

# TRUSTWORTHY HARDWARE: IDENTIFYING AND CLASSIFYING HARDWARE TROJANS

Ramesh Karri and Jeyavijayan Rajendran, *Polytechnic Institute of New York University*

Kurt Rosenfeld, *Google*

Mohammad Tehranipoor, *University of Connecticut*

**For reasons of economy, critical systems will inevitably depend on electronics made in untrusted factories. A proposed new hardware Trojan taxonomy provides a first step in better understanding existing and potential threats.**

**F**or decades, digital systems have been designed based on assumptions that the underlying hardware, though not perfectly reliable, is free of malicious elements. This assumption is increasingly questionable. Throughout the digital hardware industry, outsourced alternatives have gradually replaced in-house processes. Commercial software has supplanted homegrown CAD software. Third-party intellectual property (IP) cores have displaced in-house libraries of logic cells for synthesis. Mask production and fabrication are being outsourced, along with testing of the devices after manufacturing. Having relinquished so much control in the interest of economy, how can we establish a basis for trust in the systems we construct?

According to a 2005 Defense Science Board study, the US Department of Defense's integrated circuit (IC) consumption is only about one to two percent of global

consumption.<sup>1</sup> Although this is a small fraction of the commercial market, these ICs are essential to national security. Having a separate, "secure" supply chain for such chips is desirable but economically prohibitive. Inevitably, critical systems will depend on electronics made in untrusted factories.

A breakdown in security can occur at various stages in the electronics supply chain, not just at the factory. An FBI investigation in 2004-2006 found counterfeit Cisco routers in US defense, finance, and university networks.<sup>2</sup> These fakes had low manufacturing quality and high field failure rates. What made it so easy for counterfeits to pollute the supply chain? The FBI determined that US companies were procuring these electronics directly or through intermediaries from untrustworthy sources in foreign countries; they were also buying online from auction sites and resellers. Although the infiltrators' motivation appears to have been simply to make money, the example vividly illustrates the vulnerability of typical supply chains.

Concern about the low quality of counterfeit electronics is secondary to the security threat possessed by vulnerabilities in the supply chain. Examples of possible malicious activities conducted by inauthentic ICs include disabling a system by acting as a silicon time bomb, leaking secrets using a side channel, or providing an adversary with remote control over a system. As "design for manufac-



## ⇒ HARDWARE TRUST

Several areas of study inform the field of hardware trust.

*Reliability and fault tolerance* examines how to engineer systems so that the natural failure of individual components does not bring down the whole system. One example is the majority voter scheme used to build fault-tolerant computing systems. In designing critical software such as flight control systems, designers use N-version programming. In the case of the Boeing 777, they program the system independently three times and run the three versions concurrently in the plane on three computers with different microarchitectures.<sup>1</sup> Fault-tolerant techniques are effective against natural failures and mistakes, but less so at withstanding deliberate faults. An intelligent attacker can focus on the non-fault-tolerant parts of the system, such as the majority voter circuit itself.

Hardware trust also draws heavily from *design for test*. DFT provides a set of design techniques that enables efficient verification that a digital system is free of manufacturing defects. If there are defects, the DFT structures help engineers determine their location and nature. DFT's principal goals are controllability and observability of internal signals. *Automatic test pattern generation* (ATPG) algorithms assume a model of the faults that can result from flaws in manufacturing and produce test patterns that will detect those faults if they are present in chips or systems. *Test coverage* is the fraction of faults within the fault model that a given set of tests can detect. Typical models used in the IC industry focus on naturally occurring faults such as broken wires. Hardware Trojans lie outside the usual fault model. Consequently, a system can pass a set of tests that has 100 percent coverage for naturally occurring faults but can still contain a hardware Trojan.

*Tamper resistance* is another area of hardware trust. Designers of tamper resistant devices assume that an adversary aims to modify an object in some way without destroying it. Closely related is the notion of *tamper-evident* objects, which assumes that an adversary aims to modify without detection. Making a system tamper-evident is usually easier than making it tamper-resistant. Efforts at tamper resistance can raise the adversary's cost of executing a tampering attack but cannot negate it.<sup>2</sup> Various tamper-resistant computing devices have been deployed over the years, ranging from \$1 trusted-platform modules to the \$2,000 IBM 4758 Cryptographic Coprocessor.<sup>3</sup>

*Logic verification* aims to determine whether a design meets a specification. This class of techniques was developed to catch accidental errors that occur during logic design. Logic verification does not address malicious modifications at the specification phase, nor does it address modifications made during fabrication.

