THE QUARTERLY JOURNAL FOR PROGRAMMABLE LOGIC USERS

Issue 31
First Quarter
1999

Xcell

COVER STORY
With VIRTEX FPGAs you can defy conventional logic and create the extraordinary

NEW TECHNOLOGY
Internet Reconfigurable Logic

APPLICATIONS
Creating Efficient Shift Registers

SUCCESS STORIES
Using the Verilog Flow

NEWS BRIEFS
Xilinx Achieves 1GHz Performance

DATASHEET
Virtex Family

XILINX®

FROM THE EDITOR

EDITOR
Carlis Collins
editor@xilinx.com
408-879-4519

SENIOR DESIGNER
Jack Farage

BOARD OF ADVISORS
Dave Stieg, Dave Galli
Mike Seither, Peter Alfke

PUBLICATION SERVICES
Ruddle Creative
111 N. Market St., Suite 715
San Jose, CA 95113
Tel: 408-297-3000 or
1-800-7RUDDLE
Web: www.ruddle.com

XCell

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

Imagination Springs to Life...

Each new generation of programmable logic technology moves us one step closer to the ideal development environment, and the new million-gate Virtex FPGAs and development tools from Xilinx are a quantum leap forward. This new creative technology, a combination of both device and software breakthroughs, is an extraordinary advancement that can dramatically increase your productivity and help quickly bring your new creations to life. This issue of Xcell will show you some of the creative possibilities offered by the new Virtex family.

Programmable logic technology is the most direct link between your imagination and the physical world of digital systems. When this link works quickly and efficiently, you are free to explore your creative genius; your ideas can flow smoothly into new, unexplored territories, unimpeded by the often slow and mundane chores of design implementation. The total Virtex solution is a combination of advanced new technologies, including device architecture, development software, and highly efficient manufacturing processes that work in harmony to give you an ideal development environment. *There is no faster or less expensive way to get your creations into full high-volume production.*

The Virtex architecture is revolutionary, with many new system-level features that make it much easier for you to put an entire system on a single device. It was designed from the beginning to work efficiently with the High-level Description Languages, cores, and advanced software algorithms that are available today, making it much easier for the software to implement your designs. This gives you the highest system performance, in the highest-density FPGAs ever produced, with compile speeds of over 200K gates per hour – a 4X runtime improvement over previous solutions.

The Virtex runtime performance breakthroughs are due, in part, to a very flexible and predictable routing structure that works closely with our latest place and route algorithms. Plus, the new Virtex library is optimized for the Virtex architecture, taking full advantage of its unprecedented capabilities. Simulation and compile times have also been reduced because the netlist is 40% smaller, making your designs easier to debug while also consuming less disk space.

The Virtex architecture is compatible with ASIC-like RTL coding styles. This means that if you are new to HDL, you will have a reduced learning curve and achieve higher design speeds using "generic" coding styles. The vector-based interconnect structure also gives you a more accurate post-synthesis timing estimation. This allows you to spend more time designing with the front-end synthesis and RTL verification tools and less time dealing with the back-end implementation tools.

To help you get a better understanding of the many new creative possibilities that are now available to you, we have included an abbreviated Virtex family data sheet in this issue, starting on page 41. For the complete data sheet and application notes, visit: www.xilinx.com/products/virtex.htm

Thanks for reading Xcell. What do you think of the new format? E-mail your comments to editor@xilinx.com.

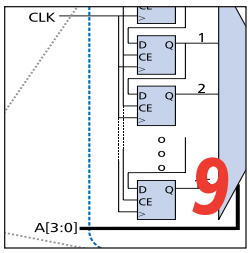
ARTICLES

COVER STORY

Virtex FPGAs allow you to design complete, highly complex, high-performance systems in a single programmable device.

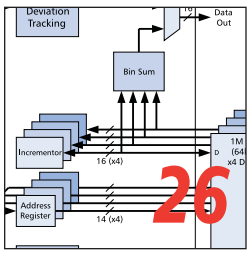
NEW TECHNOLOGY

Internet Reconfigurable Logic for creating Web-enabled devices is now available with Silicon Xpresso™ for use with Virtex FPGAs.



APPLICATIONS

Creating efficient multi-tap shift registers using the Virtex look-up-tables. A 16-bit shift register requires only 0.25 CLBs!



SUCCESS STORIES

A real life example of a Verilog design that runs as fast as a schematic-based design; a testimonial to an excellent tool flow and XC4000XL FPGAs.

NEWS BRIEFS

Xilinx Achieves 1GHz Performance

Inside This Issue:

Xcell

The New Virtex FPGA Family

Much More Than Just a Million Gates...

Now, for the first time, you can create complete, highly complex, high-performance systems in a single programmable device. Using our new Virtex FPGAs and our new high-speed development tools, your creative ideas will reach full production more quickly, more easily, and less expensively than ever before. This is a revolution in logic design.

by Carlis Collins, Managing Editor of Corporate Communications, Xilinx, editor@xilinx.com

These are fast moving times; every day that you waste in unnecessary development, debugging, and manufacturing, is lost revenue and an invitation to your competitors. You have to manage your development time, and wisely leverage all of the available resources, to remain competitive; you have to think in extraordinary ways. That's the motivation behind our new Virtex family.

From the beginning, we coordinated our research and development, in both device and software technology, to create a complete, fully integrated solution for digital logic design. The Virtex solution is extraordinary and it's the wisest use of your time and resources for gaining a competitive advantage.

Virtex is Extraordinary

The Virtex architecture is a new concept in programmable logic technology, one that is both evolutionary and revolutionary. We built on our previous success and added many new features, ones that you requested, that allow you to create a true system on a single chip. These new ASIC-like features, combined with our new fast-compile software and lower pricing, now give you the best solution for all but the highest-volume applications. Designs that once required custom ASICs, can now go straight to production using Virtex FPGAs, saving you a lot of time, trouble, and expense.

There are four key features that make all of this possible:

Virtex Overview

The Virtex architecture includes many more features than can be adequately described in this article, that's why we included an abbreviated Virtex data sheet, starting on page 41. However, here's a review of the Virtex FPGA highlights:

- Nine Virtex devices, 50,000 to 1,000,000 system gates.
- 200MHz on-chip, 160MHz I/O performance.
- Delay-Locked Loop clocking.
- Flexible on-chip RAM.
- Support for 16 I/O standards.
- Core-friendly, fast, predictable routing.
- Very fast device programming.
- Dedicated carry logic for high-speed arithmetic.
- Dedicated multiplier support.
- Cascade chain for wide-input functions.
- Internal 3-state bussing.
- IEEE 1149.1 boundary-scan capability.
- 66-MHz/64-bit PCI compatible.
- Hot-swappable for Compact PCI.
- Die temperature sensing device, on-chip.
- 0.22µ, 5-layer metal process.
- 100% factory tested.
- Prices starting at less than $10 (50K gate XCV50, in high volumes).
- 1M and 300K gate devices available today.

1. Delay-Locked Loops (DLLs)

Associated with each global clock input buffer is a fully digital Delay-Locked Loop (DLL) that eliminates skew by monitoring the input clock and the distributed clock, automatically adjusting a clock delay element, as described on page 50. This closed-loop system effectively eliminates clock-distribution delay, and can double your overall performance.

In addition, the DLL can provide four quadrature phases of the source clock, and can double the clock frequency, or divide it by 1.5, 2, 2.5, 3, 4, 5, 8, or 16. The DLL can also operate as a clock mirror; by driving the output from a DLL off-chip and then back on again, it can thus be used to de-skew board-level clocks between multiple Virtex devices.

2. Enhanced Memory Architecture

The Virtex architecture supports three types of memory:

- **Block RAM** - Each Block SelectRAM+™ cell, as illustrated on page 47, is a fully synchronous dual-port 4096-bit RAM with independent data and control signals for each port. The data widths of the two ports can be configured independently, providing built-in bus-width conversion. Table 5 on page 47 shows the amount of Block SelectRAM+ memory that is available in each Virtex device; the smallest Virtex device (XCV50) has eight blocks and the largest device (XCV1000) has 32 blocks.
- **Distributed RAM** - Virtex function generators are implemented as 4-input look-up tables (LUTs) as shown on page 47. Each LUT can be used as a 16x1-bit synchronous RAM, and the two LUTs within a slice can be combined to create a 16x2-bit or 32x1-bit synchronous RAM, or a 16x1-bit dual-port synchronous RAM. The LUT can also be used as a 16-bit shift register, ideal for capturing high-speed or burst-mode data.
- **200MHz Access to External Memory** - Using the built-in SSTL3 interface capability you can directly access external high-speed SDRAM, in addition to other types of RAM and ROM.

3. Simultaneous Interface to Multiple I/O Standards

The Virtex SelectI/O™ capability supports 16 different I/O standards, as shown on page 44. Each device includes eight separate I/O banks that can be independently configured for a different I/O standard allowing you to interface with up to eight different voltage and signal standards, simultaneously. All outputs are PCI compliant as well, giving you full 66MHz/64-bit PCI capability.

4. System Integration

To fill a million-gate device you will probably need to use predefined intellectual property, or cores. The Virtex architecture is designed to make this very easy because abundant routing resources of various lengths allow our LogiCORES™ to be placed anywhere, in any combination, with fast, predictable performance. Cores work the same way, every time, in Virtex designs, so you don't have to worry about "tweaking" or optimizing your final design.

Designed for Speed

The Virtex family was designed for speed, in all areas, including device development and programming. For example, with the Virtex family you can configure devices approximately 40 times faster than previous FPGAs. This means, for example, that you can completely configure our 100,000 gate XCV100 in less time than the vertical retrace on your video monitor. In addition, you can partially reprogram Virtex devices while they are operational. This opens a world of new possibilities such as Internet Reconfigurable Logic (IRL) as described on page 6.

Design development is faster as well, because our Alliance Series™ and Foundation Series™ tools are optimized for the Virtex architecture, which was optimized for use with High-level Description languages. This means that your compile times are significantly reduced, a real advantage for very large designs, helping you quickly evaluate design modifications.

Our development tools cover every aspect of design development from behavioral, schematic, and HDL design entry; through simulation, automatic design translation, and implementation; to the creation, downloading, and verification of your configuration bit stream. From top to bottom our software is easy to use and produces fast, accurate results.

Conclusion

FPGAs are no longer just for prototyping or low-volume logic replacement. Now you can benefit from the ease-of-use and time-to-market advantages of FPGAs for higher volume, system-on-a-chip applications. With the new Virtex family, your design possibilities are almost unlimited. Σ

For more information on Virtex see http://www.xilinx.com/products/virtex.htm.

New Internet Reconfigurable Logic

for Creating Web-enabled Devices

Continuing its Internet-based Silicon Xpresso initiative, Xilinx recently announced Java-based tools and technology that will revolutionize the development and deployment of Internet network appliances, using Virtex FPGAs.

by Wallace Westfeldt, Product Manager for Internet Reconfigurable Logic, Xilinx, wallace.westfeldt@xilinx.com

Internet Reconfigurable Logic (IRL) technologies will be the basis for new products that can be dynamically upgraded with both software and logic on the customer premises. The concept of IRL will revolutionize network-connected products.

Xilinx IRL solutions are targeted at emerging network appliances such as multi-use set top boxes, games, security systems, and process controllers. In addition, IRL will be deployed in network equipment such as ATM, cellular base stations, and satellite communications systems. The hardware for these Virtex-based products can be upgraded over the Internet to add new features or capabilities.

The combination of three fundamental technologies will empower the design of radically new IRL products: pervasive networking, Java technology, and reconfigurable FPGAs such as Virtex. These technologies will allow you to create products that can be enhanced with new features, after installation, at the customer site.

To support Internet Reconfigurable Logic, Xilinx also unveiled the new JBits™ API and ChipScope™ tools.

JBits API

The JBits API is a new Java-based tool set, or application programming interface (API), that allows you to write information directly to a Xilinx FPGA to carry out the logic operations that were designed for it. The JBits API permits the FPGA bitstream to be modified quickly, allowing fast reconfiguration of the device. With Virtex FPGAs, the JBits API can partially or fully reconfigure the internal logic.

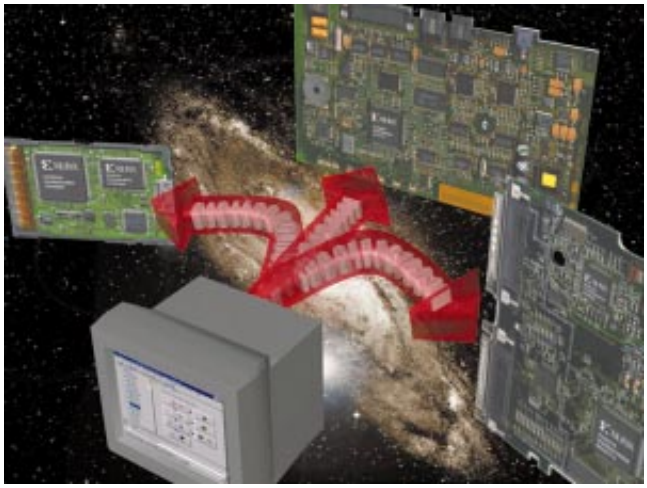
The Virtex architecture allows partial reconfiguration to occur while maintaining the logic on the remainder of the device. The JBits API also makes it possible to integrate the operations of the FPGA with other system components such as an embedded processor, a graphics coprocessor, or any digital peripheral device.

JBits applications can use the "Java API for Boundary Scan," unveiled by Xilinx in September, for platform-independent device configurations deployed locally or remotely over the Internet. These applets can be control programs, consumer interface programs, or updates. Previously, Java applets were only used to send software updates via the Internet. The JBits API now makes its possible to create Java logic applets that can be used to send new hardware updates as well.

ChipScope

ChipScope is a portable, interactive debugging tool, written in Java, that allows you to examine the operation of Xilinx FPGA circuits. The ChipScope tool, like the JBits API, is Internet enabled, allowing for remote debugging of IRL-based products. Designed to show data flow, the ChipScope tool displays the internal states of all FPGAs in the system.

The ChipScope tool simplifies the tedious design verification required for system-on-a-chip designs. A waveform display permits both bit-level signal and multi-bit busses to be viewed. Moreover, a remote access feature lets multiple users communicate with the hardware over a network for Internet team-based design. ChipScope also allows you to functionally



The combination of three fundamental technologies will empower the design of radically new IRL products: pervasive networking, Java technology, and reconfigurable Virtex FPGAs.

view tagged, secure, intellectual property. This enables discrete core manipulation for system-on-a-chip design.

The JBits and ChipScope tools follow two earlier Silicon Xpresso announcements introducing the Java API for Boundary Scan and the Webfitter tool, plus the release of new Xilinx Foundation 1.5i design tools that provide instant access to Web-enabled design.

What People are Saying about IRL

"This third phase of our Silicon Xpresso initiative is focused on advanced technology that will help electronic equipment manufacturers bring Internet-enabled products to their customers' businesses and consumers. International Data Corp. projects that this exploding marketplace will account for an installed base of more than 500 million Web-enabled hardware devices by 2003," said Rich Sevcik, senior vice president of software, cores and support solutions at Xilinx. "We're confident that the system-level features of our new Virtex FPGAs, combined with the innovative JBits and ChipScope tools, will provide the necessary foundation to help bring IRL applications into the mainstream. The Java programming language and the Internet will play a key role in the development of reconfigurable end products whose hardware literally can be upgraded over the network."

"We have utilized many of these FPGA-based reconfigurable concepts in our ATM switches at IBM," said Jean Calvignac, an IBM Fellow with IBM's Networking Hardware Division in Research Triangle Park, N.C. "Our customers have been pleased to see product updates occur automatically via the network. These seamless updates have included both software and hardware changes. With its tools for Internet Reconfigurable Logic, Xilinx is broadening the appeal of this exciting technology."

"Compaq's PCI Development Platform embodies many of the Internet Reconfigurable Logic concepts which can benefit our customers today," said Gene Nelson, vice president, Compaq Custom Systems. "We are pleased that Xilinx selected Compaq's Xilinx FPGA-based 'PCI Development Platform' in the development of JBits and ChipScope tools, saving them time and development costs by providing an open and flexible reconfigurable board on the PCI bus. By Xilinx providing standardized support, we will be able to take these IRL concepts much further in future FPGA-based products."

A number of other industry leaders, including, Siemens, Sun Microsystems Inc., and Synplicity, announced support for IRL services and products. In addition, two longtime Xilinx development partners in the reconfigurable logic arena, Annapolis Micro Systems Inc. and Virtual Computer Corp., announced Virtex-based development systems that support IRL.

Conclusion

Internet Reconfigurable Logic is a revolutionary new way to create network appliances that can easily be re-wired to perform many different applications. Xilinx is creating the tools, the software, and the device technology that makes it all happen.

Beta versions of the JBits and ChipScope tools will be available in the first quarter of 1999, with pricing to be announced at that time. For more information on Silicon Xpresso, visit www.xilinx.com/products/software/sx/sxpresso.html

Reed-Solomon Cores Excel in The Virtex Architecture

Integrated Silicon Systems has ported their Reed Solomon cores to the new Virtex architecture, which requires fewer CLBs while providing 28% better performance.

by David Mann, Marketing Communications, Integrated Silicon Systems, dmann@iss-dsp.com

Last year, Integrated Silicon Systems (ISS), a Xilinx AllianceCORE™ partner, began porting the DSP cores from their expansive semiconductor IP portfolio to the new Virtex FPGA technology. The Reed-Solomon Encoder and Decoder cores from ISS offer compact, high-performance FPGA implementations for applications from Digital Video Broadcasting (DVB) to data storage/retrieval systems (HDD, CD-ROM).

The ISS DVB Reed-Solomon cores, previously implemented in the XC4000 family, were developed using HDL techniques to make them portable across a range of technologies (such as ASIC and PLD) and processes (from 0.6µ to 0.18µ). The HDL source code originally produced by ISS for the Reed-Solomon cores had to be modified to use the new features of the Virtex device architecture. In particular, the SelectRAM™ previously used for memory elements was replaced with Block RAM. Many of the instantiated components in the Reed-Solomon core were automatically generated using LogiBLOX,™ the Xilinx component generation tool.

The Reed-Solomon decoder core is partitioned into modules as shown in **Figure 1**.

ISS DVB Reed-Solomon Decoder Block Diagram

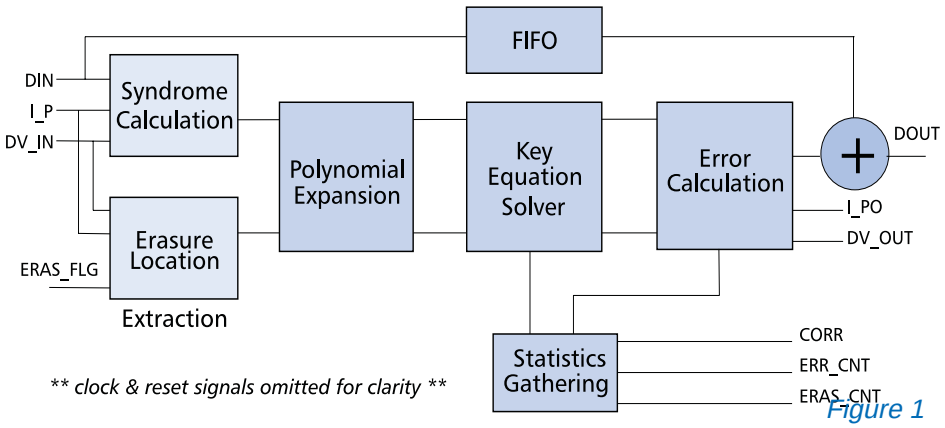


Figure 1

Targeted at the Xilinx XC4000XL series, a DVB-compliant R-S Encoder requires 105 CLBs. When retargeted at the Virtex architecture, the same core required 83 CLB slices (less than 42 Virtex CLBs). The performance of this core, as dictated by critical path timing constraints, was up from 55MHz (XC4005XL-3) and 72MHz (XC4005XL-1) to 82MHz (Virtex). So on a first pass in Virtex technology, performance was up 28%.

Conclusion

Trial implementations of the ISS-designed DVB Reed-Solomon AllianceCOREs in Virtex technology resulted in an immediate performance improvement of about 28%. Further work on optimization for the Virtex architecture is expected to provide even greater gains in performance.

About Integrated Silicon Systems

ISS is an established independent provider of reusable semiconductor IP for a wide range of applications. This silicon-proven IP is available to Xilinx FPGA designers, through the Xilinx AllianceCORE program, in the form of synthesizable or post-synthesis netlist cores. The package of deliverables for these AllianceCOREs includes design files (.xnf and constraint files) and test files (including testbenches and vectors).

For more information, e-mail info@iss-dsp.com

Creating Efficient Multi-Tap Shift Registers

The Virtex LUTs can be configured as shift registers whose depth is determined by the four inputs to the LUT.

by Paul Gigliotti, Field Applications Engineer, Xilinx, paul.gigliotti@xilinx.com

Using the method described here, creating a 16-bit shift register requires only 0.25 CLBs, while creating a 16-bit shift register using flip-flops would use four CLBs. The depth of the shift register is dynamically configurable by simply changing the inputs to the LUT.

This mechanism can also be used to "snoop" into the shift register. Between shift cycles, the LUT inputs act as addresses into the shift register, allowing various points within the shift register to be examined. This is useful for various DSP filtering applications, as well as pattern/waveform detection.

Virtex LUT-based Shift Registers

A 16-deep by eight-bit wide shift register requires eight LUTs, which is just two Virtext CLBs. The LUT's address lines are tied to "F" (hex), and the control signals are wired in parallel. The shift register's D-Q pair (see **Figure 1**) are tied to a specific bit of the input/output bus. For depths greater than 16, the shift register LUTs can be cascaded, with the address lines for the low order banks tied to F (hex), and the final bank tied to modulo16 (depth).

Virtex LUT as a Shift Register

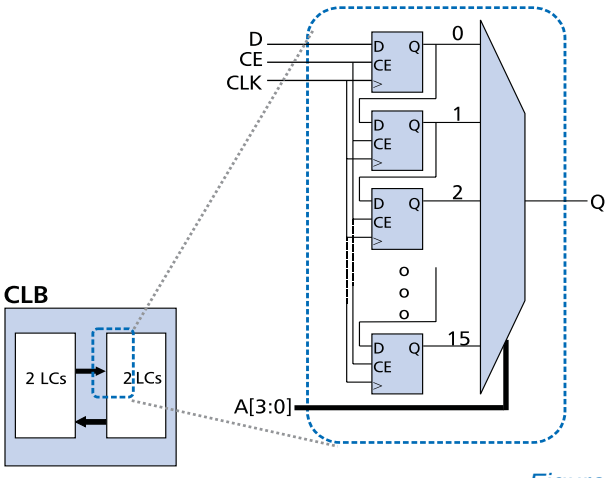


Figure 1

Snooping Into a Shift Register

As can be seen in **Figure 1**, the output of the MUX is an asynchronous read of the LUT. Snooping is accomplished by doing reads of the LUT between shift cycles. The waveform in **Figure 2** assumes that a 16-deep shift register is needed, but the design also requires that the data at Tap N be available. The design is run at twice the frequency, to allow the snooping to occur, as follows:

Snooping

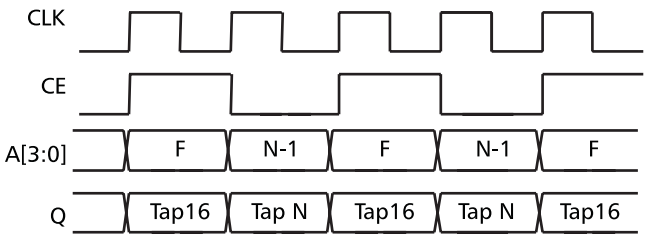


Figure 2

Dynamically Resizable Multi-Tap Shift Register

The design in **Figure 3** can be extended by cascading the multiple shift register banks to add additional depth. When the clock enable is high, a shift cycle is occurring, and the data is shifted from one bank to the next. To extend the design further, more snoop cycles can be added between shift cycles. For example, running the clock at 4X allows for one shift cycle and three snoop cycles. Note, we can only perform three snoops per shift register bank.

Tap locations can be "moved" into other shift register banks, by controlling the depth on each bank. For example, if I need to snoop on taps 3, 7, 9, and 11, the first shift register bank must be 10 deep, at most, so that Tap 11 resides in the

Dynamically Resizable Multi-Tap Shift Register

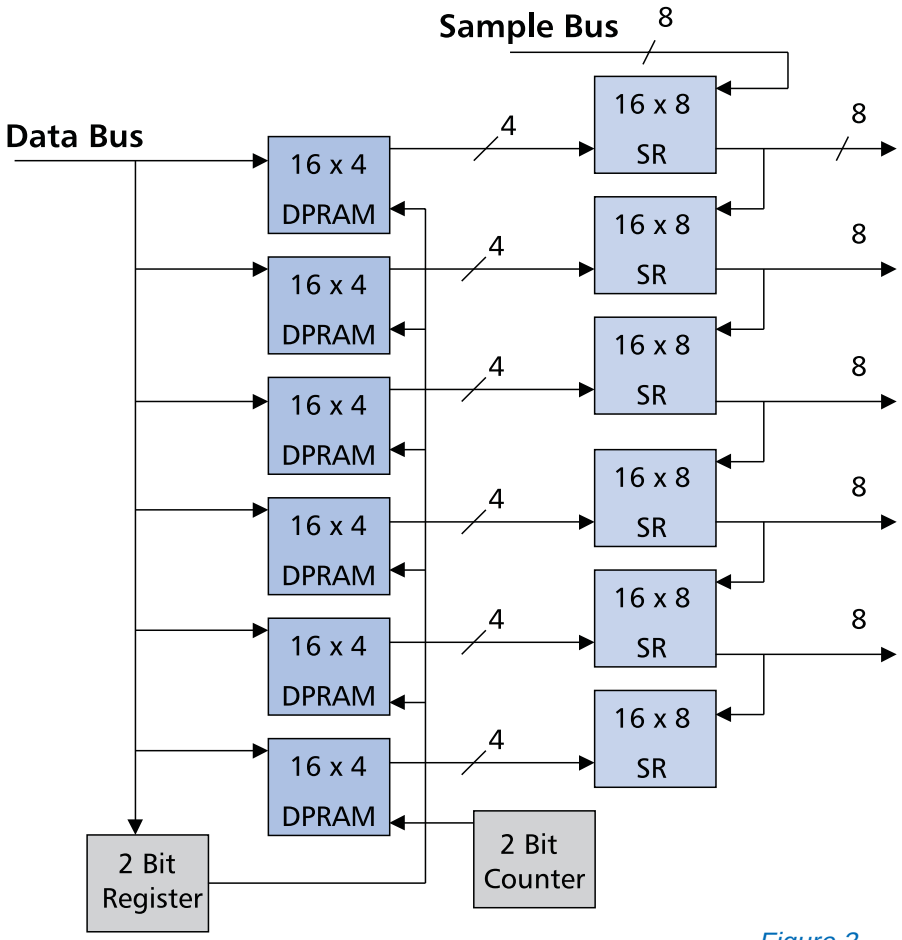


Figure 3

second bank of RAM. Controlling the shift and snoop points is most efficiently accomplished by using ROM blocks to store the look-up values. The block diagram in **Figure 3** has a maximum depth of 96 taps, and a maximum of 18 tap points.

Finally, the whole structure can be made "programmable" by storing the shift/snoop points in distributed dual port RAM, with the read port controlling the shift registers, and the write port mapped into a processor's memory space.

Conclusion

The above design requires 72 LUTs and four flip-flops, for a total of 19 CLBS. A more traditional approach, using flip-flop based shift registers, would require 196 flip-flops or 49 CLBs, just for the shift register portion of the design alone. Also, note that only four locations in the dual port RAMs are required to control the one shift and three snoops. By placing the two MSB address bits under processor control, it is possible to cache addition shift/snoop configurations into the dual port RAMs, and "hot swap" in the new settings.

Designing With Large, Fast Programmable Logic Devices

The FPGA industry has achieved a major milestone with the availability of extremely large, high-performance devices. To successfully take advantage of the design opportunities presented by these new FPGAs, today's designers must be armed with the latest tools and techniques.

by Jackie Patterson, FPGA Marketing Program Manager, Synopsys, jackiep@synopsys.com

Historically, application-specific chips comprised a relatively small portion of an electronic system. Standard function chips, such as bus interfaces, clock controllers, memory, device controllers, and microprocessors, made up the majority of the design. However, improvements in chip capacity, combined with the decreasing end-product form factor, is pushing us to much higher levels of integration.

