# Enhancing Trust in Reconfigurable Hardware Systems

Vivek Venugopalan

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Cameron D. Patterson, Chair
Peter M. Athanas
Paul E. Plassmann
Jeffrey H. Reed
Kevin A. Shinpaugh

February 3, 2017
Blacksburg, Virginia

Keywords: Secure, Trusted, & Resilient Computing, Root of Trust, Hardware Trojans,
Cyber Physical System Security, Embedded Systems, Reconfigurable Hardware

# Enhancing Trust in Reconfigurable Hardware Systems

Vivek Venugopalan

## Abstract

A Cyber-Physical System (CPS) is a large-scale, distributed, embedded system, consisting of various components that are glued together to realize control, computation and communication functions. Although these systems are complex, they are ubiquitous in the Internet of Things (IoT) era of autonomous vehicles/drones, smart homes, smart grids, etc. where everything is connected. These systems are vulnerable to unauthorized penetration due to the absence of proper security features and safeguards to protect important information. Examples such as the typewriter hack involving subversive chips resulting in leakage of keystroke data and hardware backdoors crippling anti-aircraft guns during an attack demonstrate the need to protect all system functions. With more focus on securing a system, trust in untrusted components at the integration stage is of a higher priority.

This work builds on a red-black security system, where an architecture testbed is developed with critical and non-critical Intellectual Property (IP) cores and subjected to a variety of Hardware Trojan Threats (HTTs). These attacks defeat the classic trusted hardware model assumptions and demonstrate the ability of Trojans to evade detection methods based on physical characteristics. A novel metric is defined for hardware Trojan detection, termed as HTT Detectability Metric (HDM) that leverages a weighted combination of normalized physical parameters. Security analysis results show that using HDM, 86% of the implemented Trojans were detected as compared to using power consumption, timing variation and resource utilization alone. This led to the formulation of the security requirements for the development of a novel, distributed and secure methodology for enhancing trust in systems developed under untrusted environments called FIDelity Enhancing Security (FIDES). FIDES employs a decentralized information flow control (DIFC) model that enables safe and distributed information flows between various elements of the system such as IP cores,

physical memory and registers. The DIFC approach annotates/tags each data item with its sensitivity level and the identity of the participating entities during the communication.

Trust enhanced FIDES (TE-FIDES) is proposed to address the vulnerabilities arising from the declassification process during communication between third-party soft IP cores. TE-FIDES employs a secure enclave approach for preserving the confidentiality of the sensitive information in the system. TE-FIDES is evaluated by targeting an IoT-based smart grid CPS application, where malicious third-party soft IP cores are prevented from causing a system blackout. The resulting hardware implementation using TE-FIDES is found to be resilient to multiple hardware Trojan attacks.

# Enhancing Trust in Reconfigurable Hardware Systems

Vivek Venugopalan

## General Audience Abstract

The Internet-of-Things (IoT) has emerged as one of the most innovative multidisciplinary paradigms combining heterogeneous sensors, software architectures, embedded hardware systems, and data analytics. With the growth in deployment of IoT systems, security of the sensors and trustworthiness of the data exchanged is of paramount significance. IoT security approaches are derived from the vulnerabilities existing in cyber-physical systems (CPS) and the countermeasures designed against them. An unauthorized penetration due to the absence of safeguards can cripple the system and leak sensitive data.

This dissertation studies the vulnerabilities posed due to the presence of hardware Trojans in such IoT-based CPS. FIDelity Enhancing Security (FIDES), named after the Greek Goddess of Trust, is a novel, distributed and secure methodology proposed to address the security requirements and enhance trust of systems developed in untrusted environments. FIDES utilizes a distributed scheme that monitors the communication between the Intellectual Property (IP) cores using tags.

Trust Enhanced FIDES (TE-FIDES) is proposed to reduce the vulnerabilities arising from the declassification process of the third-party soft IP cores. TE-FIDES employs a secure enclave approach for preserving the integrity of the sensitive information in the system. In addition, TE-FIDES also uses a trust metric to record snapshots of each IP core's state during the declassification process. TE-FIDES is evaluated by mapping an IoT-based CPS application and subjecting it to a variety of hardware Trojan attacks. The performance costs for resilient and trustworthy operation of the TE-FIDES implementation are evaluated and TE-FIDES proves to be resilient to the attacks with acceptable cyber costs.

To my *Amma*,

who will always remain etched in my memory forever;

to my wife, *Mansi*

who has been the pillar of my strength;

and my daughter, *Maya*

who has taught me how to love unconditionally.

# Acknowledgements

*"The important thing in Life is not triumph, but the struggle; the essential thing is not to have conquered but to have fought well." — Pierre de Coubertin*

---

A dissertation is a marathon with intermediate milestones and graduate students are like runners. We all move forward with our arms and legs swaying, run through the pain and keep our head up. I have been blessed to meet a lot of extraordinary people during my marathon who have taught me poise, patience and above all motivated me to reach one of many finishing lines.

First and foremost, I would like to thank my advisor Prof. Cameron D. Patterson for his invaluable guidance and infinite patience. He has helped me improve my research and technical writing skills. I would like to express my sincere gratitude to him for the considerable time he has invested in reviewing this dissertation. I have learned a lot of valuable lessons from Dr. Patterson and I hope to continue working with him well past this milestone.

I would like to thank my committee members Prof. Peter M. Athanas, Prof. Paul E. Plassmann, Prof. Jeffrey H. Reed, and Prof. Kevin A. Shinpaugh for their valuable comments and feedback that helped shape this dissertation. I especially thank Dr. Shinpaugh for introducing me to High Performance Computing and providing me an opportunity to work with hardware accelerators at the Advanced Research Computing facility in Virginia Tech.

I worked on my dissertation while being employed full time at United Technologies Research Center (UTRC). I am thankful to my colleagues and friends: Parag Kshirsagar, Shashank Krishnamurthy, Xin Wu, Devu Manikantan Shila, Kishore Reddy, and Cagatay Tokgoz. Thank you Parag for mentoring me at UTRC and being a constant source of inspiration during my PhD. Thank you Shashank for giving me an opportunity to work with FPGA-based power electronics applications. Thank you Xin for inspiring me to run my first

half marathon and I have been hooked ever since. I am indebted to Dr. Isaac Cohen, Dr. Brian Murray, and Dr. David Parekh for giving me the time flexibility to complete this dissertation.

At Virginia Tech, my friends have supported me in my toughest times. Manas, Neeraj, Jairaj, Prasad, Rinu, Pranjal, Arun, Raghu, Suraj, Vijesh, Amit (Mavle) have provided great company. I would also like to thank my NJ travel companions Ranjana and Aparna who helped me endure the long 8 hour drive from Blacksburg. Many of my close friends from Wright State: Beena, Kiran, Neel, Kashmira, Ravi, and Salil have always encouraged me and also served as a sounding board when my morale was low.

I would like to thank my grandmother *Amma* who gave me a lot of love while growing up. I lost her a few years back and I hope she is still looking out for me from wherever she is and is proud of my achievements. Most of all, I met my life partner and my wife Mansi at Virginia Tech and I am indebted to all the sacrifices she has endured while supporting me to complete my dissertation. I am also thankful to Saasu Dad, Saasu Maa, Mandar, and Shirin who welcomed me with open arms and always gave me words of encouragement and support. I am grateful to my niece Meher and my daughter Maya for giving me moments of happiness and breaking the monotonousness. Last but not the least, thank you Mom, Dad, and Vineeta for believing in me.

Thank you to every one who has touched my life in various ways !

# Contents

# List of Figures

# List of Tables

# Glossary

**AES** Advanced Encryption Standard.

**ASIC** Application Specific Integrated Circuit.

**COTS** Commercial Off-The-Shelf.

**CPS** Cyber-Physical System.

**CPU** Central Processing Unit.

**DARPA** Defense Advanced Research Projects Agency.

**DoS** Denial of Service.

**DSP** Digital Signal Processor.

**EDA** Electronic Design Automation.

**EMI** Electro-Magnetic Interference.

**FIDES** FIDelity Enhancing Security.

**FPGA** Field Programmable Gate Arrays.

**GPU** Graphics Processing Unit.

**HDL** Hardware Description Language.

**HTT** Hardware Trojan Threat.

**IC** Integrated Chip.

**IoT** Internet of Things.

**IP** Intellectual Property.

**MITM** Man-In-The-Middle.

**OS** Operating System.

**RoT** Root of Trust.

**RTL** Register Transfer Logic.

**SoC** System-On-Chip.

**TEA** Tiny Encryption Algorithm.

**UAV** Unmanned Autonomous Vehicle.

**XTEA** Extended Tiny Encryption Algorithm.

# Chapter 1

# Introduction

*"We are all now connected by the Internet, like neurons in a giant brain" — Stephen Hawking*

---

Cybersecurity encompasses "cyber-physical" security of systems, software security and hardware security. The boundary of hardware and software security in a Cyber-Physical System (CPS) is very blurred and in most cases, overlaps by a huge margin. A typical CPS consists of embedded devices such as Field Programmable Gate Arrays (FPGA), ARM-based System-On-Chip (SoC) or a combination of both. In a mission-critical CPS, trustworthy operation of the embedded system is of the utmost importance. However, it is difficult to quantify the measure of trust enforced by the different parts of a CPS on the system-level, software-level and hardware-level.

## 1.1 Cyber Physical System Attacks

An advanced cyber attack such as Stuxnet has demonstrated that malware that can be triggered to destroy critical infrastructure (e.g. in this case the destruction of centrifuges in

a nuclear reactor) [1]. The Stuxnet worm was introduced by infected USB flash drives [2] and was able to infect other computers inside the facility using peer-to-peer connections [3]. This incident demonstrates the futility of enforcing trust in isolation, where the network is itself isolated from the outside environment. The Stuxnet worm has also been claimed to be a part of a joint US-Israeli operation codenamed "Operation Olympic Games" [4] as per Edward Snowden's revelations to the German newspaper "Der Spiegel" [5].

Investigators have found evidence of malware in systems that control U.S. power transmission grids, wind turbines, nuclear power plants, water treatment plants and the oil/gas infrastructure [6]. A malicious hardware Trojan was used to disable a Syrian anti-aircraft missile system in the event of an air-attack as per Operation Orchard [7]. A more recent non-invasive approach describes the capture of the Lockheed RQ-170 drone in northeastern Iran based on a Man-In-The-Middle (MITM) attack involving GPS spoofing [8]. Another attack that crippled cars connected to the Internet through entertainment systems has been recently documented [9]. These examples show possible vulnerabilities for a CPS deployed in mission-critical scenarios.

The above attacks describe the instances of attacks to a CPS using malware, Hardware Trojan Threat (HTT) for Denial of Service (DoS) and unmonitored I/O ports. HTT have been designed to mainly cripple a system and leak information. One of the first instances of sensitive information leakage using HTT was through an espionage effort known as Project GUNMAN during the Cold War [10]. Sixteen IBM Selectric typewriters at the U.S. Embassy in Moscow and U.S. Mission in Leningrad were tampered such that every keystroke from these typewriters were transmitted by radio to nearby Soviet listening posts. The HTT attack was ingenious consisting of a HTT chip with implanted antenna and was uncovered using X-ray equipment by a team of more than twenty-five engineers from the National Security Agency (NSA). Such HTT attacks arise from a "subversive chip" that is capable of destroying the processor and disabling the system by shunting power to ground or leaking sensitive information.

## 1.2 Assumptions in a CPS Hardware Platform

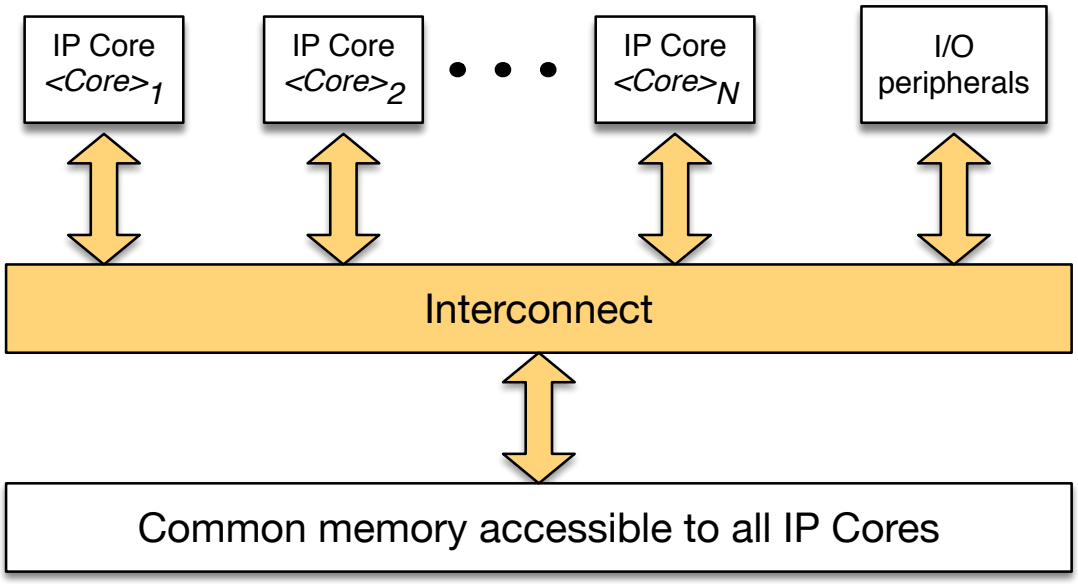


Figure 1.1: Untrusted FPGA-based CPS Hardware Platform

This dissertation targets a CPS hardware platform built using a FPGA-based SoC that consists of a combination of general hard IP and specialized third-party soft IP cores. Figure 1.1 shows an untrusted FPGA-based CPS platform with third-party soft IP cores. The IP cores have access to common memory locations for accessing data which may be sensitive in nature. The I/O peripherals are accessed by the IP cores through an interconnect such as AXI. The AXI interconnect is a subset of the ARM Advanced Microcontroller Bus Architecture (AMBA) and it permits synchronized communication between different components. Partial reconfiguration of the AXI interconnect is not permitted in this platform. The Xilinx Zynq SoC consisting of both ARM and FPGA regions is used in this work and is described in Section 2.7.

It is truly difficult to build a secure and resilient hardware platform without underlying assumptions on trust in the hardware, software, tools, or even the designer. In this dissertation, the tools and the designer are considered to be trusted whereas the third-party soft

IP cores are considered untrusted. These third-party IP cores are obtained by the designer in the form of obfuscated netlists with datasheets providing interface, latency, and resource utilization information. In some cases, timing-accurate simulation models are also provided to test for system correctness and verify the functionality of the IP cores.

The attack surface in Figure 1.1 consists of the insecure memory locations accessible by all IP cores and the possibility of a "rogue" IP core with a HTT designed to leak the memory information through unmonitored I/O peripherals. Other attacks also include *Always On* HTT, DoS HTT, MITM HTT that are described in Section 2.3. The exceptional case of a HTT that operates as expected but leaks sensitive information through the normal outputs without affecting latency is not addressed in this dissertation. Also, a HTT that can leak information by modulating something measurable is not detected. On the other hand, message spoofing by third-party IP cores after accessing data from I/O peripherals is detected by the safeguards and trust extensions proposed in Chapter 7.

## 1.3 Dissertation Overview

This dissertation looks at the evolution of a robust and trustworthy architecture FIDelity Enhancing Security (FIDES) starting with a testbed consisting of cryptographic IP cores subjected to different Hardware Trojan attacks. In contrast to existing efforts, this work investigates a system model from a designer perspective to increase the security of the device and an adversary model from an attacker perspective to expose and exploit vulnerabilities in the device. Using existing Trojan implementations and Trojan taxonomy as a baseline, seven HTTs were designed and implemented on the testbed. These Trojans generate a variety of threats ranging from sensitive information leakage, DoS, and defeat 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 57% of designed HTTs

were detected. The cryptographic testbed provides a platform for implementing various HTTs and comparing the detection techniques described in existing literature. Any IP core can be used to demonstrate the drawbacks in the existing detection methods; the selection of a cryptographic core highlights the leakage of keys (considered as sensitive information). A novel metric for hardware Trojan detection called HTT *detectability metric* (HDM) is proposed 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 Integrated Chip (IC) will be tagged as malicious. HDM is not a solution for detecting all HTTs, as the legitimate trigger HTT remains undetected.

FIDES utilizes a combination of access control policies and behavior learning techniques for anomaly detection. FIDES enforces trust within a system by: (i) deploying a layer of monitors that capture the interactions of the IP cores using a tagging and tracking mechanism; and (ii) embedding a supervisory watchdog module that monitors the flow of information with regards to the security policies. FIDES builds on the lessons learned from the vulnerabilities of the RoT architecture when subjected to the HTT. FIDES enforces "trust" by employing a distributed control flow for validating and verifying the information exchanged between the IP cores and the I/O peripherals. FIDES examines the adversary/threat model and derives the attack scenarios along with the security requirements. However, FIDES is vulnerable to an attack if an IP core performs an unexpected behavior during the declassification process. The *tag declassification* rule enables the higher tag to declassify to the lower tag to receive information. Moreover, it is important to track the specific IP core being declassified, in case further diagnostics are required to identify the malicious core. To mitigate this vulnerability and keep track of all IP cores communicating with each other, trust enhanced FIDES (TE-FIDES) is proposed. In TE-FIDES, each IP core's trust metric is defined and tracked to preserve the integrity of the system during the declassification process.

The dissertation is organized as follows:

- Chapter 2 presents the background work and compares the different secure and trustworthy solutions by various research groups. This chapter also surveys the RoT hardware architecture and a taxonomy of the different HTTs. A brief introduction to the selected hardware implementation platform is also presented. This chapter has also been submitted to the Special Issue on Security and Privacy in Cyber-Physical Systems, IEEE Internet of Things Journal.

- Chapter 3 is a manuscript presented at the 2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) and the 2013 ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA 2013). This work developed a low latency, high throughput implementation of the Tiny Encryption Algorithm (TEA) and Extended Tiny Encryption Algorithm (XTEA) cryptographic algorithms. These algorithms are mapped on hardware platforms such as FPGA and Graphic Processing Unit (GPU) with experimental results providing comparisons on overall latency and throughput.

- Chapter 4 is a manuscript presented at the 2014 IEEE International Conference on Communications (ICC) and the 2014 ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA 2014). This work explored the vulnerabilities on a cryptographic testbed when subjected to seven different HTTs. These HTTs can be designed optimally and placed inside a device masked within its golden parameter profiles such as power consumption, timing variation/latency (clock cycles) and resource utilization (gate count). A HTT detectability metric (HDM) is proposed that leverages a weighted combination of normalized golden physical parameters. Experimental results show the effect of each HTT on the testbed to defeat the classic "trusted" hardware model assumptions.

- Chapter 5 is a collection of manuscripts presented at the 2015 IEEE High Performance Extreme Computing Conference (HPEC 2015), 9th International Conference (NSS 2015) and the 2014 ACM/SIGDA International Symposium on Field Programmable

Gate Arrays (FPGA 2014). This work defines FIDES and the need for an anomaly detection capability within FPGAs to continuously monitor the behavior of the embedded IP cores and their communication with I/O peripherals. The decentralized flow of information is highlighted with a distributed monitoring scheme that tags the communication between the IP cores. Experimental results demonstrate the efficacy of the architecture when deployed to detect, prevent and track different HTTs.

- Chapter 6 is a manuscript presented at the 2016 IEEE International Workshop on Cyber-Physical Systems Security (CPS-Sec). This work examines the requirement of a secure enclave-based approach in an existing CPS. Existing implementations that employ secure enclaves are discussed. TE-FIDES is proposed to mitigate the vulnerability arising from IP core declassification process. In addition, TE-FIDES enforces a secure enclave approach to preserve the confidentiality and integrity of the sensitive information in the presence of an attack. TE-FIDES highlights the resiliency of a smart grid CPS to attacks emanating from the introduction of third-party soft IP cores. The experimental results demonstrate the acceptable latency overheads incurred when deploying a smart grid CPS application using TE-FIDES.

- Chapter 7 is a manuscript submitted to 2017 3rd ACM Cyber-Physical System Security Workshop (CPSS 2017). This work addresses the shortcomings in the existing security requirements based on the CIA-Triad of confidentiality, availability, and integrity. An updated set of metrics based on the IAS-Octave set is used to define the security requirements for an industrial Internet of Things (IoT) system. An IoT-based CPS consisting of a motor application is implemented using TE-FIDES and subjected to multiple hardware Trojan attacks. TE-FIDES is shown to be resilient to the attacks with acceptable cyber costs.

- Chapter 8 presents the conclusions and future work. This chapter also includes a list of the relevant publications resulting from this work.

# Chapter 2

# Literature Survey

*"If I have seen further than others, it is by standing upon the shoulders of giants." — Isaac Newton*

---

Cyber-Physical Systems (CPS) are comprised of a large-scale network of embedded nodes, where these nodes can be embedded platforms. Such embedded platforms consist of processors or FPGAs integrated to execute the different functions within the system. Hardware Trojans can be introduced at different stages of the integration or chip lifecycle and easily cripple a CPS. Within this context, it is essential to understand the HTT insertion points, their activation triggers, detection methods and countermeasures.

This chapter is organized as follows: Section 2.1 gives a brief introduction to hardware Trojans, Section 2.2 describes the entry points for hardware Trojans within an IC's lifecycle, Section 2.3 provides a taxonomy of hardware Trojan threats and their trigger mechanisms, Section 2.4 explains the various Trojan detection techniques, and Section 2.5 lists the countermeasures to prevent Trojan attacks. Section 2.6 surveys the trust primitives, software assurances, and enclave-based approaches implemented in existing platforms. Finally the Zynq hardware platform for mapping the different detection techniques/countermeasures is

elaborated in Section 2.7.

## 2.1 Introduction to Hardware Trojans

Economic reasons dictate that a major share of the Application Specific Integrated Circuit (ASIC) and FPGA chips are nowadays manufactured in offshore facilities and involve IP cores supplied by many third-party IP providers [11, 12, 13]. Companies also rely on outsourced design and test services, and employ Electronic Design Automation (EDA) tools provided by different vendors to lower production costs and drive higher profits. One study has shown that about 78% of FPGAs and 60% of ASICs are in fact manufactured in China and Taiwan [14] as shown in Figure 2.1. Such a business model provides an adversary with numerous opportunities to tamper the hardware with counterfeit or malicious modifications, referred to as hardware Trojan horses that in turn have the potential to incapacitate the hardware functions used in critical applications. There are multiple solutions for addressing these supply chain vulnerabilities: (i) All ICs can be manufactured in trusted foundries; and (ii) Identify the ICs that are critical to specific missions related to national security. However, these solutions are not feasible and affordable at the scale required to sustain the production of these ICs.

A hardware Trojan is defined as a deliberate and malignant modification of a circuit resulting in erroneous operation of the circuit when activated. Since the hardware Trojan is embedded in the lowest level within the device, the threat of the hardware Trojan is severe as long as the device is in operation. A hardware Trojan alters the behavior of an infected IC by thwarting the software or hardware security measures present inside the IC. Any type of IC, including Digital Signal Processor (DSP), ASICs, SoCs, and FPGAs are vulnerable to such malicious modifications. Several undesired effects result from such modifications to the IC, such as the leakage of critical and sensitive information, DoS attacks, and undetected service degradation resulting in failure of the system. Hardware Trojans can also provide

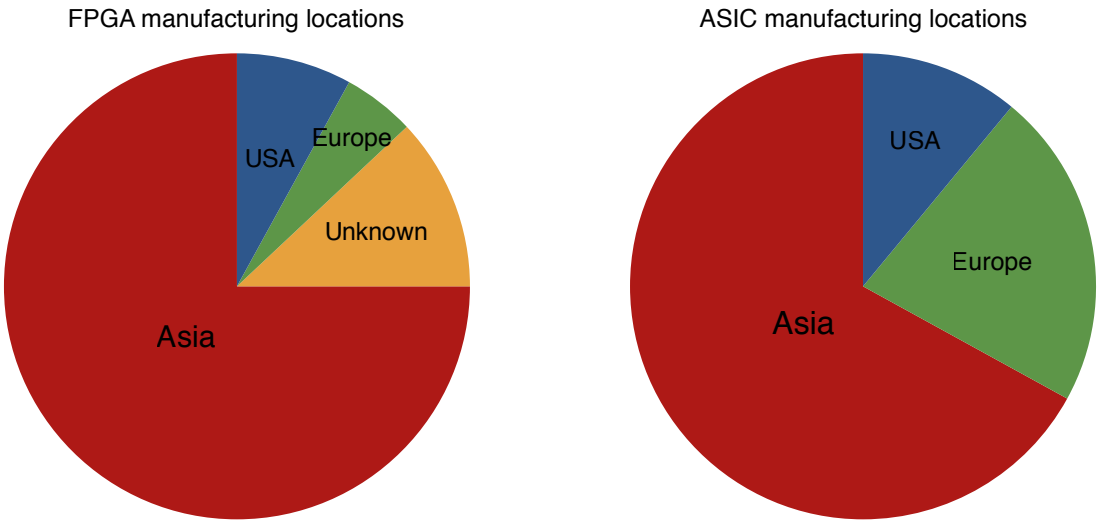


Figure 2.1: FPGA and ASIC manufacturing locations

backdoor access to a system, where the high-level software works in conjunction with the hardware Trojan to access and disable highly secure systems [15, 16]. Hardware Trojans have been classified and described in great detail in previous surveys [11, 17]. To understand the scope of the Trojans studied in this dissertation, it is necessary to examine the following questions:

1. What are the insertion points for HTTs?
2. What are the threats caused by hardware Trojans and what are their trigger mechanisms?
3. What are the existing HTT detection techniques?
4. What countermeasures exist for preventing the activation of HTTs?

The following sections survey existing work seeking to answer the above questions.

## 2.2 Hardware Trojan Insertion

The design and manufacture of an IC consists of several steps such as specification, Register Transfer Logic (RTL) design, synthesis, fabrication, testing and assembly. All these stages directly influence the approach taken by an adversary to introduce a HTT [18]. The system functionality and its usage model are defined during the specification stage. The RTL design stage captures the system behavior. This step also involves the integration of third-party IP cores that provide a specific sub-functionality. The integrated system model is synthesized into the target technology with an emphasis on the functional and physical constraints imposed during the specification and design steps. These constraints are verified when the IC mask set is created and the wafers are produced. The wafers are cut into die, packaged and tested before being shipped or deployed. All these stages provide an insertion point for a HTT, as third-party vendors provide IP cores, design tools and the manufacturing and test facilities. Hence, the complete IC design lifecycle can be considered as vulnerable and requiring effective detection and prevention techniques.

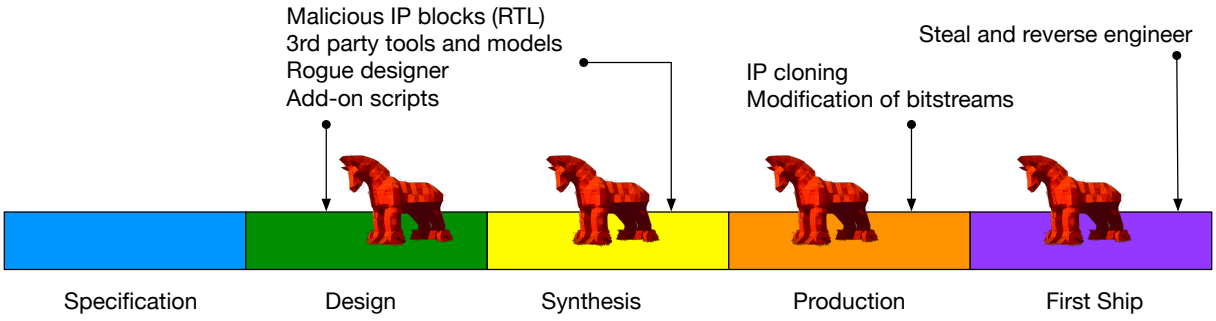


Figure 2.2: FPGA design cycle illustrating Trojan entry points

Figure 2.2 shows the different entry points for HTTs from the perspective of a FPGA. A HTT may be inserted by adding third-party IP cores for I/O interface blocks (e.g. PCI controllers). In some cases, third-party IP cores are available only in the form of a netlist that abstracts the RTL code and prevents unauthorized snooping into the IP core contents. This vulnerability can easily be exploited by an untrusted party by embedding rogue functionality

within this "blackbox" IP core. The FPGA bitstream is vulnerable to cloning and can be modified to introduce malware. By modifying the bistream the malware need not be limited to the IP cores and can be distributed across various blocks within the design such as memory and clock grid [18]. Although not a vulnerability in terms of system security, the modification of the FPGA bitstream can also enable stealing or reverse engineering. Eventually, the reverse engineered design can be counterfeited to become a serious supply chain vulnerability.

## 2.3 Hardware Trojan Threats and Triggers

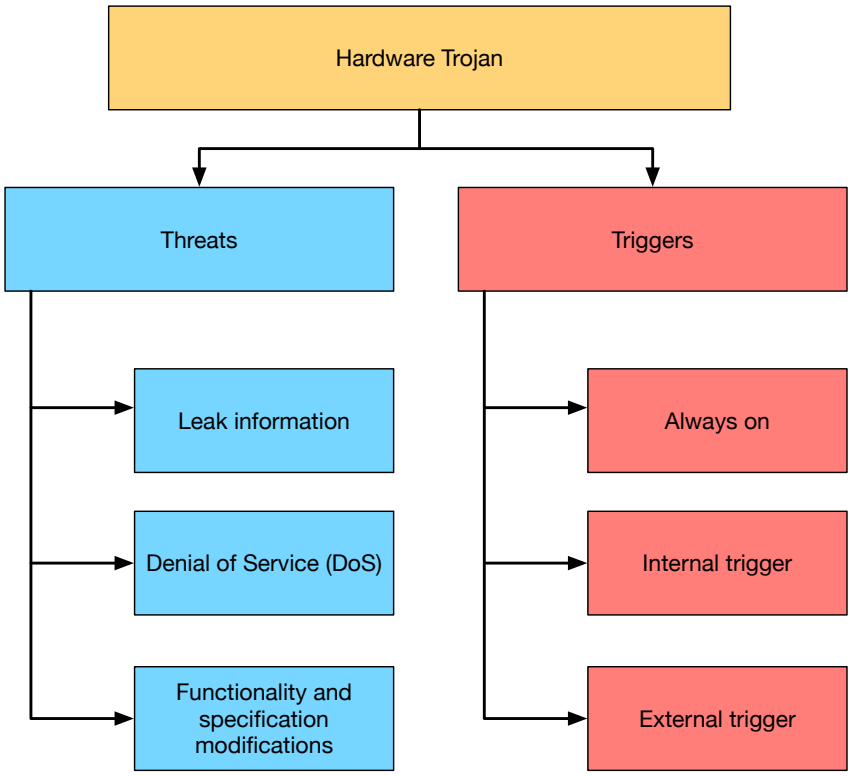


Figure 2.3: Hardware Trojan threats and triggers

Hardware Trojans are persistent; the threat remains inside the system once infected and has the capability to cripple the system as long as the system is powered on. Section 2.2

describes the insertion points for these HTTs and their influence in the type of security attack. HTTs extend the range of the system attack surface which initially focuses on the software vulnerabilities. This poses a very challenging scenario: software security measures are implemented on the premise of trust in the underlying hardware and these measures can easily be bypassed by the malicious HTT.

In order to understand the broad range of threats posed by these HTTs, it is imperative to classify these threats and their trigger mechanisms [19]. Figure 2.3 shows the main types of HTTs and the different ways they can be triggered. Section 2.3.1 describes the effects of leaking information within a system. HTTs disrupting the normal operation of a system using DoS methods are explained in Section 2.3.2. Modification of the system's functionality or specification by HTTs are described in Section 2.3.3.

Hardware Trojans are designed such that they remain quiescent until triggered, after which they perform malicious activity to cripple the targeted system. The activation mechanism can be of any type: direct, random or predetermined. The activation phase is the most relevant stage for formulating strong detection methods, as effective counteractive actions can then be adopted to nullify or flag the trigger. Section 2.3.4 describes the effect of a system that can be crippled by always active HTT. Different types of internal triggers are categorized in Section 2.3.5, while external triggers mechanisms are discussed in Section 2.3.6.

### 2.3.1 Threat: Leak Information

One of the most common types of HTT is leakage of critical information, which aims to transmit sensitive information to an adversary without the knowledge of the system. In some cases, the sensitive data can also be hidden within the status messages authorized by the user/system. Interfaces such as RS232 and JTAG are the most common I/O ports used to leak information, whereas other means such as radio frequency, optical, thermal and power have been used [20]. The information transmission may also be hidden within the

noise margins of the circuit's normal operation.

A novel way of leaking information was demonstrated in [21], where a spread spectrum side-channel technique was used to leak data below the CMOS process noise floor. This study used a technique described as Malicious Off-chip Leakage Enabled by Side-channels (MOLES) implemented in an AES cryptographic module for a 45 nm CMOS process. This technique was considered novel due to its ability to resist detection methods such as optical inspection, functional testing and device fingerprinting analysis.

Another instance of information leakage is provided in [22, 23], where a DES encryption core was attacked by a hardware Trojan. The Trojan extracts the 56-bit encryption key and leaks it by hiding one bit in the block of transmitted data. Assuming that the encryption module is called 56 times, the entire key is transmitted thus compromising the encryption process. The Trojan attack was able to pass validation testing and was demonstrated during the 2008 Cyber Security Awareness Week (CSAW) Embedded Systems challenge.

### 2.3.2 Threat: Denial of Service (DoS)

DoS is a threat resulting from an embedded HTT that has disabled the IC or the system incorporating the infected IC. This type of Trojan came into prominence when Syrian anti-aircraft guns were disabled through a backdoor implemented within the electronics system ICs [7, 10]. It was suggested that a "kill switch" executed a DoS type attack by completely disabling the system. There are various ways of implementing DoS Trojans. By increasing the power consumption and preventing a circuit from reverting to an idle mode or sleep state, a battery powered system can easily be disabled [24]. In some cases, a DoS HTT can affect the system by exhausting critical resources required for the operation of the system such as bandwidth, computational capability, or battery power in case of an IoT device [20]. A latent form of DoS HTT has been described in [25], where the IC's lifespan is reduced by excessive clocking activity that accelerates the aging process of the IC. This HTT example does not affect the functionality, but can cause an economic burden when ICs have to be

continuously replaced.

### 2.3.3 Threat: Functionality and Specification Modifications

A hardware Trojan can compromise the integrity of an IC by modifying the IC's functionality. The IC may perform unintended functions such as modifying the contents of memory, or introducing errors into the computation. The HTT is implemented by removing or bypassing the existing modules, or through additional modules. An example where the IC is disabled after a counter reaches a certain threshold has been described in [26]. In addition to the threshold counter, the HTT implementation also consists of a comparator that monitors the data bus against a fixed value and alters the computation when it finds a match. The utilization of simple circuits such as counters and comparators makes it difficult to detect the HTT, as the gate count for these extra circuits is within the tolerance of the IC's complete resource utilization.

