Contents

List of Figures

Chapter 1

General Concepts of Hardware Trojans

1.1 Introduction

Here in this chapter I will cover the concepts of hardware Trojans, their classification, threats, taxonomy, location where possibly going to find them, their effects on the entire circuit, comparison of some hardware Trojan attacks and a complete study case to show real impact. This chapter presents concepts from available literature on hardware Trojans.

1.2 Definition

A hardware Trojan is a malicious modification of the circuitry of an IC chip. It is done during the design or fabrication of chip. It is also a piece of hardware, which is hiding inside another larger piece of hardware. It wakes up at unpredictable times and does something malicious, which is again unpredictable. Hardware Trojans may be introduced as hidden "Back-doors" that are inserted while designing an IC, by using a pre-made circuit known as intellectual property core (IP core) that have been purchased from a non-reputable source. There are two main things that categorize a Hardware Trojan: first, its physical Representation that is how it behaves, and how it looks like, second its behavior that is how it shows up and what are its effects. Among the properties of a Hardware Trojan are that it can take place pre- or post-manufacturing, that it is inserted by some intellectual adversary, and that it is stealthy and nearly impossible to detect. Hardware Trojan consists of the Trigger that decides when the HT will wake up and the Payload that decides what will happen when the Trojan will wake up. It is maliciously placed in the original circuit; user does not know about this because most of the time circuit will behave normally, but sometimes it behaves unpredictably and maliciously whenever it wakes up.

1.3 Hardware Trojan threats

Economic reasons dictate that most of the modern integrated circuits (ICs) are manufactured in offshore fabrication facilities. Moreover, modern IC design often involves intellectual property (IP)[1] cores supplied by third-party vendors, outsourced design and test services as well as electronic design automation (EDA)[2] and different vendors supply software tools. Thus, security level of such ICs is often affected. As the full potential of the threat posed by the hardware Trojan has been realized and acknowledged by the electronics industries many different conspiracy theories have emerged.

In this section, we first introduce an IC market model proposed by Zhang and Qu [Zhang and Qu, 2014], and then describe the potential threats from HTs in the model.

1.3.1 IC market model

In the context illustrated in Figure 1.1, the IC design, manufacturing, and application process typically involve five distinct entities. Each party's role is outlined below:

1. Foundries:
    - **Definition:** Semiconductor manufacturers such as TSMC and IBM.
    - **Role:** Contract with System-on-Chip (SoC) designers to manufacture integrated circuits (ICs).
2. **SoC Designers:**
    - **Definition:** Entities responsible for designing and producing commercial products, incorporating various intellectual properties (IPs).
    - **Role:** Collaborate with foundries for IC fabrication.
3. **IP Vendors:**
    - **Definition:** Developers of intellectual property cores, such as memory blocks and DSP cores, tailored for SoC designers.
    - **Role:** Supply essential IP components for SoC design.
4. **EDA Tool Vendors:**
    - **Definition:** Providers of Electronic Design Automation (EDA) tools, such as Altera and Xilinx.
    - **Role:** Offer tools to facilitate the design of large-scale integrated circuits for both SoC designers and IP vendors.
5. IC end users: Companies or individuals purchase commercial products from SoC designers.

[1]An IP core is a reusable unit of logic or integrated circuit (IC) layout design. It is the IP of one party and may be licensed by others for use in their own ICs and semiconductors. For more information visit (https://www.techtarget.com/whatis/definition/IP-core-intellectual-property-core)

[2]Electronic Design Automation, or EDA, is a market segment consisting of software, hardware, and services with the collective goal of assisting in the definition, planning, design, implementation, verification, and subsequent manufacturing of semiconductor devices, or chips. For more information visit (https://www.synopsys.com/glossary/what-is-electronic-design-automation.html)

This representation captures the key participants in the IC design system, highlighting their distinct roles and interactions throughout the design, manufacturing, and application phases. The information in this passage serves as a valuable reference for understanding the dynamics of the semiconductor industry (Figure 1.1).

Figure 1.1 illustrates the interactions among the various entities within the IC market model. In this model, a directional arrow signifies the flow of services from the supplier to the receiver. Broadly speaking, each party involved in this model contributes competitive products to others. Specifically, System-on-Chip (SoC) designers form connections with different entities in the IC market. As recipients of services, SoC designers acquire intellectual properties (IPs) from IP vendors to streamline the development process, utilize licensed Electronic Design Automation (EDA) tools from EDA tool vendors to enhance their design capabilities, and engage with foundries for chip fabrication. Conversely, as service providers, SoC designers offer their products to end-users lacking a chip-level development team who require chips for specific applications. Furthermore, IP vendors also procure software tools from EDA tool vendors. This model, originating from the evolution of the IC industry, enables each party to concentrate on their respective areas of expertise. Nevertheless, the involvement of multiple parties in the production of a single product introduces vulnerabilities to hostile insertion of Hardware Trojans (HT).

1.3.2 HT threat between SoC designers and foundries

Throughout the fabrication process, there is no assurance that foundries refrain from incorporating a specific type of Hardware Trojan (HT) into the chips. Chips manufactured in foundries face potential threats from untrusted personnel or external entities with access to the fabrication process. To illustrate, a Trojan may infiltrate the integrated circuit (IC) by deliberately or inadvertently altering the dopant level or mask layout during sample or mass production [Jacob et al., 2014]. Furthermore, foundries possess proprietary tools capable of manipulating chip fabrication for potentially malicious purposes. Additionally, foundries might delegate mask generation tasks to third-party entities, presenting an opportunity for the intentional inclusion of malicious mask macros in the GDSII.

1.3.3 HT threat between SoC designers and IP vendors

In the interaction between System-on-Chip (SoC) designers and Intellectual Property (IP) vendors, it is imperative for the former to ensure that the acquired IPs do not conceal malicious function units, as their detection becomes exceedingly challenging later on. A skilled adversary can design various types of Trojans during the presilicon stage. The majority of Hardware Trojans (HTs) proposed in existing literature are introduced and integrated into the design at the Register Transfer Level (RTL), before the stages of synthesis, placement, and routing [King et al., 2008]. An untrusted insider within IP vendors can easily manipulate the RTL, introducing malicious codes, altering macros during design synthesis, and even modifying the placement and routing to accommodate the Trojan circuitry. Moreover, an untrusted contractor involved in specifying components for IP vendors and SoC designers also has the opportunity to incorporate malicious elements.