Xilinx is a leading innovator in high-density, high-performance FPGAs. Customers have voiced a keen interest in the Virtex family of FPGAs, not only for their speed and density, but also for the new architectural features. On-board RAMs give you greater integration and the incentive to pull more of the system function onto the chip, while the wide variety of I/O types helps to integrate the chip into your system, eliminating external level translators.

Until recently, this integration capability was only available to ASIC designers. However, system-on-a-chip design opportunities have increased dramatically with the introduction of these affordable Virtex FPGAs possessing the density, architectural features, and software support you need.

Applications

Depending on your end application, you can plan to design-in these chips in a variety of ways. You can push the bulk of the system functionality onto a single chip, or divide the system according to flexibility requirements, split between an ASIC and an FPGA. For example, an ATM system may have a very complex 50K- to 100K-gate control logic requirement, prone to design error and specification change. This portion is best mapped into an FPGA.

Industries where standards are in flux are also prime candidates for high-density, high-performance FPGAs, where there is a need for last minute product customization. The bulk of the system (a consumer graphics product for example) can be realized in masked silicon, while an FPGA customizes it for a variety of different market segments, perhaps starting with PCI interfaces and moving to USB or AGP.

Design Techniques and Technologies

Silicon advances are not the only factor enabling systems-on-a-chip. Taking advantage of the increased chip capacity within the real world constraints of product market windows and performance specifications requires a whole host of supporting technologies, such as those available from Synopsys. Some key design techniques for high gate count, high-speed FPGA designs are:

- Hardware Design Languages (HDLs) and synthesis above 10,000 gates for productivity design re-use.
- Optimizing the full design flow, including FPGA place and route and requirements for compatibility with the ASIC design flow.
- Use of timing analysis to pinpoint critical timing issues early.
- Precise design methodologies and hierarchical synthesis support for team design.

Summary

The age of system-on-a-programmable-chip has become a reality, and the Virtex family from Xilinx is a leading architecture in that market. The Virtex architecture contains many powerful features, which lend themselves to smaller, faster systems. The Synopsys and Xilinx R&D teams continue to work closely together to develop a combined solution of powerful design tools, intellectual property, and high-density, high-performance FPGAs to bring you unprecedented opportunities. Σ

For more information on Synopsys visit www.synopsys.com

Efficient Multi-Channel SERIAL to PARALLEL Converter

A significant size reduction is accomplished by taking advantage of the dual port mode of the distributed SelectRAM available in Spartan,™ XC4000, and Virtex devices.

by Paul Gigliotti, Field Applications Engineer, Xilinx, paul.gigliotti@xilinx.com

Often, data from remote sensors is sent back to a central location in serial format, to reduce the cost of cabling. The data streams then need to be converted into a parallel format for processing. This is typically accomplished using a shift register, followed by a holding register, thus requiring two flip-flops per bit.

For example, eight 16-bit channels of data would require eight 16-bit shift registers and eight 16-bit registers, for a total of 256 flip-flops. This requires 128 CLBs in a Spartan or XC4000 family device, or 64 CLBs in a Virtex device. The following implementation requires only 30.5 CLBs in a Spartan or XC4000, and only 15.25 CLBs in a Virtex device, and requires 128 (8x16) clock cycles for one complete scan.

The ability of Spartan, XC4000, and Virtex FPGAs to support distributed blocks of RAM allows for significant design efficiencies.

Double Buffering for "free"

The dual port RAM is used to both convert the data from serial to parallel format, as well as provide for double buffering. The data is written into one half of the memory, and read out from the other half. The double buffering is accomplished by "ping-ponging" between the memory halves.

Memory Map

The 16 dual port 16x1 RAMs are structured such that each DPRAM contains a specific bit of data from all channels. For example, DPRAM 0 contains all the bit zeros, DPRAM 1 contains all the bit ones, and so on. As discussed above, the upper/lower half contains the data currently being converted, while the lower/upper half contains the data previously converted.

Double Buffering

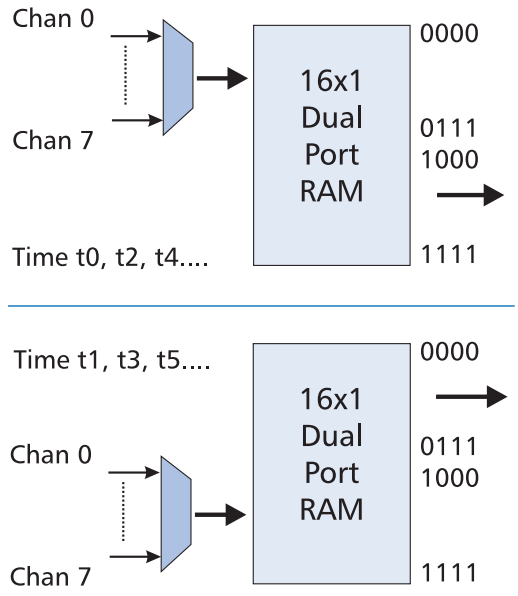


Figure 1

Memory Map

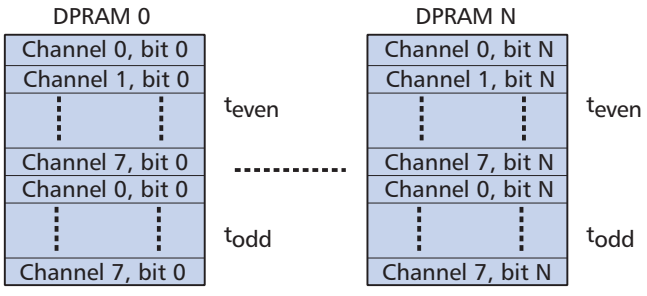


Figure 2

Control Logic

The control logic consists of an eight-bit counter, an 8-to-1 multiplexer, and a 4-to-16 decoder. In a Spartan or XC4000 device, the control logic uses four CLBs for the counter, 2.5 CLBs for the multiplexer, and eight CLBs for the decoder, for a total overhead of 14.5 CLBs. The three LSBs of the eight-bit counter drive the select lines of the 8-to-1 mux, as well as the write port's three LSB address lines. The MSB of the write port is driven by the inversion of the counter's MSB. The four MSBs of the counter drive the read port's addresses. Finally, bits 3, 4, 5, and 6 of the counter source the 4-to-16 decoder, with the outputs of the decoder driving the write enables of the dual port RAMs.

Conclusion

The ability of Spartan, XC4000, and Virtex FPGAs to support distributed blocks of RAM allows for significant design efficiencies. The above design can be extended to 16 channels by adding another bank of sixteen dual port RAMs, increasing the CLB count by 16, rather than an increase of 128 CLBs using a more traditional approach.

The Code

```
entity ser2par is
port (
CLK: in STD_LOGIC;
CHANNEL: in STD_LOGIC_VECTOR (7 downto 0);
DATA: out STD_LOGIC_VECTOR (15 downto 0)
);
end ser2par;

architecture ser2par_arch of ser2par is

signal DATA_BIT:STD_LOGIC;
signal WRITE_ADDRESS:STD_LOGIC_VECTOR(3 downto 0);
signal COUNTER:STD_LOGIC_VECTOR(7 downto 0);
signal WRITE_ENABLE:STD_LOGIC_VECTOR(15 downto 0);

component RAM16X1D
PORT(
A: IN std_logic_vector(3 DOWNTO 0);
DI: IN std_logic;
WR_EN: IN std_logic;
WR_CLK: IN std_logic;
DPO: OUT std_logic;
DPRA: IN std_logic_vector(3 DOWNTO 0));
end component;

begin

process (CLK)
begin
 if CLK='1' and CLK'event then
   COUNTER <= COUNTER + 1;
 end if;
end process;

DECODER:
process (COUNTER)
begin
 for I in 0 to 15 loop
   if (COUNTER(7 downto 4) = I) then
       WRITE_ENABLE(I) <= '1';
   else
       WRITE_ENABLE(I) <= '0';
   end if;
 end loop;
end process;

MUX:
process (COUNTER)
begin
 case COUNTER(2 downto 0) is
   when "000" => DATA_BIT <= CHANNEL(0);
   when "001" => DATA_BIT <= CHANNEL(1);
   when "010" => DATA_BIT <= CHANNEL(2);
   when "011" => DATA_BIT <= CHANNEL(3);
   when "100" => DATA_BIT <= CHANNEL(4);
   when "101" => DATA_BIT <= CHANNEL(5);

   when "110" => DATA_BIT <= CHANNEL(6);
   when others => DATA_BIT <= CHANNEL(7);
 end case;
end process;

WRITE_ADDRESS <= (not(COUNTER(7)) & COUNTER);

RAM:
for I in 0 to 15 generate
RAMBANK:
 RAM16X1D port map (
   A => WRITE_ADDRESS,
   DI => DATA_BIT,
   WR_EN => WRITE_ENABLE(I),
   WR_CLK => CLK,
   DPO => DATA(I),
   DPRA => COUNTER(7 downto 4)
   );
end generate RAM;
end ser2par_arch;
```

New ASIC Estimator – For Cost Modeling

To help you choose the best development process, Xilinx is now providing, as part of the Silicon Xpresso framework, a new on-line cost modeling tool called the Xilinx ASIC Estimator.

by Rob Schreck, HardWire Product Manager, Xilinx, rschreck@xilinx.com

For many designs it's smart to use FPGAs in volume production, even when the unit cost is higher than ASICs. The time-to-market and ease-of-use advantages of FPGAs often far outweigh the cost advantage that ASICs are traditionally known for. This is proven by a McKinsey study showing that late market entry has a larger effect on profits than development or production cost overruns, especially in very competitive markets.

The Xilinx ASIC Estimator will help you compare various logic development alternatives and decide which to use. The following chart gives some comparisons of the different alternatives:

Comparing Options

	FPGA	Gate Array	Std. Cell
NRE	None	Low-to-Medium	High
Unit Cost	Highest	Low-to-Medium	Lowest
Development time	Lowest	Medium-to-High	Highest
Flexibility	Highest	Low	Lowest

Figure 1

To help you assess the advantages and disadvantages of various production scenarios, the ASIC Estimator allows you to use over 30 different variables to model your project costs and schedule. You can use "what-if" analysis to see how differing NRE costs, unit costs, development time, and production inflexibility can impact overall profitability of the product you are planning or developing.

The ASIC Estimator provides bar charts that show the total cost of using ASICs or FPGAs (**Figure 2**). In addition, a break-even chart shows what unit production volumes are required to justify using an ASIC (**Figure 3**).

Cost of Ownership

Figure 2

Break-even Bar Chart

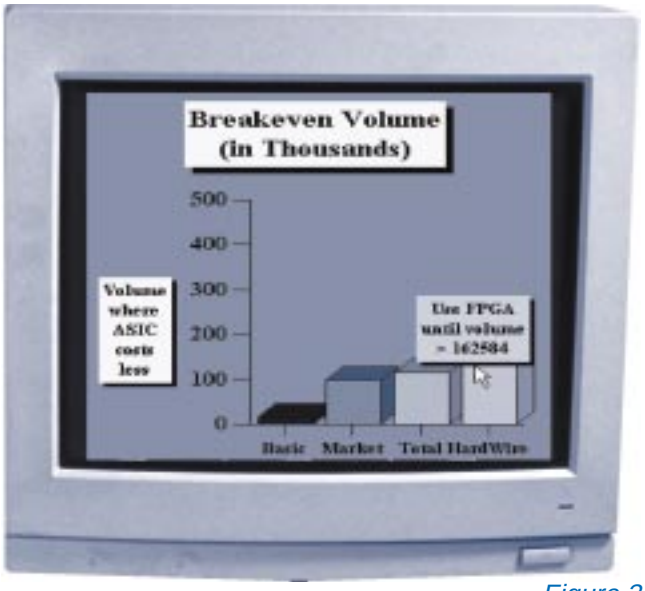


Figure 3

Project Timeline

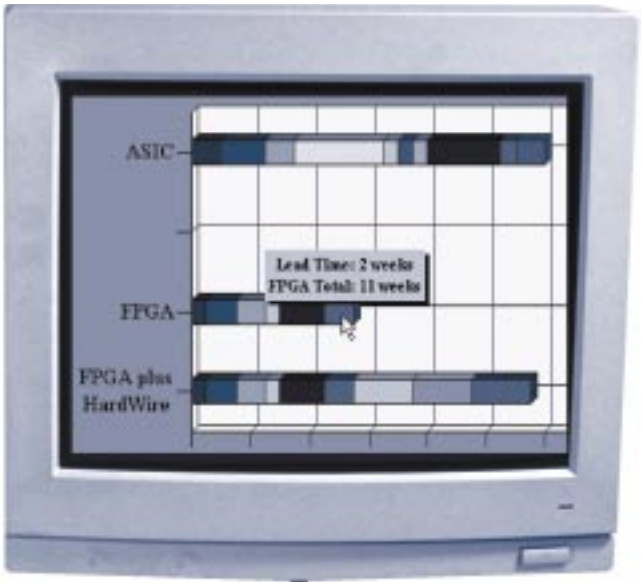


Figure 4

The Advantages of FPGAs

Many companies have a history of using FPGAs only for prototyping and generating proof of concept, eventually converting their designs to ASICs. However, there are often significant delays in the conversion process, and with those delays come the costs of:

- Being late to market and disappointing customers.
- Being late to market and allowing the competition to enter the market first.
- Letting some other company establish a standard based upon their concept.

A McKinsey study showed that late market entry has a larger effect on profits than development or production cost overruns, especially in very competitive markets.

FPGAs offer a very flexible solution for both prototyping and for early production. With FPGAs, you can design a system, then quickly move to environmental, alpha, beta, and even field tests. You don't have to wait for prototype fabrication and assembly. In addition, you can often be first to market, satisfying customer demands, and establishing a market position. Plus, you can even use the same FPGA design as a platform for the next generation product, often without redesigning your PC board. Using Xilinx Virtex FPGAs for prototype system development and production, you get the fastest time to market compared to gate arrays and standard cell designs (**Figure 4**).

Summary

When you use the Xilinx ASIC Estimator, you will see that it takes a surprisingly large number of units to justify the costs and delays of using ASICs. This has never been more true now that Xilinx has introduced the Virtex product line. Σ

You can access the Xilinx ASIC Estimator and our "Cost of Ownership" article (describing the cost models and how the Xilinx ASIC Estimator can help you make the right decision about using ASICs or FPGAs in your next project) at: http://www.xilinx.com/products/hardwire/hardwirehome.htm

FPGA-Link System Level Integration of FPGAs

FPGA-Link from TRILOGIC is a product that extracts information from "post-route" FPGA design files and automatically creates all the necessary symbols, schematics, and hierarchical associations to integrate the FPGA into a system-level design ready for simulation.

by Cammie Salmon, EDA Products and Services Manager, TRILOGIC, csalmon@trilogic.com

Once your FPGA design is complete you need to incorporate the FPGA into a system-level schematic design and manage the effect of FPGA design changes. This task may seem trivial, but it involves many tedious processes to ensure correct integration for system-level simulation and PCB design. For each FPGA in the system, a "PCB-level" symbol must be constructed and a simulation model created or integrated. When changes, such as pin reassignments, are made to the FPGA, the resulting changes must be reflected and verified on the schematic and symbol.

To effectively manage the process of design changes, FPGA-Link creates a top-level symbol, or optionally connects to an existing symbol, that links the FPGA design with the rest of the system. This allows the FPGA symbol to change without modifying the system level schematic. The connectivity between hierarchy levels is maintained and can be verified.

Xilinx Design Input Flow - Alliance or Foundation Series Software

Two output files, .pad and .dly, are output by the Xilinx Place and Route program. The .pad file includes all used pins and their locations. The .dly file is used to determine pin types. FPGA-Link reads all specified pins from the .pad file, the .dly file, and an optional configuration file, then looks in the Viewlogic IntelliFlow database to find other information about the Xilinx device, such as power and ground pins. If FPGA-Link cannot determine the pin type of any of the pins in the .pad file you will be presented with a list of those pins and asked to specify their types.

Xilinx Design Input Flow - XACT Series

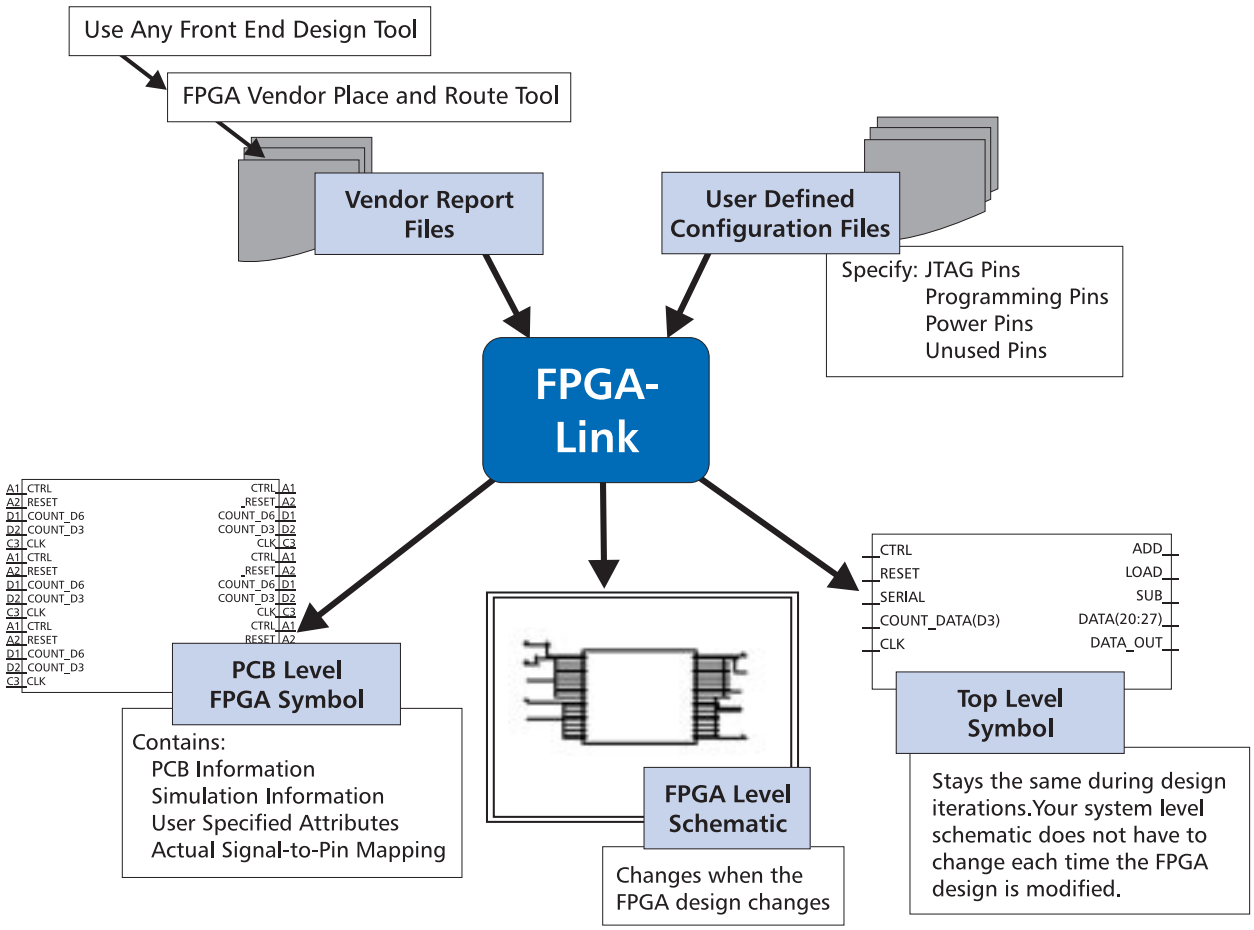
The older Xilinx XACT Place and Route tools output an .xnf file, which includes all used pins, their locations, and types. To generate this file, make sure the "Produce Timing Simulation Data" option is turned on in the Design Implementation Options dialog box. FPGA-Link reads all specified pins from the .xnf file, and from an optional user-defined configuration file, then looks in the Viewlogic IntelliFlow database to find other information about the specific Xilinx device (such as power and ground pins).

User Customizable Options

You have the ability to supply additional information to FPGA-Link for creating more intelligent design models. Some of the advanced functions include:

- Explicitly defining JTAG/programming pins, power pins, and unused or unconnected pins.
- Automatically adding capacitors for power and signal pins.
- Linking to FPGA Simulation models (Schematic, EDIF, VHDL, Verilog).
- Including specific symbol attributes such as part number, cost, description, and so on.
- Automatically creating bus pins.
- Controlling symbol object sizes such as pin length, pin spacing, label and attribute text.

FPGA-Link provides tremendous time savings to the FPGA and system design engineer. It quickly and correctly integrates FPGAs into system-level designs, and maintains design integrity during FPGA design changes.



FPGA-Link Outputs

When FPGA-Link executes, it combines the data provided from the place and route tools with the optional user configuration data, and automatically generates the following outputs:

- A symbol containing PCB attributes and simulation information.
- A schematic with the FPGA symbol on it (and optional capacitors).
- Either attributes to link to simulation models, or underlying schematics to represent the design.
- Either a new top-level symbol for use in system-level schematics, or a link to an existing top-level symbol.

For linking to simulation data, you can specify an "Underlying Model" type as EDIF, VHDL, or Verilog. If the model is set to EDIF, FPGA-Link creates underlying schematics for the design. If it is VHDL or Verilog, FPGA-Link adds the required attributes to perform simulation of the symbol.

Conclusion

FPGA-Link provides tremendous time savings to the FPGA and system design engineer. It quickly and correctly integrates FPGAs into system-level designs, and maintains design integrity during FPGA design changes. FPGA-Link operates in conjunction with Viewlogic's Workview Office design environment. FPGA-Link is priced at $1,500 and is distributed through a Value Added Reseller channel. Free 30-day evaluations can be obtained through the TRILOGIC website or by sending an email request to info@trilogic.com Σ.

For more information, contact TRILOGIC at 1-800-486-3585, info@trilogic.com or www.trilogic.com.

New Virtex Card Provides FPGA-based Real-Time Processing

A new card designed to help you quickly implement a complete Virtex-based system for applications such as DSP or image processing.

by Allan Cantle, Managing and Technical Director, Nallatech Ltd., a.cantle@nallatech.com

Nallatech Ltd. has released the first PCI card, called "Ballynuey," that uses Virtex FPGAs for developing data processing systems. We provide a family of cards with different densities, ranging from the currently available XCV300 up to the XCV800 device. Using this card, you can quickly and easily:

- Evaluate the Virtex family.
- Implement full custom designs utilizing the four expansion 'DIME' module sites.
- Get immediate and simple access to the PCI bus without hindering the functions that are implemented in the Virtex FPGA.

Operation

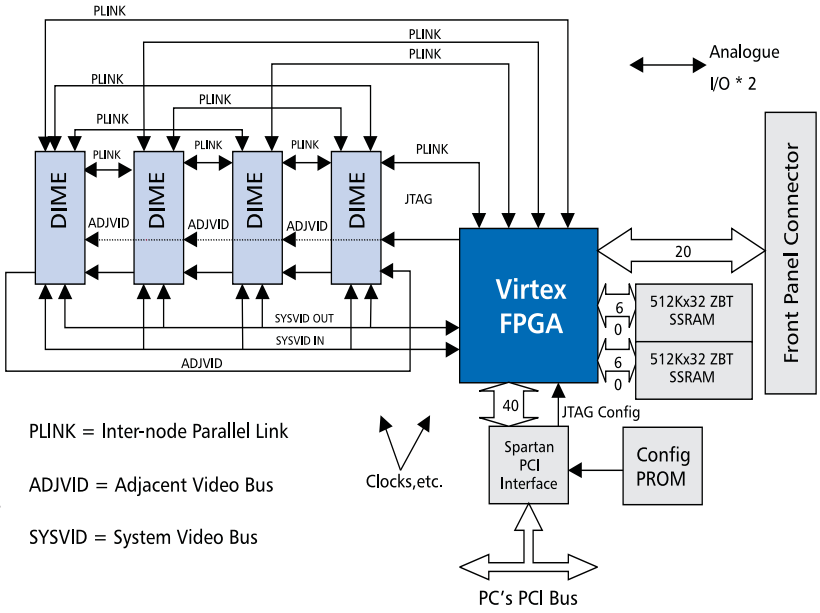
The card uses a Xilinx Spartan device, pre-programmed to interface between the software on the PC system and the user applications running on the Virtex device. The combination of the software library (running on Windows 95/98 and Windows NT) and the Spartan firmware handles the following operations:

- PCI Interfacing through the Xilinx PCI LogiCORE.
- PCI Master DMA Engine capable of burst transfers.
- Two separate FIFO interfaces between the PCI master DMA engine and the Virtex device.
- Mailbox registers.
- Interrupt handling.
- Virtex reconfiguration from software.

This enables you to quickly produce Virtex-based solutions for PCI systems. Additional facilities are also offered by the card to support complete systems, including:

- Digital I/O: 20 bidirectional.
- Analog I/O: two connectors on backplate for analog signal routing.
- Four DIME modules sites. (**D**SP and **I**mage processing **M**odules for **E**nhanced FPGAs: a new board level standard for FPGA-based processing systems, enabling system customization through the use of standard modules.)
- Three independent clock sources.
- 12 status LEDS.

A functional diagram of the card detailing the data flow is shown below. It shows the basic topology used in this DIME module implementation aimed at image processing applications.



Conclusion

The Ballynuey card gives you an ideal platform for evaluating the Virtex FPGA family, providing a range of system level facilities, including PCI interfacing and four DIME module expansion sites. Standard DIME modules are available covering video capture and display, high-speed communications, and data capture. This enables you to construct full-custom FPGA-based DSP systems using standard products while reducing costs, reducing risk, and reducing time to market.

For additional information on Ballynuey, or the DIME standard, see: www.nallatech.com.

Hierarchy Management in Synplify

A look at how Synplify automatically manages hierarchy for all Xilinx architectures while giving you additional controls if required.

by Allen Drost, Corporate Applications Manager, and Jim Tatsukawa, Partner Programs Manager, Synplicity, allen@synplicity.com, jimt@synplicity.com

As your designs reach ASIC complexities with the Xilinx Virtex devices, you will face new design problems. Of particular importance is how synthesis tools handle your design's hierarchy. Simple solutions of just turning on or off hierarchy are inefficient and cumbersome. Synthesis tools need to look beyond just optimization and be used as part of an overall design flow that includes constant changes and recompiles.

Hierarchy Pro's and Con's

Traditional synthesis tools treat hierarchy as "artificial" hard boundaries. Thus preventing logic on one level of hierarchy from being combined with logic on another level. For example, if the output of an inverter crosses a level of hierarchy and drives a second inverter, this hierarchy boundary would prevent a traditional synthesis tool from optimizing the double inversion, resulting in a final circuit that has two inverters instead of a more optimal circuit with zero inverters. See **Figures 1 and 2** for more examples.

Synthesis tools need to look beyond just optimization and be used as part of an overall design flow with constant changes and recompiles.

Alternatively you could try and dissolve the design so that there were no hierarchical boundaries. This flattening of the design may produce a smaller or faster design from a logic perspective, but it creates other complications in the design flow. First of all, blindly flattening the entire design may produce a single block of logic large enough to overwhelm the capacity of the synthesis tool, resulting in unmanageable runtimes, or a sub-optimal netlist. To deal with this problem, you may need to specify which levels of hierarchy to dissolve, but this requires you to decide what the optimal hierarchical boundaries should be.