*Side channels* let an adversary gain information about a system's internal states without defeating its access control mechanisms. For example, the electrical current that a microprocessor consumes is affected by the operations it performs and the operands involved. An attacker who can monitor the current waveform can perform power analysis<sup>4</sup> to learn the cryptographic key being used. Design techniques to avoid side-channel information leakage are available, but they rely on faithful fabrication of the system.

In the field, systems are vulnerable not only to passive observation but also to active techniques such as *fault injection*.<sup>5</sup> For example,

heating nonvolatile memory containing a key somewhat above its rated temperature range may erase the key, causing all key bits to become ones. If this is done slowly, the bits can change, one by one. An attacker who can observe the faulty computation at each stage while the memory is being corrupted can gain information about the original key. Fault attack countermeasures exist<sup>5</sup> but are mostly design-level techniques and rely on trustworthy fabrication.

*Watermarking* protects against unauthorized copying of designs.<sup>6</sup> The techniques are designed to be robust—small changes in the watermarked logic should not cause the watermark to fail detection. Unfortunately, this robustness makes watermarks unsuitable for protecting against Trojans inserted during fabrication, which requires sensitivity to small changes.

*Reverse-engineering* starts with a device such as a chip and documents its design, how it works, what it does, and how it is made. Because the IC reverse-engineering process involves analyzing chips layer by layer under a microscope, which is very expensive, reverse-engineering entire chips to check for Trojans usually is not practical.

In some applications, defending against *cloning* is important. In a stored-value card system, for example, card cloning generates value for the attacker while depriving the system's operators of their income. In access systems that use physical-token-based authentication, cloning of the token lets an attacker access a building, for example, after returning the access card. This threat led to the development of physical unclonable functions.<sup>7</sup> A PUF is a low-cost hardware module that takes in a challenge and returns a response. The challenge-response mapping is unclonable and unique to each PUF example, making PUFs suitable for authentication and key generation. PUF security depends on correct fabrication. No amount of black-box testing can distinguish a genuine PUF from a maliciously modified one—an attacker who replaces a PUF with a message authentication code and chooses the MAC key can create a challenge-response mapping that is predictable to the attacker but seemingly random to others.

### References

1. Y.C. Yeh, "Triple-Triple Redundant 777 Primary Flight Computer," *Proc. 1996 IEEE Aerospace Applications Conf.*, vol. 1, IEEE Press, 1996, pp. 293-307.
2. R. Anderson and M. Kuhn, "Tamper Resistance: A Cautionary Note," *Proc. 2nd Usenix Workshop Electronic Commerce*, vol. 2, Usenix Assoc., 1996, pp. 1-11.
3. J.G. Dyer et al., "Building the IBM 4758 Secure Coprocessor," *Computer*, Oct. 2001, pp. 57-66.
4. P.C. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis," *Proc. 19th Ann. Int'l Cryptology Conf. Advances in Cryptology (Crypto 99)*, LNCS 1666, Springer-Verlag, 1999, pp. 388-397.
5. H. Bar-EI et al., "The Sorcerer's Apprentice Guide to Fault Attacks," *Proc. IEEE*, Feb. 2006, pp. 370-382.
6. G. Qu and M. Potkonjak, *Intellectual Property Protection in VLSI Design: Theory and Practice*, Kluwer Academic, 2003.
7. S. Devadas et al., "Design and Implementation of PUF-Based 'Unclonable' RFID ICs for Anti-Counterfeiting and Security Applications," *Proc. IEEE Int'l Conf. RFID*, IEEE Press, 2008, pp. 58-64.

turability" and "design for testability" were past mantras, *design for trust* must be the new mantra. The "Hardware Trust" sidebar gives an overview of research aimed at increasing hardware trustworthiness.

To be trustworthy, hardware must

- exhibit only the functionality for which it was designed, nothing more and nothing less;

- conceal any information about the computation performed through any side channels such as power and delay; and
- be transparent only to the designer while remaining opaque to others, who should know nothing about its design and internal states.

