FPGA and Structured ASIC JOURNAL
TRENDS TECHNOLOGY TOOLS TECHNIQUES
www.fpgajournal.com

Logic Lockdown

Design Security Part 2

Engineers are trained problem solvers. While various fields of engineering require different types of technical training and expertise, the techniques of problem solving are universal to all branches of the profession. If engineers are problem solvers, could one infer that reverse engineers are problem creators? In a narrow view, probably so – but reverse engineering has its place in the innovation cycle as well. Reverse engineers also help us hone our security skills to prevent attacks from those who wish to do us (and our design IP) harm.

Reverse engineering is not a back-alley, cloak and dagger, business-in-the-shadows affair – quite the contrary, in fact. Companies specializing in reverse engineering operate openly and have a long and public history, particularly in the semiconductor arena. In the United States, reverse engineering has the protection of law, with the Supreme Court ruling that "A trade secret law, however, does not offer protection against discovery by fair and honest means, such as by independent invention, accidental disclosure, or by so-called reverse engineering, that is by starting with the known product and working backward to divine the process which aided in its development or manufacture."

The US Semiconductor Chip Protection Act specifically legalizes reverse engineering of competitors' chips, both for the purpose of making compatible chips and for the purpose of producing a better, competing product. If you're protecting something copyrightable (like software, music, images or video) the Digital Millenium Copyright Act spreads its umbrella a little bit in your direction. It apparently isn't legal to reverse engineer in order to defeat protection schemes for copyrighted content, but even the DMCA has an exception permitting semiconductor reverse engineering.

Reverse engineering companies such as Semiconductor Insights and Chipworks have reverse engineered thousands of devices, publishing the results in detailed technical reports. Their business comes primarily from sources like semiconductor companies wanting to know how their competitors' products work and from legal professionals wanting to prove that their clients' patents were infringed or that their devices were copied in their entirety. Reverse engineering companies also consult on security issues for highly security-conscious industries like financial, pay television, and smart cards. What this really means is one-stop shopping for all things security. The same companies can help you protect your design, help your competitors break through that protection, and help you reverse engineer their new product in order to sue them for infringement. Chances are these companies even invented many of the security threats you're working to protect yourself from in the first place.

In the US, the best legal protection for your designs (which arguably isn't much) probably comes through the patent process. Ironically, the patent process essentially requires you to reverse engineer your own product and publish the results in exchange for the possible protection of law for your invention. The law is specifically designed so that people can understand your invention and try to one-up you by making something better that doesn't violate your patent. As we've discussed before, patent enforcement can also be tricky, unreliable, slow, and you won't necessarily recoup the cost of defending your invention even if you win.

In short, if you really want to protect your stuff, you probably have to rely on technical means. As we explained last week – many of the decisions regarding how and how much to protect your design are economic. You need to balance how much you'll spend implementing security features, how much you'll lose if someone copies your design, and how much inconvenience for your customers (and support burden for you) might be caused by your security measures. You'll also want to analyze the potential reliability and manufacturability impact of any security features you choose to incorporate.

Before we start building walls and fences, it pays to ask ourselves – "Who are the bad guys?" Obviously, we're the good guys wearing the white hats. We want to create something beautiful, (…like maybe a hardware accelerator for cracking encryption algorithms… Oh wait! That's a bad example...) and then make sure nobody steals it. Wearing grey hats, perhaps, are the companies that both create and defeat security hazards for a living. They're an inevitable consequence of the value of technology. But who exactly is wearing the black hats? We need to understand the villains, the bad guys, these diabolical digital ne'erdowells who want to steal our hard-designed logic for their own insidious purposes.

A well-known classification system proposed in 1991 in the IBM Systems Journal says the bad guys fall into one of three basic categories: Class I

– Clever Outsiders, Class II – Knowledgeable Insiders, and Class III – Funded Organizations. We all know Class I. He was the guy in the dorm room "next to ours" in EE or CS school. His primary motivation for cracking security was ego. He wanted to prove that he was smart enough to pull it off. Fortunately for everyone, his budget only amounted to what he could earn selling back his Mountain Dew cans at the grocery store. If he got really good – he got hired into Class II or III.

Class II is what most of us developing commercial applications are facing. Even Class II attackers can be quite well funded. They tend to be financially motivated, however, so if we can make breaking our security a non-profitable enterprise, we've conquered the bulk of Class II. Class II attackers will make use of state-of-the-art techniques such as "decapping" a chip and analyzing it with focused ion beams, thermal imaging, and other techniques. Non-invasive methods can also be used such as dynamic power analysis (DPA) - performing statistical signal processing on plots of power consumption data to find signatures of encryption keys loading into registers.

They also frequently attack the most vulnerable part of any security system – the human. Exploiting the human element is what has been called "rubber-hose cryptanalysis" – a somewhat tongue-in-cheek term for torturing an insider to obtain crypto keys or other enabling information. Typically, rather than resorting to the rubber hose, Class II attackers rely on a much more practical (and legal) technique – corporate attrition. With Silicon Valley job tenures typically measured in months rather than years and corporate allegiances shifting as fast as stock options expire, so-called "security by obscurity" is practically impossible. If your system doesn't correct for the former employee that now works for a competitor, you truly have no security at all.

