

Figuring out reconfigurable logic

HIGH-DENSITY PROGRAMMABLE LOGIC IS BLURRING THE HISTORICAL DIVISION BETWEEN WHAT YOU DO IN HARDWARE AND WHAT YOU DO IN SOFTWARE. ALTHOUGH THE CHIPS HAVE SUPPORTED IN-SYSTEM UPDATES FOR MORE THAN A DECADE, THE REST OF THE PIECES REQUIRED TO SOLVE THE RECONFIGURABLE-SYSTEM PUZZLE ARE ONLY NOW MOVING INTO PLACE.

FLASH- AND SRAM-BASED PROGRAMMABLE-LOGIC devices' (PLDs) have some unique capabilities: logic you can alter to fix bugs or add features even after the system's in your customers' hands; hardware-accelerated algorithms that run tens or hundreds of times faster and at significantly lower power than software-based

equivalents; and designs much larger than the gate counts of the chips containing them would otherwise imply. Vendors of flash- and SRAM-based PLDs would love to see you use these capabilities because neither antifuse-based FPGAs nor ASICs can make the same claims.

But how achievable are these goals, even if the silicon theoretically supports them? Will reconfigurable logic eventually achieve widespread usage, or will it remain restricted to academic research, intriguing but low-volume niche applications, and government-funded top-secret military projects (see **sidebar** "And

At a glance 104
Hybrid chips and innovative design techniques 106
And now for something completely different..... 108
Ready-made hardware and software 112
For more information 114

now for something completely different”)? Before diving into the details of implementing in-system reprogramming, here’s a short list of real-life examples to whet your appetite:

- a laser printer that dynamically emulates a variety of printer command sets and requires only a low-end control processor;
- a satellite network that supports numerous hardware-based communications algorithms, including algorithms that no one imagined when the system was first operational;
- ATM equipment that dynamically tunes its characteristics for a given network’s data traffic;
- ultra-low-power cellular phones, usable throughout the world, that avoid obsolescence as industry standards evolve;
- audio- and image-filtering and pattern-recognition circuits that adapt their behavior to the surrounding environment; and
- neural net nodes that also learn, compression circuits that also decompress, and encryption engines that also decrypt, all without increasing the required device gate count.

WHAT’S IN A NAME?

Reconfigurable logic means different things to different people. On one end of the implementation spectrum, you may want to reprogram the entire device only once or a few times throughout the system’s life. On the other end, you may want to reconfigure only a portion of the device—to fine-tune filter coefficients, for example—but do it thousands of times per second. Somewhere in the middle of these two extremes is a system that periodically changes its personality in response to an external stimulus, such as a fighter plane that switches from standby to attack mode in response to a pilot’s action or other input.

Traditionally, you probably implemented repetitive, frequently occurring, and unchanging tasks in hardware (ASICs), where they executed faster and with lower total energy consumption (average power multiplied by the time it takes to complete the task). In contrast, you implemented unpredictable or occasionally executed tasks in software. You typically stored these tasks in ROM and

TODAY, THE DIVIDING LINE BETWEEN HARDWARE AND SOFTWARE ISN’T AS SOLID AS IT MIGHT APPEAR.

ran them on a processor instead of using costly ASIC gates. If the function was dynamic, your best bet was to both implement it in software and put it on a floppy disk or hard drive instead of in ROM.

Even today, the dividing line between hardware and software isn’t as solid as it might appear. A DSP is simply defined as a special-purpose μ P that hardware-accelerates commonly executed mathematics functions. Even general-purpose μ Ps are getting into the act; consider the various multimedia-optimized instruction sets or the matrix-multiply logic block that dominates the die size of Hitachi’s (www.hitachi.com) SH-DSP.

Flash memory was one of the first semiconductor technologies to significantly influence hardware-versus-software partitioning and placement. Initially used only during system development

and prototyping, flash memory has replaced ROM for all but the lowest density and most cost-sensitive systems in the production line because it is flexible and enables a rapid time to market. An increasing number of applications have taken the next step and offer in-field firmware upgrades; PC BIOS is just one example.