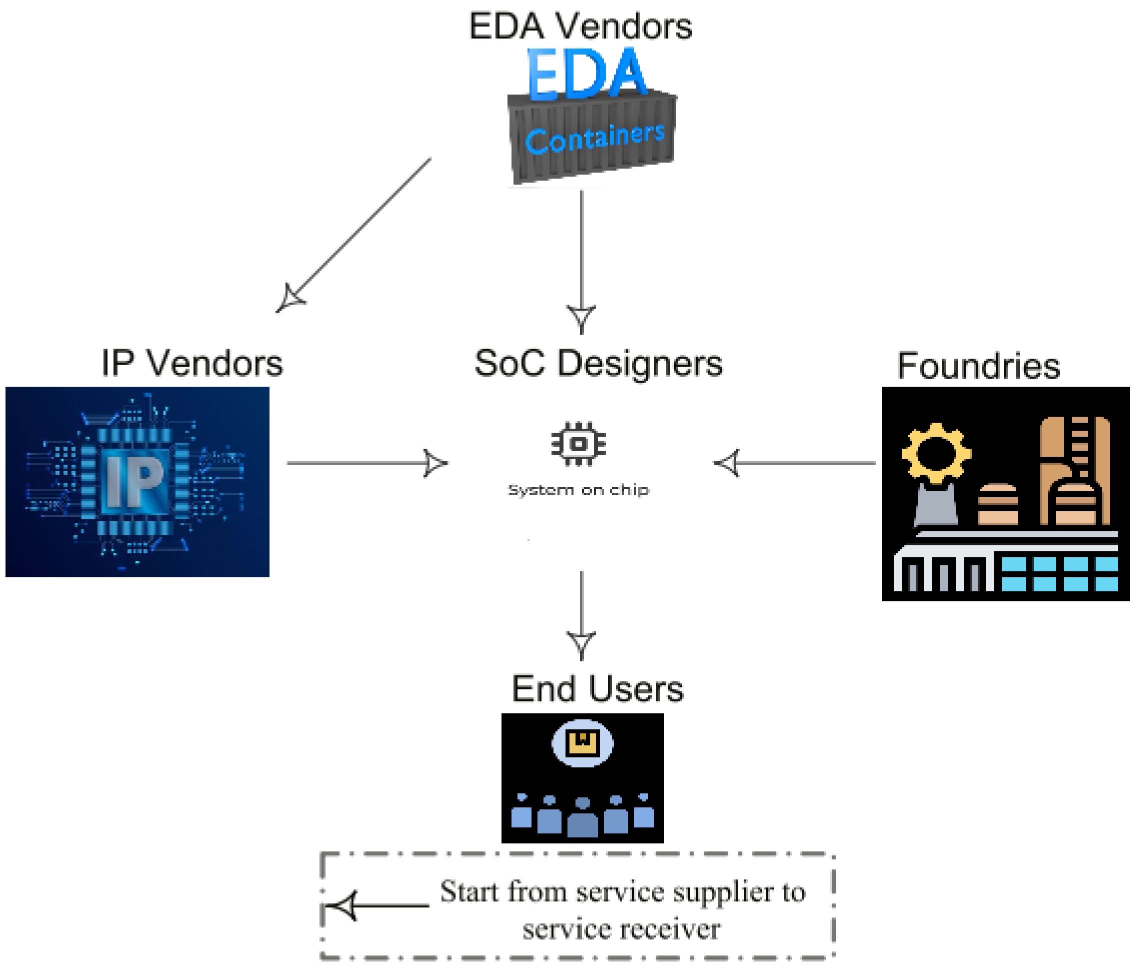


Figure 1.1: IC market model

1.3.4 HT threat between IP vendors (or SoC designers) and EDA vendors

EDA tools play a crucial role in various significant phases of design. The software tools created by Electronic Design Automation (EDA) vendors could potentially harbor malicious codes, leading to the unauthorized collection of valuable data within Intellectual Properties (IPs) and System-on-Chips (SoCs). In a recent study by Qu and Yuan [Qu and Yuan, 2014], an examination of security vulnerabilities in EDA design tools revealed that logic implementations inferred by these tools may exceed the necessary requirements. This holds true regardless of the trustworthiness of the design team, the origin of the EDA tools, or the reliability of IP providers. These unforeseen vulnerabilities could be exploited by adversaries to execute attacks.

1.3.5 HT threat between end users and SoC designers

End users lacking an in-house chip-level design team express concerns about Hardware Trojan (HT) attacks in products procured from System-on-Chip (SoC) designers. These HTs can be surreptitiously inserted into the chips during the SoC design phase, circumventing software security measures and enabling unauthorized surveillance of users. Detecting this type of hardware-level security threat proves challenging for end users, eroding their trust in traditionally reliable chips. Instances of HTs and backdoors concealed in critical systems such as weapons control, nuclear power plants, and public transportation have been documented in [Skorobogatov,].

1.4 Critical fields affected by Trojans

1.4.1 Military

Since modern military machinery rely on advanced electronic systems to perform its work, it becomes more vulnerable to hardware Trojan attacks. In September 2008, Israeli jets bombed a suspected nuclear installation in northeastern Syria. Among the many mysteries still surrounding that strike was the failure of a Syrian radar supposedly state-of-the-art to warn the Syrian military of the incoming assault. It was not long before military and technology bloggers concluded that this was an incident of electronic warfare and not just any kind. Commercial off-the-shelf microprocessors in the Syrian radar might have been purposely fabricated with a hidden "backdoor" inside. By sending a preprogrammed code to those chips, an unknown antagonist had disrupted the chips function and temporarily blocked the radar.

1.4.2 Banking and Finance

Modern banking rely on the use of the new technology ranging from credit cards to ATMs. The advanced electronics behind the fabrication of these devices open the door to more security concerns. Hardware Trojan designs could be specially crafted in ATM circuit, and attacker could remotely trigger such intruder component, thus taking over the whole money transactions.

1.4.3 Nuclear centers and drilling Rigs

In order to be able to monitor and control drilling operations, drilling rigs are doted by huge number of electronic sensors and micro-electronic systems that give remote operation centers complete control of the drilling platform, besides these systems provide real time data. If a black hat designer mess with the circuits of this systems, millions of dollars would be lost, and the country economic would be damaged. Advanced embedded systems are being used in nuclear power facilities to control and monitor different operations needed.

1.5 Trojan Taxonomy and Classification

In this section, I present detailed classification for hardware Trojans according to several studies, delineating the stages of the design process where hardware Trojans may be incorporated. The design phases of System-on-Chip (SoC), Application-Specific Integrated Circuit (ASIC), and Field-Programmable Gate Array (FPGA), encompassing specification, design, fabrication, assembly, and testing, are susceptible to potential hardware Trojan insertions. The design phase encompasses diverse abstraction levels, including System Level, Register-Transfer Level, Gate Level, Transistor Level, and Physical Level. Given that multiple teams collaborate on the design evolution across these abstraction levels, the surreptitious insertion of Trojans becomes feasible. Hardware components in digital designs encompass Processors, Memory, Input/Output ports, Power supply, Clock, etc., providing potential points for Trojan insertion.

HT circuits can be classified into various forms based on different characteristics. Numerous research works have presented comprehensive taxonomies to encompass a broad spectrum of hardware trojan instances.

1.5.1 Banga et al. classification

In their study [?], they Categorized hardware trojans into two groups based on logical types, namely combinational and sequential.

Combinational

A combinational circuit activates when a specific condition arises in the internal signals and/or circuit flip-flops or a segment of it.

Sequential

A finite state machine (FSM) observes a section of the internal circuit signals and activates the output when a specific sequence(s) occurs. Typically, Trojans involve sequential sub-circuits. However, combinational Trojans may be employed when targeting a hard property of the system.

1.5.2 Tehranipoor et al. classification

Wang, Tehranipoor, and Plusquellic developed the first detailed taxonomy for hardware Trojans [Wang et al., 2008], they decomposed the Trojan taxonomy into three main categories as shown in Figure 1.2 according to their physical, activation, and action characteristics.