Another issue with flattening hierarchy is that, although it may lead to a more optimal netlist from synthesis, the simple fact that the topology of the design has changed may cause problems with tools downstream. For example, you may invest significant amounts of time and effort developing constraints for placement and routing, as well as a simulation testbench. By altering the topology of the design, the grouping of large blocks of logic may change, as well as the names of the registers within the design. These changes could invalidate both the place and route constraints and the simulation testbench. If this occurs,

Continued on the following page

RTL & Technology Views of Hierarchical Design

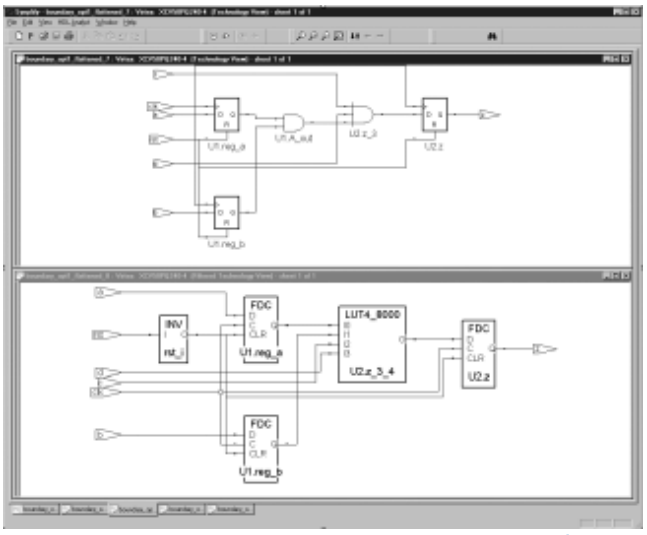


Figure 1

This design simply ANDs four signals together, however as shown in the top (RTL) view, two of the signals are ANDed in hierarchical block U1 and two more are ANDed with that result in block U2. Notice in the bottom (Technology) view that the two AND structures were merged together into a single 4-input LUT in block U2. This type of simple boolean optimization occurs regardless of the value of syn_hier.

Source code

```
/**** Sub-block A description ****/
module block_A (clk, rst, a, b, A_out);

input clk, rst, a, b;
output A_out;

reg  reg_a, reg_b;

always @(posedge clk or negedge rst) begin
 if (!rst) begin
    reg_a <= 1'b0;
    reg_b <= 1'b0;
 end
 else begin
    reg_a <= a;
    reg_b <= b;
 end
end

assign A_out = reg_a & reg_b;

endmodule
```

```
/**** Sub-block B description ****/
module block_B (clk, rst, c, d, A_out, z);

input clk, rst, c, d, A_out;
output z;

reg  z;

always @(posedge clk or negedge rst) begin
 if (!rst)
    z <= 1'b0;
 else
    z <= c & d & A_out;
end

endmodule

/**** Top level description ****/
module boundary_opt1 (clk, rst, a, b, c, d, z);

input clk, rst, a, b, c, d;
output z;

wire  A_out;

block_A U1 (clk, rst, a, b, A_out);
block_B U2 (clk, rst, c, d, A_out, z);

endmodule
```

Figure 2

you would need to update the constraints and testbench, creating a lot of extra work, particularly if it is necessary to iterate this process several times.

Hierarchy Solution

Synplify handles these hierarchical issues automatically. During the synthesis process, Synplify dissolves as much of the design's hierarchy as possible to allow efficient optimization of logic across hierarchical boundaries while maintaining fast runtimes. Synplify then rebuilds the hierarchy to be as close as possible to the original source.

With the exception of any optimizations that occurred on the logic that straddles the hierarchy boundaries, the final netlist will have the same hierarchy as the original source code, ensuring that hierarchical register names remain consistent, and that major blocks of logic remain grouped together. This method of handling hierarchical boundaries, combined with the architecture-specific mapping, creates an efficient and effective optimization engine.

Additional Control for Hierarchy

Although Synplify does a good job at managing hierarchical designs automatically, from time to time you may have a specific reason to want manual control over your design's hierarchy. Synplify offers the "syn_hier" attribute to provide this manual control. The syn_hier attribute is applied to instances, modules, or architectures. It takes one of three values:

- The "remove" option dissolves a level of hierarchy.
- The "soft" option is the default option, and gives Synplify control over which hierarchical boundaries to dissolve.
- The "firm" option prevents a level of hierarchy from being dissolved, however, simple boolean optimizations will still take place across hierarchical boundaries (see **Figure 1**).

Notice that these options control the way Synplify handles a design during optimization only. Regardless of which option is selected (remove, soft, or firm), Synplify will rebuild the

Optimization With Soft Hierarchy

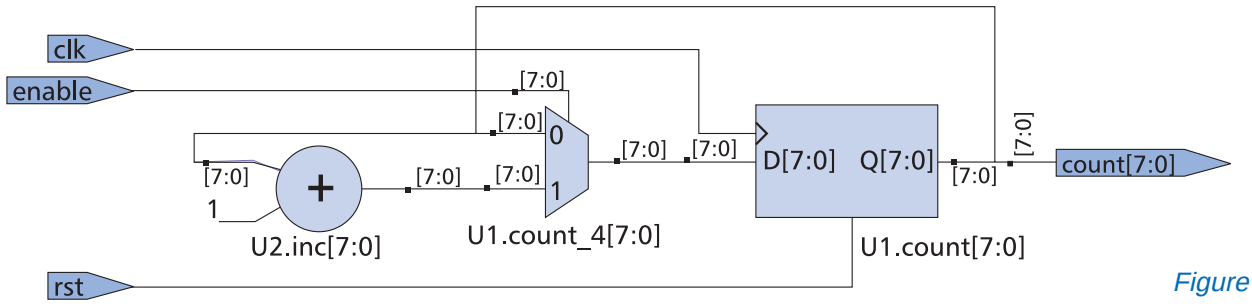


Figure 3

hierarchy before the final netlist is created, ensuring that the netlist created by Synplify is efficient with regard to hierarchical boundary optimizations, and structurally as close as possible to the source code.

The syn_hier attribute can be placed directly in the source code, in the Synplify constraints file (.sdc), or in the graphical constraint editor in SCOPE. See **Figure 4** for example usage. The syntax is shown below:

Verilog:

```
module block_A (clk, rst, enable, inc, count) /*
 synthesis syn_hier = "firm" */;
```

VHDL:

```
attribute syn_hier: string;
attribute syn_hier of block_A : architecture is
 "firm";
```

Constraint file (.sdc):

```
define_attribute { U1} syn_hier { firm}
```

Summary

To control Virtex designs over multiple compiles, synthesis tools need a hierarchical solution that does not compromise the overall performance of the design. Synplify understands the hierarchies of a design, and automatically produces a netlist with efficient logic optimization across hierarchical boundaries, while preserving a topology as close as possible to the source code, thus ensuring that the entire design flow remains intact. See www.synplicity.com for more information.

Source Code:

```
/**** Sub-block A description ****/
module block_A (clk, rst, enable, inc, count);

input    clk, rst, enable;
input [7:0] inc;
output [7:0] count;

reg  [7:0] count;

always @(posedge clk or negedge rst) begin
 if (!rst) count = 8'h00;
 else if (enable) count = inc;
end

endmodule

/**** Sub-block B description ****/
module block_B (count, inc);

input [7:0] count;
output [7:0] inc;

assign inc = count + 1;

endmodule

/**** Top level description ****/
module boundary_opt1 (clk, rst, enable, count);

input    clk, rst, enable;
output [7:0] count;

wire  [7:0] inc;

block_A U1 (clk, rst, enable, inc, count);
block_B U2 (count, inc);

endmodule
```

Figure 4

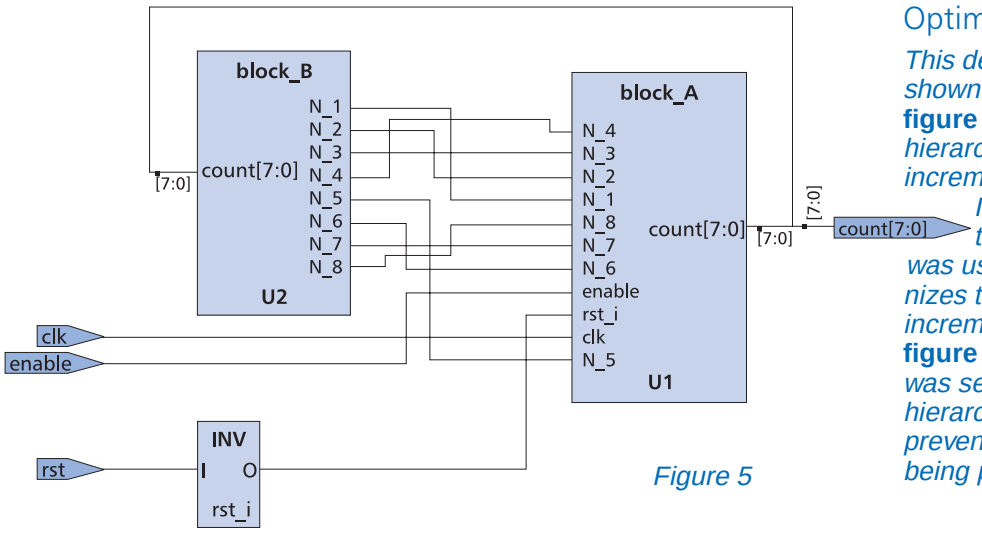


Figure 5

Optimization With Firm Hierarchy

This design is a simple counter. As shown by the component names in **figure 3** *(RTL View), the register is in hierarchical block U1 and the incrementor is in hierarchical block U2. In* **figure 6** *(Technology View), the default value of syn_hier was used (soft), and Synplify recognizes the counter and pulls the increment logic into block U1. In* **figure 5** *(Technology View), syn_hier was set to the value firm. This kept the hierarchical boundary in tact and prevented the increment logic from being pulled into block U1.*

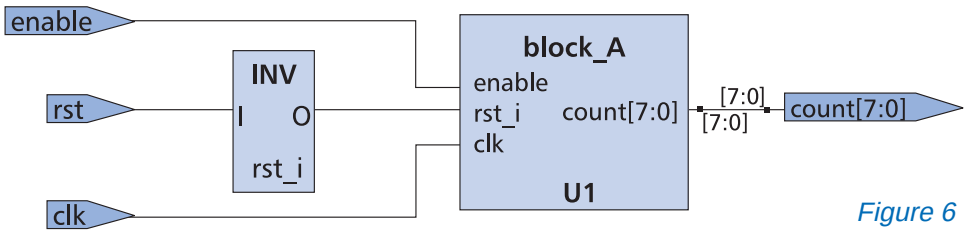


Figure 6

Anna-Liz
A New Core for Debugging PCI Designs

A new PCI core that solves a wide range of problems including transaction duration measurement, retry analysis, and post-mortem analysis.

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

"Anna-Liz" is a 33-MHz PCI bus analyzer core developed by the French company Multi Video Designs. It can be associated with any version of Xilinx PCI LogiCORE (because it fits in the "Userapp" module of the top-level schematic) and can be mixed with any related design without extra hardware. You can use it in a separate board (any type of PCI board using the Xilinx PCI Interface) or include it in an existing design.

Analysis Capabilities

Anna-Liz (**Figure 1**) captures PCI cycles and stores related data according to a programmable mask on addresses and transaction codes. During an address cycle an 8-bit counter is reset and its value is stored concurrently with the data. Up to 31 transactions can be analyzed. Addresses, Data, CBE[3:0], IRDY#, TRDY#, DEVSEL#, and STOP# are snooped on the PCI bus and stored in CLB RAMs (32x84) together with the duration counter.

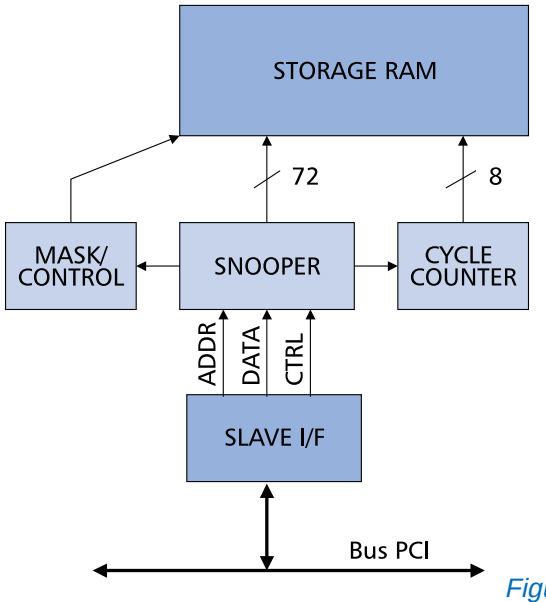


Figure 1

Storage is limited to a depth of 32 – 1 for pipelining reasons; LOCK#; SBO#, SDONE#, and REQ#/GNT# are not analyzed.

The information is stored internally only if the mask matches and when IRDY# and TRDY# (or STOP#) are active. A valid disconnect, with and without data and abort cycles, can therefore be stored.

The basic version uses an 84x32 synchronous memory arranged as follows:

- 32 bits for address.
- 32 bits for data.
- 4 bits for transaction code.
- 4 bits for byte enables.
- 3 bits for TRDY#, STOP#, and DEVSEL# values.
- 1 bit for burst signaling.
- 8 bits for cycle counter.

Analyzing Configuration Cycles (Read and Write) in the Range 1340-134F

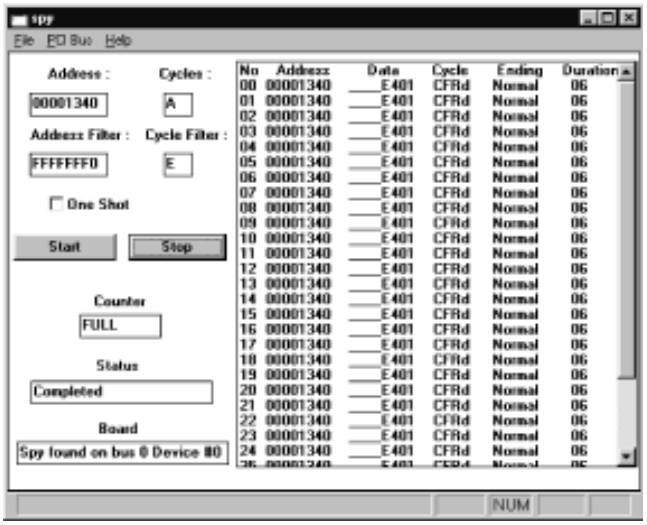


Figure 2

Anna-Liz can save you a lot of time and trouble when you are developing PCI-based products.

A programmable bit selects either single-shot or a continuous storage. In single-shot mode, storage stops when the capture memory is full. In continuous mode, storage stops when you issue a software command.

If the RESET# signal is disconnected from the device, the capture memory is not cleared when resetting the target system, so Anna-Liz can be used for post-mortem analysis; this can be very useful during testing.

Software Support

Anna-Liz is supported by DOS and Windows:

- The DOS version includes an initialization program associated with a result analyzer program. Initialization is made by a single line command which includes Hex mask programming (to filter specific transactions). Some shortcuts are available for transaction code masking. The result analyzer program displays captured cycles using a clean and understandable form (text form with text display of transaction type and end of transaction). Other programs allow displaying of PCI and mother board capabilities.

  Analysis is started by executing the setspy.exe program (parameters are defined in the command line).

- The Windows version includes the DOS programs in a single window with easy-to-use commands and displays (command buttons and control windows). Results are displayed as text in a window. Analysis is started by clicking on a button (parameters are defined in specific windows).

- Source code (C/C++) is included with the product and can be very easily adapted to any specific use. PCI BIOS accesses were written in assembler for better portability.

Implementation

Anna-Liz was developed in Viewdraw, and source code is included in the product, making specific adaptations very easy to do. For example, a scope synchronization output can be added very easily. Anna-Liz works with any Xilinx LogiCORE-based PCI interface. It only needs a slave core plus 280 CLBs, and uses a single I/O base (32 bytes). It can be mixed with an existing design if the target part holds enough room, or implemented as stand-alone on any existing board.

Spartan, Spartan XL, and XC4000E,/XLA/XLT devices can be used (Virtex version to be issued soon).

Conclusion

Anna-Liz can save you a lot of time and trouble when you are developing PCI-based products. Σ

See page 27 for more information on Multi Video Designs.

Help Window Displaying PCI Command Codes

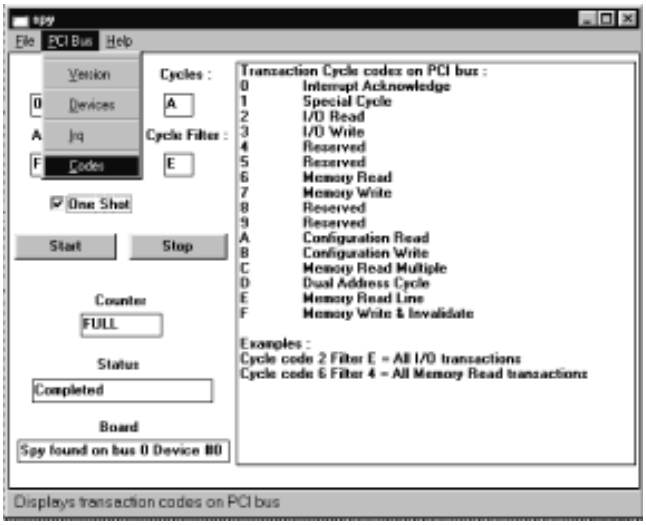


Figure 3

Displaying the Available Devices and Configuration Registers on the Target Machine

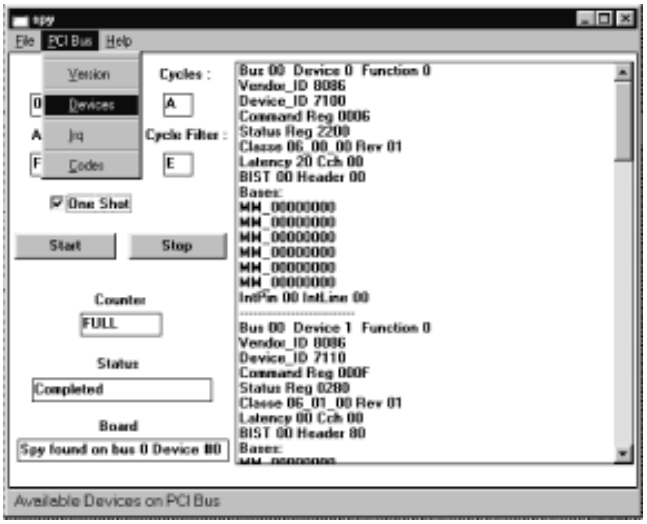


Figure 4

New CardBus and PCMCIA Cores for Xilinx FPGAs

Mobile Media Research, a leading supplier of FPGA development kits and PCMCIA tools is introducing two CardBus cores and a new multi-function core for 16-bit PCMCIA applications, each supported by a comprehensive test bench and development boards.

by Faisal Haque, Mobile Media Research, Inc,
faisal_haque@BayNetworks.com

The CardBus cores are offered as target only and target/master. They support the following features:

- Full CardBus configuration header support.
- Support for an external CIS ROM.
- A generic card-side interface that can be customized for a variety of applications.
- 33MHz operation with Xilinx XC4000XL family.
- Three memory base address registers.
- One I/O base address register.
- Interrupt support.
- Support for PCMCIA-specific event control signals.
- Full compatibility with Mobile Media's upcoming PCI cores.

The CardBus cores, implemented in Verilog, are supported by a comprehensive set of development boards such as the CardBus prototyping environment, and CardBus extender. **Figure1** shows a block diagram of a CardBus PC card based on the CardBus target/master core.

PCMCIA Multi-Function Core

The PCMCIA Multi-Function Core (PCMC) is a universal PCMCIA interface. It is Mobile Media's third PCMCIA core offering, designed to support multi-function cards as well as any application of single function cards. It provides a simple card-side interface which can be customized to your specific devices. The PCMC comes fully supported with a test bench, a PCMCIA host model, and a comprehensive set of PCMCIA development solutions. The PCMC can be implemented in any Xilinx device, including the XC3000, XC4000, or XC5000 series devices.

The main features of the PCMC are:

- 100% programmable single-chip PCMCIA interface for PC cards.
- Function configuration registers per PC card 97.
- Support for up to 32KB of attribute memory.
- Interface for an external EEPROM.
- Interface for an external SRAM.
- Configurable address space.
- Supports the following configuration options:
  - I/O Base Address configurable within a 64KB address range
  - Memory base address configurable within a 64MB space
  - Support for multi-function cards
  - Support for DMA
  - Support for speaker/audio output through the PCMCIA bus
  - Digital audio (enable/disable)
  - Ring indicate (enable/disable)
  - Card enable/disable
  - Power down (enable/disable)
  - Card reset

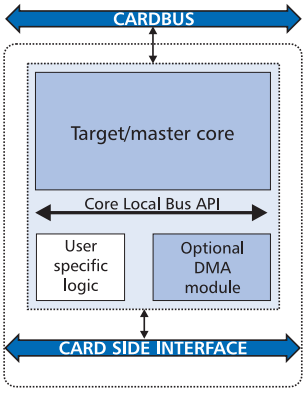


***Figure 1** - CardBus target/master core block diagram*

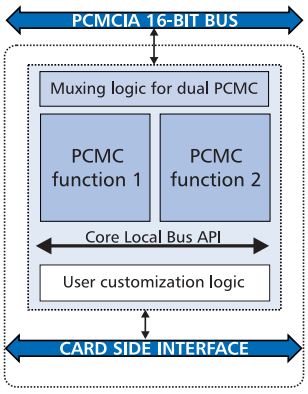


***Figure 2** – PCMC-based dual function PC card block diagram*

Figure 2 shows a block diagram of a multi-function PC card design using PCMC.

Support Products

The CardBus cores are supported with a variety of development tools, including:

- CardBus prototyping card.
- CardBus extender/logic analyzer card.
- CardBus test suite.
- CardBus host model.

The PCMCIA Cores are supported by the following development solutions:

- PCMCIA logic analyzer/extender card.
- PCMCIA host card.
- PCMCIA exerciser.
- PCMCIA type converter.
- PCMCIA prototyping card.

Availability

The PCMC is available for beta customers now. The CardBus cores will be available for beta customers in June.

For more information:

Tel : 510-657-4890
Fax: 510-657-4892
Email : sales@mobmedres.com
http://www.mobmedres.com

Concept® HDL
New Standard in Schematic Capture

Concept HDL replaces SCALD, adding many new features for FPGA design.

by Randy Hartgrove, Product Marketing Manager, Cadence Design Systems, randyh@cadence.com

Cadence Design Systems has introduced Concept HDL, a new HDL-based design creation tool to replace the older SCALD-based system. SCALD is a proprietary format that offered many advantages in its time, but now yields to the future. Concept HDL, based on the industry standards Verilog-HDL or VHDL, offers all the capabilities of its predecessor and in the PE13.5 release offers many new features that are essential for FPGA designs including support for the new Virtex family from Xilinx.

Enter Concept HDL

The HDL-centric data model upon which Concept HDL is built provides well-defined industry standards for expressing design structure, configuration, expansion, inheritance, scoping, constraints, and property annotations. The use of the standard also provides for easy third-party tool integration and a tighter, more natural union with simulators like the Verilog®-XL simulator, NC Verilog® simulator, Leapfrog® simulator, or Affirma™ NC VHDL simulator.

Concept HDL also features the new Occurrence Property File (OPF) that supports the proper reuse of hierarchical logic blocks. In this way, redundant logic does not need to be duplicated in the design, but is merely referenced as many times as needed. The OPF keeps separate property annotations for the different instances of reused logic but permits proper design expansion to either an FPGA or PCB design.

Concept HDL also offers the On-Line Electrical Connectivity Server (OLECS), a new feature that expands the design to a fully evaluated state for a variety of analyses or electrical rule checks. OLECS gives Concept HDL an understanding of the complete design, and moves it out of the realm of schematic capture into design creation. Combine the OPF and OLECS features together and you begin to understand what a step forward Concept HDL offers you.

Concept HDL and FPGA Design Creation

Concept HDL supports top-down, mixed-level, and bottom-up FPGA design flows. For the top-down and mixed-level flows Concept HDL is integrated with Synplify from Synplicity for design synthesis. Concept HDL writes Verilog and VHDL directly from a block diagram schematic that is defined in either Verilog or VHDL. The design is used by Synplify to synthesize the structural netlist that can be passed to the Xilinx place and route tools. This combination of high-level block diagram definition in Concept HDL and synthesis in Synplify make a powerful FPGA top-down design environment.

A mixed-level design consists of modules in Concept HDL defined in FPGA vendor primitives as well as modules that are described using Verilog or VHDL. Concept HDL can also treat modules as black boxes for designs where the FPGA layout or synthesis is complete and should not be redone. The top-down and mixed-level flows feature a tight integration between Concept HDL and Synplify that includes generation of Synplify project and constraint files. This provides Concept HDL with the ability to pass constraints to the synthesis engine to control the way the design is implemented in the physical device.

The End of SCALD as You Know It

With the future of design safely established with Concept HDL, the SCALD architecture will no longer be available. Cadence Design Systems, Inc. has offered all of its customers who are using the SCALD architecture a no-charge migration to the new HDL solution. As previously mentioned, in the PE 13.5 release Concept HDL also provides support for the Xilinx Virtex technology, which is not supported by SCALD. In fact, neither Xilinx nor Cadence Design Systems, Inc. will support the SCALD architecture with new technology, so now is the time to make the switch.

For more information about the powerful features of Concept HDL, or to learn more about the transition from SCALD to HDL, please visit the Cadence website at www.cadence.com. Registered users of Cadence tools may also visit SourceLink for more information on the transition.

Using the Xilinx Verilog Flow for Efficient High-Speed Design

A real-life example of a Verilog design that runs as fast as a schematic-based design; a testimonial to an excellent tool flow and to the capability of XC4000XL FPGAs.

by Rob Weinstein, Senior Member Technical Staff, and Timothy Smith, Managing Director, Memec Design Services, rob_weinstein@memecdesign.com, tim_smith@memecdesign.com

For many years we have heard that to get real speed and performance out of FPGAs you must use a schematic design flow. However, this is no longer true. The following is an example of a Xilinx design that had to work at 100MHz, and we completed it using the Xilinx Alliance Series 1.5 tools, Synplicity, and the Silos III simulator. This design flow was significantly faster than schematic entry (we estimate a 40% time savings).

Design Overview

This design is used to characterize high-speed analog to digital converters. It accumulates histogram data on 14-bit data words, at 100M words per second. Each incoming 14-bit data value is presented to the address bus of an external SRAM. The data value stored at that address is read into the FPGA, incremented by one, and written back to the same location. It basically sorts the input samples into bins corresponding to the value of the sample. For instance, each time an input sample of 0 is detected, the "0" bin is incremented, likewise, each time an input sample of 42 is detected, the "42" bin is incremented. There is a bin for each possible input value.

The input samples are directed sequentially to four separate SRAMs. This allows 40ns for each read-increment-write cycle while maintaining a 10ns sample period. When the acquisition is complete, the SRAM values can be read out of the FPGA for analysis.

The FPGA also contains logic to keep track of the maximum deviation between adjacent samples. The maximum deviation value is computed every 10ns.

Device utilization is shown in **Figure 1** and a block diagram is shown in **Figure 2**.

Device Utilization

Device	CLBs Used	Flip Flops	4 Input LUTs	3 Input LUTs	IOBs Used	Max Clock Speed
XC4013XL-09 PQ240C	303 (52%)	307	329	69	160 (83%)	100MHz

Figure 1

Block Diagram

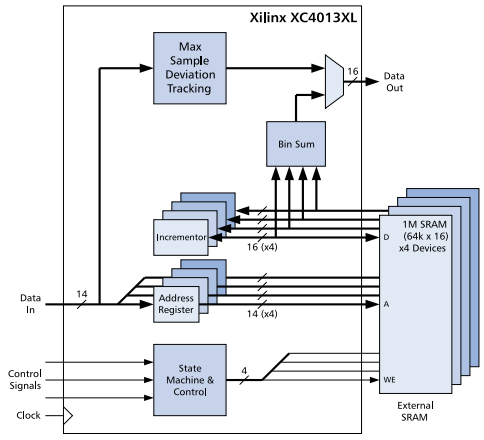


Figure 2

Design Flow

We have been using schematic entry for Xilinx designs since 1992; this year we switched to Verilog. We were somewhat apprehensive about using an HDL. Using Synplicity, however, with its HDL Analyst, let us see the synthesized implementation, in schematic form, without performing a place and route. This let us hone our coding style to achieve one level of logic between registers.

In addition, Synplicity does an excellent job of using the Xilinx carry logic. It builds optimized arithmetic functions to whatever word width I am using. Using schematics, it is a tedious process to modify "standard" width arithmetic functions to the width I need.

Functional simulation is also much faster with an HDL simulator than with the gate-level simulator used with schematic designs. Also, using Verilog as a simulation language is easier for developing complete test benches than the command language of schematic based gate-level simulators. For example, it was trivial to model the external SRAMs along with the Xilinx device. You basically have a complete programming language at your disposal.

