

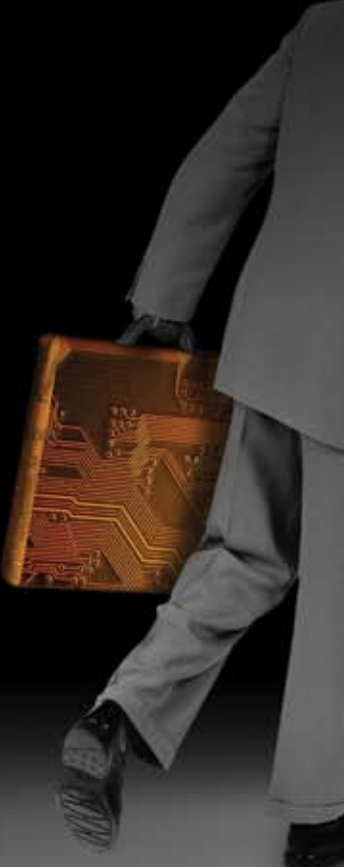
LOW-COST KITS: THE NEW FPGA- DESIGNER TREND



DOMINATING COMMUNICATIONS AND COMPLEX INDUSTRIAL APPLICATIONS, FPGA MAKERS ARE LOOKING TOWARD DESIGN WINS IN LOWER END AUTOMOTIVE AND CONSUMER DEVICES. WITH TODAY'S DEVICES BLURRING THE DISTINCTIONS BETWEEN CPLDs AND FPGAs, LOW-COST DESIGN KITS HELP FIRST-TIME USERS TACKLE SOPHISTICATED SYSTEMS.

BY DAVID MARSH • CONTRIBUTING TECHNICAL EDITOR





In the 1970s, low-density and power-hungry TTL packages filled every digital designer's schematics. But, before long, silicon architect Ron Cline at Signetics developed the world's first PLA (programmable-logic array)—a device that could pack multiple TTL functions on a single chip by configuring fuses between its AND-OR gate planes. Since that time, programmable logic has evolved to the point that today's FPGAs (field-programmable gate arrays) routinely furnish as many as 10 million gates to address complex applications, notably within the communications infrastructure. And there's no sign of any slowdown in this burgeoning market; market-research company In-Stat estimates that the \$1.895 billion 2005 FPGA market will balloon to \$2.7567 billion by 2010 (**Reference 1**).

Furthermore, the company believes that the largest user groups will continue to be communications and industrial sectors, growing from 73.8% in 2005 to 76.8% by 2010.

Given this focus, it's unsurprising that many designers perceive programmable-logic devices as difficult to use, expensive, and power hungry. But at the lower end of the spectrum, the silicon vendors are making a concerted effort to secure design wins within the automotive and consumer markets. As *EDN* recently reported, part of this drive consists of producing lower-gate-count devices with the lowest possible power consumption to enable use in portable electronics and "always-on" automotive applications (**Reference 2**). But what does it take for a potential user to get started using low-end programmable silicon in terms of approachability and ease of use, future extensibility to suit target applications, and cost?

With these considerations in mind, *EDN* assembled a selection of low-cost development kits that support in-system-reprogrammable devices. Such devices are the natural choices for prototyping, as well as for production applications, when the ability to perform field upgrades can provide a competitive advantage. Given volume pricing of around \$1.50 for entry-level devices, cost is no longer a barrier to using reprogrammable logic—and evaluation kits are available for as little as \$49. So

what do you get for your money and how easy is it for a novice to use?

300,000 GATES

Costing approximately \$285, Actel's ProASIC Plus starter kit contains a 134×125-mm evaluation board that carries the company's 300,000-equivalent-gate APA300 FPGA surrounded by a mass of headers that make available every device pin. Various jumpers enable, for instance, an array of eight user LEDs, and five pushbutton switches provide user input. A high-density, 26-pin header connects to the accompanying FlashPro Lite programmer, which a parallel-port cable links to a host PC. A universal-input ac-line adapter provides 9V dc that a pair of TPS776xx voltage regulators downconvert to 2.5 and 3.3V levels. The chip's core requires 2.5V, and jumpers select 2.5 or 3.3V for the I/O blocks. The starter-kit CD includes a user's guide and resources such as schematics and design files that complement the printed *User's Guide & Tutorial*. Two further CDs furnish the Libero IDE (integrated design environment) and Logic Navigator debugging software. Usefully, the board's FPGA comes preprogrammed with a test program that confirms hardware operation.

Like other ProASIC Plus family members, the APA300 is built in an "instant-on," reprogrammable flash technology that dispenses with external configuration memories and their support logic. The chip's architecture us-



es a “sea-of-tiles” fabric that’s surrounded by embedded dual-port SRAM modules and a package-dependent number of I/O pins. With the exception of a three-input XOR gate, it’s possible to configure each of the APA300’s 8192 tiles as a three-input, single-output logic function—such as a D-type flip-flop—and connect it with others using the chip’s routing resources (**Figure 1**). Available routing options are local, long line, very long line, and global. Of these, local is the fastest and allows a tile’s output to connect to every input of its eight neighbors in a nine-tile array. The long-line resources run horizontally and vertically across the device and can span one, two, or four tiles. The very-long-line resources similarly cover the device in a grid pattern to suit high-fan-out nets, and the global resources typically suit clock and reset-line distribution. Other distinguishing features include programmable Schmitt-trigger inputs on each I/O pin, together with two clock-conditioning blocks that comprise an analog phase-lock loop, delay lines, a quadrature phase-shifter, and clock multipliers and dividers. A clock-tree network built from spines and ribs that reach every tile within their respective regions permits the APA300 to efficiently distribute as many as 32 clocks.