The "Illinois malicious processor" is an instance of a malicious HTT infecting a Central Processing Unit (CPU) [16]. The functional modifications to the CPU consisted of only two changes: (i) the protected memory region is exposed to outside attackers through a memory access scheme; and (ii) the presence of a *shadow mode* which can bypass the normal operation of the CPU and allow execution of hidden firmware. These changes were introduced at the Hardware Description Language (HDL) stage and were synthesized and mapped to a 40MHz Leon 3 SPARC platform. Since the platform is compromised, the software executing on the CPU exposes the system to various attacks such as stealing passwords, enabling privilege escalation, and unauthorized login to allow complete and high level access to the system.

HTT variations due to functional modifications are limitless; depending on the information that needs to be accessed or the type of attack required to cripple a system, an adversary can always fashion a HTT to execute the specific type of attack. System integrity can be affected by: altering the order of instructions being executed by a CPU, data leakage, modification of a system's memory contents, or error correction module manipulation to accept

erroneous inputs [18].

A system can also be crippled by a HTT that is capable of modifying the system's core specifications or requirements. Clock signal manipulation can affect the timing parameters of a system, e.g. a system that needs to be clocked at 100 MHz may be only clocked at 90 MHz, introducing timing errors. During the IC fabrication, intrinsic specifications such as wire length and transistor geometry can be modified to disrupt the IC performance once deployed [19]. Specific computational modules within a IC may be easily replaced with functionally equivalent circuits exhibiting higher latency and reduced throughput. In the case of FPGAs or ASICs, the clock or I/O buffers may be altered irrespective of the correct specification. This results in chip degradation especially if the buffers cannot withstand the high load activity on the affected wires.

### 2.3.4 Trigger: Always On

HTTs are usually activated by specific events happening inside or outside a system. However, some HTTs are designed such that they do not require a trigger for activation. These HTTs affect a system by altering its specification, functionality or timing as described earlier in Section 2.3.3. *Always On* Trojans have subtle trigger mechanisms as described in the clocking scheme example. The modification of the IC's wire width or buffer load capacity makes the IC susceptible to failure over a long period of time [19]. In some cases, IC wafers are modified to degrade after a certain time period which may be within a few months to years of being in service [27]. The post-fabrication tests are not capable of detecting such time-based IC degradation effects due to the short duration of the tests. In addition, these HTTs can also be hidden by implementing them within the noise margin of the CMOS semiconductor manufacturing process. It is also very difficult to rely on a golden model to detect *Always On* Trojans, as power consumption or logic resource utilization of the affected system matches the golden model.

### 2.3.5 Trigger: Internal Activation

HTT internal activation is through a combination of events or events occurring in a sequence specified by the attacker. The HTT lies dormant until a certain value at the input or output of a module is detected. Once a match occurs, the HTT is activated to attack the host system. Sequentially triggered HTTs are dependent on a certain sequence of events to begin an attack. The design of internally activated HTTs require detailed study of the system to understand the probability of a sequence or pattern occurring that will successfully trigger a HTT. A HTT is activated only if a specific pattern occurs at a certain time as shown in an attack demonstrated during the 2008 CSAW Embedded Systems challenge [28]. The first trigger for the HTT was designed based on the combination of keys pressed within a specific time interval. A second thermal trigger was also embedded in this attack, where a certain input value was used to drive a ring oscillator generating heat. Another ring oscillator embedded in the system detects the delays due to the heat generated in the system and activates the HTT. Detecting one or more triggers obfuscated within the design requires a testing time of more than $3 \times 10^{35}$ years.

A combination of events such as the change to the encryption key and overflow buffer detection in a system served as the trigger for HTT, which in turn leaked critical information [22]. As described earlier in Section 2.3.1, the HTT remains active until the complete key is leaked – the length of the the key is tracked during leakage using an internal counter.

The sequential triggers for activating the HTT are complex to track and detect, e.g. changes in values of variables due to state machine transitions [25]. If a certain parameter increases above a set threshold, it can serve as an internal activation mechanism for a HTT. This hypothesis is investigated in the same study, where the charging and discharging of a capacitor inside a device triggers the HTT. Sequential changes to the physical properties of an IC can also serve as a source of internal activation as demonstrated in [20]. Sensors that can track the change in a IC's temperature or power consumption belong to this category of trigger. Specific input patterns used to access memory or control handshaking can trigger

HTTs [11]. A combination of patterns such as the value fetched from the memory or accessing a specific address location along with the state of certain internal registers can all contribute towards designing and implementing a very complex HTT trigger.

Synchronous hardware counters are used as trigger mechanisms for HTTs described in [12]. Asynchronous sequence counters can also be used, e.g. the transitions at the input or output of certain gates may activate HTTs. Specific codes in the data at specific clock cycles constituted the backdoor implemented in this study. A more comprehensive trigger can be designed by increasing the length of the sequences or the time interval at which the sequence appears. These steps may ensure that the infected system will remain operational for a long time and still remain vulnerable to a attack.

### 2.3.6 Trigger: External Activation

External events that can influence a system can provide the means of triggering an embedded HTT. The HTT activation can occur at any time due to the external source. The HTT embedded within the system is capable of reacting to the external event whenever it occurs, e.g. an antenna was embedded in a system that made it receptive to decoded signals sent by an attacker, which in turn cripples the system [19]. External activation can also occur through unmonitored I/O ports, where an external device can communicate with the embedded HTT [22]. On-board input devices such as push buttons or switches can be used as an external trigger to activate a HTT residing inside the IC [20]. On-chip sensors [11] have been advocated for monitoring the changes in external environment with respect to parameters that can influence an embedded HTT. The parameters that need to be monitored are diverse such as temperature, voltage levels, humidity, altitude, Electro-Magnetic Interference (EMI), etc. At the same time, it is not feasible to incorporate a multitude of sensors in a system/IC that checks for variations in the external environment.

## 2.4 Hardware Trojan Detection Techniques

It is difficult to prevent the insertion of HTTs into an IC during the design lifecycle. Countermeasures are deployed to prevent the insertion of Hardware Trojans, while detection techniques are used to identify the presence of Hardware Trojans in a system. Various techniques exist to detect HTTs, however some of them can accurately detect only a specific class of HTTs. Most of the research efforts have focused on detecting HTTs during the production stage of the design lifecycle as shown in Figure 2.2. The underlying assumption that third-party IP core designs can be trusted is incorrect. Figure 2.4 presents the different Hardware Trojan detection methods investigated in the taxonomy defined by Chakraborty *et al.* [25]. The detection techniques can be classified into mainly two categories: (i) destructive, and (ii) non-destructive. Section 2.4.1 describes the destructive approaches, while Sections 2.4.2 to 2.4.5 explain the various non-destructive techniques.

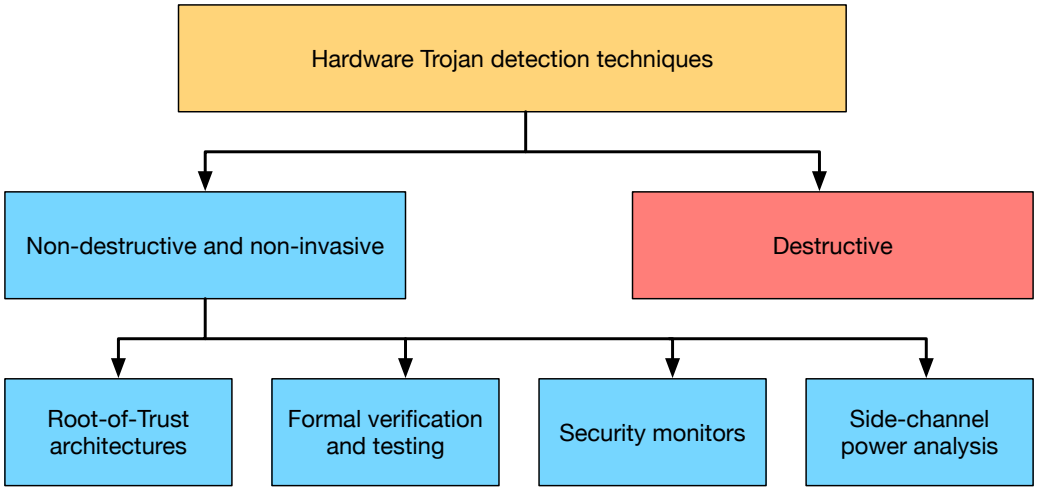


Figure 2.4: Hardware Trojan detection techniques

### 2.4.1 Destructive Approach

Destructive methods are not feasible to detect HTTs, as they destroy the IC. However, in some mission critical applications, a destructive approach is considered as a fail-safe method to reverse engineer the IC and search for any anomalies or malicious HTTs. Although a destructive approach seems the easiest to detect HTTs, it is actually very expensive and time consuming. On the other hand, chip decapsulation may indicate the presence of a counterfeit IC or a completely different IC with malicious functionality embedded inside [29]. Reverse engineering of an IC is done using image reconstruction after subjecting the suspicious IC to a scanning electron microscope. The images of the IC can be compared with the golden model of the IC to detect the presence of a HTT or an anomaly. The visual comparison fails when the HTT is added before the fabrication process. Also, reverse engineering a single IC is not an exhaustive method to detect HTTs since ICs are manufactured in different foundries and provide several insertion sources. It may not be economically feasible to sample a IC from all the various foundries or supply lines for a specific IC and subject it to destructive methods.

### 2.4.2 Root-of-Trust Architectures

Various studies have highlighted the detection of hardware Trojans. Root of Trust (RoT) in a system is the underlying hardware as the software runs on it. The RoT concept based on unused circuit identification (UCI) is described in the approach taken by Hicks *et al.* [30]. This work tags unused circuits to detect malicious hardware. The malicious activity on used wires is then communicated to the Operating System (OS) that functions as a supervisory controller. The BlueChip concept in this work was prototyped on a Xilinx Virtex-5 FPGA. However, the limitation of BlueChip is in its requirement for a trusted co-processor that can emulate the behavior of the removed/untrusted hardware. Also, malicious circuits were developed that could evade the UCI approach and also pass testing during the design phase [31].

RoT along with a variation of UCI is the primary trust mechanism in Functional Analysis of Nearly unused Circuit Identification (FANCI) [12, 32]. FANCI uses boolean functional analysis to identify the influence of wires or circuits on other wires or circuits in the design. A truth table format is used to save the average of the "control value". This approach is more feasible for the detection of backdoors and does not look for any event-based trigger HTT. Also, this approach fails to detect HTTs that produce incorrect computational results or are embedded inside control logic.

A dynamic RoT is proposed by UCI researchers in Secure and Minimal Architecture for Root of Trust (SMART) [33]. SMART is tailored towards a embedded microcontroller such as the TI MSP430 with software attestation using an embedded ROM. This architecture is simple and is not capable of providing any security guarantees in the presence of trigger-based attacks. The security is mainly provided through the verification of the bootup code embedded in the ROM.

VeriTrust is proposed to identify the trigger inputs in RoT system [34]. In this work, a tracer is used to identify unused inputs and a checker mechanism is used to analyze the signals generated by redundant inputs. The VeriTrust approach was able to identify more HTTs as compared to the UCI and FANCI approaches described above. However, VeriTrust suffers from several limitations: (i) VeriTrust fails to detect computational error or bug-based HTTs, as the trigger input is part of the circuit's normal operation; (ii) VeriTrust is not able to identify the HTTs that can be triggered using a functional input i.e. legitimate trigger; and (iii) VeriTrust misses the detection of always on HTTs that compromise design parameters such as timing, power or reliability.

### 2.4.3 Formal Verification and Testing

Ideally, with a golden model and complete system specification, it is possible to formally verify the design. This scenario will ensure that all HTTs inserted at the RTL level can be detected. Formal verification has been proposed as a tool to detect the presence of HTTs

and backdoors within a system [35]. However, the cost of formal verification is high and increases the design cost. Small systems or critical functionality can be tested using formal verification to assess the trustworthiness. At the same time, trust assessment through fault testing is inadequate [36]. Testing for faults can yield the faults within a circuit and fails in the case of trustworthiness, as an attacker can cleverly spoof a legitimate operation as an attack. A more correct implementation results when the formal model of a design is tied to its design specification. Representative of formal modeling and verification is the proof carrying codes method proposed by Jin *et al.* [37]. In this approach, the IP core vendor will construct a formal proof of the design adhering to certain security properties, which will be later verified by an IP consumer to ensure that design is free from modifications. Besides the overhead involved in creating large proofs for even smaller codes, loopholes can also occur while defining many security properties. The main loophole assumes that the third-party IP vendor is trustworthy and will not add Trojans.

### 2.4.4 Security Monitor-based Architectures

Centralized run-time monitors for enforcing security in embedded systems have been proposed. Design-for-Enabling-Security (DEFENSE) [38] adds reconfigurable logic to the functional design to implement a centralized run-time security monitor. The hardware-based monitors are configurable finite-state machines (FSMs) that check the current set of signals for the properties specified by the designer. In the event of an attack, DEFENSE deploys countermeasures to disable the suspect hardware modules. However, it is difficult to provide complete coverage of HTTs with run-time security checks.

HTTs are detected by executing the same function on multiple processing blocks [39]. The output of each of these blocks can be compared to identify the function affected by the HTT. The affected function can then be recomputed on other blocks or by another system. However, the infected hardware block may produce correct output and leak the critical information. Also, the IC performance can be degraded if the hardware block takes

excessive time for computation.

Isolation primitives proposed by Huffmire *et al.* [40, 41, 42] target FPGAs and helps monitor the communication flow within the FPGA. However, these approaches are vulnerable to a MITM and spoofing attacks. A centralized security monitor can easily be targeted and the cost incurred by the attacker is only the time to search for the monitor block/logic within the FPGA.

### 2.4.5 Side-channel Power Analysis

Side-channel power analysis examines the changes in the IC characteristics when a HTT is triggered. The power consumed by specific sections of the IC, changes in temperature and processing delay are part of the side-channel analysis technique.

Side-channel power analysis has been described as a technique to detect HTTs [26]. However, this method has two drawbacks: (i) where a system is being built by procuring third-party IP cores, it is assumed the system integrator has access to the IP core without the hardware Trojan; and (ii) the power analysis is adequate only for a small circuit with less than 1000 gates.

A sustained vector technique is used in [43], where certain inputs are repeated in order for a system to reach a stable state. The HTT can be detected by comparing the power consumption when new test vectors are applied. The increase in power consumption implies the presence of unnecessary hardware blocks which may contribute to the HTT.

Path delay is one of the parameters used in [44], where the HTTs have explicit or implicit payloads. Explicit payloads can alter the value of the data signal, whereas implicit payloads leak data through the side-channel or I/O. Similarly, both path delay and leakage current are used as side-channel analysis metrics in [45]. Ring oscillators are used as sensors to detect leakage current changes in [46]. However, the main limitation with all these approaches is the requirement of a golden model of the IC. This golden model provides the basis for

comparison between authentic and infected ICs. If the HTT is added in the design stage before the generation of a golden model, these approaches may not be effective.

## 2.5 Countermeasures for Hardware Trojans

Detection techniques and countermeasures are closely associated when targeting hardware Trojans. As new attacks present themselves to cripple a system, stronger countermeasures need to be designed to protect a system from the HTTs. A countermeasure is considered successful when the host system is resistant to the HTT attack. An attack is orchestrated to gain control of the system and execute malicious behavior or access sensitive information. In this context, data is of the utmost importance and hence countermeasures have evolved to guard sensitive data and prevent HTTs from accessing it.

### 2.5.1 Bus Guard

A custom SoC bus architecture for detecting the presence of a HTT has been implemented using the Altera Stratix-II FPGA with a hard-fused ARM core and AMBA bus [47]. This bus guard countermeasure is specifically designed for HTTs desiring to take control of the system bus. A bus arbiter typically gives the bus control to a master accessing memory (read/write operation) by asserting a wait signal to other slave modules. A HTT may exploit this process by issuing an indefinite wait signal and halting the system. A custom module known as bus matrix keeps track of the time after a wait signal is issued. If the bus still remains locked after a certain threshold time, the bus matrix issues a wait signal and classifies the rogue module as a malicious HTT. This specific countermeasure can be used to successfully blacklist suspicious master/slave modules connected to a system bus. However, an additional module on the system bus can result in delays and the authors fail to quantify the delay incurred by deploying the bus matrix.

The bus guard concept is extended further in the design of a Gatekeeper module for bus arbitration [48]. This specialized module encrypts the original address during a memory access operation and writes this encrypted address into a special register. By keeping track of the memory read and write from the encrypted memory address, the Gatekeeper module ensures the legitimate operation of the system bus. This approach exposes the presence of the HTT when it accesses a specific memory location and reads or writes from a different address. The biggest drawback here is that although secure read and write from memory is guaranteed, the Gatekeeper module is explicitly trusted. With no checks on the information being accessed, a MITM attack can render this countermeasure useless.

### 2.5.2 Data Guard

A variety of data guarding techniques have been studied by Waksman *et al.* [12, 35]. In case of a HTT being activated by an external trigger, bus scrambling has been used to prevent the HTT from receiving the activation message. Simple encryption schemes have been suggested to obfuscate the data that may be examined by a HTT for a specific pattern for its activation. The encryption scheme only targets memory access, ideally placed between the local memory and the memory controller. However, additional resources need to be employed for decryption, as incorrect results will be produced if the scrambled data is fed as input to the computational blocks inside an IC. Homomorphic encryption is suggested as an alternative for computational modules to process encrypted or scrambled data. Although homomorphic encryption can allow normal operation without affecting the overall latency, efficient implementation of such specialized encryption schemes is difficult.

### 2.5.3 Bitstream Encryption

In addition to protecting a system from HTTs, counterfeiting also poses a problem when mapping applications to reconfigurable devices. Encryption of configuration bitstreams [49]

is a reliable method adopted by all FPGA vendors. The integrity of the bitstream [50] can be preserved by the following steps: (i) read back the configuration from the bitstream and check for correctness; (ii) partially configure the FPGA in the event of an incorrect configuration from previous step; and (iii) utilization of a challenge-response inside the FPGA to identify the presence of a HTT. Reconfigurable logic blocks can be optimally placed within a design during the synthesis and bitstream generation stage [51]. The reconfigurable barriers are programmed by the end user using a secure key, thereby completing the design and making it functional. This approach may fail when the HTT is already embedded in the design before placing the reconfigurable barriers. Design obfuscation of the configuration bitstream has been employed as another approach to deter HTTs [52, 53]. The original netlist for the design is obfuscated using an algorithm that randomizes the state space of the design by selecting random state elements, state transitions and initialization sequences. The resulting RTL for the obfuscated design is then synthesized and realized as the bitstream. This countermeasure will fail if the Trojan is inserted into the original design via a third-party IP core.

### 2.5.4 Design Replication

Design replication through redundancy is considered as an effective countermeasure to detect the presence of HTTs in FPGAs [54]. The proposed scheme is built using the Triple Modular Redundancy (TMR) approach with only two replicas of the design. In case of a hardware Trojan activation, the replica of the affected module can take over the functional execution of the IC. This countermeasure is based on the following assumptions: (i) a golden model exists for the original design; and (ii) extra resources are available for replication of the design. Although TMR has been proposed as a secure scheme, this does not enforce trust in the design. Another instance of design replication is proposed, where multiple versions of the same functional module is procured from different IP core vendors [12]. The outputs from the different IP cores can be compared and used to effectively identify the rogue IP vendor. However, this approach is expensive and time-consuming when working with large

designs.

## 2.6 Notion of Security and Trust

Security and trust are two loosely coupled co-existing concepts in a CPS. Research efforts have focused on enforcing secure operation of a system in the presence of HTTs [12]. However, these efforts fail to quantify trust as a metric to guarantee the normal operation of a system. A system can be secured by incorporating various techniques to prevent the activation of HTTs and foil the attack [35]. Several techniques exist to secure a system and these approaches are targeted towards different attack surfaces [50, 51]. It is difficult to construct a truly secure system with every possible countermeasure without tradeoffs in terms of performance (latency, throughput, resource utilization). In this discussion on security, trust is not considered as a metric to evaluate the security of a CPS.

Various methods of approaching CPS security have been documented [47, 48], which try to preserve the confidentiality, integrity and availability of sensitive information in the presence of an attack. Focusing on the system's sensitive information blinds the detection technique to consider other attacks which may cripple the system. At the same time, system reliability analysis normally assumes random, uncoordinated failures rather than a systematic attack by determined and intelligent adversary who has complete knowledge of the system. Correct functioning of a CPS is often timing dependent, which can serve as another exploit. Backup controllers have been advocated as a means of securing a system when anomalies are detected [55]. The security of a feedback control system resilient to DoS attacks has been studied previously [56]. The DoS attack interrupts the communication between the sensors and the control system, and may also affect system integrity by modifying sensor data. In contrast, protecting control algorithm code does not also ensure trust in sensor and actuator communication that ultimately bridges the cyber and physical domains.

Consider the scenario shown in Figure 2.5, where a house symbolizes a CPS. An autho-

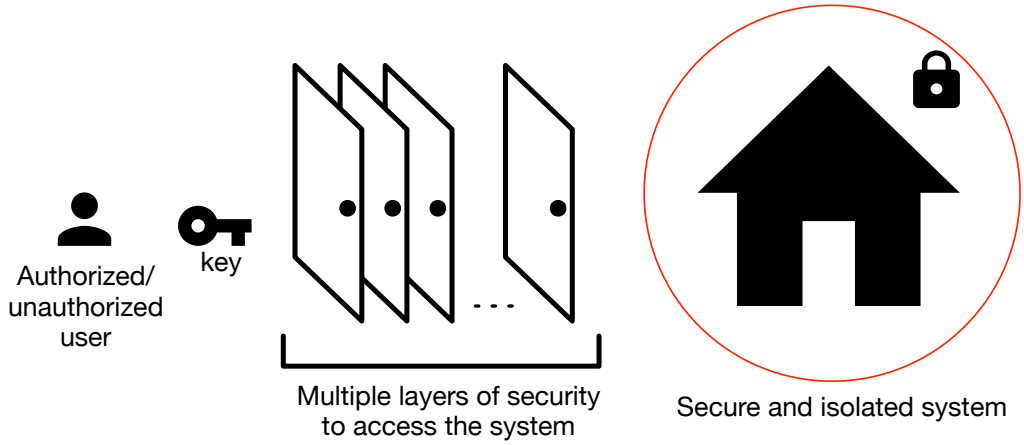


Figure 2.5: Security and trust primitives

rized/unauthorized user requires a key to gain access to the house. Research efforts looking at cryptographic methods for securing access is depicted with the multiple layers of authentication required to gain entry to the house. Cryptographic algorithms are not necessarily a complete solution since certificates may be stolen (e.g. Stuxnet) or keys may be extracted by side-channel techniques. Physical isolation of a large-scale CPS may not be possible, and even when possible malware might still be injected surreptitiously (e.g. Stuxnet) [3]. Instead of isolation, trust needs to be embedded inside the CPS and maintained through analysis of the communication within the CPS. Data packet inspection, supervisory monitoring schemes and behavior policies may have acceptable cyber costs compared to the economic and human costs of the physical system's degradation or destruction.

Trusted architecture and security primitives for the ARM platform are described in Section 2.6.1. The evolution of assurance with integration of formal methods for correctness, security and safety is explained in Section 2.6.2. Enclave-based approaches for securing sensitive information are detailed in Sections 2.6.3 and 2.6.4.

### 2.6.1 ARM TrustZone

The ARM TrustZone architecture [57, 58] is an example of promoting trust by isolation. ARM's TrustZone is derived from the creation of a trusted platform, partitioned into secure and non-secure regions. Trusted applications execute in the secure world enabling a trusted execution environment with its own kernel separate from the normal world. A security monitor switches between the secure and non-secure world with specific access policies for the peripherals and memory. The segregation of components are dependent on the security options considered during the initial design phase. The non-secure components have restricted access to the trusted peripherals in the secure world. The security monitor functions as a gatekeeper similar to a bus guard interface discussed in [48]. Unauthorized resource access attempts cause AXI bus exceptions. The implementation of a security monitor in software with no ties to hardware makes it susceptible to attacks such as MITM [59]. Figure 2.6 shows the trusted and non-trusted partitions within ARM's TrustZone framework.

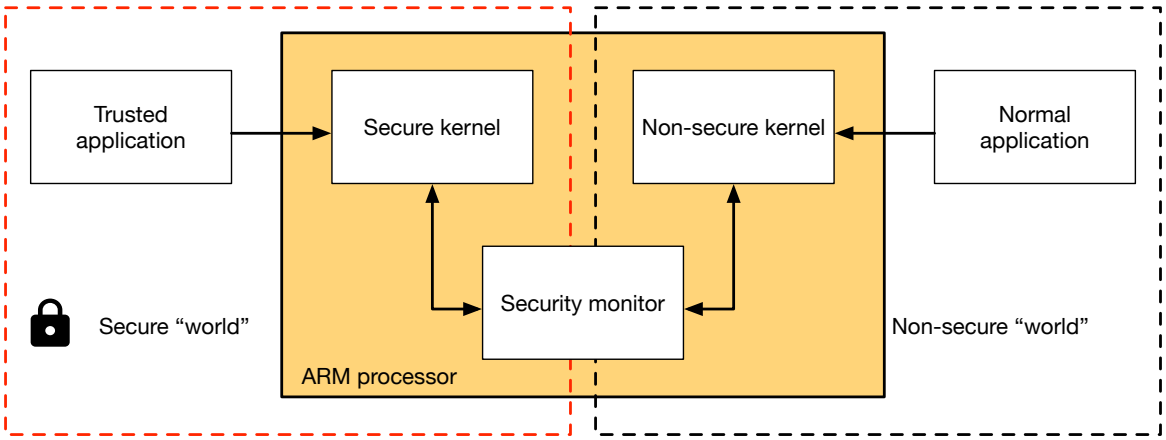


Figure 2.6: ARM TrustZone framework with trusted and non-trusted applications

ARM TrustZone has been adopted by FPGA devices such as the Zynq SoC [60] and also on commercially available smartphones with ARM processors such as A7, e.g. Apple iPhone with iOS version 9.0 and above [61]. TrustZone also supports a secure boot flow in conjunction with the security monitoring module. Since the system security is derived from

the boot process, the secure boot flow ensures that the first stage boot loader (FSBL) is verified. If the FSBL image does not authenticate, the system goes into a lock-down mode. The secure boot flow complements the TrustZone approach by ensuring that the lowest level of software can initialize the upper software stacks successfully and the secure peripherals are not tampered with during startup.

### 2.6.2 High Assurance Cyber Military Systems (HACMS)

Trust in a CPS may also follow a top-down approach. A networked CPS is ubiquitous in today's era of IoT systems. These connected systems are vulnerable to remote attacks. The Defense Advanced Research Projects Agency (DARPA) High Assurance Cyber Military Systems (HACMS) program was devised to make such systems resistant to cyber attacks [62, 63]. HACMS is built on the foundation of ensuring high assurance at each layer in the software stack. Formal verification and validation of each stack component seeks to assure functionally correct operation. A variety of development environments have been used to construct HACMS and ensure correctness, safety, and security: (i) code synthesis, (ii) domain-specific languages, and (iii) interactive theorem provers. The software components consist of hypervisors, microkernels, file systems, control algorithms, and data loggers.

Figure 2.7 shows the construction of the HACMS components with assurances embedded from top to bottom. The system requirements and the sensor inputs are translated using Architecture Analysis and Design Language (AADL) to generate glue code for binding the components. The control algorithms are synthesized using embedded domain-specific languages such as Ivory and Tower [64]. The OS kernels are formally verified to ensure correctness of the kernel binary and are synthesized from a combination of components such as functional specifications, hardware descriptions, resource constraints, environmental descriptions, safety and security policies, and verified libraries. The *seL4* microkernel [65] is an example with mathematically proven strong security properties including integrity and confidentiality. A high-assurance Real Time Operating System (RTOS) example deployed

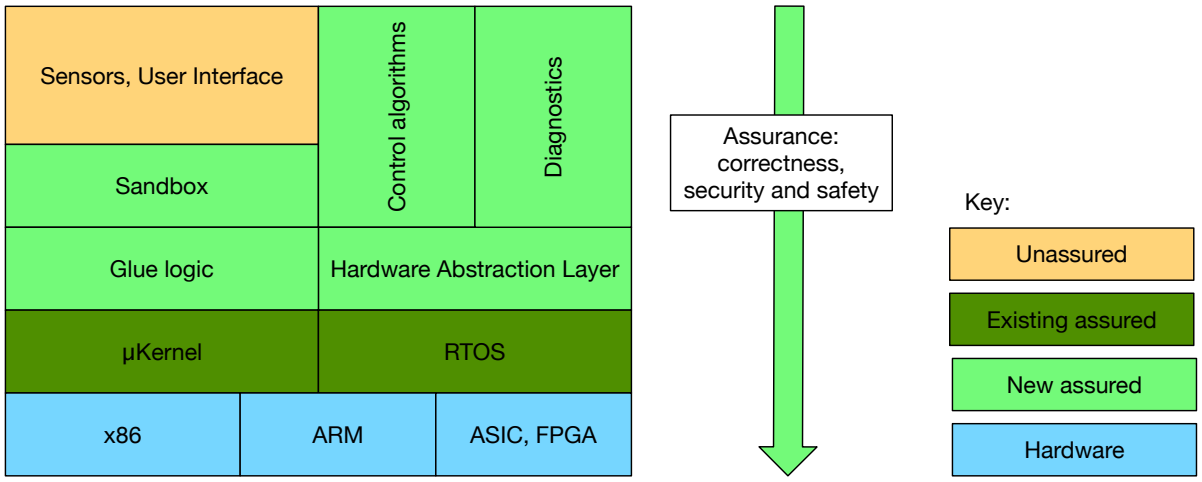


Figure 2.7: HACMS components

in conjunction with *seL4* is the *eChronos* RTOS. The lowest stack layer is the underlying hardware, which can be an x86 processor, ARM processor, ASIC or FPGA. Applying formal methods is difficult while integrating Commercial Off-The-Shelf (COTS) components. Quantifying trust through assurance and validating it for components in the complete stack is a complex task. Moving the RoT from hardware into software and firmware is an ideal concept [66], however a firmware-based RoT is vulnerable to MITM attacks as shown in [9]. Distributing the RoT between the software and the processing hardware such as an FPGA can verify the functioning of the firmware with sensitive information firewalled from the code execution path and contained inside the device's *secure enclave*.

### 2.6.3 Intel Software Guard Extensions

Intel Software Guard Extensions (SGX) is an architecture extension based on an *inverse sandbox* mechanism for embedding security in a device [67]. Instead of identifying and isolating malware in a system, the legitimate information is sealed inside an enclave. The secure enclave ensures that the sensitive information is secure from malware trying to access it irrespective of its privilege level. Figure 2.8 shows the SGX environment, where the enclave

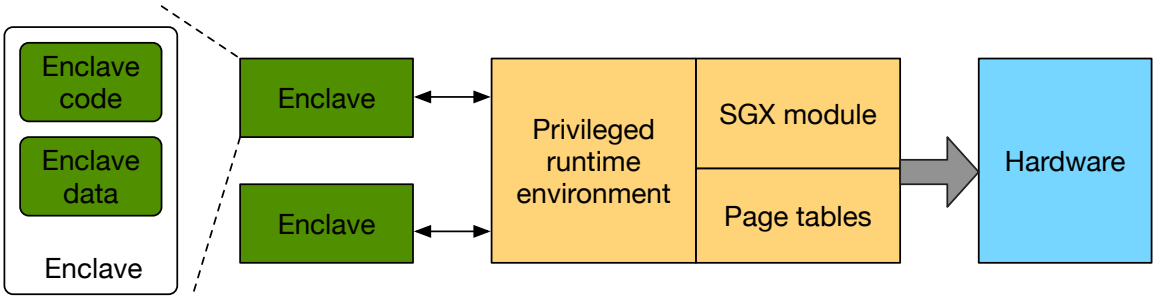


Figure 2.8: Intel SGX environment

stores the sensitive data and code.

SGX is built on secure access protocols and segregates the code execution paths of trusted and untrusted applications [68]. Trusted applications preserve the integrity of the sensitive information by identifying the secure and trusted regions of the code, and preventing it from being accessed in the event of an attack. SGX is an extension of the Trusted Platform Module (TPM) idea, where application execution is attested by the platform. SGX provides privileges to the platform to authenticate the application's trusted code and this assessment can be compared with the sensitive information in the enclave to verify the correct initialization of the application. Intel SGX is still nascent in terms of identifying the resources that can be shared between the trusted and untrusted applications.

### 2.6.4 Apple iOS Secure Enclave

The secure enclave is a countermeasure built into the hardware RoT in Apple's iOS [61]. Apple's custom processors such as the A7 and A9 are designed using the ARM architecture and leverage the strong security primitives present in ARM TrustZone. Secure boot is a part of ARM TrustZone technology [57] and ensures that components signed by Apple cryptographic keys are initialized during startup. The chain of trust includes the bootloader, kernel, and the baseband firmware. Once the chain of trust is verified, the iOS kernel is loaded and begins normal operation. The startup is stopped in case the secure boot process fails

and the iOS device switches to recovery mode.

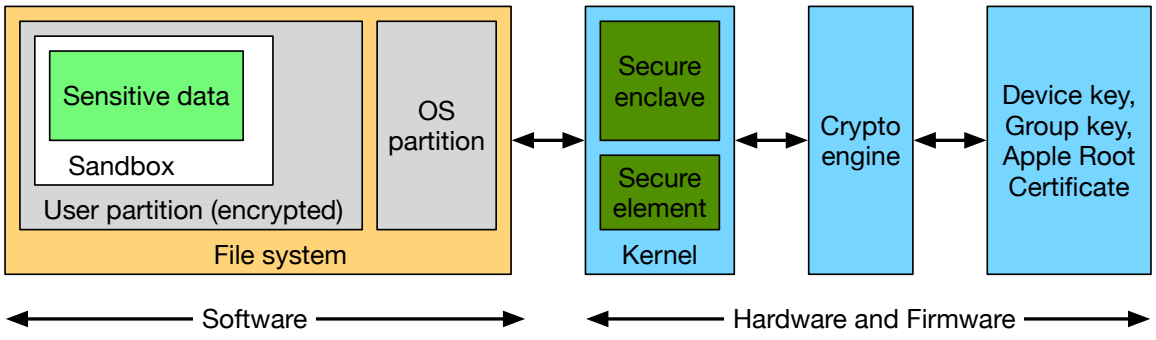


Figure 2.9: Apple iOS security partition

Figure 2.9 shows the segregation of software and hardware processes within Apple's iOS. The hardware process indicates the presence of a secure enclave built into the processor. Apple's secure enclave is a co-processor fabricated inside the Apple A7 and later A-series processors. It is hardened from the application processor's execution path and utilizes its own software update process. The secure enclave maintains the integrity of the sensitive data stored within and also leverages the secure boot flow process from ARM TrustZone. Communication between the secure enclave and the application processor is restricted through unidirectional data buffers and interrupt driven controls. The application processor can forward data to the secure enclave but cannot read it. To prevent a MITM attack or snooping of keys, the secure enclave encrypts its portion of the memory space with a key generated from its unique ID. The unique ID is generated during fabrication and resides within the secure enclave. The sensitive data stored in the secure enclave consists of a user's fingerprint data which authenticates and approves purchases from the Apple store. The secure enclave provides an effective approach to deter cyber attacks that may damage the device or steal valuable information. Since the secure enclave also runs a secure microkernel (a version of *seL4* [65]), a backdoor within the secure enclave would be the best option for providing access to critical information as required by the U.S. Government [69].

