Int. J. Inf. Secur. (2011) 10:1–14
DOI 10.1007/s10207-010-0115-0

REGULAR CONTRIBUTION

A case study in hardware Trojan design and implementation

Alex Baumgarten · Michael Steffen · Matthew Clausman · Joseph Zambreno

Published online: 3 September 2010

Abstract As integrated circuits (ICs) continue to have an overwhelming presence in our digital information-dominated world, having trust in their manufacture and distribution mechanisms is crucial. However, with ever-shrinking transistor technologies, the cost of new fabrication facilities is becoming prohibitive, pushing industry to make greater use of potentially less reliable foreign sources for their IC supply. The 2008 Computer Security Awareness Week (CSAW) Embedded Systems Challenge at the Polytechnic Institute of NYU highlighted some of the vulnerabilities of the IC supply chain in the form of a hardware hacking challenge. This paper explores the design and implementation of our winning entry.

Keywords Hardware hacking · Trojans · Security · FPGA

1 Introduction

Consider a hypothetical scenario[1] in which a government team is tasked with the design and test of a new cryptographic device, code-named *Alpha*. The Alpha device (which is expected to be in heavy use and hence is of high value to the sponsoring government) allows soldiers to transmit messages securely to other soldiers and their command station. Alpha has returned from the fabrication facility and is waiting for the team's approval to ship. From a testing and verification perspective, when the highly sensitive nature of the device is considered, several complex questions arise. How much confidence does the team have that the Alpha is ready? Even if functional requirements pass, can one be sure that there is no additional logic that is hidden from the tests? How much trust is there in the design chain? What if something malicious has been added?

At the Embedded Systems Challenge, part of the 2008 Computer Security Awareness Week (CSAW) [1] at the Polytechnic Institute of NYU, this exact scenario was proposed. Several student-led teams from around the country assumed the role of the hardware hackers that had been able to gain access to the HDL source code for the Alpha. With this source code in hand, each team had one month to implement as many undetectable hardware Trojans as possible. A hardware Trojan is a malicious modification to the circuitry that compromises the integrity of the original design, usually to the attacker's benefit. The Alpha device, augmented with Trojans, had to pass a set of functional tests, use the same reference power, maintain the configuration memory usage, pass a brief code inspection and be undetectable by a general user. With these requirements in mind, our team from Iowa State University designed a wide set of applicable Trojans and performed a proof-of-concept implementation using a provided Field-Programmable Gate Array (FPGA) board. This paper describes our winning efforts in detail.

2 Competition details

The CSAW Embedded Systems Challenge began on September 2008, concluding a month later in New York. Each team received a BASYS development board [12] containing a Xilinx Spartan FPGA along with basic peripheral I/O. Figure 1 shows a schematic and picture of our experimental

A. Baumgarten · M. Steffen · M. Clausman · J. Zambreno (✉)
Electrical and Computer Engineering,
Iowa State University, Ames, IA, USA
e-mail: zambreno@iastate.edu

[1] While seemingly far-fetched, this hypothetical mirrors real-world events, as is highlighted in Sect. 3.

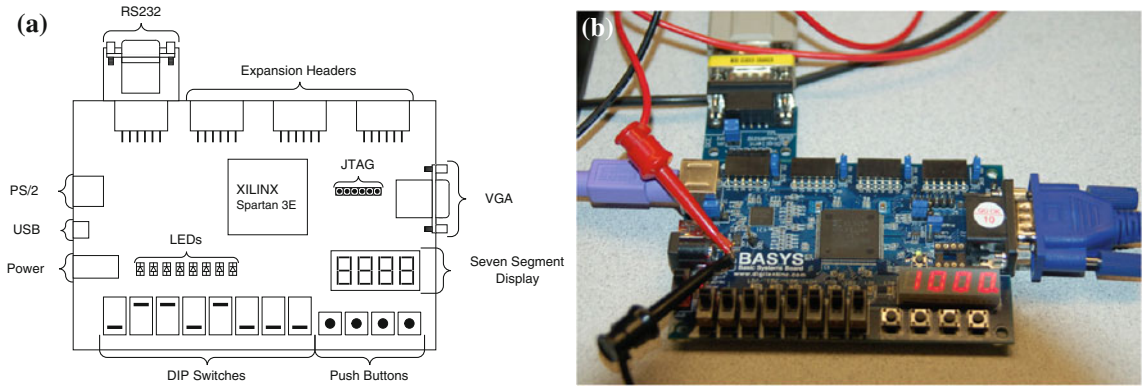

Fig. 1 **a** Diagram of components used from the BASYS board; **b** Experimental setup

setup. The original design of the Alpha required a PS/2 keyboard connected to the board for input and a VGA monitor along with a serial connection for output. Four other buttons changed the state of the system. Xilinx ISE 10.1 was used as the development environment for the mixed-mode HDL code as well as for the generation of the FPGA bitstream while Modelsim SE 6.3 was used for simulation. Additionally, our team made use of an oscilloscope, power supply, multimeter, thermometer, HAM radio, and custom circuits to verify the functionality of each Trojan.

In normal operation of the Alpha device, messages typed in plaintext on the keyboard display on the VGA monitor. Pressing the encrypt button on the board sends the message through an AES-128 encryption block with the key selected by the 5 dip switches.[2] Finally, the transmit button sends the encrypted message over the serial port. The message receiver uses the correct key to decrypt the message. For the purpose of this contest, the organizers provided a C program called encVerifier to receive and decrypt the message. For scoring the designs, the judges evaluated the completeness and uniqueness of the Trojans along with their ability to be undetectable to a set of functional tests, deviation in power consumption, deviation in bitstream size, and their ability to pass a brief code inspection.

3 Background and motivation

3.1 IC supply chain overview

The integrated circuit (IC) supply chain is the process by which a conceptual idea transforms into an IC. The process starts with the IC designer who creates the register transfer level (RTL) description of the IC according to the specifications of the project. This description, typically in a form of a hardware description language (HDL), is the intellectual property (IP) of the designer. When the design stage is complete, the RTL moves into the synthesis stage as it begins a series of electronic design automation (EDA) steps using software tools from companies such as Cadence, Mentor Graphics, and Synopsys in order to produce a finalized layout schematic (i.e. netlist). These steps begin with logic synthesis which converts the RTL design into a directed acyclic graph (DAG) representation. Logic optimization uses complex heuristics to search for graph transformations in order to optimize for a particular metric. Next, mapping heuristics convert the graph into primitive gate representations. Placing and routing give each gate a physical location on the IC and establishes their interconnections. After the synthesis is complete, the foundry receives the synthesized design, a blueprint of the circuit. The foundry can then prepare the masks and tool the machinery. Finally, the completed ICs are distributed to the end users.

In the past, each hardware company could use an in-house, vertical process for both design and fabrication. However, as transistor feature sizes and time-to-market continued to shrink, coupled with the demands for lower-power, high performance ICs, the cost to establish a full-scale foundry became prohibitive (upwards of $4 Billion) for many companies. Because of these factors, the IC supply chain flattened, clarifying the roles of the various parties. Hardware IP vendors emerged who specialize in designing functional units, memories, and bus controllers. These vendors license their technology to others for use in their own IC designs. The IC design companies integrate third-party IP along with their own IP to create an IC design. Finally, contract foundries harness economies of scale as they spread the large capital required to build the foundry among their clients.