Conclusion

This example demonstrates the power of Xilinx FPGAs for implementing high-speed designs as well as the strength of using a High-level Design Language such as Verilog. Using an HDL, you can achieve design performance that equals schematic designs.

For more information on Memec Design Services, contact: Maria Aguilar, Design Coordinator, Memec Design Services, (888) 360-9044, (602) 491-4311, maria_aguilar@memecdesign.com, http://www.memecdesign.com, or contact your local Insight Electronics office: (800) 677-7716, http://www.insight-electronics.com

A PCI Acquisition Board Using the XC4013

A unique PCI bus data acquisition design with dynamic reconfiguration management.

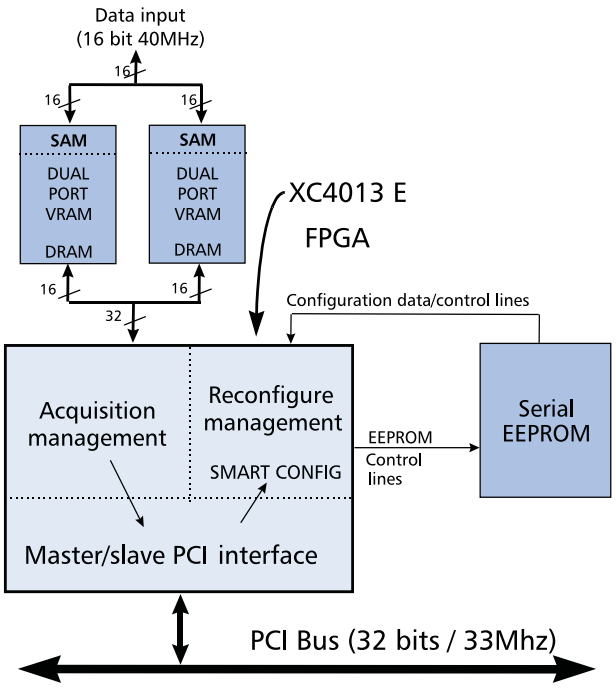
by Edgard Garcia, Xilinx consultant, Multi Video Designs, edgard.garcia@mvd-fpga.com

At Multi Video Designs, we have developed a high-speed data acquisition board for one of our customers, based on the XC4013E FPGA and a Xilinx PCI LogiCORE. The FPGA includes a Master/Slave PCI interface as well as data acquisition management and a Video-RAM controller. The board can acquire data bursts of between one pixel and one megapixel at frequencies up to 40MHz. Data is then transfered over the PCI bus in master burst mode. The corresponding control software operates under Windows and DOS.

The FPGA can be reconfigured using "Smart config," an original feature that allows you to rewrite an integrated EEPROM which reconfigures the FPGA on reset. This makes the board much more flexible than is usual for a PCI board, and it can be rapidly adjusted to your changing needs without even opening your computer.

We chose a Xilinx PCI LogiCORE because it allows us to implement both the master/slave PCI interface and application specific logic. It also makes for a very short development time.

Our customer wanted five working boards designed and built in less than 6 weeks. Thanks to the Xilinx PCI LogiCORE and our own hardware and software PCI analyzer "Anna-Liz" *(see page 22)*, our customer was completely satisfied.



Conclusion

With Xilinx FPGAs and PCI LogiCOREs you can quickly get your PCI product to market.

For further information contact: Edgard Garcia, Tel: (33) 5 62 13 52 32, Fax: (33) 5 61 06 72 60, E-mail: edgard.garcia@mvd-fpga.com, or info@mvd-fpga.com

About Multi Video Designs

Multi Video Designs is a consulting company specializing in the development of Xilinx FPGA/CPLD for video, telecom, PCI and other applications where speed and/or density is an important factor. Our services include:

- On-site training courses for Xilinx FPGA (Xilinx approved training center)
- On-site training courses for VHDL logic synthesis and simulation
- On-site PCI bus training courses
- Approved design center (to customer requirements)
- Electronic board design to customer specs
- Electronic board fabrication
- On-site technical assistance

CPLD Fitter Shootout:

Xilinx 1.5 *versus* Altera 9.01

In a recent benchmark study we compared the "push button" results for Xilinx implementation technology v1.5 versus Altera MAX+PLUS II v9.01. Take a look for yourself; the results are quite compelling.

by Dave Grace, CPLD Software Marketing Manager, Xilinx, dave.grace@xilinx.com

Xilinx recently completed an in-depth study, comparing our latest v1.5 implementation tools with Altera's latest release (v9.01). The results prove that the Xilinx solution, along with the XC9500 and XC9500XL device families, is the obvious winner over Altera's MAX7000S/A/AE CPLD offering.

Synopsys FPGA Express v2.1.3 was used for design synthesis. Only defaults were used during the implementation phase; no additional "timing driven" options or constraints were applied. Two separate implementation runs were created per design with the Xilinx tools: one when optimizing for area, the other when optimizing for speed. This required us to select the "speed" or "area" template in the Xilinx tools. Altera does not provide implementation templates in their tools. The following implementation results were recorded for each design run:

- Device targeted (part, package, and speed grade are selected automatically by both the Xilinx and Altera tools for a specified device family).
- Number of macrocells used.
- Number of product terms used.
- Main system clock frequency (Tcyc), in megahertz (MHz).
- Propagation delay (Tpd), in nanoseconds (ns).

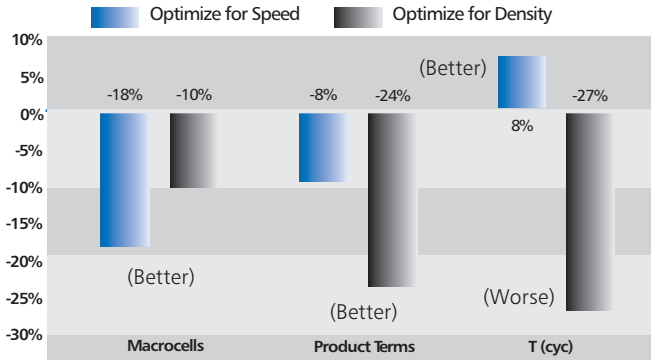
The entire process was performed twice: once to compare the results of targeting the 5V CPLD device families (XC9500 vs. MAX7000S), and again to compare the results of the 3.3V CPLD device families (XC9500XL vs. MAX7000A/AE).

Design Suite - The design suite consists of 50 HDL Designs, 25 of which are Verilog and the other 25 are VHDL. Six of the designs are control designs to test for explicit results while the remaining 44 designs are from customers. The size of the designs range from 10 to 250 macrocells in density, with the design size distribution evenly split above and below 70 macrocells.

Test Platform - Pentium 166MHz, 80MB RAM, Windows NT 4.0, Xilinx v1.5 software, Altera MAX+PLUS II V9.01 software.

Benchmark Results – 5V CPLDs

Xilinx XC9500 vs. Altera MAX 7000S Overall Results

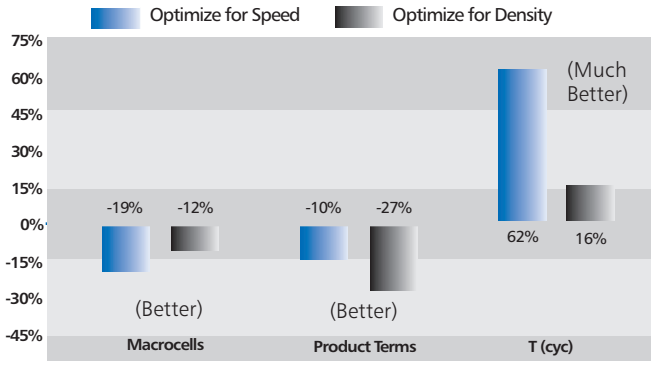


NOTES: Altera MAX+Plus II v9.01 vs. Xilinx v1.5
Percentage based on normalized Altera results.

Figure 1

Benchmark Results – 3.3V CPLDs

Xilinx XC9500XL vs. Altera MAX 7000A/AE Overall Results



NOTES: Altera MAX+Plus II v9.01 vs. Xilinx v1.5
Percentage based on normalized Altera results.

Figure 2

Results - As shown in **Figure 1**, XC9500 CPLDs use 18% fewer macrocells, 9% fewer product terms, and achieve an 8% faster clock frequency than Altera MAX7000S devices. As shown in **Figure 2**, XC9500XL CPLDs use 19% fewer macrocells and 10% fewer product terms, and achieve an impressive 62% faster system performance than Altera's MAX7000A/AE products.

Xilinx CPLD Implementation Technology for v1.5

The dramatic difference in performance demonstrated by these benchmark studies is the result of a number of important enhancements to the Xilinx Alliance Series™ and Foundation Series™ 1.5 software that improve CPLD design entry and implementation capabilities.

Design Entry Enhancements include:

- **Automatic Device Selection:** V1.5 now allows automatic CPLD device selection via either the Alliance Series Design Manager or the Foundation Series Project Manager. Both options will choose an implementation based on device, package, speed grade, or any combination. By default, the CPLD implementation tools will choose the smallest die and the smallest package with the fastest speed grade available.
- **Implementation Templates**: Both the Alliance and Foundation Series Implementation tools will allow you to use pre-defined Implementation Templates. These templates, "Optimize for Speed" and "Optimize for Density," give you an easy way of targeting your design to best suit your application.

When you first invoke the implementation tools, the default option is set to "Optimize for Speed." This template will attempt to fit a design with a bias towards the fastest system clock speed. As always, if a specific clock frequency is desired, it is best to attach a specific "timespec" to the design. This can be accomplished several ways: using the new Constraints Editor, a user constraints file (UCF), schematic symbol or attribute, or through various entry mechanisms within third-party Synthesis tools, such as Synopsys Express, Synplicity Synplify, and Exemplar Leonardo.

Fitter Enhancements Since v1.4 Software

The fitter is the backend software that places and routes your design, and then converts your design to the information and control signals that are necessary for downloading a CPLD programming file. Enhancements include:

- **35% Runtime Reduction** - The v1.5 CPLD implementation tools are now averaging 35% faster speeds than the previous v1.4 tools. This is best represented in designs that are over 108 macrocells. Smaller designs below 72 macrocells improved anywhere from 10% to 50%, depending on logic complexity.
- **15% Average Improvement in Silicon Performance through Software** - System clock speeds have improved 10% to 20% for designs placed in the same device speed grade.
- **Algorithmic Enhancements** - Improvements were made to the multi-level logic optimization algorithm, which improves logic timing and density. Xilinx also introduced a product term collapsing look-ahead feature that significantly reduces the number of product terms required for a given logic function, thus achieving higher system clock frequencies, lower logic level delays, and faster runtimes.

Summary

The results are clear. The combination of the Xilinx version 1.5 implementation technology along with the XC9000 series FPGAs produce superior results over Altera. The Xilinx software consistently uses fewer device resources, chooses a smaller device, and produces devices with shorter delays.

XC9500XL CPLDs

Immune to Power Sequencing Problems

With XC9500XL CPLDs you don't need to worry about the possibly damaging effects of improper power supply sequencing. Here's how it works.

by Jesse Jenkins, CPLD Applications Manager, Xilinx, jesse@xilinx.com

Xilinx XC9500XL CPLDs are designed to operate in either mixed 5V/3.3V systems, 3.3V only systems, or 3.3V/2.5V systems. To handle all conditions, care has been taken to assure that you need not introduce elaborate circuitry to guarantee that any power supplies rise or fall in any particular sequence.

XC9500XL CPLDs are provided with two separate power supply pins. V_{CCINT} supplies power for the internal logic, memory, and charge pumps. V_{CCIO} supplies power for the output drivers that allow the CPLD to be used in either 5V, 3.3V, or 2.5V logic level systems. The Xilinx proprietary ESD circuitry prevents high-voltage damage as well as mixed power system intermingling, as shown in **Figure 1**.

Simplified XC9500XL I/O Cell Structure

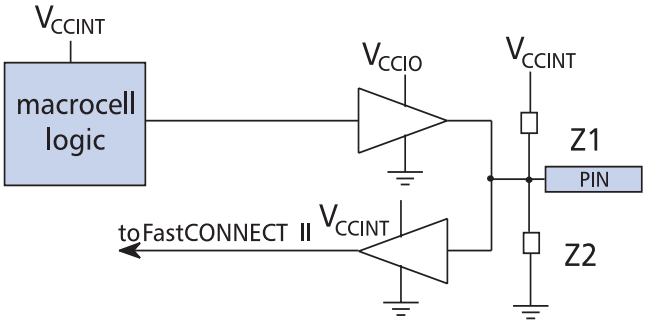


Figure 1

How it Works

Two common power supply configurations occur. First, the single supply system involves attaching both V_{CCINT} and V_{CCIO} to the same voltage, 3.3V +/- 0.3V. The second configuration is where V_{CCINT} is attached to 3.3V and V_{CCIO} is attached to 2.5V. Connecting both supplies to 2.5V is not permitted, neither is connecting V_{CCINT} to 3.3V and V_{CCIO} to 5V (so, we will not consider these two cases). Under these restrictions, it is unlikely that V_{CCIO} and V_{CCINT} will have separate 3.3V supplies, but the following analysis should also cover this case. Of specific interest is the case of having V_{CCIO} = 2.5V and V_{CCINT} = 3.3V.

Specific concern arises if one of the power supplies is off while the other is on. **Figure 2** shows the situation where V_{CCINT} is turned on and V_{CCIO} is turned off. **Figure 3** shows the case where V_{CCINT} is turned off and V_{CCIO} is turned on.

XC9500XL Output Driver with V_{CCIO} Turned Off (Input Receiver not shown)

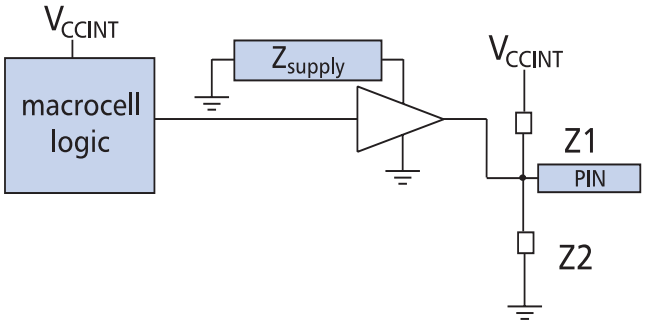


Figure 2

XC9500XL Output Driver with V_{CCIO} Turned On (Input Receiver not shown)

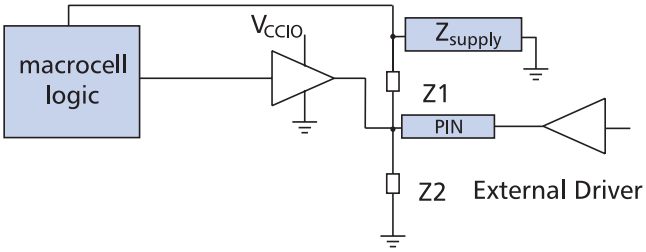


Figure 3

XC9500XL CPLDs are designed to operate in single 3.3V or mixed 3.3V/2.5V/5V systems and tolerate any power supply sequencing applied to them without damaging the CPLD, the supply, or external circuits.

Analyzing the Problem

In analyzing the potential problems, several factors must be considered:

- The supply impedance (Z_{SUPPLY}).
- The current limit of external drivers attached to the pin.
- The state of the macrocell logic driving into the output driver.

We will discuss each factor in order, to see how it affects the condition of the CPLD when power sequencing occurs.

Supply Impedance

For this discussion, we are assuming the power supplies remain attached to the chip pins electrically, whether they are powered up or not. Adding a series switch will physically break the connection, changing the assumptions.

The supply impedance is important because current may flow from the CPLD into the turned off power supply, if a path and an available source (such as the other supply) exists. This can also occur if an external CMOS chip is driving into the CPLD pin (see **Figures 2** and **3**).

Most power supplies have some silicon impedance (the impedance of the attached regulator) and a fairly large capacitor to ground. The supply output capacitor can accept substantial current, but as current arrives, the accumulated charge increases the capacitor voltage. If this turned-off supply is initially uncharged, there may be a substantial initial current. The initial current self-limits as the capacitor voltage approaches the driving voltage (from the pin). This explains how current may initially flow backward into the turned-off power supply. However, as the voltage rises this current becomes negligible.

External Driver Current Limit

Most external CMOS drivers are relatively low current drive devices. Unless a substantially large number of them can deliver a huge current into an XC9500XL pin, the risk of back drive into the pin is very small. The Xilinx proprietary ESD circuitry will limit this to a very low value, so it can usually be ignored.

Macrocell State

Figure 4 shows a simplified model of the output driving structure for an XC9500XL CPLD. If point A is LOW and V_{CCIO} is ON, the pin drives HIGH. If point A is HIGH and V_{CCIO} is ON, the pin drives LOW. If you leave Vccio powered up and V_{CCINT} powered down, static CMOS levels can be delivered to your system. In general, this is harmless.

Close-up of Simplified Output Stage
(ESD circuitry omitted)

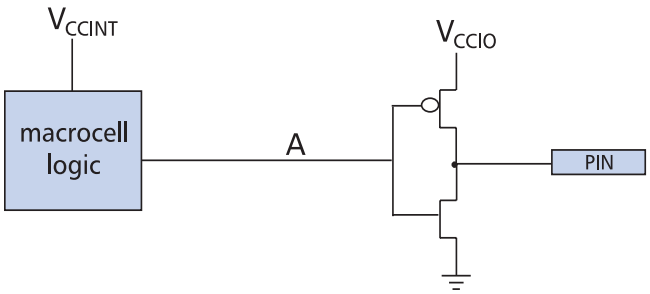


Figure 4

Conclusion

XC9500XL CPLDs are designed to operate in single 3.3V or mixed 3.3V/2.5V/5V systems and tolerate any power supply sequencing applied to them without damaging the CPLD, the supply, or external circuits. Σ

Frequency Synthesis

FPGAs can simplify frequency synthesis without adding complexity, quite often improving performance. Here are some useful techniques.

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

requency synthesis is often used in digital designs, and three major methods are used: dividers, direct digital frequency synthesis (DDFS), and fractional-N synthesis. Sometimes these techniques are combined with traditional analog phase locked loops or other analog elements to remove synthesis by-products such as unwanted side-bands or jitter.

Dividers

Taking an input clock frequency F, and dividing by any modulo of 2, such as 2, 4, 8, and so on, is a common technique. The one problem is that you must choose F in advance such that all of the other frequencies are sub multiples using the modulo 2 factors. There are circuits that rely on asynchronous delays to provide divide by 3, 5, and so on, but their outputs are not square, so they create a large harmonic content. These techniques are found most commonly in digital circuit design today. I will not elaborate further on these, because the following two techniques are more useful.

Direct Digital Frequency Synthesis

One technique that is not usually considered, due to its complexity, is the direct digital frequency synthesizer (DDFS). Here, a constant N is placed on one port of an adder, and the other port of the adder is fed back from a D-type latch whose input is connected to the output of the adder. At every clock of the latch, an incremental phase is added to the previous result. The most significant bit of the latch will transition at a frequency determined by the equation:

$$Fout = \frac{Fclock \bullet N}{2^K}$$

Where K is the number of bits of the adder and latch.

One of the interesting things that is noticed right away is that any arbitrary frequency may be generated to within a resolution of 2^{-K}. For example, with a 48 bit accumulator and latch, the resolution is $3.5*10^{-15}$, or about a thousand times better that the accuracy or resolution of a cesium clock.

So, where do you get a 48-bit latch and accumulator? In an FPGA there is usually plenty of room for not just one, but perhaps three or four DDFS functions. Not all of them have to be 48 bits, they just need to be long enough to synthesize the frequency that is desired. In fact, the Xilinx Foundation Series and Alliance Series tools provide the LogiBLOX feature which will create an adder/accumulator, of any length, with only a few keystrokes.

There are two limitations of this technique: the output frequency must be less than 1/2 the clock frequency, and the output frequency will jitter by the period of the clock frequency. To remove the jitter, you could send the output through a band pass filter, or lock a separate phase locked loop, or complete the synthesis by taking the 12 or 14 most significant bits of the latch, and using them to address a sine wave lookup table ROM. The output of the ROM then goes to a digital to analog (D/A) converter, whose analog output is then a sine wave with much less jitter. The clock jitter will still be present however, and may still require more filtering.

Fractional-N Synthesis

In fractional-N synthesis, counters are used to provide a variable modulus, or counters are used to provide for "pulse swallowing." If you have a counter that can divide by 10 and divide by 11, then you can make a circuit where the counter divides by one factor for some number of clock cycles and then divides by the other number for perhaps a different number of clocks. The resultant frequency is then the ratio of how many clocks were divided by 10 to how many were

Techniques

The advantages of using FPGAs for frequency synthesis are many: no dependence on voltage, temperature, or aging; and no external analog components.

divided by 11. This technique requires a variable modulo counter, another counter for the selection, and logic for state control.

One disadvantage of this technique is that the jitter is increased due to the difference in the modulo, with the output frequency being 10 times less, then abruptly 11 times less, and so on, in a fine time scale. The advantage is that it is considered a simpler structure (but only from the old point of view using discrete, small scale integrated circuits).

The other commonly used method of fractional-N synthesis is to drop an output clock pulse based on some rule derived from a counter state machine. For example, if you drop every 4th clock out of every 5 input clocks, the output rate is 4/5 the input rate. This is known as "pulse swallowing". This also results in a large amount of jitter due to the missing clock period.

Jitter Reduction

In all of the above techniques where the output is a digital signal transitioning at the desired rate, one can reduce the jitter by dividing the signal down. The jitter time will be less as a function of the resulting period of the signal. For example, if you want a 1.536MHz signal, and you have a 16.384MHz clock, you could synthesize a 6.144MHz signal to an arbitrary resolution, and then divide it by four. The resulting signal would have a jitter of 2.3%. This is an acceptable jitter for any data communications related application, such as T1 Frame Relay Customer Service Unit/Data Service Unit (CSU/DSU).

As mentioned before, using a phase locked loop also reduces the unwanted jitter, where the filtering of the loop is used to attenuate the unwanted side band energy which is the cause of the jitter. One of the simplest means of implementing this, is to use a voltage controlled crystal oscillator whose output is exclusive OR'd with the output of the synthesizer and whose input is the RC filtered XOR output.

Using an FPGA

Even the seemingly complex DDFS is simpler to implement in an FPGA than fractional synthesis because the DDFS circuit consists of replicated units, all interconnected simply. You don't need to find a fractional solution, and realize the state machines and modulo counters. In fact, the DDFS is the most versatile as it may be changed at any time by placing a new N at the adder. Calculating the N may get a bit difficult past 32 bits, because computer math programs are often unable to deal with the resolution and display of hexadecimal or binary numbers beyond this point. I use MathCad ™ from MathSoft, Inc. and split the calculation up into two parts to get arbitrary resolution.

The cost of implementation is the adder and the D-type flip-flop for every bit of synthesis. If examined in this fashion, the cost of a DDFS is more than the other methods. In large FPGAs, the philosophy might be best stated as "gates are free" as long as the design fits into the intended part. It takes 26 CLBs to make a 48 bit DDFS in the XC4000 family. Even for the smallest part, this is about a quarter of the total CLB count (XC4003). For larger parts, such as the XC4085XL, it is hardly a consideration.

Conclusion

The advantages of using FPGAs for frequency synthesis are many: no dependence on voltage, temperature, or aging; and no external analog components.

JTAG Boundary-Scan for Low Cost System Testing

Xilinx FPGAs and CPLDs have built-in boundary-scan capability for in-system testing and debugging. This method of incorporating special test circuitry into a device gives you complete control of, and access to, the device pins without the need for external probes. Here's what it's all about.

by Carlis Collins, Managing Editor of Corporate Communications, Xilinx, editor@xilinx.com

Multi-layer circuit boards, using surface mount devices, are extremely difficult to test; that's why the IEEE1149.1 standard, or "boundary-scan" was developed. Boundary-scan is typically used to test the interconnections between devices on a printed circuit board. Each boundary-scan-compatible device allows you to take control of its I/O pins to drive and receive test signals.

Through a simple 4- or 5-wire Test Access Port (TAP), you can shift in a stimulus for all device output pins and then read in the corresponding signals on the receiving devices' input pins. Then you compare the outputs with the inputs to verify the connectivity between devices. This can also help you repair boards by identifying the location of any open or shorted traces. By daisy chaining multiple boundary-scan-compatible devices, you can effectively test a complete circuit board using a single set of test vectors that are input through a single TAP.

Each boundary-scan-compliant device includes a shift register composed of boundary-scan cells, a 4- or 5-wire Test Access Port (TAP), and a state machine (TAP controller), as shown in **Figure 1**. The TAP pins are:

- TCK - This is a clock signal that synchronizes the internal boundary-scan state machine (TAP Controller) operations.
- TMS - This is the internal state machine mode select signal. This signal is sampled at the rising edge of TCK to determine the next state machine state.
- TDI - This is the data input pin. When the internal state machine is in the correct state, this signal is sampled at the rising edge of TCK and shifted into the device's test or programming logic.

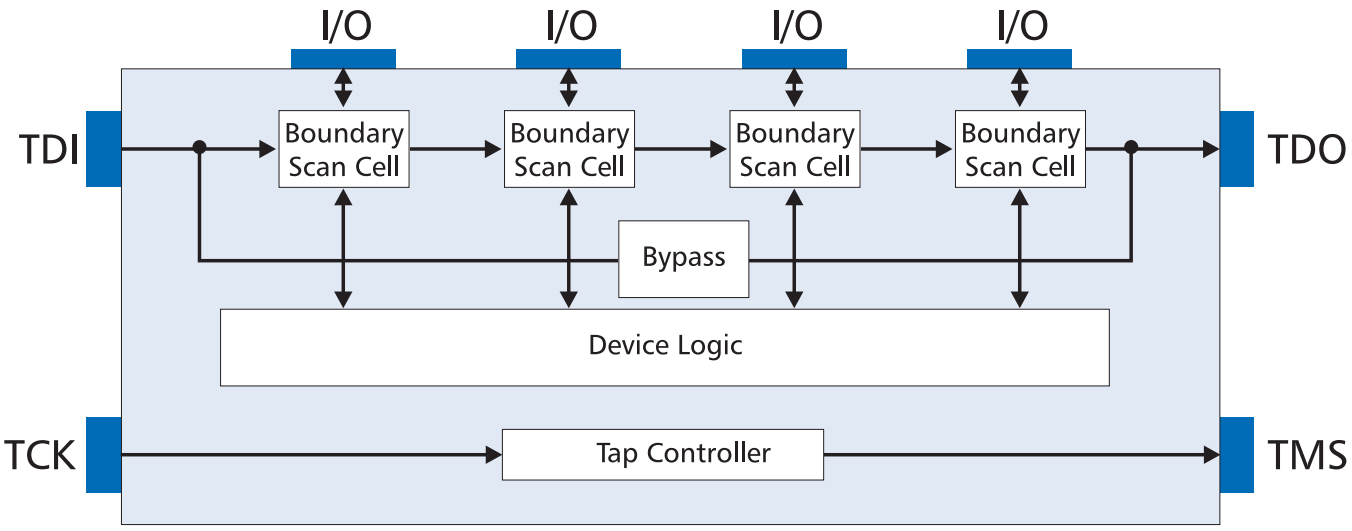


Figure 1

- TDO - This is the data output pin. When the internal state machine is in the correct state, this signal represents the data shifted out of the device's test or programming logic. The output data is valid on the falling edge of TCK.
- TRST (optional) - This is the asynchronous reset pin. When driven low, the internal state machine advances immediately to the reset state. Because the pin is optional and pins are generally high-cost additions to devices, it is usually not implemented. In addition, the internal state machine has a well defined synchronous reset mechanism.

Multiple boundary-scan devices are linked serially in a daisy chain, with all devices sharing the same TCK and TMS signal. The TDO of one device links to the TDI of the next. This results in what appears to be a single shift register of a fixed length from the system TDI pin to the system TDO.

All devices have registers composed of boundary-scan cells, internally linked between the TDI and TDO pins. Under normal operating conditions, these cells are transparent and inactive, allowing signals to pass normally. When the device is placed in the test mode, each boundary-scan cell can either read the state of its associated pin, or drive a signal onto the pin. Thus you can shift in a stimulus vector and shift out the received pattern, in one operation, for all devices in the boundary-scan chain.

