XCELL

Issue 30
Fourth Quarter
1998

THE QUARTERLY JOURNAL FOR XILINX PROGRAMMABLE LOGIC USERS

XILINX®
The Programmable Logic Company℠

Inside This Issue:

PRODUCT INFORMATION

Three New 3.3V Families

XC4000X

Industry-leading system performance with double the capacity...

See page 3

SpartanXL

Now with faster systems speeds at the same low Spartan price...

See pages 4-5

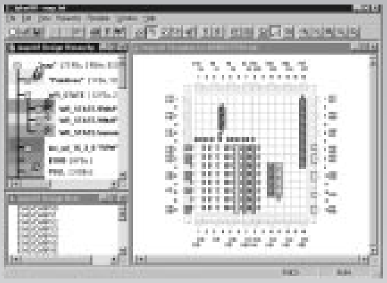
XC9500XL

New 3.3V CPLD family offers higher performance and new features...

See page 12

DEVELOPMENT SYSTEMS

Alliance Series 1.5 Software

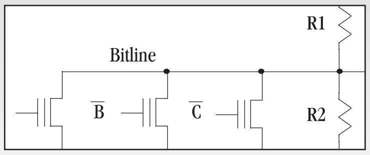


Three articles explore features of the new Alliance software release, from design stability to XC9500 design speed to design creativity...

See pages 18-20

DESIGN TIPS & HINTS

Reducing CPLD Power Consumption



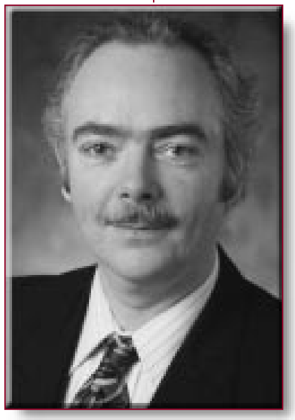
Minimizing CPLD power consumption is easy...

See pages 26-27

Xilinx – The Next Generation

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

Our mission is to help you explore new worlds, and new applications; to boldly go where no programmable logic device has gone before. And, our next generation of stellar devices are here now; no "science fiction."



We recently announced more than 20 new devices with densities ranging from 800 to 500,000 system gates, all fully supported by our highly acclaimed Alliance Series 1.5 and Foundation Series 1.5 software. A whole new world of possibility is now available, because these devices not only represent the cutting edge in performance and density, they also set new standards for low cost, high reliability, and ease of use.

This unprecedented offering brings you the broadest choice of 3.3V and 2.5V devices available anywhere, in four new families:

- **The XC9500XL family** consists of four 3.3V devices with logic densities ranging from 36 to 288 macrocells (800 to 6,400 gates). These devices are manufactured using advanced 0.35µ Flash technology for the industry's highest reliability in programming and data retention, as well as the lowest device cost and the smallest die size. These are the industry's highest performance CPLDs with pin-to-pin speeds of 4 nanoseconds and system clock frequencies of 200MHz, available in the most popular surface mount technology, including chip-scale packaging. All of our XC9500 products offer the industry's best pin locking and in-system programming capability as well as enhanced JTAG Boundary Scan support.

- **The XC4000XLA FPGA family** consists of eight 3.3V FPGAs ranging in density from 26,000 to 80,000 system gates. The XC4000XLA devices are manufactured with an advanced 0.25µ process that boosts performance by 30 percent over the current XC4000XL product line, at half the cost. The XC4000XLA family is the industry's lowest power, highest performance, full line of 3.3V FPGA products.

- **The XC4000XV family**, first unveiled last October, now consists of five 2.5V FPGAs, with densities from 220,000 to 500,000 system gates, including the newly announced XC40110XV. This second generation of 0.25µ devices offers the industry's highest performance, and includes the largest FPGA devices available today.

- **The SpartanXL family** consists of five 3.3V FPGAs. These new, very low cost devices follow the introduction earlier this year of the 5V Spartan line that features on-chip RAM and broad support for cores. The new SpartanXL products are available in densities ranging from 5,000 to 40,000 system gates.

All of these devices are supported by the Xilinx Foundation Series 1.5 and Alliance Series 1.5 software, which includes the new Xilinx AKA*speed*™ technology that delivers fast compile times and high clock speeds. These tools also support ASIC-like design features such as the reporting of minimum timing delays, prorated for both voltage and temperature. A wide variety of cores are also available, all managed by the Xilinx CORE Generator.

These new products are driving programmable logic into new applications that include digital cameras, digital television, set-top boxes, arcade games, PCMCIA modem cards, GPS driver information systems, and portable phones – applications that previously did not benefit from the many advantages of programmable logic.

And, even after all this, our million-gate Virtex family arrives next quarter, with system-level features that bring out-of-this-world possibilities – light years ahead of anything you've seen before. It's not science fiction, but it does come from the future.

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

XILINX®

New XC4000X Series FPGAs

Doubling Gate Capacities and Delivering Industry-Leading Speed

by Bruce Weyer,
Sr. Director, Marketing,
weyer@xilinx.com

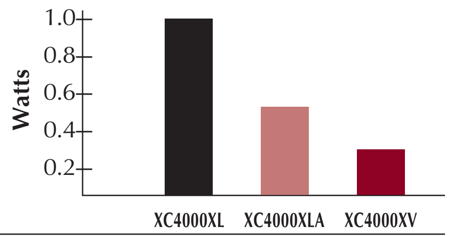
Optimized for 3.3V designs, the Xilinx XC4000X Series FPGAs double the capacity of FPGAs while delivering industry-leading system performance. Consisting of the new XC4000XLA and XC4000XV families, the XC4000X Series is an enhanced version of the industry standard XC4000 architecture. Consisting of 12 devices with capacities ranging from 30,000 to 500,000 system gates, these FPGAs feature the patented SelectRAM™ memory; offering a completely flexible logic distribution, as well as single-port or dual-port memory. Designed with advanced CMOS processes, the XC4000X Series delivers industry-leading performance while significantly reducing power consumption.

Unprecedented Performance

The XC4000X Series uses unique architectural enhancements and aggressive process technology to attain unprecedented speed at full capacity. Additional routing resources and highly buffered clock networks ensure that you get the highest array performance possible. New three-state I/O registers and FastCLK I/O buffers significantly increase system performance.

Double the Capacity

The XC4000XV Family, offering up to 500,000 system gates, is twice the capacity of competing products. Plus, it offers a high level of performance, with efficient clock buffering and abundant, fast, segmented routing that ensures minimal interconnect delay.



	XC4000XL	XC4000XLA	XC4000XV
Process tech:	0.35μ	0.35μ	0.25μ
I/O voltage:	3.3V	3.3V	3.3V
Core voltage:	3.3V	3.3V	2.5V

Figure 1: Relative Power Consumption

Power Consumption Less Than Half

The XC4000XLA consumes half the power of the equivalent XC4000XL device, and the XC4000XV only consumes a third of the power, as shown in **Figure 1**. These savings are derived from efficient design layout, smaller process geometries, and lower operating voltages.

Conclusion

The new XC4000X FPGA family represents the next generation of programmable logic technology, with the fastest, highest capacity devices available. Combined with our highly acclaimed Alliance Series 1.5 and Foundation Series 1.5 software, this family is the perfect choice for your next design.

Table 1: The XC4000X Family

	XC4013XLA	XC4020XLA	XC4028XLA	XC4036XLA	XC4044XLA	XC4052XLA	XC4062XLA	XC4085XLA	XC40110XV	XC40150XV	XC40200XV	XC40250XV
Logic Cells	1,368	1,862	2,432	3,078	3,800	4,598	5,472	7,448	10,982	12,312	16,758	20,102
System Gates	10-30K	13-40K	18-50K	22-65K	27-80K	33-100K	40-130K	55-180K	75-200K	100-300K	130-400K	180-500K
Max RAM Bits	18,432	25,088	32,768	41,472	51,200	61,952	73,728	100,352	131,072	165,888	225,792	270,848
User I/Os	192	224	256	288	320	352	384	448	448	448	448	448
Packages	PQ160 PQ208 PQ240 BG256	PQ160 PQ208 PQ240 BG256	HQ160 HQ208 HQ240 HQ304 BG256 BG352	HQ160 HQ208 HQ240 HQ304 BG352 BG432	HQ160 HQ208 HQ240 HQ304 BG352 BG432	HQ160 HQ208 HQ240 HQ304 BG352 BG432 BG560	HQ160 HQ208 HQ240 HQ304 BG352 BG432 BG560	HQ160 HQ208 HQ240 HQ304 BG352 BG432 BG560	HQ240 BG352 BG432 BG560	HQ240 BG352 BG432 BG560 PG559	BG432 BG560	BG432 BG560 PG559

The 3.3V SpartanXL™ FPGA Series

Invades New Territory with High Speed and Low Cost

by Marc Baker, Applications Engineer, marc.baker@xilinx.com

The number of different applications supported by the Spartan™ Series FPGAs has been dramatically increased with the shipment of the new 3.3V SpartanXL family. For the first time, an FPGA family can provide system speeds beyond 100MHz at prices below $3.00*, while supporting all the features needed for complete logic integration. The unique combination of blazing speed and low price eliminates the need for dedicated chips or gate arrays in high-volume applications such as digital imaging and PC peripherals.

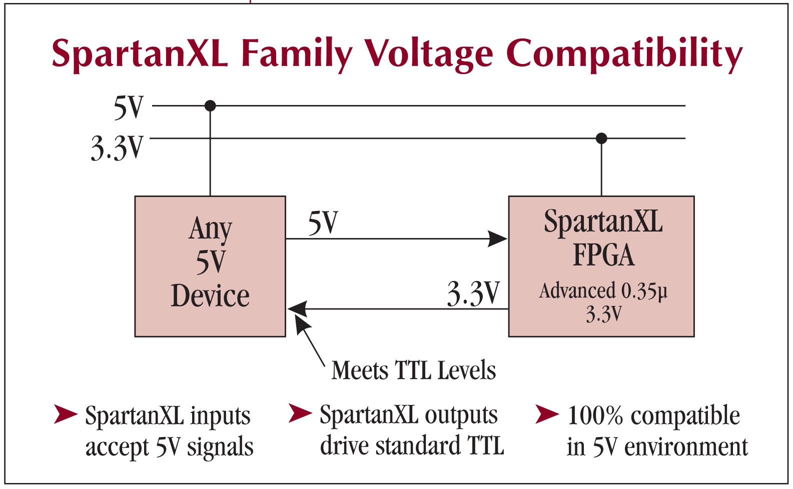
**84 PLCC, 100K units, -3 speed*

Using an advanced 0.35µ process, to achieve smaller die size and higher performance, the SpartanXL family builds on the success of the Spartan Series, the industry's fastest-growing FPGA family. This unique five-layer-metal process, developed by Xilinx, provides the most effective base for a 3.3V logic solution. The five devices in the SpartanXL family offer the same 5,000 to 40,000 gate density range as the 5V Spartan family, introduced in January 1998. Furthermore, they use the same low-cost packages, allowing easy migration between voltage levels.

The SpartanXL architecture features the same synchronous single-port and dual-port SelectRAM memory capabilities offered in the 5V family. The Spartan series is the only ASIC replacement FPGA that offers this key feature. On-chip RAM is useful for scratch-pad memories, shift registers, and FIFOs such as those used in a PCI interface.

The SpartanXL family (and the 5V Spartan family) are the industry's most cost-effective FPGAs, made possible by total cost management, which includes reduction of the assembly and test costs by using low-cost packaging and new, efficient test methodologies.

Figure 1: SpartanXL Family Voltage Compatibility


SpartanXL Family Voltage Compatibility
5V
3.3V
Any 5V Device
5V
SpartanXL FPGA
Advanced 0.35µ
3.3V
3.3V
Meets TTL Levels
➤ SpartanXL inputs accept 5V signals
➤ SpartanXL outputs drive standard TTL
➤ 100% compatible in 5V environment

New Features

The SpartanXL architecture adds several key new features to the original 5V Spartan family. The dedicated carry logic has improved performance, providing 16-bit addition in only 8 ns. Clock routing has been simplified with eight identical, global, low-skew buffers to choose from. The new Express Mode decreases configuration time by a factor of eight. Configuration through the dedicated IEEE-compatible Boundary Scan logic has been simplified, and is now supported by the JTAGProgrammer software and cable. Synthesis is simplified as well, by offering level-sensitive latches throughout the device.

The SpartanXL architecture features the industry's most versatile I/O cell, with features that include:

- 5V input tolerance even before power is applied
- 3.3V or 5V PCI compatibility
- Programmable 12 mA or 24 mA output drive
- Input fast capture latch for shorter setup times
- Output look-up table for the fastest pin-to-pin speeds

High Speed at Low Power

The SpartanXL series provides system-level speed beyond 100MHz. I/O toggle frequency achieves 100MHz, while functions such as standard 16-bit binary counters run at 120MHz internally. The SpartanXL speed grades were dramatically improved since the initial software release, and the new speed files are available on WebLINX (www.xilinx.com) in the File Download area.

The 3.3V supply reduces power consumption significantly. Xilinx FPGAs benefit from efficient segmented routing that minimizes the amount of power consumed by each net. For the latest power and speed information, see the SpartanXL Series datasheet on WebLINX.

Software Support and Core Solutions

The SpartanXL family is fully supported by the Xilinx Alliance Series 1.5 and Foundation Series 1.5 development software. New support includes libraries specifically for the Spartan and SpartanXL families, simplifying design with these products. Dozens of Xilinx Alliance partners provide design entry and verification tools.

Pre-defined system functions are available as core solutions for the SpartanXL family. Xilinx offers PCI and DSP LogiCORE solutions via the CORE Generator software, included in the 1.5 version of the Xilinx development system. Several third-party vendors provide AllianceCORE solutions, which are pre-verified for the SpartanXL family. Implementing these common functions in a SpartanXL FPGA costs less than an ASIC, due to the dramatically lower prices offered by the SpartanXL family.

Xilinx Spartan Series

5 Volt (0.5/0.35μ)	XCS05	XCS10	XCS20	XCS30	XCS40
3 Volt (0.35/0.25μ)	XCS05XL	XCS10XL	XCS20XL	XCS30XL	XCS40XL
System Gates	2K-5K	3K-10K	7K-20K	10K-30K	13K-40K
Logic Cells	238	466	950	1368	1862
Max Logic Gates	3,000	5,000	10,000	13,000	20,000
Flip-Flops	360	616	1120	1536	2016
Max RAM bits	3,200	6,272	12,800	18,432	25,088
Max I/O	77	112	160	192	205
Performance	>80MHz	>80MHz	>80MHz	>80MHz	>80MHz

**No Compromises:
Performance, RAM, Cores, and Low Price**

Figure 2: Availability Chart

Conclusion

The SpartanXL series complements the XC4000XLA family, which applies the same process technology to our higher density devices. For applications where less logic is needed, the XC9500XL family provides the fastest CPLDs in the industry. Together, these families provide the broadest choice of 3.3V devices. Rapid application of aggressive new process technologies allows these PLDs to penetrate new applications that were once the stronghold of ASICs, such as arcade games, graphics cards, and automotive cabin controls. Σ

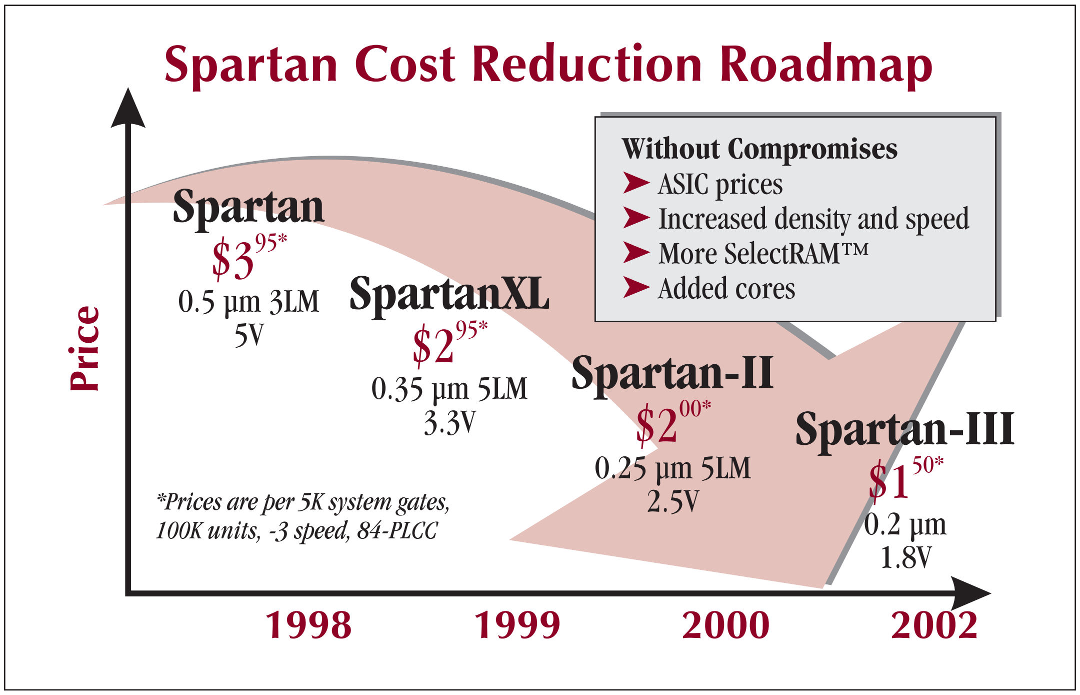


Figure 3: Spartan Cost Reduction Roadmap

FPGAs Can Be an Effective Alternative to Mask Gate Arrays

by Steve Sharp,
sharp@xilinx.com

In this fast-paced electronics industry, gate array engineers are under increasing pressure to produce new ASIC designs ever more quickly. As traditional masked gate arrays decline in usage, a new generation of programmable devices have become a viable alternative for the gate array user.