The entire supply chain is vulnerable [25], but specific attention must be focused on the HDL and the foundries. In the same way that software describes what the program should do and is vulnerable to malicious descriptions,

[2] There is an obvious cryptographic limitation in using 5 dip switches to select unique 128-bit keys, but conceivably a more elegant model could be used for obtaining the key in a production environment.

the HDL describes the operation of the circuit and is vulnerable to the inclusion of malicious circuitry. An attacker who is able to add circuitry to the design by means of the HDL has limitless potential and flexibility. Since the attack occurs very early in the process, discerning malicious intent becomes very hard to detect.

3.2 Supply chain vulnerabilities

The concerns that arise from IC fabrication fit into three basic categories: *Metering*, *Theft*, and *Trust*. Each of these categories broadly addresses the production of extra ICs beyond the purchase order, unauthorized access to information including the IP, and the tampering of ICs received from the foundry. The 2008 CSAW competition focused on the issue of trust, but a discussion of metering and theft rounds out the background.

3.2.1 Trust

As the trend of "fabless" semiconductor companies continues to increase, so does the trust placed in the hands of the fabrication facilities as they handle the IP of the world's IC supply. Recent military equipment failures around the world bring trust into the forefront as they are being tied to a hidden kill switch [5]. The kill switch, more generally a hardware Trojan, allows remote interruption of functionality of an IC. Other reports confirm that some chip manufacturers purposely implant kill switches in select ICs in order to disable the device if it falls into the wrong hands [5].

Even the United States Department of Defense has assessed this threat through several initiatives. In February of 2005, the Defense Science Board released a report entitled the "Task Force on High Performance Microchip Supply" [10] which examined long-term trust and security in the microchips used by the United States government. Their conclusion: "urgent action is recommended." Following this report, a DARPA initiative, Trust in Integrated Circuits [9], was started to provide more security to the supply chain. The program is broken into three phases with industry and government support at each step. The phases examine trust in ASIC design, trust in untrusted foundries, and trust in FPGA design, respectively. An independent effort with similar intentions, the NSA program on Trusted Foundries [21] sets certain standards of trust that must be established before a foundry can receive a stamp of approval.

The category of trust can be broken by functionality into three areas:

1. Thwarting the user's plans: Affecting the operation of the device either by making it function incorrectly or not function at all. This form of attack is a Denial-of-Service (DOS) where some subset of the functionality fails to work as intended.
2. Gaining extra knowledge: Leaking sensitive information not originally intended to be leaked from the device. This includes allowing a malicious user to capture the plaintext messages, the key used to encrypt those messages or any other information that would allow that gives the malicious user more knowledge about the system then originally intended.
3. Exercising additional functionality: Utilizing the device for a function in which it is not intended to be utilized. In this attack, the device may function correctly, produce the correct outputs, and not leak information, but it still may be used in a manner not intended by the original specifications.

3.2.2 Metering

The second category of concern, metering, deals with the number of ICs produced and for whom. The concept of watermarking has been applied to several technologies including hardware IP [24]. In this case, it is only a partial solution that identifies the IP and not the IC. The passive approach uniquely identifies each IC and registers that identity. Later, suspect ICs are checked for proper registration. The uniqueness derives from a manufacturing variability like the threshold mismatch in MOSFET arrays [19], variability of silicon [20], delay characteristics [18], and Physical Unclonable Functions (PUFs) [27,28] and [26]. The other approach, active metering, locks each IC until the IP holder unlocks it. In [6], each IC generates a unique ID. The ICs begin in a locked state, but the IP holder using an unique ID unlocks the IC. [7] augments this process by adding replicated states in a FSM requiring a similar unlocking mechanism.

3.2.3 Theft

Although this category does overlap with metering, it is not necessarily equivalent. Where metering tries to control who and how many ICs are created, information theft more generally encompasses gaining information not originally intended to be disseminated. The creation of extra ICs by unauthorized users is one of the direct consequences of stealing the netlist, but other information such as hardcoded secrets, IP cores, and algorithms may also be the target.

Many different watermarking and fingerprinting technologies add a unique signature at varying levels to the IP. This signature allows the IP to be traced through the IC supply chain and if stolen, points directly to the source of the leak. The main shortcoming of these approaches is that a unique signature does not protect the IP from theft, except as a deterrent; rather it provides a basis for litigation once the crime has been committed.

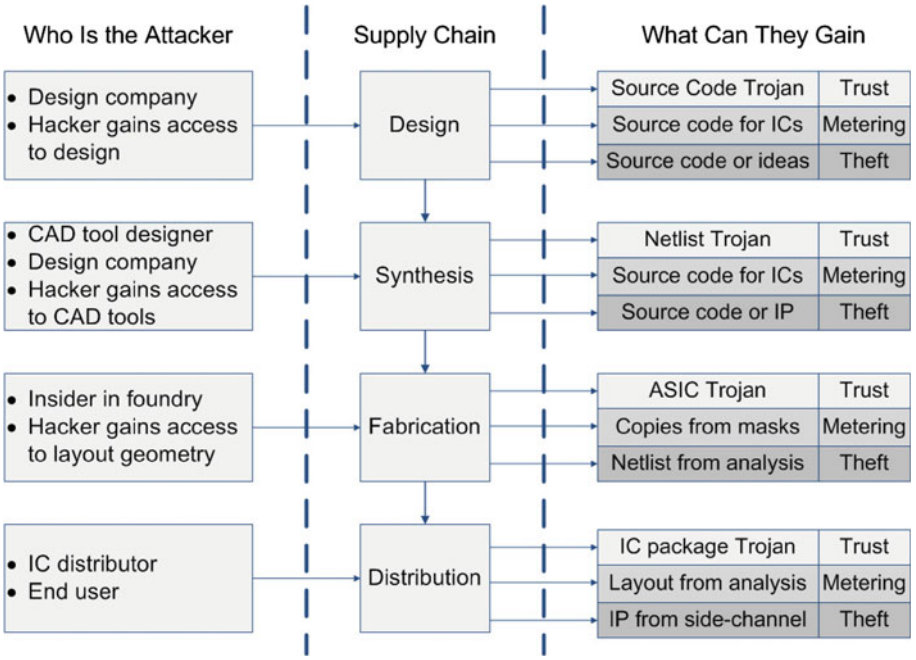

Fig. 2 A taxonomy of an attacker at various levels of the IC supply chain including their motivation and possible protections

The authors in [16,17] target the IP for FPGA's by utilizing the remaining LUTs to encode information. State transition graphs (STG) that exhibit rare patterns are added to a design by [22,23,29]. Extending upon this idea, [2–4], search the existing FSM for useable transitions. Others such as [8,15] utilize the NP-complete design space of the algorithms driving the EDA to provide the flexibility for watermarking.

4 Attacker taxonomy

Attackers can be classified based on their location in the supply chain, as well as their access level. From this assumption, this section describes a list of what an attacker might gain in terms of the previously stated goals and the possible protections to stop or hinder an attacker. In each of the cases, the attacker may be a mole, a disgruntled employee, a competing company, a hacker, or a terrorist. We assume that the attacker has significant resources, but they are limited and that the benefit from attacking the IC supply chain must outweigh the resources expended. Along with the taxonomy descriptions, Fig. 2 shows the stages of the IC supply chain, the attacker at each stage, and the information gained in each of the three attacker goals.