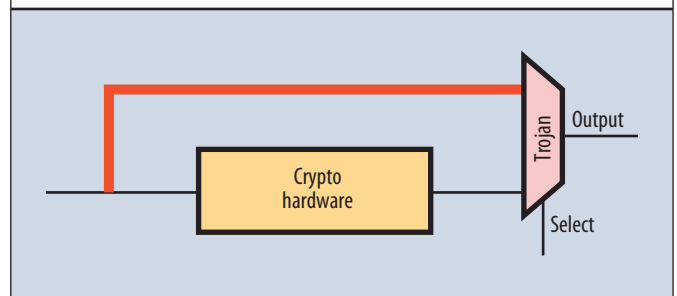
As a first step in creating a roadmap for trusted hardware, we must strive to better understand the threats, among the greatest of which are *hardware Trojans*. A hardware Trojan is a malicious and deliberately stealthy modification made to an electronic device such as an IC. It can change the chip's functionality and thereby undermine trust in the systems using that trojaned chip. In the simple example shown in Figure 1, the Trojan in the datapath is a single multiplexer. During normal operation, the chip sends encrypted data through the output. When the Trojan is active, it bypasses the crypto module and sends plaintext to the output, betraying the chip designer's intentions.

### HARDWARE TROJAN TAXONOMY

Developing a better understanding of hardware Trojans and creating effective defenses requires a framework that groups similar Trojans together to enable a systematic study of their characteristics. Detection, mitigation, and protection techniques can then be developed for each Trojan class along with benchmarks to serve as the basis for comparing countermeasures. In addition, experimental implementations can be created for Trojan classes yet to be observed in the wild, thereby fostering proactive defense.

The framework should provide terminology and descriptive names for each class. Thus, the hardware Trojan taxonomy must meet two key criteria:

- coverage—it should classify all Trojans; none should exist "outside" the taxonomy; and

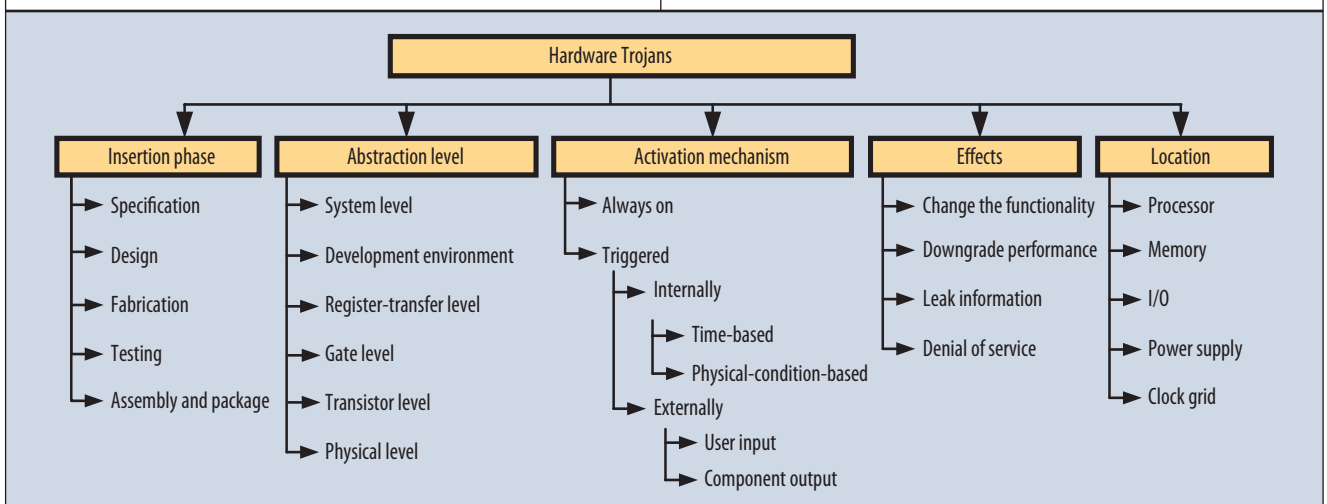


**Figure 1. Simple Trojan.** During normal operation, the chip sends encrypted data to the output. When the Trojan is active, the chip bypasses the crypto module and sends plaintext to the output, betraying the chip designer's intentions.