Class III is a different story. These folks usually work for governmental agencies, have practically unlimited budgets and patience, and are motivated by forces like fear and paranoia. If you think your security is good enough to foil Class III, you're probably wrong… and that's exactly what they want you to think. Caveat hosee.

ASIC designs are among the most secure. Although companies like the aforementioned will happily delaminate, deconstruct, and analyze an ASIC design for you, (or for your competitors) it's an expensive process. With a little cleverness, or a little help from friends like Cryptography Research, you can make the reverse engineer's job a lot tougher. "There are a number of things we help customers do to make their ASICs more secure," says Benjamin Jun, vice president of technology at Cryptography Research. "Depending on the design and the customer's objectives, we have recommended a variety of approaches." Techniques like foregoing the top layer of metal in order to put a shield in place, operating without scan, and adding encryption IP to the design are typically in the arsenal of the ASIC security conscious.

If your goal is storing data securely in your ASIC, consider technology from Kilopass, whose patented non-volatile storage technology works with standard CMOS processes and relies on a gate-oxide-based fuse to retain information. The company claims that the programming state of their storage element is almost impossible to observe with typical reverse-engineering techniques and that their storage is compact enough to be practical for keeping critical data on-chip in a SoC ASIC.

As we mentioned last week, FPGAs represent both a security risk and a potential place to secure your design. In fact, there is considerable controversy of late on the best scheme for securing your FPGA-based IP. In our research, we have yet to find evidence that any of the main FPGA schemes has been defeated, so without passing too much judgment on the vendors' claims, let's look at the technology behind the various approaches.

First, the most common technologies associated with design security in FPGAs are non-volatile devices like antifuse and flash FPGAs. Antifuse FPGAs (like those made by Actel and Quicklogic) are widely regarded as the most secure. Antifuse states are almost impossible to discern, given that a tiny whisker of metal establishes the connection and that invasive inspection can easily break that whisker. The challenge is compounded by the fact that only a tiny percentage of the antifuses on any given die are actually involved in the configuration of the device. Potentially, millions of antifuses would have to be correctly read in order to reverse engineer an antifuse FPGA configuration.

Flash FPGAs also store their configuration without relying on an external memory. Again, the state of flash cells is extremely difficult to discern, although techniques do exist. Once again, in order to determine the configuration for an FPGA, potentially millions of flash elements would have to be correctly read by the attacker, and a very difficult task would still remain of determining which of those represented configuration information. Two companies, Actel and Lattice, market flash-based FPGAs (three if you count Altera's Max II CPLD as an FPGA).

SRAM devices have long been regarded as less secure because they store their configuration bitstream in an external memory. In an unsecured scenario, it is reported that a second grader equipped with nothing more than crayons and bailing wire can copy the design. We don't believe these claims (unless the second grader also has access to a logic analyzer, then we're good to go.)

The axis of the current controversy, however, is defined by the apparently polar approaches taken by archrivals Xilinx and Altera in securing the bitstreams in their high-end SRAM devices. Both companies rely on encrypting the bitstream data stored in external memory and then decrypting it using a key stored in the FPGA itself. There the similarity ends.

Xilinx stores the encryption keys in volatile memory and keeps that memory active with an external battery. If the battery is disconnected, the keys are lost, and the FPGA returns to its native, blank state. Depending on your needs, this could potentially be a good thing or a bad thing. If you're primarily worried about system reliability with security as a secondary concern, long-term dependence on a battery soldered into the system may be reason for concern. If security is your primary goal, you'll be happy to know that your keys are erased (and thus protected) as soon as someone tries to disconnect the device for detailed inspection.

Altera's recently announced strategy relies on non-volatile keys stored in poly-fuses on the FPGA. Since the keys are not volatile - with enough time and money (the order of magnitude of each being the subject of considerable debate), the key values could eventually be determined through invasive inspection by a determined attacker. Altera claims that there are considerable measures designed in to prevent such an attack from succeeding. We'd guess that might include hiding the bits at undisclosed locations on the die, burying them under layers of metal, and performing additional processing on the data so the raw keys are not stored directly in the fuses.

Advantages of the non-volatile approach, of course, are that the configurability of the device doesn't rely on an external battery. This means that an FPGA with encryption keys loaded could not be re-used for another purpose without knowledge of those encryption keys. This also means that an in-system device could not be reprogrammed with an entirely different bitstream designed by the attacker – a potential problem in the Xilinx scheme if your goal is more tamper-resistance than security.

Both schemes might theoretically be vulnerable to techniques such as DPA or EM monitoring, as the keys are XORed with the corresponding data inbound from the external memory, but the complexity of FPGAs and the amount of both EM and power noise (including possible noise injected for security purposes) would make such an attack very difficult.

In the end, you as the engineer are responsible for determining the type and amount of security required for your design. Remember to trust your own engineering expertise over marketing claims of vendors, and do your homework. Also remember to balance the complete list of considerations including the cost of protection – in ways such as IP licensing, consulting, processing time, convenience for customers, reliability, supportability, and manufacturability, the cost of a security failure, and the likelihood of an attack. Without examining all of these variables, you're likely to end up with dangerous tunnel vision on issues like whether a security crack is theoretically possible rather than the engineering practicality of the whole problem.

by Kevin Morris, FPGA and Structured ASIC Journal

June 27, 2006