4.1 Design attacker

Who: A design attacker enters the supply chain during the design phase, between the conception of the idea and when the idea has fully materialized in the form of an RTL description. This attacker has full access to the design files and source code. The attacker is likely an insider given the access intentionally although they may also be a more traditional hacker that compromises a computer system.

What is gained:

- Trust - If the attacker has write access, they can add components to or remove components from the design. Without that access, the design can be analyzed in order to facilitate a future attack.
- Metering - With access to the source code, the attacker can create extra ICs with access to the foundry or enough resources to fabricate the ICs.
- Theft - Since the attacker has access to the entire design and source code, IP theft is trivial.

Protection: The measures necessary to protect computer systems that store IP are too numerous to mention and are outside the scope of this paper, but hardened security-conscious networks can aid in IP protection. In the same manner, protecting IP from those who design it is also very difficult. The addition of Trojans may be minimized by careful code reviews and adequate checks and balances or through the use of tools such as TRUTH [11]. The TRUTH tool analyzes HDL source code pointing out potentially unsafe structures that may indicate a Trojan. Protecting from a design attacker requires a holistic security policy in order to minimize the risk.

4.2 Synthesis attacker

Who: A synthesis attacker usually appears well before the IP is actually synthesized. By compromising the computer-aided

design (CAD) tools or the scripts that run them, the attacker can modify the IP at any level from preprocessing the HDL all the way to the generation of a netlist [25]. Since the attack happens during the synthesis phase inside the design house on a usually trusted platform, it is less suspicious and very difficult to discover as the logic is embedded into the design. Also, with the increase in industry acceptance of open-source CAD tools, a synthesis attacker could compromise a system by hosting malicious pre-compiled binaries or direct modification of the source code. Finally, the automated scripts are vulnerable in several ways to an attacker.

What is gained:

- Trust - The attacker can add Trojan logic to the design or cripple critical logic such as a random number generator used for seeding a cryptographic unit.
- Metering - By stealing the IP, the attacker could create extra ICs with the ability to fabricate them.
- Theft - The attacker has access to all levels of the CAD tool flow and can steal any information that exists in the IP.

Protection: Because of the prohibitive cost to design all of the CAD tools in-house, there is a necessary dependence on CAD software creators. A level of trust needs to be established with the creators of the CAD tools and a holistic security policy established to hinder in-house tampering of the CAD tools.

4.3 Fabrication attacker

Who: A fabrication attacker is usually external to the IP designer as contract foundries produce much of the world's ICs. After the IP designer creates, synthesizes, places and routes their design, they generate a physical layout geometry file that is the exact blue print of the IC. In the horizontal business model, the foundry receives the complete design along with its specifications.

What is Gained:

- Trust - The foundries necessarily have exposure to the layout level geometry and masks, which affords them the ability to add or remove components to every IC through layout geometry modification. Alternatively, after the creation of ICs, a select number of ICs may be modified using a Focused Ion Beam (FIB).
- Metering - The foundry produces ICs in large volume and once production begins, creating extra ICs beyond the purchase order is inexpensive and trivial. The non-recurring engineering (NRE) cost paid for by the IC designer is the most costly part of the process. Current IC fabrication practice does not incorporate any measures to limit the number of ICs created by a foundry beyond the use of contractual agreements.
- Theft - With the layout level geometry of the entire design, it is feasible, albeit difficult, to reverse-engineer back to a netlist or even further to HDL.

Protection: Significant research has been done in each of these three areas. For metering, both passive and active schemes have been studied that leave the fabricated IC in a locked state and can meter the number activated instead of the number produced. Theft can be controlled by one of the watermarking approaches discussed previously. Finally, trust in the IC is ensured by either a secure framework within which the IC is created or running a post-fabrication verification tool.

4.4 Distribution attacker

Who: A distribution attacker enters the supply chain after the fabrication and packaging of the ICs. They have access to neither the HDL source code, nor the layout level geometry. This type of attacker is likely to be associated with either the IC distributors or end users. The attacker probably does not have a set of input/output test vectors, but instead a set of specifications to which the IC is supposed to perform.

What is gained:

- Trust - An attacker at this stage is severely limited in their freedom, which raises the granularity of an attack. Instead of being able to deal with individual gates, they must deal with a per-package or possibly per-component level of granularity.
- Metering - To copy the IC requires reverse engineering the design to re-establish a netlist from which ICs can be fabricated. This is considered to be a difficult yet feasible task.
- Theft - Information can be stolen using various methods by either deconstructing the IC or passively observing the IC through side-channel attacks both of which have been extensively researched.

Protection: Besides the difficulties imposed by small feature sizes, many schemes have been implemented in academia and commercially to address specific vulnerabilities of this type. These include anti-tampering packaging, chemical passivation, and obfuscation against side-channel attacks [13].

5 Threat model

Before discussing possible attacks against the Alpha device, the attacker needs to be defined. We assume the attacker

has significant resources temporally, fiscally, and computationally, but that these resources are finite. The attacker is motivated by either profit or a desire to harm the device owner, but the potential must outweigh the cost to insert a Trojan. By the nature of the contest, the attacker enters at the design stage and attempts to compromise the project's trust. The mole has gained access to the source code and a method for reinsertion after it has been modified (which will occur before fabrication). The goal of the attacker is to modify the source code to insert a Trojan that will thwart the user's plans, gain extra knowledge, or exercise additional functionality.

5.1 Constraints

Even though the CSAW contest left the attacker's goal open-ended in order to allow for flexibility and creativity in the solutions, we can define some constraints so that our solutions can be classified. These constraints are not hard requirements imposed by the contest but are a discussion of factors that can be explored.

5.1.1 Proximity

After manufacturing the Alpha device with the Trojan, the Trojan must do something useful for its creator. The proximity of the attacker once the Trojan is active is a key constraint. We classify our solutions into four proximity categories: "physical access", "near the device", "near a communication channel", and "far away". Physical access means that the attacker can physically interact with the device. They could have captured one of the Alpha devices being used in the field, routinely work with the device even in the presence of others, or have momentary access. Near the device means that the attacker can get close enough to interact with the device. This could take the form of some wireless transmission to or from the device or being able to see or hear the device. Although the distance is vague, it only represents a proximity class. Near a communication channel implies that the interaction with the device does not occur directly but rather over some communication channel. An attacker that is near the communication channel, even if the device is around the world, still has the ability to interact with it. This scenario could take the form of message transmissions over the internet or through satellites. Finally, far away means that the attacker does not have access to the Trojan physically or through any of its communication channels. Imagine a Denial of Service (DOS) triggered by an event outside the control of the attacker, the temperature of the room, the number of bits encrypted, etc. For this contest, each solution applied different proximity constraints mentioned during the attack discussion.

5.1.2 Stealth

The Trojan must be well hidden in the HDL code before it gets a chance to be synthesized, and then must remain hidden throughout the testing and operation of the device. The worth of a Trojan is directly connected to its stealth as a discovered Trojan ceases to be of value to the attacker. The contest defined the requirements for the stealth of the HDL code as being able to pass a brief code inspection and for the stealth of the synthesized code as being able to pass a set of functional tests, use the same reference power, maintain the configuration memory usage, and be undetectable by a general user. Although these requirements are subjective, it was left to the judges to score.

