Design, Implementation and Security Analysis of Hardware Trojan Threats in FPGA

Devu Manikantan Shila and Vivek Venugopal
United Technologies Research Center
411 Silver Lane, E Hartford, CT USA
Email:{manikad, venugov}@utrc.utc.com

Abstract—**Hardware Trojan Threats (HTTs) are stealthy components embedded inside integrated circuits (ICs) with an intention to attack and cripple the IC similar to viruses infecting the human body. Previous efforts have focused essentially on systems being compromised using HTTs and the effectiveness of physical parameters including power consumption, timing variation and utilization for detecting HTTs. We propose a novel metric for hardware Trojan detection coined as HTT *detectability metric* (HDM) that uses a weighted combination of normalized physical parameters. HTTs are identified by comparing the HDM with an optimal detection threshold; if the monitored HDM exceeds the estimated optimal detection threshold, the IC will be tagged as malicious. As opposed to existing efforts, this work investigates a system model from a designer perspective in increasing the security of the device and an adversary model from an attacker perspective exposing and exploiting the vulnerabilities in the device. Using existing Trojan implementations and Trojan taxonomy as a baseline, seven HTTs were designed and implemented on a FPGA testbed; these Trojans perform a variety of threats ranging from sensitive information leak, denial of service to beat the Root of Trust (RoT). Security analysis on the implemented Trojans showed that existing detection techniques based on physical characteristics such as power consumption, timing variation or utilization alone does not necessarily capture the existence of HTTs and only a maximum of 57% of designed HTTs were detected. On the other hand, 86% of the implemented Trojans were detected with HDM. We further carry out analytical studies to determine the optimal detection threshold that minimizes the summation of false alarm and missed detection probabilities.**

Index Terms—**Design; Resiliency; Security; Hardware Trojans**

I. Introduction

On an average, the human body is attacked daily by millions of viruses spoofing as antigens. However, not all viruses can infect the human body due to the immunity present inside the human body. At the same time, the human body is susceptible to an infection if the virus is successful in crippling the immune system. Learning and understanding from biologically inspired systems, Hardware Trojan Threats (HTTs) are *virus-like stealthy malicious components* that can infect the Integrated Circuit (IC). With the increasing practice of outsourcing design and manufacturing steps, various stages of an IC lifecycle are vulnerable to attacks; examples include but not limited to integration of unauthenticated intellectual property (IP) blocks by third party vendors using untrusted tools at the design stage, unmonitored fabrication and assembly of tampered components due to outsourcing at external foundries at the silicon prototyping and integration stage, reverse engineering and tampering of FPGA configuration stream with malicious logic at the production and shipping stage [1].

Typically HTTs are designed to lie quiescent and imperceptible until it is activated via an internal or external trigger. Current formal verification or hardware fault detection algorithms are not capable of detecting such rarely triggered HTTs. Similar to viruses infecting human body, successful perpetration and activation of HTTs could lead to chaos in civilian infrastructure (aerospace, transportation or energy domain), sabotage critical military applications and missions, disable missile and weapon systems, leak sensitive information or provide backdoor access to highly secure systems [2]. In the past few years, several detection algorithms in the arena of hardware security have been proposed [3]. Destructive techniques such as de-packaging, reverse engineering and imaging of ICs are very expensive and can be applied to only a selected quantity of ICs. Non-destructive techniques such as side-channel analysis (aka post-silicon analysis) detect malicious intrusions by vetting the physical characteristics of IC (power consumption, timing variation, temperature, layout structures) with a trusted reference model. These models are scalable as opposed to destructive techniques. Proof-carrying codes [4] are also proposed where the vendor will construct a formal proof of the design adhering to certain security properties, which will be later verified by customer to ensure that design is free from modifications. Besides the overhead involved in creating huge proofs for even smaller codes, loop holes can also occur while defining many security properties. Some efforts propose to use additional circuitry to monitor chip functionality (e.g., guard modules such as DEFENSE, RETC) and physical parameters (scan chains), commonly referred to as invasive techniques, see [5].