The new "ASIC Replacement" FPGAs have continued to narrow the price gap with mask ASICs while maintaining the user advantages of quick production and in-system reprogrammability. These FPGAs have begun to replace mask gate arrays in traditional ASIC volume applications, from networking encryption engines to PC adapters to digital camcorders.

In this article we examine recent advances in programmable technology and the advantages of the new ASIC Replacement FPGA.

For a list of Spartan and SpartanXL devices, see the *Xilinx Spartan Series* Table on page 5 in the "3.3V SpartanXL FPGA Series" article.

ASIC Replacement FPGAs

The Xilinx Spartan Series FPGAs were created to provide a cost effective and flexible replacement for low-end (<40K system gates) ASICs in volume production. These new FPGAs offer the ASIC designer the advantages of in-system reprogrammability at prices that are competitive with masked gate arrays. To become an effective ASIC solution, the Spartan Series had to substantially reduce die-size over the previous generation FPGAs while measurably improving gate-area density and system performance.

Although FPGAs have historically lagged the ASIC industry by one or two fab-process generations, current Spartan Series FPGAs were able to surpass most of today's gate arrays by employing a leading-edge, multi-feature size 0.35µ/0.25µ technology. In the past, a larger die size was necessary to provide sufficient logic density for typical FPGA designs but caused FPGAs to be priced out of range for higher volume production.

Because the high-end Xilinx FPGAs (such as the Virtex Series) contain more transistors (75 million) than the Pentium II microprocessor, Xilinx wafer foundry partners have chosen these complex devices to debug new fab processes, replacing DRAMs as the technology driver. Becoming a fab process driver means that Xilinx FPGAs will remain on the leading edge of process technology for years to come.

FPGAs Close the Price Gap

The SpartanXL family (3V version) is currently the lowest cost FPGA family in the industry. The entire Spartan/SpartanXL Series incorporates ASIC-like features such as dual-port synchronous on-chip memory and supports frequencies up to 80+MHz.

The key to the Spartan family's low production pricing is an "I/O pad-limited" die size. "I/O pad-limited" means that a die is reduced to the limits imposed by the I/O bonding pads. Pad-limited enables the Spartan Series FPGA die to be cost equivalent to most mask gate arrays of up to 205 I/O pins. With Spartan or SpartanXL prices starting at $2.95 (84 PLCC, 100K units, -3 speed), the series is able to realistically compete with mask gate arrays for production based upon the same I/O count.

For example, the 160 I/O SpartanXL S20XL, shown in **Figure 1**, has a comparable die size and cost as the 160 I/O 0.35µm gate array, even though the gate array contains a denser architecture (and higher gate count). By sizing the die at the pad-limits, the FPGA cost can be equivalent to most gate arrays.

Lower Manufacturing Cost

Because Spartan Series FPGAs were designed for low power consumption, inexpensive plastic packages can be used to help keep manufacturing costs low. Other production savings accrue from a

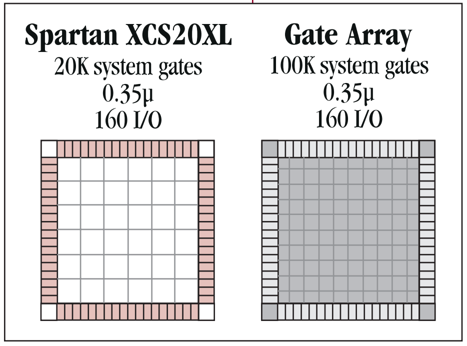


Figure 1. Spartan FPGAs match die size with mask gate arrays.

streamlined test methodology, built-in self-test features, and shorter test times. The combination of lower manufacturing overhead and small die-size eliminate cost barriers and enables Spartan Series FPGAs to be effective for both prototyping and for mass production.

FPGA Production Parts "Off-the-shelf!"

It is paramount to attain the best development time-to-production because today's product life cycles are often brief (9-18 months). Programmable logic is uniquely able to support both rapid prototyping and a quick ramp to full manufacturing. For many Xilinx customers the immediate availability of production is the most important benefit of programmable logic. After development, early production shipments are critical to market acceptance.

When FPGAs are used in production, marketing channels are quickly stocked for initial sales and new product revenue flow begins. Because the standard ASIC has an 8-16 week production leadtime, a 3-4 month sales delay would substantially decrease revenues and profits throughout the life of the product (see **Figure 2**). By using FPGAs in production; however, your market penetration is immediate.

The well-known McKinsey study found that a six-month delay costs one third of the profits over the lifetime of the product.

FPGA-to-ASIC Conversions

FPGA-to-ASIC conversions have been a popular approach to reduce unit production costs. However, it has become more difficult to cost justify these conversions in lower-density designs because of the new low price FPGAs along with the flexibility that FPGAs offer in today's quickly changing markets.

Conversion to a lower density ASIC means losing FPGA advantages of "off-the-shelf" production deliveries and simple field software updates, while incurring re-design risks (see **Figure 3**). Conversion costs such as NRE, silicon re-spins, test vectors, new device characterization, and internal engineering costs, usually outweigh the nominal unit cost difference between FPGAs and ASICs.

In higher density designs, using an ASIC that is crafted to exactly mirror the FPGA features can minimize the costs and the associated re-design risks. For example, the Xilinx HardWire™ family offers exact ASIC replacements for our higher-density FPGAs, making conversion very quick, easy, and inexpensive.

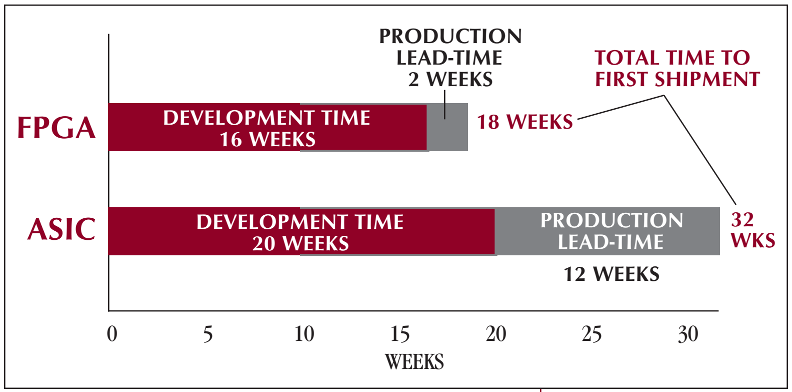


Figure 2: FPGA beats ASIC by more than 3 months to first customer shipment!

However, for most other manufacturers' FPGAs, the conversion time-to-production is frequently under-estimated. The time from start to full production typically exceeds four months. The common milestones are:

- Conversion/internal engineering time — three weeks
- Prototype fab time — three weeks
- Full production deliveries — 10 weeks

In total, 16 weeks are spent before production is fully ramped and the transition to the ASIC is complete. When a short product life or a mid-life product enhancement is likely, conversions become worthless.

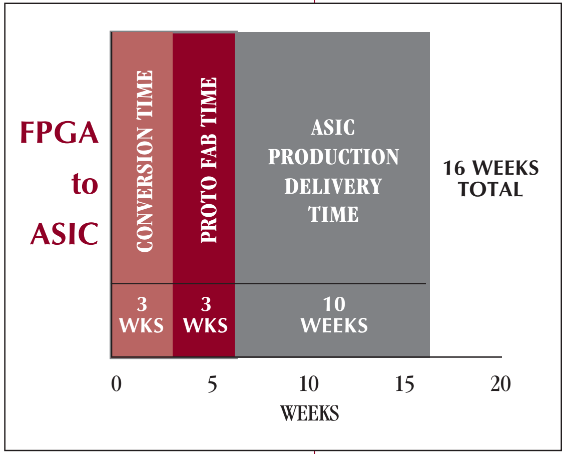


Figure 3: FPGA to ASIC conversion to production takes at least four months.

Conclusion

There are compelling advantages to use programmable logic for both development and production. Today's FPGAs support standard Verilog and VHDL design flows that help ASIC designers transition to programmable logic. Advanced process technology has leveled the playing field, and allowed FPGAs to be very price competitive with low-density gate arrays. When ASIC users now consider pricing, time-to-production, and reprogrammability, the preferred ASIC technology becomes the new Spartan Series FPGAs.

by Shelly Davis,
HardWire Marketing
Manager,
sdavis@xilinx.com

The Rapidly Changing ASIC Conversion Market

As programmable logic devices continue to grow in density, designers are increasingly using FPGAs where they previously used ASICs. The advantages of off-the-shelf availability and rapid prototyping make FPGAs a very attractive solution. However, you must answer a key question: will you use FPGAs for both development *and* production volumes, or will you convert the design to some form of ASIC, such as a gate array or standard cell, for cost reduction?

Third-Party ASIC Conversion Problems

The FPGA-to-ASIC conversion market has been dynamic over the past few years. Several companies have entered the market, only to find themselves in financial trouble. Microchip Technologies exited after 15 months of business and the D.I.I. group who purchased Orbit Semiconductor took a $60M loss last quarter due to the difficulties they continue to experience. It is not due to a lack of conversion business in the marketplace that problems are caused for the small ASIC vendor. The problem is caused instead by the difficulty of accurately converting today's complex PLDs. There are several factors contributing to this.

"Will you use FPGAs for both development and *production volumes, or will you convert the design to some form of ASIC, such as a gate array or standard cell, for cost reduction?"*

Most third party FPGA-to-ASIC conversion companies use gate array technology for the translation. The features of today's FPGAs, such as PCI compliance and the ability to implement 50K bits of RAM or more, exceeds the capability of most gate array vendors. In addition, the growing requirement for fast, on-chip RAM is perfectly suited to SRAM-based FPGAs, or fully diffused standard cell embedded RAM, but not for gate array processes. Even the most efficient gate array process will require 5 to 6 gates per RAM bit to convert FPGA RAM to ASIC RAM. For a design with 15K bits of RAM, this can translate into a minimum of 90K gates on a gate array. Therefore, a design that was slated for cost reduction from an FPGA to a smaller gate array may achieve only a small cost reduction because of the increase in die area required for the RAM.

Gate array price erosion has been fierce in the past few years. While this price reduction has benefited companies using gate arrays, some of the smaller gate array vendors are in poor financial condition, making it difficult for those companies to sustain innovation. This lack of new product development is now causing them to have difficulty converting many of the more complex FPGA designs.

Leaving an FPGA conversion to a third party gate array company is complicated, not well suited technologically, and doesn't offer much cost reduction because a 100K-gate FPGA often becomes a 500K gate array under these circumstances.

Pad Limitation

True pad limitation is achieved when there is such an abundance of gates available in a device, that the size of the die is determined solely by the number of required pads. The standard cell providers, with their dense core offerings, have been pad limited for some time. At process geometries below 0.5µ, many architectures, including FPGAs and gate arrays, become pad limited. For an FPGA-to-ASIC conversion company, who depends on achieving cost reduction through a die area shrink, pad limitation reduces the cost benefit of the gate array. In many cases, because the customer needs all the pads provided on the FPGA, the gate array device will be of equal size, in order to include the same number of pads. Size reduction due to translating programmable SRAM gates to much smaller metal vias is nullified.

Diverging Architectures

While the features and performance of FPGAs continually increase to include many ASIC-like features, the actual implementation and design methodology are becoming dissimilar; architecturally, FPGA technology and ASIC technology are diverging. The ability of one architecture to be "converted" to the other will require more than just re-targeting to a specific ASIC vendor's libraries. Gate array processes without embedded RAM structures that are specific to the original FPGA will quickly exceed gate count capability. In addition, in-depth knowledge of the FPGA's functionality and detailed specifications of industry standards like PCI will be basic requirements. Furthermore, the ability to provide accurate timing of I/Os and critical system performance will be essential to convert these newer, more complex designs.

FPGA designers who depend on ASIC cost reductions will find their options changing over the next 18 months. Many smaller ASIC vendors will de-emphasize FPGA conversions because they lack the capability to convert them in a cost-efficient and technically effective manner. In the meantime, FPGA price per gate continues to decline to a point that, for 40K system gates and below, FPGAs can be considered for production volume in "formerly ASIC" applications. Companies that continue to provide FPGA-to-ASIC conversions will need to offer increasingly FPGA-specific solutions, because a generic gate array process will not serve the requirements of all PLD features.

Xilinx is one example of a company that provides a specialized solution for FPGA conversions. The Xilinx HardWire Business Unit continues to develop new ASIC technologies suitable for converting complex, RAM-intensive FPGAs. Xilinx recently introduced the XH3 FpgASIC architecture which provides dense gate array logic surrounded by an I/O ring that replicates the Xilinx FPGA I/O. FPGA features are built into the base arrays, further reducing the risk of conversion problems.

Xilinx HardWire devices are an excellent cost reduction path for FPGA's above 40K system gates and are especially suited for most dense FPGAs.

Another company specializing is Clear Logic Corporation, who offers a solution for Altera FPGAs only. Clear Logic offers their proprietary ClearFire ™ technique for laser cutting metal fuses in base arrays which closely resemble the logic resources in the Altera Flex8000 family of FPGAs. The advantage to the customer is that by optimizing processes, libraries, and feature sets to convert Altera PLDs exclusively, the risk of converting the design incorrectly is reduced. In the future, this type of focus will be required to provide accurate FPGA cost reductions.

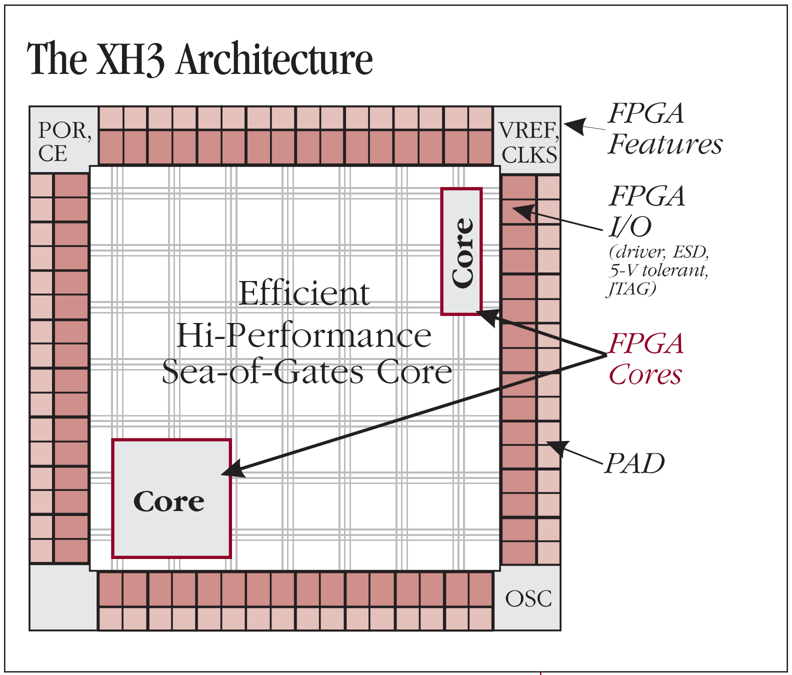


Figure 1: XH3 Architecture

Conclusion

In the future, FPGA technology will be increasingly suited for applications previously considered as gate array or standard cell territory. Many logic designers are realizing that they can take advantage of FPGA time-to-market benefits and still achieve a gate array cost point for volume production.

However, options for translating the FPGA to an ASIC are changing. Because of the complexity of the FPGA features and the density of RAM, many smaller gate array conversion vendors are dropping out of the market. Pad limitation for both FPGA's and gate arrays can minimize cost reduction benefits unless creative pad options can be implemented. Architecturally, FPGA's and gate arrays are also diverging. Differing design methodology and RAM implementations can be very inefficient if not specifically accounted for in the conversion process. The new models for success in the conversion market will be companies who specialize in converting a single architecture, such as Clear Logic with Altera devices, Lucent with MACO™ and Xilinx with HardWire FpgASIC's. These will be the options that provide the closest match and the most expertise for the 200K+ gate FPGA's of today.

Many ASICs Can Easily be REPLACED with FPGAs

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

One of the first things you usually notice about a printed circuit board design is the use of the "big chips." It is not hard to find the microcontroller, the memories, the FPGAs, and the ASICs. And, sometimes the unused capacity in one FPGA is enough to replace several ASICs. If you want to reduce your overall development/ manufacturing/test costs it often makes sense to incorporate ASIC functions into your FPGAs.

