learn when you need an in-depth understanding of a specialized topic. In the day-to-day race to market, however, time is a luxury few designers can afford.

Online Learning, On the Spot

With Xilinx e-Learning, you can choose from more than 70 online classes or modules covering a broad range of topics and skills involving Xilinx products and services. For example:

- Introduction to FPGA Design.
- Timing Constraints.
- Spartan-II Architecture.
- ModelSim XE.
- Virtex-EM Architecture.

Each module is an hour in length, and enrollment is quick and easy. Modules are taught weekly and presented at different times throughout the day to support worldwide access. Moreover, Xilinx e-Learning won't interfere with your project timeline, because there's no lost productivity due to travel time.

We can help you determine which e-learning modules are right for you through our brief self-assessment pre-tests. These tests help you gauge your knowledge and determine what you need to know so you won't be spending money on training you've already had.

Live e-Learning Environment

Live instructors present classes and modules in real time. During each session, you will have the opportunity to interact with the instructor, as well as collaborate with online subject experts. You may pose questions to the instructor, view slides, share whiteboards, and discuss issues with other students in chat rooms. Pop quizzes appear periodically throughout the session–and you get instant feedback.

Participating in an e-learning session is simple. You access the e-learning module using your Web browser and a phone connection. No additional software is required. Ten minutes before the class, you call into the conference, log on to the URL, download training documents, and you're ready to go.

Xilinx e-Learning classes are open to everyone, and each e-learning module costs $100 per session.

Customized e-Learning

If you have multiple designers who want to take the same course at the same time, or if you prefer a date or time that is more convenient for you, simply call us and we will schedule an instructor to deliver a module of your choice for your group only. The only requirement is that you have a minimum of six people who want to attend the session. The maximum number of people is 100.

A private session gives you more control over the pace of delivery. Just ask the instructor to speed up or slow down. In addition, the questions and discussion can be focused on issues and ideas important to you.

If you have designers located in remote locations, or if you have designers who are interested in different modules, call us and ask about a Bundle Package Program. With a bundle purchase, your people can complete modules when it's most convenient for them. Any of the designers can sign in to attend a session at any time throughout the year, or until you have used up all the modules you purchased.

Conclusion

Xilinx e-Learning is the most cost-effective solution to help you keep your technical skills sharp and up-to-date. To learn more about Xilinx e-Learning, visit the Xilinx e-Learning website at www.support.xilinx.com/support/education-home.htm or call the registrar at 877-959-2527.

Xilinx in the Community—Champions for Change

by Autumn Conrad
Public Relations, Xilinx, Inc.
autumn.conrad@xilinx.com

As leaders in the global semiconductor market, we take pride in sharing our success with the communities in which we do business. Since the founding of our company in 1984, Xilinx has maintained close relationships with several community partners by establishing and supporting innovative programs in the areas of education, health, and welfare.

2000 Outstanding Corporate Grantmaker

This tradition of community involvement was recently recognized by the National Society of Fundraising Executives (NSFRE) who honored Xilinx as the "2000 Outstanding Corporate Grantmaker" of the year. With this award, Xilinx joins past recipients like Hewlett Packard, Aspect Communications, Applied Materials, Sun Microsystems, AMD, Therma, and Apple Computer as philanthropic leaders dedicated to building strong communities.

This prestigious award is given to corporations that demonstrate a commitment to the community through financial support and exceptional employee involvement. Xilinx was nominated by five community organizations and educational institutions that have maintained a close relationship with us over the past 16 years. It is through such partnerships that we can truly make a difference in our communities and the lives of many.

Walk for Aids

The Santa Clara County Walk For AIDS nominated Xilinx for our active participation and support of the event over the past nine years. Once again, we hosted the annual kick-off rally and urged other corporations in Silicon Valley to get involved through increased corporate sponsorship and participation. In Santa Clara County, the Walk For AIDS serves as the largest fundraising event for nine organizations that provide services and prevention education to individuals living with HIV and AIDS. Agencies like Health-Connections, a community service of The Health Trust which also nominated us for the award, provide critical health and social services to individuals and their families affected by HIV/AIDS.

Children's Shelter of Santa Clara County

Xilinx employees remain active year round by volunteering at organizations such as the Children's Shelter of Santa Clara County. Our relationship with the Children's Shelter started when we helped contribute to the $13.7 million building fund, which is located adjacent to Xilinx headquarters in San Jose. In November of 1995, the Children's Shelter opened its doors to provide shelter, 24-hour care, and services to abused and neglected children. Since that time, the Children's Shelter has provided care to over 15,000 children.