Promising start-to-finish design-flow guidance and control for novice and experienced users, Libero’s default installation occupies slightly more than 1 Gbyte of disk space. The software runs under Windows 2000 Pro SP4/XP Pro SP2, Sun’s Solaris, or Red Hat’s Linux. Following installation, Web-site registration results in the arrival of an e-mail that contains a license file for the Gold IDE edition that node-locks the installation to the PC’s hard disk. Crucially, this file also enables the Synplicity Identify AE (Actel edition) software components. Running the IDE then checks the Web for updates—in this case, offering a 443-Mbyte self-extracting archive that contains Version 7.2, plus 56 Mbytes more of Service Pack 2. Because the download is a complete copy of the IDE that requires you to uninstall the original or install it in another directory, you may wish to ignore the CD in favor of the Web package. Download and read the IDE’s *Quick Start Guide* while the procedure completes.

AT A GLANCE

- ▶ Programmable devices target automotive and consumer markets.
- ▶ Midrange devices blur traditional CPLD/FPGA (complex-programmable-logic-device/field-programmable-gate-array) definitions.
- ▶ Low-cost development kits promise to speed design.
- ▶ Accessibility is the key to complex software-tool environments.

The tutorial starts off with compiling an AND gate in VHDL (very high-speed-IC hardware-description language), neatly demonstrating the step-by-step discipline that Libero enforces (**Figure 2**). Users who are new to VHDL and its popular alternative, Verilog, can find a great deal of introductory material on the Web. (See **references 3 and 4** for Verilog-specific links and similar VHDL resources, respectively.) The power of these design-entry tools instantly becomes clear when you’re attempting something as simple as a 16-bit counter—although the schematic-entry route traditionally requires you to replicate each flip-flop and all of its connections, modifying a few parameters within a standard VHDL text file allows you to build a counter of arbitrary length.

Because it’s easy to get lost within some environments, Libero’s sequential procedure is a boon to new users without being too prescriptive for regular use. Sadly for users, a typo in the library package reference causes your very first VHDL compilation to fail: The “iee” in the third line should read “ieee.” Fix this glitch, and

the next step runs SynaptiCAD’s WaveFormer Lite, a subset of the \$2500 WaveFormer Pro tool that generates test-stimulus data from graphical inputs. You can also create the testbench data directly within Libero’s HDL (hardware-description-language) editor. Right-clicking the `andgate.vhd` sample file offers options that differ from the tutorial—which was written around Libero Version 2.3 SP2—yet this step and later ones that differ are easy to transpose. Selecting Run Presynthesis Simulation offers the opportunity to associate the stimulus file with `andgate.vhd`; accept this option, and the Actel version of Mentor Graphics’ ModelSim launches to compile the stimulus data. Among the mass of information that floods the screen is a waveform window that displays the A and B inputs that you created in WaveFormer Lite, as well as the simulator’s logical output of these two signals.

Having verified that the design works correctly, you next generate an EDIF (electronic-design-interchange-format) netlist file using Synplify. Libero seamlessly translates the resulting output file into a VHDL netlist, which then appears within the IDE’s file-manager window. If any errors occur, you can edit the file within Synplify, which back-annotates changes to Libero. Running a post-synthesis simulation in ModelSim now displays the output waveform, including propagation delays. To implement the design, run Designer, select the device and its package, and click Compile. When this button turns green—signifying successful compilation—assign the AND gate’s pins by drag-and-drop within Pin Editor, run Layout and Back-An-

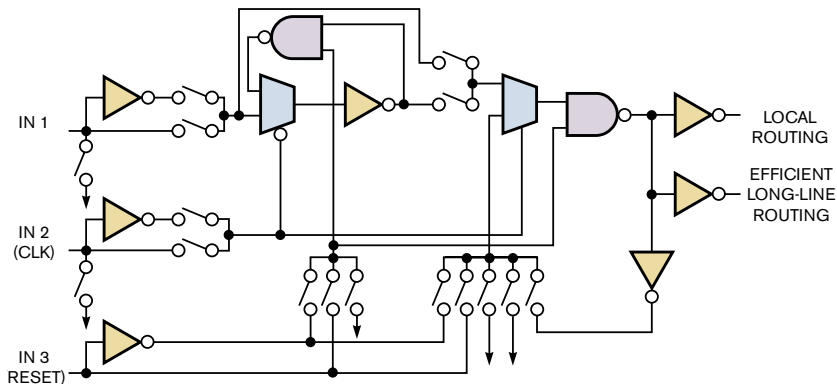


Figure 1 Except for a three-input XOR gate, each tile in Actel’s ProASIC devices can accommodate any three-input, one-output function.



notate, and the IDE will save a file called `andgate.ahb`, whose timing characteristics you can again simulate within ModelSim before programming the device. The Programming File button within Designer produces a STAPL (standard-test-and-programming-language) file that it deposits within the Implementation Files section of the IDE's file-manager window. In the meantime, the IDE's design-flow window tracks each of these process steps and leaves you ready to program the device.

The details of the FlashPro 4.2 programming software differ somewhat from the printed description that relates to an earlier revision, but the steps are obvious to follow. Pressing switches 1 and 2 then illuminates LED 1 and signals correct compilation and programming of this VHDL textual example. Users wishing to explore schematic entry using ViewDraw then have to use some initiative; there is a limited amount of tutorial material on the Web site, but the main information sources lie within the user guides that the ViewDraw directory contains. It appears that this tool borrows from the Innoveda eProduct Designer suite from Mentor Graphics, which a number of documents describe—notably the “old” and current versions of the ViewDraw manual, with the former usefully including much tutorial material. Even without these resources, ViewDraw is sufficiently intuitive that it's easy to generate another gate example and redirect its output to another LED, by which time the design-flow process seems familiar.