Telecommunications

I'll discuss three different common tasks performed in telecommunications: framing, multiplexing, and performance monitoring.

In a typical digital communications application, there may be a T1 or E1 framer ASIC. These devices usually cost about $20 in quantity, and they may be under-utilized by the application. Sometimes, only the transmit section, or just the receive section are used, or sometimes the pattern being generated is fixed, and the device is not being used to its fullest. The description of these functions is available from a number of sources such as ITU G-series documents (G.703 for example), or from ANSI, ATIS T1 documents.

"If you want to reduce your overall development/ manufacturing/test costs it often makes sense to incorporate ASIC functions into your FPGAs."

Digital multiplexers/demultiplexers (MULDEM's) are also common ASICs. They take a number of T1 or E1 signals, multiplex and demultiplex them, to and from a higher rate signal. Again, you can refer to the ITU or ANSI standards to get all the information you need to implement the function in an FPGA. Quite frequently, if the application also involves a fiber optic channel, or a radio link, other logic is required, which can easily be implemented on the same FPGA.

Performance monitoring consists of accumulating bit errors, coding violations, out of frame conditions and CRC errors. It must also keep track of these in light of error statistics. In these applications, monitoring bit streams at rates from 1 to 50 megabits per second is not a task that is easily performed by a microcontroller. To detect and keep track of the events, and then present them in a digested fashion to the microcontroller, allows for more features and higher performance. These circuits are all simple counters, shift registers and multiplexers, easily implemented in an FPGA.

Signal Processing

Digital finite impulse response filters (FIRs) are common elements of any communications system as well as control systems. Depending on the speed, resolution and number of taps required, an FPGA may be a good choice.

One such application is the root Nyquist transmit filter. In any channel, to provide for zero inter-symbol interference (ISI), you need to filter the transmit symbols. Many designs take the optimal ISI-free filter for a channel and split it into two parts: half at the transmit end, and half at the receive end. This also minimizes transmit bandwidth in the channel. Taking half of a filter is the same as the square root of the response, hence the name root Nyquist filter.

"By designing ASIC functions into an FPGA, you usually save money, power, and board space. Once designed, the function becomes part of your company's intellectual property, and can be re-used."

A typical FIR structure for the transmit filter is a shift register which is clocked at three times the symbol rate (or more), where the outputs or taps of the flip-flops in the register pass through resistors to a summing junction. The resistor values are chosen to set the gains in the taps of the filter. The sign of the value to be summed is chosen by selecting the normal, or the inverting output of the register. In one example, a 22-tap FIR is easily implemented in the I/O blocks of the FPGA along one side of the device. The output of the summing junction need only pass through a simple low pass filter to remove the sampling clock and the harmonics. Such filters are commonly used in all digital radio systems. Each bit of a modulation format's symbol requires such a filter, so for QAM, two such filters are required, and for 16QAM, four such filters are required.

Resistor/register FIR structures are useful to symbol rates up to a few mega symbols per second, and are easily implemented in an FPGA.

General Purpose Applications

Another good example of the "no more ASICs" design philosophy is in forward error correcting. Most communication channels have errors that occur (fiber, radio, magnetic, or metallic based channels) and need some amount of error correction to make the channel useful. Rather than buying ASICs to do the job, again it makes sense to perform the functions in an FPGA. Some forward error correction algorithms require a large memory block, but using an external RAM device is still less expensive than the ASIC alternative.

Some error correcting schemes are fairly easy to implement, and require only feedback shift register structures, such as Reed-Solomon codes. The simple schemes have a high overhead; they do not correct many errors per block or byte in relation to the extra bits required. These codes typically require 50% or more bits as check bits.

More powerful error correcting codes are popular where bandwidth or more bits becomes a liability. These are the Bose-Chaudhuri-Hocqueghem (BCH) codes. These codes can correct both random and burst errors, and can recognize when they cannot correct the errors. The more powerful the error correction scheme, the more memory is required. Some BCH codes, 511 bits for a 493-bit block for example, will correct one, two, or three bit errors in any block. The efficiency of this code is that it only adds less than 4% more bits to the channel.

Frequency synthesis is another area where the often expensive and single-sourced parts may be easily replaced by an FPGA. Fractional synthesizers, pulse swallowers, direct digital frequency synthesizers, as well as the phase detectors for phase locked loops, are all easily implemented in FPGAs.

Conclusion

By designing ASIC functions into an FPGA, you usually save money, power, and board space. Once designed, the function becomes part of your company's intellectual property, and can be re-used. For example, if an application needs to change from being a T1 to and E1 design, often only the FPGA program needs to change, and the board remains the same. In fact, some designs initialize as T1 or E1 depending on configurations stored in memory, once the application is selected.

If at some point the standards change, or the competition adds some highly desirable feature (or you want to add a new feature), the FPGA also gives you a future - it can be easily changed. If an ASIC is part of the design, you will end up with a board re-layout, and probably have to add an FPGA to "band-aid" the design. Why not get the design right the first time?

by David Chiang,
Manager, CPLD
Technical Marketing,
david.chiang@xilinx.com

XC9500XL 3.3V FastFLASH CPLDs

Even More Speed and Features At Lower Costs

With 10 million 5V XC9500 devices shipped, the world's leading FLASH CPLD family continues to break all records as the fastest growing CPLD family in the industry. Building on that success, we are introducing the new 3.3V XC9500XL. Best of all, the XC9500XL family is already supported by your Alliance Series 1.5 and Foundation Series 1.5 development systems.

"The XC9500 architecture is already widely recognized as among the most advanced in the world."

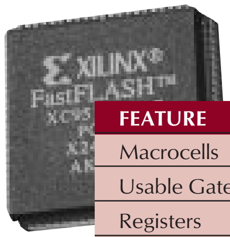
The XC9500 architecture is already widely recognized as among the most advanced in the world. The new 3.3V XC9500XL improves on this success with all-new features:

- High-speed FastCONNECT II switch matrix for up to 200MHz system performance
- New ultra-wide block fan-in of 54 for extra-wide functions
- Three global clocks with local clock inversion
- Global and individual output enables (OEs) with local OE inversion
- Dedicated clock-enable signal in each register
- Input hysteresis and bus-hold for all user I/O pins
- Inputs compatible with 5V, 3.3V, and 2.5V signals
- Leading-edge 0.35µ feature-size FastFLASH technology

Of course, the XC9500XL architecture also supports the leading-edge features available in the XC9500 family:

- Superior pin-locking characteristics
- Up to 90 product-term functions per macrocell
- Built-in D-type or T-type flip-flop option
- 18-macrocell function blocks for efficient 16-bit look-ahead logic implementations
- Dedicated JTAG/ISP pins for immunity from "ISP Lock-Out"
- Highest programming reliability:10,000 program/erase cycles and 20-year data retention

The XC9500XL devices were developed to operate with leading-edge FPGAs in today's advanced communications and computing systems using a 3.3V power supply. You can optimally partition fast state machines and control functions into XC9500XL devices and partition complex subsystem functions (including cores) into Spartan-XL or XC4000X FPGAs, all using a unified software environment.



FEATURE	XC9536XL	XC9572XL	XC95144XL	XC95288XL
Macrocells	36	72	144	288
Usable Gates	800	1,600	3,200	6,400
Registers	36	72	144	288
Fastest tPD (ns)	4	5	5	6
Fastest fSYS (MHz)	200	178	178	151
Package Options	PC44 (34)	PC44 (34)		
(# user I/O pins)	VQ64 (36)	VQ64 (52)	TQ100 (81)	
		TQ100 (72)	TQ144 (117)	TQ144 (117)
				PQ208 (168)
				BG352 (192)
	CS48 (36)	CS48 (38)	CS144 (117)	

Table 1: XC9500XL CPLD Family

Conclusion

Now you have the best 3.3V CPLDs available, fully supported by the Xilinx Alliance Series 1.5 and Foundation Series 1.5 software. The XC9500XL family offers the best in speed, flexibility and reliability, along with the high-quality support you expect from Xilinx.

New Chip Scale Packaging for Small, Lightweight Designs

by Frank Toth, Marketing Manager for FastFLASH Products, frank.toth@xilinx.com

Xilinx has just introduced two new chip scale packages for the new XC9500XL family: A 48-pin version for the XC9536XL and a 144-pin version for the XC95144XL. The 12x12 millimeter package for the XC95144XL features 117 I/Os and has a seven times smaller footprint than the 160-pin plastic quad flat pack device (see **Figure 1**). In addition, the 48-pin CSP package for the 3.3V XC9536XL gives you access to 36 I/Os while the XC9572XL has 38 I/Os available.

According to Electronic Trend Publications in San Jose, the use of CSP packages is expected to grow by 108% per year over the next few years, reaching more than 6.1 billion units shipped worldwide by 2002. This dramatic growth is occurring because CSP packages bring you the benefits of:

- An extremely small form factor for such applications as PCMCIA cards, portable and wireless designs, and PC add-in cards.
- Lower inductance and lower capacitance
- The absence of the thin, fragile leads found on other packages
- A very thin, very light weight package
- You can take advantage of existing circuit board lithography and assembly equipment. (Board-level assemblers, like Solectron, have already qualified CSPs.)

Xilinx also has a simple board layout solution (see **Figure 2**) for the 48-lead package that uses widely available 5-mil board traces without resorting to more expensive fine line printed circuit board technologies that require features like buried and micro vias. This gives you all the advantages of the smaller CSP footprint without the added expense of fine-line lithography boards.

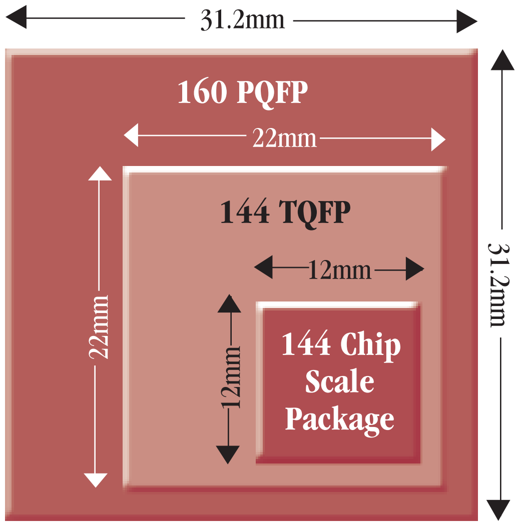
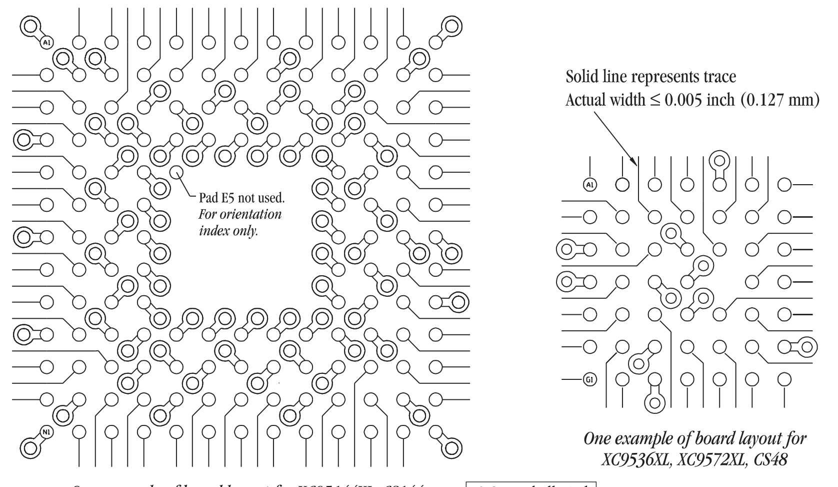


Figure 1: Package Size Comparison

Conclusion

Xilinx continues to be an innovator in leading-edge, easy-to-use package technology. These two new chip scale packages are another milestone in our continuing efforts to make programmable logic even more flexible and accessible to you. Σ

Figure 2: Two Sample Board Layouts


Pad E5 not used. For orientation index only.
A1
N1
Solid line represents trace
Actual width ≤ 0.005 inch (0.127 mm)
A1
G1
One example of board layout for XC95144XL, CS144
0.8 mm ball pitch
One example of board layout for XC9536XL, XC9572XL, CS48

Notes: Solder land diameter 0.013 inch (0.33 mm) NSMD (non-solder mask defined), via land diameter 0.020 inch (0.5 mm) and via hole diameter 0.012 inch (0.3 mm) are recommended.

by Dave Chiang, Manager, CPLD Technical Marketing, david.chiang@xilinx.com

Choosing A 3.3V CPLD? "ARM" Yourself…

Leading digital system manufacturers are rapidly adopting 3.3V components for higher performance, lower costs, lower power, and higher system reliability. With many new 3.3V CPLD families being introduced, the choices can seem confusing and overwhelming. To simplify your decision, "ARM" yourself with the three most important criteria when choosing a new 3.3V CPLD family: **Architecture, Reliability,** and **Manufacturing-friendliness.**

Architecture

CPLDs are commonly used in state machine and control applications, and they are often used to implement the last design fixes before board production. As such, they need the architectural flexibility to adapt to last minute changes without pin assignment changes. For maximum protection against unexpected design iterations, the architecture should have superior pin-locking characteristics and excellent logic resource allocation.

A RCHITECTURE

R ELIABILITY

M ANUFACTURING-FRIENDLINESS

You should look for superior switch matrix routability, wide block fan-in, an abundance of clocking options, and flexible macrocell capability. This will give you the utmost flexibility to make those last-minute changes without reworking your entire board.

Reliability

Device reliability is a critical issue, yet many CPLD manufacturers have not kept up with the latest advances. The reliability of CPLDs is a large part of overall system reliability for several reasons:

- CPLDs are being programmed and tested within the system instead of being externally programmed and tested prior to board assembly. Thus, any programming failure involves not only the device cost but also expensive board rework costs as well.
- Leading-edge manufacturers are continuing to increase the operating life of new digital systems. This puts a strain on older CPLD technologies developed for a system life of less than 10 years.
- More systems are incorporating field upgrade capabilities to prolong system life. Because in-system programming is done under variable field conditions, programming reliability is critical.

You should look for a CPLD family that offers the highest level of programming reliability and data retention, to ensure that your designs will continue to work under any conditions. Flash technology currently offers the highest reliability, with an endurance rating of 10,000 program/erase cycles and 20 years data retention. Most older technologies can offer only 100 cycles and 10 years of data retention. Fortunately, the new Flash CPLD technologies can easily support increased programming reliability levels with no additional cost.

Manufacturing-friendliness

Working prototypes do not provide sales dollars until they are produced and sold. CPLDs enable rapid test development and production release, which makes industry-standard, JTAG, in-system programming support invaluable. The IEEE Std 1149.1 (JTAG) interface is the most popular method for in-system programming, because it is supported by the most CPLD and FPGA vendors. Full conformance to the JTAG standard, enables suppliers of in-system programming and test tools to easily support your future designs.

For full compatibility with third party in-system programming suppliers, you should look for full JTAG Boundary-Scan test capability and JTAG in-system programming interface.

The FastFLASH XC9500XL Advantage

The XC9500XL 3.3V CPLD family uniquely excels in all three "ARM" criteria, and offers the highest level of programming reliability in a JTAG-compatible, in-system programmable family. The XC9500XL family features:

- The most flexible architecture
- Wide 54-input function blocks*
- Up to 90 product-terms per output*
- Three global clocks, with local inversion capability
- Immunity from all power supply sequencing problems
- Compatibility with 5V, 3.3V, and 2.5V signals
- Highest reliability rating
- 20 year data retention*
- 10,000 endurance cycles*
- Full IEEE Std 1149.1 (JTAG) test and programming
- The most complete Boundary-Scan support with eight instructions*
- JTAG supported in-system programming instructions.

"...you can rest assured that your designs will remain trouble free."

Conclusion

Arm yourself with the XC9500XL family for all your 3.3V CPLD needs, and you can rest assured that your designs will remain trouble free. Σ

**These are the highest available in the 3.3V CPLD industry.*

Get Max Headroom with XC9500 CPLDs

by John Spencer Ahn, CPLD Product Marketing Manager, john.ahn@xilinx.com

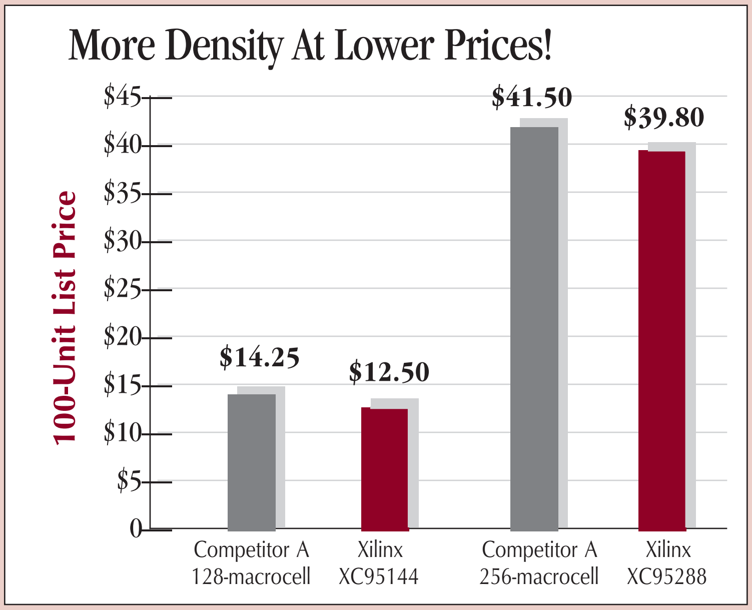
The XC9500 CPLDs are not only the most-advanced in-system programmable CPLDs in the industry, they are also the roomiest. If you are currently using other 128- or 256-macrocell CPLDs, you may be missing out on valuable density "headroom."