Stock for Students

As part of our strong commitment to education, we support innovative, results-orientated programs that provide students with the skills necessary to succeed in tomorrow's technological world. We recently started a stock-sharing program called "Stock for Students" which encourages employees to donate one share of their personal stock to Xilinx-adopted schools. Through funds raised by the program, Oster Elementary School will build the "Xilinx Science Laboratory" to provide students with hands-on learning in the life, earth, and physical sciences that would otherwise be unavailable.

Faculty Endowment Fund

In higher education, the Department of Electrical Engineering at San Jose State University nominated us for establishing a faculty endowment fund for the university. Due to the high cost of living in Silicon Valley, the university was having difficulty attracting and retaining faculty. The faculty endowment fund enables the university to augment the salaries for three qualified professors annually. In addition, we recently funded the establishment of the "Xilinx Digital Laboratory" which will help prepare students to meet the challenges they will face once they graduate.

Conclusion

Through our partnerships, we have come together to explore ways to build communities that reflect the success and generosity of our times. We are proud of our partnerships with these organizations and take honor in supporting their efforts to make our community a better place for all people to live, work, and prosper. It is only by working together that we can truly make a difference.

Software Solutions

Version 3 Development Systems Quick Reference Guide

Xilinx development systems give you the speed you need. With the initial release of our version 3 solutions, Xilinx place and route times are as fast as two minutes for our 200,000 gate, XC2S200 Spartan™-II device, and 30 minutes for our one million gate, system-level XCV1000E Virtex™-E device. That makes Xilinx developmen systems the fastest in the industry.

And with the push of a button, our timing-driven tools are creating designs that support I/O speeds in excess of 800 Mbps, and internal clock frequencies in excess of 300 MHz. With each quarterly release, we are further accelerating your design process.

Xilinx desktop design solutions combine powerful technology with an easy to use interface to help you achieve the best possible designs within your project schedule, regardless of your experience level. For more information on any Xilinx products, visit www.xilinx.com

Alliance Series Solutions:
The Alliance Series Solutions contain powerful open systems implementation tools that are engineered to plug and play your existing design flow. This combination of advanced features delivers high performance results on the toughest designs.

Foundation Series ISE Solutions:
Foundation Integrated Synthesis Environment (ISE) is Xilinx next generation design environment, optimized to deliver the benefits of an HDL methodology. Foundation ISE is packed with technologies that help you bring your product to market faster.

Foundation Series Solutions:
The Foundation Series solutions are complete, ready-to-use design environments for programmable logic design based on industry-standard schematic, HDL, and pushbutton design flows.

Xilinx Web-based Design Solutions provide designers the ability to engage in digital design activities, on-line, using Xilinx application servers, or download design and implementation software modules for use in their own design environment. These applications include:

WebFITTER:
The WebFITTER is a free Web-based design tool that allows system designers to evaluate their designs using Xilinx XC9500 Series CPLDs.

WebPACK:
The WebPACK is a collection of four free downloadable software modules including ABEL v7.1, VHDL and Verilog synthesis, design implementation tools, and device programming software.

WebPACK now includes support of the entire Spartan-II FPGA family as well as the 300,000 system gate Virtex XCV300EFPGA.

Base and Base Express Configurations
The Base and Base Express configurations provide push button design flows and support a broad array of FPGA and CPLD devices targeted for low density and high volume applications.

Standard and Express Configurations
The Standard and Express configurations combine push button flows with powerful auto-interactive tools. These tools give designers more influence and control over implementation while maintaining the benefits of design automation.

Elite Configurations
The Elite configurations are designed to support powerful design flows that deliver high-performance designs for even the highest density, multi-million gate FPGA devices from Xilinx.

WebFITTER URL:
Go to the Xilinx website http://www.xilinx.com and jump to "WebFITTER"

WebPACK URL:
Go to the Xilinx website http://www.xilinx.com and jump to "WebPACK"