Dennis Kish, senior vice president of sales and marketing at Actel, advises new users to consider the ProASIC3/E product family that the company introduced last year. This family spans 30,000 to 3 million system gates, packs approximately 25% more logic, and operates about twice as fast as similar ProASIC Plus devices. It includes features such as hot-swappable I/O structures that greatly ease complex board-management applications. Kish also points to the company's Fusion PSCs (programmable-system chips), which integrate analog functions such as ADCs to particularly suit system-management roles. Kish says that Actel will offer for sampling its Igloo family, which targets portable electronics

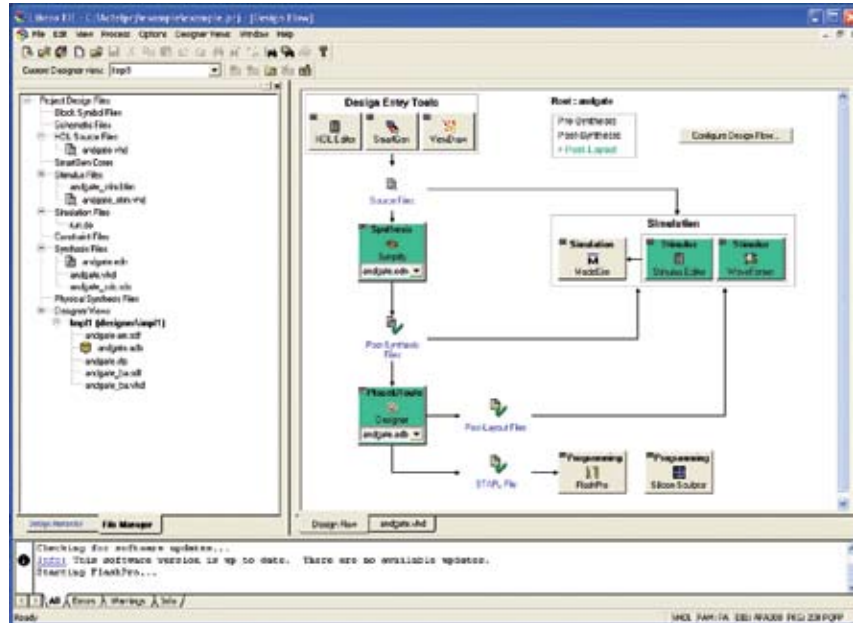


Figure 2 Libero's IDE guides users through every design and implementation step.

and low power consumption, in the first quarter of 2007. This new family is compatible with ProASIC3/E but includes the FlashFreeze power-save input pin, which effectively tristates the device and retains all state information. This approach reduces the power consumption for the smallest AGL030 device to approximately 5 μ W.

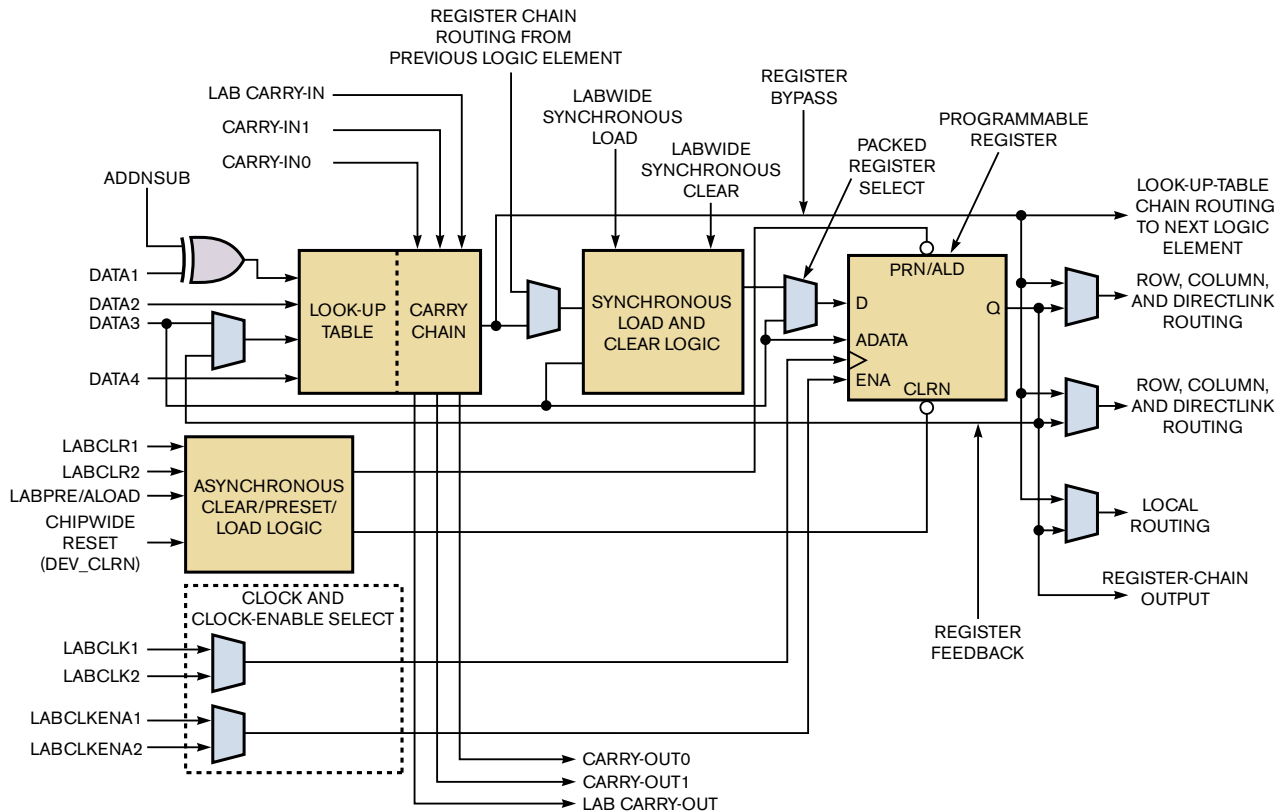
In the meantime, because Actel is the only FPGA vendor to license the ARM7 core, the company's relationship with ARM flourishes, notably with the availability of a low-cost edition of ARM's high-end RealView suite at a price that targets the FPGA market. Over the next few months, Kish promises new reference designs to complement free tools that range from a GNU-based ARM development-tool set that he reckons is 25 to 30% less efficient than RealView—"not so important when Fusion offers half a megabyte of flash"—to IP (intellectual-property) blocks that emphasize on DSP functions: "With our tools, we've optimized our easy-to-use software for new FPGA designers and retained options and short cuts for power users," he says.

