Cunning circuits confound crooks

Illustration by Daniel Guidera

AS PROGRAMMABLE LOGIC INCREASINGLY ENCROACHES ON HIGH-VOLUME TERRITORY FORMERLY DOMINATED BY ASICs, UNSCRUPULOUS OPERATORS ARE LICKING THEIR LIPS AT THE PROSPECTS OF EASILY DUPLICATED, OR EVEN REVERSE-ENGINEERED, DESIGNS. READ ON TO LEARN HOW YOU CAN STOP THE LOGIC LIFTERS IN THEIR TRACKS.

HAVING A BAD DAY? Maybe you've just received word that a competitor (perhaps in a country whose copyright- and patent-protection laws and corresponding enforcement are less stringent than those in your abode) is manufacturing a near-duplicate of your latest FPGA- or PLD-based product, differing only in its cosmetic appearance. Or maybe the innovative programmable-logic-housed circuit you developed just weeks or months ago has suspiciously appeared in another company's device or system in a slightly modified or enhanced form. Or maybe you're an IP provider, and you've heard through the grapevine that your customer is building more chips or boxes than the license agreement they signed with you gives them the right to do.

All of these situations, as well as other detrimental scenarios that a few minutes of "what-if" thinking will uncover, unfortunately suggest that you're the victim of design theft. In one manifestation of the general concept, someone might make an exact duplicate of your entire design, a technique that this article refers to as "cloning." The other harder but possibly even more damaging form of theft is "reverse-engineering." In that scenario, someone uses the information stored in the programmable-logic device to reconstruct the original circuit details and then alters and incorporates those details in part or whole into other designs.

Design theft has historically been of greatest concern with military projects. But, as programmable logic's integration capability and performance rise and as cost per gate plummets, the potential for cloning and reverse-engineering becomes a concern for an increasing number and variety of broader-based applications. Fundamental limits still exist as to the

types of systems for which design theft is a temptation. Low-volume designs may have long lifetimes, but their limited volumes tend to give an insufficient return on the "cracking" investment. They are also often highly complex projects with extensive interaction between multiple hardware and software subsystems, making them exponentially more difficult to clone or reverse-engineer.

High-volume systems, on the other hand, are frequently profitable only in the early stages of their life cycles, although the minimal to no R&D expenses that cloning and reverse-engineering provide may significantly extend the profitability time span. Time to crack is critically important to thieves of high-volume designs, because fickle consumer tastes often result in a product's disappearing from store shelves after a single holiday shopping season. Between these two volume extremes, plenty of programmable-logic-based systems that in the past might have employed the often harder to surmount ASIC attract the "cracker's" attention.

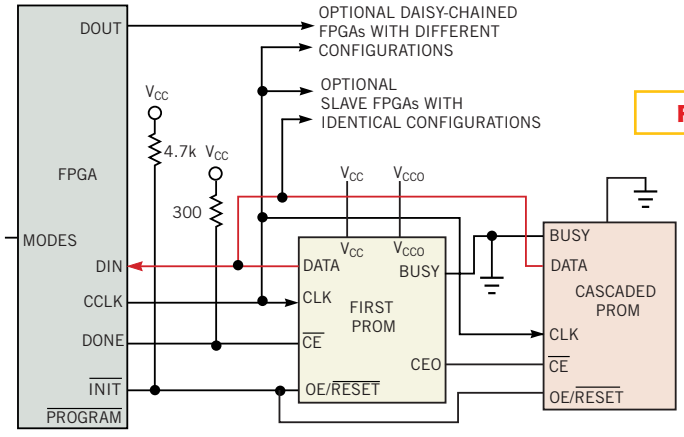
AT A GLANCE

- As you migrate from ASICs to programmable-logic devices, your concerns about design theft may increase, depending on the technology you choose.
- Reverse-engineering a design can be tough for all but the simplest PALs, but a clone is often good enough to satisfy a cracker's objectives.
- SRAM-based parts are the most vulnerable, although secure variants will be available for sampling by year-end.
- Nonvolatile-memory-based chips are challenging to crack, though they are easier to break into than antifuse devices are.
- A variety of security techniques can make your design too challenging for all but the most determined and well-financed thieves.