Version 3 Development Systems

Feature Comparison Guide

Design Entry	Alliance	Foundation	Foundation ISE	WebPACK
Schematic		●	●	●
VHDL, Veriolog HDL, ABEL, HDL		●	●	●
State Diagram Editor		●	●(1)	●(1)
Floorplanner	●	●	●	●
CORE Generator	●	●	●	●
Timing Constraint	●	●	●	●
Modular Design	(Optional)		(Optional)	
Design Synthesis	**Alliance**	**Foundation**	**Foundation ISE**	**WebPACK**
Xilinx Synthesis Technology (XST)			●	●
FPGA Express / Incremental Synthesis		●(5)	●	
Design Verification	**Alliance**	**Foundation**	**Foundation ISE**	**WebPACK**
Timing Simulation	●	●	●	●
Gate Level Simulator		●	●(2)	●
HDL Simulator	●(1)	●(1)	●(1)	●(1)
HDL Testbench Generator			●(1)	●(1)
Integrated Logic Analysis (ChipScope ILA)	(Optional)	(Optional)	(Optional)	
Static Timing Analysis	●	●	●	●
Design Implementation	**Alliance**	**Foundation**	**Foundation ISE**	**WebPACK**
Constraints Editor	●	●	●	●
CPLD ChipViewer	●	●	●	●
FPGA Editor	●	●	●	●
Error Navigation to Xilinx Web			●	●
Command Line Operation	●		●	●
HTML Timing Reports	●	●	●	●
Data Book I/O Timing	●	●	●	●
Timing-Driven Place and Route	●	●	●	●
Multi-pass Place and Route	●	●	●	
Project Archiving	●	●	●	●
System Interfaces	**Alliance**	**Foundation**	**Foundation ISE**	**WebPACK**
EDIF In	●	●		CPLD Only
PROM File Generator	●	●	●	●
JTAG Download Software	●	●	●	●
IBIS	●	●	●	●
STAMP	●	●	●	●
VHDL, Verilog Out	●	●	●	●
HDL Simulation Libraries	●	●	●	●
Environment	**Alliance**	**Foundation**	**Foundation ISE**	**WebPACK**
Operating System	PC / UNIX	PC	PC	PC

Device Comparison Guide

Elite	Standard/Express	Base/Base Express	WebPACK ISE
All Virtex-II Family	Virtex-II Family up to XC2V1000	Virtex-II Family up to XC2V80	Virtex XCV300E only
All Virtex-E Family	Virtex-E Family up to XCV1000E	Virtex-E XCV50E only	All Spartan-II Family
All Virtex Family	All Virtex Family	Virtex XCV50 only	All CoolRunner Series(4)
All Spartan Series	All Spartan Series	All Spartan Series	All XC9500 Series
All XC9500 Series	All XC9500 Series	All XC9500 Series	
All XC4000E/L/EX	All XC4000E/L	All XC4000E/L	
All XC4000XL/XLA	All XC4000XL/XLA/EX/XV(3)	XC4000XL/XLA up to XC4020	
All XC3000(3)	All XC3000(3)	All XC30003	
All XC5200(3)	All XC5200(3)	All XC52003	

1. Evaluation functionality available through the Xilinx ALLSTAR program. For more information on the ALLSTAR program, go to www.xilinx.com.
2. Functional and timing simulation is performed using a HDL simulator in the ISE product.
3. XC3000, XC5200, and XC4000XV devices are not supported in the Foundation Series ISE configurations.
4. CoolRunner series is only available in WebFITTER and WebPACK at this time.
5. Foundation Base does not include a license for FPGA Express.

Virtex and XC4000X Series FPGAs

Each Virtex family has its own unique features to meet different application requirements. All devices have both distributed RAM and block RAM, and between four and eight DLLs for efficient clock management.

- The Virtex family, consisting of devices that range from 50K up to 1 million logic gates, supports 17 I/O standards, and offers 5V PCI compliance.
- The Virtex-E family offers the highest logic gate count available for any FPGA, ranging from 50K up to 3.2 million system gates, and supports 20 I/O standards including LVPECL, LVDS, and Bus LVDS differential signaling.
- The Virtex-EM Extended Memory family consists of two devices that have a high RAM-to-logic gate ratio that is targeted for specific applications such as gigabit per second network switches and high definition graphics.

The XC4000X Series is part of the broad spectrum of Xilinx "XL" products unveiled September, 1998. As a result, Xilinx offers the broadest choice of 3.3V and 2.5V devices available from a single supplier, with densities ranging from 800 to 500,000 system gates. With 12 family members ranging from 30,000 to 500,000 system gates, the devices feature patented SelectRAM memory, with a highly flexible arrangement of logic, single-port, or dual-port memory. Designed in an advanced 0.25 micron process, the XC4000X series delivers industry-leading performance while significantly reducing power consumption.