A. Motivation and Contribution

The wide-spread focus on post-silicon techniques to test the trustworthiness of IC's lead us to the following question: *how reliable are existing post-silicon testing techniques?* Using existing Trojan implementations and Trojan taxonomy as a baseline, this paper addresses the question by designing seven hardware Trojans at the design level in a FPGA testbed. HTTs were designed to perform a variety of attacks ranging from sensitive information leak to denial of service attack. Security analysis on the implemented Trojans clearly showed that existing post-silicon techniques based on physical characteristics such as power consumption, timing variation or utilization does not necessarily expose the existence of HTTs as HTTs can be optimally designed and placed into the hardware that masks within these parameters. For instance, using power consumption, timing variation and utilization, only 43%, 57% and 43% of the Trojans were detected, respectively. An in-depth investigation of the existing efforts show that less attention has been devoted to the monitoring of the system to analyze the HTT infection using a combination of affected physical parameters. We observe that a single parameter lacks the ability to capture HTTs with distinct features, and hence a combination of parameters capturing different characteristics of the Trojan need to be designed. *Motivated by this observation, we propose a novel metric for hardware Trojan detection, termed as HTT detectability metric (HDM) that leverages a weighted combination of normalized physical parameters.* HTTs are identified by comparing the HDM with an optimal detection threshold; if the monitored HDM exceeds the estimated optimal detection threshold, the FPGA will be tagged as malicious. *We also carry out analytical studies to derive the optimal detection threshold that minimizes the summation of false alarm and missed detection probabilities.* Our security analysis results in fact showed that using HDM and optimal detection threshold of 3.1, 86% of the implemented Trojans were detected as opposed to using power consumption, timing variation and utilization alone. Existing works also suggest the use of hardware encryption and authentication (aka Root of Trust) as a possible methodology to defeat activation of hardware Trojans. *To the best of our knowledge, we first document the experiences of designing and implementing three hardware Trojans at the design level in FPGA Root of Trust (RoT) testbed to defeat the classic trusted hardware model assumptions.*

Roadmap: The rest of the paper is organized as follows: In Sections II and III, the adversary models, system models and the testbed designs are discussed; Section IV explains the design and implementation of the HTTs; the proposed detectability metric is described along with the results in Section V and finally Section VII concludes this work.

II. Adversary Model

It is important to understand the attacker model and the system vulnerabilities when attempting to secure or attack a system. The attacker model leads to potential attack surfaces that can be exploited to successfully leverage an attack. HTTs are then designed and developed based on the exposed system vulnerabilities, however it is to be noted that the developed HTT is functional only if it can evade the detection mechanisms built into the system. In this work, we assume that the attack can be leveraged during the design phase of the IC lifecycle. This is the initial entry point of an adversary and the above assumption is reasonable due to the hefty interaction of third parties at the design phase, in particular with IP blocks, models and Electronic Design Automation (EDA) tools. With this assumption, all attacks presented in this paper are realized at the Register Transfer Layer (RTL) level.

Once the initial entry point of adversary is modeled, the next step is to understand and categorize the attacks based on the level of access needed to activate the HTT and compromise the system. As in a classic cyber security adversary model, these levels range from physical access to remote access to the system. Physical access implies the attacker can physically interact with and monitor the system; e.g. access to the LED, LCD. Local access implies the attacker is in close range of the system; e.g., wireless access to the system. Remote access refers to an attacker being able to launch an attack from a remote computer external to the local network; e.g., Internet access to the system. Based on the levels of access available, the attacker can determine the trigger and the leakage points to target the system and conduct a task-specific attack. In Table I, we outline various components of Xilinx Spartan-3AN FPGA development board and breakdown each component in terms of trigger, leakage and access points.