Our team spent significant effort hiding the Trojans, in order to pass the code inspection. The Trojan description sections do not describe many of the hiding details, as they deal primarily with specific wires and modules that are not generalizable to other hardware Trojan circuits. In general, methods such as extending bus widths, rerouting signals, utilizing naming conventions, decentralizing the Trojan logic, writing misleading comments, and exploiting language nuances helped hide the various attacks.

The synthesized code remained undetectable to functional tests as none of the Trojans modified the core functionality of the Alpha device.[3] The reference power and memory usage were maintained by decreasing the footprint of the Trojan. By using a trigger, the Trojan could remain dormant and therefore not incur a power penalty. Also, piecing together the Trojan out of built-in functionality had a twofold benefit of increasing HDL stealth and minimizing power and memory consumption. A complementary approach was taken by the second-place team from Yale University, in which the original Alpha code base was aggressively optimized in order to provide slack (in terms of lines of code, logic resources, and power consumption) for the attacker to design and implement various hardware Trojans [14].

5.2 Triggering

Some Trojans begin working on power-up while others remain dormant until explicitly instructed to begin. This instruction is the Trojan's trigger and comes in as many forms as the Trojan itself. The trigger is closely tied to the proximity discussed previously as the attacker governs the trigger. The trigger mechanism is decoupled from the Trojan, allowing the two to be interchanged, since it is only the trigger event that the Trojan is waiting on, not the details of how it was triggered. We focused our efforts on the actual Trojan attacks

[3] The DOS attack by its nature affects the functionality of the device. By using a triggering mechanism, the DOS Trojan can remain dormant until a point in time after the functional tests.

and not on various triggers. Although we added triggers to our Trojans, much more sophisticated triggers could replace our simple ones.

The trigger, like the Trojan, needs to be well hidden to avoid detection, but it also needs to be deliberate so that spurious events do not accidentally set it off. The trigger mechanism must take into account the proximity the attacker will have to the device to determine if the attacker will directly engage the trigger or if another condition will set it off. This could be a time delay or the assertion of a condition that will occur after some time such as number of keystrokes, distance moved (if for instance the device had a GPS unit), etc. If the attacker had access to a communication channel, then a unique packet could trigger it. With closer proximity access, even more possibilities exist: controlling the light to an optical sensor, a specialized memory card could be inserted into the device, a rare pattern could be entered from the keyboard or any of the buttons on the device, etc. There are endless triggering events, but they all serve the same purpose, to notify the Trojan of an event.

6 Implemented attacks

6.1 RS232 end sequence information leakage

6.1.1 Explanation

We created three Trojans using the RS232 module to modify the transmission of data (Fig. 3). Each Trojan requires proximity to the communication channel as the information leaks during communication with other devices. The first Trojan takes advantage of the message structure beginning with the key index used to decide the key needed to decrypt the message. The dip switches found on the physical board determine the key index by adding some entropy to the master key. Following the key index is the ciphertext, the original message encrypted using the AES-128 algorithm along with the private key. Finally, the message terminates with an ending sequence composed of fourteen bytes of 0xFF.

The program used to decrypt the messages on the receiver side, encVerifier, was provided as part of the contest. It uses the key index to know the encrypting key and consequently the decrypting key. This does trivialize the security, but provides a convenient key distribution mechanism. The encVerifier program reads the cipher text using the ending sequence

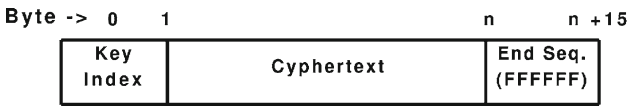

Fig. 3 RS232 Message format

to determine the variable length. The source code for the encVerifier program checks for the presence of five bytes of 0xFF in the last available eight-byte segment. It then loops through the data from the last read byte to the first read byte in the eight-byte sequence and increments a counter for each byte of 0xFF observed. If at least five non-consecutive bytes of 0xFF have been read in the last eight-byte segment, the encVerifier program decrypts the message and displays it to the screen.

Exploiting the message structure, specifically the ending sequence, allows for two attacks. The first attack places information after the transmission of the ending sequence, but still in the original message. This attack is flexible in the information and the amount of information transmitted. We chose to leak the entire key at the end of one message. The encVerifier program does not notice extra information placed on the RS232 stream since it stops reading from the stream after detecting the ending sequence. A malicious program monitoring the stream would be able to read past the ending sequence and receive the extra data at the end of the message.

6.1.2 Results

After enabling the Trojan, the ending sequence is embedded with extra data. Using the original encVerifier program produces the expected output as it ignores the added information. However, the malicious encVerifier program ignores the ciphertext message and recovers the leaked key.

6.2 RS232 end sequence information leakage 2

6.2.1 Explanation

The second exploit to the end sequence takes advantage of the number of ending sequence bytes sent. Since the ciphertext can be of variable length and the encVerifier program checks for the presence of five bytes of 0xFF in the last eight-byte segment, more than five bytes of 0xFF must be sent, but less than the fourteen bytes of 0xFF that are sent by the reference system. With a variable length message, there are eight possible locations within an eight-byte block where the end sequence can start. Figure 4 depicts each of these cases, where the fourteen byte wide table represents the fourteen bytes of 0xFF sent as the ending sequence of a message transmission, the rows represent the eight cases, and the numbers in the row represent the location of that byte in an eight-byte segment. The two shades of gray show adjacent messages and the white cells are the five bytes of 0xFF that the encVerifier program uses to determine the end of a sequence. The fourteen bytes of 0xFF are more than enough to create an ending sequence and other data can be contained in the original message size as seen in Fig. 5. The markings in this figure are the same as Fig. 4 with the addition of dotted white cells

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Case 1	1	2	3	4	5	6	7	8	1	2	3	4	5	6
Case 2	8	1	2	3	4	5	6	7	8	1	2	3	4	5
Case 3	7	8	1	2	3	4	5	6	7	8	1	2	3	4
Case 4	6	7	8	1	2	3	4	5	6	7	8	1	2	3
Case 5	5	6	7	8	1	2	3	4	5	6	7	8	1	2
Case 6	4	5	6	7	8	1	2	3	4	5	6	7	8	1
Case 7	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Case 8	2	3	4	5	6	7	8	1	2	3	4	5	6	7

Fig. 4 Ending sequence cases. *Gray* shading differentiates eight-bytes segments. *White* indicates which bytes cause enc Verifier to stop receiving

Byte	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Case 1	1	2	3	4	5	6	7	8	1	2	3	4	5	6
Case 2	8	1	2	3	4	5	6	7	8	1	2	3	4	5
Case 3	7	8	1	2	3	4	5	6	7	8	1	2	3	4
Case 4	6	7	8	1	2	3	4	5	6	7	8	1	2	3
Case 5	5	6	7	8	1	2	3	4	5	6	7	8	1	2
Case 6	4	5	6	7	8	1	2	3	4	5	6	7	8	1
Case 7	3	4	5	6	7	8	1	2	3	4	5	6	7	8
Case 8	2	3	4	5	6	7	8	1	2	3	4	5	6	7

Fig. 5 Ending sequence cases. Same coloring as Fig. 4 but the *dotted white cells* indicate possible locations for hidden messages

representing the five bytes of data that could be used for adding arbitrary information. In the latter figure, each case gives the encVerifier program sufficient amounts of 0xFF bytes for that program to correctly terminate, but also allows for five bytes of arbitrary data to be embedded in the stream.