See www.xilinx.com for more information

FPGA Product Selection Matrix

		DENSITY							FEATURES						
DEVICES	KEY FEATURES	Logic Cells	Maximum Logic Gates	Typical System Gate Range	Max. RAM Bits	CLB Matrix	CLBs	Flip-Flops	Max. I/O	Output Drive (mA)	PCI Compliant	1.8 Volt	2.5 Volt	3.3 Volt	5.0 Volt
XC4013XLA	XC4000 Series: Density Leadership/ High Performance/ SelectRAM Memory	1368	13K	10K-30K	18K	24x24	576	1536	192	12/24	Y	–	–	–	X
XC4020XLA		1862	20K	13K-40K	25K	28x28	784	2016	224	12/24	Y	–	–	–	X
XC4028XLA		2432	28K	18K-50K	33K	32x32	1024	2560	256	12/24	Y	–	–	–	X
XC4036XLA		3078	36K	22K-65K	42K	36x36	1296	3168	288	12/24	Y	–	–	–	X
XC4044XLA		3800	44K	27K-80K	51K	40x40	1600	3840	320	12/24	Y	–	–	–	X
XC4052XLA		4598	52K	33K-100K	62K	44x44	1936	4576	352	12/24	Y	–	–	X	*
XC4062XLA		5472	62K	40K-130K	74K	48x48	2304	5376	384	12/24	Y	–	–	X	*
XC4085XLA		7448	85K	55K-180K	100K	56x56	3136	7168	448	12/24	Y	–	–	X	*
XCV50	Virtex Family: Density/ Performance Leadership BlockRAM Distributed RAM SelectI/O 4 DLLs	1728	21K	34K-58K	56K	16x24	384	1536	180	2/24	Y	–	–	X	*
XCV100		2700	32K	72K-109K	78K	20x30	600	2400	180	2/24	Y	–	–	X	*
XCV150		3888	47K	93K-165K	102K	24x36	864	3456	260	2/24	Y	–	X	I/O	*
XCV200		5292	64K	146K-237K	130K	28x42	1176	4704	284	2/24	Y	–	X	I/O	*
XCV300		6912	83K	176K-323K	160K	32x48	1536	6144	316	2/24	Y	–	X	I/O	*
XCV400		10800	130K	282K-468K	230K	40x60	2400	9600	404	2/24	Y	–	X	I/O	*
XCV600		15552	187K	365K-661K	312K	48x72	3456	13824	512	2/24	Y	–	X	I/O	*
XCV800		21168	254K	511K-888K	406K	56x84	4704	18816	512	2/24	Y	–	–	X	*
XCV1000		27648	332K	622K-1,124K	512K	64x96	6144	24576	512	2/24	Y	–	–	X	*
XCV50E	Virtex-E Family: Density/ Performance Leadership BlockRAM Distributed RAM SelectI/O+ 8 DLLs LVDS, BLVDS, LVPECL	1728	21K	47K-72K	88K	16x24	384	1536	176	2/24	Y	X	I/O	I/O	**
XCV100E		2700	32K	105K-128K	118K	20x30	600	2400	196	2/24	Y	X	I/O	I/O	**
XCV200E		5292	64K	215K-306K	186K	28x42	1176	4704	284	2/24	Y	X	I/O	I/O	**
XCV300E		6912	83K	254K-412K	224K	32x48	1536	6144	316	2/24	Y	X	I/O	I/O	**
XCV400E		10800	130K	413K-570K	310K	40x60	2400	9600	404	2/24	Y	X	I/O	I/O	**
XCV600E		15552	187K	679K-986K	504K	48x72	3456	13824	512	2/24	Y	X	I/O	I/O	**
XCV1000E		27648	332K	1,146K-1,569K	768K	64x96	6144	24576	660	2/24	Y	X	I/O	I/O	**
XCV1600E		34992	420K	1,628K-2,189K	1062K	72x108	7776	31104	724	2/24	Y	X	I/O	I/O	**
XCV2000E		43200	518K	1,857K-2,542K	1240K	80x120	9600	38400	804	2/24	Y	X	I/O	I/O	**
XCV2600E		57132	686K	2,221K-3,264K	1530K	92x138	12696	50784	804	2/24	Y	X	I/O	I/O	**
XCV3200E		73008	876K	2,608K-4,074K	1846K	104x156	16224	64896	804	2/24	Y	X	I/O	I/O	**
XCV405E	Virtex Extended Memory Capabilities	10800	130K	1,068K-1,307K	710K	40x60	2400	9600	404	2/24	Y	X	I/O	I/O	**
XCV812E		21168	254K	2,569K-3,062K	1414K	56x84	4704	18816	556	2/24	Y	X	I/O	I/O	**

* I/Os are 5V tolerant
** 5 Volt tolerant I/Os with external resistor
X = Core and I/O voltage
I/Os = I/O voltage supported

Say hello to a new level of performance; the Spartan-II family now includes devices with over 200,000 system gates. You get 100,000 system gates for under $10, at speeds of 200 MHz and beyond, giving you design flexibility that's hard to beat. These low-powered, 2.5V devices feature I/Os that operate at up to 3.3V with full 5V tolerance. Spartan-II devices also feature multiple Delay Locked Loops, on-chip RAM (block and distributed), and versatile I/O technology that supports over 16 high-performance interface standards. You get all this in an FPGA that offers unlimited programmability, and can even be upgraded in the field, remotely, over any network.