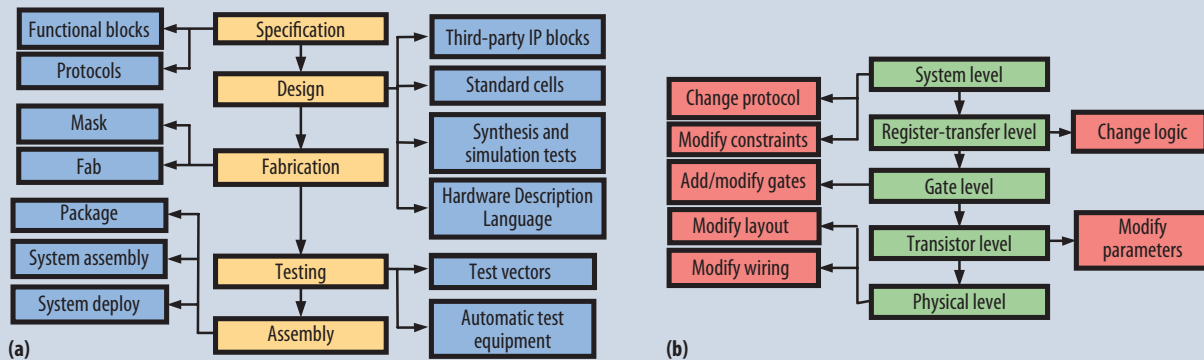
- *resolution*—it should distinguish Trojans with significantly different capabilities or required countermeasures.

Several researchers have proposed a taxonomy based on hardware Trojans' physical, activation, and functional characteristics.<sup>3,4</sup> Trojan taxonomies that group Trojans based on their triggering and leaking mechanisms have also been developed. Still other researchers have proposed classifying hardware Trojans according to their damage objectives, attack components and mechanisms, insertion and mechanism phases, and triggering mechanism.<sup>6,7</sup>

All of these taxonomies assume that hardware Trojans are inserted only at the fabrication phase; however, they can be inserted at other phases and have different functionalities. Hence, we propose a new taxonomy that has a broader set of attributes. As Figure 2 shows, we classify hardware Trojans according to the *insertion phase*—the point at which alteration takes place; *abstraction level* at which the alteration occurs; *activation mechanism*; *effects*; and *location*.



**Figure 2. Proposed hardware Trojan taxonomy based on five different attributes.**



**Figure 3. IC development cycle and hardware abstraction levels. (a) A chip's design is vulnerable to modifications throughout the development cycle. The center boxes show the different phases, while the outer boxes show possible vulnerabilities. (b) Trojan circuits can be inserted at various hardware abstraction levels. The center boxes show abstractions in a top-down VLSI design flow, while the red boxes show example Trojans on that level.**

### Insertion phase

Figure 3a shows the typical development cycle of an IC. The design is vulnerable to modifications throughout the cycle.

**Specification phase.** In this phase, chip designers define the system's characteristics: the target environment, expected function, size, power, and delay. While the IC's development is in this phase, functional specifications or other design constraints can be altered. For example, a Trojan at the specification phase might change the hardware's timing requirements.

**Design phase.** Developers consider functional, logical, timing, and physical constraints as they map the design onto the target technology. At this point, they can use third-party IP blocks and standard cells. Trojans might be in any of the components that aid the design. For example, a standard cell library can be infested with Trojans.

**Fabrication phase.** During this phase, developers create a mask set and use wafers to produce the masks. Subtle mask changes can have serious effects. In an extreme case, an adversary can substitute a different mask set. Alternatively, chemical compositions might be altered during fabrication to increase the electromigration in critical circuitry such as power supplies and clock grids, which would accelerate failures.<sup>8</sup>

**Testing phase.** The IC testing phase is important for hardware trust not because it is a likely phase for Trojan insertion but because it provides an opportunity for Trojan detection. Testing is only useful for such detection if trustworthy. For example, an adversary who inserted a Trojan in the fabrication phase would want to have control over the test vectors to ensure that the Trojan is not detected during testing. Trustworthy testing ensures that the test vectors will be kept secret and faithfully applied, and that the specified actions—accept/reject, binning—will be faithfully followed. Additionally, an attacker can identify

*strategic faults*, which, if they were to occur, would give the attacker an advantage. Here, an attacker can either omit the test vectors used to detect the strategic faults or he can craft the design so that these faults are untestable.

**Assembly phase.** Developers assemble the tested chip and other hardware components on a printed circuit board (PCB). Every interface in a system where two or more components interact is a potential Trojan insertion site. Even if all the ICs in a system are trustworthy, malicious assembly can introduce security flaws in the system. For example, an unshielded wire connected to a node on the PCB can introduce unintended electromagnetic coupling between the signal on the board and its electromagnetic surroundings. An adversary can exploit this for information leakage and fault injection.

### Abstraction level

Trojan circuits can be inserted at various hardware abstraction levels, as Figure 3b shows.