## 2.7 Xilinx Zynq Platform

ARM core processors have become the ubiquitous solution in the embedded space exemplified by smart phones and tablets. FPGA vendors such as Xilinx, Altera and Actel provide SoCs with ARM processors. These SoCs incorporate a different strategy: the ARM processors are implemented along with the FPGA device part on the same die package. This allows the designer to access the programmable software (PS) and programmable logic (PL) sections separately. A variety of single- and dual-core ARM processors clocked at different frequencies are available on these embedded FPGA devices.

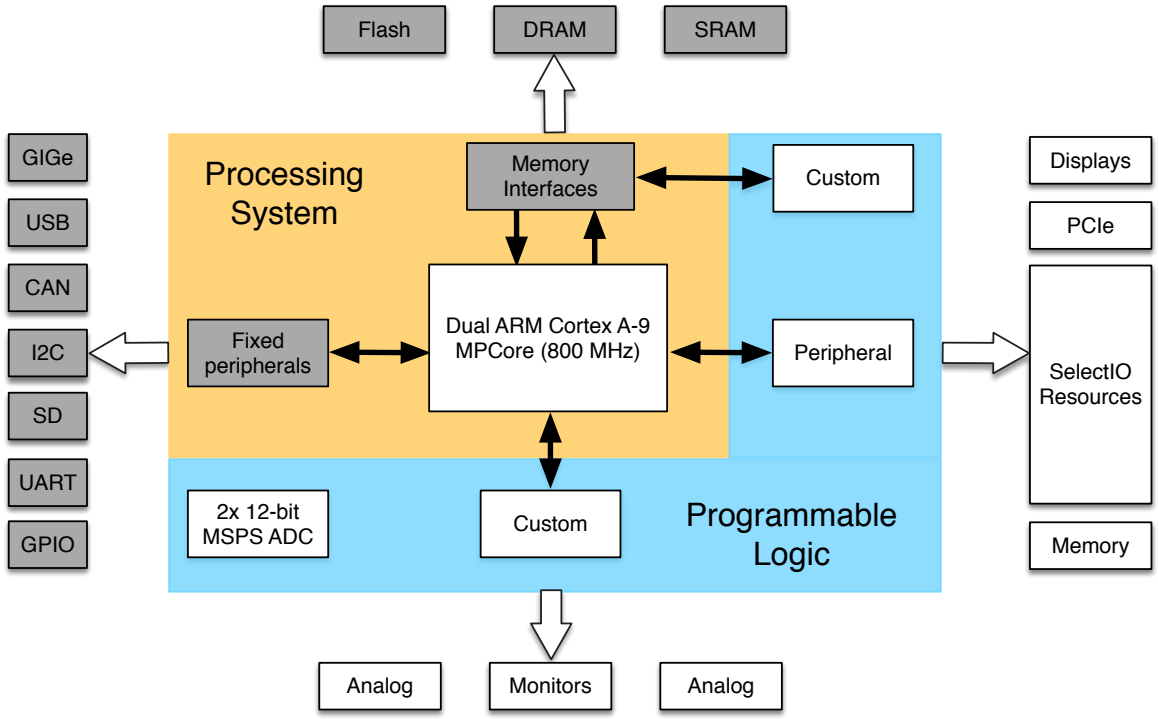


Figure 2.10: Internal architecture of the Xilinx Zynq platform

The Xilinx Zynq platform [70] as shown in Figure 2.10 consists of a dual-core ARM processor and a Xilinx Artix or Kintex FPGA implemented on the same die. The data transfer between the PS and PL section is through the high speed Advanced eXtensible Interface (AXI) bus. The Zynq platform provides a variety of I/O interfaces such as USB,

PCIe, CAN, SPI, UART, GbE, etc. Two 12-bit ADCs are also included on the device and the target applications include industrial motor control, real-time image and video processing, etc.

The MicroZed Xilinx Z7020 development board is targeted as the implementation platform for a cryptographic testbed. The various HTTs are implemented on this testbed. The advantage of the hard silicon ARM core is explored in FIDES, where the ARM core is used to implement the supervisory Trusted Anchor module.

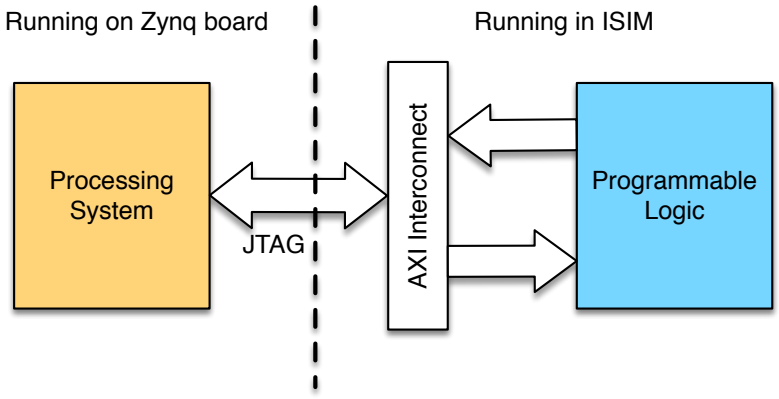


Figure 2.11: Hardware In Loop (HIL) setup showing system partitioning on Zynq board

Figure 2.11 shows the Hardware-in-Loop (HIL) setup for testing both the PS and PL sections of the Zynq platform [71]. The HIL test relies on the AXI bus to exchange data between the PS and PL sections. In the HIL test setup, the Zynq board consisting of the PS section generates signals and test vectors which is fed to the PL section. The PL section consisting of the hardware blocks is then simulated using the Xilinx ISim tool. This step validates the performance of the hardware blocks in real-time and helps to troubleshoot the hardware implementation on the FPGA fabric (PL section).

# Chapter 3

# Cryptographic Testbed for Evaluation of Hardware Trojans

## 3.1 Introduction

Cryptography algorithms are ranked by their speed in encrypting/decrypting data and their robustness to withstand attacks. Real-time processing of data encryption/decryption is essential in network based applications to keep pace with the input data inhalation rate. The encryption/decryption steps are computationally intensive and exhibit high degree of parallelism. FPGAs and GPUs are being employed as cryptographic co-processors to target different cryptography algorithms. We target different encryption; TEA and XTEA on GPU and FPGA platforms. We investigate the performance of the algorithms in terms of latency, throughput, gate equivalence, cost and ease of mapping on both platforms. We employ optimization techniques to realize high throughput in our custom configured implementations for coarse-grained parallel architectures. We propose a tool called Cryptographic Hardware Acceleration and Analysis Tool (CHAAT) that selects an optimal algorithm depending on the user's constraints with respect to hardware utilization, cost and security.

The emerging era of ubiquitous computing (pervasive computing) will be characterized by a large number of smart products (e.g., RFID, smart meters, smart thermostats) that can communicate unobtrusively and are always available, thus enabling users to access the information and services anywhere and anytime. MIT Media "*things that think*" [72] and DARPA's "*smart space*" [73] projects are few examples that will help to realize the dream of seamless integration of digital infrastructure into everyday lives.

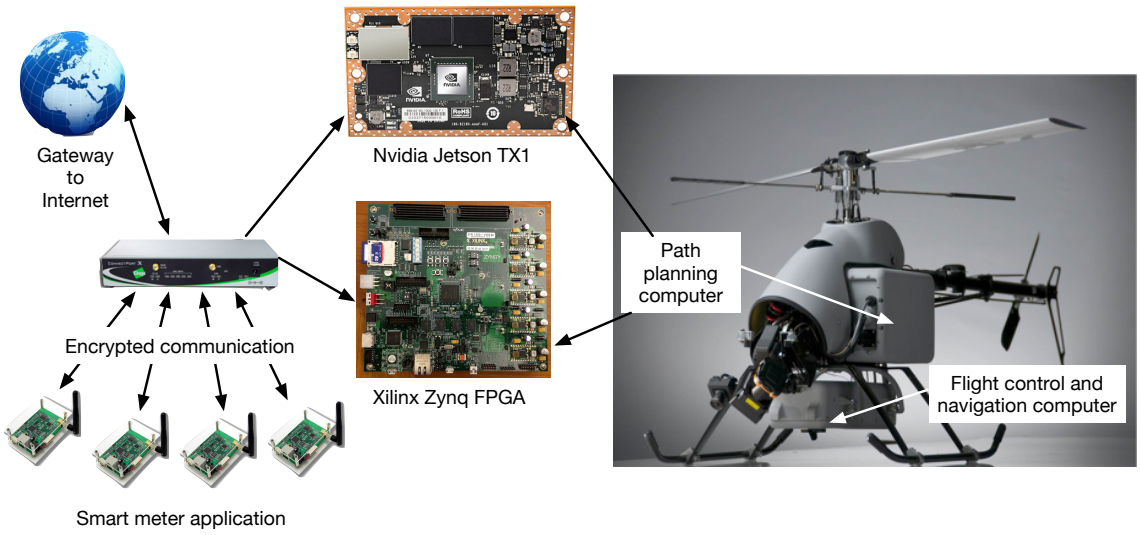


Figure 3.1: Smart meter and UAV platforms

Two real-time applications that require the efficient implementation of lightweight cryptography algorithms are considered in this chapter as shown in Figure 3.1: (1) Unmanned Autonomous Vehicle (UAV) and (2) smart grids. UAVs continuously exchange dynamic information regarding the urban environment with a gateway. The gateway also provides feedback regarding the optimization parameters that need to be fed into the UAV's path planning algorithm for mapping different routes to reach it's destination safely. In smart grids, the network consists of electric meters for reporting the usage or power outage in real-time to the power utility company. Sensitive information such as power consumption, price update, or outage awareness is exchanged between the meters and the power utility company in real-time over the Internet. Cyber attacks on such critical and dynamic information can

lead to severe losses of resources and finance. Cryptographic algorithms can be exploited as a first line of defense to provide sufficient protection to these devices from majority of the cyber attacks. Nevertheless, due to the limited power and memory constraints, the traditional cryptographic algorithms may not be suitable. Among the handful of lightweight cryptographic algorithms [74], due to the faster execution time and smaller code size, this research effort considers TEA and XTEA algorithms for resource constrained devices.

**Problem statement**: This chapter assumes an urban environment that consists of millions of wireless sensor devices (nodes) that exchange sensitive or relevant information in real-time with the gateway. The gateway is further connected to the Internet via wired or satellite links to provide services to the downstream sensor devices. As mentioned earlier, devices have tight resource constraints and are assumed to employ lightweight cryptographic algorithms such as TEA and XTEA for minimal power and area consumption. It is assumed that gateways have no power or memory constraints. One could easily observe that all the information from/to these millions of these devices need to be decrypted/encrypted at the gateway, which in turn can lead to very large response times. A larger response time in turn implies poorer performance in terms of both throughput and latency. Motivated by this fact, in contrast to the existing work, the use of FPGAs and GPUs in gateways is proposed to speed up the TEA/XTEA encryption and decryption of bulk information for improved throughput and latency.

The main objective of this work is to implement lightweight cryptographic algorithms in FPGA and GPU and thus realize a high level of security and performance, without sacrificing cost. The implementation results indicate the following:

(a) Nvidia's Tesla C2070 and GeForce GT 650M GPUs outperform the CPU implementation for data sizes varying from 8KB to 8 GB. Specifically, Nvidia Tesla C2070 and Nvidia GeForce GT 650M GPU can achieve a maximum throughput of $\approx$ 13 Gbps when compared to 500 Mbps achieved from the CPU based implementation.

(b) FPGA implementation of TEA and XTEA provided a throughput of 5 Gbps for smaller

plaintext sizes yielding a speedup of 6-9x over the GPU.

The problem of characterizing the performance of TEA and XTEA implementation using both FPGA and GPU is rarely discussed and is hence considered in this chapter.

**Roadmap**: The rest of the chapter is organized as follows: Section 3.2 provides related work and a detailed description of the cryptographic algorithms employed in this chapter. In Section 3.3, the GPU and FPGA implementation of TEA and XTEA is elaborated. Section 3.5 discusses the performance of TEA and XTEA on CPU, GPU and FPGA platforms. Finally, Section 3.6 concludes this chapter.

## 3.2 Lightweight Cryptographic Algorithms

In this section, an overview of TEA (Tiny Encryption Algorithm) and XTEA (Extended Tiny Encryption Algorithm) is explained. Several light-weight algorithms such as TEA, XTEA, HIGHT and PRESENT have been mapped on FPGAs and ASIC [75, 76, 77]. However, these implementations have been custom configured for specific ASIC technology and older FPGA devices. GPUs have been used previously as cryptographic co-processors for targeting Advanced Encryption Standard (AES) [78] encryption algorithms. Although several cryptographic algorithms have been proposed for low cost pervasive devices, this chapter utilizes TEA and XTEA due to the following reasons:

(a) very simple to implement as compared to HIGHT and PRESENT;

(b) faster execution time as compared to AES;

(c) minimal storage space as compared to DES.

Besides, TEA as well as XTEA employs simple addition, shifts and XOR, and therefore has a very small size. Hence, these algorithms are an ideal solution to provide information security services for networks with resource-constrained devices.

### 3.2.1 Tiny Encryption Algorithm

Tiny Encryption Algorithm (TEA) was introduced by David Wheeler and Roger Needham [79]. TEA uses addition, XOR (denoted by $\oplus$) and shift operations on 32-bit words and has a very small code footprint. Moreover, TEA does not require a lot of memory as it does not use a substitution box (S-box) as in Advanced Encryption Standard (AES). The memory requirements of TEA is one fourth the requirements of AES as per the implementation of both algorithms on sensor nodes. Although TEA qualifies as a software encryption scheme, it's simple arithmetic operations are suited for embedded hardware implementations. Despite the several advantages, TEA has security holes and weaknesses for smaller rounds, especially the *Avalanche Effect* seen for 6 rounds [80]. Therefore, it is recommended to have 32 rounds for TEA to be considered as a secure encryption algorithm. This work also assumes 32 rounds for TEA.

Given $N$ = number of rounds, delta = 0x9E3779B9, $v_1$ = most significant 32-bits of data block, $v_0$ = least significant 32-bits of data block, $key[0], key[1], key[2], key[3]$ are 32-bit chunks of the 128-bit key,

**Algorithm 1:** Pseudo-code for TEA

**for** i = 0 to N **do**

$sum+ = delta$

$v_0+ = ((v_1 << 4) + k_0) \oplus (v_1 + sum) \oplus ((v_1 >> 5) + k_1);$

$v_1+ = ((v_0 << 4) + k_2) \oplus (v_0 + sum) \oplus ((v_0 >> 5) + k_3);$

**end for**

### 3.2.2 Extended Version of TEA

The Extended Tiny Encryption Algorithm (XTEA) was introduced after weaknesses for smaller rounds were found in TEA [80]. In XTEA, the key scheduling is modified to reflect different patterns for mixing the data and key continuously per round. Both TEA and XTEA

are Feistel-type ciphers, where each input block is split into two halves and mixed with the data using XOR as a reversible function. Both TEA and XTEA employ a cryptographic key of 128 bits split into chunks of 32 bits for encrypting or decrypting the input 64-bit data in blocks of 32 bits.

**Algorithm 2:** Pseudo-code for XTEA

**for** i = 0 to N **do**

$v_0 += ((v_1 << 4) \oplus (v_1 >> 5) + v_1) \oplus (sum + key[sum\&3])$

$sum += delta$

$v_1 += ((v_0 << 4) \oplus (v_0 >> 5) + v_0) \oplus (sum + key[sum\&3])$

**end for**

## 3.3 Cryptographic Co-processors

Both TEA and XTEA are benchmarked on CPUs based on the work done in [79, 80]. In this section, the implementation of TEA and XTEA on GPU and FPGA platforms is discussed.

### 3.3.1 GPU Implementation

GPUs are inexpensive, COTS devices used to accelerate a large number of applications such as astrophysics simulations [81, 82], biomedical applications [83], sensor perception and path planning for UAVs [84, 85], computational electromagnetics, etc. [86, 87]. The NVIDIA Kepler series GPUs are more FLOPS/Watt efficient and are being used to drive real-time image processing capabilities in emb,edded form factors such as the Nvidia Jetson development board [88].

GPUs consist of $M$ streaming multiprocessors with $N$ cores each. All the streaming cores have common access to 64 KB of shared memory within a MP. Each multiprocessor has one set of 32-bit registers per processor, constant memory and texture caches. Each

streaming core can execute the same instruc,tion on different data making it similar to a Single Instruction Multiple Data (SIMD) processor. The multiprocessors communicate with the CPU through the GPU device memory using the PCI Express interface. GPUs have high memory bandwidth to the on-board device memory, which ranges from 256 MB to 6 GB in size. The GPU is referred to as a co-processor or device and the CPU is referred to as the host. Therefore, the application is partially executed on the host and the device. The host program copies the data to the device memory and the device program launches computational kernels which run on the multiple streaming cores. The GPU is programmed using Nvidia's C-based Compute Unified Device Architecture (CUDA) environment. The CUDA compiler assumes that the host and the device have separate accesses to their memory, also referred to as host memory and device memory.

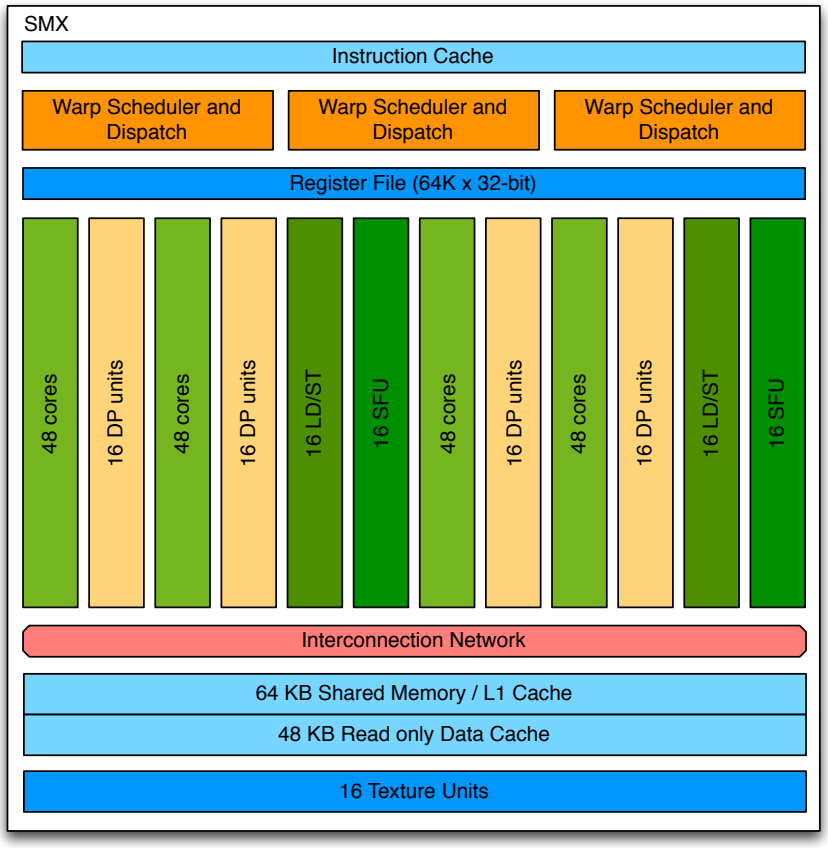


Figure 3.2: Streaming processor architecture within NVIDIA's Kepler GPU

NVIDIA's Kepler series GPU [89] consists of a maximum of 15 Streaming Execution (SMX) units and up to six 64-bit memory controllers. Each SMX unit has 192 single-precision CUDA cores as shown in Figure 3.2 and each core comprises of fully pipelined floating-point and integer arithmetic logic units. Since double precision accuracy is desired in many applications, 64 double precision (DP) units are included in each SMX. Fast approximate transcendental operations are provided by 32 special function units (SFU). The single instruction multiple datapath (SIMD) execution model is one of the main differences between CPU and GPU. However, the SIMD model imposes constraints on the GPU when executing threads with task-level parallelism. Each SMX can execute only one task-level thread on a single core, while the remaining cores remain idle. The idling of a dominant number of cores in a SMX when executing task-level threads leads to poor utilization of the GPU's processing capability. The execution of divergent flow threads reduces the instruction throughput, as the multiple flow control threads are executed sequentially. Therefore, not all algorithms can run as it is on GPUs and still obtain speedup. In most cases, existing algorithms need to be re-written and optimized to increase core utilization with minimum divergent threads.

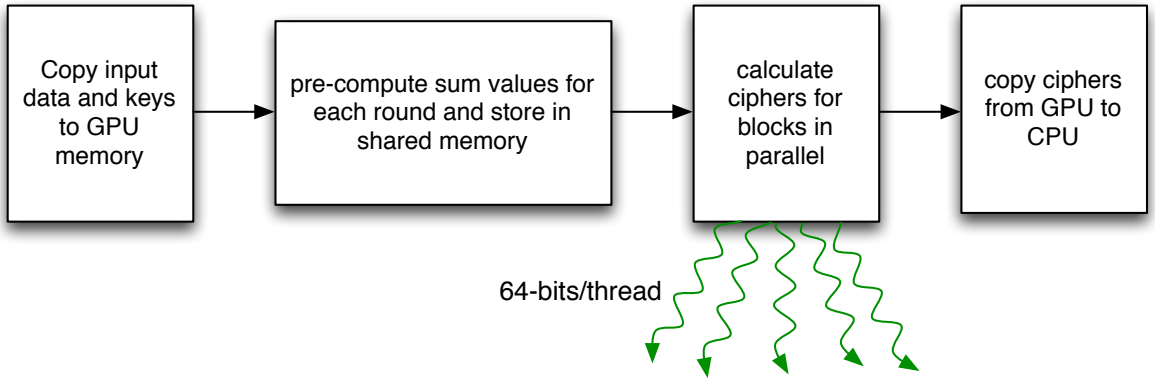


Figure 3.3: GPU implementation flowchart of TEA and XTEA

The first step in the GPU implementation of TEA and XTEA consists of copying the block of messages and the key from the CPU to GPU global memory. The initial delta value *(0x9E3779B9)* is accumulated during each round and mixed with the input data. This

accumulated value is pre-computed and cached in the GPU's shared memory. Each thread operates on the 64-bit chunk of the input data block. Several threads are launched in parallel to process the block of messages. The selection of block size and the number of threads is an optimization parameter that compensates for the tradeoff in copying the data to the GPU. To keep all the GPU cores completely utilized, more threads are launched thereby hiding the latency. Once the data is encrypted or decrypted, it is copied from the GPU global memory to the CPU. Figure 3.3 presents the flowchart for the GPU implementation.

### 3.3.2 FPGA implementation

FPGAs are high density gate arrays of programmable logic and interconnects that can be used to implement custom logic functions with tradeoff on area or speed. They are commonly used in embedded systems to rapidly prototype algorithms and verify the complete system solution before fabricating the final ASIC.

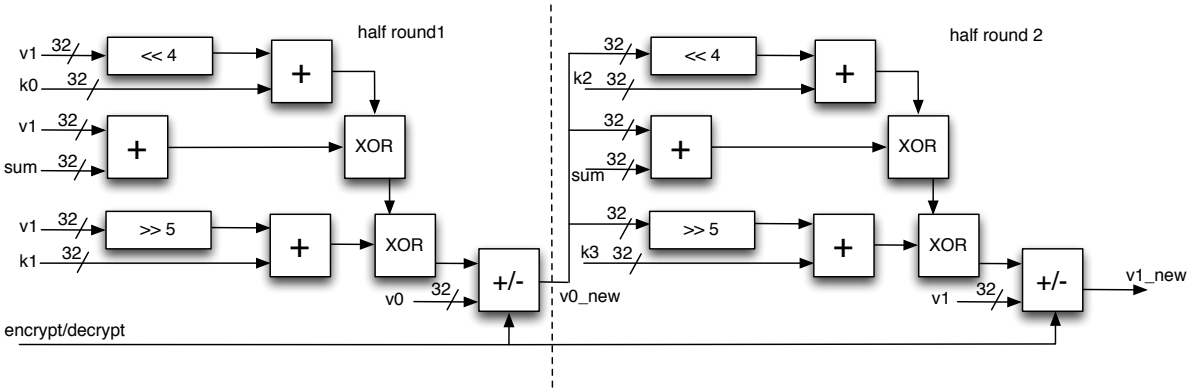


Figure 3.4: Hardware block diagram of single round in TEA

Figure 3.4 shows the block diagram of a single round in TEA. Depending on the encryption or decryption mode, the values of $v_0$ and $v_1$ are added or subtracted, respectively. The value of sum is precomputed from delta for each round and provided as input to each module. Consequently, this design choice saves logic resources required for computing the sum for every round. The complete implementation consists of 32 cascaded modules with the values

of $v_0\_new$ and $v_1\_new$ serving as input to the next module.

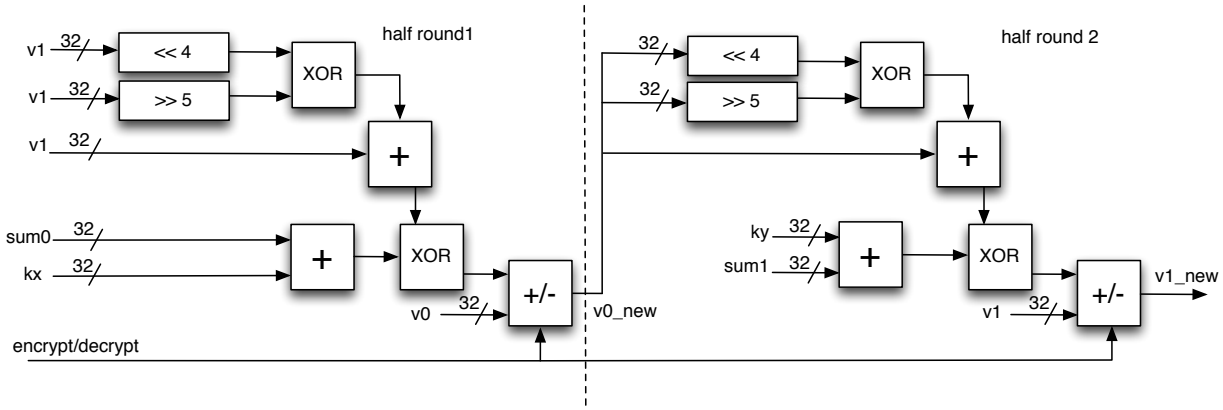


Figure 3.5: Hardware block diagram of single round in XTEA

The hardware implementation of XTEA is different from TEA, as it utilizes different cipher key combinations for mixing the half rounds. Figure 3.5 shows the hardware block diagram of single round in XTEA. Two key values are provided as input to each round and they correspond to a specific sequence, e.g., round 1 gets least significant 32-bit chunk and the most significant 32-bit chunk of the 128-bit key. Similar to the hardware implementation of TEA, the values of $v_0$ and $v_1$ are added or subtracted depending on the mode. XTEA utilizes two different values of delta and these values are provided as inputs $sum_0$ and $sum_1$ to each round. Each round utilizes a different sequence of keys for adding with the $sum_0$ and $sum_1$ inputs. This sequence is precomputed and the corresponding key value is provided as $k_x$ and $k_y$ to each round. The values of $v_0\_new$ and $v_1\_new$ are passed to the input of the next cascaded module.

Both the FPGA implementations are designed so that the datapath can provide better pipelining and guaranteed throughput for streaming input data blocks. The design was simulated using Xilinx ISE and synthesized to provide resource utilization and achievable clock frequency.

## 3.4 Cryptographic Hardware Acceleration and Analysis Tool (CHAAT)

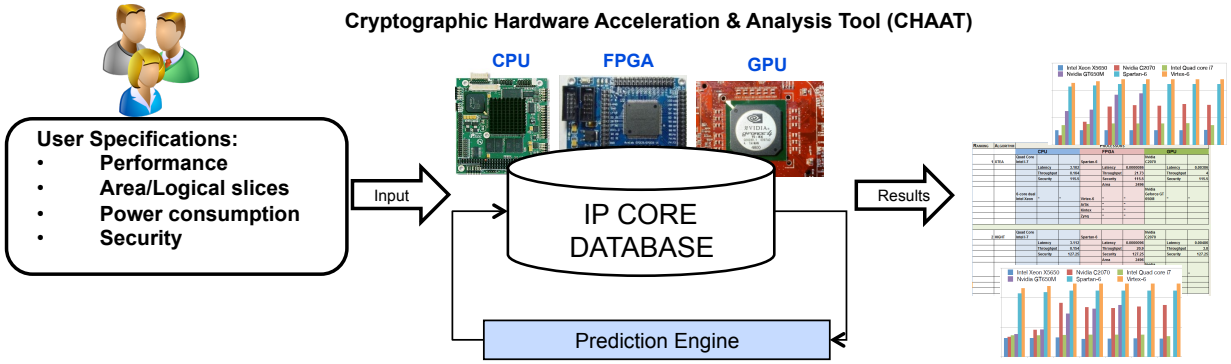


Figure 3.6: Cryptographic Hardware Acceleration and Analysis Tool

In this section, we propose a tool called Cryptographic Hardware Acceleration and Analysis Tool (CHAAT) for hardware benchmarking of cryptographic suites. The main aim of the tool is to facilitate a designer/user to select an optimal cryptography algorithm for the platform of their choice in an automated fashion. CHAAT tool comprises an IP core database of cryptographic suites implemented in different platforms like CPU, GPU, FPGA; for each platform, the tool essentially captures the following features: performance (throughput, latency, clock cycles, and power consumption), cost (memory requirements) and security of the cryptographic suites. Among these features, the most difficult one to compute is the security of the algorithm. Cryptographic algorithms provide different strengths of security, depending on the algorithm and the key size used. We measure an algorithm's security level based on the best known attack on the algorithm. For instance, the best known attack against TEA is related-key attack which requires $2^{23}$ chosen plaintexts under a related-key pair, with $2^{32}$ time complexity. The security of TEA will be considered as low as $2^{32}$ falls below the 80 bits of security recommended by NIST. Besides the IP core database, CHAAT also comprises of a prediction engine that will predict the performance and cost of suites that are not present in the database.

The working of CHAAT is as follows: We will denote the three main features, performance, cost and security, of a given cryptographic suite $i$ in platform $j$ as $P_{ij}, C_{ij}$ and $S_{ij}$, respectively. As each feature is measured on different scale, we use "normalization" to adjust the range of each feature to a value between 0 and 1. For instance, throughput and the memory requirements have different ranges, one measured in bits/sec and the other in bytes. To achieve the normalization, CHAAT uses the designer specifications.

Let the designer specifications be $P_u, C_u$ and $S_u$. Then, the final ranking of each algorithm is computed as follows:

$$W_p \frac{P_{ij}}{P_u} + W_c \frac{C_{ij}}{C_u} + W_s \frac{S_{ij}}{S_u} \tag{3.1}$$

where $W_p, W_c, W_s$ denotes the weight for performance, cost and security. For simplicity, we consider $W_p = W_c = W_s = 1$. CHAAT applies the ranking formulae for each algorithm and outputs the topmost three highly ranked algorithms.

## 3.5 Results

The main objective of this chapter is to study the performance of TEA and XTEA using reconfigurable and multi-core platforms. We use the following platforms:

1. Nvidia's Tesla C2070 high-end GPU clocked at 1.15 GHz with 6 GB of GDDR5 device memory. The Tesla C2070 is based on Fermi architecture and consists of 14 streaming multi-processors with 32 cores each (total of 448 cores). The host system consists of 2 hexa-core Intel Xeon processors clocked at 2.6 GHz running Red Hat Linux OS.

2. Nvidia's GeForce GT 650M notebook GPU consisting of 384 cores, 2 streaming multi-processors with 192 cores each. The GeoForce GT 650M GPU is clocked at 900 MHz and is based on Nvidia's Kepler architecture. The host notebook consists of a quad-core Intel Core i7 clocked at 2.3 GHz running Mac OSX.

3. Xilinx's Zynq-7000 SoC ZC702 evaluation board [90] consisting of the XC7Z020 device. The Zynq-7000 platform consists of a dual ARM Cortex-A9 processor clocked at 800 MHz and Artix-7 FPGA as the programmable logic.