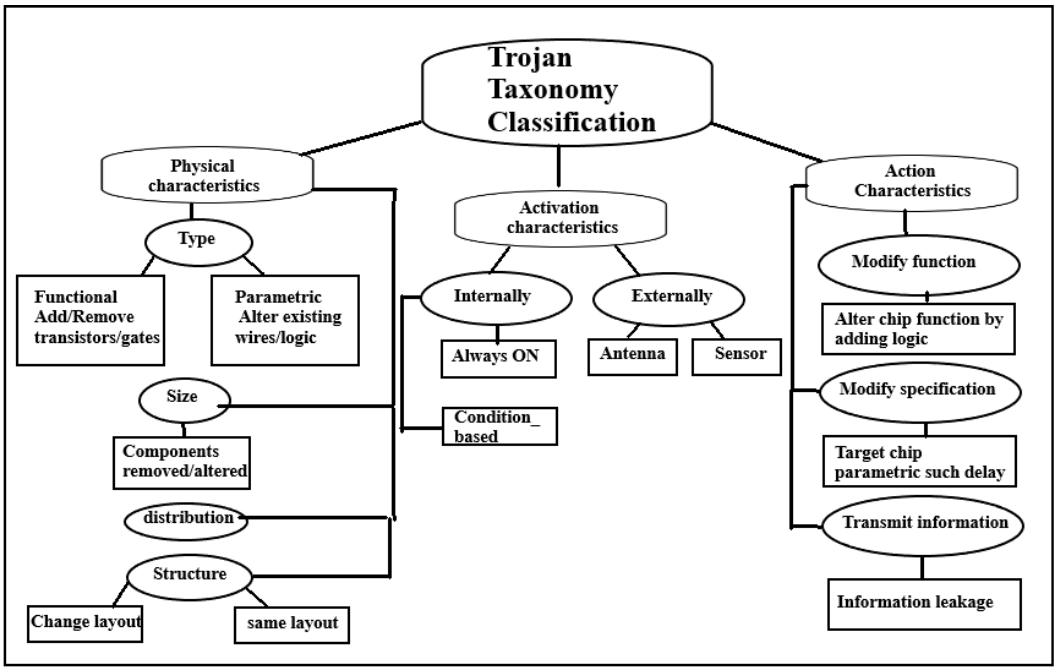


Figure 1.2: Taxonomy of Trojans

Physical characteristics

The physical characteristics category delineates the diverse hardware manifestations of Trojans. Trojans are further categorized into functional and parametric classes in the type category. The functional class encompasses Trojans that manifest physically through the addition or removal of transistors or gates, while the parametric class pertains to Trojans realized through alterations to existing wires and logic. The size category accounts for the quantity of components on the chip that have been introduced, removed, or compromised. The distribution category details the Trojan's location within the chip's physical layout. The structure category comes into play when an adversary is compelled to regenerate the layout to insert a Trojan, potentially altering the chip's physical form factor. Such alterations could lead to a different placement for some or all design components. Any malevolent modifications in the physical layout that influence the chip's delay and power characteristics would facilitate the detection of Trojans.

Activation characteristics

Activation characteristics pertain to the criteria triggering a Trojan to become active and execute its disruptive function. These characteristics can be broadly classified into two categories, as depicted in Figure 1.2: externally activated (e.g., triggered by an antenna or a sensor interacting with the external environment) and internally activated. The latter category is further divided into two subcategories: "always on," where the Trojan remains constantly active, capable of disrupting the chip's function at any moment, and condition-based, where the Trojan remains inactive until specific conditions are met.

The "always on" subclass involves Trojans implemented by modifying the chip's geometries, rendering certain nodes or paths more susceptible to failure. Adversaries may insert Trojans at nodes or paths seldom exercised, thereby enhancing their stealth. The condition-based subclass encompasses Trojans lying dormant until specific conditions are satisfied. Activation conditions may be contingent on external environmental factors monitored by a sensor (e.g., electromagnetic interference, humidity, altitude, or temperature). Alternatively, conditions may be tied to an internal logic state, a specific input pattern, or the value of an internal counter. In these cases, the Trojan is implemented by introducing logic gates and/or flip-flops to the chip, representing a combinational or sequential circuit.

Action characteristics

Action characteristics delineate the various types of disruptive behaviors instigated by a Trojan. The categorization illustrated in Figure 1.2 divides Trojan actions into three distinct classes: modify function, modify specification, and transmit information. Trojans falling into the modify function class are those that alter the chip's function either by introducing additional logic or by eliminating or bypassing existing logic. Trojans categorized under the modify-specification class target the chip's parametric properties, such as delay, achieved by an adversary manipulating the geometries of existing wires and transistors. Finally, the transmit-information class encompasses Trojans designed to convey crucial information to an adversary.

1.5.3 Zhang et al. classification

In their study [Zhang et al., 2015], they Categorized hardware trojans into two groups: bug-based HTs and parasite-based HTs, determined by their effects on the regular functionalities of the circuits.

Bug-Based HT

A bug-based HT instance alters the circuit in a way that leads to a loss of some of its typical functionalities. a simple inverter on input can change the whole functionality of the original design. Figure1.3 illustrates the Bug-Based HT impact on a target circuit.

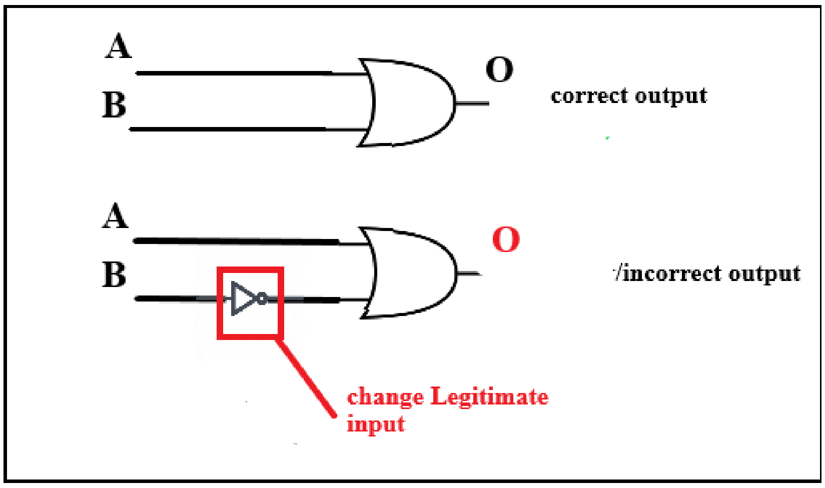


Figure 1.3: Bug-Based HT using logic inverter

The bug-based HT instance can be considered simply as a design flaw, albeit with malicious intent, since the design fails to realize all of its normal functionalities specified in the design requirements. Consequently, extensive simulation/emulation is likely to identify this type of HT scenario. In this regard, bug-based HT are generally not a preferred choice for attackers seeking stealthiness in HT attacks.

Parasite-Based HT

A parasite-based hardware trojan instance coexists with the original circuit without compromising any of the normal functions of the original design. consider Figure 1.4 , To regulate the execution of the design, switching between its regular and malicious functions, the attacker may utilize supplementary inputs(X1,X2) as triggering mechanisms. To evade trust validation, trigger inputs are typically meticulously chosen, and the trigger condition is crafted to be an exceedingly uncommon event that is rarely encountered during verification tests.

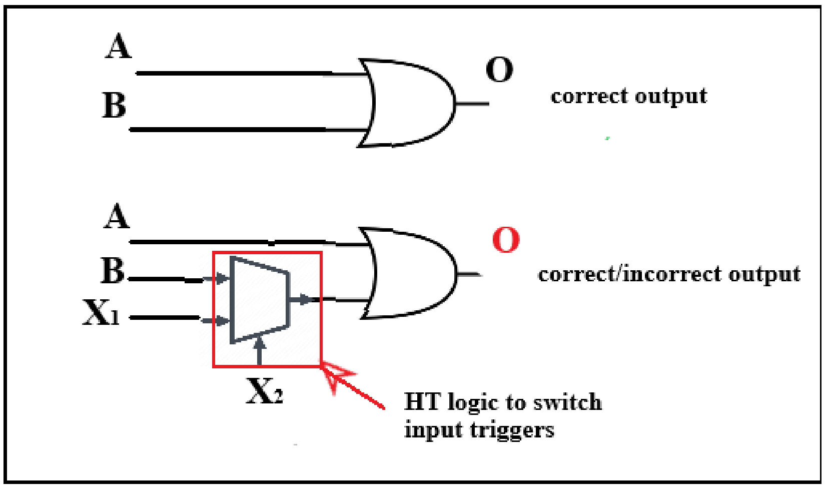


Figure 1.4: Parasite-Baset HT using logic Multiplexer to trigger malicious input.

By evaluating the output, we observe that The circuit can then perform the normal and malicious functions alternately, controlled by trigger inputs.

1.5.4 Bhunia et al. classification

in their study [Bhunia et al., 2014], They suggested categorizing HTs into **analog Trojans** and **digital Trojans**, depending on the trigger and payload mechanisms.

Various versions of the taxonomy for Trojan circuits have been introduced, and it is continuously developing with the discovery of new Trojan types and attacks. In this context, a broad classification is presented in Figure 1.5 , focusing on variations in activation mechanisms and Trojan effects. Hardware Trojans are categorized into analog and digital Trojans based on the trigger condition. Analog Trojans are activated by analog factors such as temperature, delay, or device aging effects, while digital Trojans are triggered by Boolean logic functions. Digitally triggered Trojans can be further divided into combinational and sequential types. Analog Trojans encompass attacks on process steps that compromise the reliability of either all or specific chips.