Two members of the XC9500 family, the XC95144 and the XC95288 offer 12% more macrocells than competing CPLDs. The XC95144 offers 144 macrocells, 16 more than the competing 128-macrocell devices currently available. The XC95288 has 288 macrocells, 16 more than the competing 256-macrocell CPLDs. This extra headroom that Xilinx offers gives you a significant advantage – you can always use more macrocells.

To top it off, the XC95144 and the XC95288 cost less than the competitor's higher-density offerings.

Conclusion

Xilinx CPLDs offer 12% more density, with up to 12% less cost – that's something to be excited about. Σ



Silicon Xpresso™ Designing with the Web

by Wallace Westfeldt, Frank Toth, Neil Jacobson and Scott Lewis

Last September, Xilinx announced a new framework for hardware development called Silicon Xpresso. As its name implies, this framework brings the World Wide Web and Java® into the hardware development system.

The Web today is already a powerful tool for the programmable logic designer. The technical nature and time to market pressure of hardware design requires immediate access to up-to-date information. All the semiconductor vendors invest significant resources in creating and maintaining these informational websites. And with this investment, their customers receive significant returns.

These websites contain application notes, solutions databases, updates to software releases, and in-depth technical descriptions of the products and their architectures. Some of these sites, such as WebLINX from Xilinx, even provide a large array of downloadable IP in the form of parameterizable and predefined cores. The usage of pre-existing verified cores as well as the automatic generation of new cores has become an invaluable time saver for programmable logic development. Thus, today's Web has become the informational infrastructure of choice of the hardware designer.

Leveraging the existence of this infrastructure and the acceptance of the powerful Java language as a standard development environment for Web-based tools and applications, Xilinx has initiated a new paradigm where the Web is also used as an interactive component in the development and design of applications. You can now use the Web for development, debug, and deployment as well as making the Web an integral part of your developed end-product. Furthermore, this means that your customer, the end user, can take advantage of this environment for field upgrades, deployment, and real-time usage of your products.

At this time, the first two phases of Silicon Xpresso have been announced. Phase 1, provides new technology in form of two new products: WebFitter and a Java API for Boundary Scan. Phase 2, is a significant enhancement of our existing technologies on the Web and in our existing development tools.

WebFitter

WebFitter is a design evaluation tool that allows you to quickly and efficiently evaluate your CPLD designs using the latest revision of our fitter software. You need not learn any software details or use any of your computing power to evaluate your design.

Typically, you need to go through many steps to evaluate a PLD design. Valuable time is spent loading, configuring, and learning design software before the design can be evaluated. You also need to make sure you have the latest software from the PLD vendor and then use your own computer to evaluate the design.

WebFitter makes it much easier to evaluate your design in silicon, and designs run fast over the network. WebFitter eliminates the need for licenses, and software CDs, freeing system time for other tasks. WebFitter is easy to use; you simply register, and submit your design. Then you receive a return email pointing to your data and analysis reports. You can select files that specify the pin-to-pin delays, mapping information, and the resulting pinout. A JEDEC Programming File can also be downloaded for use with the Xilinx

Java is a registered trademark of Sun Microsystems, Inc.

JTAG Programmer Software (and download cable) to program the Xilinx device. If an error occurs, you can gather information from the log files. You can download a zipped version of all the files, and you can choose to save or delete the design once it is completed.

Java API For Boundary Scan

The proposed Sun Microsystems Java API for Boundary Scan devices allows complete support of all JTAG and JTAG-based ISP operations for all IEEE 1149.1-compliant (JTAG) devices. Because the API is Java based it also facilitates access to this functionality over the Internet. This enabling technology then provides the framework for you to develop and deploy systems based on any Boundary Scan programmable logic device.

The Java API for Boundary Scan provides you with cross-product, multi-platform support for in-system programming, test, and debug. It takes advantage of the write once/run anywhere feature of Java and provides complete support of Boundary Scan through the entire product life cycle (prototyping, manufacturing, and field upgrade).

Currently each PLD (or other Boundary Scan product) vendor writes a separate format stimulus file for each device which requires separate translators and compilers for each platform. Having more than one type of device on a board means that you must integrate all translators and compilers together in one chain, and call separate routines. Supporting all devices and platforms is an arduous task. Any changes or additions must be painstakingly "edited in" every time a device is added or a change is made.

The Java API for Boundary Scan solution is simple and easy to use across all platforms and can seamlessly incorporate PLDs from different vendors. The write once/run anywhere feature ensures that new parts can easily be integrated into the JTAG environment, simplifying the programming flow and setting the stage for you to easily program and test multiple vendors' devices on the same board. You need not re-write code for separate translators and compilers every time a new device is introduced into the environment. In addition, because of the rich set of existing Java reference materials and development tools, systems incorporating the Java API for Boundary Scan can be quickly developed, debugged, and deployed.

The use of Java's rich set of existing class libraries enables tight integration of hardware subsystems in very powerful ways. This can facilitate the development of systems that interact with one another remotely, to re-program the system PLD's, allowing you to incorporate new functionality on-demand.

Web-Enabled Design Software

In the current release of the Foundation Series 1.5 and the upcoming release of the Alliance Series 1.5i, we have added a direct connection from the Project Manager and Design Manger to our website. This direct connection (PC only for now) means immediate access to the latest Xilinx technical information as well as CoreLINX where valuable and updated IP can be downloaded.

"Today's Web has become the informational infrastructure of choice of the hardware designer."

The Future of Silicon Xpresso

Silicon Xpresso is designed to provide increased benefits to you and your customers. By providing enabling technologies that are Web intelligent we are providing not only the environment for design but also the environment for deployment and usage. This will enable you to provide more flexible applications in a broader spectrum of markets.

FOR MORE INFORMATION, CONTACT:

- Frank Toth (Java API for Boundary-Scan) 408-8796836 frank.toth@xilinx.com
- Neil Jacobson (Java API for Boundary-Scan) 408-879-4885 neil.jacobson@xilinx.com
- Scott Lewis (WebFitter) 408-879-4556 scott.lewis@xilinx.com
- Wallace Westfeldt (Silicon Xpresso) 303-413-3280 wallace.westfeldt@xilinx.com

Guaranteeing Designs Work in All Conditions

Using Minimum and Prorated Delay Information in New Alliance Series 1.5 Software

by Julie Callow, Technical Manager, Alliance EDA Program, julie@xilinx.com; and Mahadevan Ramasame, Technical Marketing Engineer,Alliance Series, mahadeva@xilinx.com

The new Alliance Series 1.5 software provides two significant new features that support the XC4000XL family. Version 1.5 now provides minimum delay information to help you create race-free asynchronous circuits and it provides prorated delay information that allows you to analyze your circuit under varying voltage and temperature conditions.

Minimum Delays

FPGA densities now approach one million system gates, making the availability of minimum delay information a significant factor for system-level timing analysis. Simulation models generated using minimum delays ensure that race conditions are not generated with a best-case process, and guarantees correct design operation when implemented on devices made from multiple fabrication lines.

Minimum delays for timing simulation are generated using a utility called NGDANNO, which can annotate a minimum or maximum delay value in the design.sdf file. Running NGDANNO with no specific command line option will annotate worst-case delays. To generate minimum delays, run NGDANNO with the "-s min" command line option. By using the "-s min" and "-o" (output) switches, you can generate both maximum and minimum delay values.

Generate minimum delays using the following command at the DOS prompt:

```
ngdanno –s min design.ncd map.ngm –o design_min.nga
ngd2vhdl –w design_min.nga design_min.vhd
```

Generate maximum delays using the following command at the DOS prompt:

```
ngdanno design.ncd map.ngm –o design.nga
ngd2vhdl –w design.nga design.vhd
```

After the design.nga netlist is created, NGD2VER (Verilog Netlister), NGD2VHDL (VHDL Netlister), or NGD2EDIF (EDIF Netlister) is used to create the structural simulation netlist. Using NGD2VER or NGD2VHDL, the minimum delays are output into an SDF file, whereas NGD2EDIF writes minimum delays in the EDIF netlist as pin/instance properties.

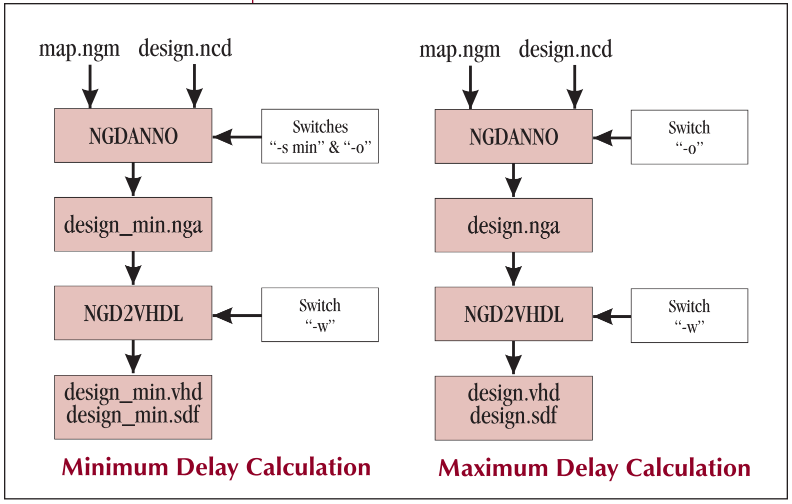
Timing Analyzer (TRACE) has the option to view the minimum delays for XC4000XL devices. Minimum delays for the nets can be viewed by setting the speed grade options to "min" and analyzing the design. In the Timing Analyzer window, click on "options," choose "speed grade" and select "min." Selection of other speed grade options generates the maximum delays.

Prorated Delays for Voltage and Temperature

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 from worst case to best 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. The default value is 70°C and 3.3V.

To calculate the prorated delays for timing simulation, specify operating conditions in the Physical Constraints File (.pcf) and run the NGDANNO utility. The syntax for specifying the temperature and voltage is:

Figure 1



```
VOLTAGE = value[units];
value = a real or integer number specifying voltage
    Units = unit of measure (volts) [optional]
TEMPERATURE = value[units];
 value = a real or integer number specifying
  temperature
    Units = unit of measure ( F, K, or C ) [optional]
    (default unit is C)
```

An example design.pcf file:

```
VOLTAGE = 3.15 ;
TEMPERATURE = 70 ;
```

Generate the prorated delays for voltage and temperature using the following command at the DOS prompt:

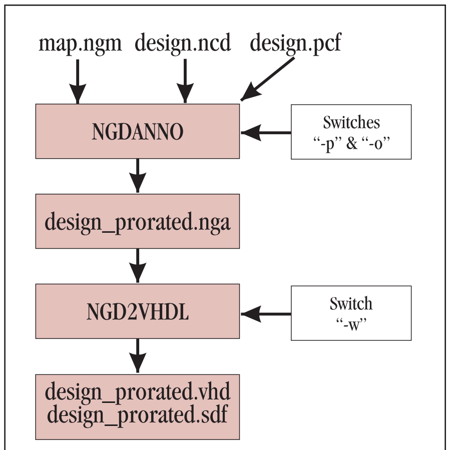
```
ngdanno -p design.pcf design.ncd map.ngm -o design_prorated.nga
ngd2vhdl -w design_prorated.nga design_prorated.vhd
```

If the specified voltage and temperature range doesn't fall within bounds, a warning will be issued and the constraint won't apply. NGDANNO will generate a warning when the standard delays (calculated at standard operating conditions) are not used during back annotation.

Timing Analyzer (TRACE) has the option to view the prorated delays for voltage and temperature for XC4000XL devices. Enter the voltage and terperature in the design.pcf file and invoke the Timing Analyzer. Click on "options," and choose the speed grade to generate the prorated delays for the specified voltage and temperature.

Conclusion

As Xilinx continues to provide higher density ASIC replacement devices, minimum delays together with voltage and temperature prorating analysis becomes an integral part of the FPGA design flow. Minimum delays along with voltage and temperature prorating support will be extended to the XC4000XLA, Spartan, and Virtex families in future Alliance Series software releases.



Prorated Delay Calculation

Figure 2

Speed Up Your XC9500 Design With The New v1.5 Software

by Dave Grace, Software Product Manager - CPLD Division, dave.grace@xilinx.com

Our latest advances in CPLD Implementation tools (v1.5) are now included in both the Alliance 1.5 and Foundation 1.5 products. Enhancements to the speed, area, and timing-driven optimization algorithms have resulted in faster runtimes and faster silicon performance.

Faster Runtime

This new release of the Xilinx implementation technology has improved runtime by 36%* over the previous V1.4 release. Plus, you will see up to 50% reductions in runtime for designs that use more than 100 macrocells.

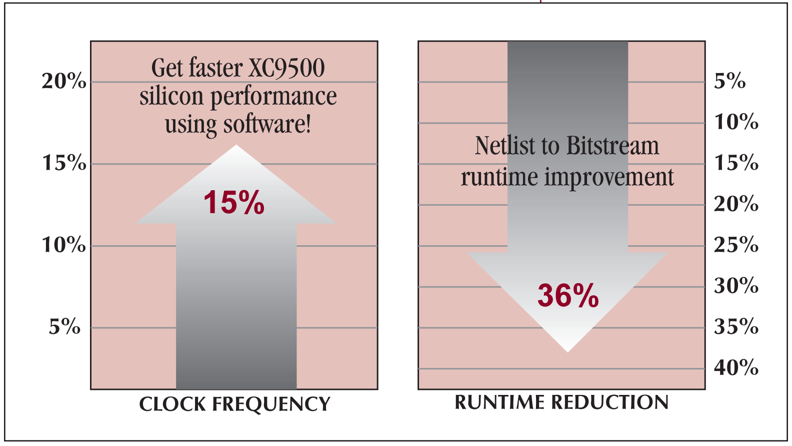
Faster Performance

Using the same device, package, and speed grade, V1.5 delivers a 15%* improvement in system clock performance. This allows you to choose the best device/package/speed grade combination, based on your cost/performance requirements.

Conclusion

If you want to create the most efficient CPLD designs, with the least time and effort, upgrade now to our new v1.5 development tools. For information on upgrading contact your local Xilinx sales representative or authorized distributor. For a complete listing of Xilinx sales contacts, visit WebLINX at **www.xilinx.com**.

**Design Test Suite of 50 HDL (VHDL & Verilog) Designs. Windows NT 4.0 & Windows 95/98.*



Unleash Your Creative Potential With The New Alliance Series 1.5 Software

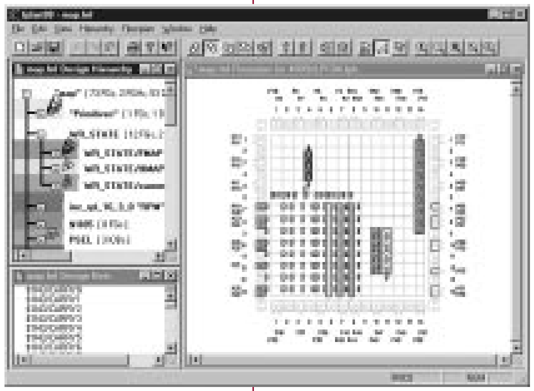
by Hitesh Patel,
Technical Marketing Manager,
hiteshp@xilinx.com

Our latest Alliance Series 1.5 software not only improves your design performance and reduces runtimes, it also provides support for the industry's first million-gate FPGAs, the new Virtex family from Xilinx. The Alliance Series also works seamlessly with industry-leading synthesis and simulation vendors like Exemplar, Synopsys, Synplicity, and Model Technology, making your job easier than ever before.

"**The Alliance Series 1.5 delivers higher frequency of operation and faster runtimes**. The intuitive design flow, with its graphical constraints editor and flexible report generation capabilities, offers greater insight to design processing, allowing the engineer to achieve the design goal quickly, and with ease…" said Praveen Shekokar, Director of Engineering at Comit Systems Inc. (www.comit.com).

Productivity Enhancements

The Alliance Series 1.5 software uses our new AKA*speed*™ technology to deliver a suite of new algorithms and features that give you the industry's fastest timing-driven compile times — up to 30x faster than MAX+plus II — with clock performance improvements up to 30%.