Table I: Trojan Trigger, Leakage and Access Points

Component	Usage	Type of Access
FX2 Expansion Port	Leakage/ Trigger	Physical
Expansion Headers	Leakage / Trigger	Physical
ADC, DAC	Leakage	Physical
Audio Jack	Leakage	Physical/Local
USB	Leakage/Trigger	Physical
LEDs	Leakage	Physical/Local
LCD	Leakage	Physical/Local
PS/2	Leakage/Trigger	Physicall
VGA	Leakage	Physical/Local
RS-232	Leakage/Trigger	Physical/Local
Ethernet (RJ-45)	Leakage/Trigger	Remote
External Clock	Trigger	Remote

III. System Model

To demonstrate the concept of lightweight, but highly functional HTT design, two different testbeds were developed. The testbeds were designed to mimic the typical critical systems which an attacker may come across when initiating a hardware based attack. Both testbeds were optimized to begin with as an IP block for providing generic solutions, however were not entirely optimized for the exact solution they were being used for, as this is typically the case in industrial designs. As a result, an attacker was given very limited room to leverage an attack and still be unnoticeable in terms of device utilization.

A. Cryptosystem testbed design

The first testbed consists of a cryptosystem using the block cipher based on Feistel Networks, known as the Tiny Encryption Algorithm (TEA) [6]. The HTTs implemented on this cryptosystem testbed [7], [8] as shown in Figure 1, were designed to attack the normal system design rather than the TEA algorithm.

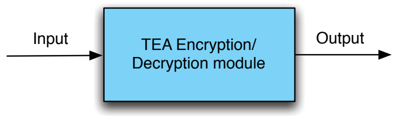

Figure 1: Cryptosystem testbed

B. *Root of Trust testbed design*

The second testbed shown in Figure 2 is a classic Root of Trust (RoT) design that consists of a secure memory and a key guard [9]. In the testbed, the authorized module is allowed to access the contents of memory only via a guard module. The operation of RoT testbed is as follows: (a) The authorized module initally sends an address of the memory location that it needs access to guard; (b) The guard module issues a challenge to the authorized module; (c) The authorized module computes and sends the correct response to the challenge; (d) The guard module verifies it with its response generator and retrives the requested content from memory and sent to the authorized module in encrypted form; and (e) The authorized module decrypts the data and gains access to the contents in the memory. The strength of the RoT testbed in turn lies in the secrecy of the cryptographic keys used in challenge-response and encryption operations. A simple encryption technique consisting of an XOR operation with an arbitrary chosen key was adopted in this testbed. More complicated encryption techniques are avoided, as the targeted HTT does not engage in a cryptanalysis based attack. The challenge-response pair can be implemented using a Finite State Machine (FSM) approach or a predefined mapping of challenges to responses.