Xilinx XC9500 CPLDs also use the TAP pins for device programming. This allows you to both program and test our CPLDs using the same equipment. Plus, this guarantees a reliable programming operation with no danger of lock-up (as seen in some competing devices). XC9500 CPLDs also implement optional boundary-scan instructions that provide additional capabilities (see the XC9500 or XC9500XL data sheets for details).

Conclusion

Boundary-scan is a fast, convenient, and simple method for testing and debugging systems, supported by a wide number of device and test equipment manufacturers.

All Xilinx XC9500 and XC9500XL CPLDs, and all Xilinx FPGAs since the original XC4000 series, include boundary-scan capability.

For More Boundary-Scan Information

For the full IEEE1149.1 standard,
***see**: http://standards.ieee.org/*

A wealth of information on the standard is also available from the Texas Instruments Boundary Scan Page at: http://www.ti.com/sc/docs/jtag/jtaghome.htm.

Serial Vector Format (SVF)

Serial Vector Format (SVF) is the de facto standard for interchange of boundary-scan-based stimulus information. Note that this is not an open standard. It is currently copyrighted and controlled by Asset Intertech but freely distributed.

***See**: http://www.asset-intertech.com/releases/svf.htm*

JEDEC

JEDEC Programming File - more formally known as JESD3-C Standard Data Transfer Format Between Data Preparation System and Programmable Logic Device Programmer.

***See**: http://www.eia.org/eng/allstd/std/files/jedec/jesd3%2Dc.htm*

JEDEC Chain Description File - more formally known as JESD32 Standard for Chain Description File. This file format is meant to describe the connection of arbitrary programmable devices in a serial chain. It is somewhat confused in execution, precariously trying to balance a description of both non-IEEE1149.1 and IEEE1149.1 types of serial chains in the same language. Notably, it is also unable to describe complex boundary-scan chain configurations like hierarchical or multidrop architectures.

***See**: http://www.eia.org/eng/allstd/std/files/jesd32.htm*

BSDL

Boundary-Scan Description Language (BSDL) Standard 1149.lb is used to describe the IEEE1149.1 TAP Controller and boundary-scan register on a boundary-scan-compliant device. BSDL is also implemented as a subset of the VHDL standard.

***See**: http://standards.ieee.org/*

Creating the Most Efficient Comparators

by Roberta Fulton, Technical Marketing Engineer, Xilinx, roberta.fulton@xilinx.com

Comparators are best modeled with word-wise compares within a **PROCESS** or an **ALWAYS** block that contains the **IF** statement and an **ELSE** clause, and no **ELSE-IF** clauses. Conditional signal assignments in VHDL or conditional continuous assignments in Verilog could be used, but at a high cost in simulation time. Without the sensitivity list in VHDL or the event list in Verilog the simulators would constantly be checking the statements even when the inputs are unchanging, thus slowing simulation time considerably. If compared bit-wise some synthesizers may not see optimizations available to them such as the use of H-MAPs.

Three alternative representations to infer an 8-bit equality comparator are shown below. The first, COMPARATOR_A does a bit by bit compare (**Figure 1**), the second COMPARATOR_B sets the default first, then compares (**Figure 2**), so does not have an **ELSE** clause, the third has a complete **IF-THEN-ELSE** statement (**Figure 3**).

Comparator A - Bit by Bit Compare

```
library IEEE;
use IEEE.STD_LOGIC_1164.all,
 IEEE.NUMERIC_STD.all;
entity COMPARATOR_A is
 port (AIN1, AIN2: in unsigned(7 downto 0);
     AEQ: out std_logic);
end entity COMPARATOR_A;
architecture RTL of COMPARATOR_A is
begin
EQUALITY:process (AIN1, AIN2)
begin
— Compare each bit in turn in a “for” loopp
AEQ <= ‘1’;
for I in 0 to 7 loop
 if(AIN1(I) /= AIN2(I)) then
  AEQ <= ‘0’;
  exit;
 else
  null;
 end if;
end loop;
end process EQUALITY;
end architecture RTL;

module COMPARATOR_A
(AIN1, AIN2, AEQ);
input [7:0] AIN1, AIN2;
output AEQ;

integer I;
reg AEQ;

//Compare each bit in turn in a “for” loop
always@(AIN1 or AIN2)
begin: EQUALITY
AEQ = 1;
for (I=0; I<7; I=I+1)
if (AIN1[I] != AIN2[I])
AEQ=0;
else
;
end
endmodule
```

Figure 1

Comparator B – No Else Clause

```
library IEEE;
use IEEE.STD_LOGIC_1164.all,
 IEEE.NUMERIC_STD.all;
entity COMPARATOR_B is
 port (BIN1, BIN2: in unsigned(7 downto 0);
     BEQ: out std_logic);
end entity COMPARATOR_B;
architecture RTL of COMPARATOR_B is
begin
EQUALITY:process (BIN1, BIN2)
begin
— No “else” is required since default is
 defined before the “if””
BEQ <=’0’;
if (BIN1 = BIN2) then
BEQ <= ‘1’;
end if;
end process EQUALITY;
end architecture RTL;

module COMPARATOR_B
(BIN1, BIN2, BEQ);
input [7:0] BIN1, BIN2;
output BEQ;

integer I;
reg BEQ;

//No “else” is required since default is
 defined before the “if”
always@(BIN1 or BIN2)
begin: EQUALITY
BEQ = 0;
if (BIN1 == BIN2)
BEQ=1;
end
endmodule
```

Figure 2

Complete IF-Then-Else Clause

```
library IEEE;
use IEEE.STD_LOGIC_1164.all,
 IEEE.NUMERIC_STD.all;
entity COMPARATOR_C is
 port (CIN1, CIN2: in unsigned(7 downto 0);
    CEQ: out std_logic);
end entity COMPARATOR_C;
architecture RTL of COMPARATOR_C is
begin
EQUALITY:process (CIN1, CIN2)
begin
— Easiest to read versionn
if (CIN1 = CIN2) then
CEQ <= '1';
else
CEQ <= '0';
end if;
end process EQUALITY;
end architecture RTL;
```

```
module COMPARATOR_C
(CIN1, CIN2, CEQ);
input [7:0] CIN1, CIN2;
output CEQ;

integer I;
reg CEQ;

// Easiest to read version
always@(CIN1 or CIN2)
begin: EQUALITY
if (CIN1 == CIN2)
CEQ=1;
else
CEQ=0;
end
endmodule
```

Figure 3

Reviewing RTL-level schematics or design browsers in the synthesizers we can see how the synthesizer sees the code immediately upon parsing it before any optimization or technology mapping.

Comparing the RTL level schematics of COMPARATOR_A (**Figure 4**) and COMPARATOR_B (**Figure 5**) we see that in both cases the EQUALITY operator is assigned to a generated module, a SELECT type in COMPARATOR_A and EQUALITY type in COMPARATOR_B. But we see that extra logic is inferred for comparator A to do the bit-by-bit comparison looping. COMPARATOR_C's RTL schematic is similar to COMPARATOR_B's so is not shown.

As the RTL schematics show, the initial logic for COMPARATOR_A and COMPARATOR_B is very different. Since the overall functions are equivalent we would normally assume that the optimization step would reduce them to the same logic.

But as seen in the gate-level schematics of **Figures 6** and **7**, COMPARATOR_B's word-wise compare and its more condensed optimization seed netlist allowed the synthesizer to "see" the opportunity to use two HMAPs instead of a fifth FMAP. This meant that 2 CLBs are required instead of 3 and that one level of logic is required instead of two. (An FMAP-HMAP combination is considered 1-level since no external CLB routing is required). Performance will be slightly enhanced because CLB internal routing is usually lower than between CLBs. Once again COMPARATOR_C's gate-level schematic is similar to COMPARATOR_B and is not shown. It's best to use COMPARATOR_C's code because of its readability.

When using inequality operators like ">" or "<" look for the better synthesizers to use carry logic (CYx cells) when mapping the generated modules to Xilinx technolgies. This is synthesizer dependent, if you do not see carry logic in your

COMPARATOR_A.RTL schematic

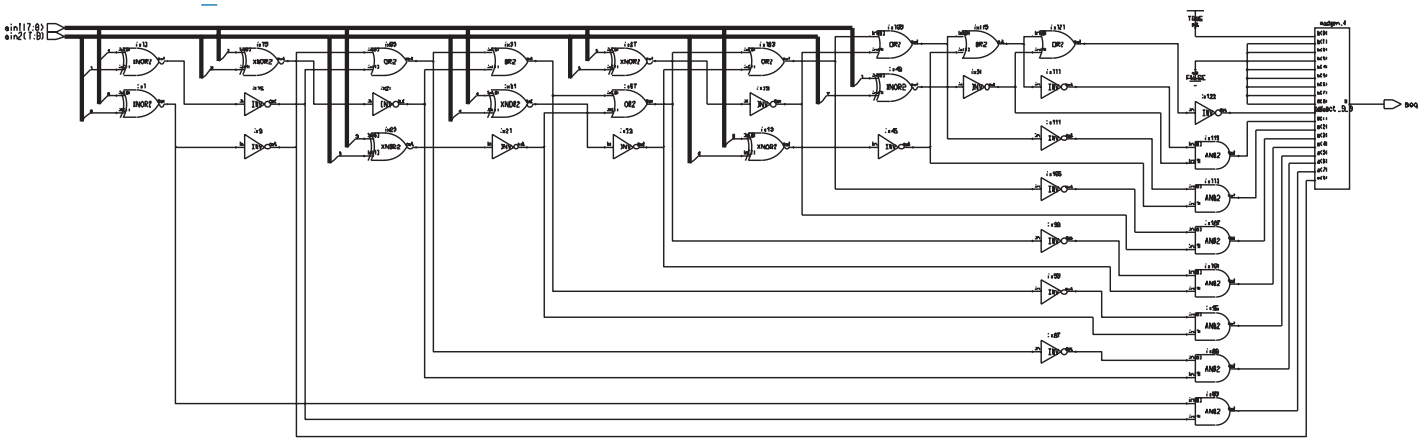


Figure 4

COMPARATOR_B
RTL Schematic

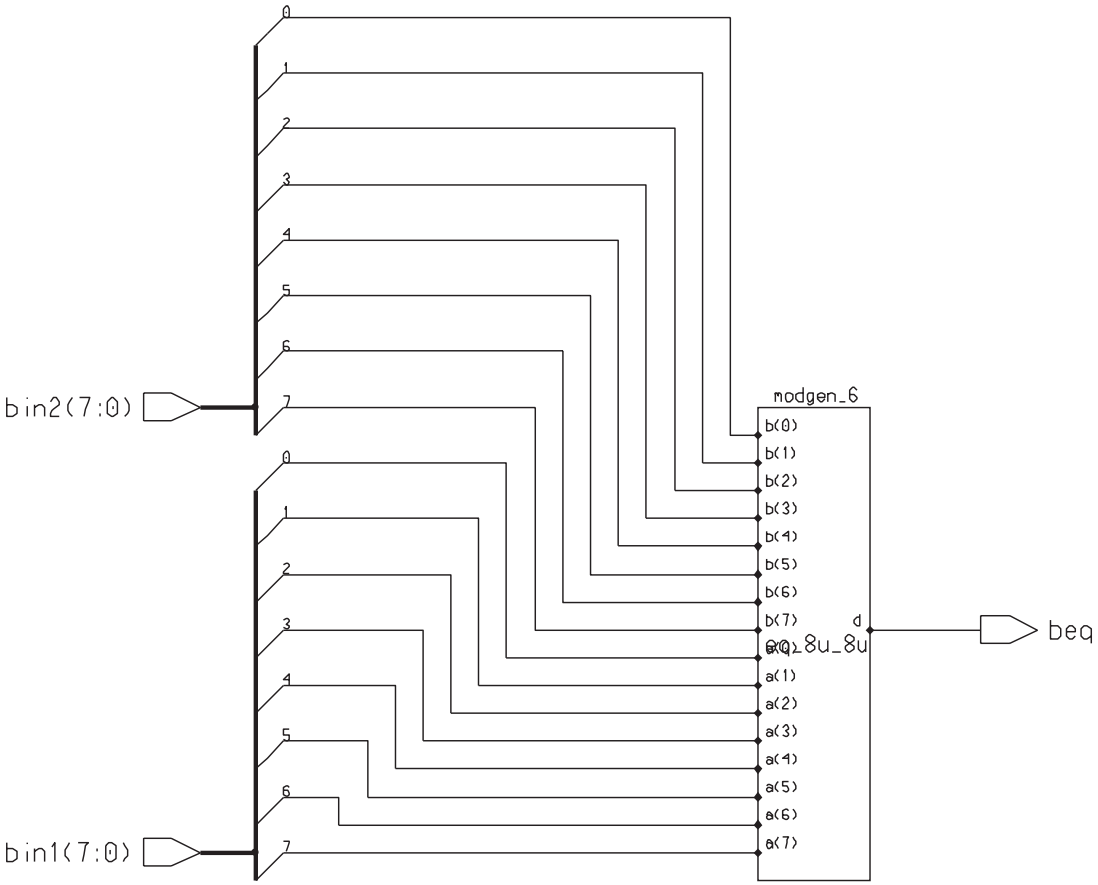


Figure 5

COMRATOR_A
Gate-level
Schematic

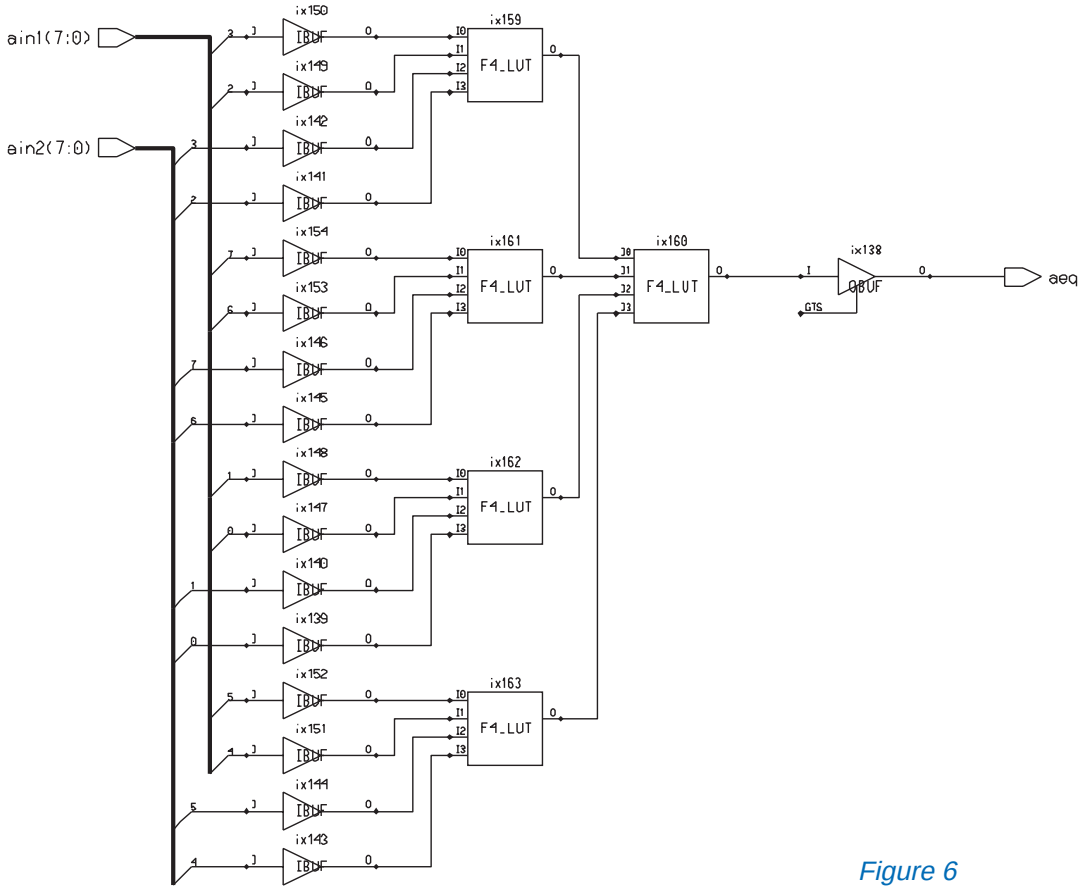


Figure 6

COMPARATOR_B Gate-level Schematic

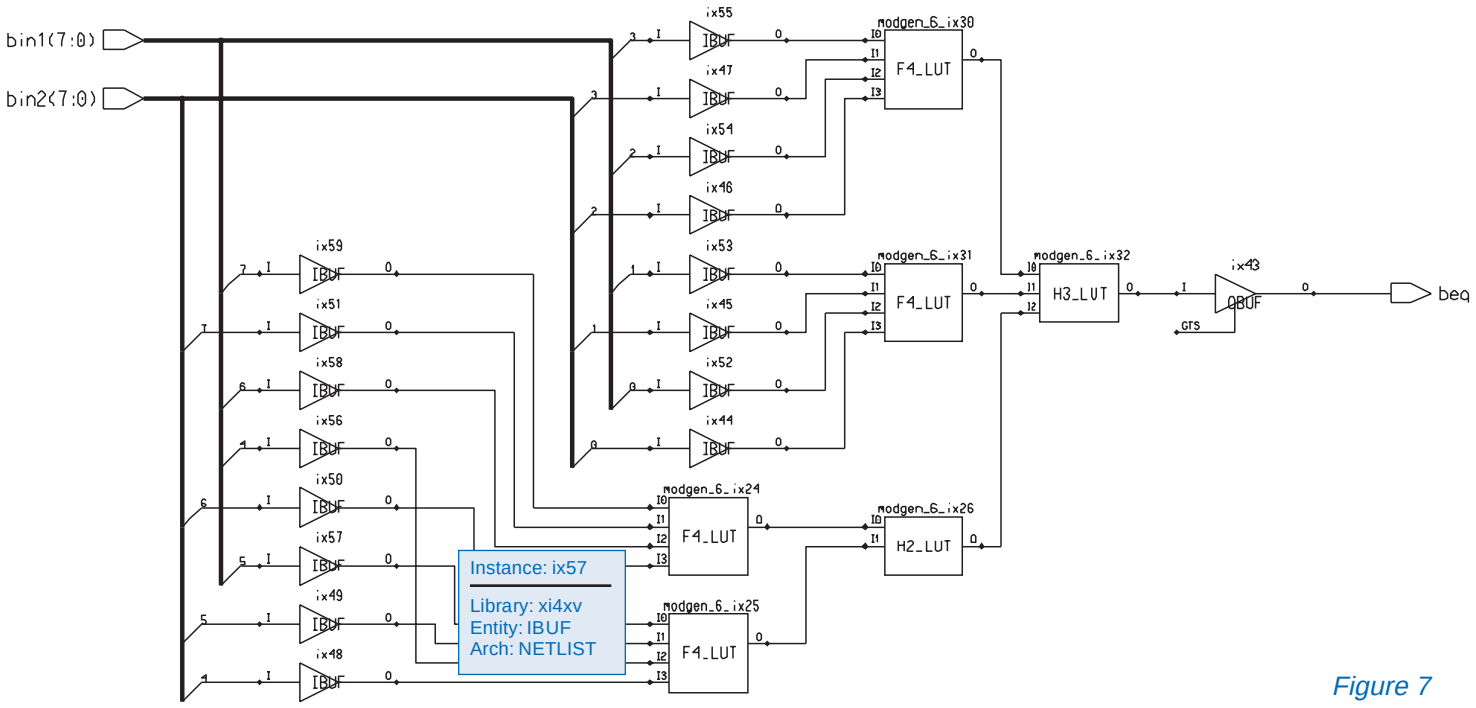


Figure 7

```
module COMPARATOR_D
(DIN1, DIN2, DGT, DLT, DEQ);
input [7:0] DIN1, DIN2;
output DGT, DLT, DEQ;

reg DGT, DLT, DEQ;

always@(DIN1 or DIN2)
begin: EQUALITY
DGT=0;
DLT=0;
DEQ=0;
if (DIN1 > DIN2)
  DGT=1;
 else
  if (DIN1 < DIN2)
  DLT=1;
  Else
// The “if” and equality compare is unnecessary
// only DEQ=1; is required after the “else”
  if (DIN1 == DIN2)
    DEQ=1;
end
endmodule
```

Figure 8

mapped operators ask your synthesis company about possible coding style dependencies or a future enhancement.

To prevent extra uneccessary logic watch for redundant compares when coding complex code sections. Some synthesizers will optimize out the redundancy, others will not.

Compares to a constant are usually smaller and faster than comparing two signals. **Figure 8** shows an example where the equality compare is redundant.

Conclusion

For best results, code your compares word-wise rather than bit-wise, check to see that your synthesizer uses carry logic for inequality operators (asking for help if it doesn't), remove all redundant compare operations, and compare to a constant rather than a signal when possible.

Code your compares word-wise rather than bit-wise, check to see that your synthesizer uses carry logic for inequality operators, remove all redundant compare operations, and compare to a constant rather than a signal when possible.

HOTLINE Q&A

by The Xilinx Hotline Staff

Q: Can the TAP (JTAG) pins of a Virtex device be set to SelectI/O standards other than LVTTL?

The TAP pins of a Virtex device cannot be changed. They always use the LVTTL SelectI/O standard.

Q: Can the TAP pins of a Virtex device be used as regular I/O?

The TAP pins of a Virtex device are fully dedicated boundary-scan pins. They cannot be used as regular I/O.

Q: When performing a functional simulation of a Virtex BlockRAM with VerilogXL, the following timing violation is reported by VerilogXL on the BlockRAM. What does this timing violation mean?

```
xxx: Timing violation in top.U1
 $recovery(posedge CLKB: 800, posedge CLKA: 800,
 1.0: 10);
```

The UNISIM Virtex BlockRAM Verilog model in Alliance Series 1.5i incorrectly models the relationship between the BlockRAM clocks. It does not allow both clocks on a BlockRAM to change at the same time. This is not accurate behavior in the case of a read.

There are two workarounds to this problem. The first workaround is to not let both clocks of the BlockRAM change at the same time, in functional simulation. This has the benefit of detecting a possible simultaneous write from both ports. Another workaround is to remove the $recovery directive from the UNISIM BlockRAM Verilog simulation model. This workaround is potentially dangerous because the illegal operation of performing a simultaneous write to the same location will not be flagged. This incorrect behavior will be fixed in an upcoming patch of the 1.5 software.

Q: How do you turn on the Virtex boundary-scan feature? Is there a Virtex boundary-scan symbol that must be instantiated?

The Virtex boundary-scan feature is always active. Nothing needs to be done to turn this feature on.

Q: What does the following error message mean if a Virtex design is compiled with FPGA Compiler I?

```
The target library does not contain all required gates.
Either a NOR, or an AND and an OR gate(two-input)
 is required for mapping. (OPT-102)
```

This message means that the replace_fpga command was used to compile a Virtex design with FPGA Compiler I. The command replace_fpga does not apply when compiling a Virtex design with FPGA Compiler I.

Q: What are the restrictions in using the MUXF6 in Virtex?

The data inputs of the MUXF6 must be connected to the output of MUXF5s.

Q: How do you place an external clock on one of the Virtex global clock nets?

Connect the top-level port in your HDL design to the input of an IBUFG. Next, connect the output of the IBUFG to a BUFG. The output of the BUFG will use the Virtex dedicated clock resources.

Q: Can A TNM be applied to a Virtex shift register LUT?

A TNM can be applied to a Virtex shift register LUT (SRL). Additionally, a Virtex shift register LUT is considered a part of the FFS timegroup. This means that if a constraint file has a TNM that applies to all FFSs in a design that the TNM will apply to all flip-flops and shift register LUTs. The **TNM=FFS:designregisters** attribute would place the TNM 'designregisters' on all flip-flops and shift register LUTs.

Virtex™ 2.5 V Field Programmable Gate Arrays

November 9, 1998 (Version 1.1 - ADVANCE) Product Specification

Features

- Fast, high-density Field-Programmable Gate Arrays
  - Densities from 50k to 1M system gates
  - System performance up to 200 MHz
  - 66-MHz PCI Compliant
  - Hot-swappable for Compact PCI
- Multi-standard SelectIO™ interfaces
  - 16 high-performance interface standards
  - Connects directly to ZBTRAM devices
- Built-in clock-management circuitry
  - Four dedicated delay-locked loops (DLLs) for advanced clock control
  - Four primary low-skew global clock distribution nets, plus 24 secondary global nets
- Hierachical memory system
  - LUTs configurable as 16-bit RAM, 32-bit RAM, 16-bit dual-ported RAM, or 16-bit Shift Register
  - Configurable synchronous dual-ported 4k-bit RAMs
  - Fast interfaces to external high-performance RAMs
- Flexible architecture that balances speed and density
  - Dedicated carry logic for high-speed arithmetic
  - Dedicated multiplier support
  - Cascade chain for wide-input functions
  - Abundant registers/latches with clock enable, and dual synchronous/asynchronous set and reset
  - Internal 3-state bussing
  - IEEE 1149.1 boundary-scan logic
  - Die-temperature sensing device
- Supported by FPGA Foundation™ and Alliance Development Systems
  - Complete support for Unified Libraries, Relationally Placed Macros, and Design Manager
  - Wide selection of PC and workstation platforms
- SRAM-based in-system configuration
  - Unlimited reprogrammability
  - Four programming modes
- 0.22-µm five-layer metal process
- 100% factory tested

Description

The Virtex FPGA family delivers high-performance, high-capacity programmable logic solutions. Dramatic increases in silicon efficiency result from optimizing the new architecture for place-and-route efficiency and exploiting an aggressive 5-layer-metal 0.22-µm CMOS process. These advances make Virtex FPGAs powerful and flexible alternatives to mask-programmed gate arrays. The Virtex family comprises the nine members shown in Table 1.

Building on experience gained from previous generations of FPGAs, the Virtex family represents a revolutionary step forward in programmable logic design. Combining a wide variety of programmable system features, a rich hierarchy of fast, flexible interconnect resources, and advanced process technology, the Virtex family delivers a high-speed and high-capacity programmable logic solution that enhances design flexibility while reducing time-to-market.

Table 1: Virtex Field-Programmable Gate Array Family Members.

Device	System Gates	CLB Array	Logic Cells	Maximum Available I/O	BlockRAM Bits	Max Select RAM Bits
XCV50	57,906	16x24	1,728	180	32,768	24,576
XCV100	108,904	20x30	2,700	180	40,960	38,400
XCV150	164,674	24x36	3,888	260	49,152	55,296
XCV200	236,666	28x42	5,292	284	57,344	75,264
XCV300	322,970	32x48	6,912	316	65,536	98,304
XCV400	468,252	40x60	10,800	404	81,920	153,600
XCV600	661,111	48x72	15,552	500	98,304	221,184
XCV800	888,439	56x84	21,168	514	114,688	301,056
XCV1000	1,124,022	64x96	27,648	514	131,072	393,216

Virtex Architecture

Virtex devices feature a flexible, regular architecture that comprises an array of configurable logic blocks (CLBs) surrounded by programmable input/output blocks (IOBs), all interconnected by a rich hierarchy of fast, versatile routing resources. The abundance of routing resources permits the Virtex family to accommodate even the largest and most complex designs.

Virtex FPGAs are SRAM-based, and are customized by loading configuration data into internal memory cells. In some modes, the FPGA reads its own configuration data from an external PROM (master serial mode). Otherwise, the configuration data is written into the FPGA (SelectMAP™ and slave serial modes).

The standard Xilinx Foundation™ and Alliance Series™ Development systems deliver complete design support for Virtex, covering every aspect from behavioral and schematic entry, through simulation, automatic design translation and implementation, to the creation, downloading, and readback of a configuration bit stream.

Higher Performance

Virtex devices provide better performance than previous generations of FPGA. Designs can achieve synchronous system clock rates up to 200 MHz including I/O.

Table 2: Performance for Common Circuit Functions

Function	Bits	Virtex -6
Register-to-Register		
Adder	16 64	5.0 ns 7.2 ns
Pipelined Multiplier	8 x 8 16 x 16	5.1 ns 6.0 ns
Address Decoder	16 64	4.4 ns 6.4 ns
16:1 Multiplexer		5.4 ns
Parity Tree	9 18 36	4.1 ns 5.0 ns 6.9 ns
Chip-to-Chip		
HSTL Class IV		200 MHz
LVTTL,16mA, fast slew		180 MHz

Virtex inputs and outputs comply fully with PCI specifications, and interfaces can be implemented that operate at 33 MHz or 66 MHz. Additionally, Virtex supports the hot-swapping requirements of Compact PCI.