Floorplanner

A graphical constraint editor guides you to the best constraint methodology, reducing mistakes and eliminating the need to learn cryptic syntax. You enter design constraints through an easy-to-use graphical user interface that allows you to select constraints such as clock rates, input setup delays, and clock-to-out delays. This powerful editor allows you to easily create groups based on net names for multi-cycles path constraints.

AKA*speed* technology brings the power of ASIC development tools to the FPGA world by allowing timing analysis and delay simulation with both minimum and maximum delays as well as voltage and temperature prorating factors. In high-performance systems, minimum delay analysis is a requirement for creating race-free asynchronous circuits and guaranteeing correct design operation.

Design Planning

The floorplanner has been rewritten from the ground up to leverage your expertise. Physical design floorplanning allows you to analyze a design's hierarchy, identify structured and unstructured logic, and create an optimal physical hierarchy for your design. Most designs do not need floorplanning to meet the design requirements, however if you need maximum design performance, this tool delivers the power and flexibility of area-based floorplanning. For example, you can precisely place large RAM arrays or relatively place structured logic.

Device Support

The Alliance Series 1.5 software supports our new Virtex family, which features a synthesis-friendly architecture, fabricated on an advanced 0.25µ process. It is the first FPGA architecture to offer system design capabilities with 160MHz chip-to-chip communication, DLLs, 133MHz access to external and internal memory, and true dual-port memory. Virtex offers large on-chip memory for buffering and processing as well as flexible interfaces that connect to existing and emerging I/O standards.

In addition, the Alliance Series software supports the new XC4000XLA, SpartanXL, and the XC9500XL device families.

Installation and Security

No license servers or dongles are required. Simply enter the serial number, found on the back of your CD package when you install the Alliance Series software, for instant access.

Conclusion

The new Alliance Series 1.5 software enables you to quickly create high performance designs using the most advanced FPGAs and CPLDs in the industry.

COLUMN

Using Nested If Statements

THE XILINX HDL ADVISOR

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

Improper use of the "NESTED IF" statement can result in increased area and longer delays. Each IF keyword specifies priority-encoded logic. To avoid long path delays, do not use extremely long NESTED IF constructs as shown in the following VHDL/Verilog examples. These designs are shown implemented in gates in **Figure 1**. Following these examples are VHDL and Verilog designs that use the CASE construct with the NESTED IF to more effectively describe the same function. The CASE construct reduces the delay by approximately 3 ns (using an XC4005E-2 part). The implementation of this design is shown in **Figure 2**.

Inefficient Use of Nested If Statement

VHDL EXAMPLE

```
– NESTED_IF.VHD
– May 1997
Library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
use IEEE.STD_LOGIC_ARITH.all;
entity nested_if is
   port (ADDR_A: in std_logic_vector (1 downto 0); – ADDRESS Code
        ADDR_B: in std_logic_vector (1 downto 0); – ADDRESS Code
        ADDR_C: in std_logic_vector (1 downto 0); – ADDRESS Code
        ADDR_D: in std_logic_vector (1 downto 0); – ADDRESS Code
        RESET: in std_logic;
        CLK : in std_logic;
        DEC_Q: out std_logic_vector (5 downto 0)); – Decode OUTPUT
end nested_if;

architecture xilinx of nested_if is
begin
––––––– NESTED_IF PROCESS –––––––
NESTED_IF: process (CLK)
begin
   if (CLK'event and CLK = '1') then
if (RESET = '0') then
   if (ADDR_A = "00") then
     DEC_Q(5 downto 4) <= ADDR_D;
     DEC_Q(3 downto 2) <= "01";
     DEC_Q(1 downto 0) <= "00";
       if (ADDR_B = "01") then
         DEC_Q(3 downto 2) <= unsigned(ADDR_A) + '1';
         DEC_Q(1 downto 0) <= unsigned(ADDR_B) + '1';
           if (ADDR_C = "10") then
             DEC_Q(5 downto 4) <= unsigned(ADDR_D) + '1';
             if (ADDR_D = "11") then
               DEC_Q(5 downto 4) <= "00";
         end if;
           else
             DEC_Q(5 downto 4) <= ADDR_D;
           end if;
         end if;
       else
         DEC_Q(5 downto 4) <= ADDR_D;
         DEC_Q(3 downto 2) <= ADDR_A;
         DEC_Q(1 downto 0) <= unsigned(ADDR_B) + '1';
       end if;
     else
       DEC_Q <= "000000";
     end if;
   end if;
end process;
end xilinx;
```

VERILOG EXAMPLE

```
////////////////////////////////////////
// NESTED_IF.V            //
// Nested If vs. Case Design Example //
// August 1997            //
////////////////////////////////////////
module nested_if (ADDR_A, ADDR_B, ADDR_C, ADDR_D, RESET, CLK,
  DEC_Q);
   input [1:0] ADDR_A ;
   input [1:0] ADDR_B ;
   input [1:0] ADDR_C ;
   input [1:0] ADDR_D ;
   input RESET, CLK ;
   output [5:0] DEC_Q ;
   reg [5:0] DEC_Q ;
// Nested If Process //
always @ (posedge CLK)
begin
   if (RESET == 1'b1)
     begin
       if (ADDR_A == 2'b00)
         begin
           DEC_Q[5:4] <= ADDR_D;
           DEC_Q[3:2] <= 2'b01;
           DEC_Q[1:0] <= 2'b00;
             if (ADDR_B == 2'b01)
               begin
                 DEC_Q[3:2] <= ADDR_A + 1'b1;
                 DEC_Q[1:0] <= ADDR_B + 1'b1;
                   if (ADDR_C == 2'b10)
                     begin
                       DEC_Q[5:4] <= ADDR_D + 1'b1;
                         if (ADDR_D == 2'b11)
                           DEC_Q[5:4] <= 2'b00;
                     end
                   else
                     DEC_Q[5:4] <= ADDR_D;
               end
         end
       else
         DEC_Q[5:4] <= ADDR_D;
         DEC_Q[3:2] <= ADDR_A;
         DEC_Q[1:0] <= ADDR_B + 1'b1;
     end
   else
       DEC_Q <= 6'b000000;
end
endmodule
```

Continued on the following page

Continued from the previous page

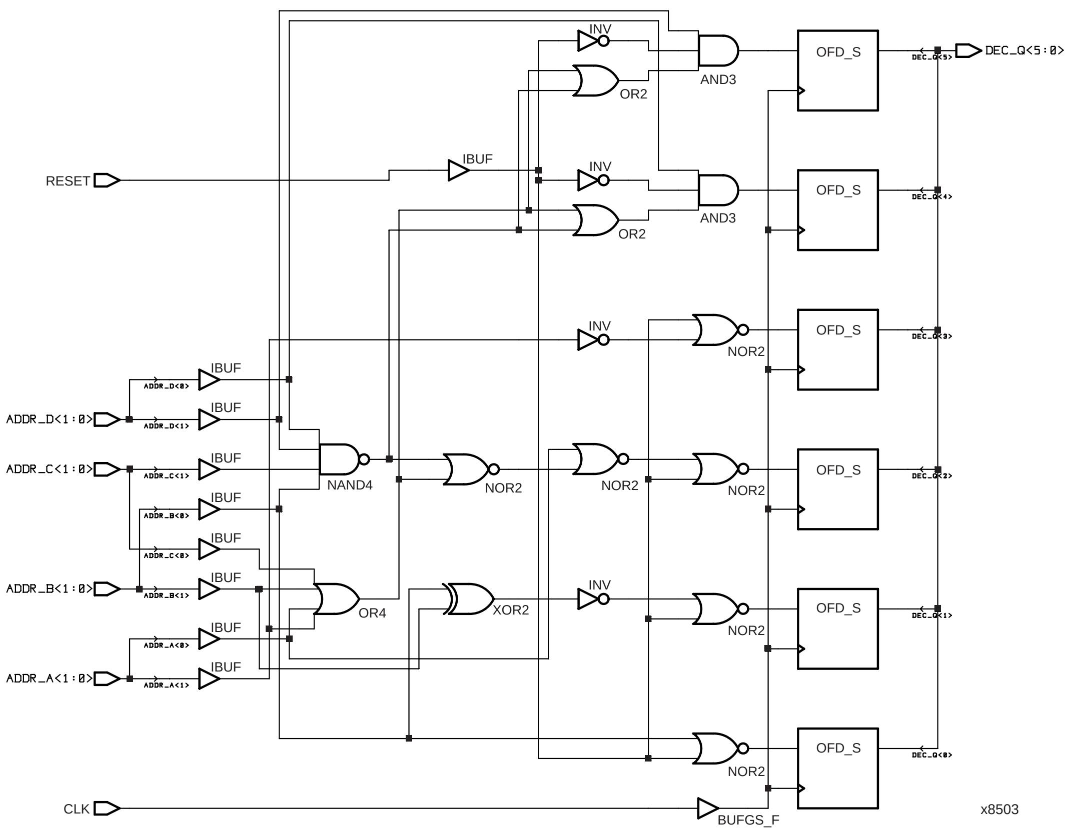


Figure 1: The gate implementation for a design with an inefficient use of a nested if statement.

Nested If Example Modified to Use If-Case

***Note:** In the following example, the hyphens ("don't cares") used for bits in the CASE statement may evaluate incorrectly to False for some synthesis tools.*

VHDL EXAMPLE

```
— IF_CASE.VHD
— May 1997
Library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
use IEEE.STD_LOGIC_ARITH.all;
entity if_case is
   port (ADDR_A: in std_logic_vector (1 downto 0); — ADDRESS
  Code
        ADDR_B: in std_logic_vector (1 downto 0); — ADDRESS Code
        ADDR_C: in std_logic_vector (1 downto 0); — ADDRESS Code
        ADDR_D: in std_logic_vector (1 downto 0); — ADDRESS Code
        RESET: in std_logic;
        CLK : in std_logic;
        DEC_Q: out std_logic_vector (5 downto 0)); — Decode
  OUTPUT
end if_case;

architecture xilinx of if_case is
signal ADDR_ALL : std_logic_vector (7 downto 0);
begin
——concatenate all address lines ———————————-
ADDR_ALL <= (ADDR_A & ADDR_B & ADDR_C & ADDR_D) ;
————Use 'case' instead of 'nested_if' for efficient gate
  netlist———
IF_CASE: process (CLK)
begin
   if (CLK'event and CLK = '1') then
      if (RESET = '0') then
         case ADDR_ALL is
           when "00011011" =>
              DEC_Q(5 downto 4) <= "00";
              DEC_Q(3 downto 2) <= unsigned(ADDR_A) + '1';
              DEC_Q(1 downto 0) <= unsigned(ADDR_B) + '1';
           when "000110—" =>
              DEC_Q(5 downto 4) <= unsigned(ADDR_D) + '1';
              DEC_Q(3 downto 2) <= unsigned(ADDR_A) + '1';
              DEC_Q(1 downto 0) <= unsigned(ADDR_B) + '1';
           when "0001——" =>
              DEC_Q(5 downto 4) <= ADDR_D;
              DEC_Q(3 downto 2) <= unsigned(ADDR_A) + '1';
              DEC_Q(1 downto 0) <= unsigned(ADDR_B) + '1';
           when "00———" =>
              DEC_Q(5 downto 4) <= ADDR_D;
              DEC_Q(3 downto 2) <= "01";
              DEC_Q(1 downto 0) <= "00";
           when others =>
              DEC_Q(5 downto 4) <= ADDR_D;
              DEC_Q(3 downto 2) <= ADDR_A;
              DEC_Q(1 downto 0) <= unsigned(ADDR_B) + '1';
         end case;
      else
         DEC_Q <= "000000";
      end if;
   end if;
end process;
end xilinx;
```

VERILOG EXAMPLE

```
///////////////////////////////////////
// IF_CASE.V              //
// Nested If vs. Case Design Example  //
// August 1997             //
///////////////////////////////////////
module if_case (ADDR_A, ADDR_B, ADDR_C, ADDR_D, RESET, CLK,
  DEC_Q);
   input [1:0] ADDR_A ;
   input [1:0] ADDR_B ;
   input [1:0] ADDR_C ;
   input [1:0] ADDR_D ;
   input RESET, CLK ;
   output [5:0] DEC_Q ;
   wire [7:0] ADDR_ALL ;
   reg [5:0] DEC_Q ;
// Concatenate all address lines //
assign ADDR_ALL = {ADDR_A, ADDR_B, ADDR_C, ADDR_D} ;
// Use ‘case’ instead of ‘nested_if’ for efficient gate netlist
  //
always @ (posedge CLK)
begin
   if (RESET == 1’b1)
     begin
        casex (ADDR_ALL)
          8’b00011011: begin
             DEC_Q[5:4] <= 2’b00;
             DEC_Q[3:2] <= ADDR_A + 1;
             DEC_Q[1:0] <= ADDR_B + 1’b1;
          end
          8’b000110xx: begin
             DEC_Q[5:4] <= ADDR_D + 1’b1;
             DEC_Q[3:2] <= ADDR_A + 1’b1;
             DEC_Q[1:0] <= ADDR_B + 1’b1;
          end
          8’b0001xxxx: begin
             DEC_Q[5:4] <= ADDR_D;
             DEC_Q[3:2] <= ADDR_A + 1’b1;
             DEC_Q[1:0] <= ADDR_B + 1’b1;
          end
          8’b00xxxxxx: begin
                    DEC_Q[5:4] <= ADDR_D;
             DEC_Q[3:2] <= 2’b01;
             DEC_Q[1:0] <= 2’b00;
          end
          default: begin
             DEC_Q[5:4] <= ADDR_D;
             DEC_Q[3:2] <= ADDR_A;
             DEC_Q[1:0] <= ADDR_B + 1’b1;
          end
        endcase
     end
   else
     DEC_Q <= 6’b000000;
end
endmodule
```

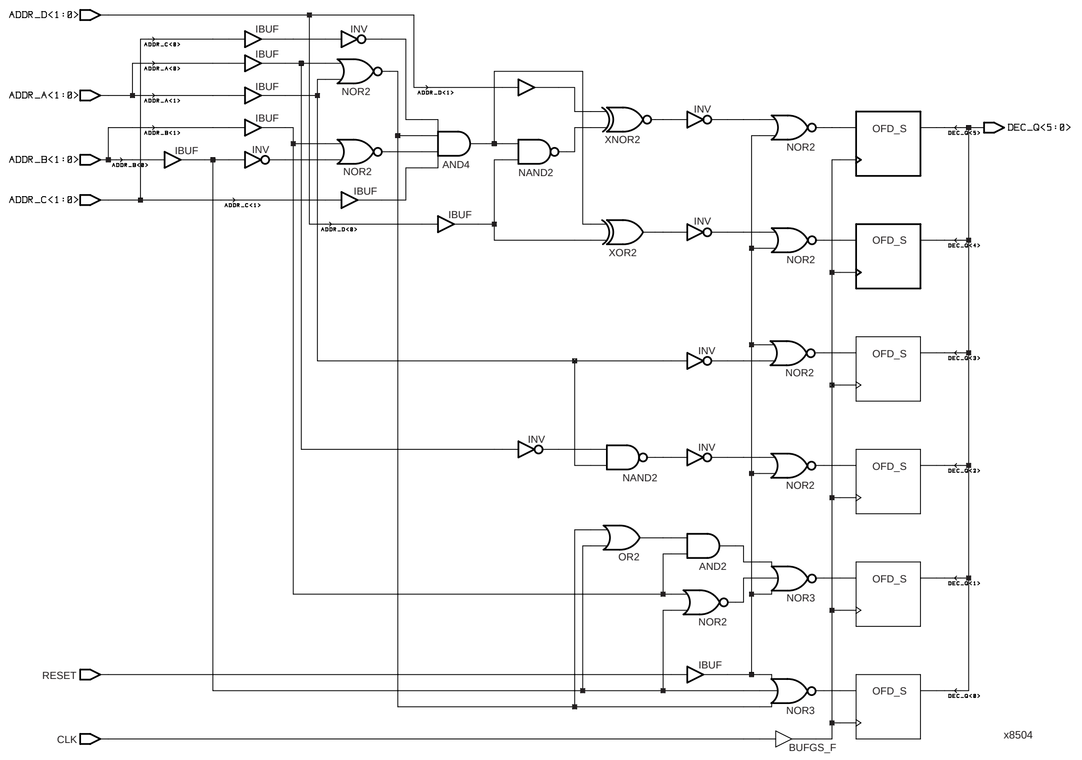


Figure 2: The gate implementation for a design modified to use if-case.

Comparing the If Statement and the Case Statement

The IF statement generally produces priority-encoded logic and the CASE statement generally creates balanced logic. An IF statement can contain a set of different expressions while a CASE statement is evaluated against a common controlling expression. In general, use the CASE statement for complex decoding and use the IF statement for speed critical paths. Most current synthesis tools can determine if the IF-ELSEIF conditions are mutually exclusive, and will not create extra logic to build the priority tree.

System Emulation and Rapid

by Roberta Fulton,
Alliance EDA Technical
Marketing Engineer,
roberta@xilinx.com

Several third-party products now use Xilinx FPGAs to provide you with system emulation and rapid-prototyping tools that help you get your designs up and running as quickly as possible. The same features that make Xilinx FPGAs excellent ASIC replacements also make them excellent emulation devices.