Robust Feature Set

- Flexible on-chip distributed and block memory.
- Four digital Delay Locked Loops for efficient chip-level/board-level clock management.
- Select I/O Technology for interfacing with all major bus standards such as HSTL, GTL, SSTL, and so on.
- Full PCI compliance.
- System speeds over 200 MHz.
- Power management.

Extensive Design Support

- Complete suite of design tools.
- Extensive core support.
- Compile designs in minutes.

Advantages Over ASICs

- No costly NRE charges.
- No time consuming vector generation needed.
- All devices are 100% tested by Xilinx.
- Field upgradeable (remotely upgradeable, using Xilinx Online technology).
- No lengthy prototype or production lead times.
- Priced aggressively against comparable ASICs.

For more information see www.xilinx.com/products/spartan2

FPGA Product Selection Matrix

		DENSITY							FEATURES						
DEVICES	KEY FEATURES	Logic Cells	Maximum Logic Gates	Typical System Gate Range	Max. RAM Bits	CLB Matrix	CLBs	Flip-Flops	Max. I/O	Output Drive (mA)	PCI Compliant	1.8 Volt	2.5 Volt	3.3 Volt	5.0 Volt
XCS05	Spartan Family: High Volume ASIC Replacement/ High Performance/ SelectRAM Memory	238	3K	2K-5K	3K	10x10	100	360	77	12	Y	–	–	–	X
XCS10		466	5K	3K-10K	6K	14x14	196	616	112	12	Y	–	–	–	X
XCS20		950	10K	7K-20K	13K	20x20	400	1120	160	12	Y	–	–	–	X
XCS30		1368	13K	10K-30K	18K	24x24	576	1536	192	12	Y	–	–	–	X
XCS40		1862	20K	13K-40K	25K	28x28	784	2016	205	12	Y	–	–	–	X
XCS05XL	Spartan-XL Family: High Volume ASIC Replacement/ High Performance/ SelectRAM Memory	238	3K	2K-5K	3K	10x10	100	360	77	12/24	Y	–	–	X	*
XCS10XL		466	5K	3K-10K	6K	14x14	196	616	112	12/24	Y	–	–	X	*
XCS20XL		950	10K	7K-20K	13K	20x20	400	1120	160	12/24	Y	–	–	X	*
XCS30XL		1368	13K	10K-30K	18K	24x24	576	1536	192	12/24	Y	–	–	X	*
XCS40XL		1862	20K	13K-40K	25K	28x28	784	2016	224	12/24	Y	–	–	X	*
XC2S15	Spartan-II Family: High Volume BlockRAM Distributed RAM SelectI/O 4 DLLs	432	8K	6K-15K	22K	8x12	96	384	86	2/24	Y	–	X	I/O	*
XC2S30		972	17K	13K-30K	36K	12x18	216	864	132	2/24	Y	–	X	I/O	*
XC2S50		1728	30K	23K-50K	56K	16x24	384	1536	176	2/24	Y	–	X	I/O	*
XC2S100		2700	53K	37K-100K	78K	20x30	600	2400	196	2/24	Y	–	X	I/O	*
XC2S150		3888	77K	52K-150K	102K	24x36	864	3456	260	2/24	Y	–	X	I/O	*
XC2S200		5292	103K	71K-200K	130K	28x42	1,176	4704	284	2/24	Y	–	X	I/O	*

* I/Os are tolerant
X = Core and I/O voltage
I/Os = I/O voltage supported

XC9500 and CoolRunner CPLDs

Whether performing high-speed networking or power-conscious portable designs, Xilinx CPLDs provide you with a complete range of value oriented products.

XC9500 – Offers industry-leading speeds, while giving you the flexibility of an enhanced customer-proven pin-locking architecture along with extensive IEEE Std. 1149.1 JTAG Boundary-Scan support.

CoolRunner – Offers the patented Fast Zero Power (FZP™) design technology, combining low power and high speed. These devices offer standby currents of less than 100 microamps, operating currents 50-67% lower than traditional CPLDs, and pin-to-pin speeds of 5.0 ns.

WebPOWERED Software Solutions – Offer you the flexibility to target Xilinx CPLD and FPGA products on-line or on the desktop, including:

- **WebFITTER –** an on-line device fitting and evaluation tool that accepts HDL, ABEL, or netlist files and provides all reports, simulation models, and programming files, along with price quotes. Available to support all Xilinx CPLD products.
- **WebPACK ISE –** downloadable desktop solutions that offer free CPLD and FPGA software modules for ABEL/HDL synthesis and simulation, device fitting, and JTAG programming.

Through leading performance, free internet-based WebPOWERED software, and the industry's lowest power consumption, Xilinx has the right CPLD and FPGA for every designer's need.