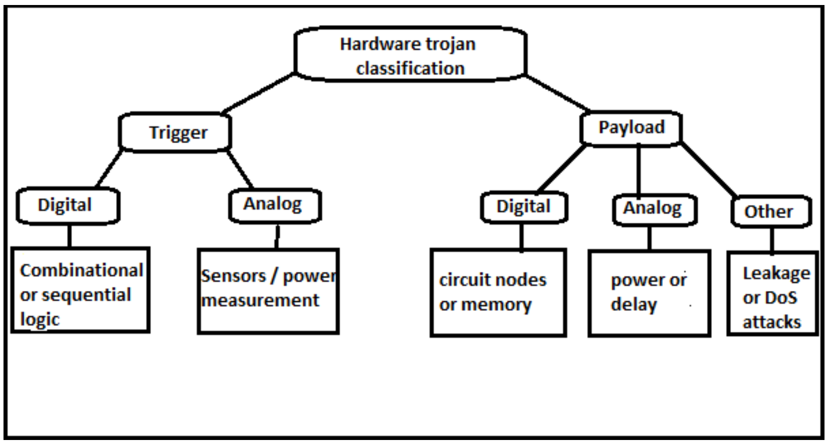


Figure 1.5: Analog/Digital HT classification

1.5.5 Huang et al. classification

In their study [Huang et al., 2020], a novel and organized classification is established based on the correlation between the placements of Hardware Trojan (HT) circuits within a System-on-Chip (SoC) and the specific targets impacted by the Trojan attacks upon activation. This results in the categorization of HT circuits into three types: IP-level HTs, bus-level HTs, and SoC-level HTs.

IP-level Trojans

Hardware trojan is directly implanted into the IP core of the targeted SoC, activation is performed internally by rare condition inputs. Upon activation, its impact is limited solely to the specific IP cores in which it has been implanted (see Figure 1.6).

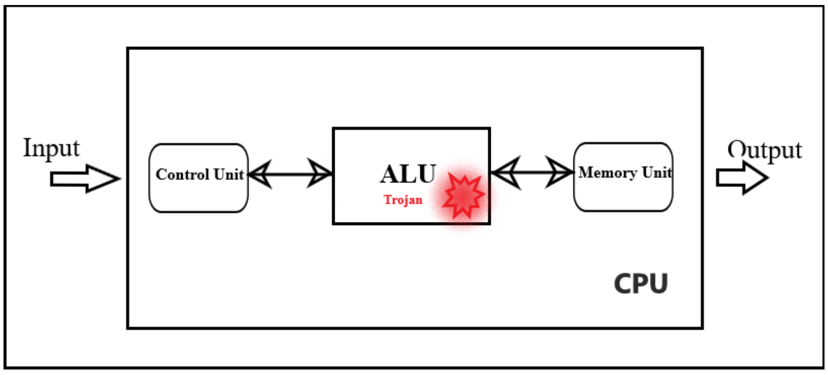


Figure 1.6: Ip-level HT

Bus-level Trojans

This particular type of Hardware Trojan (HT) is associated with the network fabrics of buses, either through linkers, or integration into the modules of on-chip buses, including routing nodes, control units, and network interfaces. Activation is primarily induced by internal uncommon signals or the flow of data within the on-chip buses (see Figure 1.7).

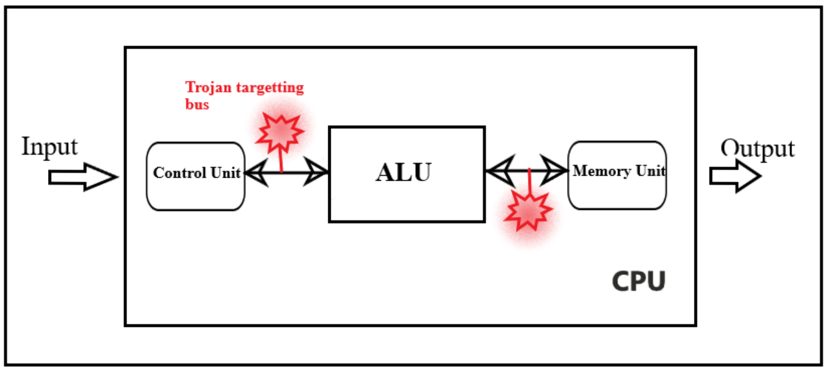


Figure 1.7: Bus-level HT

SoC-level Trojans

Moreover, the trigger conditions of this type of HT vary, e.g., rare conditions occurring within the implanted IP cores, special instructions or external sequences, etc. This type of HT is designed to affect the overall system functions rather than the infected IP cores (see Figure 1.8).

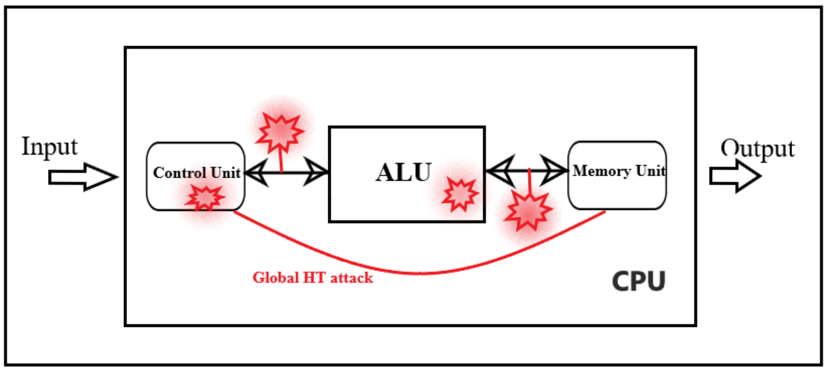


Figure 1.8: SoC-level HT

1.6 Activation Mechanism

Trojans can be categorized into two types based on their activation: those that are constantly active (always ON) and those that are triggered.

1.6.1 Always ON

An "always ON" hardware Trojan is a type of malicious circuit that remains continuously active within an integrated circuit (IC) or electronic system. Unlike triggered Trojans that activate under specific conditions, an always ON Trojan operates persistently, potentially exerting its detrimental effects without any external stimulus. The continuous activity of an always ON hardware Trojan makes it particularly challenging to detect, as it remains in an operational state, possibly compromising the integrity and security of the affected system over an extended period. Detection and mitigation strategies for always ON hardware Trojans are crucial for ensuring the reliability and trustworthiness of electronic devices.

1.6.2 Internally triggered

An internally triggered hardware Trojan is a type of malicious circuit or modification within an integrated circuit (IC) that activates based on specific internal conditions or triggers, as opposed to external stimuli. In contrast to always ON Trojans that remain continuously active, internally triggered Trojans initiate their malicious behavior in response to predetermined internal factors. These triggers could be associated with certain logic states, input patterns, counter values, or other internal conditions within the circuit.

Internally triggered hardware Trojans add a layer of complexity to their detection, as their activation may not be externally visible or easily discernible during routine testing. Detecting and mitigating internally triggered Trojans require sophisticated analysis techniques and comprehensive testing strategies to uncover their presence and prevent potential security threats within electronic systems.

There are in general two types of internally triggered mechanisms one is combinational and other is sequential.

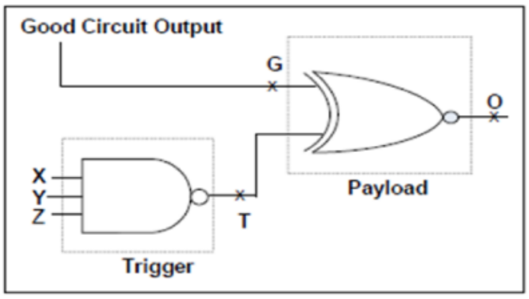
Combinational Trigger Mechanism

A combinational trigger mechanism in a hardware Trojan refers to a method by which the malicious circuit is activated based on specific conditions within the combinational logic of an integrated circuit. In this context, "combinational" denotes that the trigger mechanism relies on the instantaneous input values to determine when the Trojan becomes active.