CPLDs BLUR FPGA DIVIDE

As the market leader in CPLDs (complex-programmable-logic devices) for

some 15 years, Altera continues to use that term to describe its MAX II product line. The family comprises four devices with 240 to 2210 logic elements—which the company equates to typically 192 to 1700 equivalent macrocells—in packages that range from 100-pin TQFPs to 324-pin BGAs. Maximum user-I/O count spans 80 to 272 pins, and the hot-socketable devices support I/O logic levels of 1.5, 1.8, 2.5, and 3.3V. An internal voltage regulator performs the 3.3/2.5 to 1.8V downconversion necessary for the core logic. Other useful features include programmable Schmitt-trigger inputs and compatibility with the PCI (peripheral-component-interconnect) 2.2 standard for 3.3V operation at 66 MHz. (The device's maximum count frequency is 304 MHz.) The flash memory retains configuration data to enable instant-on operation, with an 8-kbit area set aside for user data.

The fact that these devices employ look-up-table architecture suggests that they have more in common with FPGAs than PLDs (see sidebar "CPLDs go fast and wide" at the Web version of this article at www.edn.com/061123cs). Denny Steele, senior marketing manager for Altera's low-cost products, explains that the fabric that generates



NOTE: LAB=LOGIC-ARRAY BLOCK.

Figure 3 Altera's MAX II CPLDs have more in common with FPGA logic cells.

product terms traditionally differentiates CPLDs from FPGAs: "A traditional CPLD embodies a product-term fabric that's typically built using an AND-OR array, which is just great for applications such as address decoders," he says. By comparison, FPGAs employ a four-input-look-up-table architecture that implements every logical function from effectively four address lines: "Because this design is so fast at performing arithmetic, FPGAs have naturally gravitated toward DSP and communications applications," says Steele. Another key differentiator, he says, is the CPLD's use of crosspoint-switch-routing resources that run everywhere, versus an FPGA's segmented routing, which directly and deterministically links large logic blocks.

For Altera's EPM1270 device, the hardware within the company's \$150 MAX II development kit comprises the development board, a ByteBlaster II parallel-port download cable, and a USB cable for connecting the board to a PC host. The board's PCI-format outline ac-

commodates the CPLD, a 1-Mbit Cypress SRAM, and an ADC with current-sensing circuitry to monitor the CPLD's power consumption. In addition, the board has a two-line, 16-character LCD and the usual array of LEDs, switches, and headers, together with USB and PCI interfaces. The PCI interface is 3.3 and 5V-tolerant. One CD provides a Web edition of the Windows NT/2000/XP-compatible Quartus II development software, and another furnishes system documentation, including an electronic copy of the printed *Getting Started* guide.

The EPM1270 arranges 127 LABs (logic-array blocks) into a 2-D row-and-column structure. Each LAB comprises 10 logic elements, each of which is built from a four-input-look-up-table core with surrounding control, feedback, and routing resources (Figure 3). Using a look-up table allows every logic element to implement any four-input function; LAB-wide carry logic enables fast arithmetic across multiple logic elements. Output signals can route directly to the

next logic element, to the local and global routing system, or through a programmable register before connecting into routing resources. The design accommodates two operating modes—normal and dynamic arithmetic—that the compiler within the Quartus software exploits to best suit normal logic or arithmetic operations. A fast local interconnection system routes signals between adjacent logic elements within the same LAB, and row-and-column interconnections enable global connectivity.

Install the starter-kit material, obtain a license from Altera's Web site, and check the site for a newer version of Quartus II Web edition before installing it; in this instance, Version 6 Service Pack 1 replaces the CD's Version 5 with a 263-Mbyte download. You can optionally install an evaluation copy of Altera's PCI MegaCore IP library, which—if you decide to license the capability—is available for \$1995, representing a \$3000 discount for purchasers of this development kit. When installing the system software,