TECHNOLOGICAL WEAKNESSES

If you haven't yet employed security measures in your designs, you may be disconcerted at this point and eager to dive into solutions. Resist this knee-jerk reaction temptation. Before jumping directly to answers, it's important to first fully understand the problem so that your approach doesn't create other bigger issues. This need to research before reacting is particularly important because different programmable-logic technologies exhibit different Achilles' heel points of vulnerability.

The classic method of reverse-engineering a programmable-logic design involves cycling through all possible device input combinations and capturing the corresponding output bit patterns. By using essentially a huge Karnaugh map (often with high-powered computer help) or through visual inspection of data patterns, a thief can derive the Boolean equations that define the internal logic. This technique works particularly well for low-density PALs with dedicated inputs and outputs and for designs containing only asynchronous circuits and latches. Synchronous designs, particularly state machines and linear-feedback shift registers, complicate the puzzle. Today's burgeoning gate counts with corresponding increases in input and output pin counts also exponentially increase the reverse-engineering complexity.



The interconnect between configuration memory and programmable logic is the SRAM-based device's weak link (courtesy Xilinx).

Most of today's chips contain not only dedicated inputs and outputs but also a large percentage of pins that can be input, output, or both (which is the case with bidirectional buses). Unless the device pin connects to a defined connector signal or another device's pin of known function, such as a microprocessor interrupt input, figuring out whether to drive a pin with a voltage, sense its output state, or both isn't a straightforward exercise. Nongeneric embedded-logic circuits, such as hard PCI cores, and embedded-memory arrays are additional complications that make Karnaugh map reverse-engineering increasingly impractical. Remember, though, that reverse-engineering is overkill if the crook needs only to clone the design.

Another common loophole, which theoretically shouldn't exist but that Internet newsgroups indicate is all too widespread, involves simply reading back the internal configuration bit stream through a chip's JTAG or proprietary programming interface. Don't forget to set the security bit after programming a device. Unfortunately, "locking" your chip might not be good enough to prevent theft. It's easy to surmount some device families' security safeguards by applying high-voltage sequences to certain pins. This action puts the chips into manufacturing test modes, which re-enable internal-device-state access. I'd suggest that if cloning or reverse-engineering is a concern, ask other designers at the newsgroups in **Reference 1** for advice on particularly robust (and conversely vulnerable) chips before settling on a vendor and an architecture.

Figure 1

SRAM-based CPLDs and FPGAs are in many respects the programmable-logic devices most vulnerable to design theft. All a thief needs to do to copy them is to siphon the configuration bit stream that is stored in a separate nonvolatile memory, in CPU firmware, or somewhere else on the network and is loaded into them upon power-up (**Figure 1**).

Mapping a proprietary, undocumented bit stream to a device's internal logic structures and from there reverse-engineering the original design is more complicated than simple cloning, but few things are impossible. For example, independent EDA vendor NeoCAD several years ago developed place-and-route software for Xilinx FPGAs by translating pieces of the bit stream back to their corresponding look-up tables; pass transistors; and other interconnect, logic, and memory elements. Xilinx subsequently acquired NeoCAD.

Ironically, Xilinx may have inadvertently given the design thieves a head start with its JBits tool, intended for use in reconfigurable-logic applications. JBits lets you modify internal FPGA resources with very fine-granularity resolution. In the process, though, it gives you insight into what portion of the bit stream corresponds to those resources. Other programmable-logic vendors haven't yet introduced tools of comparable function (and dubious value).

Altera, the other heavyweight in today's programmable-logic business, is not without its own skeletons in the closet. Start-up Clear Logic, which makes laser-configurable logic devices, has proved that it can not only replicate Altera CPLD and FPGA silicon in smaller die, but also redirect Altera-intended bit streams to that cloned silicon. In fact, Clear Logic's marketing pitch to use Altera chips for prototyping and Clear Logic chips for production has an interesting beneficial security twist: no bit stream, unlike Altera FPGAs, and no easily probed configuration transistors, unlike Altera CPLDs.