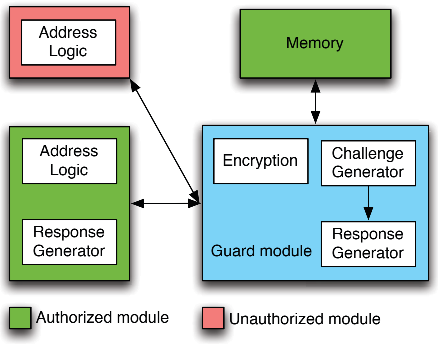

Figure 2: Guard system testbed

IV. Hardware Trojan Threats

When considering how to design Trojans on each testbed, careful consideration was given to the target Spartan-3AN FPGA platform by breaking down all the possible input and output points, see Table I. These led to an understanding of the potential leakage points, access points and external triggers an attacker could utilize to design the Trojan. For instance, with regards to the leakage points, Trojans can be designed to leverage different peripheral components based on the access the attacker may have to the device. Based on these attack surfaces, and a comprehensive Trojan taxonomy, several Trojans were designed and implemented in order to compromise both testbeds, ranging from internally activated Trojans to externally activated Trojans. Within these types of Trojans, some compromised the device by leaking critical information, while others compromised by performing a denial of service attack. As mentioned before, all of the Trojans were implemented at the Register Transfer Layer (RTL). To do so, modules were simply added or changed allowing quick implementation and flexibility, granting the ability to integrate Trojans together, giving a much more powerful attack. With optimization, prior to and after the Trojan being inserted, the footprint on the device was reduced, which helped the power, timing and utilization profiles match more closely that of the trusted system. As opposed to hardware Trojan attacks, such as the Illinois Malicious Processor (IMP) project or the Embedded Systems Challenge during Cyber Security Awareness Week (CSAW) held at Polytechnic Institute of New York, instead of compromising the system alone, constant analysis was applied to determine the effectiveness of each Trojan [10], [11], [12].

A. *Always On Trigger*

An *Always On* HTT constantly leaks sensitive information through any of the leakage points available on a system. Since it does not require activation, the attacker needs to be aware of the instance the leakage occurs. The leakage point determines the attacker model, as the attacker needs to recover the data. For instance, if the attacker has physical access to the system, the data can be leaked through the LEDs by toggling it at an extremely fast rate, such that the human eye cannot detect the change. The attacker can extract the leaked information using photo-transistors connected to an Arduino or Raspberry Pi board. Moreover, since the Trojan is always on, it must provide a pre-defined synchronization routine, which is aware to the attacker to indicate the start and end of leakage data. This synchronization routine may include e.g. repeated toggling of LEDs in a specific sequence, transmission of some error messages through the UART module, etc.

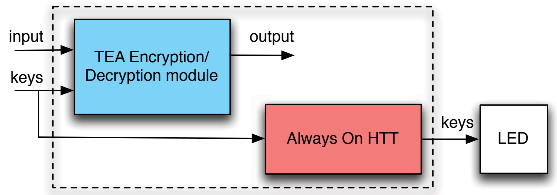

Figure 3: Always On Trigger

The *Always On* HTT, shown in Figure 3 is implemented using a counter and multiplexer, where the counter tracks the bit position of the key being leaked and the multiplexer is used to output the key 8 bits at a time. The multiplexer output is used to drive status LEDs, thereby leaking the key. The synchronization routine indicating the beginning and ending of transmission for the attacker is enabled by sending a specific value (e.g. xBE) over the LEDs four times.

B. Internal Trigger

An internally activated trigger is a subset of an *Always On* HTT, except that its chances of detection is very high if designed incorrectly. The various forms of the internal trigger mechanism may include any of the following: (i) the input or output data can be parsed for a specific sequence to trigger the HTT, (ii) the HTT can be activated after a certain period, (iii) the HTT can be activated once the temperature of the system has increased above a pre-defined threshold. The trigger must be rare enough to ensure that the HTT is not accidentally triggered during system testing.

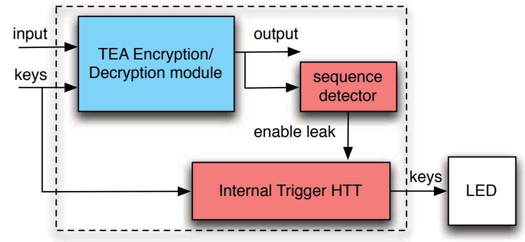

Figure 4: Internal Trigger

The trade-off during the implementation of this HTT depends on the probability of its detection due to additional resource utilization and leakage power consumption or accidental triggering during test-time. This trade-off helps in choosing the counter width when implementing the period based trigger or restricting the combinations for the input/output data sequence detector. Figure 4 shows an internal trigger-based HTT, where the sequence detector checked for the least significant two bits of the output vector. When the specific sequence is satisfied, the sequence detector sends an enable signal to the HTT, thus leaking the key through the LEDs.

C. Legitimate User Trigger

The legitimate user trigger consists of a module placed at the input of the cryptosystem testbed, which accepts user inputs and transforms it to malformed inputs. This is a classic Man-in-the-Middle (MITM) style of attack where a HTT will reside between the encryption module and the user input.