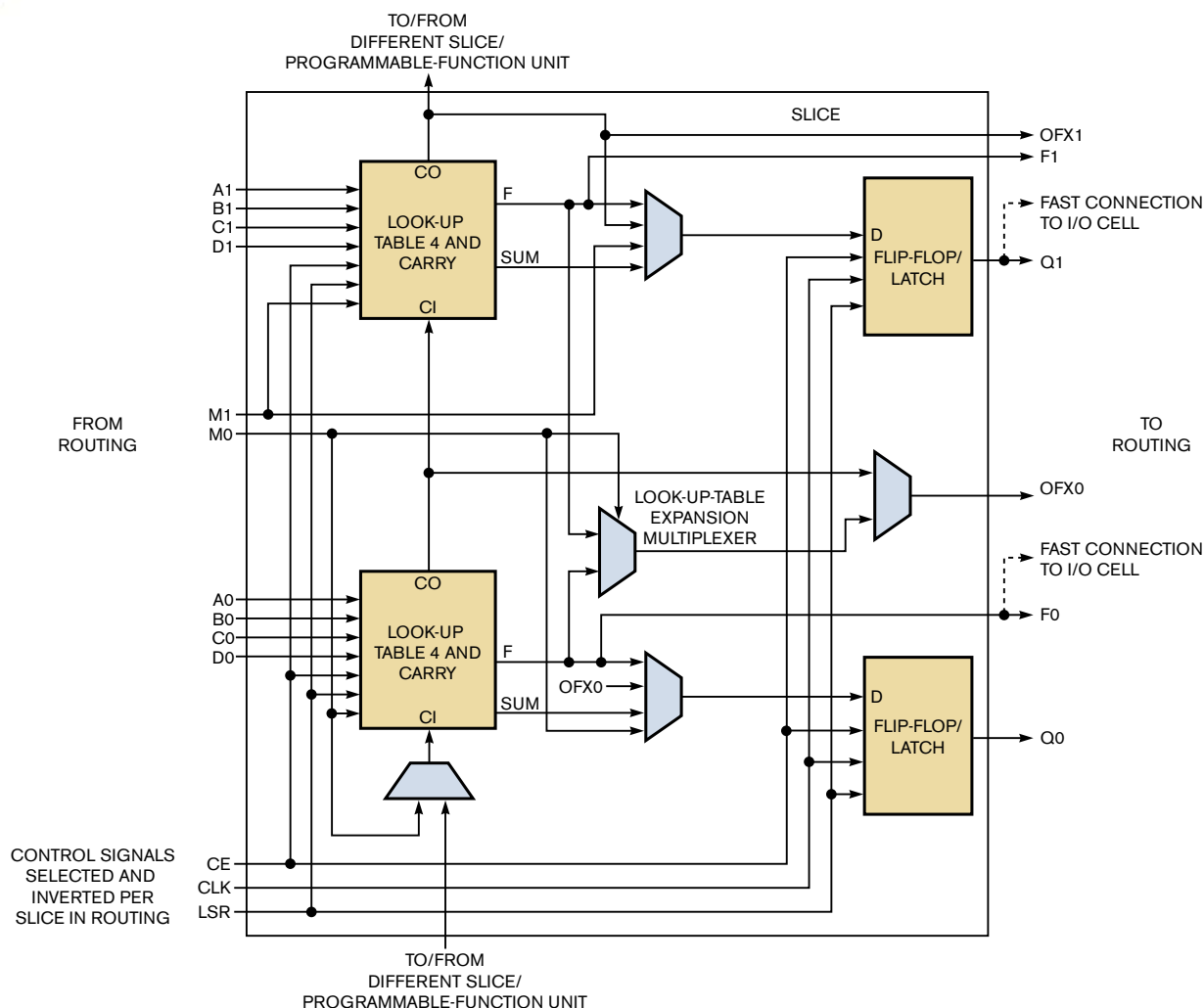


Figure 4 Each "slice" in Lattice's MachXO comprises two look-up-table functions.

be sure to select the Talkback feature, which enables the SignalTap embedded logic analyzer, SignalProbe, and FastFit facilities. It's also worth noting that, in common with all the other boards this article covers, it proved impossible to install the parallel-port programming hardware on a recent PC that has no native parallel port. No product would recognize a parallel EPP/ECP (enhanced-parallel-port/enhanced-capability-port) card that provides legacy support for printers and dongles; in this case, it returned a "no-kernel-driver-installed" message despite the driver's appearance in XP Pro's Device Manager. These issues vanished when substituting a PC with traditional LPT1 and serial ports.

In benchtop use, the MAX II board derives its power from the USB con-

nection that jumper J8:1-2 selects. The board comes with its EPM1270 preprogrammed to run a functional test program that confirms correct hardware operation; you can also reprogram the board with this code using the ByteBlaster tool. Three more demo designs reside within the starter kit's examples\HW\demos subdirectory. They show the EPM1270's power-up timing characteristics, its power consumption, and its ability to be programmed in the background while running another code set—a facility that promises seamless dynamic reconfiguration. For instance, the low-power routine offers the ability to add 150 flip-flops at a time and vary their toggle rate with the LCD reporting current draw; the in-system programming demo allows you to load another routine as the primary code

set runs. Pressing switch S_5 then interrupts the core-voltage power supply, reconfiguring the chip to run the program that you loaded in the background.

Other tutorial software includes reference designs for USB and PCI, along with a slot-machine example. Of these, only the USB example contains Verilog HDL-code sources to guide users familiar with this entry methodology. Sadly, it seems, the immediate help ends here; although the Quartus Help menu includes a step-by-step tutorial for using the IDE along with two introductory manuals for Verilog and VHDL users, the material is a poor educational tool. For instance, the IDE tutorial steps through processes without hinting at why these processes are necessary. After I spent more than two hours follow-

EXPLORING THE DIRECTORY STRUCTURE REVEALS AN EXAMPLES SUBDIRECTORY THAT INCLUDES A TUTORIAL SECTION THAT'S DEVOID OF ANY DOCUMENTATION.

ing a series of arcane steps to arrive at the compilation stage, my first experience failed with a fatal Verilog port-declaration error in the `hvalues.v` file, which I copied directly from the tutorial's instructions; two further attempts were no more successful. Altera's Web site reveals numerous design examples of functions that range from commonplace to exotic, but there does not seem to be a simple guide to getting started.

Users with greater aptitude, experience, or time will doubtless revel in the power that the Quartus IDE promises, such as its block symbolic entry, library primitives, and prebuilt "megafunctions," which you can use as is or customize. There's even a 74-series TTL function library from the earlier MAX-II software interface that may appeal to hardened discrete-logic fans. But for a first-time user like me, the absence of a beginner's tutorial renders the IDE impenetrable; the *Introduction to Quartus II* manual, which similarly lacks any such help, runs to more than 260 pages, and the five-volume Quartus II handbook set is a massive 2160-pg tome!