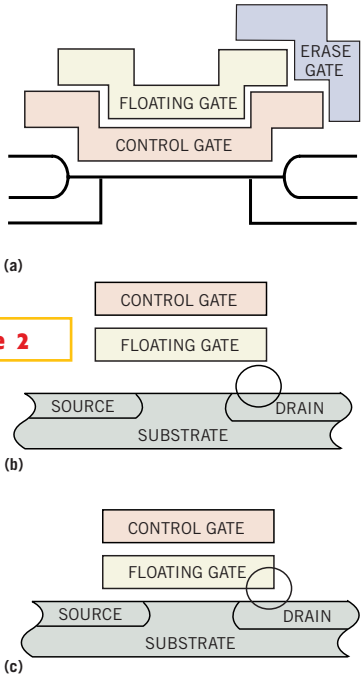


Figure 2

EEPROM (a), EPROM (b) and flash memory (c) cells all contain a floating gate whose stored charge status can be sensed (courtesy Annabooks Press).

NONVOLATILE: HARDER TO CRACK?

Next let's turn our attention to programmable-logic devices based on nonvolatile, user-programmable memory technologies, such as PROM, EPROM, EEPROM and flash memory (**Figure 2**). Unlike with SRAM-based chips, no configuration bit stream now exists into which a design thief can tap. However, as their name implies, nonvolatile memory-based chips retain their stored internal states when power is removed from them, opening them (both figuratively *and* literally) to different techniques for analysis and extraction.

The central aim of inspecting nonvolatile memory-based PALs and PLDs, available from numerous manufacturers, and FPGAs, currently only Actel's ProASIC line, is to determine whether electron charge is present on each configuration transistor's floating gate. The cracker must first gain access to the silicon die by removing the package lid. He or she may also, depending on the chosen approach, need to remove the die's upper-most passivation layer, as well as metal interconnect and other strata above the transistors. The same sorts of techniques find use in layer-by-layer reverse-engineering of ASICs.

At least three different analysis methods are currently available. Assuming that the cracker knows or can visually determine the locations of the configuration transistors, he or she can probe each of the thousands or millions of transistors' floating gates and electrically determine the presence or absence of charge and, therefore, voltage. After placing the die in a vacuum chamber and powering up the chip, the cracker can also use a secondary electron microscope to detect and display emissions. The third option involves application of a material such as lithium niobate to the die, applying an ultraviolet or other light source and, just as with LCDs on watches, observing where the liquid-crystal structure discolors.

Each of these alternatives involves time-consuming, tedious work. The return on investment of this effort depends on the value of the design stored inside the chip. Assume that the cracker succeeds in determining the overall transistor layout and the locations of unprogrammed and programmed transistors on the chip. He or she still needs to somehow (in the absence of documentation, which no silicon vendor publishes) translate that pattern back to a configuration bit stream and then (if reverse-engineering is the goal) into the design source code or schematic. This undertaking, at first glance, might seem so difficult that it doesn't justify your making the effort to prevent it. Before dismissing the threat, however, visit the Web sites of companies such as Integrated Circuit Engineering or check out **Reference 2**. ICE's goal is to understand the chip, not the designs stored within it, but the company's capabilities are symbolic of the reality that such expertise exists throughout the world. As Andy Grove from Intel once said, "Only the paranoid survive."