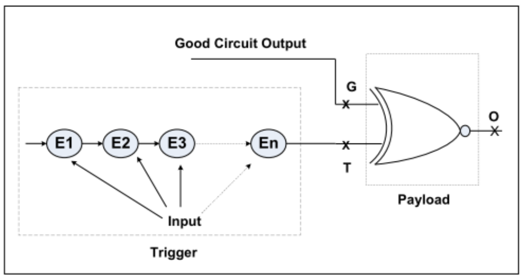
The combinational trigger mechanism involves manipulating the combinational logic gates or paths in such a way that the hardware Trojan activates when a particular combination of input values is present. This could involve introducing specific logical conditions or patterns in the circuit design, which, when satisfied, trigger the malicious behavior of the Trojan.

Compared to sequential trigger mechanisms that may rely on specific sequences of inputs over time, combinational triggers are more instantaneous, making them potentially harder to detect during functional testing. Researchers and security analysts need to develop advanced methods to identify and counter hardware Trojans with combinational trigger mechanisms to ensure the integrity and security of electronic systems.

Figure 1.9.(a) illustrates a generic representation of a hardware Trojan featuring a combinational trigger mechanism. The trigger logic will be in inactive state when node T has the value 1, the output of good circuit will appear at the same nodes O and G. the trigger logic is enabled only when X=1, Y=1, Z=1. Rare condition for trigger activation, results that the value at output node O will be complement of node G when node T has the value of 0.



(a) Combinational trigger mechanism



(b) Sequential trigger mechanism

Figure 1.9: Combinational and Sequential trigger mechanism.

Sequential Trigger Mechanism

A sequential trigger mechanism in a hardware Trojan refers to a method by which the malicious circuit is activated based on specific sequences or patterns of input values over time within the integrated circuit. In contrast to combinational triggers, which rely on instantaneous input values, sequential triggers involve a temporal aspect in their activation conditions.

The sequential trigger mechanism typically involves monitoring the state transitions of certain components, such as flip-flops or registers, within the circuit. The Trojan remains inactive until a predefined sequence of state changes occurs. This sequence might be related to specific input patterns, internal logic states, or other dynamic conditions that unfold over successive clock cycles.

A hardware counter, also known as a ticking time-bomb, serves as a straightforward sequential trigger logic. Activation of the trigger occurs when the hardware counter value reaches a predefined threshold. Implementing a basic timebomb design involves a single increment per clock cycle, making it uncomplicated to integrate into the hardware. The strength of this triggering mechanism lies in its independence from any input data, eliminating the need for software intervention to activate the hardware Trojan. However, the malicious designer must be knowledgeable about the required number of clock cycles for activation. In the case of a more intricate timebomb, the hardware counter value is not incremented per clock cycle but rather increases per occurrence of specific events, as depicted in Figure 1.9.(b).

Figure 1.10 presents a conceptual representation of a sequential triggering mechanism utilizing a hardware counter for a Hardware Trojan. In this model, the enabling input for a hardware counter can be either the clock of the design or the occurrence of a specific event within the design. The counter value increases with each transition at the enable input. The trigger logic becomes active only when all bits of the counter are set to 1, causing node T to have a value of 0. Consequently, the output node O will be the complement of node G.

The hybrid sequential trigger logic employs a combination of a hardware counter and a sequence of infrequent events to establish the trigger condition for the hardware Trojan. The intricacy of this design, concerning hardware aspects, lies in the synchronization between the hardware counter and the system software responsible for generating the sequence of rare events.

Detecting hardware Trojans with sequential trigger mechanisms poses challenges during functional testing, as the activation conditions may not be evident in isolated test scenarios. Researchers and security experts employ advanced techniques, such as dynamic analysis and specialized testing methodologies, to uncover and mitigate the risks associated with hardware Trojans utilizing sequential trigger mechanisms(see Figure 1.9.(b)) .

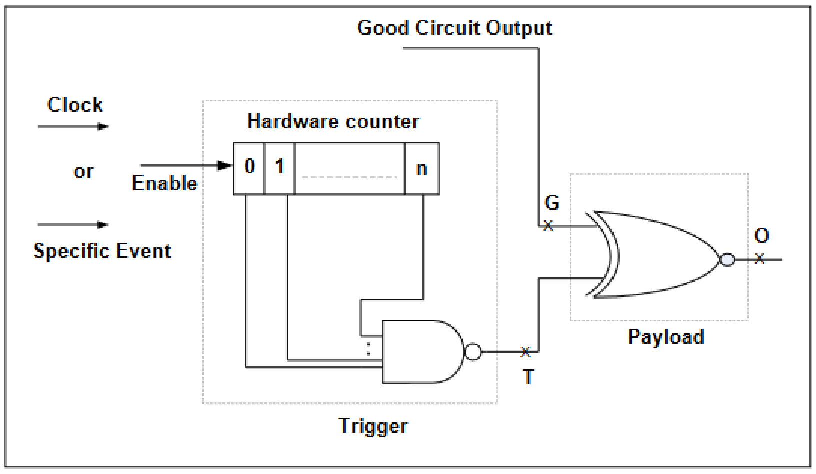


Figure 1.10: Hardware counter triggering Trojan

1.7 Trojan activation methods

Activation techniques for Trojans have the potential to expedite the process of detecting these malicious components. In certain instances, these strategies have been integrated with power analysis during the implementation phase. When a segment of the Trojan circuit is activated, it results in increased dynamic power consumption. This elevation in power consumption aids in distinguishing the power signatures between circuits with inserted Trojans and those without. The prevailing methods for Trojan activation can be classified into distinct categories.

1.7.1 Region-free Trojan Activation

These approaches do not hinge on a specific region but rather rely on the inadvertent or intentional activation of Trojans. For instance, Jha and Jha introduced a randomization-based probabilistic method for Trojan detection [Jha and Jha, 2008]. They demonstrated the feasibility of constructing a distinctive probabilistic signature for a circuit by applying specific probability-assigned patterns to its inputs. The input patterns, determined by this probability, are employed on an Identical Unit Analysis (IUA), and the resulting outputs are compared to those of the original circuit. Any disparities in the outputs signal the presence of a Trojan. When detecting Trojans in a manufactured Integrated Circuit (IC), applying patterns based on this probability helps establish a confidence level in determining whether the original design matches the fabricated chip.

Wolff et al. conducted an examination of infrequently occurring net combinations within designs [Wolff et al., 2008]. These nets, activated rarely, serve as triggers for Trojans. Simultaneously, nets with low observability are employed as payloads, as illustrated in Figure 1.11. Wolff et al. developed a series of vectors designed to activate these specific nets, proposing their integration with conventional Automatic Test Pattern Generation (ATPG)[3]test vectors. This combined approach aims to activate a

[3]ATPG s an electronic design automation method or technology used to find an input (or test) sequence that, when applied to a digital circuit, enables automatic test equipment to distinguish between the correct circuit behavior and the faulty circuit behavior caused by defects. The generated

Trojan and propagate its effects, particularly if the Trojan is linked to these identified nets.

1.7.2 Region-aware Trojan Activation

Banga and Hsiao introduced a dual-phase test generation approach aimed at amplifying distinctions between Identical Unit Analysis (IUA) and authentic design power waveforms [Banga and Hsiao, 2008]. In the initial phase involving circuit partitioning, a pattern sensitive to specific regions aids in identifying potential areas for Trojan insertion. To uncover a Trojan circuit, activity is heightened within a designated part of the circuit while simultaneously minimizing activity in the remaining portion. Flip-flops in the circuit are categorized into distinct groups based on structural connectivity. The subsequent phase involves activity magnification, where new test patterns focused on the identified regions are applied to accentuate the differences between the original circuit and the one with a Trojan inserted. Regions, defined as sets of flip-flops, displaying augmented relative activity are pinpointed by comparing power profiles using the vector sequence generated in the first stage. This phase involves generating additional vectors for these specified regions, labeled as potential Trojan areas, utilizing the same test generation approach employed in the circuit-partitioning stage.