Programmable logic is following a path reminiscent of the one flash memory already forged. High-density CPLDs and FPGAs have moved beyond the niche status of ASIC prototype vehicles and glue-logic consolidators and are capturing production designs that might have previously used gate arrays. But, in most cases, designers still think of programmable logic only as a fast time-to-market ASIC, just as they first considered flash memory a factory-configurable ROM.

IT STARTS WITH THE SILICON

The most common in-system programmable-logic reconfiguration today centers on fixing bugs and adding features. This reconfiguration is a favorite of communications companies, who were also among the first to use flash memory’s field reprogrammability. In these cases, just about any flash- or SRAM-based device works, as long as you can easily integrate its reconfiguration signals and any special programming voltages with the rest of the system. Density-dependent full-device reconfiguration times for SRAM-based FPGAs are on the order of seconds to tens of seconds. Actel (www.actel.com) claims that you can erase and reprogram its A500K ProASIC flash-based FPGA in a few minutes.

The wider the device’s configuration port and the faster its operating frequency, the less likely that the device’s port will be the bottleneck for reconfiguration speed. You have to either ensure that the system can continue limited operation with the programmable logic in an unpredictable state or put the system in a reset condition during reconfiguration. Occasional updates also require no special design software beyond the compiler and fitter tools or place-and-route tools you already use (see sidebar “Ready-made hardware and software”).

When you first ship the system, you probably don’t know what bugs you might later need to fix, so how do you en-

continued on pg 108

AT A GLANCE

▷ Reconfigurable logic, like flash memory, brings added partitioning and placement options to system architectures. However, more degrees of freedom may also create undesirable excessive complexity.

▷ Building logic, routing, and timing head room into your design up-front maximizes your probability of update success once the design is in the field.

▷ Robust silicon, design tools, and system-level software (the latter two frequently lagging behind the former) are all necessary to make programmable-logic-based algorithm acceleration a reality.

▷ μ P-plus-programmable-logic hybrids, along with new design-entry approaches, transform the traditional hardware-versus-software dividing line.

HYBRID CHIPS AND INNOVATIVE DESIGN TECHNIQUES

PROGRAMMABLE LOGIC MAY BE IDEAL for arithmetic calculations and datapath functions, but many control algorithms require a state machine that is too complex to be practical. A software-con-

trolled μ P is a more feasible option. In this era of systems on chips, therefore, it probably doesn't surprise you that single-chip processors plus programmable-logic arrays are beginning to appear.

Ironically, the first vendors of these products are—at least for the moment—not marketing them for hardware-acceleration applications. Triscend's E5 places an 8032 embedded controller core in an ASIC, along with a varying amount of embedded memory and programmable logic. FPGA-like look-up-table- and register-based logic blocks make up the programmable logic. Triscend uses the programmable logic for the 8032 peripheral functions, which lets the vendor ship a generic piece of silicon that you can customize to your desired specifications. The architecture also incorporates numerous flexible interconnections between the ASIC and programmable-logic partitions. Although the software requires additional capabilities to support coprocessor functions, the silicon seems ready-made for such applications. The vendor's upcoming ARM-based 32-bit CPUs will give the product line a performance boost, and companies such as Atmel and Lucent are also hinting at combo chips.

Both Morphics Technology and QuickSilver Technology are targeting the mobile-communications space, but the two companies are taking different approaches to the programmable logic. QuickSilver,

whose management team includes veterans from Xilinx, claims that both ASICs and mainstream ASIC-replacement FPGAs fall short in hardware-accelerated wireless applications. At 25 to 35% efficiency, ASICs and FPGAs are more optimized for devoting the total on-chip transistor count to the algorithm that the system is executing than DSPs at 5 to 10% efficiency. However, QuickSilver believes that an optimized reconfigurable engine, by eliminating unnecessary on-chip logic, memory, and configuration circuits, could achieve a maximum efficiency of 70% and would also require a configuration bit stream significantly smaller than that of a mainstream FPGA. Chameleon Systems (www.chameleonsystems.com) alludes to similar plans but is targeting the communications infrastructure, not the handset. Adaptive Silicon (www.adaptivesilicon.com), Malleable Technologies (www.malleable.com), and Silicon Spice (www.siliconspice.com) are also pursuing the processor-plus-programmable-logic vision; stay tuned for their future product plans.