The last major programmable-logic approach existing today is antifuse technology, which industry experts generally regard as the most secure of all the options. Because of this fact, along with its high radiation tolerance, it often finds use in military designs and high-volume consumer applications, such as S3's Rio

ACRONYMS

AES: Advanced Encryption Standard
ASIC: application-specific IC
CPLD: complex PLD
CPU: central processing unit
DBS: digital-broadcast satellite
DES: Data Encryption Standard
FPGA: field-programmable gate array
FPSLIC: field-programmable system-level integrated circuit
IP: intellectual property
JTAG: Joint Test Action Group
PLD: programmable-logic device (product-term based)

portable audio players. Several factors complicate attempts to electrically probe an antifuse wafer with the intent of determining open- or short-circuit state. The first impediment is the sheer number of antifuses in a typical richly populated FPGA and the low percentage of them programmed for most circuits. Ken O'Neill, director of product marketing at Actel, points out that the company's A54SX16A, for example, contains approximately 800,000 antifuses with only 2 to 5% of them programmed in an average design. By looking at the top of the die, you cannot distinguish an antifuse from a normal metal-to-metal layer via.

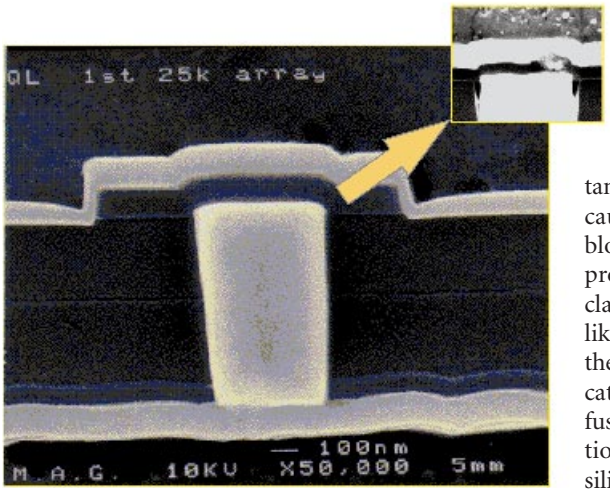


Figure 3 Antifuses are extremely small, difficult to inspect, and abundant in quantity with a low-percentage programmed number (courtesy QuickLogic).

The only reliable method of viewing the state of each antifuse is to cut it in cross sections, which will most likely destroy the rest of the die (**Figure 3**). Because the conducting filament is much tinier than the antifuse itself, a thief may have to make a cross-section cut and view each antifuse multiple times before he or she actually sees a conducting filament. How many designs are valuable enough to destroy 800,000 or more die in an attempt to crack their stored circuits? Actel claims that the company's Silicon Explorer II internal-node-inspection tool represents no security threat. Special antifuses, when blown, prevent internal probing and overwriting. They're located such that a thief cannot access and bypass them without destroying the rest of the device.

Chuck Tralka, vice president of strategic marketing at antifuse vendor QuickLogic, also points out that device programming is a multistep process. The back-end place-and-route design software generates an ASCII file containing information about which links are to be programmed. The device programmer matches this data against a database, which translates each link name to a physical die location and then tells the programmer how to access that link. Timing and commands, addressing, and programming order are all important with antifuse technology because programming some links blocks the subsequent ability to program nearby links. Tralka claims that a cracker would most likely be unable to obtain all of the information needed to duplicate or reverse-engineer an antifuse FPGA, because the information comes from either the silicon vendor (as is the case with QuickLogic, which does all device programming itself) or multiple sources (if a third-party vendor handles the programming).

FOILING THE THIEF

When concerned about design cracking, keep in mind that the objective is not necessarily to incorporate state-of-the-art security that would protect Fort Knox. Your job is to come up with an approach that, at minimum, takes slightly longer to circumvent than the time beyond which the media's worth becomes negligible (**Reference 3**). The first step you should take is to ask your vendor to sell you unmarked or custom-marked programmable-logic devices. This sim-

FOR MORE INFORMATION...

For more information on products such as those discussed in this article, go to our information-request page at www.rscahners.ims.ca/ednmag/. When you contact any of the following manufacturers directly, please let them know you read about their products in *EDN*.