Because of the AES-128 encryption scheme, the ciphertext is always in sixteen-byte blocks. Those sixteen bytes plus one byte of key index and the fourteen bytes of 0xFF for the end sequence result in a message of $(16 \cdot n + 14 + 1)$ bytes, where n is the number of cipher blocks. This means that the message transmissions will always fall into case eight from Figs. 4 and 5. Instead of being limited to five bytes by case three, the first nine bytes out of the fourteen bytes of 0xFF can be used to embed information as long as the last five bytes contain 0xFF. This method is more covert than Trojan 6.1 because the five bytes of added data appear to be part of the ciphertext and not an extension to the end of the original message. Both Trojans are invisible to the original encVerifier program and allow that program to operate correctly in every instance.

6.2.2 Results

The encVerifier program correctly decodes the messages since the modified message still adheres to the message format including the correct number of stop bytes. Using a modified encVerifier program that is aware of the encoding of the message can recover the leaked bytes.

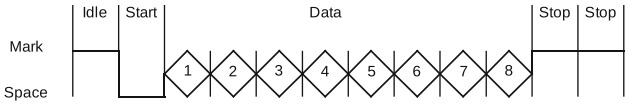

Fig. 6 A valid RS232 message frame. The data bits are shown generically, but would either be a mark or space

6.3 RS232 multiple transmission rates

6.3.1 Explanation

The third attack on the RS232 port takes a different approach and instead of exploiting the message structure, it exploits the protocol itself. The RS232 protocol uses a single data wire for transmission. When the line is idle, it is a mark condition representing a negative voltage, logic '1'. Likewise, logic '0' is a space condition produced by pulling the line up to a positive voltage. The RS232 specification allows for various combinations of baud rates, data bits in each packet, the number of stop bits, parity bits along with various other extensions. The Alpha uses a 9600 baud rate transmission with eight data bits in each packet and a start and stop bit corresponding to Fig. 6. The start bit must go from a mark to a space in order for the receiver to recognize the beginning of the asynchronous transmission. Once the receiver recognizes this condition, it can begin sampling the data bits at the agreed upon baud rate in order to extract all the data from the packet.

The Alpha board and receiver currently receive at 9600 baud, but both are also able to transmit and receive data at faster rates. This Trojan crafts packets that are transmitted at a faster rate, yet when viewed at the slower rate appear to be a valid transmission. In this manner, two overlaid transmissions allow two messages to transmit simultaneously at different baud rates. In order to maintain transmission at two baud rates, both transmissions must have complete frames including a start bit, stop bit(s), and data bits in addition to looking like valid data when reading the same data at two different rates.

115200 baud is another standard transmission rate that is twelve times faster than 9600 baud, which means that twelve bits are transmitted for every bit of data transmitted at the 9600 baud rate. In order to keep from encountering framing errors, a complete packet must fit into these twelve bits. This packet consists of an idle mark bit, a space start bit, eight data bits, and two stop bits for a total of twelve bits. This packet of twelve bits must appear similar enough to the constant '1' bit transmitted at the same time at 9600 baud for the encVerifier program to still identify the 9600 baud rate transmission. There is also some inflexibility in which bits can be shaped to look like the slower transmission since the initial mark bit and start bit must remain fixed along with two stop bits. This means that if the 9600 baud rate transmission is transmitting a mark bit, the 115200 baud rate transmission can

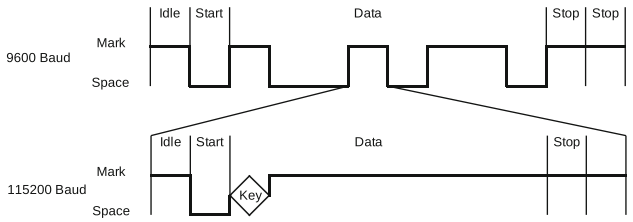

Fig. 7 A valid RS232 frame at 115200 baud can be shaped like a mark bit

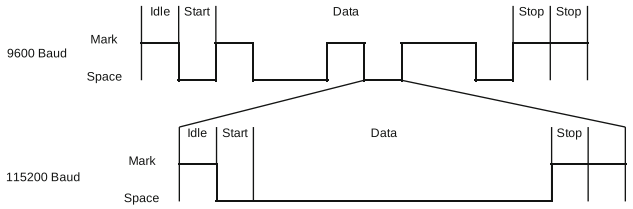

Fig. 8 A valid RS232 frame at 115200 baud can be shaped like a space bit

match it on eleven out of the twelve bits, but if it is a space, then the best that can be managed is nine out of twelve bits. It turns out that both of these are acceptable to appear as the constant 9600 baud rate value. These two scenarios can be seen in Figs. 7 and 8, respectively. In these figures, the upper transmission packet sends at 9600 baud and the lower transmission packet at 115200 baud. Since the lower transmission is twelve times faster than the upper, the bottom transmission frame represents a single bit of the upper transmission frame and has been shaped to look as much like the bit from the upper frame that it is representing.

The transmissions of the mark bit at the slower rate can be used to embed one bit of additional information, more specifically the key. Shaping the faster transmission to mimic the slower transmission more accurately represents the mark bit given that eleven out of the twelve bits are the same. One bit of information embedded in the data section of the faster transmission results in a transmission that matches either ten or eleven out of the twelve bits, both of which are accurate enough be received without error at the slower transmission rate. Our modified Alpha transmitter functions at the 115200 baud rate sending out signals that look like the 9600 baud rate but with one bit of the key embedded in each mark bit of the 9600 baud rate transmission. This allows the encVerifier program to verify that the 9600 baud rate transmission contains the correct information, but a malicious program listening at the 115200 baud rate can extract the key from the signal. Our malicious encVerifier program verifies that the 115200 baud rate transmission does in fact contain the key.

6.3.2 Results

When this Trojan is enabled, it transmits both the expected data and the secret data on the same signal. The original encVerifier program produces the expected output and each of the received bytes is identical to the sent bytes. The malicious encVerifier program listens to the signal at the 115200 baud rate in order to return the leaked key. The 115200 baud rate signal shapes its bits to appear as a '1' or as a '0', so the eight bits of data sent at the faster baud rate should all be either zeros or ones, except for the bit of information embedded in the transmission of the ones. In this way, the baud rate of the leaked information is the same as the original transmission since only one bit of information leaks for every 12 bits transmitted, but the transmission is twelve times faster. In the transmission output, the eight bits of zeros appear as 0x00 and the 8 bits of ones as 0xFF. The 0xFE bytes represent a leaked zero and the 0xFF bytes a leaked one.

6.4 Denial of service

6.4.1 Explanation

The goal of this attack was to create a DOS that occurs during normal operation, but would go unnoticed during device testing. This attack does not require any proximity to the device, although closer proximity could aid in the trigger process. The DOS attack can be implemented in many ways, but at its core, it degrades the performance or validity of the Alpha device. In order to implement a DOS, an acceptable location was chosen such that it would remain hidden during the verification tests. Finding an appropriate target is actually trivial since a modification to almost any signal produces incorrect output. The clock could be frozen to a value so that the entire device quits functioning, the transmission of the data could be corrupted so that it does not send correctly formatted RS232 frames, even the data read from the keyboard could be skewed so that incorrect messages were transmitted.