A brand-new programmable-logic architecture requires brand-new software tools, however, and silicon companies tend to stumble in this area time and time again. Morphics Technology is not interested in going down that path. Instead, the company is exploring with several ven-

dors the idea of incorporating existing programmable logic as an embedded core into a larger device that also includes programmable interconnect and a control processor. Stephen Wasson, Morphics' director of reconfigurable logic, points out that although ASICs have the best combination of power, price, and performance, these capabilities come at the expense of upgradability, flexibility, and time to market. The secret is to keep the amount of programmable logic as small as possible but not so small as to inhibit flexibility or force the algorithm to run on an even slower, more expensive, and more power-hungry DSP.

Wasson comments that the company's hardware-versus-software and ASIC-versus-FPGA partitioning efforts often begin by coding algorithms in software, then identifying the commonalities between them, the sections that Morphics doesn't anticipate will change, and the sections on which the system spends most of its time. These areas are all prime candidates for moving into hardware. Instead of rewriting the algorithms in an HDL or schematic form, however, wouldn't it be simpler to just directly compile them into logic gates from their C or Java form? Such an approach would offer the added benefit of a design-entry format that contains the time-based (temporal) references that reconfigurable-logic designs inherently assume and that traditional hardware design-entry approaches lack.

In fact, C Level Design, Embedded Solutions Ltd, and Frontier Design all sell compilers that translate C code into VHDL or Verilog. These

tools support only a subset of C, though, because, for example, concepts such as pointers make no sense in hardware. The tools also need to figure out when to leave a series of consecutive software instructions as a staged pipeline or state machine or when to extract parallelism from the serial code into one combinatorial logic structure. For several years and with limited success, TSI Telsys (www.tsi-telsys.com) designed satellite-network systems using C and its proprietary compiler; the company has now shifted its focus to Java. TSI Telsys is selling its tools in the open market under the LavaLogic name. According to Vice President Toby Bennett, Java offers a simpler memory model and requires no rewriting or additional libraries. Also, because Java is designed to run on top of other software, it's already virtualized, making hardware and software partitioning even easier. LavaLogic's ideas for the future include bypassing the interim HDL step and directly generating a programmable-logic netlist from the Java source code.

This abstraction is both a blessing and a curse. The higher the level of the design, the faster functional simulations complete. C and Java also open the hardware-design door to a broad population of engineers—not just engineers who know HDLs or schematics. However, HDLs tend to produce less efficient designs than schematics, and even higher abstraction languages will further exacerbate this situation. For these reasons and others, engineers such as Wasson indicate that, at least for the moment, they'll be blending multiple design-entry methods.

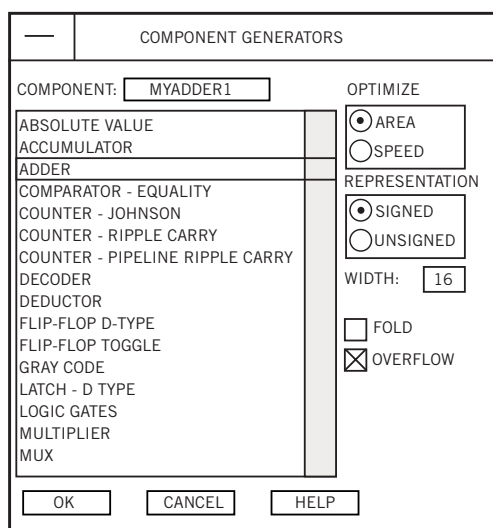
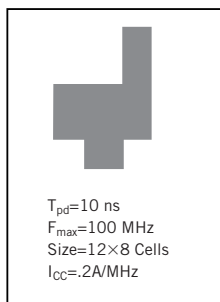