For more information about Xilinx CPLD products, see:
www.xilinx.com/xlnx/xil_product_landingpage.jsp

CPLD Product Selection Matrix				Density		Features				
Core Voltage	CPLD Family	Device	Key Features	Macrocells	Max. I/O	Pin-to-Pin Delay (ns)	System Frequency	Individual OE Ctrl	JTAG	Ultra Low-Power
2.5 VOLT ISP	XC9500XV	XC9536XV	Best Pin-Locking JTAG w/Clamp High Performance High Endurance	36	36	3.5	278	√	√	–
		XC9572XV		72	72	4	250	√	√	–
		XC95144XV		144	117	4	250	√	√	–
		XC95288XV		288	192	5	222	√	√	–
3.3 Volt ISP	XC9500XL	XC9536XL	Best Pin-Locking JTAG w/Clamp High Performance High Endurance	36	36	5	222	√	√	–
		XC9572XL		72	72	5	222	√	√	–
		XC95144XL		144	117	5	222	√	√	–
		XC95288XL		288	192	6	208	√	√	–
	XPLA3	XCR3032XL	Ultra Low Power JTAG Increased Logic Flexibility	32	36	5	175	–	√	√
		XCR3064XL		64	68	6	145	–	√	√
		XCR3072XL		512	TBD	TBD	TBD	–	√	√
		XCR3128XL		128	108	6	145	–	√	√
		XCR3256XL		256	164	7.5	140	–	√	√
		XCR3384XL		384	220	7.5	127	–	√	√
5 Volt ISP	XC9500	XC9536	Best Pin-Locking JTAG High Endurance	36	34	5	100	√	√	–
		XC9572		72	72	7.5	83.3	√	√	–
		XC95108		108	108	7.5	83.3	√	√	–
		XC95144		144	133	7.5	83.3	√	√	–
		XC95216		216	166	10	66.7	√	√	–
		XC95288		288	192	10	66.7	√	√	–

XC18V
XC17V
XC17S

FPGA Configurations

Xilinx offers a full range of configuration memory devices optimized for use with Xilinx FPGAs. Our PROM product lines are designed to meet the same stringent demands as our high-performance FPGAs, taking full advantage of the same advanced processing technologies. In addition, they were developed in close cooperation with Xilinx FPGA designers for optimal performance and reliability.

XC18V00 – Our in-system reprogrammable family provides a feature-rich, fast configuration solution available today, and provides a cost-effective method for reprogramming and storing large Xilinx FPGA bitstreams. This family is JTAG ready and Boundry-Scan enabled for exceptional ease-of-use, system integration, and flexibility.

XC17V00/XC17S00 – Out low-cost XC17V and XC17S families are an ideal configuration solution for cost-sensitive applications. XC17V PROMs are pin-compatible with our XC18V family to allow for a cost-reduction migration path as your production volumes increase. The XC17S family is specially designed to provide a low-cost, integrated solution for our Spartan families of FPGAs.

Configuration PROMs for Virtex-E/Virtex-EM

Device	Configuration Bits	XC17xx Solution	XC18Vxx Solution	8-pin TSOP	20-pin PLCC	20-pin SOIC	44-pin PLCC	44-pin VQFP
XCV50E	630,048	17V01	18V01	X*	X	X	–	X**
XCV100E	863,840	17V01	18V01	X*	X	X	–	X**
XCV200E	1,442,106	17V02	18V02	X*	X*	X*	X**	X**
XCV300E	1,875,648	17V02	18V02	–	X*	–	X	X
XCV400E	2,693,440	17V04	18V04	–	X*	–	X	X
XCV405E	3,430,400	17V04	18V04	–	X*	–	X	X
XCV600E	3,961,632	17V04	18V04	–	X*	–	X	X
XCV812E	6,519,648	17V08	2 of 18V04	–	–	–	X	X
XCV1000E	6,587,520	17V08	2 of 18V04	–	–	–	X	X
XCV1600E	8,308,992	17V08	2 of 18V04	–	–	–	X	X
XCV2000E	10,159,648	17V16	2 of 18V04	–	–	–	X	X
XCV2600E	12,922,336	17V16	3 of 18V04 + 18V512	–	X***	–	X**	X
XCV3200E	16,283,712	17V16	4 of 18V04	–	–	–	X	X

* Available in XC17Vxx only.
** Available in XC18Vxx only.
*** Available in XC18V512 only.