Because Xilinx FPGAs offer internal performance of over 100MHz, these third-party emulators can run at system speeds when efficient EDA tools and verification flows are used to create the emulation netlist. In addition, the ability to reprogram during system debugging is a key feature

System, and system-on-a-chip (SOC), ASIC emulation with Xilinx FPGAs provides you with three distinct advantages:

- **Speed** - In a matter of hours, with an emulation engine, you can exercise more product functions than are possible in days or weeks of simulation.
- **Rapid prototyping -** With the automated emulation software and ready made emulation hardware, system prototypes involving ASICs, FPGAs, standard parts, and microprocessors can be produced very quickly. Development and debugging of the prototypes is much easier and faster with an emulation system than with a home-grown breadboard.
- **Promotes hardware/software co-design** - Real software can be run on the emulated system, so hardware and software interface issues are caught early. Overall system development time is greatly reduced because costly re-work cycles are shortened in an emulated system that can be reprogrammed rather than remanufactured.

"The same features that make Xilinx FPGAs excellent ASIC replacements also make them excellent emulation devices."

The Advantages of Xilinx High-density FPGAs

System designers, or their team members in verification, can emulate vast sections or even whole systems-on-a-chip (SOC) using high capacity, high-performance Xilinx FPGAs. "In a system prototyping environment, the large capacity, high pin count FPGAs, such as the Xilinx XC4000XV family, greatly simplify the process of building system prototypes with large ASICs. Our customers are able to preserve the hierarchy of their design by mapping large design blocks into individual FPGAs for prototyping. Consequently design debugging becomes much more intuitive," stated Michel Courtoy, Director of Product Marketing at Aptix Corporation.

Whether a Xilinx FPGA, an ASIC, or both are used in your system design, emulation can be a critical tool for system verification because simulation speed becomes a major barrier to on-time delivery as the complexity of chips and systems increase. The functional simulation and verification of high-density HDL is a major portion of the system design schedule. Design verification and the resulting timing and functional closure of the design will often be two to four times greater than the design entry time.

Running a significant number of well-constructed vectors, in a timely manner, is crucial to catching the maximum number of bugs in the shortest time. The earlier in the design and implementation cycle the functional errors are caught and corrected the greater the savings of time and effort. Emulators can run greater than 10^6 times faster than workstation simulators. The thoroughness of the verification is especially true if several components of the system can be simulated together. And, if more of those components can be placed on the same high-density Xilinx FPGA, you need to do less partitioning and therefore the emulation can be created that much sooner. The high-density parts from Xilinx with capacities of

Prototyping Using Xilinx FPGAs

100K to one million gates can aid in the retention of design hierarchy and in reducing the partitioning overhead.

High-End Emulation Systems

There are a number of fine methodologies and tools for emulation and rapid-prototyping that use Xilinx FPGAs to their advantage. FPGA-based emulation is recognized as the most flexible hardware emulation technology. It provides the highest in-circuit emulation speeds and supports all design styles from fully synchronous to fully asynchronous. The Aptix System Explorer series and the Quickturn System Realizer are two high-end emulation products using Xilinx FPGAs for emulation.

This technology as expressed in Quickturn's System Realizer product can support designs from 100k to three million gates. HDL code is compiled, partitioned into blocks, and mapped into the logic elements of the Xilinx FPGAs within the System Realizer. The product can support IP as soft cores in either HDL or netlist forms, including vendor encryption if necessary. Or the core can be bonded out in the Component Adapter Card in the Programmable Target Interface Module (PTIM) which is a standard part of System Realizer.

Once the design is in the system, powerful debugging tools can be brought into play as a large number of vectors are processed at speeds several orders of magnitude higher than simulators. The Vector Debug mode reads vector data off of your workstation disk and applies it to the circuit in the emulator. The Regression Test mode provides automatic vector comparison for quick-go/no-go testing. 128K of vectors can be run at speeds up to 1MHz in an IC-tester-like validation environment. Add-in cards can increase the vector capacity. The In-Circuit Emulation mode is the most powerful and comprehensive way to verify complex systems. The emulation hardware becomes part of the system hardware prototype that runs real data and software.

The Aptix solution offers an open architecture approach. The Aptix System Explorer MP3A and MP4 products are based on Field Programmable Interconnect Components (FPIC) and Field Programmable Circuit Boards (FPCB). The FPCBs allow component insertions directly on the board or through a daughter-board module using the standard hole patterns on the FPCBs. The Aptix Axess software connects your components on the FPCB through the reprogrammable FPIC.

The Aptix emulation systems are very flexible. The programmable interconnect devices, FPGAs, and other devices can be mounted on a board that provides routing paths such that each pin hole on the board connects to a programmable interconnect path. You can also mount them on daughter boards allowing you to mix and match different Xilinx FPGAs and quickly upgrade to the latest versions to obtain the best performance. Other system devices such as RF components or microprocessors can also use this daughter board approach.

Because of this open architecture, multiple projects can use the System Explorer. All popular EDA design tools are supported as well. This system can be used as a "faster simulation" engine or as a hardware prototype allowing real data and software to be run through the emulated hardware system. With flash memory to hold the interconnect paths the emulated systems can be detached from the workstation and you have a rapid-prototype you can put in the real system environment.

These Aptix and Quickturn solutions provide the greatest flexibility for both running simulations and hardware prototyping with a wealth of development and debugging tools and options.

Conclusion

Xilinx FPGAs, with their ever-increasing capacities and speed performance, are crucial to the verification of systems and large ASICs through FPGA-emulation.

Reducing CPLD Power Consumption

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

Power usage in CMOS circuits appears to be straightforward, yet it is often deceptive. This article will help you understand CPLD power dissipation and will give you some guidelines for minimizing power consumption.

Most vendors provide a table or an equation that specifies the various components of power dissipation for a CMOS part, as shown in **Figure 1**. Typically, these include:

- A component for the input receivers, which must be de-rated if driven from TTL rather than CMOS external drivers.
- A component for the internal core of the chip, which usually has a negligible DC component
- An AC component that requires detailed knowledge of the various switching frequencies encountered. It also requires knowledge of exactly how much circuitry is used or unused at any time.
- A component attributed to the output stages which are functions of both the switching frequencies as well as the external load capacitance.

The power is found as the sum of all components:

Power = $P_{IN} + P_{CORE} + P_{OUT}$ (AC and DC)

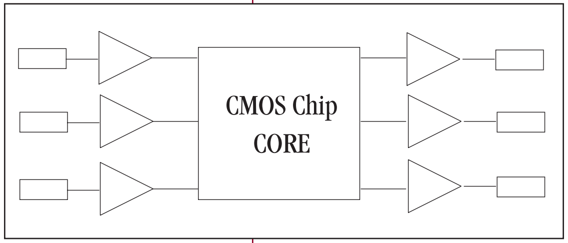


Figure 1: Power Components of a Typical CMOS Chip

This approach is intellectually satisfying, but usually you find that you have little knowledge of the load capacitance or various switching frequencies that your circuits create and encounter. You must resort to estimating the speed and loading parameters to obtain a power estimate. One estimate is often used as the basis for another, giving fuzzy, inaccurate results.

Because of the complexity of arriving at a simple number, many CPLD vendors have resorted to simplifying the power estimation process by providing a single equation. Frequently, when misused, this results in optimistic values. Often, calculation constants are introduced to simplify the process, but no limitation guidelines or explanation of their meaning is given.

Power Dissipation Factors

The basic structure of a CPLD differs from other CMOS devices primarily in its programmable internal core. If **Figure 1** was modified to insert a programmable AND array structure into the core, most of the power differences between a CPLD and other CMOS chips would be explained. Input power must be accounted for, as well as output power, but the core power is different, due to the CPLD sense amplifier approach.

Figure 2 shows a simplified structure for a programmable "AND" gate (actually, it is a NOR internally) which performs the programmable logic operations. It shows three transistors with floating gates that form a "Wired NOR" when appropriately programmed. XC9500XL devices normally have 108 transistors attached to the Bitline for each product term. The pullup and pulldown resistors (R1 and R2) attached to the Bitline are actually transistors.

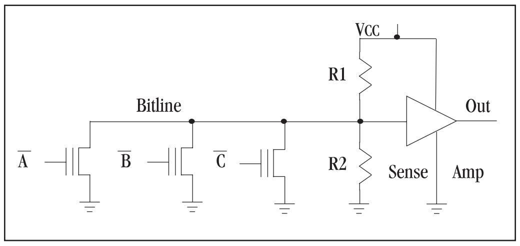


Figure 2: Simplified CPLD Programmable Structure

When the Bitline is High (exceeding the trip voltage of the sense amplifier), the output switches. Otherwise, the output remains Low. From a power dissipation viewpoint, lower consumption occurs when $V_{bitline\_hi}$ is driven onto the bitline. Because CPLDs are comprised of macrocells that include flip-flops, it is important to realize that the flip-flops consume negligible power compared to the pro-

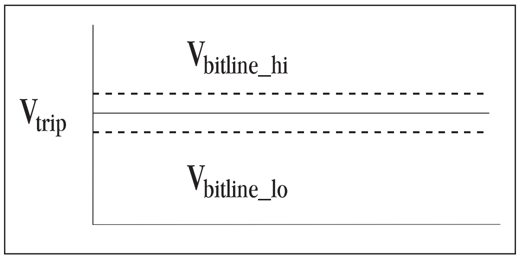


Figure 3: Trip Voltage, Bitline High and Bitline Low Relationships

grammable cells. With that in mind, a CPLD design can be viewed as being a collection of product terms driving pins and flip-flops. The switching speed of the product terms and output pins becomes the dominant factor in most cases.

There is a component of current always present in the standard CPLD programmable structure. The current will be typically one of two values passing either through R2 or through the transistors to ground. This is the primary factor contributing to DC current consumption in the CPLD core. It cannot be ignored.

R1 in **Figure 2** is actually a programmable transistor structure. R1 can be programmed to supply current to the Sense Amp input node to select between a fast ramping input signal, or a low current slower ramping signal. **Table 1** gives relative values of the current drawn by a product term depending on the value of R1 and the condition of the Bitline.

Table 1 indicates several things:

- Each product term can have several different static current values.
- The range is large (10X).
- The value is a strong function of whether the product term is driving High or Low.

For combinatorial designs, where all product term inputs are directly driven from the input pins, this is easily accounted for and an accurate static power estimation can be measured. For sequential circuits, the binary values of flip-flops will be attached to the various product terms, so it is difficult to know what the power consumption is unless careful analysis includes the state of the circuit in the estimation.

In **Figure 4**, consider the case where Input 1 and Input 2 are both logical ones. If the state variable feeds back a logical one, the Bitlines will both be High. However, if the State variable feedsback a logical zero, both Bitlines will be Low and draw significantly more current. This means that system reset may draw the highest current. Frequently, this behavior is seen when you attempt to measure accurate power when the clock is stopped. **Figure 5** gives an example of how this creates an ambiguous measurement.

As shown in **Figure 5**, there exists an ambiguous region for ICC when the switching frequency is very near DC. This is because the exact state of the sequential machine will dictate just how many specific bitlines are High or Low when the clock is turned off (or very slowly switching). Very quickly after the frequency rises, the ICC assumes a much more linear relationship with frequency. Although counter-intuitive, it is possible for a short frequency range to have ICC drop as frequency rises.

Table 1: Relative Product Term Currents

R1 Configuration	Bitline = High	Bitline = Low
High Speed	0.5I	1.0I
Low Power	0.1I	0.5I

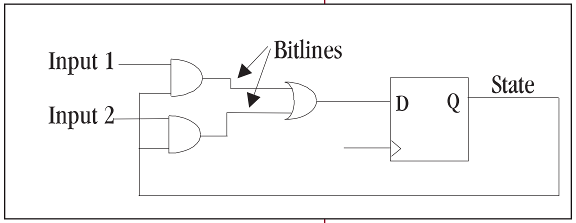


Figure 4: State Influence on CPLD Power

Power Minimization Techniques

The following checklist will help you dramatically lower power consumption:

- **Minimize HP macrocells** - By carefully selecting only those macrocells that need to be in high speed mode, others can be set into low power mode, which will reduce power.
- **Use global resources** - Product term clocks, 3-states, and set/resets may increase s-term current draw.
- **Set VCCIO to 2.5V** - Restricting the voltage swing of the output stage will lower the CV ^{2}f portion of the output power consumed.
- **Attach unused XC9500 input pins to a UPG** - Unused pins should not float. One easy way to do this is to use the User Programmable Ground Option (UPG), which drives the pins Low and gives additional noise immunity. (XC9500 devices have bus-hold circuitry that automatically sets input pins to a known state.)

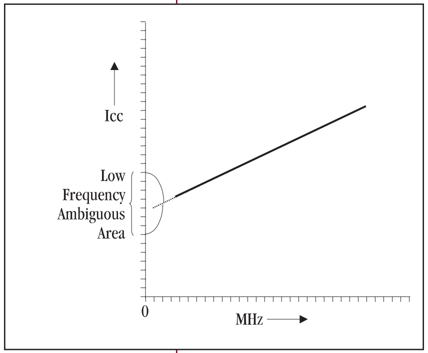


Figure 5: I_{CC} VS Frequency

Conclusion

The general ideas discussed here are applicable to most manufacturers' CPLDs, but in particular to both the Xilinx XC9500 and XC9500XL CPLDs. Xilinx CPLDs offer abundant options to substantially reduce power dissipation and still provide high performance.

Inferring RAM in Synplify

**by Allen Drost,
Corporate Applications
Engineering Group
Manager,
allen@synplicity.com**

Synplicity has added automatic RAM inferencing to Synplify version 5.0.5. Now, you no longer need to manually instantiate RAM as a black box or Xilinx-specific primitive; you can make designs that are truly technology independent.

This article describes how to successfully incorporate RAM into your next VHDL or Verilog design with Synplify. Synplify integrates both RAM timing estimates and regular timing constraints to efficiently optimize your next design, and includes:

- Automatic synchronous RAM inferencing
- SelectRAM™ or register implementations
- Timing estimates on RAM blocks
- Flexible coding styles

RAM Implementation

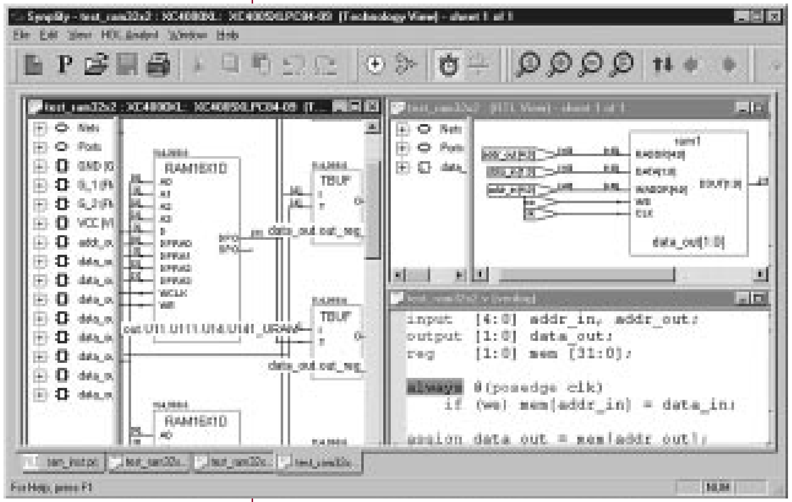
Synplify v5.0.5 can automatically infer RAM structures when coded as an indexed array or as a CASE statement. However, it will infer only synchronous RAMs; asynchronous RAMs are not supported. See the following examples for a suggestion on coding styles for RAM blocks.

When a RAM block is recognized, Synplify will automatically implement the circuit using RAM16X1S, RAM32X1S, and RAM16X1D Xilinx RAM primitives. If a RAM block is more complex than a single 16x1 primitive, Synplify creates the necessary write enable and data multiplexing logic to implement the circuit using multiple RAM blocks. HDL Analyst displays a RAM block in the RTL view, making the schematic view easier to read. To see how the large RAM blocks are implemented, use the technology view in HDL Analyst, as shown in Figure 1.

If you want to map your RAM into standard logic and registers, you can disable the usage of Xilinx SelectRAM ™, by setting the syn_ramstyle attribute to "registers." This attribute can be applied directly in the HDL source code or through Synplify's Synthesis Constraint OPtimization Environment (SCOPE ™).

To effectively optimize designs that incorporate RAM, the synthesis tool must understand the timing delays through the RAM blocks for the critical path optimization. Synplify understands the timing characteristics of the RAM primitives, and includes all RAM delays in the analysis and optimization of critical paths.