Figure 1



Parameterized component generators create logic functions with known placement, performance, and power consumption (courtesy Atmel).

sure that you can make the necessary logic changes and still hold pinout and timing? Sure, you can define target parameters via user constraints, but what ensures that the tools won't just give up? In a word, nothing. Following a few general guidelines, though, can maximize your probability of reconfiguring the chip while maintaining compatibility with the system containing it.

Employ synchronous design techniques wherever possible and base the design on maximum times (such as propagation speeds or clock-to-output specifications), not minimum delays. Also, leave a few nanoseconds or megahertz of head room in the design. For example, if the system has a 33-MHz operating frequency, don't be content if your initial compilation attempt runs at 33.1 MHz. Pick a faster part, or spend more time optimizing the design. If subsequent signal rerouting alters the path delays through the chip, you'll still meet the target system parameters.

Not only how much timing head room you put in the design, but also the amount of spare logic and routing resources in the chip after you fit your initial design revision inside it determine your design's ability to hold pinout and timing. Report files frequently list the number of various logic structures the design uses, but they rarely give you statistics on how many of the routing resources you might have remaining. Iron-

ically, routing—specifically, fast-enough routing—not logic, is frequently the scarce resource. The tools might output a graphical display of the interconnect as part of their floorplanning utilities, but the resultant wiring looks like a rat's nest on your computer monitor, so the display is of limited value.

PLD vendors also benchmark and fine-tune their new devices and tools to ensure the highest probability of pinout and timing locking throughout revisions. Altera (www.altera.com) uses its "90/90" rule. Over the years, the company has compiled a large set of reference designs. It first fits each of these designs into a given part and allows the tool to pick the pinout. The company then randomly scrambles the pinout and attempts to again fit the design. Altera claims it isn't satisfied until at least 90% of the designs fit using 90% of the revised pinouts. Although, in this case, the company is modifying the design pinout

and not the design itself, the impact on routing is similar.

ALGORITHM ACCELERATION

If you use a PLD as a coprocessor, you are, in some respects, taking a step backward from using an ASIC. Even the fastest FPGA is slower and burns more power than an ASIC, especially in designs that aren't heavily optimized. By itself, the fact that a fixed number of programmable gates can implement many logic circuits in a time-switched fashion isn't a compelling selling point, either. In the same silicon area that one programmable-logic-based circuit consumes, you can fit dozens of similar ASIC-implemented circuits. But add the ability to fine-tune in-system and fix bugs in the circuits' operation, and programmable logic's strengths become more apparent.

Granted, there are few tasks that a high-speed μ P, DSP, or multiprocessor configuration can't handle through software. Achieving increased device performance, though, requires increasingly esoteric architectures. For example, take a look at Intel's (www.intel.com) Merced IA-64 CPU or any of the very-long-instruction-word DSPs. Easily, consistently, and cost-effectively translating the specifications in device data sheets into real-life performance gains is also hit or miss. Clearly, evolutionary processor advancements are producing increasingly limited returns. Particularly in battery-operated, high-volume consumer systems in which performance, price, and power are important, some system designers are claiming that the traditional software-based approach is no longer feasible, opening the door to a more revolutionary hardware alternative.