Configuration PROMs for Virtex

Device	Configuration Bits	XC17xx Solution	XC18Vxx Solution	8-pin TSOP	20-pin PLCC	20-pin SOIC	44-pin PLCC	44-pin VQFP
XCV50	559,200	17V01	18V01	X*	X	X	–	X**
XCV100	781,216	17V01	18V01	X*	X	X	–	X**
XCV150	1,041,096	17V01	18V01	X*	X	X	–	X**
XCV200	1,335,840	17V01	18V02	X*	X*	X*	X**	X**
XCV300	1,751,808	17V02	18V02	–	X*	–	X	X
XCV400	2,546,048	17V04	18V04	–	X*	–	X	X
XCV600	3,607,968	17V04	18V04	–	X*	–	X	X
XCV800	4,715,616	17V08	18V04 + 18V512	–	X***	–	X**	X
XCV1000	6,127,744	17V08	18V04 + 18V02	–	–	–	X	X

* Available in XC17Vxx only.
** Available in XC18Vxx only.
*** Available in XC18V512 only.

Configuration PROMs for Spartan-XL/Spartan-II

Device	PROM Solution	8-pin PDIP	8-pin VOIC	20-pin SOIC	44-pin VQFP
XCS05XL	XC17S05XL	X	X	–	–
XCS10XL	XC17S10XL	X	X	–	–
XCS20XL	XC17S20XL	X	X	–	–
XCS30XL	XC17S30XL	X	X	–	–
XCS40XL	XC17S40XL	X	X	X	–
XC2S15	XC17S15A	X	X	X	–
XC2S30	XC17S30A	X	X	X	–
XC2S50	XC17S50A	X	X	X	–
XC2S100	XC17S100A	X	X	X	–
XC2S150	XC17S150A	X	X	X	–
XC2S200	XC17S200A	X	X	–	X

QML-Certified FPGAs and PROMs

The Xilinx QPro family of Radiation Hardened FPGAs and PROMs are finding homes in many new satellite and space applications. Both the XQR4000XL and XQVR Virtex products are being designed into space systems that will utilize reconfigurable technology. Numerous communications and GPS satellites, space probe, and shuttle missions are included on the growing list of programs that will be flying these devices.

The Virtex QPro family of High Reliability products is experiencing a high degree of success in the defense market. As designers find it more and more difficult to find components suitable for the harsh environments seen by defense systems, they are discovering that they can incorporate the functions of obsolete parts into Virtex QPro products. This has the added long term advantage of significantly reducing the costs of future re-qualifications, because their systems can retain consistent form, fit, and function through the use of Virtex QPro FPGAs. This cannot be achieved with costly and inflexible ASICs or custom logic.

Please visit http://www.xilinx.com/products/hirel_qml.htm for all the latest information about these products, including some new applications notes.

FPGA Product Selection Matrix

		DENSITY							FEATURES						
Device	Key Features	Logic Cells	Maximum Logic Gates	Typical System Gate Range	Max. RAM Bits	CLB Matrix	CLBs	Flip-Flops	Max. I/O	Output Drive (mA)	PCI Compliant	1.8 Volt	2.5 Volt	3 Volt	5 Volt
**XQR/XQ4013XL	XC4000 Series: Density Leadership/ High Performance/ SelectRAM Memory	1,368	13K	10K-30K	18K	24x24	576	1,536	192	12/24	Y	–	–	X	*
**XQR/XQ4036XL		3,078	36K	22K-65K	42K	36x36	1,296	3,168	288	12/24	Y	–	–	X	*
**XQR/XQ4062XL		5,472	62K	40K-130K	74K	48x48	2,304	5,376	384	12/24	Y	–	–	X	*
XQ4085XL		7,448	85K	55K-180K	100K	56x56	3,136	7,168	448	12/24	Y	–	–	X	*
XQV100	Virtex Family: Density/ Performance Leadership BlockRAM Distributed RAM SelectI/O 4 DLLs	2,700	32K	72K-109K	78K	20x30	600	2,400	180	2/24	Y	–	X	I/O	*
**XQVR/XQV300		6,912	83K	176K-323K	160K	32x48	1,536	6,144	316	2/24	Y	–	X	I/O	*
**XQVR/XQV600		15,552	187K	365K-661K	312K	48x72	3,456	13,824	512	2/24	Y	–	X	I/O	*
**XQVR/XQV1000		27,648	332K	622K-1,124K	512K	64x96	6,144	24,576	512	2/24	Y	–	X	I/O	*

* I/Os are tolerant
** XQR and XQVR devices are Radiation Hardened
X = Core and I/O voltage
I/Os = I/O voltage supported

(1) Selected XQ4000E/EX devices also available

QPRO QML-certified PROMs

		Package			
Device	Density	DD8	SO20	CC44	VQ44
XC1736D	36Kb	X			
XC1765D	64Kb	X			
XC17128D	128Kb	X			
XC17256D	256Kb	X			
XQR/XQ1701L*	1Mb		X	X	
XQR/XQ18V04*	4Mb			X	X**

* XQR devices are Radiation Hardened.
** XQ devices only.