"Inferring RAM is the most effective method of designing memory into your FPGA design. Synplify provides a full suite of features to implement, analyze, and optimize your RAM design."



```
input   [4:0] addr_in, addr_out;
output  [1:0] data_out;
reg     [1:0] mem [31:0];

always @(posedge clk)
    if (we) mem[addr_in] = data_in;

assign data_out = mem[addr_out];
```

Figure 1: The Technology View in HDL Analyst

Examples

(**Note:** Synplify supports inferencing of single and dual ported RAM in VHDL and Verilog, using either indexed arrays or case statements)

Example 1: Single-ported VHDL

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

entity ramtest is
  port ( q : out std_logic_vector(3 downto
 0);
d : in std_logic_vector(3 downto 0);
addr: in std_logic_vector(2 downto 0);
we: in std_logic;
clk : in std_logic);
end ramtest;

architecture rtl of ramtest is
  type mem_type is array (7 downto 0) of
 std_logic_vector (3 downto 0);
  signal mem : mem_type;
begin
  q <= mem(conv_integer(addr));
  process (clk, we, addr) begin
    if (rising_edge (clk)) then
      if (we = '1') then
        mem(conv_integer(addr)) <= d;
      end if;
    end if;
  end process;
end rtl;
```

Example 2: Dual-ported VHDL

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

entity ramtest is
  port ( q    : out std_logic_vector(3 downto
 0);
d    : in std_logic_vector(3 downto 0);
addr_in    : in std_logic_vector(2 downto 0);
addr_out : in std_logic_vector(2 downto 0);
we   : in std_logic;
clk    : in std_logic);
end ramtest;

architecture rtl of ramtest is
  type mem_type is array (7 downto 0) of
 std_logic_vector (3 downto 0);
  signal mem : mem_type;
begin
  q <= mem(conv_integer(addr_out));
  process (clk, we, addr_in) begin
    if (rising_edge (clk)) then
      if (we = '1') then
        mem(conv_integer(addr_in)) <= d;
      end if;
    end if;
  end process;
end rtl;
```

Example 3: Single-ported Verilog

```
module test_ram32x2 (clk, we, addr, data_in,
  data_out);
input clk, we;
input [1:0] data_in;
input [4:0] addr;
output [1:0] data_out;
reg [1:0] mem [31:0];

always @(posedge clk)
  if (we) mem[addr] = data_in;

assign data_out = mem[addr];
endmodule
```

Example 4: Dual-ported Verilog

```
module ram4x4(z, raddr, d, waddr, we, clk);
output [3:0] z;
input [3:0] d;
input [1:0] raddr, waddr;
input we;
input clk;
reg [3:0] z;

reg [3:0] mem0, mem1, mem2, mem3;

always @(mem0 or mem1 or mem2 or mem3 or
  raddr)
begin
case (raddr[1:0])
  4'b00: z = mem0;
  4'b01: z = mem1;
  4'b10: z = mem2;
  4'b11: z = mem3;
  endcase
end

always @(posedge clk) begin
  if(we) begin
    case (waddr[1:0])
    4'b00: mem0 = d;
    4'b01: mem1 = d;
    4'b10: mem2 = d;
    4'b11: mem3 = d;
    endcase
  end
end

endmodule
```

Conclusion

Inferring RAM is the most effective method of designing memory into your FPGA design. Synplify provides a full suite of features to implement, analyze, and optimize your RAM design.

For more information please see the Synplicity website at www.synplicity.com.

MINC's Upgraded PLSynthesizer Supports

by Greg Brown,
Sr. Product
Marketing Manager,
gregb@synario.com

MINC recently released a substantial upgrade of its leading edge, VHDL and Verilog synthesis product for programmable IC devices, PLSynthesizer. The new release, version 6.1, adds support for the XC4000XL and XC4000XV high-density devices from Xilinx. Runtimes for the new version are three to five times faster with vastly improved quality of results, which mean faster circuitry that occupies less space.

Key features of PLSynthesizer v6.1 include:

- Advanced VHDL/Verilog synthesis including support for IEEE and Synopsys synthesis subsets and coding styles
- Specific technology optimization for AREA / SPEED criteria with several optimization efforts
- Hierarchy handling
- Automatic resource sharing
- Technology-specific macro inference
- Automatic state machine recognition/extraction
- Several automatic state encoding algorithms
- User constraint-based synthesis and optimization
- Technology re-targeting flow
- Support of the most popular FPGA and CPLD/PLD devices
- Comprehensive and easy-to-use GUI
- Windows 95/NT and Unix support

Advanced Operator Inference Engine and Macro Generation

Operator inference refers to the procedure of identifying elements of the design that may be implemented as macro blocks rather than as more complex and costly glue-logic (Boolean equations). To achieve the highest quality of results from an HDL design, PLSynthesizer includes an extremely powerful operator inference engine that can use a built-in macro generator or Xilinx LogiBLOX. Operator inference is the procedure by which a macro is recognized within a behavioral HDL description and implemented in a technology-specific and therefore efficient implementation.

```
...
signal A,B,C,O : std_logic_vector (7 downto 0);
signal S : std_logic_vector(1 downto 0);
...
with S select
        O <= A+B when “01”,
           A when “-0”,
           B-C when others;
```

Figure 1. Operator Inference and Resource Sharing

Further, using LogiBLOX and its low-level NGD output netlist ensures you of the best operator implementation possible. This is very important on data-path and timing critical designs where inefficient operator synthesis may require you to resort to handcrafted, schematic-based designs to achieve your design goals. The operators that can be inferred by PLSynthesizer are: Adder, Adder/Subtractor, Comparator, Counter, Decrementor, Incrementor, Multiplexer, Multiplier, Registers, Subtractor, and Combinational logic shifters.

Arithmetic Resource Sharing

PLSynthesizer features automatic resource sharing (also known as folding) of adders/subtractors and multipliers. The goal is to minimize the number of such operators and the subsequent logic in the synthesized design. This optimization is based on the principle that two similar arithmetic resources may be implemented as one single arithmetic operator if they are never used at the same time. The optimizer performs both resource sharing and, if required, reduction of the number of multiplexers that are created in the process. For example, consider the code fragment shown in **Figure 1.**

PLSynthesizer can implement the architecture for this as shown in **Figure 2.**

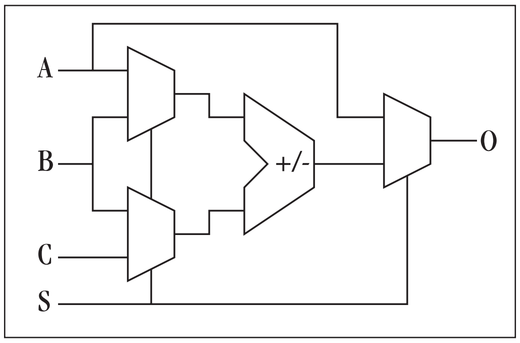


Figure 2. Intelligent Resource Sharing Results in Efficient Arithmetic Implementations

This example illustrates how PLSynthesizer can fold both adders and subtractors together and generate adder/subtractors for better optimization of the result. In addition to this technique, PLSynthesizer utilizes unique multiplexer and XOR optimizations. PLSytnthesizer can even infer combinational shifters from IF- and CASE-type code structures.

High-Density Xilinx FPGAs

Finite State Machine Extraction and Encoding

PLSynthesizer performs advanced finite state machine extraction and encoding. You can control these features on a global, entity/module, or even signal level. With this level of control, you can synthesize and optimize multiple state machine structures at the same time, each with its own optimal encoding scheme. The encoding options for Xilinx technologies are: One-hot, Compact, Sequential, Gray, Johnson, and User defined.

A unique feature of the advanced extraction process is that it is *independent* of the VHDL or Verilog coding style employed.

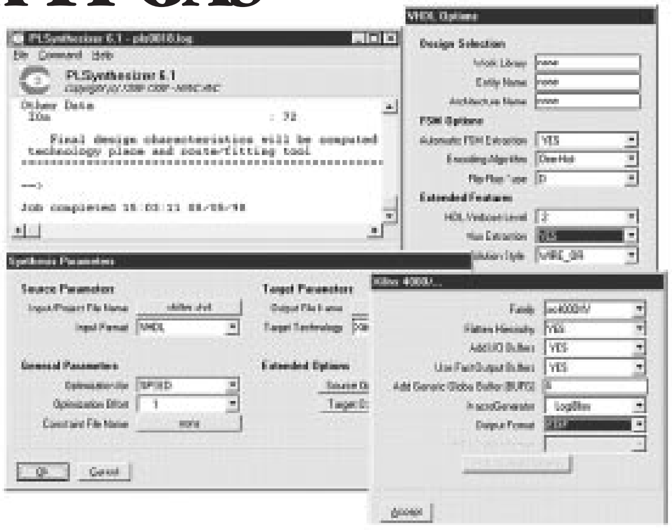
Support for Bottom-up and Hierarchical Design Methods

PLSynthesizer supports bottom-up and hierarchical design methods by giving you control over hierarchical flattening on a global and/or entity/module basis and whether or not I/O pads are to be inserted on the current top-level design. The synthesized output files can even preserve the original bus structures at the port level if desired.

Synthesis/Optimization Constraints

The use of design constraints (properties) is a powerful method to control several aspects of synthesis and post-synthesis results. Some are taken into account by PLSynthesizer to control its own operation, while others are simply passed on to back-end tools. For example, a pin assignment property applied on signals allows you to specify a given pinout in the target device.

There are three mechanisms by which constraints can be applied: within the GUI, with a separate constraint file, and with VHDL attributes. Constraints can be applied globally or locally on an entity/module basis and in some cases on individual signals or nodes. The constraints give you a high degree of control over such features as speed/area optimization tradeoffs, CPU effort level, hierarchy flattening, state machine extraction and encoding, enumerated type encoding, state register type, multiplexer and XOR extraction, combinational shifter extraction, pin assignment, and signal preservation.



```
Others Data
 IOs                                  : 72

   Final design characteristics will be computed
 technology place and route/fitting tool
=================================================

--->

Job completed 15:03:11 08/05/98
```

VHDL Options — Design Selection: Work Library: none; Entity Name: none; Architecture Name: none. FSM Options: Automatic FSM Extraction: YES; Encoding Algorithm: One-Hot; Flip-Flop Type: D. Extended Features: HDL Verbose Level: 2; Mux Extraction: YES; Resolution Style: WIRE_OR

Synthesis Parameters — Source Parameters: Input/Project File Name: shifter.vhd; Input Format: VHDL. Target Parameters: Output File Name; Target Technology. General Parameters: Optimization Mode: SPEED; Optimization Effort: 1; Constraint File Name: none. Extended Options: Source Options; Target Options

Xilinx 4000/... — Family: xc4000XV; Flatten Hierarchy: YES; Add I/O Buffers: YES; Use Fast Output Buffers: YES; Add Generic Global Buffer (BUFG): 8; MacroGenerator: LogiBlox; Output Format: XNF

Figure 3: PLSynthesizer

Simple, Yet Powerful and Easy to Use Interface

PLSynthesizer can be run from the command line on Unix platforms or from its GUI on Windows® and Unix platforms. The GUI supports a simple, three-step process (load design, select target technology, GO!) or a constraint-driven process utilizing all the power and flexibility PLSynthesizer offers. The constraint-driven mode gives you control over all PLSynthesizer features and utilizes simple and easy to understand dialogs based on source options and technology options.

Conclusion

PLSynthesizer v6.1 has the performance, capacity and quality of results to help you make the most out of Xilinx high-density FPGAs. PLSynthesizer supports the XC3000A, XC3000L, XC3100A, XC3100L, XC4000E, XC4000EX, XC4000L, XC4000XL, XC40000XV, XC5200, XC7000, and XC95000 devices. The Spartan devices will be added in v6.2, scheduled for Q4 of 1998. PLSynthesizer is available for Window® 95/NT, SunOS, Solaris, and HP-UX. PLSynthesizer is also being integrated into the MINC Synario design environment to provide a full range design entry, analysis, and implementation environment.

Greg Brown has more than ten years experience in the EDA industry and has held positions in development, consulting, management, and marketing of IC, ASIC, and programmable logic design tools at Mentor Graphics, VeriBest, and MINC Incorporated. Currently, he is the Sr. Product Marketing Manager at MINC.

For more information, contact: **MINC Incorporated,** 6755 Earl Drive Colorado Springs, Colorado 80918, www.minc.com or www.synario.com

Managing the Design Process with

by Philip Lewer,
Viewlogic Product
Marketing Manager,
plewer@viewlogic.com

VIEWlogic

With Xilinx and third-party intellectual property (IP) companies providing solutions through the Xilinx LogiCORE and AllianceCore programs, you now have the choice of making or buying functionality for your designs. This has created a situation where various design sources may reside inside a single high-density FPGA.

For example, you may have a top-level schematic tying all of your underlying macros together. These macros may consist of blocks of purchased IP, in-house VHDL and Verilog, and even some schematic primitives. When combining these various design sources with a gamut of entry, synthesis, and verification tools, developing a workable design flow becomes an overwhelming task, and that's where IntelliFlow from Viewlogic becomes extremely useful.

For example, assume that you have a design that has a top-level schematic with underlying VHDL and Verilog blocks. When you are ready to take this design through place and route and back into your simulation environment for verification, you will probably come across the following set of issues:

- How do I synthesize my language macros and integrate these with my schematic?
- What options do I choose in the place and route tools so that I get the correct netlist out for verification?
- I'm targeting a device in a BG560 package. How am I going to create a 560 pin symbol with all of my power, ground, no connect, and programming pin information to use on my board design?

"Full control over all of the Xilinx implementation tools is available through the IntelliFlow interface, giving you a combination of ease of use without loss of power."

IntelliFlow,™ which is part of Viewlogic's Workview Office ® suite of tools, is a turnkey process manager for the design of complex programmable devices. The purpose behind a process manager like IntelliFlow is to take care of these issues for you. This way you can concentrate on the design without spending time on the interface details.

With IntelliFlow, you can mix schematics, VHDL, Verilog, and blocks of IP to generate high-quality results for your Xilinx FPGAs and CPLDs. IntelliFlow understands the mixed formats of the source files in your design and automatically configures tool flows to make the proper conversions between formats automatically, allowing you to focus on the design and not the interaction of tools.

For language-only or mixed language-schematic designs, the types of tasks supported by IntelliFlow are:

- Functional Simulation
- Synthesis
- Place and Route
- Timing Simulation
- Bit and PROM File Creation
- Automatic Board-level Symbol Creation

With the IntelliFlow process manager, the first step in doing a Xilinx design is to add your design source. In the example case of a block-level diagram with underlying language blocks, you would add your top-level schematic. IntelliFlow will parse and extract the design hierarchy for you, recognizing the VHDL and Verilog blocks and automatically pulling the necessary source files into the design process.

The second step is to pick your family, die, package, and speed that you are targeting. IntelliFlow has a database that contains all of the dies, packages, and speeds that are available for Xilinx, including the new Virtex family. By allowing you to select the die, package, and speed individually, IntelliFlow protects you from picking an invalid combination.

The third step is to pick the type of simulation that you are planning on performing. If you have written a test fixture or testbench, you would select Verilog simulation or VHDL simulation respectively. On the other hand, if you are a user of Viewlogic's ViewSim™ gate-level simulator, you would choose gate-level simulation. At this point, IntelliFlow will take your schematic, create an EDIF netlist, and feed this EDIF netlist along with the underlying language blocks into FPGA *Express*™ for synthesis. This saves you from having to reformat your design manually and invoking multiple point tools.

With the design loaded in FPGA *Express*, you are ready to synthesize, place and route, and simulate. To perform these three steps, you just double-click on the Implement and Perform Timing Simulation tool. That's it. Behind the scenes, IntelliFlow will run FPGA *Express*. The netlist created by FPGA *Express* is fed into the Xilinx Alliance Series place and route tools. Since you have already chosen your simulation methodology, the correct Xilinx tools are run so that you get a VHDL, Verilog, or EDIF netlist that can be simulated. The netlist is translated, compiled, and loaded into the correct simulator.

When it is time to do board-level verification or layout, you just double-click on the Implement and Create Board Level Symbol Process. This process has many tools in common with the simulation process. Because of this, IntelliFlow knows not to re-run synthesis and place and route, which saves you time. Instead, it quickly takes the files that have already been created by the Xilinx Alliance tools and creates a symbol, on the fly. This symbol not only contains the pins that you have defined in your design, but power, ground, no connects, package information, and attributes that point to the underlying model for board-level simulation.

Although IntelliFlow handles the interactions of the various tools used in the design process behind the scenes, there is no loss of power for the Xilinx user. With IntelliFlow, you have *full* control over all of the tools used in the design flow. This includes the Viewlogic as well as the Xilinx tools. For example, if you are used to entering design constraints with the FPGA *Express* Timing and Constraint Editor, this editor is available to you via IntelliFlow. If you need to use the EPIC Design Editor or Xilinx Floorplanner, you can launch these from IntelliFlow. Individual settings for all of the Xilinx Alliance place and route tools are accessible. With context sensitive help, explanations for all of these tool settings are only a click away.