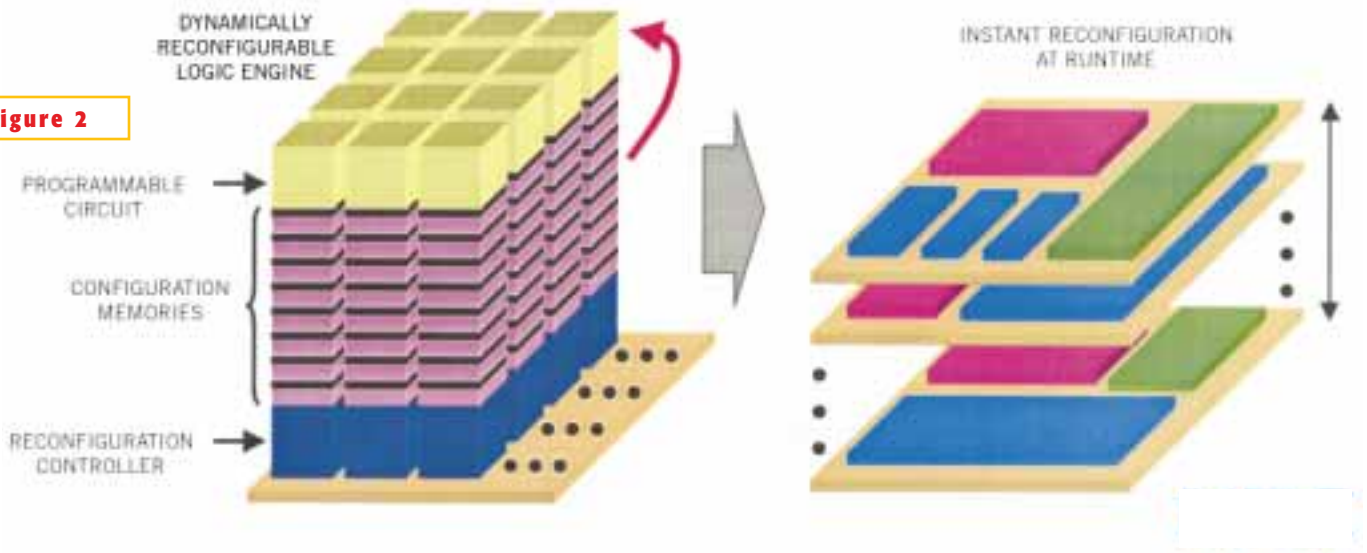
When comparing software algorithms with their hardware equivalents in CPLDs or FPGAs, make sure that you don't forget to evaluate parameters other than

AND NOW FOR SOMETHING COMPLETELY DIFFERENT

The version of this article on *EDN Access* (www.ednmag.com/ednmag/reg/1999/080599/16df2.htm) contains an additional sidebar that you might find interesting (or at least entertaining). In it, I cover some of today's more esoteric applications for reconfigurable logic, including evolutionary

algorithm-development techniques that would make Darwin proud, a high-speed self-repairing supercomputer, and, yes, even a robotic cat. I also point you toward academic and other World Wide Web-resident reconfigurable-logic resources.

Figure 2



Experimental FPGA architectures include multiple configuration planes, a costly but effective means of achieving high task-switching performance (courtesy NEC).

chip price, performance, and energy consumption. Ask yourself whether reconfiguring the portion of the PLD necessary to implement a new function will take longer than the task-switch delay of a μP . Find out what response rate the system requires. Compare the total size of the configuration memory you need to implement all of the necessary hardware-acceleration functions with the equivalent DSP or CPU firmware size. Bit-stream compression—although it can reduce the required configuration memory density—requires decompression hardware within the PLD. And, finally, brainstorm about whether a nonstandard data width—an option generally unavailable to you with a DSP but easy to achieve in a hardware-accelerated circuit—would benefit your design.

The more rapid the required reconfiguration rate and the smaller the percentage of the total device that you rewrite at a given time, the more attractive a partially reconfigurable FPGA becomes. Several device-family options exist, but the software support is often much harder for vendors to deliver than the silicon. Atmel offers arguably the most comprehensive partially reconfigurable silicon and design-software platform, which is based on the company's AT40K and earlier AT6000 FPGAs. These symmetrical, register-rich devices have their roots in

now-defunct Concurrent Logic's architecture, which also formed the foundation of National Semiconductor's (www.national.com) CLAY product line (which never made it to production). The architecture was also the predecessor of some of the emerging processor-plus-FPGA single-chip devices (see sidebar "Hybrid chips and innovative design techniques"). In developing AT40K, Atmel beefed up the on-chip routing of AT6000 and migrated from multiplexers to dual look-up tables in each logic cell. Atmel supplements the look-up tables with dedicated AND gates that target matrix-multiplication functions common in DSP applications.