**System level.** At this level, chip developers define the various hardware modules, interconnections, and communication protocols used. Trojans might be triggered by the target hardware modules—for example, interchanging the ASCII values of the keyboard inputs.

**Development environment.** A typical IC development tool chain includes synthesis, simulation, verification, and validation tools. An attacker can use CAD tools and scripts to insert Trojans.<sup>9</sup> Software Trojans inserted into these CAD tools can mask the effects of the hardware Trojans. For example, a synthesis tool might not reveal a circuit's Trojan components to the user.

**Register-transfer level.** At the RTL, chip developers describe each functional module in terms of registers, signals, and Boolean functions. A Trojan can be easily designed and inserted at the RTL because the attacker has full control over the hardware's functionality. For exam-

ple, a Trojan implemented at this level might halve the rounds of a cryptographic algorithm by making a round counter to advance in two steps instead of one.

**Gate level.** At this level, an IC is represented as an interconnection of logic gates. An attacker can carefully control all aspects of the inserted Trojan, including its size and location. For example, a Trojan might be a simple comparator consisting of XOR gates that monitor the chip's internal signals.

**Transistor level.** Chip designers use transistors to build logic gates. This level gives the Trojan designer control over circuit characteristics like power and timing. The attacker can insert or remove individual transistors, altering the circuit functionality, or modify transistor sizes to alter circuit parameters. For example, a transistor-level Trojan might be a transistor with low gate width that can cause more delay in the critical path.

**Physical level.** This level describes all circuit components and their dimensions and locations, and is the design's physical level—where a Trojan can be inserted. An attacker can insert Trojans by modifying the size of the wires and distances between circuit elements and reassigning metal layers. For example, changing the width of the clock grids' metal wires in the chip can cause clock skew.

### Activation mechanism

Some Trojans are designed to be *always on*, while others remain dormant until *triggered*. A triggered Trojan needs an internal or external event to be activated. Once the trigger activates a Trojan, it can remain active forever or return to a dormant state after a specified time.

An event that occurs within the target device activates an *internally triggered* Trojan. The event might be either *time-based* or *physical-condition-based*. A counter in the design can trigger a Trojan at a predetermined time, resulting in a time bomb. Similarly, a Trojan can trigger when the chip temperature exceeds 55°C.

An *externally triggered* Trojan requires external input to the target module to activate. The external trigger can be *user input* or *component output*. User-input triggers include pushbuttons, switches, keyboards, or keywords and phrases in the input data stream. Component-output triggers might be from any of the components that interact with the target device. For example, data coming through external interfaces such as RS-232 can trigger a Trojan.

### Effects

Trojans can also be characterized by their undesirable effects. The severity of these effects on target hardware or systems can range from subtle disturbances to catastrophic system failures.


A Trojan can *change the functionality* of the target device and cause subtle errors that might be difficult to detect. For

example, a Trojan might cause an error detection module to accept inputs that should be rejected.

In addition, a Trojan can *downgrade performance* by intentionally changing device parameters. These include functional, interface, or parametric characteristics such as power and delay. For example, a Trojan might insert more buffers in the chip's interconnections and hence consume more power, which in turn could drain the battery quickly.

A Trojan can also *leak information* through both covert and overt channels. Sensitive data can be leaked via radio frequency, optical or thermal power, timing side channels, and interfaces such as RS-232 and JTAG (Joint Test Action Group). For example, a Trojan might leak a cryptographic algorithm's secret key through unused RS-232 ports.

*Denial-of-service* Trojans prevent operation of a function or resource. A DoS Trojan can cause the target module to exhaust scarce resources like bandwidth, computation, and battery power. It could also physically destroy, disable, or alter the device's configuration—for example, causing the processor to ignore the interrupt from a specific peripheral. DoS can be either temporary or permanent.



**The effects of Trojans on target hardware or systems can range from subtle disturbances to catastrophic system failures.**

### Location

A hardware Trojan can be inserted in a single component or spread across multiple components—the processor, memory, input/output, power supply, or clock grid. Trojans distributed across multiple components can act independently of one another or together as a group to accomplish their attack objectives.

**Processor.** Any Trojan embedded into the logic units that are part of the processor can be grouped under this category. A Trojan in the processor might, for example, change the instructions' execution order.

**Memory.** Trojans in the memory blocks and their interface units fall in this category. These Trojans might alter the value stored in the memory and also block read or write access to certain memory locations—for example, change the contents of a programmable read-only memory in an IC.