Xilinx thoroughly benchmarked the Virtex family. While performance is design-dependent, many designs operated internally at speeds in excess of 100 MHz and can achieve 200 MHz. Table 2 shows performance data for representative circuits, using worst-case timing parameters.

Architectural Description

Virtex Array

The Virtex user-programmable gate array, shown in Figure 1, comprises two major configurable elements: configurable logic blocks (CLBs) and input/output blocks (IOBs).

- CLBs provide the functional elements for constructing logic
- IOBs provide the interface between the package pins and the CLBs

CLBs interconnect through a general routing matrix (GRM). The GRM comprises an array of routing switches located at the intersections of horizontal and vertical routing channels. Each CLB nests into a VersaBlock™ that also provides local routing resources to connect the CLB to the GRM.

The VersaRing™ I/O interface provides additional routing resources around the periphery of the device. This routing improves I/O routability and facilitates pin locking.

The Virtex architecture also includes the following circuits that connect to the GRM.

- Dedicated block memories of 4096 bits each
- Clock DLLs for clock-distribution delay compensation and clock domain control
- 3-State buffers (BUFTs) associated with each CLB that drive dedicated segmentable horizontal routing resources

Values stored in static memory cells control the configurable logic elements and interconnect resources. These values load into the memory cells on power-up, and can reload if necessary to change the function of the device.

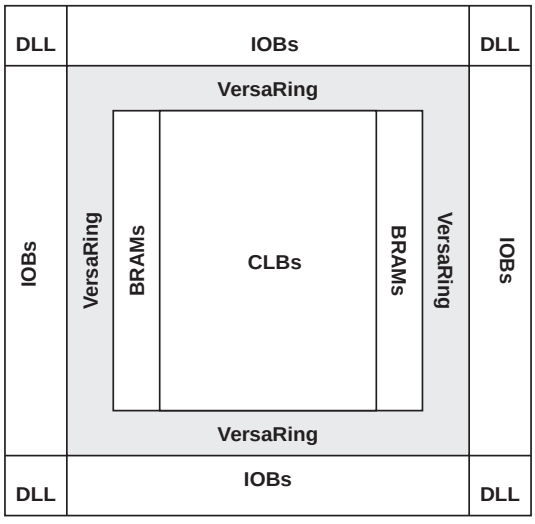


Figure 1: Virtex Architecture Overview

A set of Supplementary Description documents published separately augment the following description of the various Virtex-architecture components. The Supplementary Descriptions provide more detailed information and cover the following topics.

- Input/Output Block
- Configurable Logic Block
- Memory Resources
- Clock Distribution
- Routing Resources
- Configuration and Readback
- Boundary Scan
- Power Consumption

Input/Output Block

The Virtex IOB, Figure 2, features SelectIO™ inputs and outputs that support a wide variety of I/O signalling standards, see Table 3. These high-speed inputs and outputs are capable of supporting PCI interfaces up to 66 MHz.

The three IOB storage elements function either as edge-triggered D-type flip-flops or as level sensitive latches. Each IOB has a clock signal (CLK) shared by the three flip-flops and independent clock enable signals for each flip-flop.

In addition to the CLK and CE control signals, the three flip-flops share a Set/Reset (SR). For each flip-flop, this signal can be independently configured as a synchronous Set, a synchronous Reset, an asynchronous Preset, or an asynchronous Clear.

The input and output buffers and all of the IOB control signals have independent polarity controls.

All pads are protected against damage from electrostatic discharge (ESD) and from over-voltage transients. Two forms of over-voltage protection are provided, one that permits 5-V compliance, and one that does not. For 5-V compliance, a zener-like structure connected to ground turns on when the output rises to approximately 6.5 V. When 5-V compliance is not required, a conventional clamp diode may be connected to the output supply voltage, V_{CCO}. The type of over-voltage protection can be selected independently for each pad.

Optional pull-up and pull-down resistors and an optional weak-keeper circuit are attached to each pad. Prior to configuration all outputs not involved in configuration are forced into their high-impedance state. The pull-up and pull-down resistors and the weak-keeper circuit are inactive, and input floats. If the design requires a defined input logic level prior to configuration, an external resistor must be used.

All Virtex IOBs support IEEE 1149.1-compatible boundary scan testing.

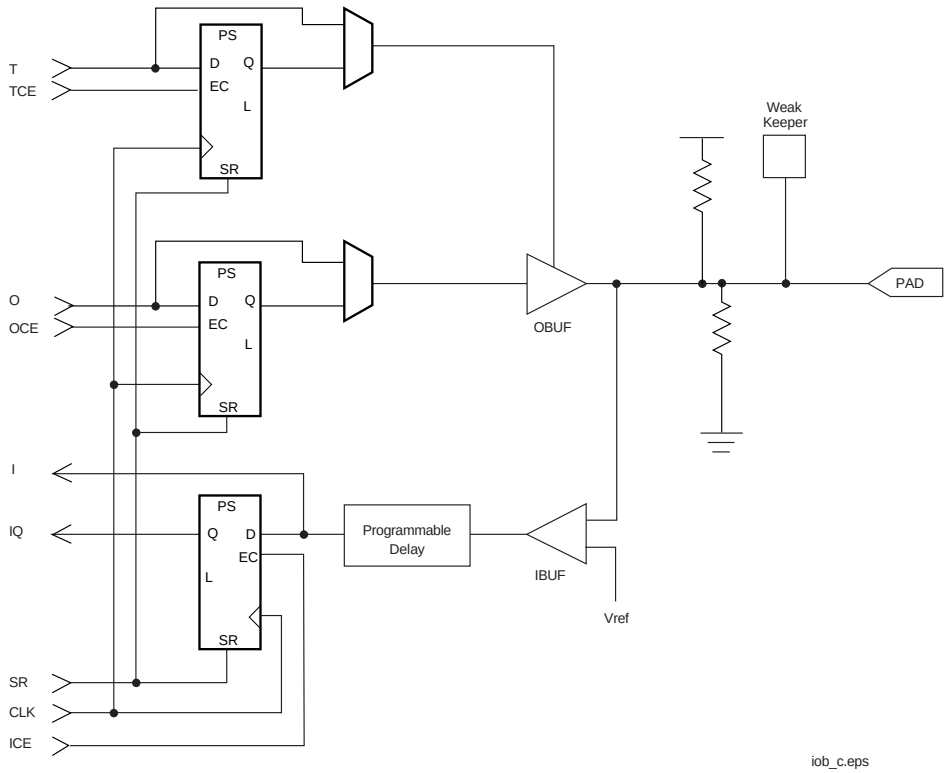


Figure 2: Virtex Input/Output Block (IOB)

Table 3: Supported Select I/O Standards

I/O Standard	Input Reference Voltage (V_{REF})	Output Source Voltage (V_{CCO})	Board Termination Voltage (V_{TT})
LVTTL 2 – 24 mA	N/A	3.3	N/A
LVCMOS2	N/A	2.5	N/A
PCI	N/A	3.3	N/A
GTL	0.8	N/A	1.2
GTL+	1.0	N/A	1.5
HSTL Class I	0.75	1.5	1.5
HSTL Class III	0.75	1.5	1.5
HSTL Class IV	0.75	1.5	1.5
SSTL3 Class I and II	1.5	3.3	1.5
SSTL2 Class I and II	1.125	2.5	1.125
CTT	1.5	3.3	1.5
AGP	1.32	3.3	N/A

Input Path

A buffer In the Virtex IOB input path routes the input signal either directly to internal logic or through an optional input flip-flop.

An optional delay element at the D-input of this flip-flop eliminates pad-to-pad hold time. The delay is matched to the internal clock-distribution delay of the FPGA, and when used, assures that the pad-to-pad hold time is zero.

Each input buffer can be configured to conform to any of the low-voltage signalling standards supported. In some of these standards the input buffer utilizes a user-supplied threshold voltage, V_{REF}. The need to supply V_{REF} imposes constraints on which standards can used in close proximity to each other. See “I/O Banking”.

There are optional pull-up and pull-down resistors at each input for use after configuration. Their value is in the range 50 – 150 kohms.

Output Path

The output path includes a 3-state output buffer that drives the output signal onto the pad. The output signal can be routed to the buffer directly from the internal logic or through an optional IOB output flip-flop.

The 3-state control of the output can also be routed directly from the internal logic or through a flip-flip that provides synchronous enable and disable.

Each output driver can be individually programmed for a wide range of low-voltage signalling standards. Each output buffer can source up to 24 mA and sink up to 48mA. Drive strength and slew rate controls minimize bus transients.

In most signalling standards, the output High voltage depends on an externally supplied V_{CCO} voltage. The need to supply V_{CCO} imposes constraints on which standards can be used in close proximity to each other. See “I/O Banking” below.

An optional weak-keeper circuit is connected to each output. When selected, the circuit monitors the voltage on the pad and weakly drives the pin High or Low to match the input signal. If the pin is connected to a multiple-source signal, the weak keeper holds the signal in its last state if all drivers are disabled. Maintaining a valid logic level in this way eliminates bus chatter.

Because the weak-keeper circuit uses the IOB input buffer to monitor the input level, an appropriate V_{REF} voltage must be provided if the signalling standard requires one. The provision of this voltage must comply with the I/O banking rules.

I/O Banking

Some of the I/O standards described above require V_{CCO} and/or V_{REF} voltages. These voltages externally and connected to device pins that serve groups of IOBs, called banks. Consequently, restrictions exist about which I/O standards can be combined within a given bank.

Eight I/O banks result from separating each edge of the FPGA into two banks, as shown in Figure 3. Each bank has multiple V_{CCO} pins, all of which must be connected to the same voltage. This voltage is determined by the output standards in use.

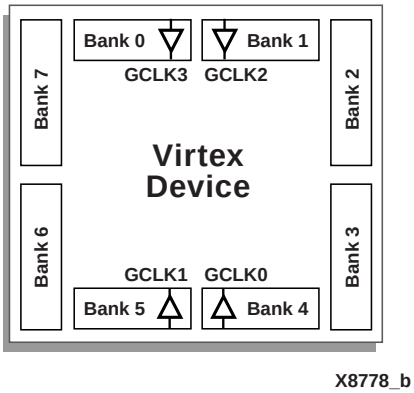


Figure 3: Virtex I/O Banks

Within a bank, output standards may be mixed only if they use the same V_{CCO}. Compatible standards are shown in Table 4. GTL and GTL+ appear under all voltages because their open-drain outputs do not depend on V_{CCO}.

Table 4: Compatible Output Standards

V_{CCO}	Compatible Standards
3.3 V	PCI, LVTTL, SSTL3 I, SSTL3 II, CTT, AGP, GTL, GTL+
2.5 V	SSTL2 I, SSTL2 II, LVCMOS2, GTL, GTL+
1.5 V	HSTL I, HSTL III, HSTL IV, GTL, GTL+

Some input standards require a user-supplied threshold voltage, V_{REF}. In this case, certain user-I/O pins are automatically configured as inputs for the V_{REF} voltage. Approximately one in six of the I/O pins in the bank assume this role.

The V_{REF} pins within a bank are interconnected internally and consequently only one V_{REF} voltage can be used within each bank. All V_{REF} pins in the bank, however, must be connected to the external voltage source for correct operation.

Within a bank, inputs that require V_{REF} can be mixed with those that do not. However, only one V_{REF} voltage may be used within a bank.Input buffers that use V_{REF} are not 5V-tolerant.

The V_{CCO} and V_{REF} pins for each bank appear in the device pin-out tables and diagrams. The diagrams also show the bank affiliation of each I/O.

Within a given package, the number of V_{REF} and V_{CCO} pins can vary depending on the size of device. In larger devices, more I/O pins convert to V_{REF} pins. Since these are always a superset of the V_{REF} pins used for smaller devices, it is possible to design a PCB that permits migration to a larger device if necessary. All the V_{REF} pins for the largest device anticipated must be connected to the V_{REF} voltage, and not used for I/O.

In smaller devices, some V_{CCO} pins used in larger devices do not connect within the package. These unconnected pins may be left unconnected externally, or may be connected to the V_{CCO} voltage to permit migration to a larger device if necessary.

In HQ and PQ packages, all V_{CCO} pins are bonded together internally, and consequently the same V_{CCO} voltage must be connected to all of them. The V_{REF} pins remain internally connected as eight banks, and may be used as described previously.

Configurable Logic Block

The basic building block of the Virtex CLB is the logic cell (LC). An LC includes a 4-input function generator, carry logic, and a storage element. The output from the function generator in each LC drives both the CLB output and the D input of the flip-flop. Each Virtex CLB contains four LCs, organized in two similar slices, as shown in Figure 4. Figure 5 shows a more detailed view of a single slice.

In addition to the four basic LCs, the Virtex CLB contains logic that combines function generators to provide functions of five or six inputs. Consequently, when estimating the number of system gates provided by a given device, each CLB counts as 4.5 LCs.

Look-Up Tables

Virtex function generators are implemented as 4-input look-up tables (LUTs). In addition to operating as a function generator, each LUT can provide a 16 x 1-bit synchronous RAM. Furthermore, the two LUTs within a slice can be combined to create a 16 x 2-bit or 32 x 1-bit synchronous RAM, or a 16x1-bit dual-port synchronous RAM.

The Virtex LUT can also provide a 16-bit shift register that is ideal for capturing high-speed or burst-mode data. This mode can also be used to store data in applications such as Digital Signal Processing.

Storage Elements

The storage elements in the Virtex slice can be configured either as edge-triggered D-type flip-flops or as level-sensitive latches. The D inputs can be driven either by the function generators within the slice or directly from slice inputs, bypassing the function generators.

In addition to Clock and Clock Enable signals, each Slice has synchronous Set and Reset signals (SR and BY). Alternatively, these signals may be configured as asynchronous Preset and Clear

All of the control signals are independently invertible, and are shared by the two flip-flops within the slice.

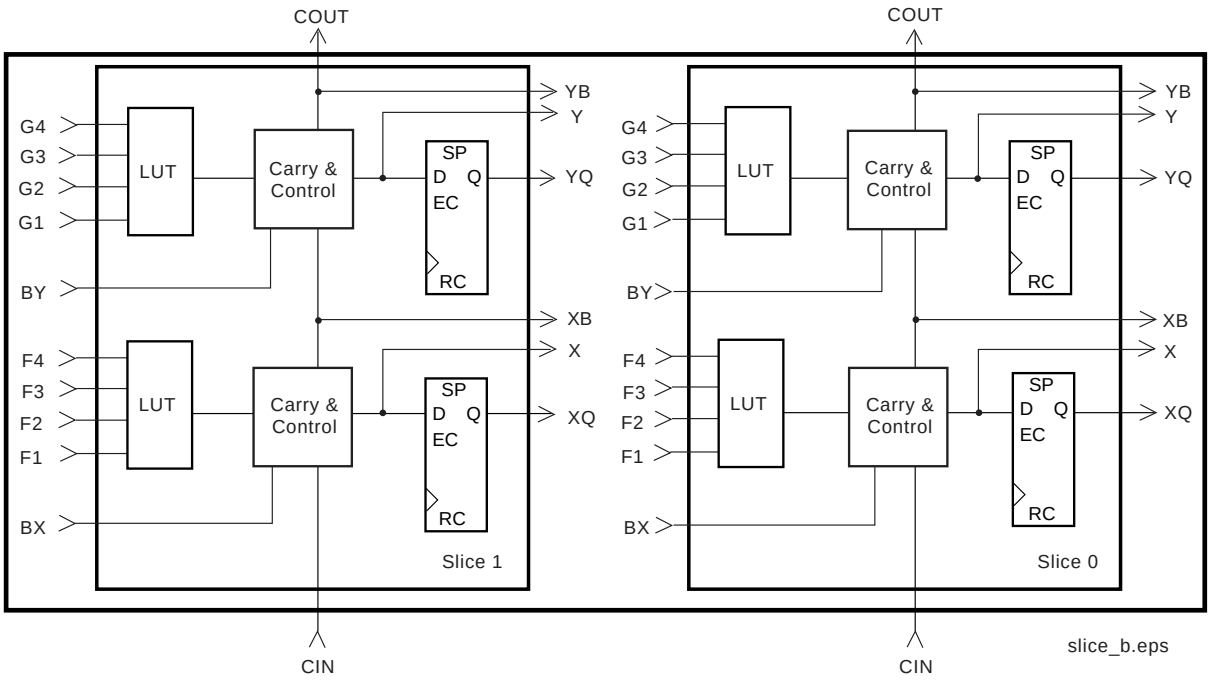


Figure 4: 2-slice Virtex CLB

Additional Logic

The F5 multiplexer in each slice combines the function generator outputs. This combination provides either a function generator that can implement any 5-input function, a 4:1 multiplexer, or selected functions of up to nine inputs.

Similarly, the F6 multiplexer combines the outputs of all four function generators in the CLB by selecting one of the F5-multiplexer outputs. This permits the implementation of any 6-input function, an 8:1 multiplexer, or selected functions of up to 19 inputs.

Each CLB has four direct feedthrough paths, one per LC. These paths provide extra data input lines or additional local routing that does not consume logic resources.

Arithmetic Logic

Dedicated carry logic provides fast arithmetic carry capability for high-speed arithmetic functions. The Virtex CLB supports two separate carry chains, one per Slice. The height of the carry chains is two bits per CLB.

The arithmetic logic includes an XOR gate that allows a 1-bit full adder to be implemented within an LC. In addition, a dedicated AND gate improves the efficiency of multiplier implementation.

The dedicated carry path can also be used to cascade function generators for implementing wide logic functions.

BUFTs

Each Virtex CLB contains two 3-state drivers (BUFTs) that can drive on-chip busses. See “Dedicated Routing” on page 4 9. Each Virtex BUFT has an independent 3-state control pin and an independent input pin.

Block RAM

Virtex FPGAs incorporate several large BlockSelectRAM+ memories. These complement the distributed SelectRAM+ LUTRAMs that provide shallow RAM structures implemented in CLBs.

BlockSelectRAM+ memory blocks are organized in columns. All Virtex devices contain two such columns, one along each vertical edge. These columns extend the full height of the chip. Each memory block is four CLBs high, and consequently, a Virtex device 64 CLBs high will contain16 memory blocks per column, and a total of 32 blocks.

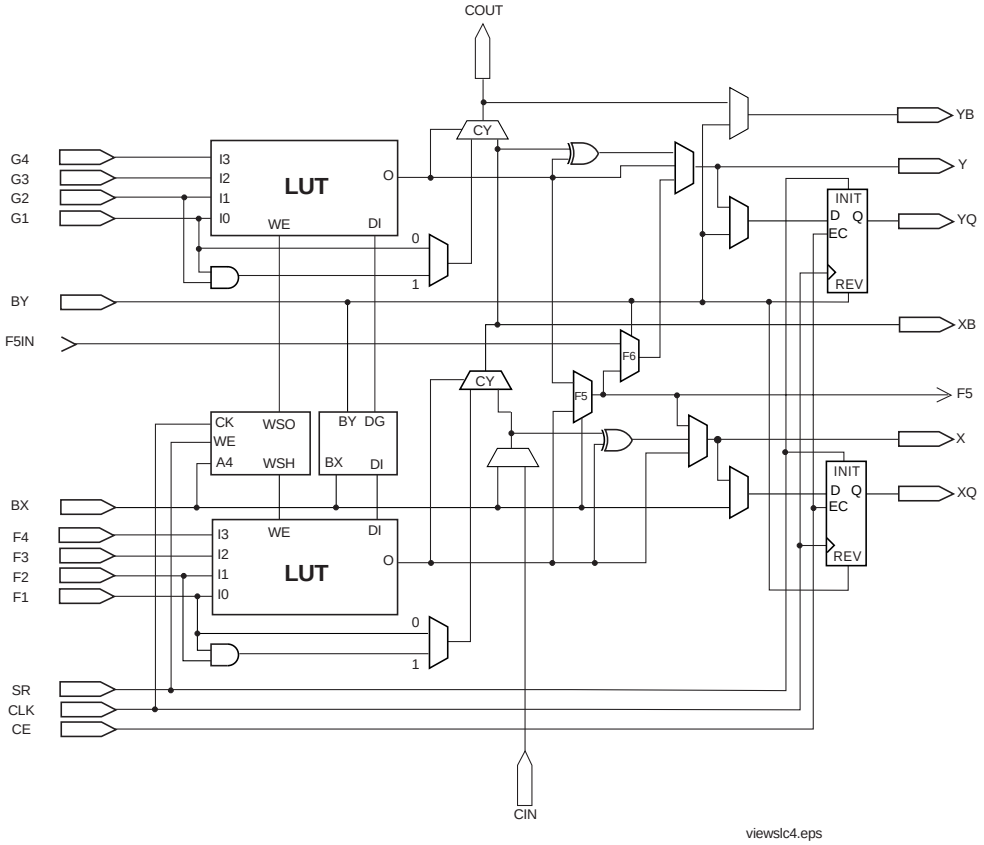


Figure 5: Detailed View of Virtex Slice

Table 5 shows the amount of Block SelectRAM+ memory that is available in each Virtex device.

Each Block SelectRAM+ cell, as illustrated in Figure 6, is a fully synchronous dual-ported 4096-bit RAM with independent control signals for each port. The data widths of the two ports can be configured independently, providing built-in bus-width conversion.

Table 5: Virtex Block SelectRAM+ Amounts

Virtex Device	# of Blocks	Total Block SelectRAM+ Bits
XCV50	8	32,768
XCV100	10	40,960
XCV150	12	49,152
XCV200	14	57,344
XCV300	16	65,536
XCV400	20	81,920
XCV600	24	98,304
XCV800	28	114,688
XCV1000	32	131,072

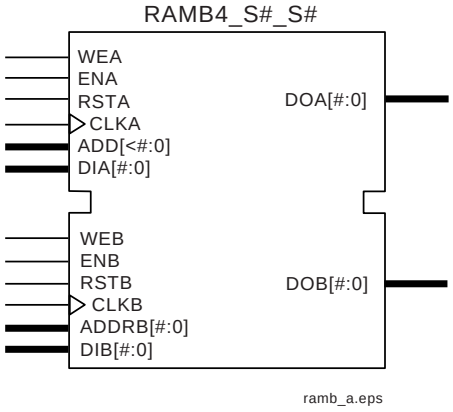


Figure 6: Dual-Port Block SelectRam+

Table 6 shows the depth and width aspect ratios for the Block SelectRAM+

Table 6: Block SelectRAM+ Port Aspect Ratios

Width	Depth	ADDR Bus	Data Bus
1	4096	ADDR<11:0>	DATA<0>
2	2048	ADDR<10:0>	DATA<1:0>
4	1024	ADDR<9:0>	DATA<3:0>
8	512	ADDR<8:0>	DATA<7:0>
16	256	ADDR<7:0>	DATA<15:0>

The Virtex block RAM also includes dedicated routing to provide an efficient interface with both CLBs and other block RAMs.

Programmable Routing Matrix

It is the longest delay path that limits the speed of any worst-case design. Consequently, the Virtex routing architecture and its place-and-route software were defined in a single optimization process. This joint optimization minimizes long-path delays, and consequently, yields the best system performance.

The joint optimization also reduces design compilation times because the architecture is software-friendly. Design cycles are correspondingly reduced due to shorter design iteration times.

Local Routing

The VersaBlock provides local routing resources, as shown in Figure 7, providing the following three types of connections.

- Interconnections among the LUTs, flip-flops, and GRM
- Internal CLB feedback paths that provide high-speed connections to LUTs within the same CLB, chaining them together with minimal routing delay
- Direct paths that provide high-speed connections between horizontally adjacent CLBs, eliminating the delay of the GRM.

General Purpose Routing

Most Virtex signals are routed on the general purpose routing, and consequently, the majority of interconnect resources are associated with this level of the routing hierarchy. The general routing resources are located in horizontal and vertical routing channels associated with the rows and columns CLBs. The general-purpose routing resources are listed below.

- Adjacent to each CLB is a General Routing Matrix (GRM). The GRM is the switch matrix through which horizontal and vertical routing resources connect, and is also the means by which the CLB gains access to the general purpose routing.
- 24 single-length lines route GRM signals to adjacent GRMs in each of the four directions.
- 96 buffered Hex lines route GRM signals to another GRMs six-blocks away in each one of the four directions. Organized in a staggered pattern, Hex lines may be driven only at their endpoints. Hex-line signals can be accessed either at the endpoints or at the midpoint (three blocks from the source). One third of the Hex lines are bidirectional, while the remaining ones are uni-directional.
- 12 Longlines are buffered, bidirectional wires that distribute signals across the device quickly and efficiently. Vertical Longlines span the full height of the device, and horizontal ones span the full width of the device.

I/O Routing

Virtex devices have additional routing resources around their periphery that form an interface between the CLB array and the IOBs. This additional routing, called the VersaRing, facilitates pin-swapping and pin-locking, such that logic redesigns can adapt to existing PCB layouts. Time-to-market is reduced, since PCBs and other system components can be manufactured while the logic design is still in progress.

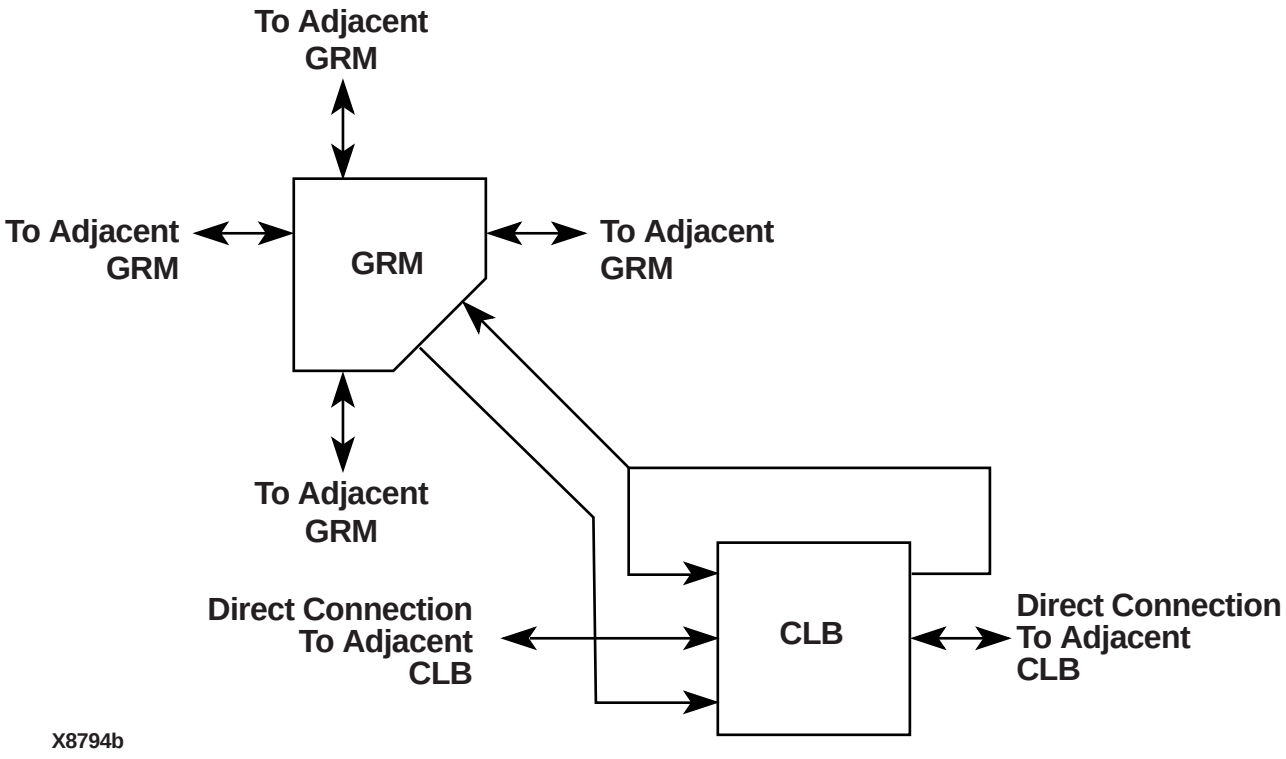


Figure 7: Virtex Local Routing

Dedicated Routing

Some classes of signal require dedicated routing resources to maximize performance. In the Virtex architecture, dedicated routing resources are provided for two classes of signal.