AT40K also contains small discrete SRAM blocks at the corners of each 16-cell matrix. Atmel intends for adjoining logic to use these SRAM arrays to store quickly accessed parameters. Atmel's architectures support reconfiguration even on a fine-grained, cell-by-cell level. Logic and routing configuration bits whose values don't change also don't have glitches during reconfiguration. This feature is important in eliminating the disruption of downstream logic, whose operation might transition into unintended modes if circuit inputs spuriously toggle or go into a tristate condition. You can also treat AT40K as a memory-mapped I/O peripheral using the devices' Config-

uration Mode 4 (also called the Synchronous RAM mode), in which you access logic structures using a 24-bit address and an 8- or 16-bit data string and communicate with the FPGA over an 8-bit, 33-MHz bus.

Atmel's design-software support is equally comprehensive. If you've created a circuit in HDL or a schematic that you want to swap into the device, Atmel's Integrated Development System has a manual floorplanning capability that can place the logic structure, then generate a corresponding partial bit stream. Alternatively, if you need to generate only new constants, such as filter parameters, the vendor's QuickChange tool outputs a bit stream without recreating the netlist. Atmel also supplies a variety of automatic component generators that creates highly flexible, location-specific implementations of filters, convolvers, edge detectors, and other circuits (Figure 1). With any partially reconfigurable design, comprehending fixed-location device resources, such as carry chains, global clock trees, PLLs, and memory blocks, is particularly challenging. Avoiding the use of these structures makes the circuit more placement-portable but decreases silicon efficiency, power consumption, and performance.

With the Virtex family, Xilinx carries forward portions of the partial reconfig-

urability of the company's XC6200 FPGAs, a long-time favorite of academics. Each column within a device comprises a density-dependent number of configuration-element chains, or "frames," and you can reconfigure the device on a frame-by-frame basis. Xilinx guarantees that unchanging configuration elements won't toggle during frame rewrite, so bit-resolution reconfiguration is possible, including the reconfiguration of the embedded Block SelectRAM. Unfortunately, Xilinx's Alliance and Foundation tools don't yet support partial reconfiguration, and—similar to vendors of all partially reconfigurable FPGAs other than the XC6200—Xilinx hasn't publicly released the specifications for its Virtex configuration bit stream. Such "bit-twiddling," which is analogous to software programming in machine language, would be an inefficient way to design with the chip, anyway.

Xilinx is, however, moving forward from a system-software standpoint. The Java-based Jbits application-programming interface works with an assumed

JBITS OPENS THE DOOR TO INTERNET-BASED DESIGNING AND UPDATING.

Java Virtual Machine running somewhere in the system. Jbits now controls full FPGA reconfiguration. In the future, it will be capable of partial reconfiguration. Along with the remaining products from the company's Xilinx Online initiative, such as the Chip Scope design and debugging tool, Jbits opens the door to Internet-based designing and updating. JEDEC is developing the STAPL protocol, which originated with Altera's Jam. The STAPL protocol and Lattice's ispVM are alternative virtual-machine-based reconfiguration approaches without the Internet "hooks."

DynaChip's DY6000 FPGAs also support partial reconfiguration, down to logic-block resolution. The company's

place-and-route software doesn't currently provide straightforward access to the silicon capabilities, but the vendor has developed some creative temporary ways to work around this situation. DynaChip can create an artificial chip description with the exact number of logic blocks and orientation that the partially reconfigured circuit requires. This circuit connects most easily to the rest of the chip through I/O pins, but the company also assists customers in generating a bit stream that maps directly to the desired internal routing tracks.