**I/O.** Trojans can reside in a chip's peripherals or within the PCB. These peripherals interface with the external components and can give the Trojan control over data communication between the processor and the system's external components. For example, a Trojan might alter the data coming through an RS-232 port.

**Power supply.** Trojans can alter the voltage and current supplied to the chip, causing failures.

## → EMBEDDED SYSTEMS CHALLENGE

The Embedded Systems Challenge ([www.poly.edu/csaw-embedded](http://www.poly.edu/csaw-embedded)) is an annual hardware hacking competition organized by Polytechnic Institute of New York University. Students of all levels from universities across the country are invited to compete. For the past three years, the challenge has focused on embedding and detecting hardware Trojans. Participants received the Hardware Description Language (HDL) code for a reference design electronically and a field-programmable gate array board and development tools by mail. As a sponsor, Xilinx provided FPGA boards for the 2008, 2009 and 2010 competitions.

The Alpha reference design for the 2008 challenge ([www.isis.poly.edu/esc08.pdf](http://www.isis.poly.edu/esc08.pdf)) was a cryptographic encoder for interactive use by a human operator. The design targeted the Digilent Basys board, which contains a Xilinx Spartan-3 FPGA. The keyboard is used as an input device for plaintext, and a video port is used as an output device for ciphertext. The cryptographic keys were hardcoded in the Alpha's HDL code. The design filled the target FPGA's hardware resources.

The challenge's requirements were to embed one or more Trojans that leak a key, leak plaintext, or create a DoS—all without being detected during normal testing. The participants fully documented their Trojans and demonstrated some of them at the final judging held at NYU-Poly. A panel of independent judges from industry scored the teams based on their submissions' effectiveness and inventiveness. This set of submissions served as a sample for studying and classifying Trojans.

After the 2008 challenge ended, we collected some statistics on the submitted Trojans and found that 59 percent were triggered on user input, 34 percent were always on, 4 percent were time bombs, and the rest were mostly triggered by physical conditions such as temperature. Some Trojans were triggered by a single character in the input stream; others required a sequence of characters. Some added timing constraints on the symbols' arrival, further reducing the likelihood of detection during normal testing or operation. Some Trojans made use of secondary inputs such as an input pin that is part of the VGA output port.

Half of the submitted Trojans changed the system's functional behavior, while 36 percent only leaked information. The remaining 14 percent created various DoS conditions. The most frequently observed method of leaking information involved using regular I/O ports, but other side channels were also used. One Trojan slowly leaked the key thermally by deliberately dissipating extra power to transmit a 1 and regular power to transmit a 0. Another Trojan deliberately emitted radio-frequency signals that encoded the key.

**Clock grid.** Trojans in the clock grid can change the clock's frequency, insert glitches in the clock supplied to the chip, and launch fault attacks. These Trojans can also freeze the clock signal supplied to the rest of the chip's functional modules. For example, a Trojan might increase the clock signal skew supplied to specific parts of a chip.

### TAXONOMY VALIDATION

To validate our proposed hardware Trojan taxonomy, we tested whether it meets the coverage and resolution criteria described earlier. We used the 38 Trojans submitted to the 2008 Embedded Systems Challenge, described

in the "Embedded Systems Challenge" sidebar, along with 18 Trojans identified in previous work.

As Table 1 shows, the taxonomy covers all 56 observed Trojans. To determine whether the taxonomy meets the resolution criterion, we examined the Trojans to determine whether they can be classified according to significantly different abilities or required countermeasures. In all, the Trojans occupy 17 classes in the taxonomy. The vast majority—37 of 56—are inserted in the design phase and triggered by user input, and among these we were able to distinguish 12 classes (2-13 in Table 1).

The same countermeasure can detect Trojans in the same class. For example, checking the RTL code of the I/O unit for changes in the I/O protocol can detect Trojans in class 2. Performing exhaustive memory testing can detect Trojans in class 3. Analyzing the side channels can detect Trojans in class 4. Exhaustive checking for resource-utilization changes can detect Trojans in class 5. Periodically communicating with the device, even after its deployment, can detect Trojans in class 6. Skewing the clocks and observing the IC's transient behavior can detect Trojans in class 7. Checking for transient characteristics while dynamically scaling the chip's supply voltage can detect Trojans in class 8. Concurrent detection for soft errors can detect Trojans in class 9.