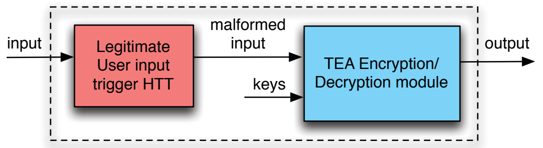
input
Legitimate User input trigger HTT
malformed input
keys
TEA Encryption/ Decryption module
output

Figure 5: Legitimate User Trigger

The input trigger module shown in Figure 5 occupies very few resources as it only complements the bits in certain positions of the input data. Hence, the detection of this module is difficult in terms of latency incurred in sending the input data to the encryption module.

D. Denial of Service

Denial of service is a significant method of not only compromising but also disabling the target system. The denial of service HTT relies on an event occurring multiple times and how this repetitive event can be used to trigger an attack. A counter is used to keep track of the event occurrences. Once the counter overflows, the HTT can either shortcircuit the V_{DD} and GND pins or deactivate the clock using clock gating mechanism.

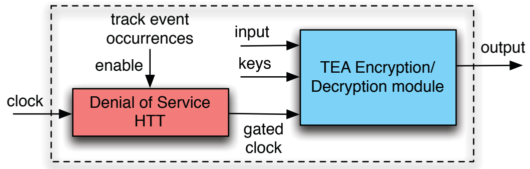

Figure 6: Denial of Service

The denial of service HTT is implemented as shown in Figure 6, where specific event occurrences are tracked to enable the clock gating. Once the clock is disabled, the system is rendered incapable.

E. Beat the Root of Trust

The memory guard system shown in Figure 2 is targeted for demonstrating HTTs developed to beat the Root of Trust (RoT) system. The testbed was slightly modified as shown in Figure 7 to include the Man-In-The-Middle module to demonstrate the vulnerabilities in the testbed. Three different attacks are implemented in the RoT testbed using the Man-In-The-Middle HTT: (i) denial of service, (ii) message spoofing and (iii) leakage of sensitive data.

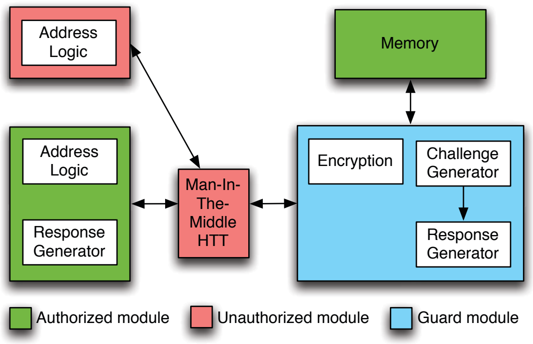

Figure 7: Man-In-The-Middle HTT for beating the authentication in the system

The first HTT in the MITM module modified the challenge prior to it reaching authorized hardware. This makes it impossible for authorized hardware to authenticate with the guard module. The challenge from the guard module is modified using combinational logic, which incurs low latency and remains undetected. The modified challenge makes the authorized module appear as unauthorized resulting in denial of service. The next HTT leverages the presence of both authorized and unauthorized modules on the same testbed. Since the unauthorized modules may fail in their response to the challenge pair given by the guard module, the HTT utilizes the authorized module to exploit the testbed. The HTT accepts the memory address from the unauthorized module and transmits it to the guard module. The guard module issues the challenge which is given by the HTT to the authorized module. The authorized module transmits the correct response which is passed to the guard module by the HTT. The data from the memory address is retrieved and sent to the unauthorized module showing the vulnerability of the system to the HTT. The HTT can also leak the data through any of the leakage points accessible from the module. To implement HTT leaking sensitive information, we placed the Trojan inside the guard module. All the three attacks shows the vulnerability of Root of Trust modules to hardware Trojans residing within the system.