NEC previewed a partially reconfigurable FPGA at January's International Solid-State Circuits Conference in San Francisco, but the company is still developing software and isn't promising devices any sooner than two years from now. NEC's device, like an experimental device Xilinx showed at 1997's IEEE Field-Programmable Custom Computing Machines conference (Napa, CA), contains multiple internal configuration mapping planes. Once you load these planes, the system can time-switch in a

READY-MADE HARDWARE AND SOFTWARE

If you'd like to further investigate the potential of reconfigurable logic but don't have a lot of time or money, or if you're sold on the FPGA-as-coprocessor approach but need to put a design into production faster than you could develop it yourself, a number of options are at your finger tips. Altera (www.altera.com), Atmel, and Dyna-

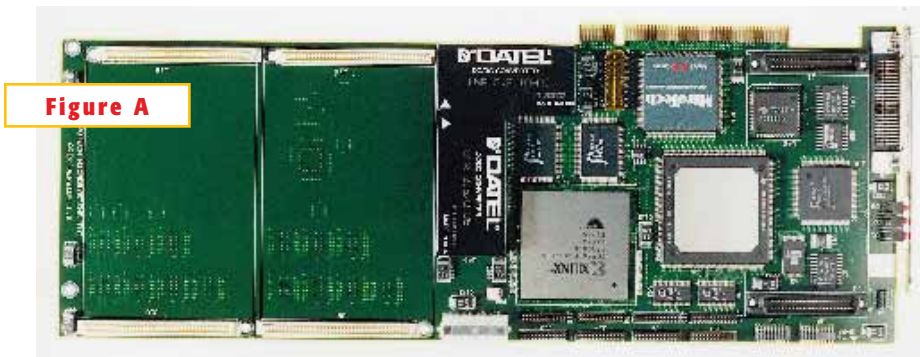
Chip all sell evaluation kits containing both reference boards and software. Neither Altera's Flex8000 chips nor its Flex10K chips are partially configurable, but the vendor simulates this capability through the multiple chips that each board contains.

Several third parties also sell evaluation kits, including Associated Professional Systems and

Virtual Computer Corp. Virtual Computer's Virtual Workbench supports Xilinx's Virtex architecture; the company also sells boards containing the XC6200. If, on the other hand, you're interested in a production-worthy FPGA-based DSP acceleration board, consider going through Annapolis Micro Systems and MiroTech Microsys-

tems. Annapolis' product line includes boards with PCI, PCMCIA, and VME buses, and MiroTech's modules come in PCI, PMC, and Texas Instruments (www.ti.com) Module form factors with add-in mezzanine-board capabilities.

You interface your system software to the boards through application-programming interfaces that call MiroTech-developed function libraries. The boards include Texas Instruments DSPs as control and interface processors and may contain multiple FPGAs (Figure A). Although MiroTech President Pierre Popovic notes that it is still best to perform full-precision floating-point operations in software today, increasing PLD gate counts are enabling some hardware-based half-precision computations.



Interested in hardware-accelerating your DSP algorithms, but no time to design and debug your own platform? Buy one that's already developed (courtesy MiroTech Microsystems).

FOR MORE INFORMATION...

For more information on products such as those discussed in this article, circle the appropriate numbers on the Information Retrieval Service card or use EDN's InfoAccess service. When you contact any of the following manufacturers directly, please let them know you read about their products in EDN.