EASY MIXED-MODE DESIGN

Lattice Semiconductor's \$99 MachXO starter evaluation board measures only 85×72 mm, and most of that space is for headers and connectors. The minimalist hardware comprises the LCMXO256C device in a 100-pin TQFP package, a 33-MHz oscillator, various switches and LEDs, a header for a JTAG connection, and no fewer than three power-supply regulators. These components downconvert the ac-line adapter's 5V dc to 3.3 and 1.2V for appropriate versions of the core, and an adjustable regulator allows users to set I/O-bank levels of 1.25 to 3.3V; out of the box, all levels are set at 3.3V. A printed *User's Guide* describes the board's

hardware and lists I/O connections. You also get an ispDownload cable to link a PC host's parallel port with the board's 10-pin JTAG header, plus a leaflet that describes the programmer.

Unusually, the box contains no software. A single-page flyer points you toward the company's Web site as the source of the ispLever-Starter development environment that's necessary to run the system. Download your choice of software files from Lattice's Web site. At a minimum, you need the base CPLD module; the FPGA extension; and the Precision synthesis engine, the Synplify synthesis engine, or both. Then, request and install a license file, and start the software. But don't forget the separate step of installing the parallel-port driver. Also, be sure to install the Help and User Guide files, because this route furnishes the most straightforward method of becoming familiar with the environment. The recently released Version 6.0 of the IDE introduces numerous updates that include huge improvements to the Help system over the Version 5.1 that I tested earlier, so be sure to update if you have an older installation.

Exploring the directory structure reveals an Examples subdirectory that includes a tutorial section that's devoid of any documentation. In keeping with Lattice's Web-resource spirit, download the *MachXO Family Handbook* to first understand the device architecture. The LCMXO256C that the evaluation board carries is the smallest member of a family of devices offering 256 to 2280 look-up tables, 2 to 7.5 kbits of distributed SRAM, and as many as three blocks of 9-kbit embedded memory in a range of packages that support 78 to 271 I/Os. The two top-specification devices also offer one and two PLLs, respectively. Lattice refers to the MachXO family as a "crossover" device, asserting that this look-up-table-fabric architecture combines the best features of CPLDs and FPGAs to optimally suit use in low-end FPGA applications. Its nonvolatile implementation enables power-up performance within a few microseconds, and the on-chip configuration data alleviates the security concerns that accompany external memories and the possibility of decoding their bitstreams.

The architecture subdivides logic blocks into PFUs (programmable-function units) and PFFs (programmable-

function-units without RAM or ROM). PFUs contain the building blocks for logic, arithmetic, register, and distributed RAM and ROM functions, and PFF blocks suit logic, arithmetic, and ROM functions. Each block of either capability comprises four interconnected slices, each of which houses two look-up tables and their output registers, together with 14 input and seven output signals (**Figure 4**). Each slice supports four operating modes: logic, ripple, ROM, and RAM. Users can concatenate the four-input combinational look-up tables in logic mode to synthesize arbitrary-width functions. The ripple mode, a 2-bit arithmetic-logic unit, also supports counter and comparator functions. The RAM mode allows users to construct 16×2-bit distributed memories including dual-port configurations. The ROM mode omits the RAM's write port, instead taking its setup data from the JTAG programming interface during initial device configuration. Routing resources span two, three, and seven PFUs or PFFs in horizontal and vertical planes, and the clock-distribution network furnishes four primary and four secondary global clocks to complement the 12 internal routing signals that can also deliver clock signals.

The MachXO256 has single-ended input and output buffers with complementary outputs on 78-pin I/O banks, eight lines of which connect to a switch and nine more to LEDs on the evaluation board; the remaining lines appear on unpopulated header pads. To confirm the board's correct operation, the chip comes preprogrammed with a walking-LEDs display, which is also available from the company's Web site. The IDE's top level is Project Navigator, from which you open or create projects. Navigating to the \examples\fpga\MachXO subdirectory within the installation path reveals six subdirectories that contain a number of Verilog- and VHDL-format design files, such as 16-bit upcounters and downcounters. Invoking Project Navigator's Help opens a Web-browser window that provides an overview of available tutorials, user manuals, and sample projects. The *FPGA Schematic and HDL Design Tutorial*, which targets MachXO devices, introduces the IDE's mixed-mode design-entry capabilities, guiding you through top-level schematic and block-symbol creation, pin assignment, design-rule

FOR MORE INFORMATION

Actel
www.actel.com

Altera
www.altera.com

ARM
www.arm.com

Atmel
www.atmel.com

Cypress Semiconductor
www.cypress.com

Digilent
www.digilentinc.com

In-Stat
www.in-stat.com

Lattice Semiconductor
www.latticesemi.com

Mentor Graphics
www.mentor.com

QuickLogic
www.quicklogic.com

Red Hat
www.redhat.com

Sun Microsystems
www.sun.com

SynaptiCAD
www.syncad.com

Synplicity
www.synplicity.com

Xilinx
www.xilinx.com

checking, and Verilog-model creation.

The product offers some neat features, such as the ability to assign multiple instances to a single symbol—in this case, a flip-flop—and the hierarchical navigator makes it easy to inspect the contents and connectivity of any object or net. All steps completed correctly until the last stage, when the design preplanner checks the pin and I/O assignments. At this point, two errors prevented the spreadsheet view of the device planner from opening. But these error messages pinpointed syntax errors within the source files, and double-clicking on each message opened a text editor with the cursor positioned at the error line, making repairs a breeze. From here, you can explore the other steps necessary to fit the design into silicon by following the flow that appears within the Processes for Current Source window, which keeps track of the design's current state. Finally, you can generate a programming file and physically test your work. Like all good software tools, the immediate appeal of this environment lies within its ease of use, and exploration will reward power users with facilities that lie beneath its surface. **EDN**

You can reach Contributing Technical Editor David Marsh at formcett@btinternet.com.