Our implementation of the DOS Trojan attacks the key used to encrypt messages and accomplishes its goals of denying service and remaining hidden by making only minor modifications to the code. The trigger used for our implementation utilizes a timer so that it intermittently functions, switching approximately every 3.5 minutes. Finally, the user in the field will not notice that the Trojan is active since it only corrupts the ciphertext sent out so that the intended receiver receives a ciphertext encrypted with a different key than they expect. The attack makes use of a counter within the seven segment driver routine and a slight modification of the AES-128 routine. The seven segment driver already contains a 12 bit counter running at 625 KHz. Adding 17 additional bits allows a 7.15 minute cycle time for the upper order bit. Adding a few more bits would substantially increase the cycle time allowing it to pass validation tests. The upper order bit is threaded through the seven segment driver as a fake enable signal which is connected to the AES-128 core as a similar enable. Within the AES-128 module, this bit is XORed

with one of the key bits to corrupt approximately 50% of the ciphertext.

6.4.2 Results

From the user's perspective, the transmission appears normal, but it transmits corrupted data. As a test, a single 'A' character transmitted with a random encryption key produces an output where 62 out of the 128 bits transmitted incorrectly (approaching the expected error rate of 50%).

6.5 Thermal leakage

6.5.1 Explanation

Another interesting route for data leakage is through thermal transmission. In this attack, the FPGA systematically heats up or operates at its normal state to create a binary code used to convey information. A malicious user can place a temperature probe on the FPGA and monitor the temperature of the device to collect the leaked data (e.g. key bits). Even though this attack requires physical access to the FPGA, it has very real applications. These could include a scenario where one of the Alpha devices is captured allowing the captors to extract the key and decrypt all previous and future Alpha transmissions. This attack would also work if a malicious user had temporary physical access to the FPGA and was able to extract the key at that time. It also has added stealth compared to simply routing this information over unused pins as many verification tests do not check for the presence of information conveyed through heat.

The FPGA must be able to generate enough heat to be sensed by a temperature probe, on the order of a couple of degrees Fahrenheit. As with most devices on an FPGA, static and dynamic current leakage account for the power dissipation and therefore heat generation. In our usage, since the configuration is fixed at run-time, the static power remains very similar between the original design and the modified thermal leakage design. Consequently, most of the power variation comes through the dynamic power dissipation. The model for dynamic power dissipation is $Power = CEq \times Vcc^2 \times F$ where CEq is the total capacitive load, Vcc is the supply voltage and F is the switching frequency. In this Trojan, a series of output pins switched at 50 MHz cause extra capacitive loads by driving the output pins. Also, since these are driven quickly, extra power dissipates when compared to the reference design causing the FPGA to heat up. To communicate the key, we represent a '0' as the temperature of normal operation and a '1' as the temperature during heated operation. A large counter allows the shifting of the bits of the key to occur slowly, on the order of a minute, and provides ample time for the FPGA to change temperature. By sampling the temperature of the FPGA at defined intervals, we obtained the key.

6.5.2 Results

We took samples of the temperature at defined intervals by using a thermocouple connected to a multimeter. We then decoded these measurements using the transmission scheme previously described. This allowed us reclaim the key used to encrypt the message.

6.6 AM transmission

6.6.1 Explanation

Modulating a pin on the FPGA generates an RF signal. This signal can be used to transmit the key bits. For the RF attacks, we utilized one of the pins of the socket since this socket is perpendicular to the ground plane of the board which creates a better antenna than the expansion header pins. To allow this attack to be verified without any specialized equipment and to demonstrate the range capabilities, it was performed at two different frequencies. One transmission at 1560 KHz and can be received with an ordinary AM radio. The other attack transmits at 50 MHz and requires a specialized radio, such as a HAM radio, to receive the signal. The AM transmission has an extremely short range, on the order of inches, when compared to the 50 MHz transmission received over 4 feet away. In both cases, touching the pin with a finger or paper clip increases the transmission range by several orders of magnitude. Since the two attacks are effectively the same, we describe the AM attack in detail and only mention the differences with the 50 MHz variant.

The data carried by the AM signal needs to be easily interpreted by a human. We utilized a beep scheme where a single beep followed by a pause represents a '0' and a double beep followed by a pause represents a '1'. Figure 9 shows the hardware necessary to generate the sequence of beeps based on an input value. After the counter recycles, a shift register shifts in the next bit of the master key. The figure also shows the top three bits of the counter used for the eight sequential states. The first state is a beep. The second state is always a pause. The third state will generate a beep only if the data

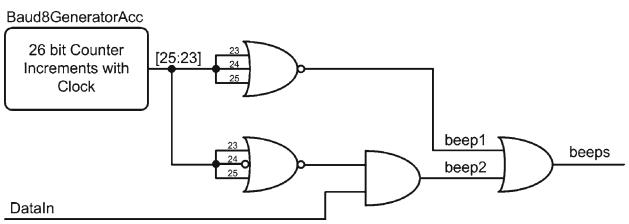

Fig. 9 Hardware to generate the beep pattern

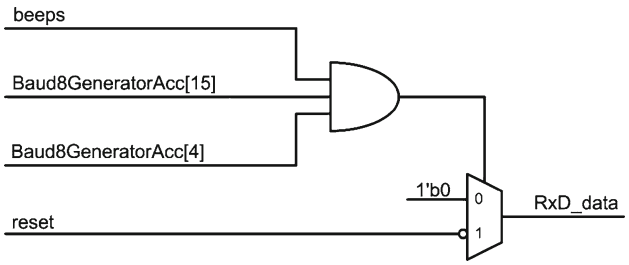

Fig. 10 Hardware to convert beeps to an audible tone

input is a one. The remaining states generate the long pause between beeps. For a person to be able to hear the beeps, the beep signal needs to be converted into an audible tone and then modulated. Figure 10 shows the logic to accomplish this. When the beep line from Fig. 10 is a '1', it is ANDed with bits fifteen and four of the counter. Bit four toggles at a rate of 1560 KHz, the AM carrier, and bit fifteen toggles at a rate of 762 Hz, the audible tone. After this AND gate, a mux enables the transmitter.

The transmission could be further obfuscated by employing a spread spectrum modulation technique such as direct-sequence spread spectrum (DSSS) which transmits the information not just on one frequency, but spread over the device's operating frequency. The result is white noise unless the receiver knows the correct modulation values to interpret the information.

6.6.2 Results

We viewed the signal generated by this Trojan with a standard oscilloscope (Fig. 11). Part a of this figure shows the beeping sequence, where two beeps represent a '1' and a single beep represents a '0'. Part b represents the beginning of the 760 Hz audio tone. Finally, part c shows the RF carrier wave.

6.7 50 MHz transmission

6.7.1 Explanation

The 50 MHz transmission demonstrates the range capabilities tied to increasing the frequency. This increased range has to do with the length of the transmission pin. As this pin length approaches 1/4 wavelength of the carrier, the quality of the radiation pattern increases as does the amount of radiated energy. Transmitting at even higher frequencies such as 300 MHz should increase the transmission distance, but it will be capped as we approach microwave frequencies due to the uncorrected parasitics on the board. Instead of using bit 4 of the baud rate counter, this Trojan uses the 50 MHz board clock to modulate the data.