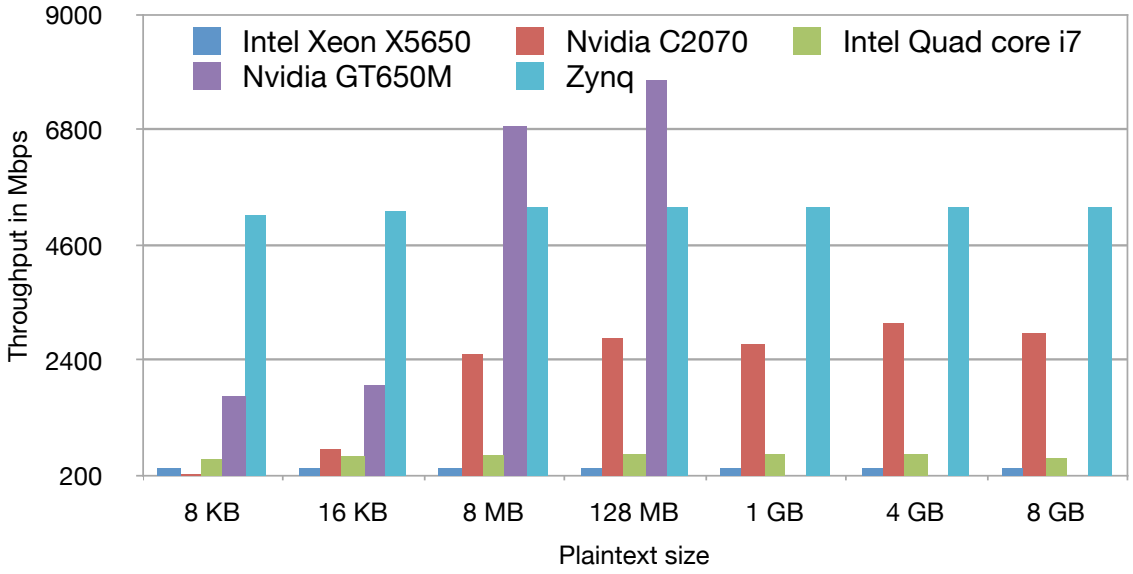


Figure 3.7: Throughput (Mbps) comparison of TEA on CPU, GPU and FPGA

Figures 3.7 and 3.8 show the throughput comparison in Mbps for the CPU, GPU and FPGA platforms. The GPU implementation is faster than the CPU implementation in all cases. The best throughput results are obtained from the Kepler architecture based Nvidia GT650M. Both GPUs are able to achieve a peak performance of 13x the throughput of the CPU based implementation. Comparing the number of cores, we expect the Tesla C2070 to provide better performance as compared to the GT650M. However, the GT650M provides better throughput results than the Tesla C2070. This is mainly due to the more number of registers and better core scheduling in the Kepler architecture of the GT650M. At the same time, the GT650M cannot execute for plaintext sizes more than 128 MB (TEA) and 1 GB (XTEA) due to it's device memory limitation of 1 GB. The results indicate that Nvidia's GT650M is more suitable for embedded cryptography applications, whereas the high-end

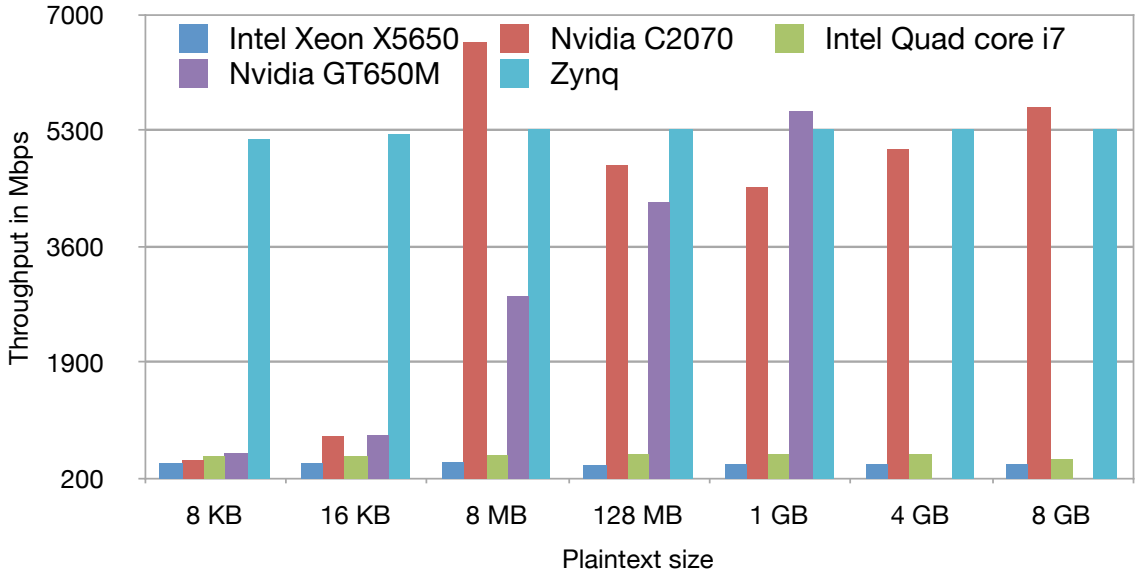


Figure 3.8: Throughput (Mbps) comparison of XTEA on CPU, GPU and FPGA

Tesla C2070 is more suited for high volume cryptography applications. GPUs provide advantage over CPUs in computing cryptographic primitives, as they are able to achieve better throughput especially because of the coarse grain parallelism in some cryptographic algorithms. It is observed that the TEA implementation is significantly better than the similar XTEA implementation on the GPU. GPUs show better speedup as the size of the plaintext increases, which indicates that more number of threads executing on larger data size increases the computational efficiency hiding the latency incurred in copying data.

| **Algorithm** | **# of logic slices** | **Utilization %age** |
|---|---|---|
| TEA | 4800 | 9% |
| XTEA | 4000 | 7% |

Table 3.1: Resource utilization of TEA and XTEA on FPGA

The FPGA design was implemented in VHDL using Xilinx ISE toolset. Table 3.1 provides the logic slice utilization obtained from place and route report. TEA utilizes 9% and XTEA

utilizes 7% of the total FPGA resources. This is useful, because more programmable logic resources can be utilized towards implementing other compute intensive sections of the original application. Both TEA and XTEA exhibits a latency of 128 clock cycles for encrypting or decrypting a single 64-bit data block. The place and route process provides an achievable clock period of 3 ns. The throughput calculations are based on the following equation:

$$Throughput = \frac{plaintextsize}{latency * (\frac{plaintextsize}{8} - 1) * t_{clock}}$$

The Zynq platform provided a throughput speedup of 6-9x over the GTX650M GPU for the 8KB and 16KB plaintext blocks. This speedup is observed due to the ability of FPGAs to compute on streaming data realizing a multiple instruction datapath architecture. The GPU and FPGA provide similar throughput for larger plaintext sizes. This result is useful in deciding the hardware accelerator depending on the volume of data that is streamed into the system.

## 3.6 Conclusions

In this chapter, the performance of lightweight cryptography algorithms on CPUs, GPUs and FPGAs is compared. GPUs and FPGAs provide better throughput for both TEA and XTEA as compared to CPUs. FPGAs perform better for smaller plaintext sizes whereas GPUs are better for larger plaintext sizes. In terms of development time and cost, GPUs are better suited as embedded cryptography co-processors as compared to FPGAs. Future research efforts may address the use of Zynq platform as a complete, low-cost cryptographic co-processor for more complex cryptographic algorithms such as AES, SERPENT, HIGHT and PRESENT.

# Chapter 4

# Design, Implementation and Security Analysis of Hardware Trojan Threats

## 4.1 Introduction

Analogous to DNA mutations or cancer affecting the human body, hardware malware attacking an IC occurs at a deeper level and cannot be removed by rebooting the system or re-installing the software. Hardware Trojan Threats (HTTs) are stealthy hardware malware introduced into the IC using unauthenticated IP or tampered software tools. In this work, we initially try to understand the drawbacks of non-destructive, non-invasive post-silicon technique that relies on physical parameters such as power consumption, timing variation and area utilization to detect HTTs. As opposed to existing works, our investigation initiates with designing a system model that enhances the security of the system and then an adversary model that exposes and exploits the vulnerabilities present in the system. Using existing Trojan implementations and Trojan taxonomy as a baseline, we designed and employed seven malicious HTTs that execute various threats ranging from sensitive information leakage to denial of service attack that disrupts the hardware Root of Trust (RoT). Leveraging optimization techniques, prior to and after the HTT insertion, we were able to mask the

HTTs within the golden parameter profiles. Security analysis on the implemented Trojans showed that existing post-silicon techniques pose limitations and only a maximum of 57% of designed HTTs were detected. To enhance the detection rate, we propose a novel detectability metric called HTT detectability metric (HDM) that leverages a weighted combination of normalized physical parameters. HDM was able to detect 86% of the implemented Trojans as compared to using power, timing and utilization alone. To further enhance the confidence in detecting HTT, we also propose an architectural refinement in the form of a trusted anchor for embedding user-defined policies that prevents HTTs from activating within the system.

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 deeply affected if a DNA mutation or cancerous disease has crippled the immune system. Learning and understanding from biologically inspired systems, HTTs are *cancer-like stealthy malicious components* that infects the 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 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 [91].

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 [12]. 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 [7]. In the past few years, several detection algorithms in the arena of hardware security have been proposed [20]. Destructive techniques such as de-packaging, reverse engi-

neering and imaging of ICs are very expensive and can be applied to only a selected quantity of ICs. These techniques are commonly used for detecting counterfeit ICs and not the ones with HTTs. Non-destructive non-invasive techniques such as side-channel analysis (i.e. 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 [22] 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, loopholes can also occur while defining many security properties. Some efforts propose to use additional detection modules to monitor chip functionality (e.g., guard modules such as DEFENSE) and physical parameters (e.g., use of scan chains), commonly referred to as invasive techniques [38, 26].

### 4.1.1 Motivation and Contribution

The wide-spread focus on post-silicon techniques to test the trustworthiness of ICs 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 optimized hardware Trojans at the design (RTL) 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 and those HTTs can be designed optimally and placed into the hardware masked within its golden parameter profiles. 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 detection of HTTs 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. The contributions of this chapter is as follows:

(a) *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.* Our security analysis results in fact showed that using HDM, 86% of the implemented Trojans were detected as compared to using power consumption, timing variation and utilization alone.

(b) *To enhance trust in a given IC, we also propose an architectural refinement in the form of a trusted anchor for embedding user-defined policies that prevents HTTs from activating within the system.* Existing works also suggest the use of hardware encryption and authentication (i.e. Root of Trust) as a possible methodology to defeat activation of hardware Trojans.

(c) *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 chapter is organized as follows: In Sections 4.2 and 4.3, the adversary models, system models and the testbed designs are discussed; Section 4.4 explains the design and implementation of various HTTs; the results are described along with the proposed detectability metric in Sections 4.6 and 4.5. Section 4.7 discusses the Trusted Anchor design that aims to enhance trust in Integrated circuits. Finally Section 4.8 concludes this chapter.

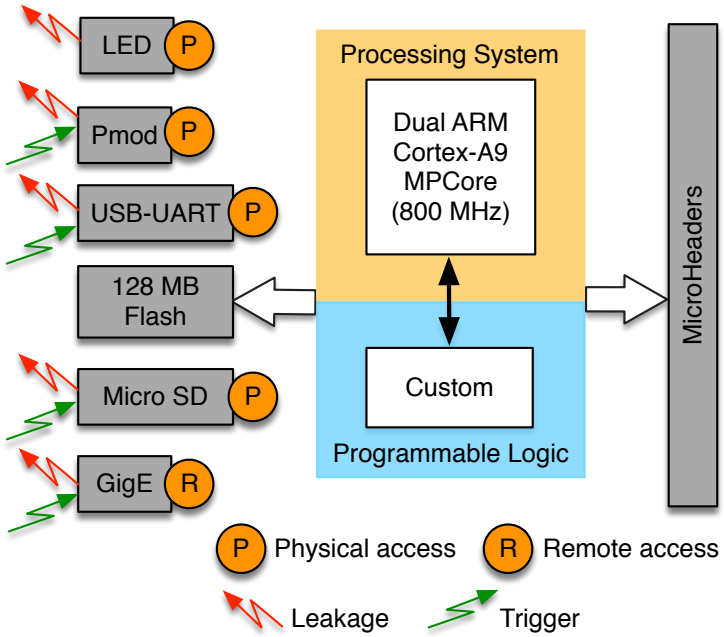


Figure 4.1: MicroZed Xilinx Zynq-based FPGA board

## 4.2 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, USB-UART, PMOD. 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. Figure 4.1 shows various components of a MicroZed Xilinx Zynq-based FPGA development board [70] with classification of each component in terms of trigger, leakage and access points. The Xilinx Zynq FPGA consists of an ARM-fused FPGA where the ARM processor is referred to as the Processing System (PS) and the FPGA fabric is referred to as the Programmable Logic (PL).

## 4.3 System Model

To demonstrate the concept of lightweight, but highly functional Trojan 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 mimic the industrial design practices. As a result, an attacker was given very limited room to leverage an attack and still be unnoticeable in terms of device utilization.

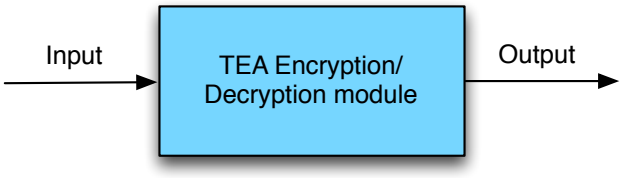


Figure 4.2: Cryptosystem testbed

### 4.3.1 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) [79]. The attacks published against TEA include key attacks and techniques arising from zero-correlation cryptanalysis, however these attacks sought to exploit the weaknesses in the algorithm rather than compromise the system integrity. The HTTs implemented on this cryptosystem testbed as shown in Figure 4.2, were designed to attack the system rather than the TEA algorithm.

### 4.3.2 Root of Trust Testbed Design

The second testbed shown in Figure 4.3 is a classic Root of Trust (RoT) design that consists of a secure memory and a key guard [33]. 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 initially sends an address of the memory location that it needs access to the 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 retrieves the requested content from memory and sends it to the authorized module in encrypted form;

(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. In short, in the RoT testbed, the key guard forces any authorized or unauthorized hardware that requests the contents from the secure memory to solve an arbitrary challenge-response pair. The challenge-response pair can be implemented using a Finite State Machine (FSM) approach or a predefined mapping of challenges to responses.

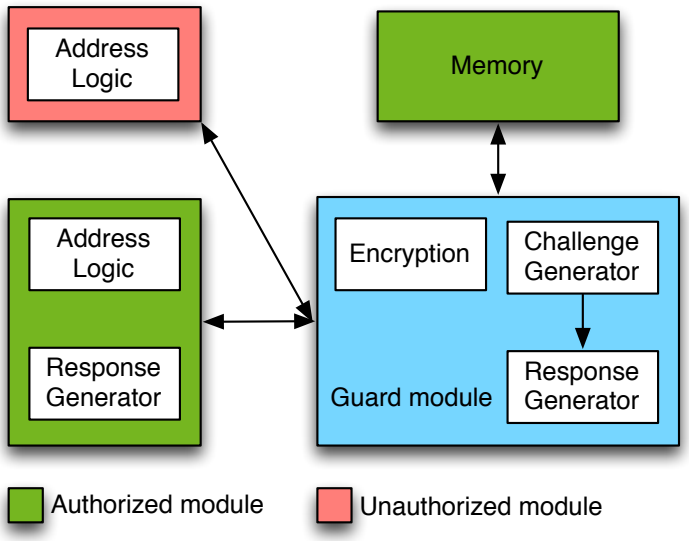


Figure 4.3: Guard system testbed

## 4.4 Hardware Trojan Threats

When considering how to design Trojans on each testbed, careful consideration was given to the target Zynq FPGA platform by breaking down all the possible input and output points as per Figure 4.1. These led to an understanding of the potential leakage points, access points and external triggers an attacker could utilize to design the Trojan. 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. 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 [16, 23, 92].

### 4.4.1 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 predefined 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.

The *Always On* HTT, shown in Figure 4.4 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

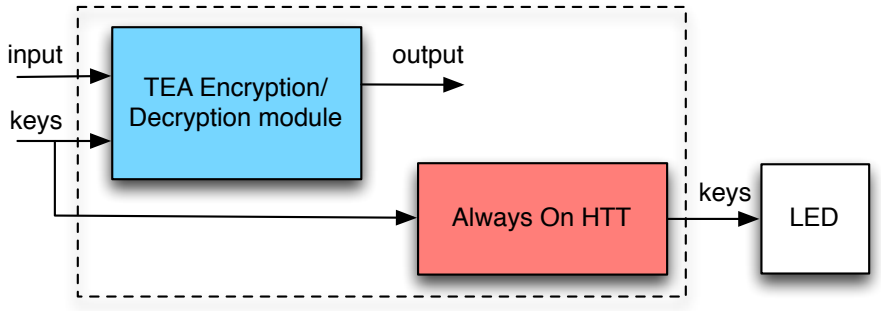


Figure 4.4: Always On Trigger

of transmission for the attacker is enabled by sending a specific value (e.g. xBE) over the LEDs four times.

### 4.4.2 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 predefined threshold. The trigger must be rare enough to ensure that the HTT is not accidentally triggered during system testing.

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 4.5 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.

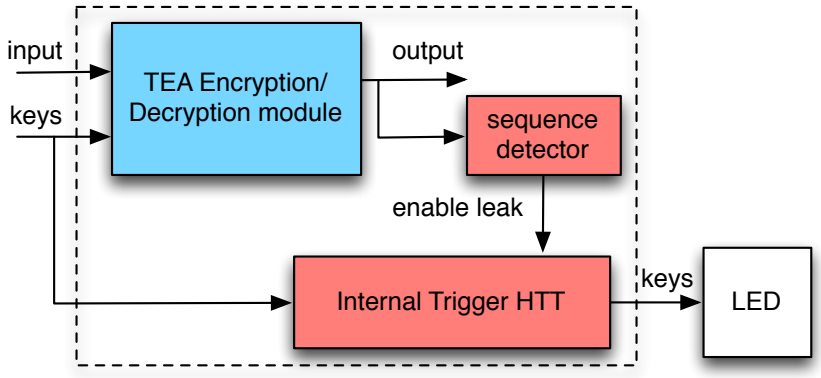


Figure 4.5: Internal Trigger

### 4.4.3 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.

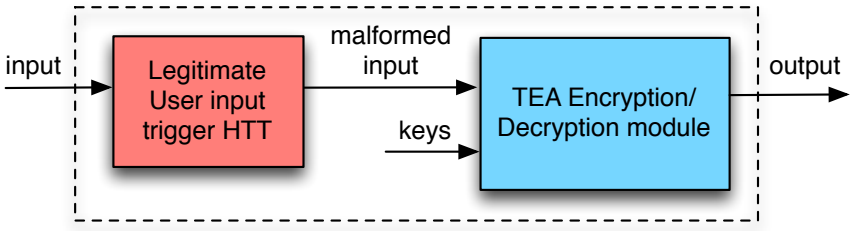


Figure 4.6: Legitimate User Trigger

The input trigger module shown in Figure 4.6 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.

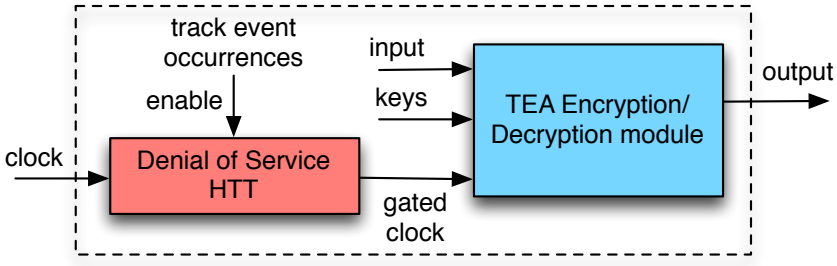


Figure 4.7: Denial of Service

### 4.4.4 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. The denial of service HTT is implemented as shown in Figure 4.7, where specific event occurrences are tracked to enable the clock gating. Once the clock is disabled, the system is rendered incapable.

### 4.4.5 Beat the Root of Trust (3 HTTs)

The memory guard system shown in Figure 4.3 is targeted for demonstrating HTTs developed to beat the Root of Trust (RoT) system. The testbed was slightly modified as shown in Figure 4.8 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.

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

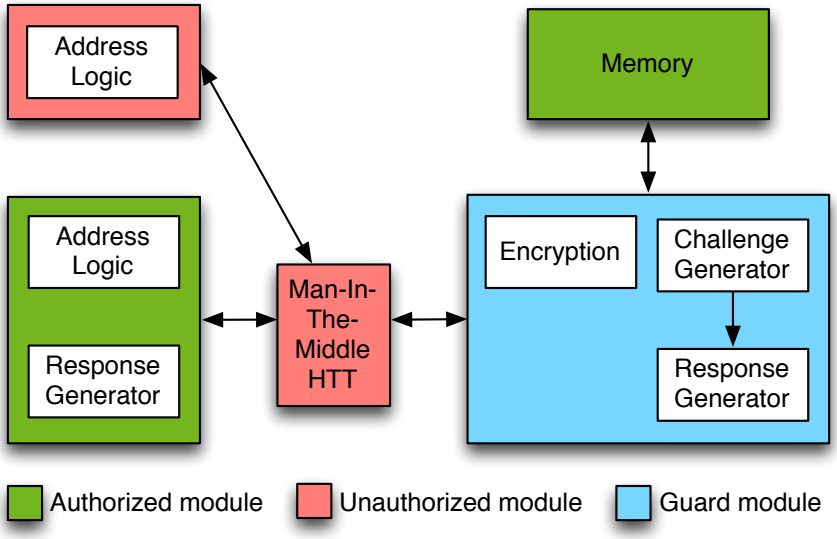


Figure 4.8: Man-In-The-Middle HTT for beating the authentication in the system

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 show the vulnerability of Root of Trust modules to hardware Trojans residing within the system.

## 4.5 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 4.6, we observed that depending on the nature of the HTT, some Trojans can be detected with power consumption, while others can be detected with timing or utilization that uses 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:

$$\text{HDM} = \sum_{i=1}^{m_p} W_i \frac{O_i}{A_i} \tag{4.1}$$

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 $\tau_i$'s are mutually exclusive, we independently compute the probability of false alarm and missed detection for each $\tau_i$.

$$O_i = A_i + \eta;\ [H_0] \tag{4.2}$$

$$= T_i + \eta;\ [H_1] \tag{4.3}$$

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.

$$P_{FA(i)} = P(\frac{A_i + \eta}{A_i} > \tau_i) = P(\eta > (\tau_i - 1)A_i) \tag{4.4}$$

$$= \frac{1}{\sigma\sqrt{2\pi}} \int_{(\tau_i - 1)A_i}^{\infty} e^{\frac{-x^2}{2\sigma^2}} dx \tag{4.5}$$

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

$$P_{MD(i)} = P(\frac{\alpha A_i + \eta}{A_i} < \tau_i) = P(\alpha < \tau_i - \frac{\eta}{A_i}) \tag{4.6}$$

$$= \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^{+\infty} \int_{0}^{\tau_i - \frac{\eta}{A_i}} \frac{e^{\frac{-y^2}{2\sigma^2}}}{u^+ - l^+} dxdy \tag{4.7}$$

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

$$\tau_i^* = \frac{d}{d\tau_i}(P_{FA(i)} + P_{MD(i)})|_{\tau_i = \tau_i^*} = 0 \tag{4.8}$$

## 4.6 Results

The testbeds shown in Figures 4.2 and 4.3 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 Vivado 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. Hinton diagrams were utilized for visualizations of the characteristic data for each HTT, with green denoting a positive change and red denoting a negative change from the trusted reference model. 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 [93]. With these tolerances, any HTT falling outside the range for a given characteristic was considered detectable, and depicted as green in the Hinton diagram. All the data used in the Hinton diagram was normalized to the respective golden model. With these tolerances, we set $\eta \sim N(0, 1)$ in our analysis.

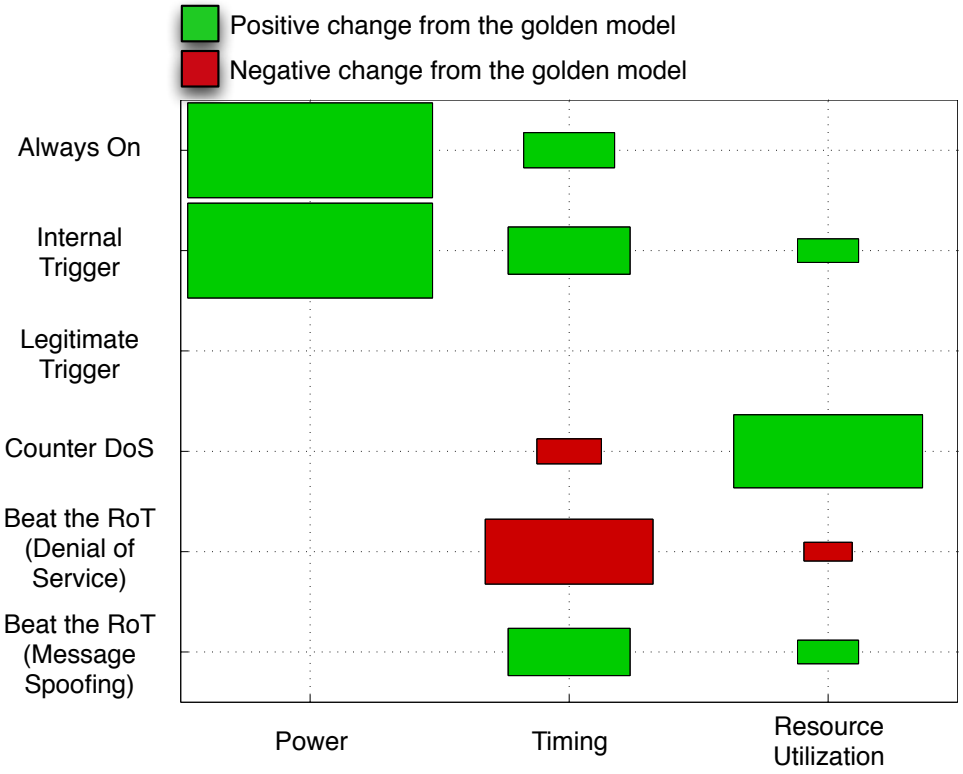


Figure 4.9: Hinton visualization for various HTTs

Since the RoT Trojan that leaked the sensitive information shared same characteristics with the Always On Trojan, the details are omitted in the Hinton Diagrams. From the visualizations in Figure 4.9, the following observations can be made:

(a) Any Trojan driving additional output (leaking keys through LEDs) was easily detectable based purely on power consumption, as shown by the always on HTT and the internal trigger HTT. Whereas in the case of counter DoS HTT and the message spoofing HTT, the additional logic branching off of the clock signal leads to a noticeable skew in the timing metric;

(b) Both the legitimate user trigger and Man-in-the-middle RoT Trojan remained stealthy under power consumption, timing and utilization since the HTT only contained a simple complement logic that inverts the bits in certain positions of the input data which in turn leads to minimum unnoticeable use of resources;

(c) Using power consumption, timing variation and utilization, 57%, 43% and 57% of the implemented Trojans remained stealthy. This in turn proves that existing post-silicon testing techniques based on physical characteristics such as power consumption, timing variations 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.

Figure 4.10 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 ($\tau_p$) is 3.1 and it has the lowest sum of false alarm and missed detection probabilities, see Figure 4.12. Using $\tau_p = 3.1$, we were able to detect 86% of the implemented HTTs. In Figure 4.11, we also observe that tuning $\tau_p \leq 3$, can increase the detection rate to 100%, however, in Figure 4.12, we note that the sum of false alarm and missed detection probabilities are

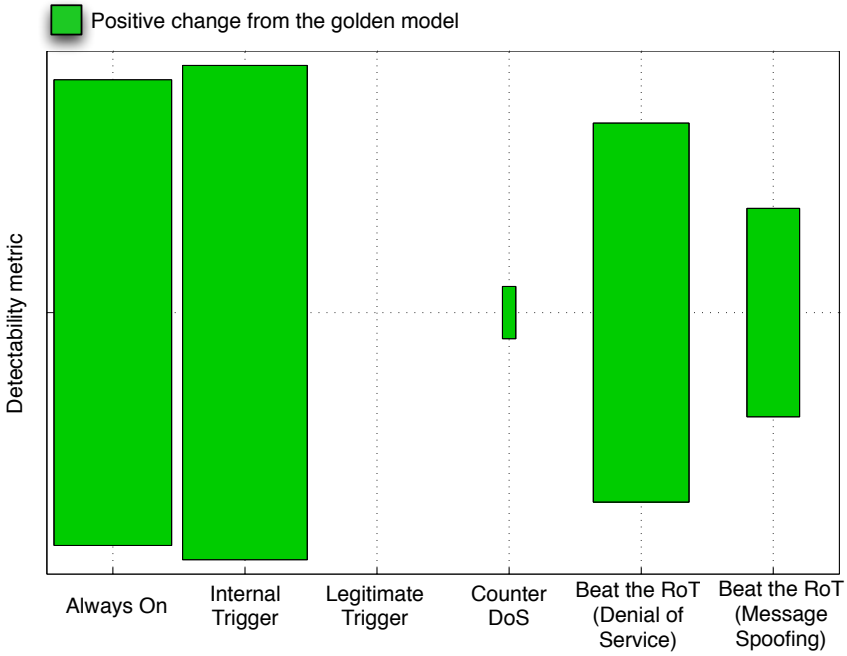


Figure 4.10: Residue Hinton Diagram for Detection

indeed high for $\tau_p \leq 3$. Hence, our model sets $\tau_p = 3.1$, see equation (4.8).

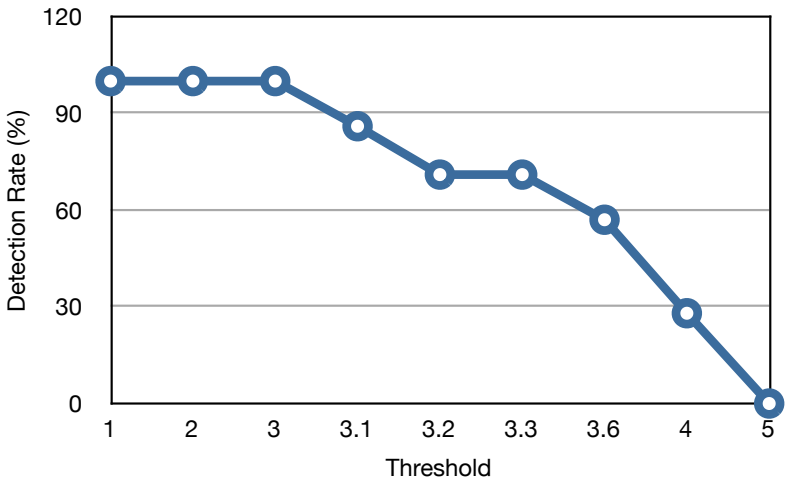


Figure 4.11: Detection rate for different detection thresholds

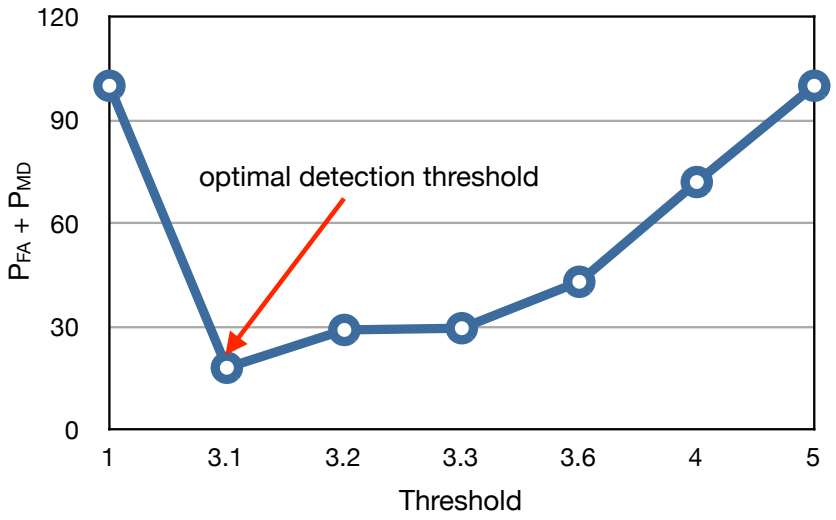


Figure 4.12: Sum of false alarm and missed detection vs detection threshold

## 4.7 Architectural Refinements to RoT Design

Based on the results obtained in Figures 4.9 and 4.10, HDM metric were able to detect only 86% of the designed HTTs. To enhance the confidence in HTT detection, a real-time intrusion detection module called Trusted Anchor is proposed. In this chapter, we consider the Trojans that leak sensitive information. Trusted Anchor is configured with the following set of policies related to information leakage:

(1) Unauthorized IP cores are restricted from memory access,

(2) The key is never allowed to leave an IP core via IP core-IP core or IP core-I/O links.

Trusted Anchor monitors the communication links for any violation of these policies and if detected, it will reset the device or perform memory/key zeroization. These set of policies are embedded inside the processing system partition of the Zynq platform, where it has more of a supervisory control and can also easily poll the I/O in a daisy chain fashion. The overhead in terms of implementation and latency need to be investigated and is addressed in the next chapter.

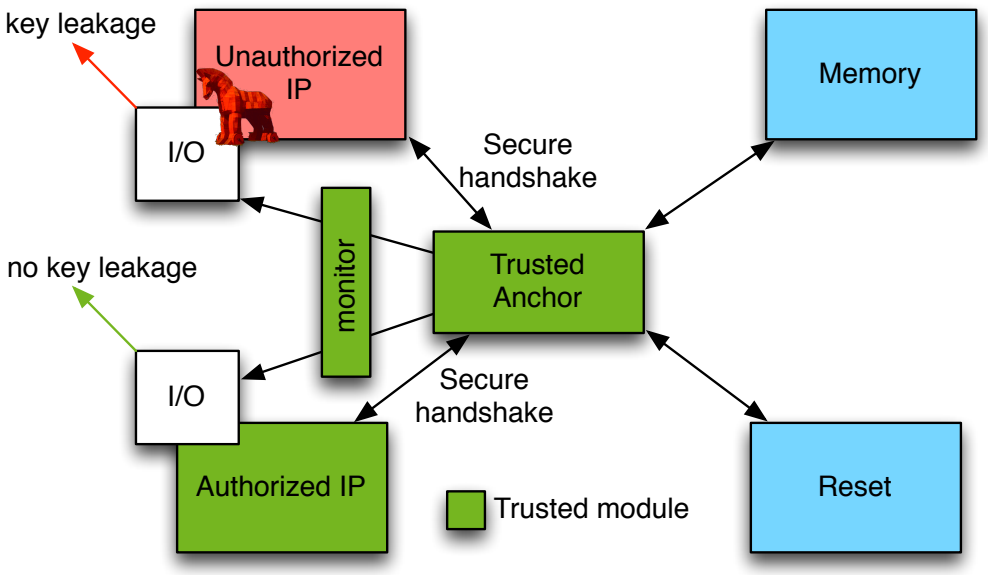


Figure 4.13: Trusted Anchor Architecture monitoring communication links for violation of policies

## 4.8 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 studies to determine the optimal HTT detection threshold that minimize the summation of false alarm and missed detection probabilities. Furthermore, to enhance trust in detection, we proposed an architectural refinement in the form of a trusted anchor for embedding user-defined policies that prevents HTTs from activating within the system.

# Chapter 5

# FIDES: Enhancing Trust in Reconfigurable Hardware Systems

## 5.1 Introduction

Extensive use of third-party IP cores (e.g., HDL, netlist) and open source tools in the FPGA application design and development process in conjunction with the inadequate bitstream protection measures have raised crucial security concerns in the past for reconfigurable hardware systems. Designing high fidelity and secure methodologies for FPGAs are still not mature and in particular, there are almost no concrete methods/techniques that can ensure trust in FPGA applications not entirely designed and/or developed in a trusted environment. This work strongly suggests the need for an anomaly detection capability within the FPGAs that can continuously monitor the behavior of the underlying FPGA IP cores and the communication activities of IP cores with other IP cores or peripherals for any abnormalities. To capture this need, we propose a technique called FIDelity Enhancing Security (FIDES) methodology for FPGAs that uses a combination of access control policies and behavior learning techniques for anomaly detection.

FIDES essentially comprises of two components:

(i) *Trusted Wrappers*, a layer of monitors with sensing capabilities distributed across the FPGA fabric; these wrappers embed the output of each IP core $i$ with a tag $\tau_i$ according to the predefined security policy $\Pi$ and also verifies the embeddings of each input to the IP core to detect any violation of policies. The use of tagging and tracking enables us to capture the normal interactions of each IP core with its environment (e.g., other IP cores, memory, OS or I/O ports). *Trusted Wrappers* also monitors the statistical properties exhibited by each IP core module on execution such as power consumption, number of clock cycles and timing variations to detect any anomalous operations;

(ii) a *Trusted Anchor* that monitors the communication between the IP cores and the peripherals with regard to the centralized security policies $\Psi$ as well as the statistical properties produced by the peripherals.