Banga and Hsiao investigated the amplification of Trojan effects through the reduction of circuit activity [Banga and Hsiao, 2009a]. This entails maintaining the input pins unchanged for multiple clock cycles, directing the circuit activity to originate solely from the state elements of the design. Consequently, the overall switching activity is minimized and can be concentrated on specific segments of the design crucial for Trojan localization. To explore different sections of the design for Trojan identification, input vectors can be modified. Simultaneously, each gate is equipped with two counters: TrojanCount and NonTrojanCount. With each vector, if the number of transitions at a gate's output surpasses a defined threshold, the TrojanCount increases, and vice versa. The gate weight, represented by the TrojanCount/NonTrojanCount ratio, reflects a gate's activity. A high gate-weight ratio signifies significant Trojan impact on the gate, as it corresponds to a substantial power difference during the activation of that gate.

Given the unknown type or size of the Trojan, it is essential for the test engineer to employ both region-free and region-aware methods. The effectiveness of the region-aware method is notable when the inputs of a Trojan circuit are functionally dependent, originating from the same logic cone. Conversely, if the Trojan inputs are randomly selected from diverse areas of the circuit, utilizing region-free methods could enhance the likelihood of detection.

1.8 Hardware Trojan Payload

Payload part of the hardware Trojan does the intended job of Trojan designer. It is the malicious functionality that is integrated into the hardware during the design or manufacturing phase. Here are some examples of hardware Trojan payloads:

patterns are used to test semiconductor devices after manufacture, or to assist with determining the cause of failure. For more information visit (https://www.fpgakey.com/wiki/details/68)

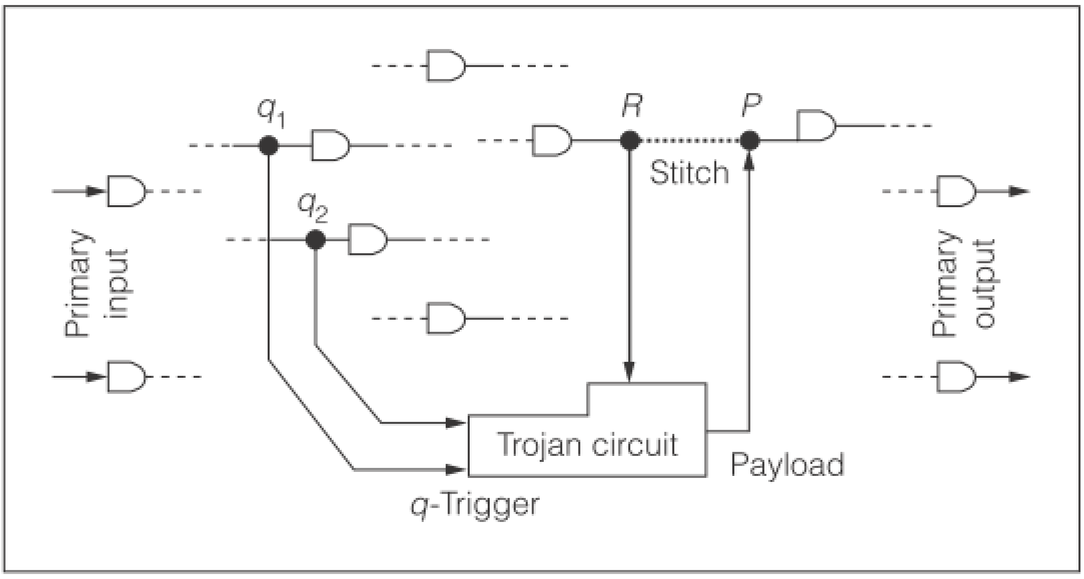


Figure 1.11: Trojan circuit model with a rare triggering condition. Note. Reprinted from "Towards trojan-free trusted ics: Problem analysis and detection scheme", by Wolff, F., (2008)

1.8.1 Data Leakage

The hardware Trojan could be designed to leak sensitive information or data from the affected system (Figure 1.12). This could involve transmitting confidential data to an external entity without the knowledge of the system owner.

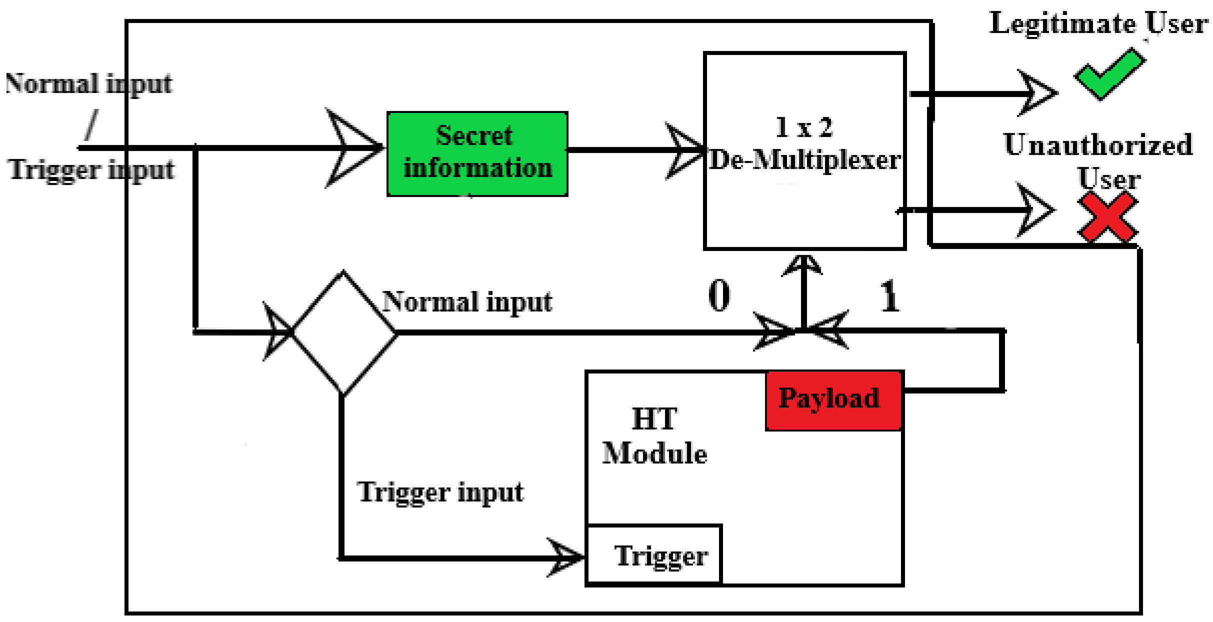


Figure 1.12: Information leakage hardware Trojan.

1.8.2 Denial of Service (DoS)

The hardware Trojan might contain a payload that disrupts the normal operation of the system or renders it unusable. This could involve triggering certain conditions that lead to a system crash or malfunction.

1.8.3 Unauthorized Access

The hardware Trojan may include a backdoor that provides unauthorized access to the system. This could allow attackers to control or manipulate the system remotely.

1.8.4 Functionality Alteration

The Trojan could modify the intended functionality of the hardware, leading to unexpected behavior. For example, it might alter the output of a cryptographic module, compromise the randomness of a random number generator, or modify the behavior of a microprocessor.

1.8.5 Triggered Malware Activation

The Trojan may be designed to remain dormant until a specific trigger condition is met, at which point it activates additional malware or malicious behavior.

1.9 Operation Based payload classification

The aim of the device attacker is to corrupt the normal operation of the entire circuit and that is ensured by the payload logic. The payload logic can be classified into two categories.

The first type of payload is designed to create new, extra processes that do not interfere with the device's regular functionality, allowing the extra work to be done discretely. The second type of payload, on the other hand, will alter the ongoing activity without producing any new ones. In order to prevent the system from crashing when changing the current operation, the malicious creator of this kind of payload needs to be aware of the current running program. The second category of payload can be further divided into two parts: payload modifying the data interface of the current operation, and payload modifying the control interface of the current operation. Figure 1.13 shows the classification of payload logic based on the operation performed when Hardware Trojan is activated.