V. Multi-parameter Detection Scheme (HDM)

HTTs can exist virtually anywhere in the system, perform different attacks to cripple the system and may also lie dormant for most of the system's lifetime. From the graphs in Section VI, we observed that depending on the nature of the HTT, some Trojans can be detected with power consumption that to additional output, while others can be detected with timing or utilization that uses of additional logic. As the type of Trojan implemented in turn depends on the adversary model, detection using power consumption, timing or utilization may not succeed always. We therefore propose to construct a metric that uses a combination of different physical parameters such as power consumption, timing variation, utilization and leakage current to detect various sorts of HTTs, irrespective of the adversary model. Our HTT detectability metric (HDM) uses a weighted combination of normalized physical parameters and is given below. The parameters are *weighted* to enforce trust in the HTT detection rate. For instance, in a highly sensitive cryptographic equipment, assigning higher weights for power consumption will lead to enhanced detection rate of HTTs leaking sensitive information. Let m_p represent the number of physical parameters, W_i represent the weight assigned to each parameter and O_i, A_i represent the observed and actual parameter values, respectively. Then the HDM is computed as follows:

$$\text{HDM} = \sum_{i=1}^{m_p} W_i \frac{O_i}{A_i} \tag{1}$$

Given the detection threshold $\tau_p = \sum_{i=1}^{m_p} W_i \tau_i$, HTT will be detected if HDM $> \tau_p$. The detection threshold should be therefore optimally designed to reduce the probability of false alarm and missed detection rates. The optimal threshold is designed as follows: we assume that all parameters are equally significant and hence $W_i = 1$. As τ_i's are mutually exclusive, we independently compute the probability of false alarm and missed detection for each τ_i.

$$O_i = A_i + \eta;\ [H_0] \tag{2}$$
$$= T_i + \eta;\ [H_1] \tag{3}$$

where H_0 and H_1 are the hypotheses that indicate the absence and presence of HTTs. $T_i = \alpha A_i$ represents the parameter values in the presence of HTT and we assume that $\alpha \sim U(l^+, u^+)$, where l^+ and u^+ denotes the lower and upper bound, respectively. $\eta \sim N(0, \sigma^2)$ represents the noise due to process variations. Let $P_{FA}(i)$ and $P_{MD}(i)$ represent the probability of false alarm and missed detection for each parameter i. A false alarm occurs when detection metric gives an alarm but no HTT exists in fact.

$$P_{FA(i)} = P(\frac{A_i + \eta}{A_i} > \tau_i) = P(\eta > (\tau_i - 1)A_i) \tag{4}$$
$$= \frac{1}{\sigma\sqrt{2\pi}} \int_{(\tau_i - 1)A_i}^{\infty} e^{\frac{-x^2}{2\sigma^2}} dx \tag{5}$$

A false clear or missed detection occurs when the detection metric does not give an alarm but a HTT exists.

$$P_{MD(i)} = P(\frac{\alpha A_i + \eta}{A_i} < \tau_i) = P(\alpha < \tau_i - \frac{\eta}{A_i}) \tag{6}$$
$$= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{+\infty} \int_{0}^{\tau_i - \frac{\eta}{A_i}} \frac{e^{\frac{-y^2}{2\sigma^2}}}{u^+ - l^+} dx dy \tag{7}$$

The optimal thresholds, τ_i^*, can be computed by minimizing the summation of the false alarm and missed detection probabilities according to

$$\tau_i^* = \frac{d}{d\tau_i}(P_{FA(i)} + P_{MD(i)})|_{\tau_i = \tau_i^*} = 0 \tag{8}$$