To thwart an adversary from tampering or disabling the proposed security components during the deployment stage, our architecture generates a secure bitstream consisting of the IP cores, Trusted Wrappers and Trusted Anchor, secured using public key cryptography. We implemented FIDES architecture on a Xilinx Zynq-7020 device running a red-black system comprising of sensitive and non-sensitive IP cores. Our results show that the FIDES implementation leads to only 1-2% overhead in terms of the logic resources per wrapper and incurs minimal latency per wrapper for tag verification and embedding. On the other hand, as compared to the baseline implementation, when all the communications within the system are routed to the Trusted Anchor for centralized policy checking and verification, a latency of 1.5X clock cycles is observed; this clearly manifests the advantage of using distributed wrappers as opposed to centralized policy checking.

Due to the size, weight and power (SWaP) advantages, FPGA devices are the primary source for both computation and communication tasks in various mission and safety critical embedded systems such as smart grid, embedded and network encryption, avionics, tactical

radios, satellite communications, finance and banking, and homeland security. While the benefits of FPGAs are great, hardware security is now emerging as a serious concern for FPGAs due to the extensive use of third-party IP cores/functions (e.g., HDL, netlist), open source tools (e.g., CAD) and unauthenticated configuration bitstreams in the application design and development process. For instance, efforts [11, 16, 7] show how one can exploit various vulnerable stages of the hardware lifecycle to inject stealthy malwares ('Trojans') into the ASIC and FPGA devices. Evidence of the existence of Trojans in the forms of backdoors in military grade and wireless communication devices are also shown in [94]. Typically, an FPGA application is a collection of intellectual property (IP) cores and soft-core processors which are in turn glued together to implement specific functionalities within a system. In many cases, the IP cores (available in the form of HDL, netlist or bitstream) are procured from multiple third-party vendors with varying trust levels which provides ample opportunities for an adversary to inject hardware Trojans. A successful triggering of one such Trojan could create 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. Enforcing hardware security at the device level and system level is very critical which in turn has motivated defense agencies such as DARPA and IARPA to launch innovative programs such as Supply Chain Hardware Integrity for Electronics Defense (SHIELD) [95] and Trusted Integrated Chips (TIC) [96].

Besides these programs, a handful of research efforts along the tangents of detecting and identifying the Trojans in hardware have been proposed. The techniques presented in works such as UCI [30], VeriTrust [34], FANCI [32] and HaTCh [97] essentially focus on the idea of identifying and flagging unused or suspicious Trojan wires within a design. In HaTCh [97], for instance, authors use functional testing on the IP cores to identify all the list of unused wire combinations and augments additional tagging circuitry to the IP cores so as to keep track of the suspicious wires; as soon as a malicious wire is activated, the tagging circuitry will raise an exception to prevent the Trojan from depicting malicious behavior.

A major drawback with all these approaches is the assumption that the source code of the IP core design is available to the end consumers in HDL/netlist format and hence, often neglect the scenarios where the consumers have access only to the configuration bitstreams ('hard IP cores'). Proof-Carrying Codes [23] are also proposed where the IP vendor will construct a formal proof of the design adhering to certain security properties, which will be later verified by IP consumer to ensure that design is free from modifications. Besides the overhead involved in creating huge proofs for even smaller codes, loopholes can also occur while defining many security properties.

Compared to above efforts, the concepts presented in [38], [40], and [98] assumes a 'black-box' methodology where no assumption is made on the availability of the source code and are perhaps closest to our design. In [38], authors propose an application-dependent security infrastructure monitoring datapath signals for illegal behaviors. This approach adds reconfigurable Design-for-Enabling-Security (DEFENSE) logic to the functional design to implement a centralized run-time security monitor. The hardware-based monitors are configurable finite-state machines (FSMs) that check the current set of signals for the properties specified by the designer. Huffmire *et al.* [40] propose a memory protection mechanism by designing and implementing policy-driven memory protection techniques using reconfigurable hardware. Their work develops an access policy language that precisely describes the fine-grained memory separation of modules on an FPGA and a policy compiler that converts the specified memory access policies into enforcement hardware modules. This memory access policy mechanism is then integrated into the hardware-based centralized reference monitor to detect malicious memory accesses at run-time. The use of centralized monitors not only affects the scalability of the security mechanisms in [42, 99, 41] but also makes it vulnerable to classic communication attacks such as spoofing, Man-in-the-Middle (MITM) that enables an attacker to spoof or tamper the signals/messages to the centralized monitor, as indicated by [100, 101]. Another drawback of these approaches is the inability to identify the malicious IP cores that conforms to the access control policies but deviates from the normal execution behavior; for e.g., a malicious encryption IP core instead of encrypting the data when trig-

gered leaks the secret key. In such scenarios, monitoring the normal execution behavior of IP cores and peripherals such as number of clock cycles, total power consumption in real-time can be used to detect any abnormalities.

**Motivation and Contribution.** The addition of third-party IP cores and the use of open source tools in the design and development process renders an FPGA application vulnerable to hardware Trojan-based attacks. There are currently no concrete methods/techniques that can be easily instrumented into the FPGAs and provide trust guarantees in FPGA applications which are not completely developed in a trusted environment. The security methods currently available have the following shortcomings:

(a) requires low level knowledge of the IP core design;

(b) vulnerable to traditional communication attacks;

(c) centralized approach;

(d) inability to detect functional behavior deviation.

Moreover most existing solutions overlook the communication activities and functional behavior of IP core which in turn plays an important role in actually carrying out the attack. This paper addresses such a significant loophole by proposing a secure communication framework and behavior monitoring module that can reliably thwart hardware Trojan-based attacks.

Alternatively speaking, motivated by the shortcomings in the existing works, we strongly argue the need for an anomaly detection capability within the FPGAs that can be easily instrumented with the application and continuously monitor the underlying IP core behavior and also the access activities of IP cores with other IP cores or peripherals for any signs of abnormalities. To capture this need, we propose a technique called FIDelity Enhancing Security (FIDES) methodology for FPGAs that uses a combination of access control policies

and behavior learning techniques for anomaly detection. FIDES essentially comprises of two components:

(i) *Trusted Wrappers*, a layer of monitors with sensing capabilities distributed across the FPGA fabric; these wrappers stamp the output of each IP core $i$ with a tag $\tau_i$ according to the predefined security policy $\Pi_i$ and also verifies the embeddings of each input to the IP core to detect any violation of policies. The use of tagging and tracking enables us to capture the generalized interactions of each IP core with its environment (e.g., other IP cores, memory, OS or I/O ports). *Trusted Wrappers* also monitors the statistical properties exhibited by each IP core functions on execution such as power consumption, number of clock cycles and timing variations to detect any anomalous operations;

(ii) a *Trusted Anchor* that monitors the communication between the IP cores and the peripherals with regard to the centralized security policies $\Psi$ and the statistical properties produced by the peripherals.

We prototyped FIDES architecture on a Xilinx Zynq-7020 device, that consists of a hard-fused ARM core and an Artix FPGA. The Trusted Wrapper is implemented in hardware logic and monitors the communication and execution behavior of IP core in a distributed manner. The Trusted Anchor is mapped to the ARM core and controls the communication of the IP cores with the external peripherals (enabled to thwart leakage of sensitive information) and provides behavioral monitoring of the peripherals (enabled to thwart covert attacks). We target FIDES architecture on a Xilinx Zynq-7020 device implemented with a red-black system comprising of sensitive and non-sensitive IP cores. Our results show that FIDES implementation leads to only 1-2% overhead in terms of the logic resources/wrapper and incurs minimal latency/wrapper for tag verification and embedding. On the other hand, as compared to the baseline implementation, when all the communications within the system are routed to the Trusted Anchor for centralized policy checking and verification, a latency of 1.5X clock cycles is observed; this clearly manifests the advantage of using distributed wrappers as opposed to centralized policy checking

To the best of our knowledge, we design and implement a distributed and secure anomaly detection framework for FPGAs that can identify abnormal FPGA behaviors (hardware Trojans) in real-time.

**Roadmap**: Section 5.2 provides the detailed description of the FIDES methodology along with the description of the adversary/threat model and the system design of the distributed information model. The hardware implementation of the FIDES architecture is described in Section 5.3 with the results presented in Section 5.4. Section 5.5 concludes this chapter.

## 5.2 Proposed Security Framework

This chapter proposes a novel, distributed and secure methodology for enhancing trust in FPGA applications designed or developed under untrusted environments called FIDES (FIDelity Enhancing Security methodology) named after the Greek God of Trust. FIDES essentially relies on the use of two major approaches: (a) *decentralized information flow control* (DIFC) model that enables safe and distributed information flows between various elements of FPGA such as IP cores, physical memory and registers by annotating/tagging each data item with its sensitivity level and the identity of the participating entities; and (b) *statistical learning techniques* that learns the normal functional behavior of IP cores, peripherals defined as *conformant core behavior* ($\mathcal{CCB}$) during the FPGA application integration/testing stage and leverages the learned $\mathcal{CCB}$ model to detect any anomalous behavior deviations in run-time.

We will initially present the adversary and threat model and then discuss the detailed FIDES system design in Section 5.2.2. Table 5.1 presents the summary of various notations used for describing the FIDES design.

Table 5.1: Table of Notations

| Notation | Meaning |
|---|---|
| $<Core>_i^s$ | Sensitive IP core $i$ |
| $<Core>_i^{ns}$ | Non-sensitive IP core $i$ |
| $W_i$ | Trusted Wrapper assigned to $<Core>_i$ |
| $\mathcal{TA}$ | Trusted Anchor |
| $\Pi_i$ | User-defined policies for $<Core>_i$ |
| $\Psi$ | User-defined policies at $\mathcal{TA}$ |
| $\tau_i$ | Tags assigned to $<Core>_i$ |
| $\sigma_i$ | $<Core>_i$ functional behavior signature |
| $\mathcal{M}^{\mathcal{S}}$ ,$\mathcal{M}^{\mathcal{N}}$ | Sensitive and Non-sensitive memory regions |
| $\alpha$ | Characteristics of IP core $(\mathcal{S},\mathcal{N},\mathcal{E})$ |

### 5.2.1 Adversary and Threat Model

Existing literature show that various layers of the FPGA lifecycle ranging from design to final deployment stage are vulnerable to Trojan intrusions. This work assumes trusted FPGA application integration and testing stage however untrusted design, development and deployment stage due to the immense involvement of the third-party IP cores, CAD tools and also, easily accessible FPGA configuration bit streams. With this assumption, a more realistic adversary model is considered: (i) at design and development stage, the Trojan might be present in the third-party IP core or CAD tool, or might be inserted by a rogue designer during the design process; (ii) at deployment stage, the Trojan might be injected into the FPGA configuration bitstream. A handful of works along the lines of cryptographic defenses (e.g., use of bitstream signatures [53], design obfuscation [52]) have been proposed to thwart tampering attacks on FPGA bitstream. FIDES architecture will leverage those existing defenses based on asymmetric cryptography.

Embedded systems often contains a mixture of critical and non-critical applications residing on the same platform. Any intentional or accidental fault on the operation of the critical application could have an impact on safety, or could cause large financial or social loss. It is therefore necessary to thwart the non-critical components from interrupting or compromising the critical operations. This paper considers a similar scenario called the red-black system implemented in FPGA that will encompass multiple IP cores procured from various third-party vendors with different sensitivity levels. In such a system, two distinct kinds of IP core ('Core') sensitivity levels are considered: (a) IP core that accesses and processes critical and sensitive information (e.g., network encryption) called *sensitive core* ($<Core>_i^s$) and (b) IP core that accesses and processes less critical and sensitive information called *non-sensitive core* ($<Core>_i^{ns}$).

Proper segregation between the sensitive and non-sensitive operations should be realized so that an untrusted IP core will be thwarted from accessing the shared resources (e.g., memory, peripherals) and transmit the information to the outside world via a covert channel (e.g., secret unnoticeable channels) or a normal channel (e.g., send sensitive cryptographic information by mixing with the noise levels or encrypting with the attacker-known keys). With regard to the red-black system design, we derive the attack scenarios given by ([A1] - [A4]) and propose the security policy requirements addressed by ([R1] - [R4]).

**Attack Scenarios.**

A1. $<Core>_i^s$ accesses the critical and sensitive information and export the information to the outside world.

A2. $<Core>_i^{ns}$ accesses the critical and sensitive information and exports the information to the outside world.

A3. $<Core>_i^{ns}$ uses trusted $<Core>_j^s$ as a conduit (a Man-in-the-middle) to access sensitive regions and exports the information to the outside world.

A4. Malicious IP core spoofs as a legitimate core and can access application resources.

**Security Requirements.**

R1. $<Core>_i^{ns}$ shall be prevented from accessing and processing the sensitive information.

R2. Cores shall be restricted from accessing the peripherals directly to limit export of data.

R3. The information flow between $<Core>_i^{ns}$ and $<Core>_i^{s}$ shall be restricted or performed in the presence of a security officer.

R4. The statistical properties of each $<Core>_i$ shall be measured and monitored to detect any anomalous functional behavior deviations.

### 5.2.2 System Design

**Initialization phase:** During the FPGA application integration/testing stage, the following components will be generated by the end consumer: (a) *Trusted Wrapper* ($W_i$) - a thin layer of hardware logic that interface with each IP core $<Core>_i$ and manages/monitors the communication activities of $<Core>_i$ with other cores ($<Core>_j$, where $j \neq i$); (b) *Trusted Anchor* ($\mathcal{TA}$) - a centralized security officer, implemented in ARM, that manages/monitors the communication of the IP cores with the external world (I/O Peripherals) according to the predefined security policy $\Psi$; (c) *Tagged Physical Memory* - a partitioned memory with *tags* annotated to the data denoting their sensitivity level; and (d) *Secure Bitstream Blob* - a cryptographically signed bitstream blob containing $W_i$; $\forall i \in \mathbb{S}$, where $\mathbb{S}$ represents the set of IP cores and $\mathcal{TA}$ modules with the private key ($K$-) of the end consumer; the signed blob together with the public key ($K+$) will be loaded into the memory. A hash of the ($K+$) will be also loaded into the trusted region within the processor to thwart any attempts to modify the bitstream signatures. The use of cryptographic signatures will prevent an adversary from subverting the protection schemes included within the configuration bitstream. The complete architecture of FIDES is presented in Figure 5.1. FIDES specifically provides three functionalities:

- **Prevention**: The use of asymmetric cryptographic signatures prevents unauthorized tampering of bitstreams (disabling or modifying the protection schemes) built with the proposed protection schemes (Trusted Wrappers and Anchor).
- **Detection**: The combination of Trusted Wrappers and Trusted Anchor detects any unauthorized communication (e.g., non-sensitive IP core accessing sensitive memory regions or communicating with sensitive IP cores) and anomalous execution of IP cores and peripherals.
- **Response**: On detection of an attack, the framework responds by logging the details of the attack in the *Logging buffer* and communicating to the external world for further diagnostics.

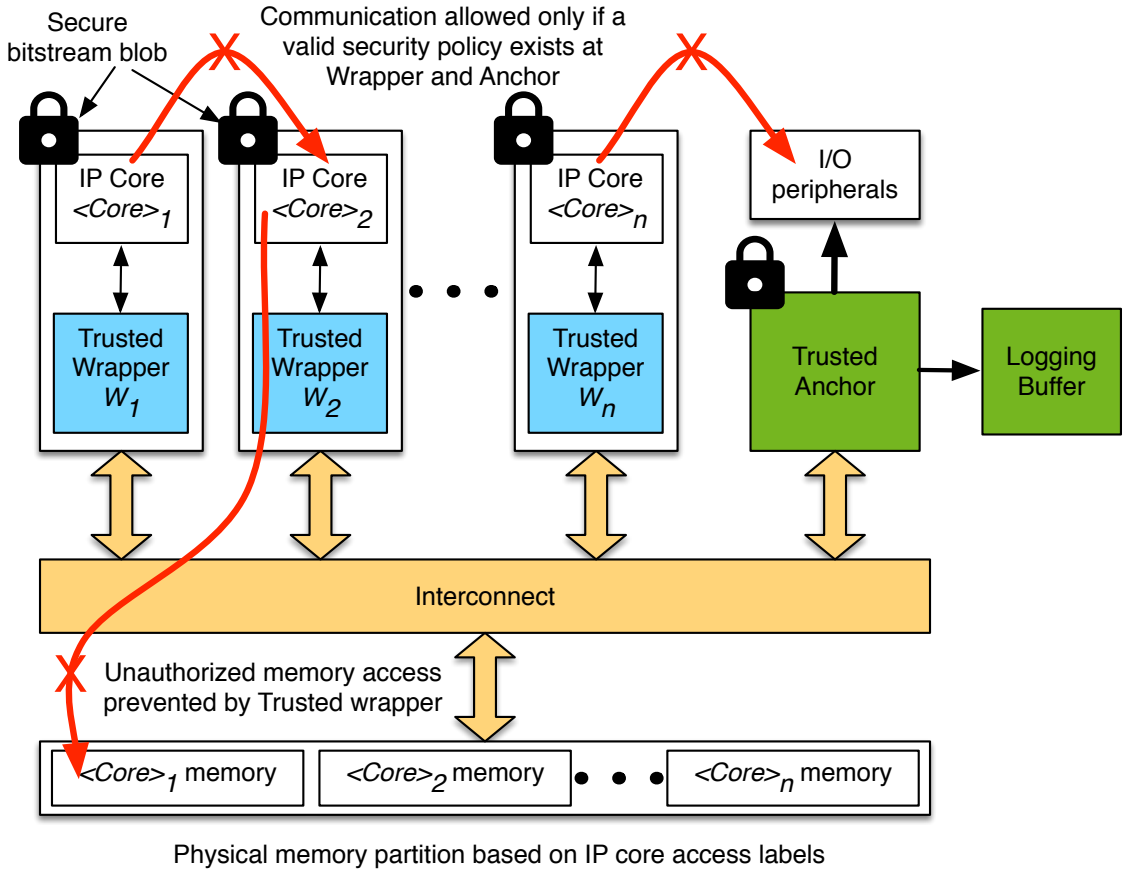


Figure 5.1: FIDES Architecture

FIDES specifically *prevents* unauthorized modification of bitstreams through the use of asymmetric cryptographic signatures and detects any unauthorized communication and execution behavior of IP cores and peripherals through the use of Trusted Anchor and Trusted Wrappers. We also present the various approaches used by FIDES to enable these security protections in the following sections.

### 5.2.3 Decentralized Information Flow Model

FIDES uses tag-based information flow control methodology to achieve decentralized and safe flow of information between various elements in the system [102]. In order to do so, during the initialization phase, each Trusted Wrapper $W_i$ will be assigned a *tag* ($\tau_i$) that reflects the characteristics ($\alpha$) of the information handled by the $<Core>_i$ and also the identity ($i$) of the participating entity (e.g., IP Core identity).

**Initialization phase:** $W_i \Leftarrow \tau_i;\ \tau_i =< \alpha, i >$

Where $\alpha = (\mathcal{S}, \mathcal{N}, \mathcal{E})$. Here $\mathcal{S}, \mathcal{N}$ and $\mathcal{E}$ denotes sensitive, non-sensitive and empty tokens, respectively. The value of $\alpha$ will be assigned based on the following policies:

- $<Core>_j^s | W_j \Leftarrow <\mathcal{S},\ \text{j}>$, $<Core>_j^{ns} | W_j \Leftarrow <\mathcal{N},\ \text{j}>$: The IP cores $<Core>_i^s$ and $<Core>_i^{ns}$ will be labeled with $\mathcal{S}$ and $\mathcal{N}$ tokens, respectively, depending on the characteristics of the information handled. In cases where an IP Core processes both sensitive and non sensitive information, the wrapper will be assigned a $\mathcal{S}$ token.

- Tagged physical memory: The physical memory regions $\mathcal{M}$ will be partitioned into sensitive and non-sensitive regions by tagging with $\mathcal{S}$ and $\mathcal{N}$ tokens denoted by $\mathcal{M}^{\mathcal{S}}$ and $\mathcal{M}^{\mathcal{N}}$, respectively. The memory regions can be also tagged with the identity $i$ of the owner to enable horizontal access control protection, if needed. Alternatively speaking, under horizontal access control protection $<Core>_i^s$ will be prevented from accessing the memory space belonging to $<Core>_j^s$, though both cores are tagged with $\mathcal{S}$.

- $\mathbb{P} \Leftarrow <\mathcal{E}>$: The I/O peripherals (external world), denoted by $\mathbb{P}$, will be labeled with empty tokens ($\mathcal{E}$). This approach of assigning empty tokens will prevent the IP cores from releasing the sensitive information to the external world as tagging-based information flow rule requires an IP core to hold an $\mathcal{E}$ token to release the information to the peripherals.

Since the system is distributed and each entity is not aware of the tag of others, the messages exchanged between the entities is labeled appropriately by the Trusted Wrappers. Then the receiving wrappers will use the tagging based information flow rules to determine whether to accept or reject the messages. Enforcing tags on the messages also ensures that the information will be always prefixed with a tag when stored in temporary or permanent memory locations and only components that follows the tagging-based information flow rules can access the information.

### 5.2.4 Tagging-based Information Flow Rule

The information labeled with the tag of $\tau_j = <\alpha_j, j>$ can flow to $\tau_k = <\alpha_k, k>$, only if the tags of $\tau_j$ is included in $\tau_k$. We define the partial order $\preceq$ (pronounced can flow to) for two tags $\tau_j$ and $\tau_k$ as

$$\tau_j \preceq \tau_k; \text{ if } \alpha_j = \alpha_k \text{ and } (j,k) \in \Pi_k$$

Where $\Pi_k$ is the security policy defined at the IP core $k$. The security policy $\Pi_i$ essentially captures the authorized set of communications for each IP core $i$ based on the design specifications. The above equation mainly states that core $k$ can accept the data from core $j$ only if the security policy defined at $k$ ($\Pi_k$) allows the communication from core $j$ and the characteristics token $\alpha$ at both $j$ and $k$ matches. Under this rule the following communication will be prevented.

$$\tau_j = (< \mathcal{S}, j >) \not\preceq \tau_k = (< \mathcal{N}, k >)$$
$$\tau_j = (< \mathcal{N}, j >) \not\preceq \tau_k = (< \mathcal{S}, k >)$$
$$\tau_j = (< \mathcal{S}, j >) \not\preceq \mathbb{P} =< \mathcal{E} >$$
$$\tau_j = (< \mathcal{N}, j >) \not\preceq \mathbb{P} =< \mathcal{E} >$$

While the above rules prevent the contaminated cores from communicating with other cores or releasing the information, it would also make it implausible for any legal communication activities or any sensitive data to get out of the system. Therefore, DIFC supports *tag declassification* capability (denoted by $\downarrow$) which enables the higher tags to declassify to lower tag to receive information; especially, we consider the following two scenarios:

(a) *IP cores that process both sensitive and non-sensitive information labeled with tag* $\mathcal{S}$ - We provide an ability for $W_{i;\forall i \in \mathbb{S}}$ to declassify its tag from $\mathcal{S}$ to $\mathcal{N}$ such that $\mathcal{S} \downarrow \mathcal{N}$. To up the privilege back to $\mathcal{S}$, the $<Core>_i^s | W_i$ will communicate with $\mathcal{TA}$ initially and $\mathcal{TA}$ will set the privilege of $W_i$ back to $\mathcal{S}$, only if security policy at $\mathcal{TA}$ allows. If $W_i$ is allowed to up its privilege without contacting $\mathcal{TA}$, it can be leveraged by a non-sensitive compromised IP core to defeat the system. As an example, consider the attack where a $<Core>_i^{ns}$ uses $<Core>_j^s$ as a conduit (a Man-in-the-middle) to access sensitive memory areas and export the information to the outside world. If in the absence of $\mathcal{TA}$ , the $<Core>_j^s$ will declassify to $\mathcal{N}$ to receive data from the $<Core>_j^{ns}$ and up its privilege back to $\mathcal{S}$ token to access the sensitive regions; once accessed, the IP core will declassify its privilege to $\mathcal{N}$ to export the information to $<Core>_j^{ns}$. To avoid such attack scenarios, we impose the following constraint: a $W_i$ can declassify only after receiving an explicit request from $\mathcal{TA}$;

(b) *IP core that exports the data to the external world (I/O peripherals)* - To export the data, declassification privileges are assigned to $\mathcal{TA}$ (declassify the label from $\mathcal{S}/\mathcal{N}$ to

$\mathcal{E}$) based on the predefined security policies, $\Psi$ at $\mathcal{TA}$. The $\Psi$ essentially captures the authorized set of peripherals for each IP core based on the design specifications.

**Remark #1:** This work does not consider the presence of Trojans on interconnect. Existing efforts can be leveraged to build secure and trustworthy interconnects such as the time-based techniques in [103] to detect malicious interconnect operations or the use of secure checksums, e.g., Message Authentication Codes (MACs) [104], to prevent the Trojans within the interconnect modifying or spoofing the messages.

**Remark #2:** The security of the proposed FIDES design depends on the policies defined by the user, $\Pi$ and $\Psi$. This work assumes that the policies are generated by the end consumer based on the design specifications and requirements.

### 5.2.5 Behavior Monitoring

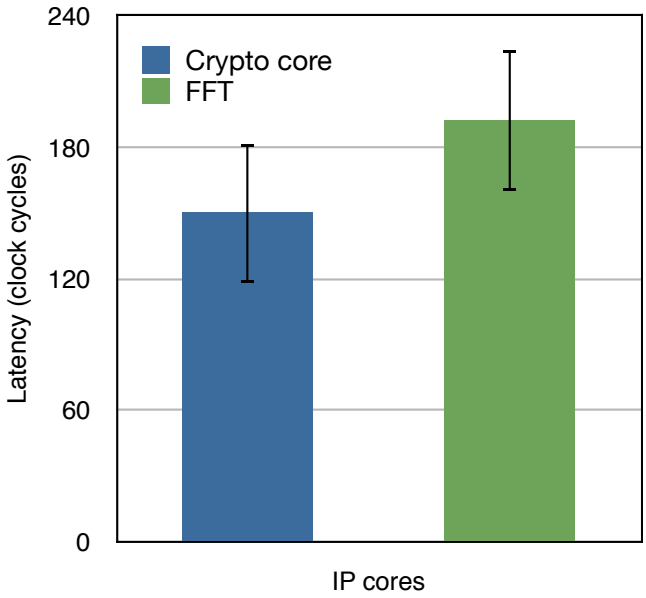


Figure 5.2: IP cores with distinct latency profiles

Besides labeling and monitoring the communication activities, to boost the trust in an

FPGA application, the characteristics of the underlying tagged IP cores and peripherals ($\mathbb{P}$) will be also observed and used by the Trusted Wrappers for identifying malicious behaviors and decision making (e.g., forward/drop the message, report to $\mathcal{TA}$). This technique identifies the malicious IP cores that conforms to the access control policies but deviates from the normal execution behavior; for e.g., a malicious IP core that performs erroneous computations on the input data. As each IP core exhibit different statistical properties on execution (e.g., power consumption, number of clock cycles, and timing variations), Trusted Wrappers will use these properties to determine the normal functional execution behavior of IP cores defined as *conformant core behavior* ($\mathcal{CCB}$). In Figure 5.2, we show that each IP core exhibits distinct behavior such as *number of clock cycles* during its execution. Therefore in the FPGA application integration and testing stage, for each $<Core>_i$, we compute a signature, denoted by $\sigma_i$, that captures these statistical properties. Together with the *tag* ($\tau_i$), during the initialization phase each wrapper $W_i$ will be assigned $\sigma_i$ as follows:

**Initialization phase:** $W_i \Leftarrow \sigma_i$

In the run-time phase, $W_i$ will monitor the statistical properties produced by $<Core>_i$ and compares with the already learned signature $\sigma_i$. If the difference between the expected and observed behavior exceeds a user-defined threshold, $W_i$ flags the observation as "anomalous" and informs $\mathcal{TA}$ for further actions. Similar to Trusted Wrapper, $\mathcal{TA}$ will also monitor the behavior of external peripherals to detect any anomalous leaking of information. The intuition behind using statistical properties for malicious behavior detection comes from [100, 101] where a combination of several statistical properties such as power consumption, clock cycles, called the Hardware Trojan Detection Metric (HDM), can be used to detect malicious IP core operations.

**Remark #3:** In this paper, we do not address heat or power-based Trojans. However, we can easily extend our work with metrics that can detect these Trojans.

### 5.2.6 Security Proofs

We here show with proofs on how FIDES can detect and prevent the attack scenarios presented in [A1]-[A4].

A1. $<Core>_i^s$ accesses the critical and sensitive information and export the information to the outside world using either of the two approaches: (a) covert channel (secret and unnoticeable) and (b) obfuscate (mixing the data with random noise) the information and transport via normal channels:

*Proof.* (a) $<Core>_i^s$ accesses the critical and sensitive information from the memory ($\mathcal{M}^{\mathcal{S}}$) and forward the information to the $\mathcal{TA}$. Since the covert channel is not an authorized peripheral for $<Core>_i^s$, $\mathcal{TA}$ will reject the information according to the predefined policies $\Psi$. (b) As obfuscation leads to noticeable changes from the learned signature $\sigma_i$, wrapper $W_i$ will flag $<Core>_i^s$ as anomalous and informs $\mathcal{TA}$ about the observation. □

A2. $<Core>_i^{ns}$ accesses the critical and sensitive information and exports the information to the outside world

*Proof.* This attack will fail as tagging-based information flow rule restricts $<Core>_i^{ns}$ tagged with $\mathcal{N}$ tag from accessing the sensitive memory regions such that $\mathcal{M}^{\mathcal{S}} \not\preceq \tau_i =< \mathcal{N}, i >$ □

A3. $<Core>_i^{ns}$ uses trusted $<Core>_j^s$ as a conduit (a Man-in-the-middle) to access sensitive regions and exports the information to the outside world

*Proof.* As tagging-based information flow rule either prevents $<Core>_i^{ns}$ tagged with $\mathcal{N}$ tag from accessing the sensitive IP cores ($\tau_i = (< \mathcal{N}, i >) \not\preceq \tau_j = (< \mathcal{S}, j >)$) or forces $<Core>_j^s$ to declassify its tag from $\mathcal{S}$ to $\mathcal{N}$, this attack fails. □

A4. Malicious IP core spoofs as a legitimate core and can access application resources

*Proof.* As each IP core $<Core>_i$ is integrated with $W_i$, all the communications originating from $<Core>_i$ will be appropriately labeled with the tag $\tau_i = <\alpha, i>$ which will prevent a malicious IP core from impersonating as a valid IP core. □

## 5.3 Hardware Implementation of FIDES

The communication framework in FPGA-based systems plays a critical role in supporting the hardware Trojan attacks and hence it is necessary to understand the existing bus-based protocols in the FPGA platform. FPGA-based systems with soft core processors consisted of interconnects such as Avalon in case of Altera and the processor local bus (PLB) in case of Xilinx previously. With the introduction of hard-fused ARM cores in FPGA devices, AXI interconnect serves as the primary protocol for communication between the ARM core and the different IP cores [105]. The AXI interconnect also serves as the communication mechanism for all the IP cores within the Xilinx ecosystem including the soft core processor such as MicroBlaze. The AXI protocol is a subset of the ARM Advanced Microcontroller Bus Architecture (AMBA) and the different AXI4 interfaces are compared in Table 5.2. The selection of AXI protocol allows a designer to focus only on the IP behavior and not the IP interconnect, when targeting different FPGA-ARM platforms. The AXI4-Lite interface is the most area efficient and best suited for control logic data transfer.

In case of Xilinx systems, the ARM core is referred to as the processing system (PS), while the FPGA fabric consisting of reconfigurable gates is called the programmable logic (PL). The IP cores are bundled with the AXI4-Lite interface by the IP Packager tool in Xilinx's Vivado Design suite. Each IP core is wrapped with a AXI wrapper, which configures the data transfer into the IP core using custom register interfaces. These memory mapped registers are configured as write registers for input ports and read registers for the output ports of the IP core. The AXI wrapper initializes these registers to a default size of 32 bits. By default,

Table 5.2: AXI4 interfaces

| **Features** | **AXI4** | **AXI4-Lite** | **AXI4-Stream** |
|---|---|---|---|
| Type | High-performance and memory mapped interface | Register-style interface | Non-address based high speed streaming interface |
| Data transfer | 256 cycles | 1 cycle | unlimited cycles |
| Data width | 32-1024 bits | 32-64 bits | unlimited bytes |
| Application | Embedded, memory | Control logic | DSP, video |

the AXI wrapper is configured to initialize write registers for the IP core's inputs and the IP core's outputs are routed to external I/O peripherals.

The Trusted Wrapper $W_1$ shown in Figure 5.3 is the top-level interface to the IP core $<Core>_1$. Each of the IP core's I/O port is assigned a tag $\tau_i$, which is mapped to a register from the AXI4 wrapper. Since the IP core's output is also monitored by $W_1$, the AXI wrapper is modified to incorporate read registers for the output ports. If the output port is directly connected to an external peripheral, the additional read registers for the output port preserves a copy of the data and its tag. Thus the $\mathcal{TA}$ supervises the data being transferred through the external I/O peripherals and is able to provide behavioral monitoring. In summary, the Trusted Wrapper $W_1$ consists of the following register configurations:

- `Write` registers for the input ports of the IP core,
- `Write` registers for tagging the input ports,
- `Read` registers for monitoring the IP core's output ports,
- `Read` registers for tagging the IP core's output ports,
- `Read` register for declassification of data transfer depending on the security policy

framed for the IP core,

- `Write` register for providing the modified output label based on the temporary downgrade of the security policy access of the IP core.

The Trusted Wrapper is embedded into the IP core during the Xilinx Vivado tool's IP Packaging and Configuration step. This ensures that the IP core's tagging functionality is available when instantiating the IP core in other designs. The trusted anchor $\mathcal{TA}$ is implemented on the ARM core, as it can easily monitor the data transfer through the AXI interconnect to the I/O peripherals, thus enabling communication and behavioral monitoring of peripherals.