- Horizontal routing resources are provided for on-chip 3-state busses. Four partitionable bus lines are provided per CLB row, permitting multiple busses within a row, as shown in Figure 8.
- Two dedicated nets per CLB propagate carry signals vertically to the adjacent CLB.

Global Routing

Global Routing resources distribute clocks and other signals with very high fanout throughout the device. Virtex devices include two tiers of global routing resources referred to as primary and secondary global routing resources.

- The primary global routing resources are four dedicated global nets with dedicated input pins that are designed to distribute high-fanout clock signals with minimal skew. Each global clock net can drive all CLB, IOB, and block RAM clock pins. The primary global nets may only

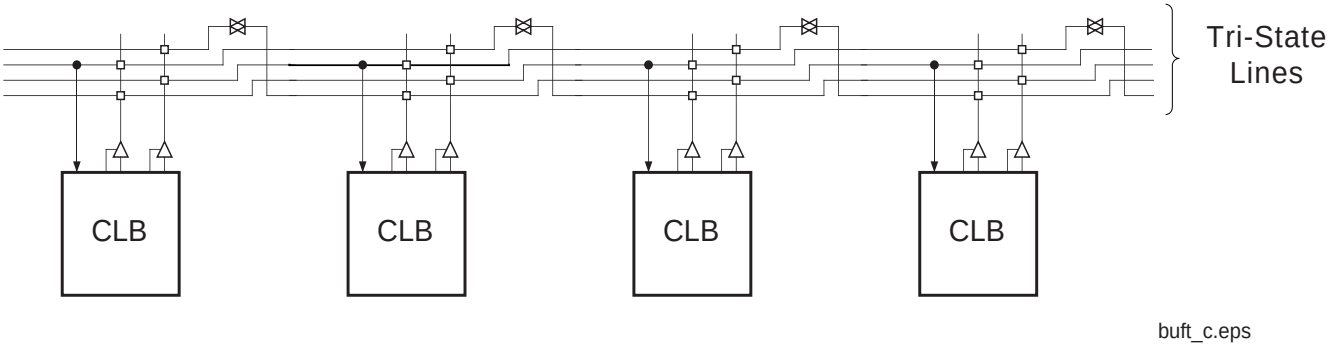


Figure 8: BUFT Connections to Dedicated Horizontal Bus Lines

be driven by global buffers. There are four global buffers, one for each global net.

- The secondary global routing resources consist of 24 backbone lines, 12 across the top of the chip and 12 across bottom. From these lines, up to 12 unique signals per column can be distributed via the 12 longlines in the column. These secondary resources are more flexible than the primary resources since they are not restricted to routing only to clock pins.

Clock Distribution

Virtex provides high-speed, low-skew clock distribution through the primary global routing resources described above. A typical clock distribution net is shown in Figure 9.

Four global buffers are provided, two at the top center of the device and two at the bottom center. These drive the four primary global nets that in turn drive any clock pin.

Four dedicated clock pads are provided, one adjacent to each of the global buffers. The input to the global buffer is selected either from these pads or from signals in the general purpose routing.

Delay-Locked Loop (DLL)

Associated with each global clock input buffer is a fully digital Delay-Locked Loop (DLL) that can eliminate skew between the clock input pad and internal clock-input pins throughout the device. Each DLL can drive two global clock networks.The DLL monitors the input clock and the distributed clock, and automatically adjusts a clock delay element. Additional delay is introduced such that clock edges reach internal flip-flops exactly one clock period after they arrive at the input. This closed-loop system effectively eliminates clock-distribution delay by ensuring that clock edges arrive at internal flip-flops in synchronism with clock edges arriving at the input.

In addition to eliminating clock-distribution delay, the DLL provides advanced control of multiple clock domains. The DLL provides four quadrature phases of the source clock, can double the clock, or divide the clock by 1.5, 2, 2.5, 3, 4, 5, 8, or 16. It has six outputs.

The DLL also operates as a clock mirror. By driving the output from a DLL off-chip and then back on again, the DLL can be used to deskew a board level clock among multiple Virtex devices.

In order to guarantee that the system clock is operating correctly prior to the FPGA starting up after configuration, the DLL can delay the completion of the configuration process until after it has achieved lock.

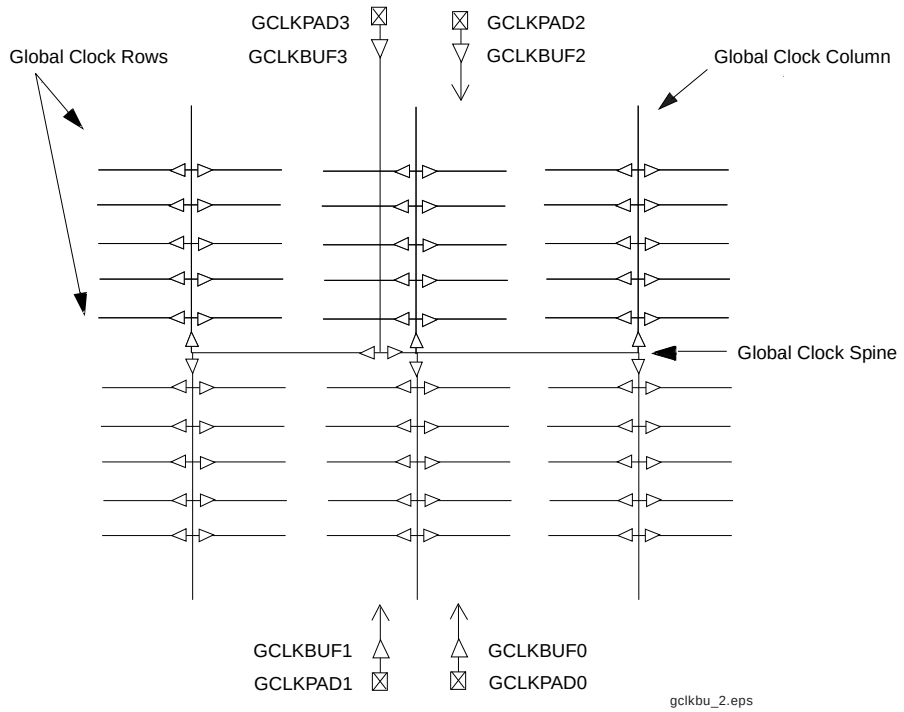


Figure 9: Global Clock Distribution Network

Boundary Scan

Virtex devices support all the mandatory boundary-scan instructions specifled in the IEEE standard 1149.1. A Test Access Port (TAP) and registers are provided that implement the EXTEST, SAMPLE/PRELOAD, and BYPASS instructions. The TAP also supports two USERCODE instructions and internal scan chains

Boundary-scan operation is independent of individual IOB conflgurations, and unaffected by package type. All IOBs, including unbonded ones, are treated as independent 3-state bidirectional pins in a single scan chain. Retention of the bidirectional test capability after configuration facilitates the testing of external interconnections.

Table 7 lists the boundary-scan instructions supported in Virtex FPGAs. Internal signals can be captured during EXTEST by connecting them to unbonded or unused IOBs. They may also be connected to the unused outputs of IOBs deflned as unidirectional input pins. This technique partially compensates for the absence of INTEST support.

Table 7: Boundary-Scan Instructions

Boundary-Scan Command	Binary Code(4:0)	Description
EXTEST	00000	Enables boundary-scan EXTEST operation
SAMPLE	00001	Enables boundary-scan SAMPLE operation
USR1	00010	Access user-defined register 1
USR2	00011	Access user-defined register 2
CFG_OUT	00100	Access the configuration bus for Readback
CFG_IN	00101	Access the configuration bus for Configuration
INTEST	00111	Enables boundary-scan intest operation
USRCODE	01000	Enables shifting out USER code
IDCODE	01001	Enables shifting out of ID Code
HIZ	01010	Tri-states output pins while enabling the Bypass Register
BUS_RST	01011	Reset the Configuration Bus
JSTART	01100	Clock the startup sequence when StartupClk is TCK
BYPASS	11111	Enables BYPASS
RESERVED	All other codes	Xilinx reserved instructions

The public boundary-scan instructions are available prior to configuration. After configuration, the public instructions remain available together with any USERCODE instructions installed during the configuration. While the SAMPLE and BYPASS instructions are available during configuration, it is recommended that boundary-scan operations not be performed during this transitional period.

In addition to the test instructions outlined above, the boundary-scan circuitry can be used to configure the FPGA, and also to read back the conflguration data.

To facilitate internal scan chains, the User Register provides three outputs (Reset, Update, and Shift) that represent the corresponding states in the boundary-scan internal state machine.

Development System

Virtex FPGAs are supported by the Xilinx Foundation and Alliance CAE tools. The basic methodology for Virtex design consists of three interrelated steps: design entry, implementation, and verifIcation. Industry-standard tools are used for design entry and simulation (for example, Synopsys FPGA Express), while Xilinx provides proprietary architecture-specifIc tools for implementation.

The Xilinx development system is integrated under the Xilinx Design Manager (XDM™) software, providing designers with a common user interface regardless of their choice of entry and verifIcation tools. The XDM software simplifies the selection of implementation options with pull-down menus and on-line help.

Application programs ranging from schematic capture to Placement and Routing (PAR) can be accessed through the XDM software. The program command sequence is generated prior to execution, and stored for documentation.

Several advanced software features facilitate Virtex design. RPMs, for example, are schematic-based macros with relative location constraints to guide their placement. They help ensure optimal implementation of common functions.

For HDL design entry, the Xilinx FPGA Foundation development system provides interfaces to the following synthesis design environments.

- Synopsys (FPGA Compiler, FPGA Express)
- Exemplar (Spectrum)
- Synplicity (Synplify)

For schematic design entry, the Xilinx FPGA Foundation and alliance development system provides interfaces to the following schematic-capture design environments.

- Mentor Graphics V8 (Design Architect, QuickSim II)
- Viewlogic Systems (Viewdraw)

Third-party vendors support many other environments.

A standard interface-file specification, Electronic Design Interchange Format (EDIF), simplifies file transfers into and out of the development system.

Virtex FPGAs supported by a unified library of standard functions. This library contains over 400 primitives and macros, ranging from 2-input AND gates to 16-bit accumulators, and includes arithmetic functions, comparators, counters, data registers, decoders, encoders, I/O functions, latches, Boolean functions, multiplexers, shift registers, and barrel shifters.

The “soft macro” portion of the library contains detailed descriptions of common logic functions, but does not contain any partitioning or placement information. The performance of these macros depends, therefore, on the partitioning and placement obtained during implementation.

RPMs, on the other hand, do contain predetermined partitioning and placement information that permits optimal implementation of these functions. Users can create their own library of soft macros or RPMs based on the macros and primitives in the standard library.

The design environment supports hierarchical design entry, with high-level schematics that comprise major functional blocks, while lower-level schematics define the logic in these blocks. These hierarchical design elements are automatically combined by the implementation tools. Different design entry tools can be combined within a hierarchical design, thus allowing the most convenient entry method to be used for each portion of the design.

Design Implementation

The place-and-route tools (PAR) automatically provide the implementation flow described in this section. The partitioner takes the EDIF netlist for the design and maps the logic into the architectural resources of the FPGA (CLBs and IOBs, for example). The placer then determines the best locations for these blocks based on their interconnections and the desired performance. Finally, the router interconnects the blocks.

The PAR algorithms support fully automatic implementation of most designs. For demanding applications, however, the user can exercise various degrees of control over the process. User partitioning, placement, and routing information is optionally specified during the design-entry process. The implementation of highly structured designs can benefit greatly from basic floorplanning.

The implementation software incorporates Timing Wizard® timing-driven placement and routing. Designers specify timing requirements along entire paths during design entry. The timing path analysis routines in PAR then recognize these user-specified requirements and accommodate them.

Timing requirements are entered on a schematic in a form directly relating to the system requirements, such as the targeted clock frequency, or the maximum allowable delay between two registers. In this way, the overall performance of the system along entire signal paths is automatically tailored to user-generated specifications. Specific timing information for individual nets is unnecessary.

Design Verification

In addition to conventional software simulation, FPGA users can use in-circuit debugging techniques. Because Xilinx devices are infinitely reprogrammable, designs can be verified in real time without the need for extensive sets of software simulation vectors.

The development system supports both software simulation and in-circuit debugging techniques. For simulation, the system extracts the post-layout timing information from the design database, and back-annotates this information into the netlist for use by the simulator. Alternatively, the user can verify timing-critical portions of the design using the TRACE® static timing analyzer.

For in-circuit debugging, the development system includes a download and readback cable. This cable connects the FPGA in the target system to a PC or workstation. After downloading the design into the FPGA, the designer can single-step the logic, readback the contents of the flip-flops, and so observe the internal logic state. Simple modi?cations can be downloaded into the system in a matter of minutes.

Configuration

Virtex devices are configured by loading configuration data into the internal configuration memory. Some of the pins used for this are dedicated configuration pins, while others may be re-used as general purpose inputs and outputs once con?guration is complete.

The dedicated pins are the mode pins (M2, M1, M0), the configuration clock pin (CCLK), the PROGRAM pin, the DONE pin and the boundary-scan pins (TDI, TDO, TMS, TCK). Depending on the configuration mode chosen, CCLK may be an output generated by the FPGA, or may be generated externally, and provided to the FPGA as an input.

For a more detailed description than that given below, see the Supplementary Description on Configuration and Readback.

For the full Virtex Data Sheet, see
www.xilinx.com/products/virtex.htm

Virtex I/V Curves for Various Output Options

January 4, 1999 (Version 1.0)

Application Note

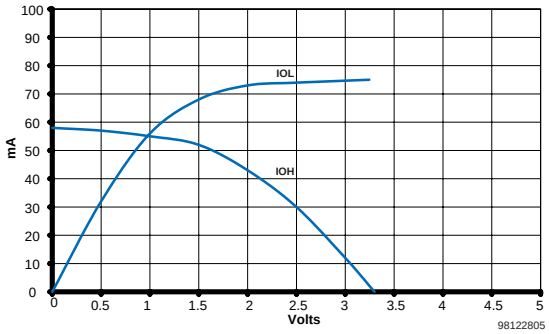
These typical curves describe the output sink and source current for average processing, nominal supply voltage and room temperature. For additional data see the Xilinx IBIS files at: http://www.xilinx.com/techdocs/htm_index/sw_ibis.htm

Note: The Xilinx IBIS files also list the extremes of the processing, temperature and voltage conditions as “min” and “max” (delay). Some tools rename these values as “fast-strong” and “slow-weak”. This has nothing to do with the FAST or SLOW slew-rate configuration options, invoked as attributes for each output pin (the default is SLOW).

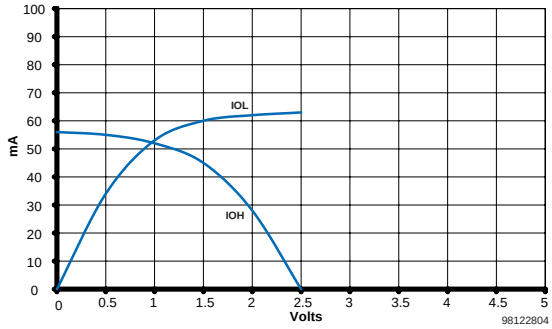
These slew-rate options are covered in separate IBIS files, with a trailing **s** for SLOW slew rate, and **f** for FAST slew rate.

Virtex Series

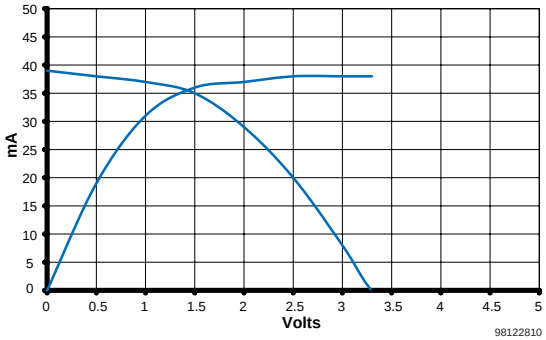
LVTTL



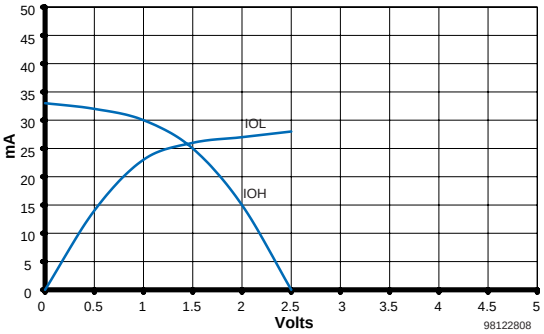
LVCMOS2



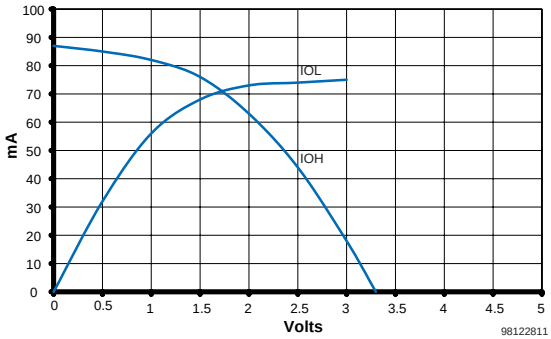
SSTL3I



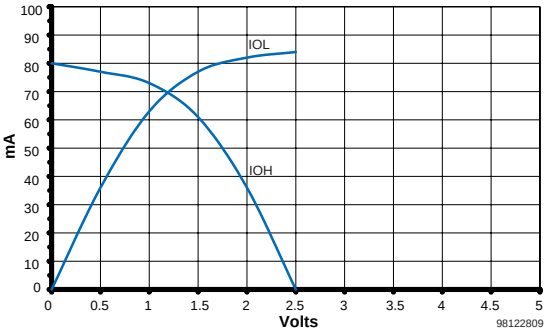
SSTL2I



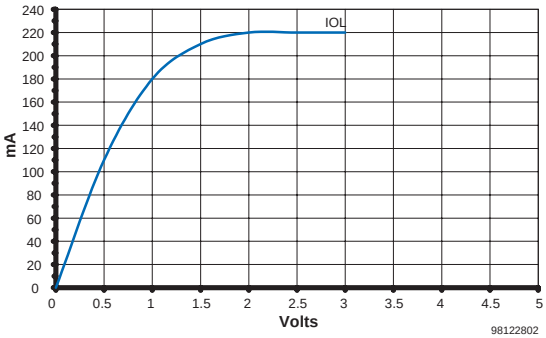
SSTL3II



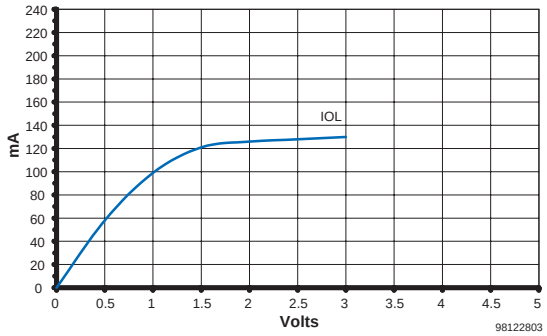
SSTL2II



GTL



GTL+



COLUMN

XILINX NEWS

Recent press releases and announcements, with Web references for further information.

Press Releases

You can find the full text of all Xilinx press releases at: http://www.xilinx.com/prs_rls/prs_rls.htm

First Million-Gate Virtex FPGA Delivered By UMC Group - Xilinx Team

November 30, 1998 – The United Silicon Inc. (USIC) wafer foundry in Taiwan, in which the UMC Group and Xilinx are equity partners, has successfully completed development of an advanced 0.22µ process technology. The new one-poly, five-layer metal (1P/5M) dual-voltage process enabled the production of the new Xilinx Virtex series of field programmable gate arrays (FPGAs), including the industry's first million-gate, 75-million transistor device. Xilinx began sampling the million-gate Virtex FPGA in October and expects to begin production of all nine members of the Virtex product line in early 1999.

See: http://www.xilinx.com/prs_rls/umcvirtex.htm

Xilinx, UMC Group Set New 1GHz Performance Milestone With 0.18µ, 1.8V Prototype FPGA

November 30, 1998 – A unique co-development model in place between the UMC Group and Xilinx, Inc., (NASDAQ:XLNX), has led to the production of the first field programmable gate arrays (FPGAs) capable of operating at speeds of one gigahertz *(see picture on page 57)*. The prototype FPGAs, derivatives of the XC4036XV family, were part of a mask set co-developed by the two companies to be used as the process driver for the new 0.18µ technology.

See: http://www.xilinx.com/prs_rls/0_18process.htm

Xilinx Internet Reconfigurable Logic Technology Boosted By New AllianceCORE Products

November 23, 1998 – Xilinx today announced two new AllianceCORE products aimed at enhancing Internet data security and bandwidth. The new cores, available immediately from AllianceCORE partners Memec Design Services and CoreEl MicroSystems, include a Data Encryption Standard (DES) engine core and an HDLC controller core designed specifically to support the emerging Point-to-Point Protocol over SONET (POS) communications standard.

See: http://www.xilinx.com/prs_rls/coreconfig.htm

Xilinx Launches Third-Party Design Consulting Program

November 16, 1998 - Xilinx today announced the creation of the Xilinx Program for Engineering Resources from Third Parties (XPERTS). This is a partnership program with third-party design consulting companies that offer programmable logic design expertise to electronic equipment manufacturers. Xilinx launched the program with an initial 26 member companies worldwide.

See: http://www.xilinx.com/prs_rls/xpertsann.htm

Xilinx Unveils Internet Reconfigurable Logic

November 10, 1998 - Xilinx recently announced Java-based tools and technology that will enable FPGA devices to revolutionize the development and deployment of network appliances. The FPGA platform for this effort will be the new Virtex series, which Xilinx began shipping earlier this year.

See: http://www.xilinx.com/prs_rls/sx3.htm

Siemens Announces Design Services For Xilinx Internet Reconfigurable Logic

November 10, 1988 – Siemens today announced their partnership to deliver Internet Reconfigurable Logic (IRL) products and services to customers worldwide. Siemens IT-DL will provide design services to customers who want to design network based products employing the Virtex FPGAs beginning in the first quarter of 1999.

See: http://www.xilinx.com/prs_rls/siemens.htm

Java Applets Empower Internet Reconfigurable Logic

November 10, 1998 – Xilinx, continuing its partnership with Sun Microsystems Inc. in utilizing Java technology for programmable logic devices (PLDs), today announced the application of the Java API for Boundary Scan to on-line reconfigurable products.

See: http://www.xilinx.com/prs_rls/java2.htm

Xilinx Announces Computer-Based Training For Verilog

November 2, 1998–Xilinx introduced today a computer-based training course, called the Verilog CBT course, as a new educational method for the Verilog hardware description language

(HDL). This new computer-based training (CBT) class provides an interactive learning environment that will teach all the essentials for writing, synthesizing, and simulating Verilog HDL for Xilinx devices.

See: http://www.xilinx.com/prs_rls/cbtveri.htm.
Also see: http://www.xilinx.com/support/training/verilog_cbt.htm

Xilinx Breaks One Million-Gate Barrier With Delivery of New Virtex Series

October 26, 1998—Dramatically expanding on the traditional uses for programmable logic, Xilinx today announced the first delivery of the Virtex FPGA series as a new FPGA platform to truly address system-level design issues. Starting at $10, the family encompasses a robust feature set across a full density range from 50,000 to one million system gates. The million-gate Virtex FPGAs—an industry first at this density—and a 300,000 gate device are available now.

See: http://www.xilinx.com/prs_rls/vtxship.htm

Xilinx Second Fiscal Quarter Revenues Up 4.1%

October 20, 1998 – Xilinx today announced the results for its second fiscal quarter ended October 3, 1998. Revenues for the quarter were $156.4 million, up 4.1% from $150.3 million in the second fiscal quarter last year, and up 3.2% from $151.6 million in the prior fiscal quarter.

See: http://www.xilinx.com/prs_rls/99q2.htm

Xilinx And Cast Expand AllianceCORE Offerings With Seven New Cores

October 12, 1998–Xilinx announced today seven new AllianceCORE products available immediately from Xilinx AllianceCORE partner CAST, Inc. These products include serial communications and microprocessor peripheral cores. They are often used in PC and embedded applications such as medical instrumentation, industrial control, aerospace, defense, and communications systems.

See: http://www.xilinx.com/prs_rls/cast1012.htm

Xilinx Software Is Web-Enabled

October 12, 1998—Xilinx today announced the new release of its Foundation Series and Alliance Series software, version 1.5i, as the second phase of its new Silicon Xpresso initiative for internet-based design. This new software release has a direct connection to the new Xilinx support website, at support.xilinx.com, built into the graphical user interface.

See: http://www.xilinx.com/prs_rls/1_5i.htm

Xilinx Announces Industry's First Line of High-Density, Radiation-Hardened SRAM-based FPGAs

September 30, 1998 – Xilinx today announced the introduction of the XQR4000XL family of radiation-hardened, SRAM-based FPGA devices as part of its QPRO line of high-reliability products. The XQR4000XL devices utilizes a 0.35µ epitaxial CMOS process that provides latch-up immunity, high total ionizing dose tolerance, and low probability of single event upsets induced by natural radiation in satellite and other space environments.

See: http://www.xilinx.com/prs_rls/radhard.html

Xilinx Set To Penetrate New Markets With Biggest Product Launch in History of PLD Industry

September 28,1998 – In a sweeping move to address new markets and enhance its product portfolio, Xilinx today announced it will begin shipment of three new lines of advanced, 3.3V programmable logic devices (PLDs) and an expanded line of 2.5V, high-density, high-performance products.

See: http://www.xilinx.com/prs_rls/exelw_xl.htm

Xilinx Unveils Framework For Interactive Web-based PLD Design

September 9, 1998 – Xilinx today announced its Silicon Xpresso initiative, a broad company program to step up delivery of Internet and Java-based applications to customers who design with programmable logic devices (PLDs). The first element of the Silicon Xpresso initiative is a new Java application programming interface (API) for the boundary-scan market from Sun Microsystems for programming, testing, and debugging PLDs from any supplier. Xilinx has entered into an agreement to license Sun's Java Card technology, which will serve as the foundation for the API. The initiative also includes the Xilinx WebFitter tool, software that allows customers to do on-line design evaluation over the Internet on a remote server.

See: http://www.xilinx.com/prs_rls/javaapi.htm

Announcements

Information about new events, data sheets, application notes, products, and so on.

12/2/98

- The New Exemplar Expert Journal is now available.
- New application note: XAPP120 describes how Spartan Series FPGAs Compete for Gate Array Production.
- New application note: XAPP123 describes how to Use Three-State Enable Registers in XLA, XV, and SpartanXL FPGAs.