Xilinx Intellectual Property Solutions

The Most Comprehensive and Highest Quality Solution in the PLD Industry

The Xilinx Intellectual Property Solutions Division offers the best selection of Intellectual Property solutions for a wide variety of industries and applications. Xilinx Smart-IP Technology delivers high performance, flexibility, and predictability, with optimized cores that give you both reduced cost and faster time to market.

LogiCORE™ Products – Licensed and supported by Xilinx, LogiCORE products such as parameterizable DSP building blocks and memory cores are included with the Xilinx CORE Generator software which is a component of your Xilinx Foundation Series or Alliance Series software.

AllianceCORE™ Products – A cooperative program with third-party IP suppliers who sell and support their cores directly with Xilinx customers. AllianceCORE products must meet criteria that ensure they deliver value and performance in a Xilinx device.

Reference Design Alliance Program - Xilinx proactively supports development of third-party system-level Reference Designs to provide fully functional, modular designs, that offer considerable development time savings.

XPERTS Program – The worldwide XPERTS Program provides over 70 consultants certified in delivering turnkey system designs for the Xilinx architecture, including PCI designs, new design methodologies, system-level design, along with IP customization and integration.

IP Delivery Tools – The Xilinx CORE Generator™ enables cataloging and generation of parameterized cores that are high performance, predictable, and integrated with our system-level design reuse tools; the cores are provided in VHDL and Verilog behavioral description languages.

The IP Center Internet portal, provides access to the latest LogiCORE and AllianceCORE products and reference designs via Smart Search; you can easily find the IP that you need at www.xilinx.com/ipcenter. Advanced function cores are available for IP evaluation and can be purchased via the IP Center.

Design Reuse – Download the "Reuse Field Guide Methodology for FPGA and ASIC Designs." Then use the Xilinx IP Capture Tool to package your IP with simulation models, testbenches, and PDF or HTML files. Then, you can catalog and share your IP using the CORE Generator.

The REAL PCI 64/66 – Parameterizable PCI cores, reference designs, prototyping boards, education, and Xilinx PCI XPERTS combined with a proven design and guaranteed timing make Xilinx PCI the lowest risk solution in the market

The Xilinx DSP Solution – Our exclusive FPGA partnership with MathWorks enables you to create complex, high performance DSP designs in a familiar environment with huge time to market advantages. Xilinx and its partners offer a complete set of cores for high-performance low-cost DSP implementation that provide:

- **Xtreme Flexibility –** Distributed DSP resources (such as look up tables, registers, multipliers, memory) and segmented routing allow optimized implementation of algorithms. Plus you get all the traditional FPGA benefits:
  - RAM-based FPGA technology, for fast and easy design changes
  - Fast time to market, to give you a competitive advantage
  - Field upgradeable systems (using IRL™), for extended product lifecycle
- **Xtreme Productivity –** The industry's first System Generator for Simulink® bridges the gap between FPGA and conventional DSP design flows, and features:
  - Unique constraint-driven Filter Generator, for performance/cost optimization
  - Power estimator tool (Xpower™), for very low-power DSP implementations
  - Eleven optimized DSP algorithms (cores) that cut development time by weeks
  - New DSP features added to the ChipScope ILA tool, rapidly reduces hardware debugging time
- **Xtreme Performance -** Table 1 illustrates the amazing performance you can achieve with Xilinx DSP.

Table 1 - Extreme Performance

Function	Industry's Fastest DSP Processor Core	Xilinx Virtex-E -08	Xilinx VIRTEX
MACs per second - Multiply and accumulate - 8 x 8-bit	4.4 Billion	128 Billion	**600 Billion**
FIR Filter - 256-tap, linear phase - 16-bit data/coefficients	17 MSPS @ 1.1 GHz	160 MSPS @ 160 MHz	**180 MSPS** @ 180 MHz
FFT - 1024 point, complex data - 16-bit real and imaginary comp.	7.7 µs @ 800 MHz	41 µs @ 100 MHz	**<1 µs** @ 140 MHz

Introducing the Xilinx 3.2i software release . . . the fastest in the industry

Watch your designs go supersonic. With the new Xilinx 3.2i release, you can place and route your next 100,000-gate design using a Spartan® FPGA in just one minute, or your next one-million-gate design using a Virtex™-E FPGA in only thirty minutes.

Faster tools improve your time-to-market advantage

With Xilinx's ultra-fast software you can beat your competition to market every time. When comparing the time it takes to complete your design, Xilinx place and route finishes up to 8 times faster for small designs, and up to 12 times faster for the most complex, high density designs.

See for yourself

Visit *www.xilinx.com/3_2i.htm* today and see how fast we can make your design. At Xilinx, we give you all the speed you need.

XILINX®
The Programmable Logic Company℠
www.xilinx.com

PN: 0010526-Q400