Actel
1-408-739-1010
www.actel.com
Enter No. 315

Altera
1-408-544-7000
www.altera.com
Enter No. 316

Atmel
1-408-441-0311
www.atmel.com
Enter No. 317

Clear Logic
1-408-361-2600
www.clear-logic.com
Enter No. 318

Cypress Semiconductor
1-408-943-2600
www.cypress.com
Enter No. 319

Integrated Circuit Technology
1-408-434-0678
www.ictpld.com
Enter No. 320

Lattice Semiconductor
1-503-268-8000
www.latticesemi.com
Enter No. 321

Lucent Technologies
1-908-582-8500
www.lucent.com
Enter No. 322

QuickLogic
1-408-990-4000
www.quicklogic.com
Enter No. 323

Xilinx
1-408-559-7778
www.xilinx.com
Enter No. 324

OTHER COMPANIES MENTIONED IN THIS ARTICLE

Cisco Systems
www.cisco.com

Dallas Semiconductor
www.dalsemi.com

Integrated Circuit Engineering (ICE)
www.ice-corp.com

Intel
www.intel.com

SUPER INFO NUMBER
For more information on the products available from all of the vendors listed in this box, enter No. 325 at www.rscahners.ims.ca/ednmag/.

ple action, in the absence of obtained schematics for your design, will effectively slow down the cracker attempting to identify the chip (particularly for SRAM-based devices in conjunction with a no-configuration-memory technique.

Next, you should embed some distinguishing, undocumented identifier in the design, such as your initials. For example, the ASICs I designed almost a decade ago contain my initials, as well as those of the other engineers on the project, in the card-identifier structure. Other engineers sometimes waste a few gates to embed circuits with no functional value to the design, such as a few-bit state machine that does nothing but loop through all possible states and doesn't hook up to any other logic. These steps don't necessarily prevent someone from cloning or reverse-engineering the chip, but they prove effective in identifying the original source of the design in a court of law.

If your system connects to a network, embed a chip-specific identifier in each system and build intelligence into the network to detect whether multiple identical identifiers are online at once. DBS, cable-television, and digital-cellular-phone vendors, for example, as well as online-computer-game software, such as *Quake 3* and *Unreal Tournament,* use this technique to spot fraud. Network connectivity also implies that you may be able to satisfy two other ideal security system characteristics—renewability and upgradability—to contain and overcome any security breach resulting from a successful crack. The downside of the unique-identifier approach is that, like the two-chip technique described later, it complicates your manufacturing flow because it requires chip-specific configuration-bit-stream files.

One way to impede or at least discourage pin and trace probing, device removal, package decapping, and the like involves covering the chip—and perhaps other connected chips and the signals running between them—with a resin epoxy or other impenetrable coating. This approach, however, prevents you from performing postmanufacturing rework, potentially reducing board yield, and may cause thermal problems due to inhibited heat transfer away from the devices. Alternatively and particularly effective with BGA-packaged parts, you could conceal the configuration bit-stream trace running between memory and programmable logic device in an inner pc-board layer.

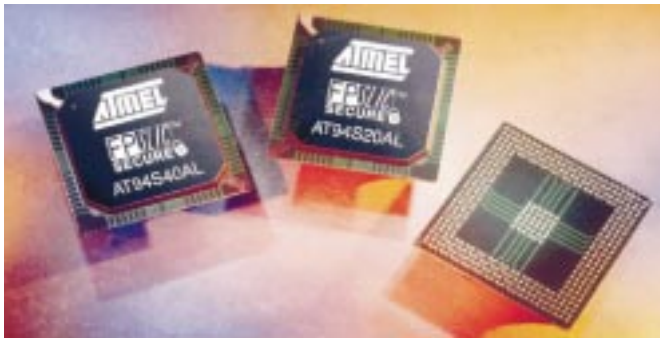


Figure 4 Secure variants of the AT49K FPSLIC family will include a custom, two-bus configuration memory in a dual-die package (courtesy Atmel).