Sometimes a generic countermeasure can detect Trojans from different classes. For example, checking for side-channel interfaces to the processor unit can detect Trojans in classes 10 and 12. Checking the chip's "fingerprints," such as power and delay characteristics, can enable Trojan detection in classes 14 through 17.

Table 1 also lists potential Trojan classes for which examples have yet to be reported. These classes are as important as those with known examples and should be considered while designing benchmarks and countermeasures. For example, a proactive countermeasure for class 24 Trojans is complete verification of interface protocols for possible trapdoors; a proactive measure to detect class 25 Trojans is to periodically check the peripherals' integrity.


**H**ardware Trojans present a very real threat. Our proposed taxonomy can serve as a useful tool to study existing Trojans and to develop systematic Trojan detection methods. Equally important, it provides a view of potential hardware Trojans so that proactive hardware security measures can be taken today to protect against the attacks of tomorrow.

The taxonomy presented here captures the spectrum of Trojans we have seen at the time of writing. To broaden the spectrum of our awareness, we are organizing an upcoming hardware hacking competition ([www.poly.edu/](http://www.poly.edu/)

**Table 1. Hardware Trojans used to verify proposed taxonomy.**

Trojan status	Class no.	Trojan class	No. of Trojans
Observed	1	Specification phase-RTL-external-component triggered-leak information-in the I/O	1
	2	Design phase-RTL-user-input triggered-leak information-in the I/O	12
	3	Design phase-RTL-user-input triggered-change function-in the memory	2
	4	Design phase-RTL-user-input triggered-leak information-in the processor	2
	5	Design phase-RTL-user-input triggered-permanently deny service-in the processor	2
	6	Design phase-RTL-user-input triggered-permanently deny service-in the I/O	1
	7	Design phase-RTL-user-input triggered-permanently deny service-in the clock grid	1
	8	Design phase-RTL-user-input triggered-permanently deny service-in the power supply	1
	9	Design phase-RTL-user-input triggered-temporarily deny service-in the processor	1
	10	Design phase-RTL-always on-leak information-in the processor	4
	11	Design phase-RTL-always on-leak information-in the I/O	9
	12	Design phase-RTL-physical-parameter triggered-permanently deny service-in the processor	1
	13	Design phase-RTL-time triggered-temporarily deny service-in the I/O	1
	14	Fabrication phase-transistor level-user-input triggered-change function-in the processor	12
	15	Fabrication phase-transistor level-always on-change function-in the processor	4
	16	Fabrication phase-transistor level-time triggered-change function-in the processor	1
	17	Fabrication phase-physical level-always on-change function-in the processor	1
Potential	18	Specification phase-system level-user-input triggered-change function-in the processor	—
	19	Specification phase-system level-time triggered-temporarily deny service-in the clock grid	—
	20	Design phase-RTL-physical-parameter triggered-change function-in the processor	—
	21	Design phase-RTL-physical-parameter triggered-permanently deny service-in the memory	—
	22	Design phase-RTL-time triggered-change function-in the I/O	—
	23	Design phase-RTL-time triggered-temporarily deny service-in the memory	—
	24	Assembly and package phase-system level-external-component triggered-leak information-in the I/O	—
	25	Assembly and package phase-system level-external-component triggered-permanently deny service-in the power supply	—
	26	Fabrication phase-transistor level-time triggered-permanently deny service-in the clock grid	—
	27	Fabrication phase-transistor level-always on-temporarily deny service-in the clock grid	—
	28	Fabrication phase-physical level-always on-temporarily deny service-in the clock grid	—
	29	Fabrication phase-physical level-physical-parameter triggered-permanently deny service-in the power supply	—

csaw-embedded). By creating an environment where participants are rewarded for new Trojan techniques, we can continually update the taxonomy ahead of the attacks reported in the field.

Like many areas of security, hardware trust requires constant effort and a proactive approach. To foster cooperation and progress in the research community, the Trojans we collected are available through Trusthub ([www.trust-hub.org](http://www.trust-hub.org)). 

### Acknowledgments

The work of Ramesh Karri and his group is supported in part by National Science Foundation grants ECCS-0621856, CNS-

0619741, CNS-0831349, and CNS-0958510 and AFRL grant FA8750-09-1-0146. The authors also acknowledge the contributions of the organizers of ESC 2007 and 2008, Vikram Padman and Efstratios Gavas. The work of Mohammad Tehranipoor was supported in part by National Science Foundation grants CNS-0716535 and CNS-0844995.