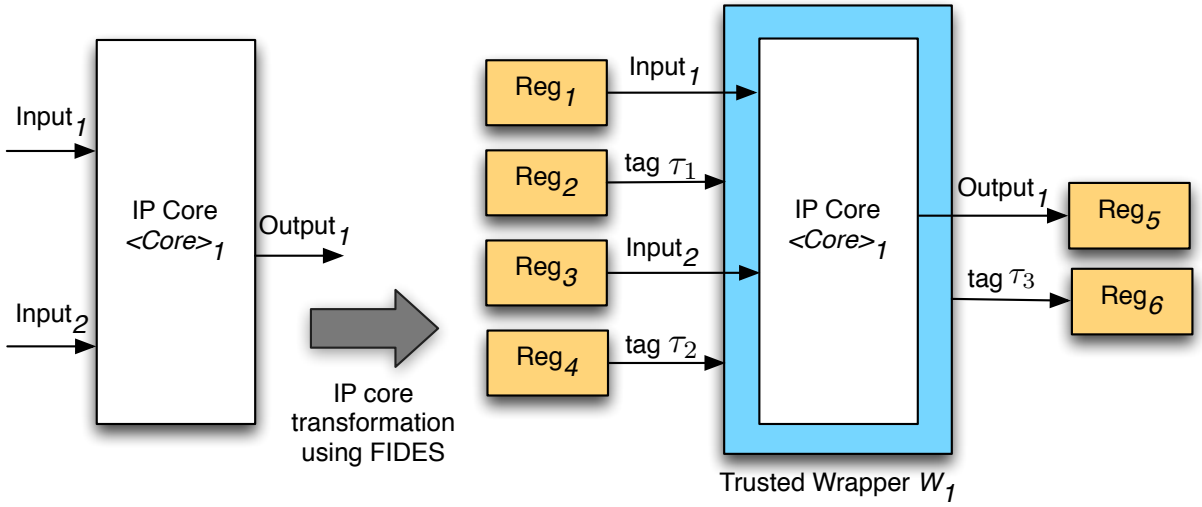


Figure 5.3: Trusted Wrapper mechanism for an IP Core

## 5.4 Results

The FIDES algorithm is prototyped on the MicroZed Xilinx Zynq-based Z7020 FPGA development board. The Zynq-7020 consists of a dual-core ARM processor and a Xilinx Artix FPGA with 85K logic cells. Both the baseline and the FIDES implementations are synthesized and implemented using Xilinx Vivado tool. The Vivado tool provides the logic resource

utilization of the complete design and also provides a definition file. The definition file is mapped using Xilinx SDK to execute the software algorithm implementation. Figure 5.4 shows the resource utilization obtained from Xilinx Vivado after embedding the IP core in the Trusted Wrapper. Interestingly, the Trusted Wrapper does not consume much resources and only increases the logic resource (LUTs and FF) utilization by about 1-2% as compared to the baseline implementation.

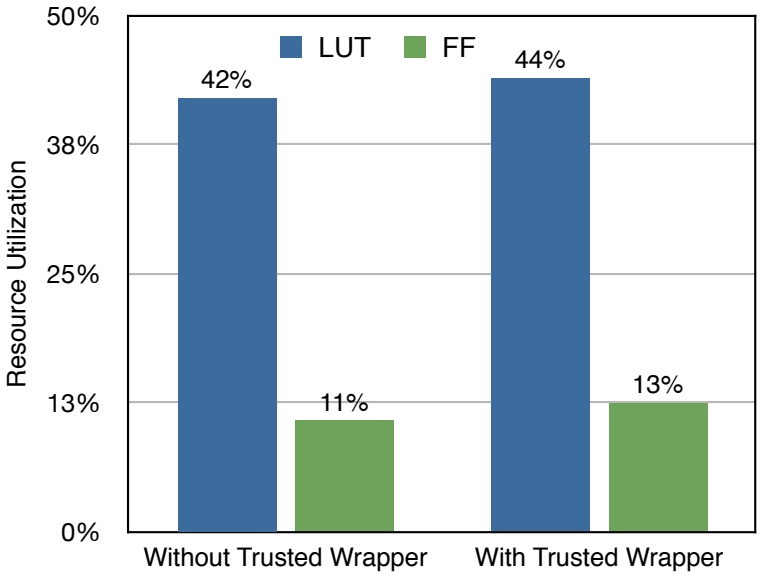


Figure 5.4: Resource utilization for an IP core with and without a Trusted Wrapper

Figure 5.5 shows the latency incurred by incorporating FIDES into the existing system. To clearly understand the latency overhead of our FIDES scheme, we implemented two cases, as given by the Procedures 1 and 2 below.

**Algorithm 3:** Communication flow between non-critical IP cores

1: $<Core>_i^{ns} \rightarrow$ Trusted Wrapper $W_i$

2: Trusted Wrapper $W_i \rightarrow$ Trusted Wrapper $W_j$

3: Trusted Wrapper $W_j \rightarrow <Core>_j^{ns}$

4: No declassification/classification messages sent from $W_i$ and $W_j \rightarrow$ to $\mathcal{TA}$

**Algorithm 4:** Communication flow between critical and non-critical IP cores

1: $<Core>_i^{ns} \rightarrow$ Trusted Wrapper $W_i$
2: Trusted Wrapper $W_i \nrightarrow$ Trusted Wrapper $W_j$
3: Request for declassification sent to $\mathcal{TA}$ by $W_j$
4: $\mathcal{TA}$ changes the privileges of $W_j$ from $\mathcal{S} \downarrow \mathcal{N}$, if policy allows
5: Trusted Wrapper $W_i \rightarrow$ Trusted Wrapper $W_j$
6: Trusted Wrapper $W_j \rightarrow <Core>_j^s$
7: $\mathcal{TA}$ up the privileges of $W_j$ from $\mathcal{N} \uparrow \mathcal{S}$, on completion

The latency measurement taken for two different IP cores crypto core and (64-point) Fast Fourier Transform (FFT) core, with regard to the Procedures 3 (denoted by 'no declassification' legend in figure) and 4 (denoted by 'declassification' legend in figure). In case of Procedure 3, as the communication is between two non-sensitive IP cores, no declassification messages were sent to $\mathcal{TA}$; the $\mathcal{TA}$ is contacted only once to export the data outside. Since the message passing through the AXI interconnect between the Trusted Wrapper and the $\mathcal{TA}$ contributes to the latency more as compared to the tag verification, we observed minimal latency for scenarios related to Procedure 1. On the other hand, in case of communication between non-sensitive and sensitive IP cores (or mismatched tags), depicted by Procedure 4, $\mathcal{TA}$ is contacted multiple times for declassification and classification of privileges which in turn led to an increase in latency ($\approx 1.5X$ clock cycles). From the results, we also note that the existing centralized communication framework for policy checking and verification can incur a larger overhead as opposed to the proposed distributed wrapper scheme since the data communication through the AXI Interconnect incurs a significant latency.

## 5.5 Conclusions

In this paper, we present the FIDelity Enhancing Security (FIDES) methodology for FPGAs to enhance trust in FPGA operations that are not completely designed or developed in a

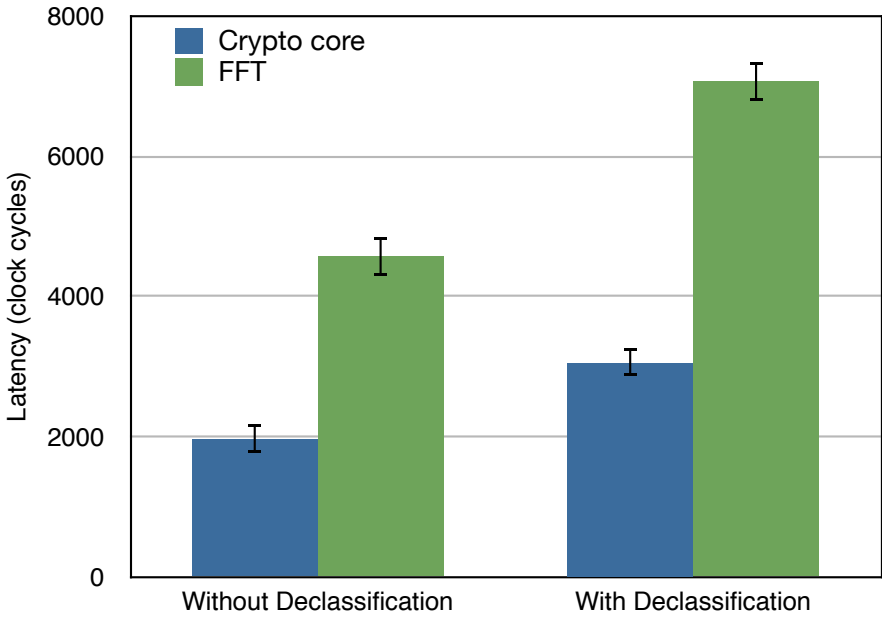


Figure 5.5: Latency measurement with the Trusted Wrapper

trusted environment. The security measures currently available for FPGAs have a number of shortcomings such as (a) require low level knowledge of the IP core design; (b) vulnerable to bus communication attacks; (c) centralized design; and (d) failure to detect hardware Trojans that deviate from normal functional behavior. We addressed a significant loophole present in these existing solutions by proposing a secure communication framework and behavior monitoring module that can reliably detect and prevent hardware Trojan based attacks. We implemented FIDES on the Xilinx Zynq-7020 device for a red-black system consisting of critical and non-critical IP cores. Our results yielded a 1-2% increase in hardware resources and 1.5X increase in latency (worst case scenario where all the communications happen between the Trusted Wrapper and Trusted Anchor) as compared to the baseline implementation. We observe that the resource utilization and latency results can be further optimized by clustering the IP cores into groups that share same characteristics, which will be addressed in future work.

# Chapter 6

# Detecting and Thwarting Hardware Trojan Attacks in Cyber-Physical Systems

## 6.1 Introduction

Cyber-physical system integrity requires both hardware and software security. Many of the cyber attacks are successful as they are designed to selectively target a specific hardware or software component in an embedded system and trigger its failure. Existing security measures also use attack vector models and isolate the malicious component as a countermeasure. Isolated security primitives do not provide the overall trust required in an embedded system. Trust enhancements are proposed to a hardware security platform, where the trust specifications are implemented in both software and hardware. This distribution of trust makes it difficult for a hardware-only or software-only attack to cripple the system. The proposed approach is applied to a smart grid application consisting of third-party soft IP cores, where an attack on this module can result in a blackout. System integrity is preserved in the

event of an attack and the anomalous behavior of the IP core is recorded by a supervisory module. The IP core also provides a snapshot of its trust metric, which is logged for further diagnostics.

A CPS combines physical processes with distributed control, computation and communication functions. Although these systems are complex, they are ubiquitous in the IoT era of autonomous vehicles/drones, smart homes, smart grids, etc. where everything is connected. These systems are vulnerable to unauthorized penetration due to the absence of safeguards for confidentiality, integrity, and availability. Examples such as the vulnerabilities in Internet-connected cars allowing hackers to control the car [9] or the Lockheed drone brought down by jamming its communication network and spoofing its GPS sensors [8] demonstrate the effect of malware introduced through communication access points. Investigators have found evidence of malware in systems that control U.S. power transmission grids, wind turbines, nuclear power plants, water treatment plants, and the oil/gas infrastructure [6]. A CPS has sensor and actuator interfaces that can be corrupted by an adversary during an attack. CPS vulnerabilities are magnified when such complex systems are assembled from a large collection of third-party software and hardware.

Cybersecurity encompasses software, hardware, and physical security. Modern SoC platforms blur the distinction between software and hardware boundaries. A CPS may use a mix of microcontrollers, ARM-based SoCs, and FPGAs. The SoCs and FPGAs may incorporate commercial off the shelf (COTS) and third-party intellectual property (IP cores). Third-party cores are seldom trusted, and existing verification methods cannot detect latent malicious behavior of complex IP cores in a timely manner. A hardware RoT serves as the foundation for building trust within a CPS and designing countermeasures that can be effective in the presence of an attack. In a mission-critical CPS, trustworthy operation of the embedded system is paramount. However, it is difficult to quantify the measure of trust enforced by physical, hardware, and software layers of a CPS.

With the integration of third-party IP cores into an embedded system such as a Zynq

SoC, the system vulnerability to hardware Trojan attacks is higher. Formal methods have been investigated to derive assurance built on a software-based RoT [63]. However, moving the RoT to hardware for correctness, security, and trust assurances is difficult [66]. Communication between the IP cores provide considerable insight into their functionality and can be used to detect anomalies during the system operation. This chapter proposes a mixed approach leveraging both hardware and software to secure the communication between the IP cores and monitor the functional behavior of the system to prevent hardware Trojan attacks.

Motivated by the shortcomings in existing efforts, the need for a hardware RoT with anomaly detection capability is used to reduce the primary attack surface arising from the integration of third-party IP cores. Prior work presents the development of a secure architecture called FIDelity Enhancing Security (FIDES) built on the premise of monitoring the communication within the system and its IP cores using tags [106]. Legal communication between IP cores is permitted through a declassification process, which enables the higher tag associated with sensitive IP core to declassify to a lower tag to receive information from a non-sensitive IP core or from an I/O peripheral. However, FIDES does not address the vulnerability when a malicious IP core compromises the integrity of the system during the declassification process. To mitigate this vulnerability, trust enhanced FIDES (TE-FIDES) is proposed, where the IP core's trust measure $Q_i$ is recorded. TE-FIDES is evaluated with respect to its overall latency and throughput by implementing a smart grid CPS application [107] using Pulse Width Modulation (PWM) modules containing third-party soft IP cores. Attacks on these modules can trigger cascading blackouts.

Section 6.2 describes existing approaches to embed trust in a CPS, Section 6.3 focuses on the TE-FIDES approach, Section 6.4 provides an overview of a smart grid application, and Section 6.5 describes the hardware implementation of the TE-FIDES approach with emphasis on the latency and throughput tradeoffs. Finally the conclusions are presented in Section 6.6.

## 6.2 Existing Approaches

A hardware Trojan is defined as a deliberate and malignant modification of a circuit resulting in erroneous operation of the circuit when activated. Since the hardware Trojan is embedded in the lowest level within the device, the threat of the hardware Trojan is severe as long as the device is in operation. A hardware Trojan alters the behavior of an infected Integrated Chip (IC) by thwarting the software or hardware security measures present inside the IC. Any type of IC, including DSPs, ASICs, SoCs, and FPGAs are vulnerable to such malicious modifications. Undesired effects may include the leakage of critical and sensitive information, DoS attacks, and undetected service degradation resulting in system failure. Hardware Trojans can also provide backdoor access to a system, where malicious software works in conjunction with the hardware Trojan to access and disable highly secure systems [15, 16]. Surveys provide hardware Trojan classifications and descriptions [11, 17].

Trust in a CPS may also follow a top-down approach. A networked CPS is ubiquitous in the IoT era, but these connected systems are vulnerable to remote attacks. The DARPA High Assurance Cyber Military Systems (HACMS) program was devised to make such systems resistant to cyber attacks [62, 63]. HACMS is built on the foundation of ensuring high assurance at each layer in the software stack. Formal verification and validation of each stack component seeks to assure functionally correct operation. The seL4 microkernel [65] is an example of an operating system used by the HACMS program with mathematically proven strong security properties including integrity and confidentiality. The lowest stack layer is the underlying hardware, which can be an x86 processor, ARM processor, ASIC or FPGA. Applying formal methods is difficult while integrating COTS components. Quantifying trust through assurance and validating it for components in the complete stack is a complex task. Moving the RoT from hardware into software and firmware is appealing [66], however a firmware-based RoT is vulnerable to MITM attacks as shown in [9]. Distributing the RoT between the software and the processing hardware such as an FPGA can verify the functioning of the firmware with sensitive information firewalled from the code execution

path and contained inside a protected portion of the device.

Intel's Software Guard Extensions (SGX) is an architecture extension based on an *inverse sandbox* mechanism for embedding security in a device [67]. Instead of identifying and isolating malware in a system, the legitimate information is sealed inside an *enclave*. The secure enclave ensures that the sensitive information is protected from malware trying to access it irrespective of its privilege level. The secure enclave is a countermeasure built into the hardware RoT in Apple's iOS [61]. Apple's custom processors such as the A7 and A9 are designed using the ARM architecture and leverage the strong security primitives present in ARM TrustZone [57]. The secure enclave provides an effective approach to deter cyber attacks that may damage the device or steal valuable information. Since the secure enclave also runs a secure microkernel (a version of seL4 [65]), a backdoor within the secure enclave would enable access to critical information as desired by the U.S. government [69].

## 6.3 Trust Enhanced FIDES (TE-FIDES) Overview

FIDelity Enhancing Security (FIDES), named after the Greek Goddess of Trust, provides a distributed enclave-like secure methodology for enhancing trust in applications mapped to FPGAs [106]. Trust Enhanced FIDES (TE-FIDES) defines and quantifies each IP core's trust metric during the declassification process. TE-FIDES specifically addresses the integrity vulnerability posed by the FIDES architecture. This chapter considers the third-party soft IP cores as the primary attack surface for Trojans with the potential to cripple the system.

TE-FIDES relies on tag-based information flow control to implement decentralized and safe flow of information between various elements in a system. During the initialization phase, each IP core $<Core>_i$ has its Trusted Wrapper $W_i$ assigned a tag $\tau_i$ corresponding to the characteristics ($\alpha$) of the information accessed/processed by the $<Core>_i$. $\alpha = (\mathcal{S}, \mathcal{N}, \mathcal{E})$, where $\mathcal{S}, \mathcal{N}$ and $\mathcal{E}$ denotes sensitive, non-sensitive, and empty tokens, respectively. Tag initialization is described in detail previously [106]. The I/O peripherals with access to the

external world are labeled with empty tokens ($\mathcal{E}$). Empty tokens have the lowest priority and the tagged information flow enforces an IP core to hold an $\mathcal{E}$ token before transmitting information through the peripherals.

To facilitate legal communication between IP cores and peripherals as defined by the core's security policy, tags can be declassified (denoted by $\downarrow$). The *tag declassification* rule enables the higher tag to declassify to the lower tag to receive information. In addition to the declassification rule, the trust metric for an IP core $i$ is defined as $Q_i$. $Q_i$ provides the snapshot of trust embedded while integrating the IP core with the Trusted Wrapper $W_i$. Similar to a secure enclave, the value of $Q_i$ can only be read by the Trusted Anchor $\mathcal{TA}$ and cannot be updated by an external event. $Q_i$ is initialized along with the Trusted Wrapper $W_i$ for each IP core $i$. $Q_i$ is initialized to 1. During declassification, the original tags associated with an IP core $i$ are changed, which is captured using $Q_i$ and recorded in the logging buffer $\mathcal{B}$. In order to understand the diagnostic information logged and to differentiate whether the tag declassification was for IP core information exchange or for transmitting through I/O peripherals, $Q_i$ is decremented by a constant value corresponding to the non-sensitive $\mathcal{N}$ and empty $\mathcal{E}$ tokens, respectively.

## 6.4 IoT-based Smart Grid Application

High Voltage Direct Current (HVDC) is more prevalent with the increasing trend in renewable energy production. Wind or solar farms require HVDC transmission combined with compact and reliable converter technology to meet the large power requirements. Modular multilevel power converters provide an easy alternative for voltage conversions and are deployed in such installations [108]. The fieldbus communication protocols (e.g. RS-485, RS-232, DeviceNet, CANopen, Profibus) in such installations are being replaced by real-time Ethernet protocols, similar to trends in other IoT domains. Ethernet provides increased bandwidth and better performance compared to the traditional fieldbus. To achieve the

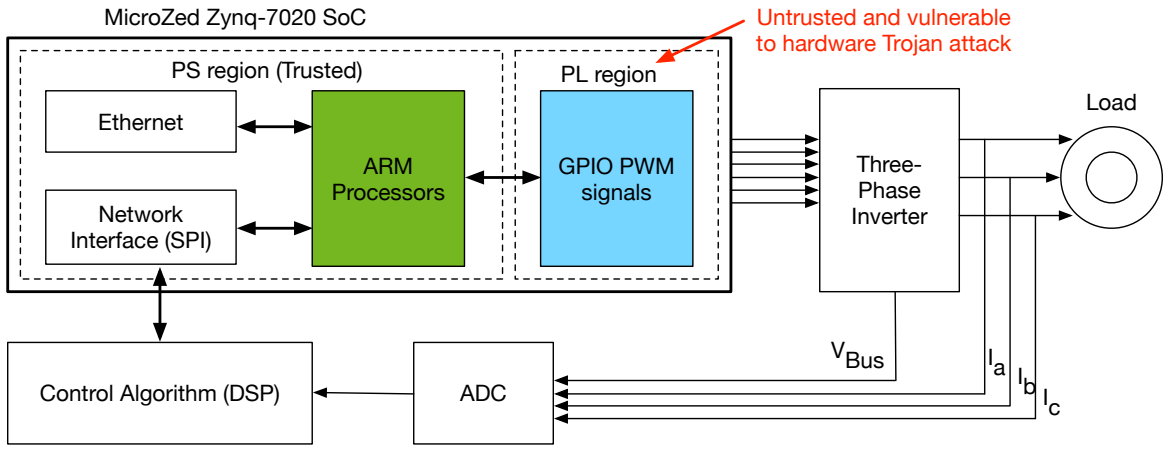


Figure 6.1: Smart grid application

low latency and real-time determinism required for such applications, FPGAs provide an implementation platform with multiple Ethernet Media Access Control (MAC) options and custom I/O to drive the multiple PWM outputs.

Figure 6.1 shows the control interface for driving the three-phase inverter connected to a load. The interface is realized on a MicroZed development board containing a Xilinx Zynq-7020 possessing a dual-core ARM processor and an Artix FPGA with 85K logic cells. The Zynq platform is partitioned into processing system (PS) and programmable logic (PL) sections. The PS region includes peripheral controllers accessible to the ARM cores, whereas the PL region's FPGA fabric can be used to implement custom computational and controller cores. The Ethernet and Serial Peripheral Interface (SPI) controller cores are trusted, fixed silicon cores implemented in the PS region. The control algorithm is mapped to the DSP, which samples the currents and voltages using built-in or discrete Analog to Digital Converters (ADC). Commercial state-of-the-art DSPs have dedicated hardware for vector control of three-phase systems [109] and therefore used in this example. Untrusted PWM blocks are instantiated in the PL. The Zynq receives system commands and parameters from a remote supervisory unit through the Ethernet controller, while the duty cycles for synthesizing the voltages are received through the SPI module from the DSP. The PWM module generates

waveforms with the appropriate duty cycle to drive the three-phase inverter. In this example, the presence of a hardware Trojan in the PWM module can provide incorrect PWM signals to the inverter switches resulting in a short or open circuit and eventually disrupt power flow.

## 6.5 Implementation and Results

The TE-FIDES architecture is evaluated by mapping the IoT module and benchmarking it for latency and overall throughput. Both the baseline and the TE-FIDES implementations are synthesized and implemented using Xilinx Vivado version 2015.4. Vivado reports the logic resource utilization of the complete design and also provides a definition file that is mapped using the Xilinx SDK to execute the software algorithms. The IP cores are wrapped with the AXI4-Lite interface by Vivado's IP Packager tool. Data transfers are performed using custom register interfaces. These memory-mapped registers are configured as write registers for input ports and read registers for the output ports of the IP core. Each of the IP core's I/O ports is assigned a tag $\tau_i$, which is mapped to a register initialized with the AXI wrapper. $Q_i$ captures the changes to the tags during the declassification process which is recorded in $\mathcal{B}$. The Trusted Wrapper implementation increases the logic resource utilization by only 1-2%.

Procedure 5 shows the steps required to receive the data from the Ethernet peripheral connected to the PS and transfer the commands to the PWM module instantiated in the PL. The PWM duty cycle received from the DSP is monitored by the Trusted Wrapper $W_i$. In the event a malicious Trojan inside the PWM module decides to change the duty cycle internally and damage the inverter, $W_i$ detects the anomaly between the specified and the issued duty cycle parameter. As a result, $W_i$ prevents the output of the PWM module from issuing the corrupted duty cycle parameter, thus preserving the state of the inverter. In addition, the IP core deviation due to the presence of an attack is also logged in $\mathcal{B}$.

**Algorithm 5:** Communication flow between PWM IP core and Ethernet peripheral

1: $<Core>_i^s \rightarrow$ Trusted Wrapper $W_i$ $[(Q_i = 1) \rightarrow \mathcal{B}]$

2: Trusted Wrapper $W_i \nrightarrow$ I/O peripheral $\mathbb{P}$

3: Request for declassification sent to $\mathcal{TA}$ by $W_i$

4: $\mathcal{TA}$ changes the privileges of $W_i$ from $\mathcal{S} \downarrow \mathcal{E}$ if policy $\Psi$ allows

5: $(Q_i = Q_i - x_\mathcal{E}) \rightarrow \mathcal{B}$, where $x_\mathcal{E}$ is the constant value corresponding to the empty token $\mathcal{E}$

6: Trusted Wrapper $W_i \rightarrow$ I/O peripheral $\mathbb{P}$

7: $\mathcal{TA}$ increases the privileges of $W_i$ from $\mathcal{E} \uparrow \mathcal{S}$ on completion and $(Q_i = 1) \rightarrow \mathcal{B}$

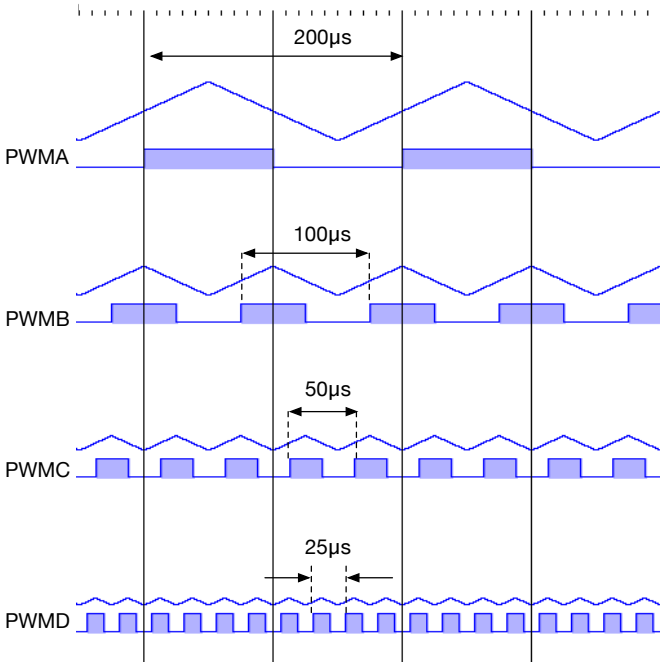


Figure 6.2: PWM waveforms with varying duty cycles generated by the Zynq SoC

Figure 6.2 shows the waveforms generated by the PWM modules ranging from 5-40 KHz instantiated in the PL section of the Zynq SoC. The latency calculation includes the clock cycles for transferring the data from the ARM core to the PWM modules. Worst case latency occurs when all the inputs and outputs of the PWM IP core are considered as sensitive requiring multiple declassification calls. The latency of the TE-FIDES approach (worst case) is observed to be 15% slower than the non TE-FIDES approach for a single PWM core as shown in Table 6.1. The execution clock cycles are measured to determine the latency and throughput comparisons, where fewer clock cycles indicate better performance. The throughput of the application after adding 4 PWM IP cores is only 8% slower than the no TE-FIDES approach.

Table 6.1: Latency and throughput comparisons

| **IP core** | **No TE-FIDES** | **TE-FIDES (normal operation)** | **TE-FIDES (worst case)** |
|---|---|---|---|
| 1 PWM | 2205 | 2330 | 2525 |
| 4 PWMs | 9540 | 10320 | 10335 |

## 6.6 Conclusions

This chapter presents the trust enhancements required for enforcing trust in a secure architecture when integrating third-party soft IP cores in an FPGA. Existing security measures target either the software or the hardware component of a CPS. In contrast, TE-FIDES creates a distributed trust embedding with bindings in both hardware and software. A distributed monitoring of the communication framework uses embedded components to detect and flag a hardware Trojan attack. In addition, the trust metric of each IP core is quantified and snapshots of each IP core's trust metric are recorded in a logging buffer for diagnostics.

In future work, these snapshots will provide the foundation for machine learning algorithms to build temporal models that can detect anomalies with lower latency. TE-FIDES latency and throughput tradeoffs are examined by mapping a smart grid application to a Zynq-7020 SoC. Resource utilization increased by only 1-2% over a baseline implementation, whereas latency and throughput increased by 15% and 8% respectively.

# Chapter 7

# Architectural Refinements for Enhancing Trust and Securing Cyber-Physical Systems

## 7.1 Introduction

A cyber-physical system primarily defines the interaction of the physical world with a system built with components drawn from the computational, communication and control system domains. Existing security measures either target the sensor hardware or the control system software component to counter cyber attacks. In addition, the current set of security requirements are inadequate when transitioning from legacy electro-mechanical control systems to current smart large scale industrial IoT control systems. This chapter focuses on the architectural enhancements for distributing the root-of-trust between both hardware and software components. This distribution of trust makes it difficult for a hardware-only or software-only attack to cripple the system. The proposed approach is applied to a motor control algorithm widely used in industrial IoT systems, consisting of third-party soft IP

cores, where an attack on such a system can cripple it. The resulting implementation is evaluated by subjecting the system to multiple hardware Trojan attacks. In all the cases, the proposed approach proves to be resilient to the cyber attacks.

A CPS is an evolving paradigm that captures the physical interactions in the real world and translates it to control, computation and communication tasks in information systems to achieve stability, robustness and efficiency. Although CPS lacks a formal definition and can be interpreted in multiple ways from either an embedded systems or control systems perspective, the essential CPS components as shown in Figure 7.1 consist of: sensors that connect to the physical world; controllers that translate the physical inputs for the system domain; and actuators that provide the feedback from the controller to the physical world. However, in the embedded systems domain, a CPS is constructed from a collection of third-party software and hardware. Hence, the metrics for evaluating the security and privacy of a CPS are different. The performance goals can be quantified as hardware/software/network reliability and availability, system security and resiliency to attacks, whereas the performance metrics consist of bandwidth, throughput, latency, and power consumption.

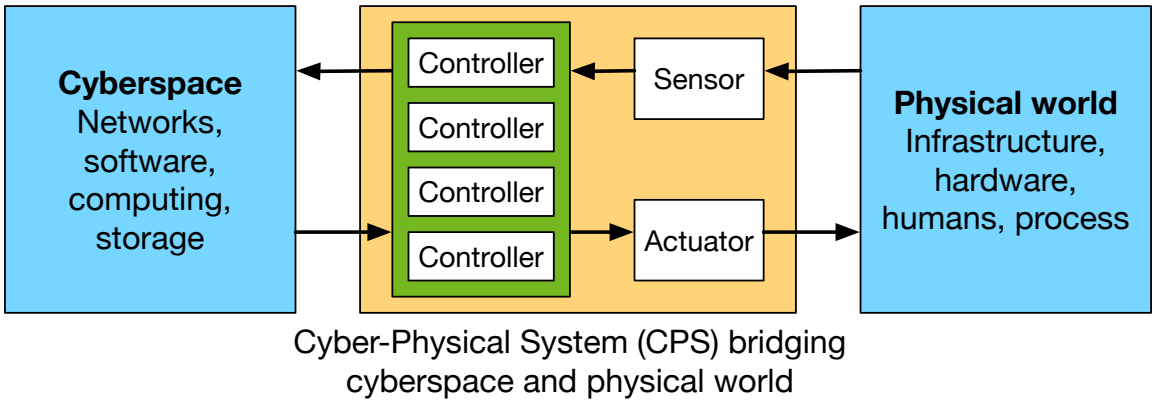


Figure 7.1: Cyber-Physical System (CPS)

The Internet of Things (IoT) represents a sophisticated interconnected CPS ubiquitous in smart grids, smart homes, autonomous vehicles/drones, energy management systems, etc. As IoT systems get integrated into critical infrastructure, a security breach in these systems

can have a catastrophic effect. Examples such as the vulnerabilities in Internet-connected cars allowing hackers to control the car [9] or the Lockheed drone brought down by jamming its communication network and spoofing its GPS sensors [8] have demonstrated the attacks induced through the physical world via sensors. Investigators have also found evidence of security and privacy attacks in IoT-based systems integrated into power transmission grids, wind turbines, nuclear power plants, water treatment plants, and the oil/gas infrastructure [6]. IoT security and privacy is an upcoming research area with studies focusing on the potential threats and effective countermeasures against them. Since the sensor hardware serves as the primary source of interaction with the physical world, a hardware root-of-trust (RoT) is a prerequisite for building trust in an IoT system. However, security requirements are vague when trying to enforce trust in the physical, hardware, and software layers of such an IoT-based CPS. The integration of third-party intellectual property (IP) cores to interface with the sensors is considered common practice when building the IoT platform and this renders the CPS vulnerable to hardware Trojan attacks.

**Motivation and Contributions:** The cybersecurity assessment of the IoT-based CPS reveal the shortcomings in the design and integration of third-party soft IP cores when such systems are vulnerable to HTTs. Prior work presents the security analysis and countermeasures for detecting HTTs in a hardware-based RoT [100]. FIDelity Enhancing Security (FIDES) builds on the lessons learned from the HTTs and proposes a distributed scheme that monitors the communication between the IP cores using tags [106]. This chapter demonstrates the threats affecting a CPS consisting of third-party soft IP cores and the countermeasures deployed to mitigate attacks from HTTs. Specifically, secure enclave-like hardware extensions embedded in the IP core wrapper enforce trust in the CPS by alerting the supervisory module to deviations from nominal behavior. Trust enhanced FIDES (TE-FIDES) is implemented on a SoC and in subjected to a variety of hardware Trojan attacks. In all cases, TE-FIDES proves to be resilient to the attacks. In addition, the performance costs for resilient and trustworthy operation of the CPS are evaluated. The main objective of this chapter is to provide a comprehensive overview of an IoT-based CPS platform with empha-

sis on the security requirements and trust enhancements needed to build a robust hardware RoT.

**Roadmap:** Section 7.2 describes the need for a comprehensive list of requirements to assess the security of a CPS along with the overview of the existing hardware Trojan model and related work. Section 7.3 focuses on the TE-FIDES approach and the resulting security analysis. Section 7.4 provides an overview of an industrial IoT-based CPS application mapped to a FPGA platform. Section 7.5 evaluates the TE-FIDES implementation when subjected to multiple hardware Trojan attacks and finally the conclusions are presented in Section 7.6.

## 7.2 Background

IoT-based systems are used in a variety of applications ranging from industrial management, prognostic health monitoring for engines or heavy machinery. These applications generate huge amount of data providing critical and sensitive information gathered from the physical world to a controller with advanced analytics. Resiliency against cyber attacks for such an IoT-based CPS that monitors, controls and protects critical infrastructure is the top priority research and development area by the U.S. President's Council of Advisors on Science and Technology for 2017 (http://www.nitrd.gov). The Cyber Security Information Assurance (CSIA) component focuses on the detection, prevention, and recovery of a CPS from cyber attacks that can compromise such a system.

### 7.2.1 Defining the Security Requirements

Traditionally, the security requirements of a CPS are adapted from the CIA-triad of confidentiality, integrity, and availability. Confidentiality enforces rules to prevent sensitive information being accessed by unauthorized users. Integrity demands that the received com-

mands/information are legitimate and are not compromised. Availability is essential for a system to provide uninterrupted service, especially when subject to a cyber attack. Various methods of approaching CPS security have been documented [47, 48], which meets the CIA-triad requirements in the presence of an attack. However, the CIA-triad is insufficient to address the security requirements in a IoT-based CPS consisting of a variety of sensors continuously interacting with the physical world [110]. As a result, a comprehensive list derived from information, assurance, and security requirements (IAS-octave) is proposed as an extension to the CIA-triad. Table 7.1 provides the security requirements in the IAS-octave along with their definitions. These requirements serve as the reference while defining the security properties of the IoT-based CPS application in this chapter.