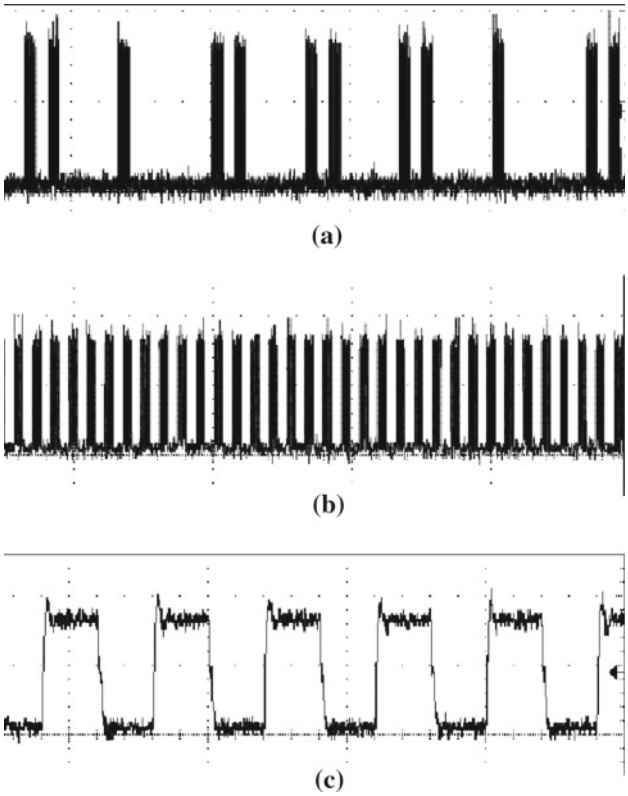

Fig. 11 Measured patterns for RF signal leakage attack. **a** Beeping Pattern; **b** 760 Hz audio tone; **c** RF carrier wave

6.7.2 Results

Enabling this Trojan allowed us to obtain the secret from over 4 feet away by using a HAM radio to audibly pick up the signal. Decoding the series of beeps allowed us to verify the leakage of the correct information. Placing a finger or paper clip on the pin allowed us to pick up the signal on the other side of the building, about 50 feet away. Viewing this transmission with an oscilloscope looked almost identical to Fig. 11 except for the change in frequency.

6.8 LED transmission

6.8.1 Explanation

The same beep pattern as the RF Trojan was used to leak the key with a high-frequency blinking LED. This time, instead of using an audible tone, two different blink rates conveyed the information. The LED blinks at 1 KHz to indicate a beep and at 2 KHz to indicate silence. In order to make it less noticeable to the user, the LED constantly blinks at a high frequency preventing the intensity from changing when switching to transmission mode. The difference between the LED blinking at 1 KHz and 2 KHz is not noticeable to the human eye simply by observing the LED. A separate circuit with a

photodiode and band pass filter displays the blinking pattern on its own LED that blinks exactly as the beeps are heard by a person using the RF Trojans.

6.8.2 Results

This Trojan requires a specialized circuit to view the transmission by shifting the frequencies in the LED. This can be seen by placing the circuit near the LED that is transmitting and reading the values by observing the LED of the external circuit. The external circuit placed a few inches from the Alpha device reveals the key used for message encryption.

7 Considered attacks

The following Trojans are a list of considered, but not implemented Trojans. For some of these Trojans, steps were made to implement them, but either their prohibitively high difficulty, infeasibility, potentially destructive nature or our lack of time and resources prevented us from implementing them. The following short descriptions provide a high-level conceptual approach to these attacks.

7.1 Keyboard LED

The Keyboard LED Trojan is similar to the LED transmission but follows a different protocol. Since the transmission of information between the keyboard and the host device is relatively slow, it is not possible to blink the keyboard LED at a rate fast enough to be invisible to the eye as Trojan 6.8 does. Instead, a different scheme must be used where one of the LEDs remains lit for the majority of the time and then only momentarily turned off and back on again. This could be used to transmit data where the light, sampled at defined intervals, transmits binary data. Preliminary tests showed that it was possible to flicker a keyboard LED such that it was almost impossible to see when staring directly at it and would be less noticeable when not paying specific attention to the LED.

7.2 Blinking cursor

Similar to the above LED Trojans, the cursor on the VGA screen blinks at a predefined rate. Altering the rate slightly produces a code for leaking information. By varying the rate at which the cursor blinks, the key could be extracted while remaining hidden from the user. This attack would require access to the device, by capturing either the Alpha or a malicious user in close proximity to the device. It is a bit easier to recover the key with this Trojan as it does not require physical access to the FPGA (to extract the temperature information, for example) since the VGA monitor is an external device that is meant to be visible to the user.

7.3 VGA sync

The horizontal and vertical syncs generated by the BASYS VGA controller create a small gap. The gap can be used to hide data in the VGA signal such that it does not affect the output as displayed on the monitor. A specialized decoder circuit that had access to the VGA signal would need to extract the information from the small gap to recreate the key.

7.4 Change bitstream on PROM

The Cypress chip on the BASYS board implements the Universal Serial Bus (USB) protocol and incorporates a fully programmable microcontroller. The Cypress chip has connections to the Erasable Programmable Read Only Memory (EPROM) used to store the FPGA configuration since the user can program the FPGA through the USB port. A Trojan implemented on the Cypress chip using the embedded microcontroller could erase the EPROM memory after some time delay causing a permanent DOS attack that can only be fixed by reprogramming the entire board. To implement a Trojan on the Cypress chip, firmware must be downloaded through USB and saved in the EPROM for the Cypress chip. A software virus could install the Trojan to the BASYS board connected by USB to an infected computer.

7.5 Destruction

Another idea for a DOS attack was to physically destroy the device when triggered. Attempts were made to get the device to self-destruct, but none were successful. The means to this attack varied significantly and included generating enough heat to be damaging, creating HDL code or tweaking the Xilinx ISE synthesis settings such that it would generate a bitstream that was self-destructive. Some progress was made in generating heat and was successfully implemented as a key leakage attack, but not enough heat was able to be generated that would damage the FPGA. Other attempts were made to write self-damaging HDL code, but no feasible structure could be created that violated safe FPGA configurations.

7.6 Store one bit

Each DOS attack, save the destruction of the FPGA, is temporary in that a reset of the bitstream would disable the Trojan. The Trojan would have to be retriggered in order to begin functioning, which might be very challenging if it required an elaborate triggering mechanism to reinstate the Trojan. But, if one bit of persistent data could be stored in the Alpha, then that bit could be set upon successful triggering of the Trojan and then used to reinstate the Trojan upon each reset. Significant effort was put forth to find a way to store one bit of data, but no way was successfully found. Attempts were

Table 1 The variance in power for each Trojan compared to the reference design for each of the states of the system

	Ref mA	T1 Δ	T3 Δ	T4 Δ	T5 Δ	T6 Δ	T7 Δ	T8 Δ
Reset	146.4	0.4	0.6	0.6	65.4	0.7	0.7	0.8
Init min	156.0	0.3	0.5	0.5	22.8	1.6	0.6	1.0
Init max	185.0	0.4	0.5	0.6	22.5	0.7	0.7	0.7
Encrypt	144.7	0.0	0.0	0.0	00.0	0.0	0.1	0.5
Transmit	153.1	0.4	0.6	0.6	22.2	1.0	0.8	1.2

made to communicate with the PROM from the FPGA and with the Cypress chip used for the USB that had its own PROM. It was discovered that it was not possible to write directly to the Cypress PROM, but instead, a USB device plugged into the USB port of the board was required in order to change that PROM.