### References

1. Defense Science Board, "Defense Science Board Task Force on High Performance Microchip Supply," Feb. 2005; [www.cra.org/govaffairs/images/2005-02-HPMS\\_Report\\_Final.pdf](http://www.cra.org/govaffairs/images/2005-02-HPMS_Report_Final.pdf).
2. J.H. Follett, "CRN Cisco Channel at Center of FBI Raid on Counterfeit Gear," 12 May 2008, CRN.com; [www.crn.com/networking/207602683](http://www.crn.com/networking/207602683).

3. X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting Malicious Inclusions in Secure Hardware: Challenges and Solutions," *Proc. IEEE Int'l Workshop Hardware-Oriented Security and Trust (HOST 08)*, IEEE Press, 2008, pp. 15-19.
4. M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *IEEE Design & Test*, Jan./Feb., 2010, pp. 10-25.
5. Y. Jin and Y. Makris, "Hardware Trojan Detection Using Path Delay Fingerprint," *Proc. IEEE Int'l Workshop Hardware-Oriented Security and Trust (HOST 08)*, IEEE Press, 2008, pp. 51-57.
6. M. Potkonjak et al., "Hardware Trojan Horse Detection Using Gate-Level Characterization," *Proc. 46th Ann. Design Automation Conf. (DAC 09)*, ACM Press, 2009, pp. 688-693.
7. J. Rajendran et al., "Towards a Comprehensive and Systematic Classification of Hardware Trojans," *Proc. IEEE Int'l Symp. Circuits and Systems (ISCAS10)*, IEEE Press, 2010, pp. 1871-1874.
8. S. Adee, "The Hunt for the Kill Switch," *IEEE Spectrum*, May 2008, pp. 34-39.
9. J.A. Roy, F. Koushanfar, and I.L. Markov, "Extended Abstract: Circuit CAD Tools as a Security Threat," *Proc. IEEE Workshop Hardware-Oriented Security and Trust (HOST 08)*, IEEE Press, 2008, pp. 65-66.

**Ramesh Karri** is a professor in the Electrical and Computer Engineering (ECE) Department at Polytechnic Institute of New York University (NYU-Poly). His research interests include trusted hardware design, side-channel attacks and resistant architectures, interaction between security and reliability, fault tolerance, and nanoscale architectures. Karri received a PhD in computer science from the University of California, San Diego. He is a member of the IEEE Computer Society and an associate editor of IEEE Trans-

actions on Information Forensics and Security and ACM Journal on Emerging Technologies in Computing. Contact him at [rkarri@duke.poly.edu](mailto:rkarri@duke.poly.edu).

**Jeyavijayan Rajendran** is a PhD student in the ECE Department at NYU-Poly. His research interests include hardware trust and nanoarchitectures. Rajendran received a BE in electronics and communication engineering from Anna University. He is a student member of IEEE. Contact him at [jrajen01@students.poly.edu](mailto:jrajen01@students.poly.edu).

**Kurt Rosenfeld** is a PhD student in the Computer Science Department at NYU-Poly and an engineer at Google. His research interests include hardware trust and the security and reliability of information systems. Rosenfeld received an MS in electrical engineering from City College of New York. He is a student member of the ACM. Contact him at [kurt@isis.poly.edu](mailto:kurt@isis.poly.edu).

**Mohammad Tehranipoor** is an associate professor in the Department of Electrical and Computer Engineering at the University of Connecticut, where he also directs the Computer-Aided Design and Test Research Lab and the Secure and Trustable Systems Lab. His research interests include computer-reliable nanoscale systems design, secure IP/IC design, hardware security and trust, and design for testability. Tehranipoor received a PhD in electrical engineering from the University of Texas at Dallas. He is a senior member of IEEE and a member of the ACM, and is currently guest editor of *Computing Now* and *IEEE Design & Test*. Contact him at [tehrani@engr.uconn.edu](mailto:tehrani@engr.uconn.edu).

**cn** Selected CS articles and columns are available for free at <http://ComputingNow.computer.org>.

# IEEE Design & Test of Computers



IEEE Design & Test of Computers covers the tools, techniques, and concepts used to design and test electronic product hardware and supportive software. D&T is a leader in analysis of current and near-future practice.

Upcoming: Emerging Interconnect Technologies, New Directions in DFT, Common Language Framework, and Post-Silicon Calibration and Repair

[www.computer.org/design](http://www.computer.org/design)