For SRAM-based devices, you might choose to do away with the onboard nonvolatile configuration memory, whose presence also blatantly unveils the identity of the programmable-logic chip even with an unmarked or a custom-marked package. Instead, you'd power up and program the part as the board moves down the production line and then use a backup lithium-battery pack to hold its configuration in the absence of a conventional system power source. This technique sounds good in theory, but reality is less straightforward. The programmable-logic device requires its own power-supply line, and both supply voltages and I/O pins must be electrically isolated from the rest of the board to prevent other chips from attempting to power themselves from the device-to-device interconnect in the absence of their V_{CC}. This electrical isolation-buffer circuitry, along with the battery and switching circuits, adds board space and system cost, and the propagation delays through the buffers sap performance.

You have to ensure that no internal device nodes or input or output buffers toggle while the device is battery-powered to avoid dynamic power consumption. PLLs; power-on-resets; and other embedded, analog-dominated circuits may consume more current than desired even in the absence of an active clock. And, ultimately, the battery-backup approach is volatile, because the battery eventually will fail; when that occurs depends on operating and storage temperature as well as the percentage of time the device is battery-powered. Chip-configuration obliteration can also come from momentary power loss as a result of system shock or vibration or due to voltage and ground transients. At that point, unless your assumption that the system will no longer be in operation is correct, you have a potentially expensive service call on your hands.

The other common security technique for systems incorporating SRAM-based programmable logic involves partitioning the design between an SRAM-based chip and a nonvolatile memory-based chip, such as a PLD, an embedded microcontroller, or an external "dongle." Both parts would contain the same per-system unique identifier, and the identifiers would have to match. Alternatively, some other system-boot handshake sequence would have to successfully complete before normal operation could begin. Multichip partitioning is sometimes impossible, though, and you cannot upgrade the portion of the design in the CPLD. Programmable-logic-design consultants report that, with a little effort, a cracker can configure an EEPROM and a CPLD or low-end microcontroller to act as an impersonator for a nonvolatile-serial-number chip, such as those from Dallas Semiconductor.

YET ANOTHER METHOD

A third security approach incorporates identical linear-feedback-shift registers in a nonvolatile memory-based CPLD (which has no other purpose except to validate the other chip) and an SRAM-based FPGA. This approach is not attractive for use with Xilinx parts because of the JBits utility's ability to reverse-engineer the circuit (**Reference 4**). However, for other SRAM-based architectures, the technique has merit. It may sufficiently complicate the reverse-engineering process to make it unattractive for a potential design thief, with complexity a function of the linear-frequency shift register's design and length.

Have no fear, because the programmable logic suppliers acknowledge the in-

creasing need for robust design security, and numerous approaches are on the way. In the vendors' defense, a fundamental trade-off exists in adding security features, just like there is with anything else that's fixed-function and integrated on the chip. The vendor needs to balance the added cost, complexity, and potential yield loss of such a feature with the number of customers who would pay extra for the added function or the number of customers who, by providing the feature, the vendor could lock out from its competition.

With otherwise-SRAM-based FPGAs, for example, adding nonvolatile memory for unique device identifiers might be cost-prohibitive. Instead, Xilinx is including a hard-wired Triple-DES decryption block, along with two sets of 56-bit key registers and dedicated battery-backup supply-voltage inputs for only those registers, on its upcoming Virtex-II FPGAs. Xilinx estimates that the registers alone consume only nanoamps of current, orders of magnitude lower than if the entire device needed to be battery-powered. Xilinx's approach not only prohibits device cloning, but also prevents unwanted rogue bit streams, such as viruses, from being downloaded to the part in this increasingly network-connected world.

In one possible scenario the company envisions, an equipment manufacturer, such as Cisco, might define and program one set of keys, allowing each of its customers (such as an Internet-service provider) to define and configure the second set. Cisco would be able to upgrade over the Internet all of its routers in the field, and each customer would retain its ability to perform proprietary updates without worrying about other Cisco customers exploiting the same capabilities. Although crackers have deciphered DES, and AES is on the way to replace it, Xilinx officials believe that triple DES provides sufficient robustness with the added benefit of the fact that manufacturers can export it outside the United States.