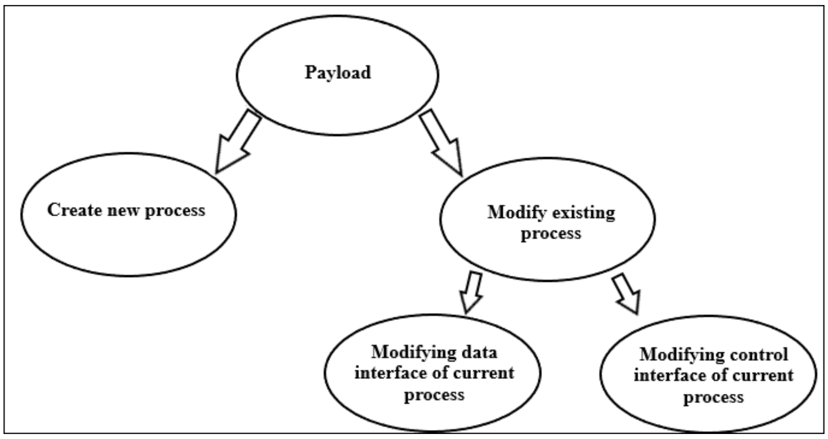

Payload
Create new process
Modify existing process
Modifying data interface of current process
Modifying control interface of current process

Figure 1.13: Operation Based payload classification.

1.9.1 Payload generates new operations

The payload circuit discreetly executes additional operations as directed by the attacker, ensuring that the regular functioning of the device remains unaffected. This is primarily employed to release confidential information and may also encompass side-channel attacks. In microprocessor-based design, one variation of this attack involves generating fresh instructions fetched from a specific address when a particular instruction is executed. Another variant is to generate additional loads or stores to a specific address whenever a specific instruction is executed. These two approaches can serve as a foundation for software-based attacks. To illustrate, both methods can be utilized to clandestinely store malicious firmware in on-chip memory, specifically within the instruction and data caches. Subsequently, the malicious software can be executed within the processor, all the while maintaining concealment from the standard software operating on the system. In alternative designs, information leakage occurs through the execution of additional tasks via interfaces like the RS-232C port or through thermal emission. This form of attack can be employed to extract encryption keys and steal passwords.

1.9.2 Payload modifies control interface for current operations

The payload circuit modifies the control interface of the design to disrupt ongoing operations, such as altering the access permissions of the current processes. This is primarily employed to elevate privileges, enabling the attacker to circumvent the standard hardware-enforced protections. A Hardware Trojan can serve as a starting point to support software-based attacks. For instance, the supervisor transition facilitated by the Hardware Trojan creates a foothold, enabling unprivileged programs to access privileged instructions and protected resources. In a microprocessor-based design, the decoder unit produces control signals for every executed instruction. In this scenario, the attack involves manipulating the control signal generated for ongoing operations, such as transforming a no-operation instruction into a load or store instruction by the decoder unit, without introducing any additional operations.

1.9.3 Payload modifies data interface for current operations

The payload circuit alters the data interface of a design to disrupt the ongoing operations. In a microprocessor-based design, this attack entails changing the data in memory access operations, modifying the address of memory access operations, altering the input data of the register file, or adjusting the address used to access the register file. This attack can also be employed to alter the order of instructions executed within a program by tampering with specific register values. In this context, a Hardware Trojan can serve as a foundation to support software-based attacks. For instance, altering the address of the current memory access operation method enables access to arbitrary memory locations. Consequently, this can serve as a basis for unprivileged malicious software to circumvent the protections enforced by the memory management unit. This attack enables the extraction of the encryption key from memory, bypassing authentication checks by modifying the content of a specific memory location, and disrupting the program flow of the system.

In summary, the most straightforward approach to implement is having the payload circuit conduct additional operations discreetly, as this method does not disrupt the regular program flow of the system. The primary rationale is that this method can be easily concealed during the normal operation of the system compared to other approaches.

1.10 Hardware Trojan architecture and design

There are two main parts in HT decomposition: the payload and the trigger. The payload is considered to be the main function that performs nefarious operations in the target circuit. The trigger is the function that serves as mechanism to activate the payload when certain activation condition will be satisfied. This strategy helps to make the HT stealthier during the verification and validation steps. Most of the time, the HT model is a dormant circuit that when triggered it modifies the targeted system original behavior (see Figure 1.14).

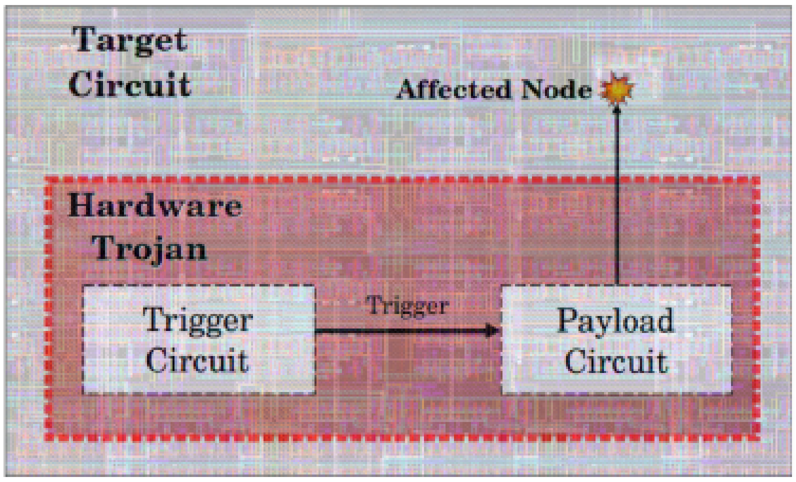


Figure 1.14: Architecture of a Trojan inserted on a target circuit.

1.10.1 Combinational Trojans

A combinational hardware Trojan refers to a specific type of malicious modification in the design of an integrated circuit (IC) at the combinational logic level. Combinational logic is a fundamental building block in digital circuits where the output is solely determined by the current input values, with no consideration of previous inputs or outputs. In the context of hardware Trojans, the term "combinational" indicates that the malicious modification occurs within this type of logic.

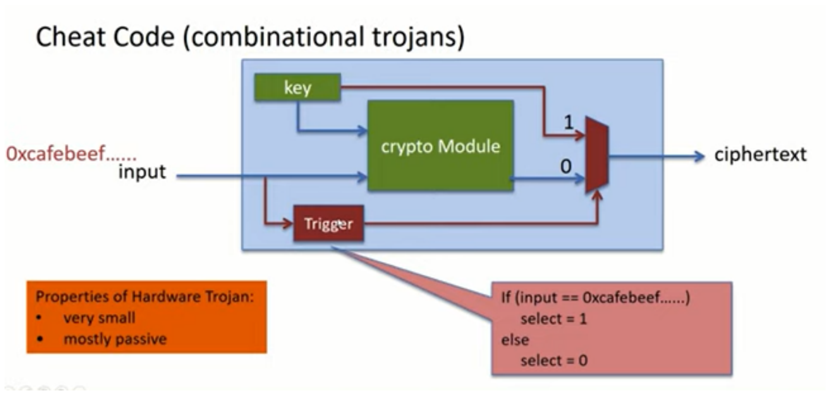
They specifically target the combinational logic components of integrated circuits. These are areas where the output depends only on the current input values, with no regard for previous inputs or outputs. These Trojans can be strategically inserted at various points within the combinational logic circuit during the design process. Common insertion points include specific gates, wires, or nodes where the Trojan can be inconspicuously integrated.

Combinational Hardware Trojans typically involve subtle alterations to logic gates. This may include modifying gate parameters, such as delay or threshold voltage, to introduce malicious behavior without raising suspicions during conventional testing.