Table 7.1: Comprehensive list of security requirements

| **Requirement** | **Definition** |
|---|---|
| Confidentiality | Prevents unauthorized users from accessing sensitive information |
| Integrity | Ensures completeness, accuracy, and absence of unauthorized modifications to received information |
| Availability | Ensures that all system components are available and operational when requested by authorized user |
| Trustworthiness | Property of system to verify the identity and establish trust in a third party |
| Privacy | Ensures that the privacy of the sensitive information is maintained |
| Non-repudiation | Proves the occurrence/non-occurrence of an event |
| Accountability | Tracks and holds users responsible for their actions |
| Auditability | Monitors all actions and events within a system |

### 7.2.2 Hardware Trojan Threat Model

Hardware Trojans are malignant modifications of a circuit resulting in faulty or undesired operation of the infected system when activated. Once a hardware Trojan is inserted into a system at the hardware level, it remains quiescent until activated for the lifecycle of the device. The HTT model dictates the use of tamper-resistant hardware (third-party soft IP cores) which is difficult to ensure during the integration of unauthenticated IP cores using untrusted tools and unmonitored fabrication/assembly foundries. Hardware Trojans have been deployed for disabling highly secure systems through the use of backdoors in conjunction with malicious software [15, 16]. Surveys have also focused on the classification and description of hardware Trojans [11, 17]. Hardware Trojans can cause a variety of undesired effects such as leakage of critical and sensitive information, Denial of Service (DoS) attacks, and undetected service degradation resulting in system failure.

Research efforts have focused on enforcing secure operation of a system in the presence of HTTs [12]. In some cases, the HTTs can be prevented from activation [35], while techniques to secure a Field Programmable Gate Array (FPGA) using authenticated bitstreams [50] and reconfigurable logic barriers [51] have been studied. In these countermeasures no emphasis is given to identify the malicious IP cores that may initially conform to the system's operating policies but deviate from the normal execution behavior.

### 7.2.3 CPS Security and Trust

A CPS is considered secure from attacks when anomalies are detected and countermeasures deployed ahead of the attack causing critical damage. However, most of the research efforts in this area focus on the controller shown in Figure 7.1. Anomaly detection and prevention using backup controllers have been studied [55], whereas efforts on resilient control systems when subjected to DoS attacks have also been investigated [56]. The DoS attack interrupts the communication between the sensors and the control system, and may also affect system

integrity by modifying sensor data. Since the sensors are frequently the easiest source of vulnerability, watermarking of the sensor signal may be used to establish an acceptable threshold for noise embedded in the system [111] to detect the presence of an attack. However, the threshold for the noise can be learned by a clever attacker using multiple simulations and can help mask the attack. Cryptographic algorithms do not secure a system as certificates can be stolen or keys can be extracted using side-channel techniques. The classic example of the Stuxnet attack highlights the incidence of malware being injected surreptitiously even when the CPS is physically isolated [3].

Trust is typically not considered as a metric to evaluate the security of a CPS. Efforts focusing on detection of malware and protection of controller does not ensure trust in sensor and actuator communication that is critical in bridging the cyber and physical domains in a CPS. Trust needs to be embedded inside the CPS and maintained through analysis of the communication within the CPS. At the hardware architecture level, safeguards need to be built to monitor the communication between third party IP cores and alert the system in the case of anomalies. Therefore, the acceptable cyber costs for evaluating safeguards implemented in the hardware platform may consist of data packet inspection, supervisory monitoring schemes, and behavioral policies as compared to the economic and human costs of the physical system's degradation or destruction.

### 7.2.4 HACMS and Secure Enclaves

Ensuring correctness, safety, and security in a CPS is a difficult task and is being addressed partially by the Defense Advanced Research Projects Agency (DARPA) High Assurance Cyber Military Systems (HACMS) program [62, 63]. HACMS components are constructed with assurances embedded from top to bottom layer of the software stack. HACMS has been constructed using a variety of development environments: (i) code synthesis, (ii) domain-specific languages, and (iii) interactive theorem provers. The software components consist of hypervisors, microkernels, file systems, control algorithms, and data loggers. Figure 7.2

shows the construction of the HACMS components with assurances embedded from top to bottom. The lowest stack layer is the underlying hardware, which can be an x86 processor, ARM processor, ASIC or FPGA and is not assured by HACMS. HACMS highlights the case of RoT built into the firmware and the software stack, which has been deemed vulnerable [9]. Distribution of a synchronized RoT between the software and the processing hardware to alert the system or end user in the event of an attack is more appealing.

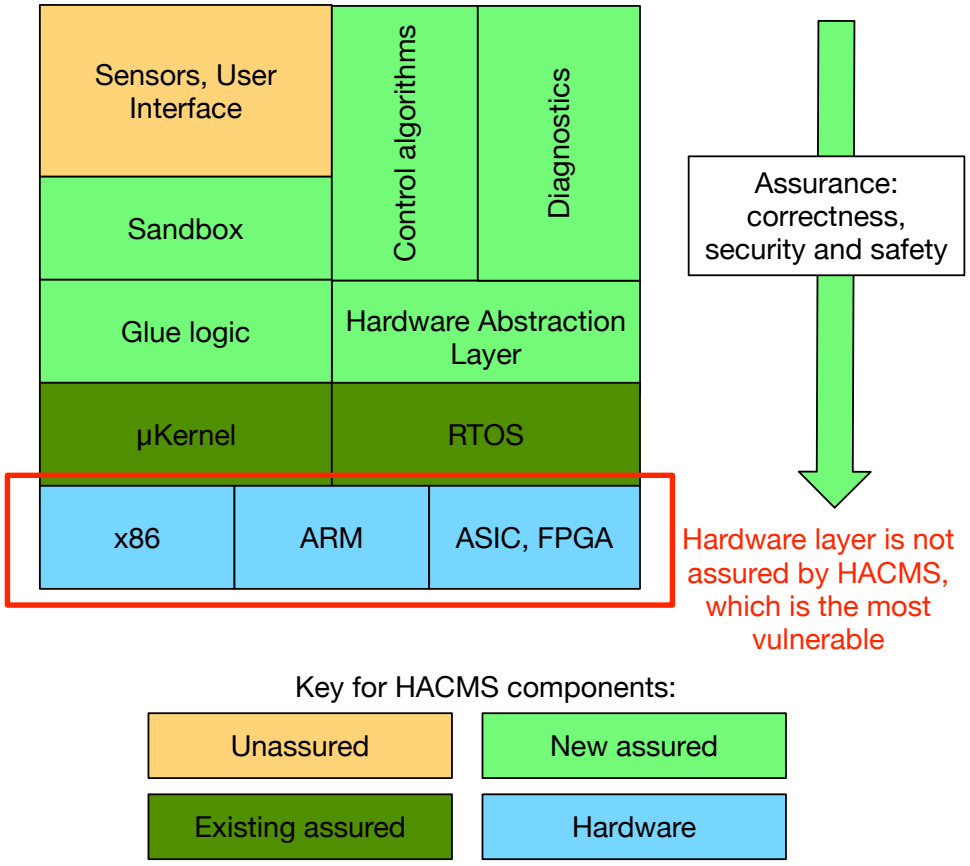


Figure 7.2: HACMS components with its assurance metrics

Intel's Software Guard Extensions (SGX) utilizes an *inverse sandbox* mechanism for embedding security in a device [67]. The legitimate and sensitive information is sealed inside an *enclave* as shown in Figure 7.3. The secure enclave ensures that the sensitive information is protected in the event of an attack. SGX is an extension of the Trusted Platform

Module (TPM) idea, where application execution is attested by the platform. SGX provides privileges to authenticate the application's trusted code and this assessment can be compared with the sensitive information in the enclave to verify the correct initialization of the application. Intel SGX is still nascent in terms of identifying the resources that can be shared between the trusted and untrusted applications. In addition, Intel SGX has also been proved to be vulnerable due to cache timing attacks [112]. Apple's iOS relies on a secure enclave-based countermeasure built into the hardware RoT to deter cyber attacks [61]. Since the secure enclave stores sensitive information, the U.S. government's request to insert a backdoor within the secure enclave would provide it access to the sensitive information [69].

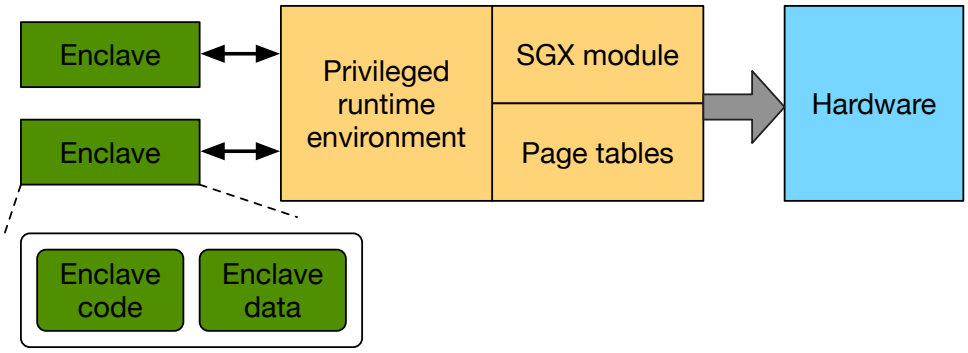


Figure 7.3: Intel SGX environment

## 7.3 Trust Enhanced FIDES (TE-FIDES) Overview

FIDelity Enhancing Security (FIDES), named after the Greek Goddess of Trust, provides a distributed enclave-like secure methodology for enhancing trust in applications mapped to FPGAs [106]. Trust Enhanced FIDES (TE-FIDES) consists of architectural enhancements to address the IAS-octave list of security requirements for securing and building trust in a CPS. TE-FIDES is tailored to specifically examine the vulnerabilities introduced through the addition of third-party soft IP cores and provide countermeasures through a combination

of its distributed monitoring scheme and access policies.

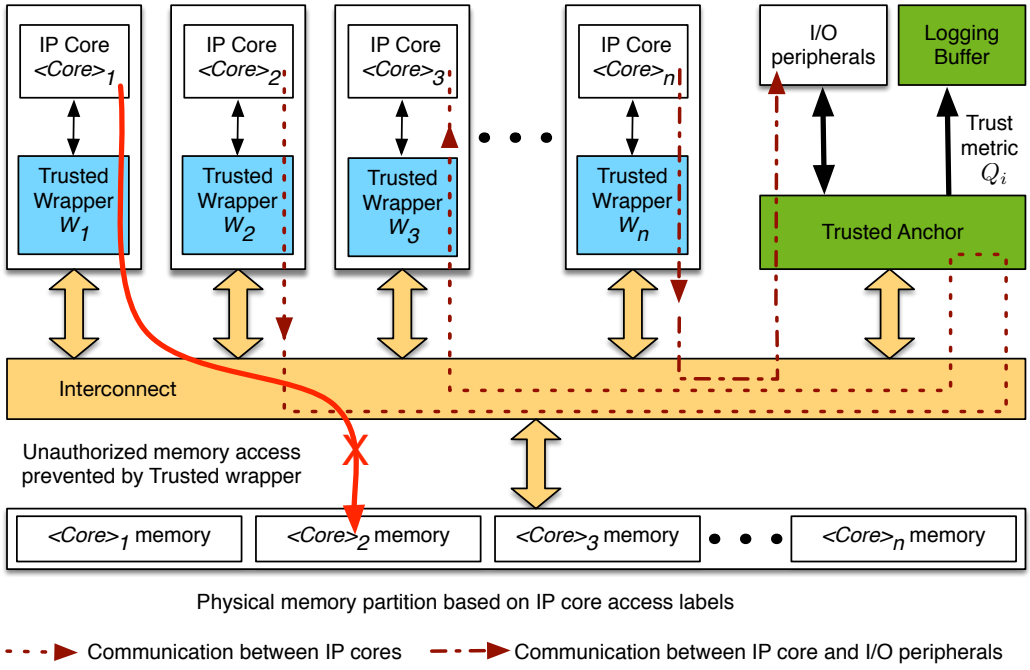


Figure 7.4: The TE-FIDES architecture components

### 7.3.1 Architecture

An embedded system consists of multiple IP cores providing various functions. Based on their functionality, the IP cores can be classified as critical or non-critical depending on the operation of the system. This chapter considers the third-party soft IP cores as the primary attack surface for Trojans with the potential to cripple the system. Hence, the IP cores are assigned sensitivity levels as follows: (a) an IP core that can access and process critical and sensitive information is a *sensitive core* ($<Core>_i^s$); (b) an IP core capable of accessing and processing non-critical information is classified as a *non-sensitive core* ($<Core>_i^{ns}$). Hardware Trojan attacks are assisted when sensitive and non-sensitive information is not segregated when sharing shared resources and I/O peripherals. Attacks such as sensitive information leakage through covert channels [22, 23], or transmitting information through a

normal channel by mixing it with noise [21] may be prevented by identifying and segregating the critical and non-critical information.

The TE-FIDES architecture consists of the following components generated by the designer during the third-party soft IP core integration phase on the FPGA:

(a) *Trusted Anchor* ($\mathcal{TA}$) - centralized security officer that supervises the communication between the IP cores and the external world (I/O peripherals) as per the predefined security policy $\Psi$.

(b) *Trusted Wrapper* ($W_i$) - wrapper embedding for each IP core $<Core>_i$ that monitors the communication of $<Core>_i$ with any other core $<Core>_j$, where $(j \neq i)$. The Trusted Wrapper emulates a secure enclave, where it provides only unidirectional access for the tags and the trust metric.

(c) *Tagged Physical Memory* ($\mathcal{M}^{\mathcal{S}}, \mathcal{M}^{\mathcal{N}}$) - segregated memory regions with *tags* corresponding to the sensitivity level of the information stored.

(d) *Logging Buffer* ($\mathcal{B}$) - logs all IP core deviation in the presence of an attack and also saves the snapshots of the trust metric $Q_i$ of each IP core $i$ embedded in the system.

Figure 7.4 shows the TE-FIDES architecture with the components described above and highlights the communication flow from IP core to IP core and from IP core to I/O peripherals.

### 7.3.2 Security Analysis

The security requirements to enforce trust in TE-FIDES are mapped to the IAS-octave list in Table 7.1 and are defined as follows:

SR1 Sensitive information cannot be accessed or processed by a *non-sensitive core* $<Core>_i^{ns}$. (IAS-octave requirements: confidentiality, privacy)

SR2 IP cores are not allowed to directly access shared resources or I/O peripherals to limit export of data. (IAS-octave requirements: integrity, trustworthiness)

SR3 The information exchange between $<Core>_i^{ns}$ and *sensitive core* $<Core>_i^{s}$ is either restricted or can be allowed in the presence of a security officer. (IAS-octave requirements: availability, accountability)

SR4 Each core $<Core>_i$ will provide a suite of measurements based on its statistical properties such as memory utilization and latency. These properties will be monitored to detect behavior deviations in the presence of a HTT. (IAS-octave requirements: auditability, non-repudiation)

TE-FIDES depends on tag-based distributed information flow control for mapping the decentralized and safe flow of information between various elements in the system. Each IP core $<Core>_i$ has its Trusted Wrapper $W_i$ assigned a tag $\tau_i$ depending on the privilege level of the information accessed by the IP core. The privilege level $\alpha = (\mathcal{S}, \mathcal{N}, \mathcal{E})$ which corresponds to sensitive ($\mathcal{S}$), non-sensitive ($\mathcal{N}$) and empty ($\mathcal{E}$) tokens respectively. The security policy $\Pi_i$ captures the authorized set of communication for each IP core $i$ such that an IP core $i$ with tag $\tau_i =< \alpha_i, i >$ can communicate with another IP core $j$ with tag $\tau_j =< \alpha_j, j >$ only if $(i, j) \in \Pi_i$. Empty tokens have the lowest priority and the tagged information flow enforces an IP core to hold an $\mathcal{E}$ token before transmitting information through the I/O peripherals. If the security policy allows legal communication between IP cores or peripherals, the IP core's higher privilege tags can be declassified (denoted by $\downarrow$). The *tag declassification* rule enables the higher tag to declassify to the lower tag to receive information.

Since a system can consist of multiple IP cores communicating, declassification may occur for multiple IP cores at the same time. To satisfy the IAS-octave requirements of accountability and auditability, the state of each IP core is defined using a trust metric $Q$. The Trusted Wrapper $W$ keeps track of $Q$ and, similar to a secure enclave, the value of $Q$ can only be read by $\mathcal{TA}$. This property prevents $Q$ from being updated by an external event and provides auditability for the IP core. During the declassification process the value of $Q$

**Algorithm 6:** Communication flow between critical and non-critical IP cores

1: $<Core>_i^{ns} \rightarrow$ Trusted Wrapper $W_i$ $[(Q_i = 1) \rightarrow \mathcal{B}]$
2: $<Core>_j^{s} \rightarrow$ Trusted Wrapper $W_j$ $[(Q_j = 1) \rightarrow \mathcal{B}]$
3: Trusted Wrapper $W_i \nrightarrow$ Trusted Wrapper $W_j$
4: Request for declassification sent to $\mathcal{TA}$ by $W_j$ $[(Q_j = 1) \rightarrow \mathcal{B}]$
5: $\mathcal{TA}$ changes the privileges of $W_j$ from $\mathcal{S} \downarrow \mathcal{N}$ if policy $\Psi$ allows
6: $(Q_j = Q_j - x_{\mathcal{N}}) \rightarrow \mathcal{B}$, where $x_{\mathcal{N}}$ is the constant value corresponding to the non-sensitive token $\mathcal{N}$
7: Trusted Wrapper $W_i \rightarrow$ Trusted Wrapper $W_j$
8: Trusted Wrapper $W_j \rightarrow <Core>_j^{s}$
9: $\mathcal{TA}$ increases the privileges of $W_j$ from $\mathcal{N} \uparrow \mathcal{S}$ on completion and $(Q_j = 1) \rightarrow \mathcal{B}$

is modified depending on the change in the original privilege level. $Q$ is decremented by an offset value that either corresponds to the $\mathcal{N}$ or $\mathcal{E}$ token. To maintain accountability, the change in each IP core's $Q$ value is recorded in the logging buffer $\mathcal{B}$.

Procedure 6 outline the steps for transferring information between the IP cores, whereas Procedure 7 describes the information exchange between an IP core and an I/O peripheral.

**Algorithm 7:** Communication flow between critical IP core and I/O peripheral

1: $<Core>_i^{s} \rightarrow$ Trusted Wrapper $W_i$ $[(Q_i = 1) \rightarrow \mathcal{B}]$
2: Trusted Wrapper $W_i \nrightarrow$ I/O peripheral $\mathbb{P}$
3: Request for declassification sent to $\mathcal{TA}$ by $W_j$
4: $\mathcal{TA}$ changes the privileges of $W_j$ from $\mathcal{S} \downarrow \mathcal{E}$ if policy $\Psi$ allows
5: $(Q_i = Q_i - x_{\mathcal{E}}) \rightarrow \mathcal{B}$, where $x_{\mathcal{E}}$ is the constant value corresponding to the empty token $\mathcal{E}$
6: Trusted Wrapper $W_i \rightarrow$ I/O peripheral $\mathbb{P}$
7: $\mathcal{TA}$ increases the privileges of $W_i$ from $\mathcal{E} \uparrow \mathcal{S}$ on completion and $(Q_i = 1) \rightarrow \mathcal{B}$

## 7.4 Field Oriented Control Application

Field oriented control (FOC), also known as vector control, is used in industrial servo motors requiring precise torque control [109]. FOC helps to reduce the torque ripple and electromagnetic interference. Industrial IoT systems, high density power systems for aircraft applications, and high voltage direct current transmission for power grid applications utilize FOC to reduce motor size, cost, and power consumption. The FOC algorithm shown in Figure 7.5 for a typical motor control application consists of the following steps:

1. The three-phase feedback currents are converted to the two axis orthogonal reference currents by the Clarke transform module. The Clarke transform outputs along with the rotor position from the encoder are fed to the Park transform module resulting in the direct and the quadrature current components.

2. The current components are provided as inputs to two proportional-integral-derivative (PID) controllers running in parallel to limit the direct current to zero and achieve the desired torque using the quadrature current.

3. The voltage outputs of the PID controllers are converted back to the three-phase reference frame by the inverse Park and Clarke transform modules.

4. The output of the inverse Clarke transform module provides the duty cycle of the Pulse Width Modulation (PWM) channels that corresponds to the three-phase voltages. The resulting voltages are used to drive the thee phase inverter.

Thus, the FOC algorithm reduces power consumption by aligning the vector of the motor phase currents to produce the required torque. The reference currents and system parameters are typically transmitted by a remote supervisory controller using fieldbus communication protocols (e.g. RS-485, RS-232, DeviceNet, CANopen, Profibus). However, such communication schemes are being replaced by real-time Ethernet protocols, similar to trends in other IoT domains. Ethernet provides increased bandwidth and better performance compared to

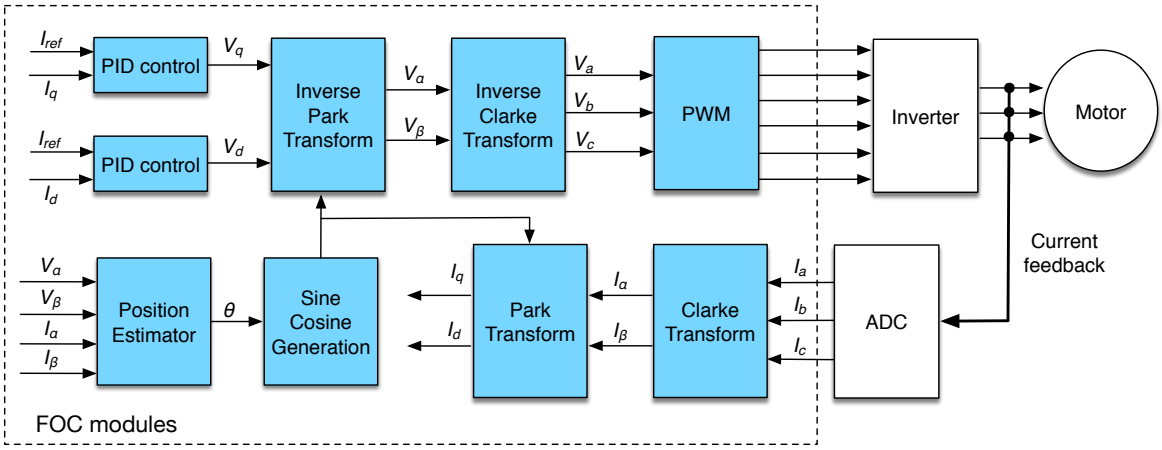


Figure 7.5: Field Oriented Control algorithm for motor control

the traditional fieldbus protocols. FPGAs provide a suitable platform with multiple Ethernet Media Access Control (MAC) options and custom I/O to implement a FOC algorithm with low latency.

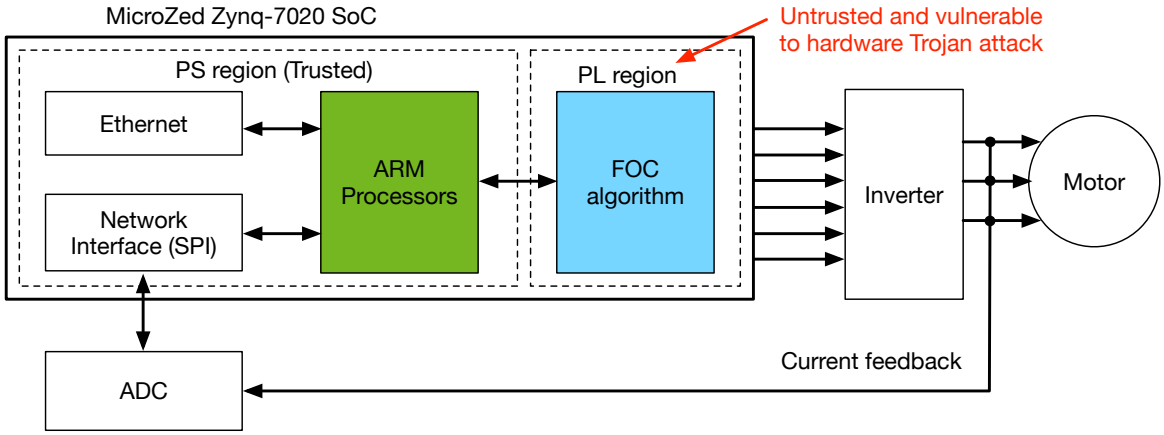


Figure 7.6: FOC implementation on a Zynq SoC

The FOC algorithm is implemented on a MicroZed development board containing a Xilinx Zynq-7020 possessing a dual-core ARM processor and an Artix FPGA with 85K logic cells. The Zynq platform is partitioned into processing system (PS) and programmable logic (PL) sections. The PS region includes peripheral controllers accessible to the ARM cores, whereas

the PL region's FPGA fabric can be used to implement custom computational and controller cores. Figure 7.6 shows the implementation of the FOC algorithm implemented on the Zynq. The Ethernet and Serial Peripheral Interface (SPI) controller cores are trusted, fixed silicon cores implemented in the PS region. The FOC algorithm consists of untrusted third-party soft IP cores are mapped to the PL region.

## 7.5 TE-FIDES Evaluation

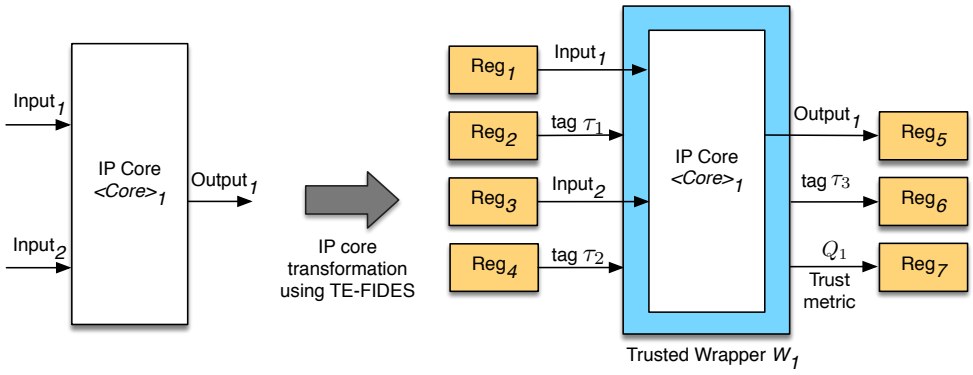


Figure 7.7: Trusted Wrapper initialization for IP core

The FOC algorithm blocks shown in Figure 7.5 are implemented using the TE-FIDES architecture on the PL fabric. The TE-FIDES implementation is synthesized and implemented using Xilinx Vivado version 2015.4. Each IP core is wrapped with the AXI4-Lite interface using Vivado's IP Packager tool. Data transfers are performed with custom register interfaces. These memory-mapped registers are configured as write registers for input ports and read registers for the output ports of the IP core. Figure 7.7 shows the Trusted Wrapper initialization. Each of the IP core's I/O ports is assigned a tag $\tau_i$, which is mapped to a register initialized with the AXI wrapper. $Q_i$ captures the changes to the tags during the declassification process which is recorded in $\mathcal{B}$. Vivado reports the logic resource utilization of the complete design and shows that the Trusted Wrapper implementation only increases the logic resource utilization by 1-2%. The latency of the TE-FIDES approach (worst-case) is

observed to be 15% slower than the non TE-FIDES approach as shown in Figure 7.8, whereas the throughput suffers by 8%. The latency and throughput overheads would normally be considered acceptable for making a system trustworthy and resilient.

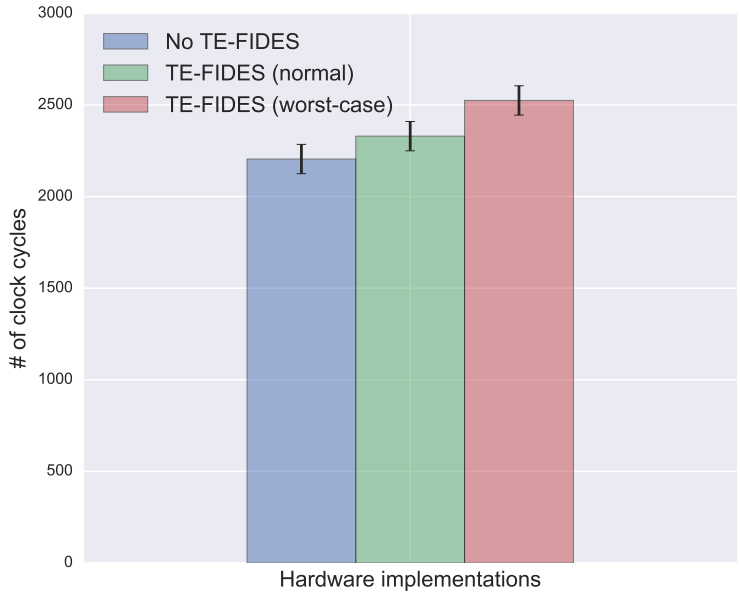


Figure 7.8: Latency comparison for non TE-FIDES and TE-FIDES implementations of the PWM IP core

The TE-FIDES architecture is evaluated for robustness by subjecting the FOC algorithm implementation to the following HTTs: (i) leakage of sensitive data, (ii) message spoofing, and (iii) DoS. The randomness of the HTT is tested by triggering the hardware Trojan on the occurrence of an event, e.g. counter value, specific input value, etc. The PWM core is implemented with all three HTTs and triggered randomly. An internal state counter is used to identify the type of attack being demonstrated by the IP core.

### 7.5.1 Leakage of Sensitive Data

An *Always On* HTT leaks sensitive information through any of the I/O peripherals connected to the system. This specific type of HTT does not require activation. The leakage point de-

termines the attacker model, as the attacker needs to recover the sensitive data transmitted. The PWM core in Figure 7.5 is designed to leak the duty cycle through the SPI interface at random. Since all the IP cores have a Trusted Wrapper interface, random transfer of data through the IP core's output port is not possible. For transferring the data to an I/O peripheral, the IP core's privileges have to be downgraded from its inherent higher sensitive privilege level. The $\mathcal{TA}$ checks the pre-defined security policy $\psi$ and detects an anomaly in the declassification request. The $\mathcal{TA}$ records the anomaly declassification request in the logging buffer $\mathcal{B}$, which can be recovered at a later stage to identify the affected IP core.

### 7.5.2 Denial of Service

DoS is a significant way of compromising the system and also disabling it. The DoS attack is also used to demonstrate message spoofing, where inputs to the system are malformed. The PWM core receives a reference duty cycle from the remote supervisory unit through the Ethernet core. The PWM core randomly malforms this input and uses a constant duty cycle intended to short the legs of the inverter and destroy the motor. The $\mathcal{TA}$ is aware of the latency incurred by the PWM core during the normal operation. Figure 7.9 shows the normal distribution of the PWM IP core's latency (# of clock cycles). The probability of observation from the latency can be calculated as follows:

$$P(t) = \frac{1}{\sigma_t\sqrt{2\pi}} e^{-(t-\mu_t)^2/2\sigma_t^2}$$

where $\mu_t$ is the mean and $\sigma_t$ is the variance of the execution time.

The latency calculation includes the clock cycles for transferring the data from the ARM core to the PWM modules. Worst-case latency occurs when all the inputs and outputs of the PWM IP core are considered as sensitive requiring multiple declassification calls. Threshold values generated from the latency measurement of the IP core help to determine the IP core deviation. The message spoofing due to a malformed input can result in a constant value being assigned as the duty cycle, resulting in reduced number of clock cycles. This results in

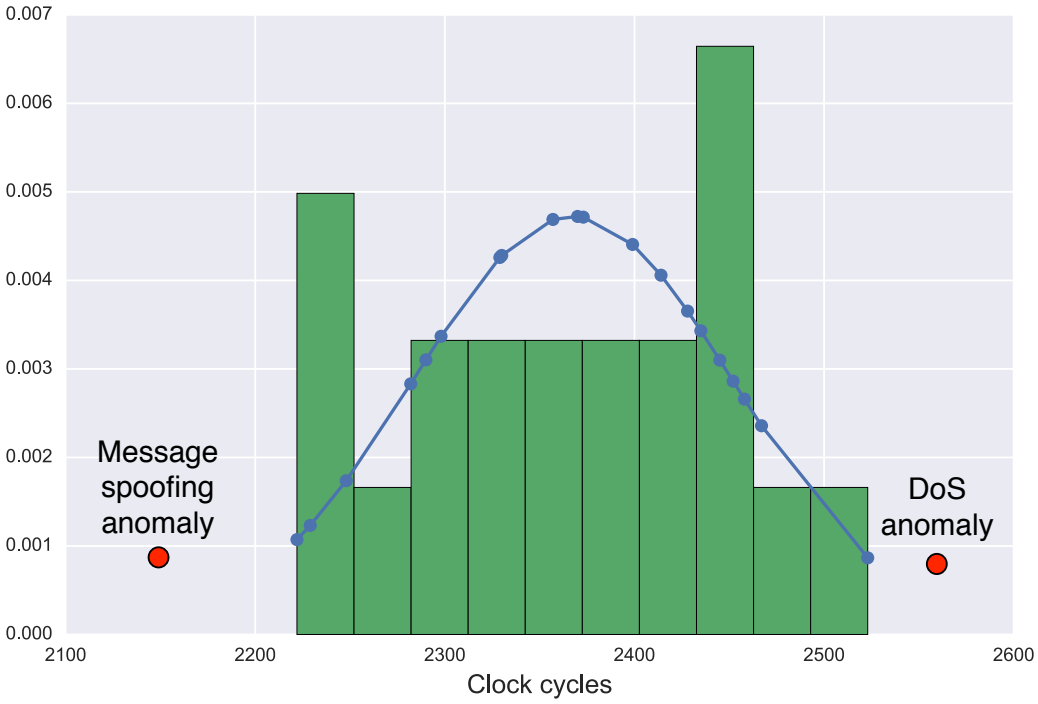


Figure 7.9: Probability of anomaly based on the PWM core's latency

the probability of the IP core's execution time to fall outside the normal distribution. The $\mathcal{TA}$ observes the latency of the IP core to be less than the normal distribution of the IP core's latency and hence detects an anomaly. The affected IP core is identified and logged in $\mathcal{B}$.

DoS attacks are primarily designed to prevent the IP core from responding in a timely fashion. For the DoS attack, the PWM IP core is designed to increase the execution time and delay the output PWM waveforms to the inverter. In this case, the probability of the increased execution time falls outside the normal distribution of the IP core. The $\mathcal{TA}$ flags the deviation of the IP core behavior and logs it in $\mathcal{B}$.

## 7.6 Conclusions

This chapter presents architectural extensions for enforcing trust in a CPS especially when integrating third party IP cores in an FPGA. Existing security requirements are outdated and do not address the challenges posed by an IoT-based CPS. In addition, the countermeasures are designed to target the software where the controller resides or the sensor hardware. TE-FIDES addresses this weakness by deploying distributed trust embedding with bindings to both hardware and software. The combination of the Trusted Wrapper bindings and a supervisory Trusted Anchor module is useful for the detection of a hardware Trojan attack. The state of the IP core is maintained using a trust metric and recorded in a logging buffer for further diagnostics. The TE-FIDES architecture is evaluated by mapping a FOC algorithm for motor control, which is commonly used in large scale industrial IoT systems. The TE-FIDES implementation is subjected to multiple hardware Trojan attacks and is found to be resilient. Future work include using the logging buffer data as a foundation for machine learning algorithms to build temporal models to detect anomalies with lower latency.