You load the keys into the chip via JTAG, read them back to ensure accuracy, then lock the part with a JTAG command write, and subsequently can send a single initial configuration bit stream to the FPGA. Beyond that point, the part accepts only encrypted bit streams that specify which key block to use. Device hardware resets and power downs do not delete the key values, as long as the system retains the key register's backup power. Invalid encrypted bit streams prohibit the FPGA from completing its start-up sequence. Any attempt to reset the part to its nonsecure mode via commands flushes both the keys and the stored configuration bit stream.

Atmel's AT49K FPSLIC CPU-plus-FPGA chips currently employ an external nonvolatile memory to store both the CPU firmware and the FPGA configuration. By the end of this year, the company will begin making single-chip, dual-die devices available for sampling. These devices will have a specially-designed, nonvolatile-memory array (**Figure 4**). Containing two data buses, one going to the system and the other running only within the package between the memory and AT49K, the memory also includes a security bit. Once you set this bit, it causes the chip to respond only to a full-chip-erase command. You must set the security bit to initiate memory-to-AT49K communication.

The biggest complication of Atmel's approach is the full-chip-erase function; it implies that the system is devoid of both a microprocessor and a dedicated logic function until you fully reprogram the configuration memory and the security bit. Also, the approach doesn't eliminate the security threat; although the external bit-stream trace between memory and AT49K is no longer present, stripping back the multidie package lid exposes it for probing. But the approach is an example of low-cost security that in many cases will be good enough to foil the efforts of would-be thieves. Atmel plans to charge a 10 to 20% premium for its secure AT49K multidie devices over the nonsecure two-chip alternative.

A few more words on hybrid chips: The Atmel example shows that a hybrid device isn't fully secure (hardware and software) unless both the nonvolatile firmware and the configuration memory reside on the same die as the CPU and the programmable logic (**references 5** and **6**). Keep this fact in mind as you evaluate the upcoming hybrid chips from Altera, Lucent Technologies, QuickLogic, Xilinx, and others. As for other hybrid chips that, for example, include an ASIC-housed PCI core or Lucent Technologies' dedicated microprocessor interface, these buses provide alternative configuration paths for the FPGA, but the embedded dedicated logic isn't "smart" enough by itself to handle an encrypted configuration file, as Xilinx's upcoming parts can do.

Finally, Actel hopes within a year to begin rolling out versions of its flash-based ProASIC FPGAs with unique chip identifiers built-in. The company sees the value in such a feature to help protect your custom logic designs. It also, however, points out that there's a benefit to IP providers, which could, for example, program a core to operate only in devices that span a certain device-ID range, preventing the use of the core outside the licensed quantity by unscrupulous customers.□

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

References

1. www.ednmag.com/ednmag/verticalmarkets/PLD.asp.
2. www.cl.cam.ac.uk/~mgk25/sc99-tamper.pdf.
3. Dipert, Brian, "Media security thwarts temptation, permits prosecution," *EDN*, June 22, 2000, pg 101.
4. www.free-ip.com/copyprotection.html.
5. Dipert, Brian, "The best (or worst?) of both worlds," *EDN*, Nov 11, 1999, pg 139.
6. Levy, Markus, "Processors drive (or dive) into programmable-logic devices," *EDN*, July 20, 2000, pg 107.
7. Dipert, Brian, "First annual programmable-logic directory," *EDN*, Aug 17, 2000, pg 54.
8. Dipert, Brian, "Low-cost programmable logic: How low should you go?," *EDN*, March 16, 2000, pg 123.
9. Dipert, Brian, "Moving beyond programmable logic: if, when, how?," *EDN*, Nov 20, 1997, pg 77.

Acknowledgments

The original inspiration for this article, as for several that I've done on programmable logic, came from a lengthy and instructive discussion thread earlier this year on Internet newsgroup comp.arch.fpga. Search for topic name "How to prevent theft of FPGA design" at Web site www.deja.com for all the details.