REFERENCES

- 1 "The field-programmable gate array (FPGA): expanding its boundaries," In-Stat, April 2006, www.in-stat.com.
- 2 Santarini, Michael, "FPGAs balance lower power, smaller nodes drip by drip," *EDN*, June 8, 2006, pg 58, www.edn.com/article/CA6339245.
- 3 www.verilog.net/docs.html.
- 4 www.vhdl.org/comp.lang.vhdl/FAQ1.html#3.

CPLDs GO FAST AND WIDE

In 1975, when PLAs (programmable-logic arrays) first appeared, devices embodied two AND-OR planes that designers configured by “blowing” unwanted fuse connections between the logic gates (Figure A). This architecture allows any combination of AND-OR terms as well as the ability to share AND terms across multiple OR gates. Although it provided users with great flexibility and the highest available logic density, the 10-micron-process technologies of the time made the devices relatively slow. Variations on the theme followed, such as the faster and simpler PAL (programmable-array-logic) devices that fix the OR plane. Today, these devices remain popular in SPLD (simple-programmable-logic-device) form, with generic parts such as the 16V8 often appearing in address-decoding roles. Proprietary enhancements allow devices such as Atmel’s ATF16V8CZ to operate in simple mode for decoders; in registered mode to add basic counting, storage, and synchronization capabilities; and in complex mode that combines output and I/O functions. In this way, the device’s “universal architecture” emulates a host of 20-pin PAL devices and offers maximum propagation delays of 12 nsec. The CZ-suffix version also adds a power-save mode to the basic ATF16V8C device that automatically puts the chip into standby mode when there’s no activity on its inputs, reducing its 5V consumption to around 5 μ A.

CPLDs (complex PLDs) furnish greater density than SPLDs but retain their general-purpose interconnection fabric. Again, proprietary enhancements furnish features that lower overall system cost, and today’s process technologies support clock rates in excess of 300 MHz. For instance, Xilinx, which introduced the FPGA in 1985 and remains the technology’s leading supplier, uses an all-CMOS process to build its CoolRunner-II CPLDs to obvi-

ate static-power drain. Optimized for 1.8V systems, the family retains the FPGA-like ability to operate its I/O banks from mixed-level supplies to provide compatibility with 1.5 to 3.3V external logic. Propagation delays range from less than 4 nsec in 32-macrocell CoolRunner-II devices to approximately 7 nsec in a 512-macrocell version. This speed and the wide-input-term architecture suit complex, high-speed-bus interfaces as well as a host of other roles in which an FPGA may represent overkill.

Available for just \$49, the CoolRunner-II Design Kit comprises a 133-mm-sq board that carries two CPLDs, a 1.8432-MHz oscillator module, power supplies, and a mass of prototyping area. Promisingly for first-time users, you get a print copy of Xilinx’s *Programmable Logic Design Quick Start Handbook*, which runs to almost 200 pages, setting an example that other vendors would do well to emulate. The wealth of useful documentation continues on the accompanying resource CD, which carries everything from the evaluation board’s schematics to a guide to replacing TTL with CPLDs. Install the free WebPack-design software from the Xilinx Web site, and the software is ready to use; there’s no messing with licensing files or limited-time use here. You can also install the Xilinx version of the ModelSim behavioral simulator and opt for a time-limited full version or a free version that supports debugging of as many as 500 lines of code.

The design kit, which Digilent built, includes the JTAG-3 parallel-port adapter cable for device programming but requires you to furnish a power supply for the board. A double-AA-cell-battery holder enables offline use, or you can apply 5 to 9V-dc wall-adaptor power to the center-positive socket. To avoid damaging the board, be sure to correctly set the power-select jumpers before apply-

ing power; the wall-adaptor input is safest because it feeds two regulators that can withstand as much as 18V. Jumpers also allow you to connect or disconnect either CPLD from the JTAG chain and supply an external clock of 32 kHz to 100 MHz in place of the onboard oscillator. Four dual-row, 40-pin connectors make available all signals from both CPLDs and are compatible with several expansion boards from Digilent’s range.

The first of the two CPLDs is the XC9572XL, a general-purpose, 72-macrocell device that offers 1600 gates, 72 registers, and 34 I/Os within a 44-pin PQFP. It suits applications that require 3.3V supplies and 5V I/O compatibility. By comparison, the XC2C256 is a 256-macrocell device from the CoolRunner-II range. Its architecture combines the macrocells into 16 function blocks that interconnect through the proprietary AIM (advanced interconnect matrix). Each macrocell is a 56-input logic block whose product terms can route to and share any of the macro-

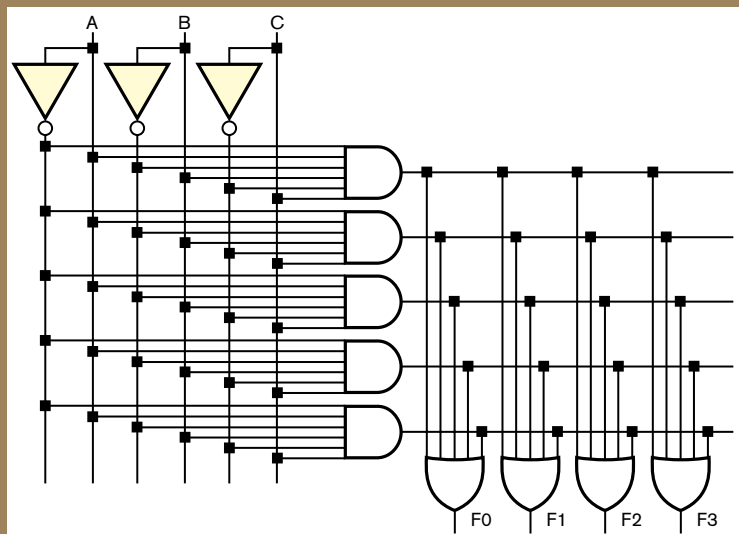


Figure A Early PLA devices used a flexible AND-OR design that particularly suits address decoders.