# Chapter 8

# Conclusions

*"The only truly secure system is one that is powered off, cast in a block of concrete and sealed in a lead-lined room with armed guards." — Gene Spafford*

---

## 8.1 Contributions

In a connected world, embedded systems form the backbone for assimilating and transmitting the data collected by various sensors in any environment. Embedding security in these systems does not enforce trust when untrusted components are glued together. Attackers are aware of the trust prevalent in these systems communicating with each other and search for methods to break the security possibly by exploiting the hardware. Countermeasures are deployed to protect the system after the vulnerabilities are identified. Attacks and countermeasures are a vicious cycle affecting the consumer and eroding the trust in such systems. Hence, it is necessary to evaluate and define the notion of trust that can guarantee protection against different hardware attacks. Therefore, a trustworthy foundation or architecture is required before the different layers of security can be built on top of the hardware and

strengthen the system against attacks.

This dissertation developed a lightweight cryptographic testbed [77, 113] and subjected it to a variety of hardware Trojan attacks. These attacks demonstrated that the detection of HTTs cannot be based on a single physical characteristic of the embedded system. Classic trusted hardware models based on the RoT architecture were also defeated using these attacks. These lessons led to the formulation of a novel HTT detectability metric (HDM) [100, 114, 115], that leverages a weighted combination of normalized physical parameters. We designed, implemented and analyzed seven types of HTTs to understand the limitations of existing post-silicon techniques. Our results show that using power, timing or utilization, at most 57% of designed HTTs were detected. The detection rate of HTTs increased to 86% with HDM. We also conducted analytical studies to determine the optimal HTT detection threshold that minimizes the sum of false alarm and missed detection probabilities.

The lessons from HDM led to the development of a trustworthy architecture FIDelity Enhancing Security (FIDES) to enhance trust in CPS that are not completely designed or developed in a trusted environment [116, 106]. FIDES essentially consists of two components:

(i) *Trusted Wrappers* are implemented in the FPGA fabric around each IP core. The Trusted Wrappers tag the output of the each IP core $i$ with a tag $\tau_i$ based on the predefined security policy $\Pi$. By tagging and tracking the interactions of the IP core within a system including the other IP cores, memory, OS or I/O ports, a nominal behavioral model of the system is generated. The deviation of the IP core behavior from this model is monitored by the Trusted Wrapper indicating an anomaly or a violation of the security policies. The Trusted Wrapper $W_i$ also records a snapshot of the trust metric $Q_i$ of the IP core, which is recorded by the $\mathcal{TA}$ in a logging buffer $\mathcal{B}$.

(ii) *Trusted Anchor*, which is more of a supervisory monitor keeping track of the communication between the IP cores and the peripherals. The Trusted Anchor enforces the compliance of the IP core communication with the centralized security policies $\Psi$.

Table 8.1: Comparison of TE-FIDES with other secure and trusted architectures

| **IAS-Octave Requirements** | **Secure and Trusted Architectures** | | | | |
|---|---|---|---|---|---|
| | UCI, FANCI, VeriTrust | Security Monitors (DEFENSE, Huffmire) | HACMS | Intel SGX | TE-FIDES |
| Confidentiality | ✓ | ✓ | ✓ | ✓ | ✓ |
| Integrity | ✓ | ✓ | ✓ | ✓ | ✓ |
| Availability | ✗ | ✓ | ✓ | ✓ | ✓ |
| Trustworthiness | ✗ | ✗ | ✗ | ✓ | ✓ |
| Privacy | ✗ | ✓ | ✗ | ✓ | ✓ |
| Non-repudiation | ✗ | ✓ | ✓ | ✓ | ✓ |
| Accountability | ✗ | ✗ | ✓ | ✗ | ✓ |
| Auditability | ✗ | ✓ | ✗ | ✗ | ✓ |

Trust Enhanced FIDES (TE-FIDES) is proposed to reduce the vulnerabilities arising from the declassification process of the third-party soft IP cores [117]. Each IP core's measure of trust is recorded by $Q_i$ and is used to identify a malicious core in the event of an attack. TE-FIDES enforces a secure enclave approach for embedding the IP cores, thereby preserving the confidentiality and integrity of the sensitive information in the system. In addition, TE-FIDES is evaluated by targeting a smart grid CPS application, where malicious third-party soft IP cores are prevented from damaging the system. Classic security requirements for a CPS are outdated and are not applicable to today's systems with multiple sensors deployed in an IoT-based CPS. Hence an updated set of requirements based on IAS-Octave is used to define the requirements for TE-FIDES architecture. Table 8.1 provides a comparison of TE-FIDES with other secure and trusted architectures in terms of meeting the IAS-Octave requirements. TE-FIDES is further evaluated by subjecting an industrial IoT application to

multiple hardware Trojan attacks.

The following is a list of publications (including full papers and posters) and invited talks related to this work:

- V. Venugopalan and C. D. Patterson, "Surveying the Hardware Trojan Threat Landscape for the Internet-of-Things", *Submitted to IEEE Internet of Things Journal – Special Issue on Security and Privacy in Cyber-Physical Systems.*

- V. Venugopalan and C. D. Patterson, "Architectural Refinements for Enhancing Trust and Securing Cyber-Physical Systems", *Submitted to IEEE International Conference on Advanced and Trusted Computing (ATC).*

- V. Venugopalan, C. D. Patterson, and D. M. Shila, "Detecting and Thwarting Hardware Trojan Attacks in Cyber-Physical Systems", *Proceedings of 2016 IEEE Conference on Communications and Network Security (CNS): International Workshop on Cyber-Physical Systems Security (CPS-Sec)*, pages 421-425, October 2016.

- D. M. Shila, V. Venugopalan, and C. D. Patterson, "Unraveling the Security Puzzle: A Distributed Framework to Build Trust in FPGAs". In *Network and System Security, of Lecture Notes in Computer Science*, volume 9408, pages 95-111. Springer International Publishing, November 2015. **[Best Paper Award]**

- D. M. Shila, V. Venugopalan, and C. D. Patterson, "FIDES: Enhancing Trust in Reconfigurable Based Hardware Systems", *Proceedings of the 2015 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1-7, Sept. 2015.

- D. M. Shila and V. Venugopalan, "Design, Implementation and Security Analysis of Hardware Trojan Threats in FPGA", *Proceedings of the 2014 IEEE International Conference on Communications (ICC)*, pages 719-724, June 2014.

- D. M. Shila, V. Venugopalan, and B. Murray, "A Multi-parameter Detection Scheme to Seek Stealthy Hardware Trojan Threats in FPGAs", presented at *CyberSecurity For*

*Cyber-Physical Vehicle Systems session at the Society of Automotive Engineers (SAE) International conference*, Detroit, MI, April 2014. **[Best Presentation Award]**

- J. Lamberti, D. M. Shila and V. Venugopalan, "xDEFENSE: An Extended DEFENSE for Mitigating Next Generation Intrusions", *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA 2014)*, Monterey, CA, Feb. 2014. [Poster]

- D. M. Shila and V. Venugopalan, "Design, Implementation and Security Analysis of Hardware Trojan Threats in FPGA", *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA 2014)*, Monterey, CA, Feb. 2014. [Poster]

- V. Venugopalan and D. M. Shila, "High Throughput Implementations of Cryptography Algorithms on GPU and FPGA", *Proceedings of the 2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 723-727, May 2013.

- V. Venugopalan and D. M. Shila, "Hardware Acceleration of TEA and XTEA Algorithms on FPGA, GPU and Multi-core Processors", *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA 2013)*, Monterey, CA, Feb. 2013. [Poster]

## 8.2 Future Work

Hardware Trojans have been detected using HDM, however HDM does not combine all physical characteristics of an embedded system. While this dissertation is written, hackers are actively developing ingenious methods to spoof normal behavior of systems to embed Trojans. Even though systems are going to be developed in future with even more secure architectures, we need to accept that systems are always vulnerable and prone to attacks.

A variation of the HDM being explored looks at the Trojan attacks from a game theoretic model to see what rewards can be given to the attacker to gain control over a small portion of a system. This part of a system may replicate a control system with all I/O and compute elements, however these signals remain sandboxed on a higher supervisory control scheme.

FIDES has demonstrated the decentralized security monitoring scheme to tag the information flow within a system and detect a behavior anomaly. This detection, prevention and logging of this behavior anomaly comes at a cost of latency and throughput to the application. TE-FIDES enhances the trust mechanism by deploying secure enclave-like hardware embeddings. TE-FIDES can be further improved by clustering IP cores based on their I/O peripheral access and the sensitivity tokens. This can result in clustered information flow and help in reducing the latency costs for declassifying the sensitive IP core communication.

Another area of ongoing research is the capability of making the *Trusted Anchor* intelligent. Machine learning methods such as Deep Belief Nets, Recurrent Neural Nets, and Inverse Deep Reinforcement Learning are being adapted to learn a system behavior and track the information flow by injecting synthetic anomalies. The resulting trained model can be then tested for identifying anomalies and triggers that can preempt an attack. Although training such a model is expensive in terms of computational cost and latency as shown in [118, 119, 120] for sensor fusion in robotics and UAV applications, offline GPU accelerated training can reduce this cost. The trained model can then be embedded into the *Trusted Anchor* residing on the ARM core to test for the deviations from the golden model.

# Bibliography


[1] J. Markoff, "A silent attack, but not a subtle one," *New York Times*, vol. 160, no. 55176, p. 6, 2010. [Online]. Available: http://www.agriculturedefensecoalition.org/sites/default/files/file/constitution_1/1G_2010_The_Stuxnet_Worm_A_Silent_Attack_but_Not_a_Subtle_One_Spread_Around_the_World_SEPT_26_2010_NYTimes.pdf

[2] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, 2011. [Online]. Available: http://i69.fr/autoblog/autoblogs/korben.info/media/0ae4f7a1.w32_stuxnet_dossier.pdf

[3] A. Matrosov, E. Rodionov, D. Harley, and J. Malcho, "Stuxnet under the Microscope," *ESET LLC (September 2010)*, 2010. [Online]. Available: https://go.eset.com/us/resources/white-papers/Stuxnet_Under_the_Microscope.pdf

[4] D. E. Sanger, "Obama order sped up wave of cyberattacks against Iran," *The New York Times*, vol. 1, no. 06, p. 2012, 2012. [Online]. Available: http://cyber-peace.org/wp-content/uploads/2013/06/Obama-Ordered-Wave-of-Cyberattacks-Against-Iran-NYTimes.pdf

[5] J. Appelbaum and L. Poitras, "Als Zielobjekt markiert," *Der Spiegel*, vol. 28, pp. 1–3, 2013.

[6] ABC News. (2014, November) 'Trojan Horse' bug lurking in vital

US computers since 2011. [Online]. Available: http://abcnews.go.com/US/trojan-horse-bug-lurking-vital-us-computers-2011/story?id=26737476

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

[8] S. Peterson and P. Faramarzi, "Iran hijacked US drone, says Iranian engineer," *The Christian Science Monitor*, 2011. [Online]. Available: http://www.csmonitor.com/World/Middle-East/2011/1215/Exclusive-Iran-hijacked-US-drone-says-Iranian-engineer-Video

[9] A. Greenberg, "Hackers remotely kill a Jeep on the highway with me in it," *Wired*, July 2015. [Online]. Available: http://www.wired.com/2015/07/hackers-remotely-kill-jeep-highway/

[10] E. Schlosser. (2016, December) World War Three, By Mistake. [Online]. Available: http://www.newyorker.com/news/news-desk/world-war-three-by-mistake

[11] M. Tehranipoor and F. Koushanfar, "A Survey of Hardware Trojan Taxonomy and Detection," *Design Test of Computers, IEEE*, vol. 27, no. 1, pp. 10–25, 2010.

[12] A. Waksman and S. Sethumadhavan, "Silencing Hardware Backdoors," in *Security and Privacy (SP), 2011 IEEE Symposium on*, 2011, pp. 49–63.

[13] S. Bhunia, M. Abramovici, D. Agrawal, P. Bradley, M. Hsiao, J. Plusquellic, and M. Tehranipoor, "Protection Against Hardware Trojan Attacks: Towards a Comprehensive Solution," *Design Test, IEEE*, vol. 30, no. 3, pp. 6–17, June 2013.

[14] D. Kaufman, "An analytical framework for cyber security," DTIC Document, Tech. Rep., 2011.

[15] S. Skorobogatov and C. Woods, "In the blink of an eye: There goes your AES key." *IACR Cryptology ePrint Archive*, vol. 2012, p. 296, 2012.

[16] 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.

[17] M. Beaumont, B. Hopkins, and T. Newby, "Hardware Trojans-Prevention, Detection, Countermeasures (A Literature Review)," DTIC Document, Tech. Rep., 2011.

[18] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy Hardware: Identifying and Classifying Hardware Trojans," *Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[19] X. Wang, M. Tehranipoor, and J. Plusquellic, "Detecting malicious inclusions in secure hardware: Challenges and solutions," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, June 2008, pp. 15–19.

[20] 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.

[21] L. Lin, W. Burleson, and C. Paar, "Moles: Malicious off-chip leakage enabled by side-channels," in *Proceedings of the 2009 International Conference on Computer-Aided Design*, ser. ICCAD '09. New York, NY, USA: ACM, 2009, pp. 117–122. [Online]. Available: http://doi.acm.org/10.1145/1687399.1687425

[22] 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.

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

[24] F. Wolff, C. Papachristou, S. Bhunia, and R. Chakraborty, "Towards Trojan-Free Trusted ICs: Problem Analysis and Detection Scheme," in *Design, Automation and Test in Europe, 2008. DATE '08*, March 2008, pp. 1362–1365.

[25] R. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware Trojan: Threats and Emerging Solutions," in *High Level Design Validation and Test Workshop, 2009. HLDVT 2009. IEEE International*, 2009, pp. 166–171.

[26] 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.

[27] Y. Shiyanovskii, F. Wolff, C. Papachristou, D. Weyer, and W. Clay, "Exploiting Semiconductor Properties for Hardware Trojans," 06 2009. [Online]. Available: http://arxiv.org/abs/0906.3834

[28] Z. Chen, X. Guo, R. Nagesh, A. Reddy, M. Gora, and A. Maiti, "Hardware Trojan Designs on BASYS FPGA Board," *Embedded system challenge contest in cyber security awareness week-CSAW*, 2008. [Online]. Available: http://isis.poly.edu/esc/2008/Reports/vt.pdf

[29] S. Skorobogatov, *Introduction to Hardware Security and Trust*. New York, NY: Springer New York, 2012, ch. Physical Attacks and Tamper Resistance, pp. 143–173. [Online]. Available: http://dx.doi.org/10.1007/978-1-4419-8080-9_7

[30] M. Hicks, M. Finnicum, S. T. King, M. Martin, and J. M. Smith, "Overcoming an Untrusted Computing Base: Detecting and Removing Malicious Hardware Automatically," in *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010, pp. 159–172.

[31] C. Sturton, M. Hicks, D. Wagner, and S. King, "Defeating UCI: Building Stealthy and Malicious Hardware," in *Security and Privacy (SP), 2011 IEEE Symposium on*, May 2011, pp. 64–77.

[32] A. Waksman, M. Suozzo, and S. Sethumadhavan, "FANCI: Identification of Stealthy Malicious Logic Using Boolean Functional Analysis," in *Proceedings of the 2013 ACM*

*SIGSAC conference on Computer & Communications Security*. ACM, 2013, pp. 697–708.

[33] K. El Defrawy, A. Francillon, D. Perito, and G. Tsudik, "SMART: Secure and Minimal Architecture for (Establishing a Dynamic) Root Of Trust," in *Proceedings of the Network & Distributed System Security Symposium (NDSS), San Diego, CA*, 2012.

[34] J. Zhang, F. Yuan, L. Wei, Z. Sun, and Q. Xu, "VeriTrust: Verification for hardware trust," in *Proceedings of the 50th Annual Design Automation Conference*. ACM, 2013, p. 61.

[35] S. Sethumadhavan, A. Waksman, M. Suozzo, Y. Huang, and J. Eum, "Trustworthy Hardware from Untrusted Components," *Communications of the ACM*, vol. 58, no. 9, pp. 60–71, 2015.

[36] F. Koushanfar and R. Karri, "Can the SHIELD protect our integrated circuits?" in *Circuits and Systems (MWSCAS), 2014 IEEE 57th International Midwest Symposium on*. IEEE, 2014, pp. 350–353.

[37] E. Love, Y. Jin, and Y. Makris, "Enhancing security via provably trustworthy hardware intellectual property," in *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 12–17.

[38] M. Abramovici and P. Bradley, "Integrated Circuit Security: new Threats and Solutions," in *Proceedings of the 5th Annual Workshop on Cyber Security and Information Intelligence Research: Cyber Security and Information Intelligence Challenges and Strategies*, ser. CSIIRW '09. New York, NY, USA: ACM, 2009, pp. 55:1–55:3. [Online]. Available: http://doi.acm.org/10.1145/1558607.1558671

[39] D. McIntyre, F. Wolff, C. Papachristou, S. Bhunia, and D. Weyer, "Dynamic Evaluation of Hardware Trust," in *Hardware-Oriented Security and Trust, 2009. HOST '09. IEEE International Workshop on*, 2009, pp. 108–111.

[40] 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.

[41] T. Huffmire, T. Sherwood, R. Kastner, and T. Levin, "Enforcing Memory Policy Specifications in Reconfigurable Hardware," *Computers & Security*, vol. 27, no. 5, pp. 197–215, 2008.

[42] T. Huffmire, T. Levin, T. Nguyen, C. Irvine, B. Brotherton, G. Wang, T. Sherwood, and R. Kastner, "Security Primitives for Reconfigurable Hardware-Based Systems," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 3, no. 2, p. 10, 2010.

[43] M. Banga and M. Hsiao, "A Novel Sustained Vector Technique for the Detection of Hardware Trojans," in *VLSI Design, 2009 22nd International Conference on*, 2009, pp. 327–332.

[44] Y. Jin and Y. Makris, "Hardware Trojan Detection Using Path Delay Fingerprint," in *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*. IEEE, 2008, pp. 51–57.

[45] M. Potkonjak, A. Nahapetian, M. Nelson, and T. Massey, "Hardware Trojan Horse Detection Using Gate-Level Characterization," in *Design Automation Conference, 2009. DAC'09. 46th ACM/IEEE*. IEEE, 2009, pp. 688–693.

[46] K. M. Zick and J. P. Hayes, "Low-Cost Sensing with Ring Oscillator Arrays for Healthier Reconfigurable Systems," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 5, no. 1, pp. 1:1–1:26, 2012. [Online]. Available: http://doi.acm.org/10.1145/2133352.2133353

[47] L.-W. Kim, J. Villasenor, and C. Koc, "A Trojan-Resistant System-on-Chip Bus Architecture," in *Military Communications Conference, 2009. MILCOM 2009. IEEE*, Oct 2009, pp. 1–6.

[48] A. Das, G. Memik, J. Zambreno, and A. Choudhary, "Detecting/Preventing Information Leakage on the Memory Bus Due to Malicious Hardware," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2010*, 2010, pp. 861–866.

[49] S. Trimberger, "Trusted Design in FPGAs," in *Proceedings of the 44th Annual Design Automation Conference*, ser. DAC '07. New York, NY, USA: ACM, 2007, pp. 5–8. [Online]. Available: http://doi.acm.org/10.1145/1278480.1278483

[50] J. B. Webb, "Methods for Securing the Integrity of FPGA Configurations," Master's thesis, Virginia Polytechnic Institute and State University, 2006.

[51] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing IC Piracy Using Reconfigurable Logic Barriers," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 66–75, 2010.

[52] R. Chakraborty and S. Bhunia, "Security against Hardware Trojan through a Novel Application of Design Obfuscation," in *Computer-Aided Design - Digest of Technical Papers, 2009. ICCAD 2009. IEEE/ACM International Conference on*, Nov 2009, pp. 113–116.

[53] R. Chakraborty, I. Saha, A. Palchaudhuri, and G. Naik, "Hardware Trojan Insertion by Direct Modification of FPGA Configuration Bitstream," *Design Test, IEEE*, vol. 30, no. 2, pp. 45–54, April 2013.

[54] S. Mal-Sarkar, A. Krishna, A. Ghosh, and S. Bhunia, "Hardware Trojan Attacks in FPGA Devices: Threat Analysis and Effective Counter Measures," in *Proceedings of the 24th Edition of the Great Lakes Symposium on VLSI*, ser. GLSVLSI '14. New York, NY, USA: ACM, 2014, pp. 287–292. [Online]. Available: http://doi.acm.org/10.1145/2591513.2591520

[55] A. A. Cárdenas, S. Amin, and S. Sastry, "Research Challenges for the Security of Control Systems," in *Proceedings of the 3rd Conference on Hot Topics in Security*, ser. HOTSEC'08. Berkeley, CA, USA: USENIX Association, 2008, pp. 6:1–6:6.

[56] S. Amin, A. A. Cárdenas, and S. S. Sastry, "Safe and Secure Networked Control Systems under Denial-of-Service Attacks," *Hybrid Systems: Computation and Control: 12th International Conference, HSCC 2009, San Francisco, CA, USA, April 13-15, 2009. Proceedings*, pp. 31–45, 2009.

[57] ARM TrustZone. [Online]. Available: http://www.arm.com/products/processors/technologies/trustzone/index.php

[58] *ARM security technology - Building a secure system using TrustZone technology*, ARM Inc., April 2009. [Online]. Available: http://infocenter.arm.com/help/topic/com.arm.doc.prd29-genc-009492c/PRD29-GENC-009492C_trustzone_security_whitepaper.pdf

[59] D. Rosenberg, "Reflections on trusting TrustZone," *BlackHat USA*, 2014. [Online]. Available: https://www.blackhat.com/docs/us-14/materials/us-14-Rosenberg-Reflections-on-Trusting-TrustZone.pdf

[60] Y. Gosain and P. Palanichamy, "TrustZone Technology Support in Zynq-7000 All Programmable SoCs," *Xilinx Report*, pp. 1–11, May 2014.

[61] Apple, "iOS Security: iOS 9.0 or later," pp. 1–60, September 2015. [Online]. Available: http://www.apple.com/business/docs/iOS_Security_Guide.pdf

[62] K. Fisher, "High Assurance Cyber Military Systems (HACMS): Making sure you are in control of your vehicle," 2013.

[63] K. Fisher, "Using Formal Methods to Enable More Secure Vehicles: DARPA's HACMS Program," in *Proceedings of the 19th ACM SIGPLAN International Conference on Functional Programming*, ser. ICFP '14. New York, NY, USA: ACM, 2014, pp. 1–1.

[64] P. C. Hickey, L. Pike, T. Elliott, J. Bielman, and J. Launchbury, "Building Embedded Systems with Embedded DSLs," *Proceedings of the 19th ACM SIGPLAN international conference on Functional programming*, vol. 49, no. 9, pp. 3–9, Aug. 2014. [Online]. Available: http://doi.acm.org/10.1145/2692915.2628146

[65] G. Klein, J. Andronick, K. Elphinstone, T. Murray, T. Sewell, R. Kolanski, and G. Heiser, "Comprehensive formal verification of an OS microkernel," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 1, pp. 2:1–2:70, Feb. 2014.

[66] L. Chen, J. Franklin, and A. Regenscheid, "Guidelines on Hardware-Rooted Security in Mobile Devices," *NIST Special Publication*, pp. 800–164, 2012.

[67] "Intel Software Guard Extensions Programming Reference," October 2014. [Online]. Available: https://software.intel.com/sites/default/files/managed/48/88/329298-002.pdf

[68] M. Hoekstra, "Intel SGX for Dummies (Intel SGX Design Objectives)," November 2015. [Online]. Available: https://software.intel.com/en-us/blogs/2013/09/26/protecting-application-secrets-with-intel-sgx

[69] T. Cook. (2016, February) Apple's message to its customers. [Online]. Available: http://www.apple.com/customer-letter/

[70] Xilinx Inc., "MicroZed Evaluation Kit with Xilinx Zynq-7000 All Programmable SoC," January 2014. [Online]. Available: http://www.microzed.org/product/microzed

[71] Xilinx Inc., "XAPP744 - Hardware In The Loop (HIL) Simulation for the Zynq-7000 All Programmable SoC," Tech. Rep., 2013.

[72] G. Neil, "When things start to think," 2000.

[73] L. Rosenthal and V. Stanford, "NIST smart space: pervasive computing initiative," in *Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000.(WET ICE 2000). Proeedings. IEEE 9th International Workshops on.* IEEE, 2000, pp. 6–11.

[74] T. Eisenbarth, S. Kumar, C. Paar, A. Poschmann, and L. Uhsadel, "A survey of lightweight-cryptography implementations," *IEEE Des. Test*, vol. 24, no. 6, pp. 522–533, Nov. 2007. [Online]. Available: http://dx.doi.org/10.1109/MDT.2007.178

[75] J.-P. Kaps, "Chai-Tea, Cryptographic Hardware Implementations of xTEA," in *Proceedings of the 9th International Conference on Cryptology in India: Progress in Cryptology*, ser. INDOCRYPT '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 363–375. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-89754-5_28

[76] P. Yalla and J.-P. Kaps, "Lightweight Cryptography for FPGAs," in *Proceedings of the 2009 International Conference on Reconfigurable Computing and FPGAs*, ser. RECONFIG '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 225–230. [Online]. Available: http://dx.doi.org/10.1109/ReConFig.2009.54

[77] V. Venugopal and D. Manikantan Shila, "Hardware Acceleration of TEA and XTEA Algorithms on FPGA, GPU and Multi-Core Processors (abstract only)," in *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, ser. FPGA '13. New York, NY, USA: ACM, February 2013, pp. 270–270. [Online]. Available: http://doi.acm.org/10.1145/2435264.2435326

[78] N. Nishikawa, K. Iwai, and T. Kurokawa, "High-Performance Symmetric Block Ciphers on CUDA," in *Networking and Computing (ICNC), 2011 Second International Conference on*, 2011, pp. 221 –227.

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

[80] D. J. Wheeler and R. M. Needham, "TEA extensions," Cambridge University, England, Tech. Rep., October 1997.

[81] V. Venugopal, K. Richards, S. Barden, T. Rimmele, S. Gregory, and L. Johnson, "Accelerating Real-time processing of the ATST Adaptive Optics System using Coarse-

grained Parallel Hardware Architectures," in *International Conference on Engineering of Reconfigurable Systems and Algorithms (ERSA)*, July 2011, pp. 296–301.

[82] V. Venugopalan, "Evaluating Latency and Throughput Bound Acceleration of FPGAs and GPUs for Adaptive Optics Algorithms," in *IEEE High Performance Extreme Computing Conference (HPEC)*, Sept 2014, pp. 1–6.

[83] V. Venugopal, C. D. Patterson, and K. A. Shinpaugh, "Accelerating Particle Image Velocimetry Using Hybrid Architectures," in *Symposium on Application Accelerators in High-Performance Computing (SAAHPC)*, July 2009.

[84] V. Venugopal and S. Kannan, "Accelerating Real-Time LIDAR Data Processing Using GPUs," in *Circuits and Systems (MWSCAS), 2013 IEEE 56th International Midwest Symposium on*, August 2013, pp. 1168–1171.

[85] M. Giering, V. Venugopalan, and K. Reddy, "Multi-Modal Sensor Registration for Vehicle Perception via Deep Neural Networks," in *IEEE High Performance Extreme Computing Conference (HPEC)*, September 2015, pp. 1–6.

[86] C. Tokgoz and V. Venugopal, "GPU Accelerated Iterative Physical Optics to Predict RF Propagation in Urban Environments," in *Antennas and Propagation Society International Symposium (APSURSI), 2013 IEEE*, July 2013, pp. 1878–1879.

[87] V. Venugopalan and C. Tokgoz, "GPU acceleration of iterative physical optics-based electromagnetic simulations," in *IEEE High Performance Extreme Computing Conference (HPEC)*, September 2015, pp. 1–6.

[88] Nvidia Inc. (2015, October) Nvidia Jetson: The world's first embedded supercomputer. [Online]. Available: http://www.nvidia.com/object/jetson-tk1-embedded-dev-kit.html

[89] NVIDIA Inc., "NVIDIA's Next Generation CUDA Compute Architecture: Kepler TM GK110," Whitepaper, May 2012.

[90] Xilinx Inc., "Xilinx Zynq-7000 SoC ZC702 Evaluation kit." [Online]. Available: http://www.xilinx.com/products/boards-and-kits/EK-Z7-ZC702-G.htm

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

[92] 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.

[93] 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.

[94] S. Skorobogatov and C. Woods, *Cryptographic Hardware and Embedded Systems – CHES 2012: 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ch. Breakthrough Silicon Scanning Discovers Backdoor in Military Chip, pp. 23–40. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33027-8_2

[95] *Supply Chain Hardware Integrity For Electronics Defense (SHIELD)*. Defense Advanced Research Projects Agency (DARPA), Microsystems Technology Office/MTO Broad Agency Announcement, 2014.

[96] *Trusted Integrated Chips (TIC) Program Broad Agency Announcement 11–09*. Intelligence Advanced Research Projects Activity (IARPA), 2011.

[97] S. K. Haider, C. Jin, M. Ahmad, D. M. Shila, O. Khan, and M. van Dijk, "HaTCh: Hardware Trojan Catcher," *Cryptology ePrint Archive, Report 2014/943*, 2014. [Online]. Available: http://eprint.iacr.org

[98] M. Bilzor, T. Huffmire, C. Irvine, and T. Levin, "Security Checkers: detecting Pro-

cessor Malicious Inclusions at Runtime," in *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on*. IEEE, 2011, pp. 34–39.

[99] T. Huffmire, S. Prasad, T. Sherwood, and R. Kastner, "Policy-Driven Memory Protection for Reconfigurable Hardware," in *Computer Security–ESORICS 2006*. Springer, 2006, pp. 461–478.

[100] D. M. Shila and V. Venugopal, "Design, Implementation and Security Analysis of Hardware Trojan Threats in FPGA," in *IEEE International Conference on Communications (ICC)*, June 2014, pp. 719–724.

[101] J. Lamberti, D. Manikantan Shila, and V. Venugopal, "xDEFENSE: An Extended DEFENSE for Mitigating Next Generation Intrusions (abstract only)," in *Proceedings of the 2014 ACM/SIGDA International Symposium on Field-programmable Gate Arrays*, ser. FPGA '14. New York, NY, USA: ACM, 2014, pp. 253–253.

[102] A. C. Myers and B. Liskov, *A Decentralized Model for Information Flow Control*. ACM, 1997, vol. 31, no. 5.

[103] M. M. Farag, L. W. Lerner, and C. D. Patterson, "Interacting with hardware Trojans over a network," in *Hardware-Oriented Security and Trust (HOST), 2012 IEEE International Symposium on*. IEEE, 2012, pp. 69–74.

[104] C. H. Gebotys, *Security in Embedded Devices*. Springer Science & Business Media, 2009.

[105] Xilinx Inc. (2014) LogiCORE IP AXI Interconnect v2.1 Product Guide.

[106] D. M. Shila, V. Venugopalan, and C. D. Patterson, "Unraveling the Security Puzzle: A Distributed Framework to Build Trust in FPGAs," in *Network and System Security*, ser. Lecture Notes in Computer Science, vol. 9408. Springer International Publishing, November 2015, pp. 95–111.

[107] S. M. Amin and B. F. Wollenberg, "Toward a smart grid: power delivery for the 21st century," *IEEE power and energy magazine*, vol. 3, no. 5, pp. 34–41, 2005.

[108] R. Marquardt, "Modular multilevel converter: An universal concept for hvdc-networks and extended dc-bus-applications," in *Power Electronics Conference (IPEC), 2010 International*, June 2010, pp. 502–507.

[109] D. G. Holmes and T. A. Lipo, *Pulse width modulation for power converters: principles and practice*. John Wiley & Sons, 2003, vol. 18.

[110] Y. Cherdantseva and J. Hilton, "A Reference Model of Information Assurance & Security," in *2013 Eighth International Conference on Availability, Reliability and Security (ARES)*. IEEE, 2013, pp. 546–555.

[111] Y. Mo and B. Sinopoli, "Secure Control against Replay Attacks," in *2009 47th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Sept 2009, pp. 911–918.

[112] V. Costan, I. Lebedev, and S. Devadas, "Sanctum: Minimal Hardware Extensions for Strong Software Isolation," in *25th USENIX Security Symposium (USENIX Security 16)*. Austin, TX: USENIX Association, 2016, pp. 857–874. [Online]. Available: https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/costan

[113] 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.

[114] D. Manikantan Shila and V. Venugopal, "Design, Implementation and Security Analysis of Hardware Trojan Threats in FPGA (abstract only)," in *Proceedings of the 2014 ACM/SIGDA International Symposium on Field-programmable Gate Arrays*, ser. FPGA '14. New York, NY, USA: ACM, 2014, pp. 247–247. [Online]. Available: http://doi.acm.org/10.1145/2554688.2554713

[115] D. M. Shila, V. Venugopal, and B. Murray, "A Multi-parameter Detection Scheme to Seek Stealthy Hardware Trojan Threats in FPGAs," in *CyberSecurity For Cyber-Physical Vehicle Systems*. Society of Automotive Engineers (SAE) International conference, April 2014.

[116] D. Shila, V. Venugopalan, and C. Patterson, "FIDES: Enhancing Trust in Reconfigurable Based Hardware Systems," in *IEEE High Performance Extreme Computing Conference (HPEC)*, September 2015, pp. 1–7.

[117] V. Venugopalan, C. D. Patterson, and D. M. Shila, "Detecting and Thwarting Hardware Trojan Attacks in Cyber-Physical Systems," in *2016 IEEE Conference on Communications and Network Security (CNS): International Workshop on Cyber-Physical Systems Security (CPS-Sec)*, October 2016, pp. 421–425.

[118] M. Giering, M. Shashanka, S. Sarkar, and V. Venugopalan, "Sensor Data Fusion for Prognostics and Health Monitoring," 2016, WO Patent App. PCT/US2015/066,673. [Online]. Available: https://www.google.com/patents/WO2016100816A1?cl=en

[119] M. Giering, K. Reddy, and V. Venugopalan, "Multi-Modal Sensor Data Fusion for Perception Systems," 2016, WO Patent App. PCT/US2015/066,664. [Online]. Available: https://www.google.com/patents/WO2016100814A1?cl=en

[120] S. Sarkar, V. Venugopalan, K. Reddy, J. Ryde, N. Jaitly, and M. Giering, "Deep Learning for Automated Occlusion Edge Detection in RGB-D Frames," *Journal of Signal Processing Systems*, pp. 1–13, December 2016. [Online]. Available: http://dx.doi.org/10.1007/s11265-016-1209-3