BOARDS

Annapolis Micro Systems Inc

1-410-841-2514
www.annapmicro.com
Circle No. 382

Associated Professional Systems

1-410-569-5897
www.associatedpro.com
Circle No. 383

MiroTech Microsystems

1-514-744-6476
www.mirotech.com
Circle No. 384

Virtual Computer Corp

1-818-342-8294
www.vcc.com
Circle No. 385

CHIPS

Atmel Corp

1-408-441-0311
www.atmel.com
Circle No. 386

DynaChip Corp

1-408-481-3100
www.dyna.com
Circle No. 387

I-Cube

1-408-341-1888
www.icube.com
Circle No. 388

Lattice Semiconductor Corp

1-503-268-8000
www.lattice.com
Circle No. 389

Lucent Technologies

1-610-712-4331
www.lucent.com
Circle No. 390

Morphics Technology

1-408-863-6100
www.morphics.com
Circle No. 391

NEC Electronics Inc

1-408-986-1020
www.nec.com
Circle No. 392

QuickSilver Technology

1-408-558-2709
www.quicksilvertech.com
Circle No. 393

Triscend Corp

1-650-968-8668
www.triscend.com
Circle No. 394

Xilinx Inc

1-408-559-7778
www.xilinx.com
Circle No. 395

SOFTWARE

C Level Design

1-408-369-0555
www.cleveldesign.com
Circle No. 396

Embedded Solutions Ltd

+011-44-0-1235-863656
www.embeddedsol.com
Circle No. 397

Frontier Design Inc

1-510-445-8500
www.frontierd.com
Circle No. 398

LavaLogic

1-410-872-3900
www.lavalogic.com
Circle No. 399

SUPER CIRCLE NUMBER

For more information on the products available from all of the vendors listed in this box, circle one number on the reader service card. Circle No. 400

matter of nanoseconds (Figure 2).

Both Lucent Technologies' Orca2 and Orca3 architectures are partially reconfigurable on a bit-by-bit granularity analogous to Xilinx's Virtex, and the built-in μ P interface in Orca3 targets configurations in which the FPGA acts as a coprocessor. Lucent plans to include partial-reconfiguration support in the next release of Orca Foundry, currently scheduled for early next year. Lucent also points out that its OR3TP12 device, which includes an ASIC-based PCI core, could—along with additional user-designed logic—support partial reconfiguration over the PCI bus.

What if your design spans multiple chips, and you'd like to be able to reconfigure several of them in-system? Unfortunately, your probability of pin- and performance-locking success diminishes as the number of chips increases. But as long as you don't mind complicating the system with yet another reconfiguration variable (along

with software and logic), consider the programmable interconnect chips that I-Cube and Lattice Semiconductor offer.

Lattice has been shipping its 5V ispGDX devices for roughly two years and has recently begun offering 3.3V variants. Propagation delays through the interconnect are as fast as 5 nsec. Because these devices aren't fully populated cross-point switches, you may be unable to connect one pin to another pin in all device configurations. However, they do offer a level of flexibility beyond the options that each PLD's routing structures contain. □

REFERENCES

1. Dipert, Brian, "Innovation ignores economy: the 1999 ISSCC," *EDN*, March 18, 1999, pg 11.
2. Dipert, Brian, "Focusing on chip programming," *EDN*, Jan 7, 1999, pg 58.
3. Maxfield, Clive, "Logic that mutates while-u-wait," *EDN*, Nov 7, 1996, pg 137.

4. Conner, Doug, "Fast and flexible: FIR filters in reconfigurable logic," *EDN*, July 4, 1996, pg 65.

5. Conner, Doug, "Reconfigurable logic: built-in adaptability," *EDN*, June 20, 1996, pg 42.

6. Levy, Markus, "DSP design tools target FPGAs," *EDN*, June 20, 1996, pg 75.

7. Conner, Doug, "Reconfigurable logic: hardware speed with software flexibility," *EDN*, March 28, 1996, pg 52.

8. Application note "Implementing Cache Logic with FPGAs," Atmel Corp, 1997.

ACKNOWLEDGMENTS

I'd especially like to recognize (or is that "reconfignize"?) the following individuals for sharing their observations on reconfigurable logic and its system-level implementation: Wendy Lockhart from Atmel, Pierre Popovic from MiroTech Microsystems, Stephen Wasson from Morphics Technology, Randy Harr from Synopsys, and Toby Bennett from TSI Telesys' LavaLogic subsidiary.



You can reach Technical Editor Brian Dipert at 1-916-454-5242, fax 1-916-454-5101, e-mail bdipert@pacbell.net.