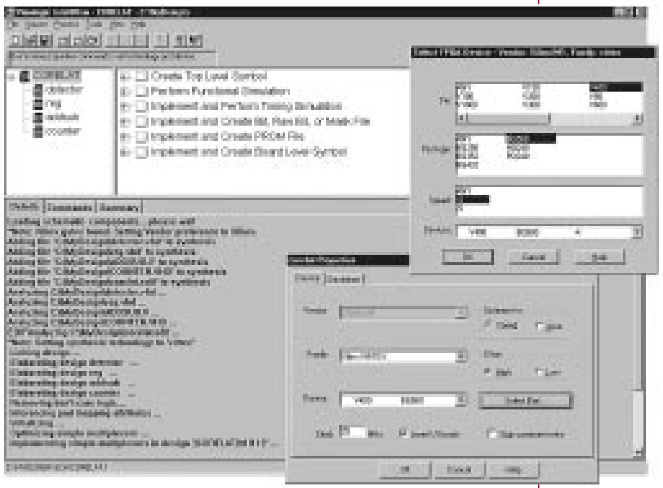

Create Top Level Symbol
Perform Functional Simulation
Implement and Perform Timing Simulation
Implement and Create Bit, Raw Bit, or Mask File
Implement and Create PROM File
Implement and Create Board Level Symbol

Figure 1: IntelliFlow Targeting the Xilinx Virtex Family

Conclusion

IntelliFlow provides a single graphical user interface for controlling design entry, functional simulation, synthesis, place and route, timing simulation, and board-level symbol creation. IntelliFlow performs data format changes automatically and runs the specific tools with the correct parameter settings for each of the Xilinx technologies. Full control over all of the Xilinx implementation tools is available through the IntelliFlow interface, giving you a combination of ease of use without loss of power. With IntelliFlow, you can focus on your design without having to deal with details of each tool used in the process or to learn new, complex design flows, reducing design errors and shortening your product development cycle.

For more information on the Viewlogic solution for FPGAs and other types of programmable devices, contact Viewlogic at 1-800-873-8439 or visit our website at www.viewlogic.com.

Foundation

by Kamal Koraitem,
Xilinx Product
Applications Group,
kamalk@xilinx.com

What are these new "Project Flow" types in Foundation 1.5, and when should I use one versus the other?

Foundation 1.5 contains two distinct project "flow" types: Schematic and HDL. The flow type is selected when the project is created. Once the project has been created, it is not possible to change the project flow type. If you wish to change the project flow type after the project has been created, you must create a new project.

SCHEMATIC FLOW:

The schematic flow is used for top-level schematic designs, as well as top-level ABEL designs. Top-level schematics may contain any number of lower-level HDL modules (VHDL, Verilog, ABEL), LogiBLOX, State Editor modules, or black-box netlist modules. This flow is very similar to the standard flow used in Foundation 1.4 and earlier.

HDL FLOW:

The HDL flow is used for top-level VHDL or Verilog designs. A project may include both VHDL and Verilog files. The design may also reference black-box modules, such as LogiBLOX components, schematic modules, or other netlists. These black-box modules will not be analyzed and optimized by the Express compiler, however, because the contents are unknown to Express.

In a future update of the Foundation Series software, the HDL flow will support schematic files, so they may be analyzed and optimized by Express. This will be particularly useful for creating top-level schematic designs which serve as a "block diagram" of a design containing several HDL blocks. The Express compiler will therefore have the ability to optimize these HDL blocks across their boundaries. In the initial Foundation Series 1.5 release, however, schematic files are only supported as black-boxes within an HDL flow project.

How do I migrate a design done with pre-Foundation 1.5 software to Foundation 1.5?

By default, when you open a project in Foundation 1.5 which was created with Foundation 1.4 (or earlier), the project type will remain as the old project type (such as XACT *Step* M1 or XACT*Step* v6). Once the project has been opened in Foundation 1.5, however, it is possible to change the project type to the new Foundation 1.5 project type. The Project Type dictates things such as what target devices are available, what the menus and GUIs look like, and what type of output netlist format is used. Changing to the Foundation Series 1.5 project type will give you access to the new Foundation 1.5 features including the embedded implementation tools and FPGA Express HDL compiler. It will also allow you to target devices supported only with the 1.5 release, such as SpartanXL, XC9500XL, XC4000XLA, and Virtex.

To change the project type to the new Foundation 1.5 project type, select File -> ProjectType from the Project Manager. Change the type to: **Foundation Series v1.5**.

If your design contains State Editor modules which are VHDL-based, please see Xilinx Solution 4402 (http://www.xilinx.com/techdocs/4402.htm) for more information about necessary modifications.

Note: If your design contains VHDL, and was done within the Foundation environment using the XVHDL (Metamor) compiler, changing the project type from XACTStep M1 to Foundation Series v1.5 will also cause the VHDL compiler used to change from Metamor (XVHDL) to

Synopsys FPGA Express. For more detailed information about this conversion process and issues involved, please see the Application Note entitled "Metamor to Express Conversion Guide" located in the Foundation Online Help System. From the Project Manager, select Help->Foundation Help Contents -> Application Notes.

For existing Foundation Express 1.4 designs, there is no migration path to bring the existing Express and M1 projects into the Foundation 1.5 project environment. This limitation is due to the new version/revision management employed by the integrated Foundation 1.5 tools. You will have two options:

1. Create a new HDL-type project in Foundation 1.5. Copy all source files (source HDL, NGO from LogiBLOX/Coregen, other XNF or EDIF files, .UCF file) to the new project directory and add the HDL files to the project. You will have to recreate all synthesis implementations as well as place and route revisions, as this information may not be imported from existing Express projects.

   If you plan on using the Foundation Simulation tool to perform a post-synthesis simulation on an HDL design containing LogiBLOX, you will need to create new LogiBLOX simulation models. Copy the .MOD file for each module into your project directory. Open the LogiBLOX GUI from the Project Manager (Tools -> Design Entry -> LogiBLOX module generator) and select the module using the Module Name pulldown. Click OK to create a new .NGC file (for implementation) and .ALR file (for functional simulation). This must be done for each module in the design.

2. Use the Foundation 1.5 tools in the same manner as the F1.4 tools were used. The GUIs for Foundation Express and the Xilinx Design Manager are available from Start -> Programs -> Xilinx Foundation Series -> Accessories. The Foundation Simulation flows and all project management issues are identical to the flow described in the Foundation Express Application Note Supplement that was shipped with Foundation 1.4.

Install

While installing Alliance 1.5 or Foundation 1.5, I received a "ComponentMoveData Error'? How can I workaround this?

This error may occur in a couple of different scenarios. The complete solution is documented under: http://www.xilinx.com/techdocs/4298.htm

Implementation

Starting any Xilinx GUI application from a workstation command line issues a message informing me that I have an old version of the registry program running on my machine. How did this happen?

The version 1.5 graphical applications may require you to update your registry information to a newer version compatible with the 1.5 software release, depending on if you had an older version running on your machine from a previous release. For more information on this, please reference: http://www.xilinx.com/techdocs/4024.htm

Heat Gun and Cold Spray
Your Best Debugging Tools

by Peter Alfke,
Director of Applications Engineering,
peter@xilinx.com

If you suspect timing problems of any kind, it's best to test your circuit under extreme operating conditions. All CMOS circuits get slower when hot, and faster when cold; the delay change is about 0.3% per degree centigrade. Testing at both temperature extremes can give you a good indication of your real timing margins.

Don't be too cautious in your temperature testing. Our devices tolerate well over +100°C (which is the boiling point of water, so if you wet your finger before touching the package, it will sizzle) and well under -40°C (where your finger will stick to the package). If you can keep your finger on the package for 10 seconds, then the surface temperature is below +65°C.

Two failure modes

Digital circuits can fail for two very different reasons:

- *The circuit is too slow* — As you know, a lack of circuit speed (excessive delays) limits the performance. It is therefore a common practice to check for proper design operation by increasing the clock frequency until the circuit fails. In large, complex systems it may be impossible or impractical to change the clock frequency.
- *The circuit is too fast* — Excessive circuit speed can also cause failure, but this cannot be tested by varying the clock rate. This type of failure can show up at any clock rate, and is more likely to affect slow designs that have not taken high speed issues, such as potential race conditions and hold times, into consideration.

"You should never release to production a digital design that fails at cold temperature or high Vcc!"

Luckily there are easy ways to manipulate the performance of any CMOS device, even after it has been soldered into the system. To verify timing margins, you can:

- Apply cold spray and cool the chip down as much as possible. At -40°C, the circuit delays are about 20% shorter than at room temperature, where they are about 20% shorter than at the worst-case test temperature of +85°C. That means a 36% decrease in delay, or a 56% increase in performance, compared to the worst-case spec. This is equivalent to several speed grade improvements.
- Raise the supply voltage to 10% above nominal. Raising the supply voltage increases performance, roughly proportional with the voltage.

By applying heat (up to +100°C) and lowering the supply voltage (to 10% below nominal) you can create the opposite effect, decreasing performance.

The beauty of the extreme temperature tests is that you can do them in a working system, on individual devices, without any expensive or destructive re-work.

A general test of design stability

Test your circuit both at high temperature and low Vcc, as well as at cold temperature and high Vcc:

- If your design fails at high temperatures and low Vcc, it is due to insufficient performance margins — the chip is too slow — and you should use a faster speed grade to improve your performance margin.
- If your design fails at cold temperatures and high Vcc, it is due to a poor asynchronous design that must be corrected, because the design has race conditions.

You should never release to production a digital design that fails at cold temperature or high Vcc!

Conclusion

If your design passes these hot/cold temperature and high/low Vcc tests, you are assured that it will work, in the field, under the worst operating conditions.

Customer Education Services Class Schedule: North America

1998 through June 1999

by Stacey Pinckert, Education Coordinator, Stacey.Pinckert@ xilinx.com

FPGA Tools Courses (2 days)

LOCATIONS	SEPT.	OCT.	NOV.	DEC.	JAN.	FEB.	MAR.	APRIL	MAY	JUNE
Huntsville, AL						8-9				
Phoenix, AZ			12-13							
San Jose, CA - Xilinx, Inc.	22-23	27-28	17-18	8-9	6-7	3-4	3-4, 31-4/1	28-29	25-26	23-24
Los Angeles, CA								28-29		
Irvine, CA									14-15	
San Diego, CA							17-18			
Toronto, Ontario, Canada			16-17							
Orlando, FL										7-8
Boston, MA			16-17			1-2			3-4	
Raleigh, NC		28-29								7-8
Fairfield, NJ					7-8					
Long Island, NY										
Houston, TX	17-18				20-21			28-29		
Dallas, TX		29-30				17-18		21-22		
Austin, TX			19-20						5-6	

Foundation Interface Courses (1 day)

LOCATIONS	SEPT.	OCT.	NOV.	DEC.	JAN.	FEB.	MAR.	APRIL	MAY	JUNE
San Jose, CA	21	26		7	5	2	2, 30	27	24	22

Foundation Express Courses (1 day)

LOCATIONS	SEPT.	OCT.	NOV.	DEC.	JAN.	FEB.	MAR.	APRIL	MAY	JUNE
Huntsville, AL						10				
Phoenix, AZ			11				9			
San Jose, CA		29	29	10	8	5	5	2, 30	27	25
Irvine, CA									13	
Los Angeles, CA								27		
San Diego, CA							16			
Montreal, Canada						25				
Orlando, FL										9
Atlanta, GA					22					
Boston, MA						3			5	
Raleigh, NC										9
Long Island, NY					6					
Austin, TX			18							
Dallas, TX		28								
Houston, TX								27		

Synthesis Courses (3 days)

COURSE TYPE	LOCATIONS	SEPT.	OCT.	NOV.	DEC.	JAN.	FEB.	MAR.	APRIL	MAY	JUNE
Verilog Methodology	San Jose, CA			11-13				8-10			
Verilog Methodology	Bethesda, MD						22-24				
Verilog Methodology	Columbia, MD			18-20							
Verilog Methodology	Portland, OR				2-4		17-19			19-21	
Verilog Methodology	Dallas, TX										16-18
VHDL Methodology	San Jose, CA					11-13				19-21	
VHDL Methodology	Orlando, FL				10-12						
VHDL Methodology	Portland, OR		26-28			20-22			14-16		

FPGA/CPLD Package Options and User I/O

	XC4000EX (5 Volt)		XC4000XLA (3.3 Volt)								XC4000XV (2.5 Volt)				Spartan (5V, XL 3.3V)					XC9500 (5 Volt) CPLDs						XC9500XL (3.3 Volt) CPLDs			
	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
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**
PLCC Packages																													
44																				34	34					34	34		
84															61	61					69	69							
CS Packages																													
48																				34						36	38		
144																												117	
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										
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			113	113												112	113	113										117	117
176			145	145																									
CBFP Packages																													
228	192								192																				
BGA Packages																													
225																													
256			192	205	205													192	205										
352	256	288			256	288	289	289	289	289	289	289												166	192				192
432		288				288	320	352	352	352	352	352	352	352															
560								352	384	448	432	432	432	432															
PGA Packages																													
299	256				256																								
411		288				288	320	352																					
475									384																				
559										448	448	448	448	448															

XC4000XLA Available Sept. (XC4013, XC4062 (HQ240)). Remainder available Q4. SpartanXL Available Q4

For information on other product families, see WebLINX at www.xilinx.com

FPGA/CPLD Product Selection Matrix

	Usable Gates (K)	Logic Gates	System Gates[1]	Logic Cells[2]	Total CLBs	Total Flip-Flops	Max. I/O	Max. RAM Bits	Config. Memory (Kb)	XC1700 Serial PROM Requirements
XC4000EX Family - 5 Volt										**0.5µm Triple Layer Metal Process**
XC4028EX	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 0.35µm Five Layer Metal Process**
XC4013XLA	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
XC4036XLA	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
XC4062XLA	40 - 130	62K	130K	5,472	2,304	5,376	384	73.8K	1,434	XC1702L
XC4085XLA	55 - 180	85K	180K	7,448	3,136	7,168	448	100.4K	1,925	XC1702L
XC4000XV Family - 2.5 Volt										**0.25µm 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 0.5µm & 0.35µm (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)

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*

	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

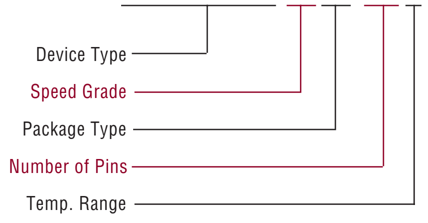
	Macrocells	Max. I/O	Pin-to-Pin Delay (ns)
XC9500XL Family - 3.3 Volt CPLDs			
XC9536XL	36	36	4
XC9572XL	72	72	5
XC95144XL	144	117	5
XC95288XL	288	192	6

	Density	PD8	SO8	VO8	PC20	SO20	VQ44	3 Volt	5 Volt
Serial PROM Package Options									
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
XC1701L	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	

	Slowest	Fastest	Comments
Speed Grade Options			
XC4000EX	-4	-2	
XC4000XLA	-3	-08	
XC4000XV	-2	-09	
Spartan	-3	-4	
SpartanXL	-4	-5	

Ordering Information for FPGAs / CPLDs

Example: XC4036XLA-08PQ160C



Corporate Headquarters
Xilinx, Inc.
2100 Logic Drive
San Jose, CA 95124
Tel: 408-559-7778
Fax: 408-559-7114

Europe
Xilinx, Ltd.
Benchmark House
203 Brooklands Road
Weybridge
Surrey KT14 0RH
United Kingdom
Tel: 44-1-932-349401
Fax: 44-1-932-349499

Japan
Xilinx, KK
Daini-Nagaoka Bldg. 2F
2-8-5, Hatchobori,
Chuo-ku, Tokyo 104
Japan
Tel: 81-3-3297-9191
Fax: 81-3-3297-9189

Hong Kong
Xilinx Asia Pacific
Unit 4312, Tower II
Metroplaza
Hing Fong Road
Kwai Fong, N.T.
Hong Kong
Tel: 852-2424-5200
Fax: 852-2494-7159

To subscribe to the Xilinx *XCell* Journal, cancel your *XCell* subscription, or order other Xilinx literature, send your request to: literature@xilinx.com

Please be sure to include:
1. Your full name and mailing address
2. Your title and your company name
3. Your request (*XCell* subscription)

You can view previous issues of *XCell*, on-line, at: www.xilinx.com/xcell/xcell.htm

So then I said "let s not use Xilinx FPGAs." Ya know, I miss being VP of Engineering.

In the high-tech systems market, you can t afford to make mistakes. Like choosing the wrong logic solution. Because late-to-market means missed opportunity, and we all know what that means. So, in an effort to ensure your company s future success (and yours), we introduced the XC4000XL Series.™ A high performance, high density FPGA family that delivers up to 265K system-level-gates at speeds beyond 100 Mhz. That makes them the fastest and densest devices in the business.

It s All About Making Choices.

The result? The XC4000XL has become the fastest growing programmable logic device family in history.

Protect your future. Choose the XC4000XL Series from Xilinx. Do it for yourself, because it s fun being VP of Engineering.

www.xilinx.com

XILINX®
The Programmable Logic Company℠

PN:XLQ498

XILINX®
2100 Logic Drive
San Jose, CA 95124-3450

First Class Presort
U.S. Postage
PAID
Permit No. 2196
San Jose, CA