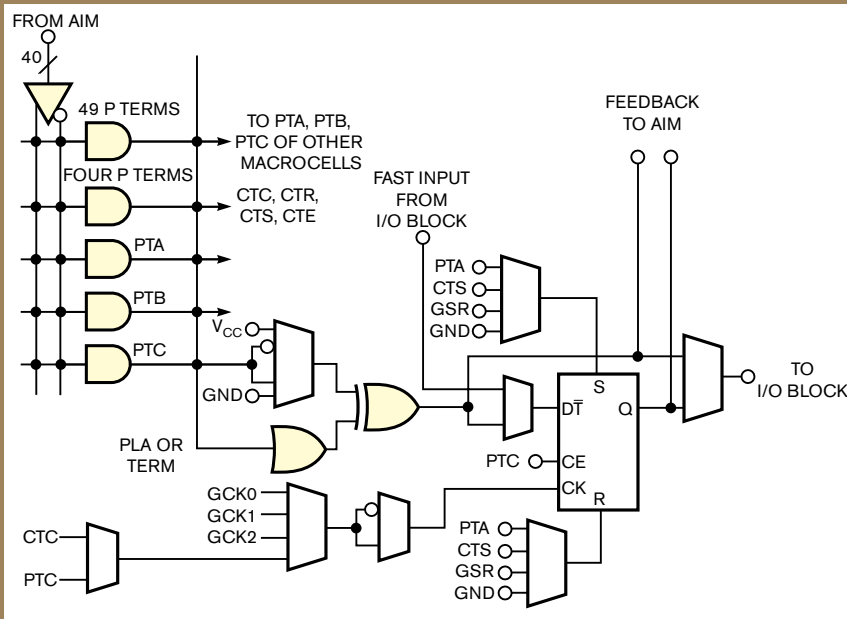


Figure B CoolRunner-II macrocells accommodate as many as 56 inputs.

cells within its function block (Figure B). This arrangement maximizes routing flexibility and minimizes propagation delays, thereby overcoming the variable-timing model that can result from the necessity of “borrowing” unused product terms from adjacent macrocells to expand product-term width. The clock system includes a dedicated input pin that connects to a programmable-times-two to -16 divider complete with synchronous reset circuitry, which prevents runt pulses from entering the global distribution network during power-up. The 144-pin TQFP device in the design kit offers 118 I/Os that can optionally offer about 500 mV of Schmitt-trigger-input hysteresis.

Apply power to the board, and a confidence-check rou-

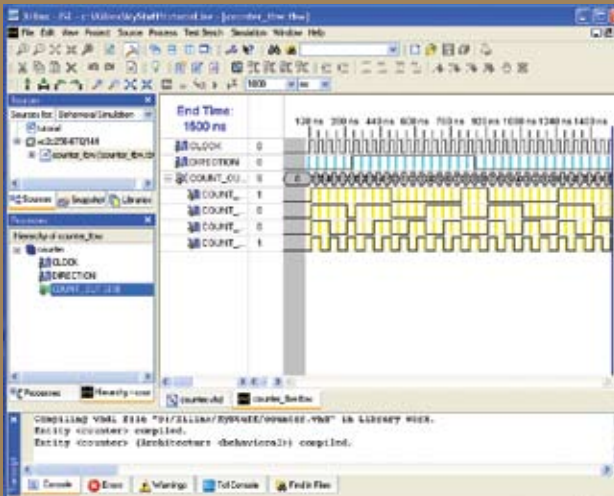


Figure C WebPack's timing simulator interface is straightforward to use.

tine runs, flashing a pair of LEDs. Pressing the adjacent pushbutton causes the routine to cycle through four possible states. You then have several options to familiarize yourself with the WebPack IDE: Read the printed manual, view a Web-based tutorial that's available from the IDE's help menu, or download an in-depth PDF tutorial that opens within the IDE's work space. Although a VHDL-for-CPLD tutorial appears on the resource CD and Digilent's Web site offers a free five-minute video introduction to VHDL, you will not find a dedicated tutorial that targets this board and how to program it. But the IDE's quick-start tutorial, which supports the Spartan-3 demo board, is easy to follow—describing how to implement an upcounter or downcounter, the sample steps

through design entry, behavioral simulation, and timing-constraint entry. If you initially selected the XC2C256, only this last step fails to follow the Spartan-3 sample, because the CPLD generates a fitter report rather than the FPGA's design summary. The pin-assignment editor is different, too, but again, the built-in help quickly guides its use. And the timing-simulator interface is a powerful demonstration of how to make a complex function truly accessible (Figure C).

The 168-pg, in-depth tutorial includes, for instance, schematic entry. You can examine how to implement mixed-mode macro-, schematic-, and VHDL-based design by opening the watch_sc_cr2 example from the IDE's File-Open Example menu. Furthermore, this example targets a CoolRunner-II device, the XC2C128. In fact, the biggest difficulty came when trying to program—or rather, read back—the devices on the board to save the test program before downloading a counter design. Unlike many environments that have separate device-programmer interfaces, WebPack integrates the functions beneath the Generate Programming File/Configure Device tab in the Processes window. Leave all the JTAG jumpers in their default positions, connect the programming cable, invoke the programmer software, select the Automatically Connect to a Cable & Identify Boundary-Scan Chain option, and the software identifies both devices. Right-clicking on either device then presents a programming menu. According to Digilent's *Impact Device Configuration Notes*, the IDE doesn't generate JTAG-programming configuration files by default. So, right-click on Generate Programming File, select Properties, and tick the box Create IEEE 1532 Configuration File. Further assistance is available at the Xilinx Web site, which includes a range of free CPLD reference designs and code files.