VI. Results

The testbeds shown in Figures 1 and 2 were implemented on the MicroZed Xilinx Zynq-based FPGA development board. Baseline estimates for power consumption, timing variation and resource utilization were obtained after implementing the testbeds without any HTTs. For the power measurements, Xilinx XPower tool was used to measure the total output power and the power utilization of the cryptographic blocks in the FPGA testbed. Xilinx's Chipscope tool and the timing analysis report were used to determine the different parameters such as net skew and maximum propagation delay as part of the timing measurements. The resource utilization was determined from the number of occupied slices and the number of four input Look-Up-Tables (LUTs) provided by Xilinx's Place and Route (PAR) report. Variations in gate length, effective channel widths and the resultant effects on the threshold can affect power consumption and timing variations of the individual slices on FPGAs. Therefore, to account for such variations, a tolerance around the trusted model was predicted [13]. With these tolerances, we set $\eta \sim N(0, 1)$ in our analysis.

Figure 8 shows that with the power consumption, timing variation and resource utilization metrics, 57%, 43% and 57% of the implemented Trojans remained stealthy. This in turn shows the limitations of existing post-silicon testing tech-

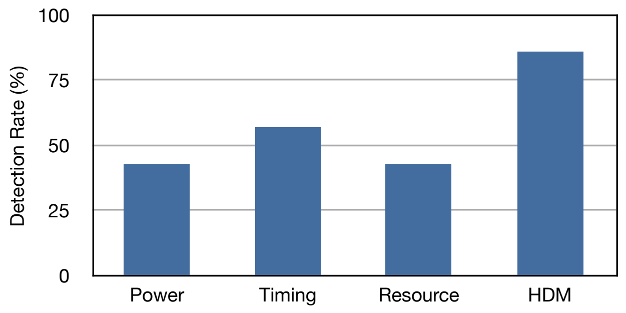

Figure 8: Detection rate for power, timing, resource utilization and HDM

niques. Figure 8 also shows the detectability of all HTTs using HDM, where 86% of the Trojans were detected. Interestingly, we observed that the legitimate trigger remained undetected as it neither contributed to additional output nor logic. In our analysis, using $\alpha \sim U(0.7927, 1.6695)$ and $\eta \sim N(0, 1)$, we observed that the optimal detection threshold (τ_p) is 3.1 and it has the lowest sum of false alarm and missed detection probabilities, see Figure 10. Using $\tau_p = 3.1$, we were able to detect 86% of the implemented HTTs. In Figure 9, we also observe that tuning $\tau_p \leq 3$, can increase the detection rate to 100%, however, in figure 10, we note that the sum of false alarm and missed detection probabilities are indeed high for $\tau_p \leq 3$. Hence, our model sets $\tau_p = 3.1$, see equation (8).

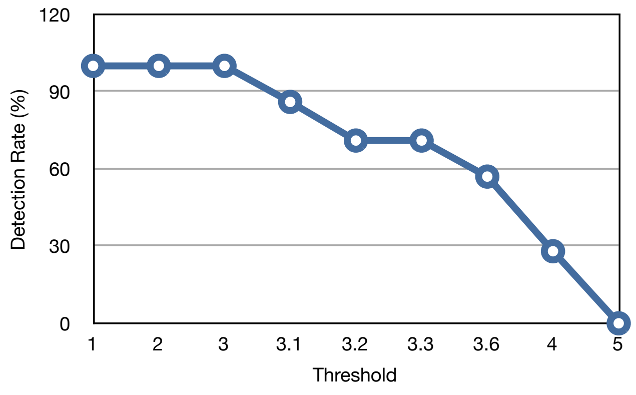

Figure 9: Detection rate for different detection thresholds

VII. Conclusions

In this work, we design, implement and analyze hardware Trojan threats (HTTs) of seven types to understand the limitations of existing post-silicon techniques. Our results manifest that using power, timing or utilization, only a maximum of 57% of designed HTTs were detected. Motivated by this observation, we proposed a novel metric called HTT detectability metric (HDM) that uses a weighted combination of various physical parameters. The detection rate of HTTs increased to 86% with HDM. We also carry out analytical