8 Judging

We presented our hardware Trojans at CSAW 2008 to a panel of industry and academic judges along with the other contest entrants. The judges evaluated the Trojans on their use of power, variance in bitstream size, stealth, and novelty. Table 1 shows each Trojan for the given states of the system and their variance in power usage varies from the reference design. The measurements were made using a bench multimeter in series with the power supply. Within a 20- min window, these values varied by 0.5 mA from the same measurement a few minutes earlier. It should be noted that Trojan T5 (Sect. 6.5) requires large amounts of power by the nature of its design. In order to heat up the FPGA, heat needs to be generated by consuming more power. The bitstreams for each of the designs augmented with the Trojans matched the size of the original reference design and each Trojan was well hidden in both the source code and by inspection of the operating device. Using these metrics, our team won first place.

9 Conclusion

Security vulnerabilities in the IC supply chain continue to increase as both consumers and the government increase their dependence on foreign ICs. The 2008 CSAW competition at the Polytechnic Institute of NYU explored these vulnerabilities by creating a competition where hardware hacking proofs of concept showcase supply chain weaknesses. This paper outlines our experiences in developing hardware Trojans, detailing each of the attacks that helped us win first place at the competition.

Acknowledgments The authors would like to thank Prof. Nasir Memon at the Polytechnic Institute of NYU for creating the Computer Security Awareness Week, and specifically the Embedded Systems Challenge. We would also like to thank the anonymous referees and the editor of this paper (Úlfar Erlingsson) for providing useful feedback and suggestions for improvement.

References

1. Cyber security awareness week 2008 (2008)
2. Abdel-Hamid, A., Tahar, S.: Fragile IP watermarking techniques. In: Proceedings of the Conference on Adaptive Hardware and Systems (AHS), pp. 513–519 (2008)
3. Abdel-Hamid, A., Tahar, S., Aboulhamid, E.M.: A public-key watermarking technique for IP designs. In: Proceedings of the Conference on Design, Automation and Test in Europe (DATE), pp. 330–335 (2005)
4. Abdel-Hamid, A., Tahar, S., Aboulhamid, E.M.: Finite state machine IP watermarking: A tutorial. In: Proceedings of the Conference on Adaptive Hardware and Systems (AHS), pp. 457–464 (2006)
5. Adee, S.: The hunt for the kill switch. IEEE Spectrum **45** (May, 2008)
6. Alkabani, Y., Koushanfar, F.: Active hardware metering for intellectual property protection and security. In: Proceedings of USENIX Security Symposium, pp. 1–16 (2007)
7. Alkabani, Y., Koushanfar, F., Potkonjak, M.: Remote activation of ICs for piracy prevention and digital right management. In: Proceedings of International Conference on Computer Aided Design (ICCAD), pp. 674–677 (2007)
8. Caldwell, A., Choi, H.-J., Kahng, A., Mantik, S., Potkonjak, M., Qu, G., Wong, J.: Effective iterative techniques for fingerprinting design IP. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 208–215 (2004)
9. DARPA. TRUST in integrated circuits (TIC) (2007)
10. Defense Science Board. Task force on high performance microchip supply. 200502HPMSReportFinal.pdf (2005)
11. Di, J.: Trustable recognition of undesired threats in hardware (TRUTH) analysis tool, for analysis of pre-synthesis behavioral and structural VHDL designs. http://comp.uark.edu/~jdi/truth.html, 2009. Accessed on 06/2009
12. Digilent. Basys system board (2008)
13. Hwan, D., Tiri, K., Hodjat, A., Lai, B.-C., Yang, S., Schaumont, P., Verbauwhede, I.: AES-based security coprocessor IC in 0.18-μm CMOS with resistance to differential power analysis side-channel attacks. In: IEEE Transactions on Solid-State Circuits, pp. 781–792 (2006)
14. Jin, Y., Kupp, N., Makris, Y.: Experiences in hardware Trojan design and implementation. In: Proceedings of the International Workshop on Hardware-Oriented Security and Trust (HOST), pp. 50–57 (2009)
15. Kahng, A., Lach, J., Mangione-Smith, W., Mantik, S., Markov, I., Potkonjak, M., Tucker, P., Wang, H., Wolfe, G.: Constraint-based watermarking techniques for design IP protection. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 1236–1252 (2001)
16. Lach, J., Mangione-Smith, W., Potkonjak, M.: Fingerprinting digital circuits on programmable hardware. In: Proceedings of the International Workshop on Information Hiding (IH), pp. 16–31 (1998)
17. Lach, J., Mangione-Smith, W., Potkonjak, M.: FPGA fingerprinting techniques for protecting intellectual property. In: Proceedings of the Custom Integrated Circuits Conference (CICC), pp. 299–302 (1998)
18. Lee, J., Lim, D., Gassend, B., Suh, G.E., van Dijk, M., Devadas, S.: A technique to build a secret key in integrated circuits for

identification and authentication applications. In: Proceedings of VLSI Circuits, pp. 176–179 (2004)
19. Lofstrom, K., Daasch, W.R., Taylor, D.: IC identification circuit using device mismatch. In: Proceedings of International Solid-State Circuits Conference (ISSCC), pp. 372–373 (2000)
20. Maeda, S., Kuriyama, H., Ipposhi, T., Maegawa, S., Inoue, Y., Inuishi, M., Kotani, N., Nishimura, T.: An artificial fingerprint device (AFD): a study of identification number applications utilizing characteristics variation of polycrystalline silicon TFTs. In: Electron Devices, IEEE Transactions on, pp. 1451–1458 (2003)
21. NSA. Trusted access program office (2009)
22. Oliveira, A.: Robust techniques for watermarking sequential circuit designs. In: Proceedings of the Design Automation Conference (DAC), pp. 837–842 (1999)
23. Oliveira, A.: Techniques for the creation of digital watermarks in sequential circuit designs. In: IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 1101–1117 (2001)
24. Qu, G., Potkonjak, M.: Intellectual property protection in VLSI designs: theory and practice. Kluwer Academic Publishers, Boston, MA (2003)
25. Roy, J.A., Koushanfar, F., Markov, I.L.: EPIC: Ending piracy of integrated circuits. In: Proceedings of Design, Automation, and Test in Europe (DATE), pp. 1069–1074 (2008)
26. Su, Y., Holleman, J., Otis, B.: A 1.6j/bit stable chip ID generating circuit using process variations. In: Proceedings of International Solid-State Circuits Conference (ISSCC), pp. 406–407 (2007)
27. Suh, G.E., Devadas, S.: Physical unclonable functions for device authentication and secret key generation. In: Proceedings of Design Automation Conference (DAC), pp. 9–14 (2007)
28. Suh, G.E., O'Donnell, C.W., Sachdev, I., Devadas, S.: Design and implementation of the AEGIS single-chip secure processor using physical random functions. In: Proceedings of International Symposium on Computer Architecture (ISCA), pp. 25–36 (2005)
29. Torunoglu, I., Charbon, E.: Watermarking-based copyright protection of sequential functions. In: IEEE Transactions on Solid-State Circuits, pp. 434–440 (2000)