11/30/98

- Join Xilinx and Synplicity in Europe for the Value of Time, a Design Productivity Seminar. See: http://www.aspen.uk.com/synplicity/
- Synopsys Offers a Free 30-Day Evaluation of a Xilinx FPGA Model Family in their SmartModel Library. See: http://www.synopsys.com/fpgamodels/
- Insight Electronics Offers its Customers a New Starter Kit for Either a Xilinx CPLD or Xilinx DSP Solution. See: http://www.insight-electronics.com/linecard/xilinx/cpld_starter_kit/

11/23/98

- New datasheets add timing specifications for XC4000XLA and XC4000XV FPGAs and SpartanXL FPGAs (PDF file). See: http://www.xilinx.com/whatsnew.htm

11/17/98

- New Spartan Series application notes. XAPP098 describes low-cost, efficient serial configuration, while XAPP122 describes express configuration of SpartanXL FPGAs (PDF file). See: http://www.xilinx.com/whatsnew.htm

11/16/98

- New Virtex Series FPGA datasheet (PDF file). See: http://www.xilinx.com/whatsnew.htm

11/3/98

- XC9536 5V ISP CPLD Family datasheet revised to include new AC characteristics and Internal Timing Parameters (PDF file). See: http://www.xilinx.com/whatsnew.htm
- New Product Selector Guide now available. See: http://www.xilinx.com/psguide/index.htm

10/21/98

- New XC4000XLA and XC4000XV datasheet and updated specifications for XC4000XL Series (PDF file). See: http://www.xilinx.com/whatsnew.htm

10/20/98

- New application note, Xilinx Implementation Tools Release 1.5 features (PDF file). See: http://www.xilinx.com/whatsnew.htm
- The 1.5 Service Pack is available from support.xilinx.com under Updates & Files. See: http://support.xilinx.com/

10/13/98

- New XCell Journal #30 for the Fourth Quarter, 1998. Includes over two dozen new technical articles. See: http://www.xilinx.com/xcell/xcell30.htm

10/8/98

- New XC9500XL Datasheets for XC9536XL, XC9572XL, and XC95288XL. See: http://www.xilinx.com/partinfo/databook.htm#CPLD
- New A1.5/F1.5 Service Pack Update available from our Software Updates Page. See: http://www.xilinx.com/support/techsup/sw_updates/index.htm

10/2/98

- New XC9500XL Application Notes. See: http://www.xilinx.com/apps/epld.htm

9/28/98

- New XC9500XL Applications Brief: XC9500XL Versus MAX7000A (PDF file). See: http://www.xilinx.com/whatsnew.htm
- New XC9500XL Datasheets now available. See: http://www.xilinx.com/partinfo/databook.htm#CPLD

9/9/98

- Xilinx Introduces a new Java API for Boundary-Scan and WebFitter. See: http://www.xilinx.com/products/software/sx/sxpresso.html

9/3/98

- Datasheets for all cores shipped on the V1.5 CORE Generator CD now available. See: http://www.xilinx.com/support/techsup/journals/coregen/docs.htm
- New Synopsys / Xilinx High-Density Design Methodology application note (PDF file). See: http://www.xilinx.com/whatsnew.htm
- New Alliance 1.5 Watch Tutorial modules will familiarize you with the new and improved Xilinx design flows from design entry to verification and debugging. See: http://www.xilinx.com/support/techsup/tutorials/index.htm
- The CORE Generator Expert Journal has been updated with new information on the 1.5 release. See: http://www.xilinx.com/support/techsup/journals/coregen/index.htm

Corrections

In *XCell 30*, the article on page 18 "Guaranteeing Designs Work in All Conditions," the first paragraph under "Prorated Delays for Voltage and Temperature" should read:

Design operating conditions vary based on your application. Voltage and Temperature prorating allows you to test your design under real operating conditions. The delays vary for worst case based on the voltage and temperature and can be calculated for a voltage range of 3.0V to 3.6V and a temperature range of 0°C to 85°C. If no prorating constraints are provided, the worst case commercial temperature and voltage range values of 85° C and 3.0V are used. Also, the flow examples in the article suggest adding the prorating constraints to the Physical Constraints File (.pcf), and do not mention the User Constraints File (.ucf). For customers who know that they want to take advantage of the prorating constraints before they implement a design, the recommended flow is to enter the constraints into the User Constraints file (.ucf). The constraints will then be used by PAR during place and route, NGDANNO for simulation, and TRCE for static timing analysis.

1GHz Performance Milestone – 0.18µ, 1.8V FPGA

Xilinx recently achieved 1GHz performance in a prototype 0.18µ FPGA, using a frequency counter design. These FPGAs, derivatives of the XC4036XV family, were part of a mask set used as the process driver for our new 0.18µ technology.

by Mike Seither, Director of Public Relations, Xilinx, mseither@xilinx.com

The aggressive development of the 0.18µ process, currently one of the most advanced in the world, reflects the very close working relationship between Xilinx and UMC Group (Taiwan). The partnership is based on an on-going, joint effort to be at the leading edge of semiconductor industry process development. This unprecedented level of cooperation gives you early access to the most advanced technology for low-power, high-performance applications.

"Xilinx is an ideal development partner because it is dedicated to being an early adopter of advanced foundry technology. Moreover, the regularity of the SRAM-based Xilinx FPGA architecture facilitates defect analysis and fault testing, making FPGAs excellent vehicles to troubleshoot the most up-to-date UMC Group processes," said Jim Kupec, president of UMC Group (USA).

"Our strategy is to provide maximum value to our customers by aggressively pushing FPGA technology to the limits in order to provide more advanced features and performance," said Wim Roelandts, president and CEO of Xilinx. "Like Xilinx, UMC Group is clearly focused on being at the leading edge of process technology. This common objective has led to a highly successful partnership."

1GHz Frequency Counter Operation

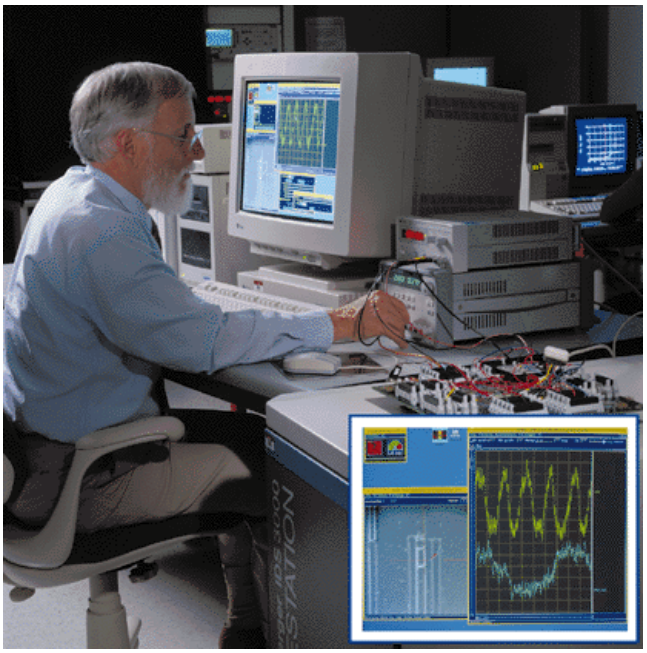


Figure 1

"Xilinx is an ideal development partner because it is dedicated to being an early adopter of advanced foundry technology. Moreover, the regularity of the SRAM-based Xilinx FPGA architecture facilitates defect analysis and fault testing, making FPGAs excellent vehicles to troubleshoot the most up-to-date UMC Group processes."

—Jim Kupec, president of UMC Group (USA).

Xilinx Trade Show Programs

See our preliminary trade show schedules for 1999 and plan to attend where possible.

by Darby Mason-Merchant, Trade Show Manager, Xilinx, darby@xilinx.com

In 1998, Xilinx participated in more than 30 industry conferences and trade shows world-wide including the 35th Design Automation Conference (DAC) in San Francisco, Electronica 98 in Munich, Germany, and EDA&T '98 in Hsinchu, Taiwan. Xilinx took a more focused approach to industry conferences and shows addressing such vertical markets as PCI, ATM / Networking, and Telecommunications while continuing to increase our strong presence in DSP and IP-related events.

In 1999, Xilinx will concentrate on even more vertical events that fit into our overall marketing mix while enhancing our presence in those industry conferences that can most effectively showcase our core competencies in silicon, software, cores, and support. Industry events like these are the best opportunities for you to get hands-on experience with the leading PLD technologies such as Virtex, Silicon Xpresso, Foundation Series and Alliance Series Software, and CORE Solutions.

1998 Trade Show Highlights

Electronica '98

Xilinx proudly presented the new Virtex family at the Electronica show in Munich. Electronica '98, the 19th International Fair for Components and Assemblies in Electronics, took place at the new exhibition center located at the old Munich airport during November 10–13. Approximately 130,000 visitors passed through the halls over the four days.

The Xilinx booth covered 110 square meters and presented an impressive display of the revolutionary million gate Virtex device as well as our latest software and core offerings. Supporting Xilinx at the booth were our partners Exemplar, Synplicity, Frontier, and MiroTech Microsystems. The Bavarian Minister of Economy, Dr. Wiesheu visited the Xilinx stand and also met with Wim Roelandts, Chief Executive Officer of Xilinx.

1999 Spotlight Event

IIC '99 - The International IC Conference & Exhibition '99

Show Dates: April 8-16, Guanzhou, Shanghai and Beijing, China

In partnership with our local distributor Insight, a division of Memec (Asia Pacific), Xilinx will be exhibiting and presenting a paper at all three IIC events. China is one of the fastest growing semiconductor markets, and local designers are looking for IC solutions covering the entire range of applications. A team of experienced technical and sales support staff will be on hand in the Xilinx booth to help designers choose the right PLD products to fit their needs.

1999 N.A. Trade Show Schedule

Feb.1-4	ICEPD99 (Portable Design 99)	San Diego, Calif.
Feb. 21-23	FPGA99 Conference	Monterey, Calif.
Mar.15-19	ICASSP99	Phoenix, Ariz.
April 26-28	DSP Spring Conference 99	Santa Clara, Calif.
April 99	FCCM Conference 99	Napa, Calif.
May 24-27	PC Developers Expo 99	Santa Clara, Calif.
June 9-11	WITI Technology Summit 99	Santa Clara, Calif.
June 21-25	36th Design Automation Conf.	New Orleans, La.
Nov 1-4	DSP World / ICSPAT Conf. 99	Orlando, Fla.

European Trade Show Schedule

April 13-16	Intertronic	Paris
April 14-15	Electronic Design Solutions	NEC Birmingham
May 11-12	PLD Days - UK	Sandown, UK
May 18	PLD Day - Sweden	Stockholm
May 26-27	Embedded Systems Show	Olympia, UK
June 2-4	Embtech World 99	Paris
Sept. 10-14	IBC 99	Amsterdam
Oct. 10-17	Telecom 99	Geneva
Nov. 1-2	IP Europe 99	Edinburgh
Nov. 16-18	Embedded Systems Conf. 99	Maastricht

Japanese Schedule

June 30	Japan FGPA/PLD Conf. 99	Tokyo

South East Asian Schedule

April 8-9	IIC '99	Guanzhou, China
April 12-13	IIC '99	Shanghai, China
April 15-16	IIC '99	Beijing, China
Oct. 21-22	EDA&T	Hsinchu,Taiwan
Oct. 25-26	EDA&T	Beijing, China
December	Multi-Tech	Taipei,Taiwan

For more information about Xilinx worldwide trade show programs, please contact one of the following Xilinx team members or see our Web page at: www.xilinx.com/company/seminars.htm

For North American shows, contact Darby Mason-Merchant at: darby@xilinx.com

For European shows, contact Andrea Fionda at: andrea.fionda@xilinx.com

For Japanese shows, contact Tetsuo Souyama at: tetsuo.souyama@xilinx.com

For Southeast Asian shows, contact Mary Leung at: mary.leung@xilinx.com

support.xilinx.com

For the Designing Engineer

Xilinx offers a wealth of on-line support services and information through our new ***support.xilinx.com*** *website. Here you will find easy access to:*

- **Answers Search** – This is a powerful troubleshooting tool, with over 3000 problem solving documents, fully indexed and accessible through our search engine.
- **Configuration Problem Solver** – This support tool provides a decision tree that asks you questions about your configuration problem and quickly guides you to the solution. Any answer is no more than six clicks away.
- **Application Notes** – This is an extensive library of technical briefs and reference designs, providing a variety of solutions.
- **Expert Journals** – These journals contain tips and techniques from programmable logic design experts, helping you become a Xilinx wizard.
- **Software** – This easy-to-use interface helps you keep your software up to date, so your system will always be running at peak performance levels.
- **BSDL, IBIS, Speed and Package Files** – These critical files are now organized in one easy to use location, making it easier than ever for you to find exactly what you need.
- **Free Cores and IP Library** – Here you will find our free cores that you can easily download for unrestricted use in your designs.
- **Education Resources** – Here you can sign up for any of our extensive list of courses which are available at many locations worldwide. We can create a course specifically for your needs and present it at your location as well.
- **support.xilinx.com Service Centers** – If you can't find the answer you need on-line, contact one of our Service Centers. Our applications staff is always available to help you quickly resolve issues that are not answered on our website.
- **Order Inquiry System** – Our on-line order status system allows you to easily track your orders.

How To Find Answers Quickly

1. **Go to our http://support.xilinx.com website.** All of our support resources are found here.
2. **Use our Answers Search tool.**
3. **Look through our Expert Journals.**
4. **Get the latest Software Updates.**
5. **Contact the Xilinx Hotline.** If the steps above did not answer your question, contact the support.xilinx.com Service Center near you.

support.xilinx.com Service Centers

Xilinx Service Centers give you direct access to Xilinx Application Engineers worldwide. If you didn't find what you need at support.xilinx.com, call or E-mail your technical questions to one these locations:

North America	Phone: 1-800-255-7778 or (408) 879-5199 Email: hotline@xilinx.com
United Kingdom	Phone: 44 1932 820821 Email: ukhelp@xilinx.com
France	Phone: 33 1 3463 0100 Email: frhelp@xilinx.com
Germany	Phone: 49-89-93088-130 Email: dlhelp@xilinx.com
Hong Kong	Phone: (85)2-2424-5200 Email: hongkong@xilinx.com
Korea	Phone: (82)2-761-4277 Email: korea@xilinx.com

	Usable Gates (K)	Logic Gates	System Gates (see note 1)	Logic Cells (see note 2)	Total CLBs	Total Flip-Flop	Max. I/O	Max. RAM Bits	Config. Memory (K Bits)	XC1700 Serial PROM Req.
XC4000EX Family - 5 Volt										.5µm Triple Layer Metal Process
XC/XQ4028EX	18-50	28K	50K	2,432	1,024	2,560	256	32.8K	668	XC1701
XC4036EX	22-65	36K	65K	3,078	1,296	3,168	288	41.5K	833	XC1701
XC4000XLA Family - 3.3 Volt										Advanced .35µm Five Layer Metal Process
XC/XQ4013XLA	10-30	13K	30K	1,368	576	1,536	192	18.4K	393	XC17512L
XC4020XLA	13-40	20K	40K	1,862	784	2,016	224	25.1K	522	XC17512L
XC4028XLA	18-50	28K	50K	2,432	1,024	2,560	256	32.8K	668	XC1701L
XC/XQ4036XLA	22-65	36K	65K	3,078	1,296	3,168	288	41.5K	833	XC1701L
XC4044XLA	27-80	44K	80K	3,800	1,600	3,840	320	51.2K	1,015	XC1701L
XC4052XLA	33-100	52K	100K	4,598	1,936	4,576	352	62.0K	1,215	XC1702L
XC/XQ4062XLA	40-130	62K	130K	5,472	2,304	5,376	384	73.8K	1,434	XC1702L
XC4085XLA	55-180	58K	180K	7,448	3,136	7,168	448	100.4K	1,925	XC1702L
XC4000XV Family - 2.5 Volt										.25µm Five Layer Metal Process
XC40110XV	75-200	110K	220K	9,728	4,096	8,704	448	131.1K	2,488	XC1704L
XC40150XV	100-300	150K	300K	12,312	5,184	11,520	448	165.9K	3,373	XC1704L
XC40200XV	130-400	200K	400K	16,758	7,056	15,456	448	225.8K	4,551	XC1704L & XC17512L
XC40250XV	160-500	250K	500K	20,102	8,464	18,400	448	270.9K	5,434	XC1704L & XC1702L
Spartan Family - 5 Volt and 3.3 Volt (XL)										Advanced .5µm & .35µm (XL) Process
XCS05/XL	2-5	3K	5K	238	100	360	80	3.2K	54	XC17S05 (XL)
XCS10/XL	3-10	5K	10K	466	196	616	112	6.3K	95	XC17S10 (XL)
XCS20/XL	7-20	10K	20K	950	400	1,120	160	12.8K	179	XC17S20 (XL)
XCS30/XL	10-30	13K	30K	1,368	576	1,536	192	18.4K	249	XC17S30 (XL)
XCS40/XL	13-40	20K	40K	1,862	784	2,016	224	25.1K	330	XC17S40 (XL)
Virtex Family - 2.5 Volt										
XCV50	20-50	20K	50K	1,728	384	1,536	180	57K	559	XC1701L
XCV100	30-100	30K	100K	2,700	600	2,400	180	79K	781	XC1701L
XCV150	45-150	45K	150K	3,888	864	3,456	260	104K	1,041	XC1701L
XCV200	60-200	60K	200K	5,292	1,176	4,704	284	133K	1,336	XC1702L
XCV300	80-300	80K	300K	6,912	1,536	6,144	316	164K	1,752	XC1702L
XCV400	130-400	130K	400K	10,800	2,400	9,600	404	236K	2,546	XC1704L
XCV600	185-600	185K	600K	15,552	3,456	13,824	500	319K	3,608	XC1704L
XCV800	250-800	250K	800K	21,168	4,704	18,816	512	416K	4,715	XC1704L & XC1701L
XCV1000	330-1M	330K	1M	27,648	6,144	24,576	612	524K	6,128	XC1704L & XC1702L

QPRO™ High-Reliability QML Products	Usable Gates (K)	Logic Gates (k)	System Gates (k) (see note 1)	Logic Cells (see note 2)	Total CLBs	Total Flip-Flop	Max. I/O	Max. RAM Bits (k)	Config. Memory (K Bits)	XC/XQ SPROM Req.
XQ4000EX Family - 5 Volt										.5µm Triple Layer Metal Process
XQ4005E	3-9	5	9	466	196	616	112	6.3	9.5	XC17256D
XQ4010E	7-20	10	20	950	400	1,120	160	12.8	178	XC17256D
XQ4013E	10-30	13	30	1,368	576	1,536	192	18.4	248	XC17256D
XQ4025E	15-45	25	45	2,432	1,024	2,560	256	32.7	422	XC17256Dx2
XQ4028EX	18-50	28	50	2,432	1,024	2,560	256	32,7	422	XC17256Dx2
XC4000XL Family - 3.3 Volt										Advanced .35µm Five Layer Metal Process
XQ4013XL	10-30	13	30	1,368	576	1,536	192	18.4	393	XQ1701L
XQ4036XL	22-65	36	65	3,078	1,296	3,168	288	41.5	833	XQ1701L
XQ4062XL	40-130	62	130	5,472	2,304	5,376	384	73.8	1,434	XQ1701Lx2 or XQ1704L
XQ4085XL	55-180	85	180	7,448	3,136	7,168	448	100.4	1,925	XQ1701Lx2 or XQ1704L
Virtex Family - 2.5 Volt										.25µm Five Layer Metal Process
XQV100	30-100	30	100	2,700	600	2,400	180	79	781	XQ1701L
XQV300	80-300	80	300	6,912	1,536	6,144	316	164	1,751	XQ1701Lx2 or XQ1704L
XQV600	185-600	185	600	15,552	3,456	13,824	500	319	3,608	XQ1704L
XQV1000	330-1M	330	1M	27,648	6,144	24,576	512	524	6,128	XC1704LX2

	Macrocells	Max. I/O	Pin-to-Pin Delay (ns)
XC9500 Family - 5 Volt CPLDs			
XC9536	36	34	5
XC9572	72	72	7.5
XC95108	108	108	7.5
XC95144	144	133	7.5
XC95216	216	166	10
XC95288	288	192	15
XC9500XL Family - 3.3 Volt CPLDs			
XC9536XL	36	36	4
XC9572XL	72	72	5
XC95144XL	144	117	5
XC95288XL	288	192	6

Speed Grade Options	Slowest	Fastest
XC4000EX	-4	-2
XC4000XLA	-09	-07
XC4000XV	-09	-07
Virtex	-4	-6
Spartan	-3	-4
SpartanXL	-4	-5
XQ4005X	-4	-4
XQ4000XL	-3	-3

Serial PROM Package Options	Density	PD8	SO8	VO8	PC20	SO20	VQ44	3 Volt	5 Volt
XC1736E	36K	Y	Y	Y	Y				Y
XC1765E (EL)	65K	Y	Y	Y	Y			Y (EL)	Y
XC17128E (EL)	128K	Y		Y	Y			Y (EL)	Y
XC17256E (EL)	256K	Y		Y	Y			Y (EL)	Y
XC17512L	512K	Y			Y	Y		Y	
XC1701	1M	Y			Y	Y			Y
XC/XQ1701L	1M	Y			Y	Y		Y	
XC1702L	2M						Y	Y	
XC1704L	4M						Y	Y	
XC17S05	54K	Y		Y					Y
XC17S05XL	55K	Y		Y				Y	
XC17S10	95K	Y		Y					Y
XC17S10XL	96K	Y		Y				Y	
XC17S20	178K	Y		Y					Y
XC17S20XL	179K	Y		Y				Y	
XC17S30	248K	Y		Y					Y
XC17S30XL	249K	Y		Y				Y	
XC17S40	329K	Y				Y			Y
XC17S40XL	331K	Y				Y		Y	

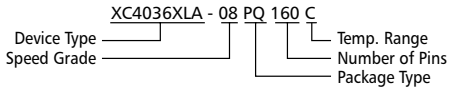
Serial PROM Package Options	Density	DD8	CC44	S020
XC17256D	256K	Y	-	-
XQ1701L	1M	-	Y	Y
XQ1704L	4M	-	Y	Y

Notes:

1. System Gates include 20-30% of CLBs used as RAM
2. A Logic Cell is defined as a 4 input LUT and a Register

Ordering Information for FPGAs / CPLDs

Example:



FPGA / CPLD Package Options and User I/O

	XC4000EX (5 Volt) FPGAs		XC4000XLA (3.3 Volt) FPGAs								XC4000XV (2.5 Volt) FPGAs				Spartan™ (5V, XL 3.3V) FPGAs					XC9500 (5 Volt) CPLDs						XC9500XL (3.3 Volt) CPLDs				Virtex™ (2.5 Volt)								
	XC4028EX	XC4036EX	XC4013XLA	XC4020XLA	XC4028XLA	XC4036XLA	XC4044XLA	XC4052XLA	XC4062XLA	XC4085XLA	XC40110XV	XC40150XV	XC40200XV	XC40250XV	XCS05/XL	XCS10/XL	XCS20/XL	XCS30/XL	XCS40/XL	XC9536	XC9572	XC95108	XC95144	XC95216	XC95288	XC9536XL	XC9572XL	XC95144XL	XC95288XL	XCV50	XCV100	XCV150	XCV200	XCV300	XCV400	XCV600	XCV800	XCV1000
IOBs	256	288	192	224	256	288	320	352	384	448	448	448	448	448	80	112	160	192	224	34	72	108	133	166	192	36	72	117	192	192	240	288	336	384	480	576	672	768
PLCC Packages																																						
44																				34	34					34	34											
84															61	61					69	69																
CSP Packages																																						
48																				34						36	38											
144																												117		94	94							
PQFP / HQFP Packages																																						
100																					72	81	81															
160			129	129	129	129	129	129	129	129												108	133	133														
208	160		160	160	160	160	160	160	160	160							160	169	169					166	168			168										
240	193	193	192	193	193	193	193	193	193	193	193	193						192	193											166	166	166	166	166	166	166	166	
304	256	256			256	256	256	256	256	256																												
VQFP Packages																																						
44																				34																		
64																										36	52											
100															77	77	77	77																				
TQFP / HTFP Packages																																						
100																					72	81	81				72	81										
144																112	113	113										117	117	98	98							
CBFP Packages																																						
228	192																																					
BGA Packages																																						
256			192	205	205													192	205											180	180	180	180					
352	256	288			256	288	289	289	289	289	289	289												166	192			192				260	260	260				
432		288				288	320	352	352	352	352	352	352	352																				316	316	316	316		
560								352	384	448	432	432	432	432																					404	404	404	404
PGA Packages																																						
299	256																																					
411		288																																				
559												448		448																								
FinePitch BGA																																						
256																														176	176	176	176					
456																																260	284	312				
600																																			404	404	404	
680																																				500	512	512

Available Now

Available Q199

XC4000XLA Available Sept. (XC4013, XC4062 (HQ2400)
Remainder available Q4

SpartanXL available Q4

XC9500XL Available Oct. (X9536XL, XC9572XL, XC95144XL)
XC95288XL Available Q199

Xilinx Alliance Series and Foundation Series Features

Features Included	Alliance Series		Foundation Series Design Environment			
	Schematic & Synthesis		Schematic & ABEL		Schematic & Synthesis	
	ALI-BAS	ALI-STD	FND-BAS	FND-STD	FND-BSX	FND-EXP
EDA Libraries and Interfaces for Cadence, Mentor, Synopys, and ViewLogic	✓	✓				
Turns Engine (Workstation Only)	✓	✓				
Synthesis Constraint Editor and Timing Analyzer						✓
Esperan MasterClass Lite VHDL Tutorial					✓	✓
HDL Synthesis Tools (ABEL, VHDL, and Verilog)					✓	✓
HDL Design Tools: HDL Wizard, Context Sensitive Editor, Graphical State Editor, and Language Assistant			✓	✓	✓	✓
Schematic Editor			✓	✓	✓	✓
Simulator (Functional and Timing)			✓	✓	✓	✓
HDL Synthesis Libraries (UniSim and Simprim)	✓	✓	✓	✓	✓	✓
Implementation Tools: Design Manager, Flow Engine, Timing Analyzer, Hardware Debugger, LogiBLOX, JTAGProgrammer, PROM File Formatter, Graphical Constraints Editor, Graphical Floorplanner	✓	✓	✓	✓	✓	✓
EDIF, VHDL (VITAL), and Verilog Back Annotation	✓	✓	✓	✓	✓	✓
LogiBLOX™ Module Generator	✓	✓	✓	✓	✓	✓
Xilinx CORE Generator	✓	✓	✓	✓	✓	✓
CPLD Devices (XC9500 and XC9500XL)	✓	✓	✓	✓	✓	✓
FPGA (Low Density/High Volume Devices): XC4000E/XL (Up to XC4010E/XL) Spartan and SpartanXL (All) XC3000A, XC3000L, XC3100A, XC3100L XC5200 (Up to XC5210)	✓		✓		✓	
FPGA (Unlimited Device Support): Virtex XC4000E/X (All) Spartan and SpartanXL (All) XC3x00A/L (All) XC5200 (All)		✓		✓		✓
Xchecker Cable (Workstation Only)	✓	✓				
JTAG Cable (PC Only)	✓	✓	✓	✓	✓	✓

MARKETING INNOVATIVE PRODUCTS VOL.3
A SERIES OF TECHNICAL REPORTS FROM:
INSIGHT ELECTRONICS

THE XILINX XC9500 CPLD DESIGN KIT

Our new CPLD starter kit is so complete, the only other design tool you'll need is your brain.

The Insight Xilinx CPLD Starter Kit only $99

Foundation base software V1.4

XC9500 demo board with sample

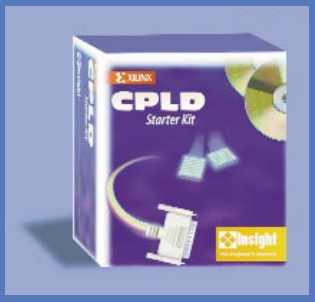
XC9500 information sheet

VHDL system upgrade voucher

XC4000XL information

AppLINX CD-Rom

MDS company information



- **High Performance**
  - 5 ns pin-to-pin timing
- **Competitively Priced**
  - Lowest cost per macrocell in the industry
  - Wide range of product package styles
- **Wide Density Range**
  - 36 to 288 macrocells
- **Customer Tested & Proven Pin Locking Architecture**
  - Flexible 36V18 function block
- **Designed for True 5 Volt In-System Programmability**
  - Industry standard JTAG interface
  - Endurance of 10,000 program/erase cycles
  - 20 years data retention
- **Programmable Outputs**
  - 24 mA drive on all pins
  - 3 volt or 5 volt I/O capability
  - Slew rate control on each output
- **Industry-Leading IEEE JTAG Boundary Scan**
  - Extended boundary scan functions
  - Design security
- **Easy Integration of ATE and Third-Party JTAG Programming and Test Tools**

SPECIAL FEATURES	High Speed (5ns tpd) Full Family (36 to 288 macrocells)	Flexible ISP w/pin locking JTAG	Low Price ($99.00)	XILINX®

FOR ENGINEERING USE ONLY:	WWW.INSIGHT-ELECTRONICS.COM	PROJECT RELEASE DATE
DESIGN KIT:CPLD Technology	Refer to Technical Specifications	A.S.A.P.
VENDOR:INSIGHT ELECTRONICS		

The new Insight CPLD STARTER KIT (DS-CPLD-KIT-PC1) is the solution to all your CPLD development needs in one low cost $99 package. The CPLD STARTER KIT offers logic designers a complete development system, tailored for developing the Xilinx XC9500 high performance CPLD product family. This development kit is supplied with the latest revision of Xilinx development software, plus a programming cable and XC9500 development board. In addition, it contains masses of useful technical and applications data to help get designs done in a *FastFLASH*. To order your CPLD kit today, visit our website at www.insight-electronics.com/CPLD/

or call us directly at 888 488-4133 extension 503.

The Engineer's Resource. Insight

PH 888 488-4133 x503 **www.insight-electronics.com/CPLD/**

Insight Electronics is a specialist within the Memec International Components Group.