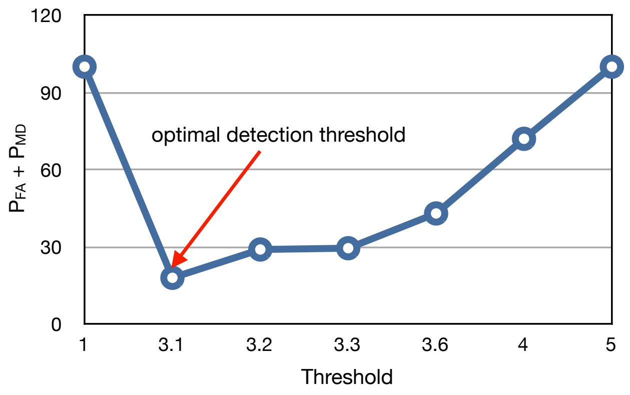

Figure 10: Sum of False Alarm and Missed Detection Vs Detection Threshold

studies to determine the optimal HTT detection threshold that minimize the summation of false alarm and missed detection probabilities.

References

[1] F. Koushanfar, "Hardware metering: A survey," in *Introduction to Hardware Security and Trust*. Springer, 2012, pp. 103–122.

[2] S. Adee, "The Hunt For The Kill Switch," *Spectrum, IEEE*, vol. 45, no. 5, pp. 34–39, 2008.

[3] J. Rajendran, E. Gavas, J. Jimenez, V. Padman, and R. Karri, "Towards a comprehensive and systematic classification of hardware trojans," in *Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on*, 2010, pp. 1871–1874.

[4] Y. Jin, N. Kupp, and Y. Makris, "Experiences in Hardware Trojan design and implementation," in *Hardware-Oriented Security and Trust, 2009. HOST '09. IEEE International Workshop on*, 2009, pp. 50–57.

[5] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan Detection using IC Fingerprinting," in *Security and Privacy, 2007. SP '07. IEEE Symposium on*, 2007, pp. 296–310.

[6] D. J. Wheeler and R. M. Needham, "TEA, a tiny encryption algorithm," pp. 363–366, 1995.

[7] V. Venugopal and D. M. Shila, "High throughput implementations of cryptography algorithms on GPU and FPGA," in *Instrumentation and Measurement Technology Conference (I2MTC), 2013 IEEE International*, May 2013, pp. 723–727.

[8] V. Venugopal and D. Manikantan Shila, "Hardware acceleration of TEA and XTEA algorithms on FPGA, GPU and multi-core processors," in *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*, ser. FPGA '13. New York, NY, USA: ACM, 2013, pp. 270–270.

[9] T. Huffmire, B. Brotherton, G. Wang, T. Sherwood, R. Kastner, T. Levin, T. Nguyen, and C. Irvine, "Moats and drawbridges: An isolation primitive for reconfigurable hardware based systems," in *Security and Privacy, 2007. SP '07. IEEE Symposium on*, May 2007, pp. 281–295.

[10] S. T. King, J. Tucek, A. Cozzie, C. Grier, W. Jiang, and Y. Zhou, "Designing and implementing malicious hardware." *LEET*, vol. 8, pp. 1–8, 2008.

[11] Y. Jin and Y. Makris, "Hardware Trojans in Wireless Cryptographic ICs," *Design Test of Computers, IEEE*, vol. 27, no. 1, pp. 26–35, 2010.

[12] S. Wei, K. Li, F. Koushanfar, and M. Potkonjak, "Hardware trojan horse benchmark via optimal creation and placement of malicious circuitry," in *Design Automation Conference (DAC), 2012 49th ACM/EDAC/IEEE*, 2012, pp. 90–95.

[13] S. Srinivasan and V. Narayanan, "Variation aware placement for FPGAs," in *Emerging VLSI Technologies and Architectures, 2006. IEEE Computer Society Annual Symposium on*. IEEE, 2006, pp. 2–pp.