This type of Trojans relies on logic gates, for that a trigger that is taken from the primary inputs of a circuit and a payload that can be activated once trigger is asserted. The trigger can be designed from an AND gate with p-inputs. Any other combinational logic can also serve the purpose of trigger, which produces logic 1 upon activation, such combinational Trojans delivers the payload in the original netlist and manifests its effects once a unique specification condition is satisfied (see Figure 1.15). The most important thing for such type of Trojans to be effective is that it should not come across any condition that activates the Trojan during functional tests.

Trojans in the combinational logic stage may manipulate the output of the circuit under certain conditions. This could involve altering the logic values produced by the affected gates, potentially leading to unauthorized information leakage or system disruption. Besides,They aim to bypass security measures by exploiting vulnerabilities in the combinational logic stage. This enables them to avoid detection and infiltrate systems, making them a potent threat to overall semiconductor security.

Due to their subtle nature, Combinational Hardware Trojans are often challenging to detect through traditional functional testing methods. Their activation may be triggered by specific conditions, making them dormant during routine testing scenarios.



```
If (input == 0xcafebeef......)
   select = 1
else
   select = 0
```

Figure 1.15: Combinational Trojan

1.10.2 Sequential Trojans

The mechanism of a sequential hardware Trojan involves the insertion of malicious modifications into the sequential logic elements of an integrated circuit (IC) or hardware design. The objective is to manipulate the behavior of the sequential logic in a way that compromises the security or functionality of the system.

Sequential Trojans activate their payload either when a specific sequence of input patterns occurs or after a designated time period has passed since being triggered. The triggering mechanism of a sequential Trojan incorporates state elements along with combinational logic, as illustrated in Figure 1.16. The payload becomes effective only when the Finite State Machine (FSM) of the trigger reaches its final state. This characteristic adds to the complexity of detecting sequential Trojans since it is improbable for particular test patterns or inputs to occur consecutively multiple times during the testing or normal operations of an Integrated Circuit (IC).

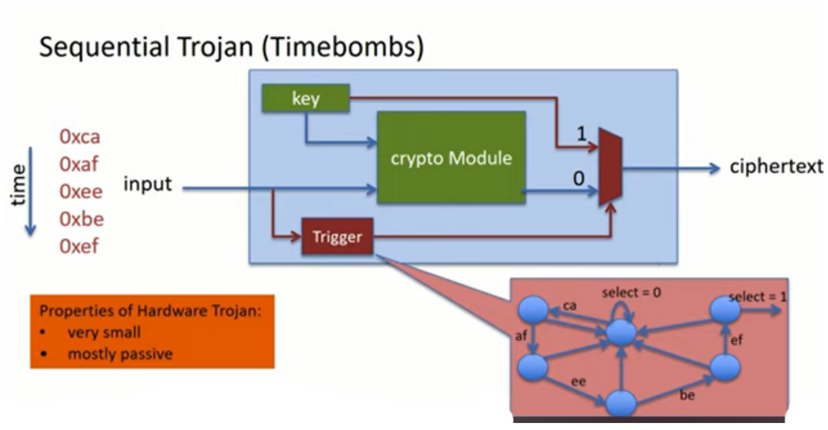


Figure 1.16: Sequential Trojan (Timebombs).

1.10.3 Analog/RF Trojans

In designing hardware Trojans, adversaries can exploit analog characteristics, as discussed in [Ghandali et al., 2016]. The trigger implementation varies for analog/RF Trojan designs, with Yang et al. [Yang et al., 2016a] proposing a capacitor-based trigger circuit. This circuit activates when the charge accumulated from the toggling of a nearby victim wire surpasses a certain threshold, causing the voltage of the capacitor to rise.

In [Subramani et al., 2020], a threat scenario has been presented. In this scenario, Device_1 and Device_2 are two wireless devices adhering to established standards and engaged in legitimate communication. Unbeknownst to Device_1, the wireless hardware mechanism has been compromised by an attacker who introduced a hardware Trojan circuit. The focal point of this malicious component is located in the analog/RF front-end of Device_1's, systematically altering transmission power to illicitly extract confidential information. Simultaneously, Device_3, a third wireless device representing the rogue receiver, monitors these systematic distortions and captures the leaked data (Figure 1.17).

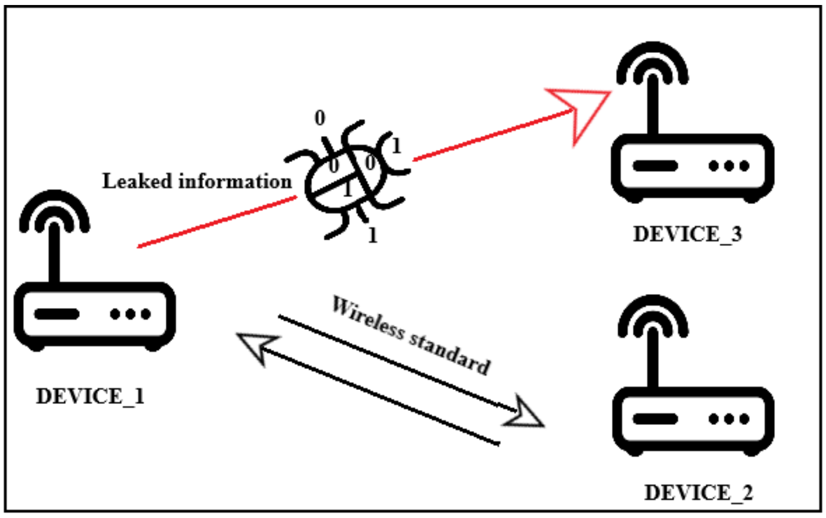


Figure 1.17: Threat model

The threat model posits that the hardware Trojan was implanted either during the design phase or during IC fabrication of the transmitter and can be activated once the device is in use. The information leaked through a covert channel, such as an encryption key, plaintext, or other sensitive data, resides in the baseband part of the wireless device. This information is then directed to the analog/RF front-end—where the attack occurs—via additional malicious modifications, as illustrated in Figure 1.18. Consequently, the leaked information bits become embedded in the transmitted signal through subtle amplitude modifications.

It's noteworthy that an analog Trojan can be conceptualized as a specific type of sequential Trojan, requiring multiple triggers or affecting the circuit after a set period. The primary distinction lies in the trigger design: sequential Trojans involve state elements (e.g., counters), while analog Trojans involve discrete elements (e.g., transistors and capacitors). Additionally, both sequential and analog Trojans can be modeled using combinational Trojans.

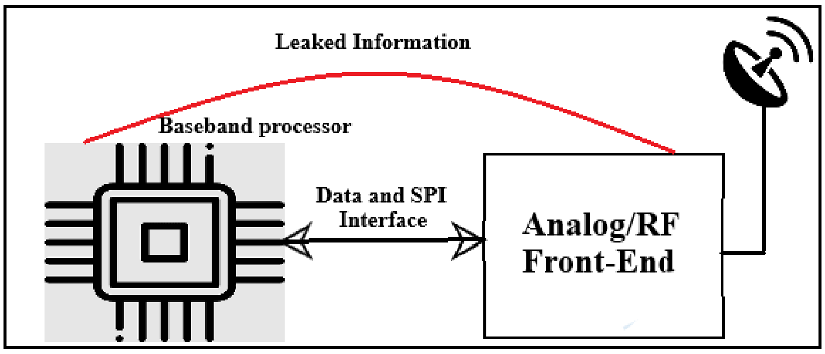


Figure 1.18: Amplitude modulating hardware Trojans.

1.10.4 Piggybacking

- Insertion into IP Cores: Adding Trojan functionality within existing intellectual property (IP) cores.
- Malicious Components: Including Trojan components within the design alongside legitimate components.

1.10.5 Fault-based Trojans

- Fault Injection: Inducing faults to trigger malicious behavior when specific conditions are met.
- Fault Tolerance Exploitation: Exploiting the fault tolerance mechanisms to trigger Trojans.

1.11 Location of hardware Trojans

Trojans might operate at different levels on the IC. According to that, they are also classified regarding to their location in the design, which leads to multiple possible attacks.

1.11.1 Processor

A hardware trojan in a processor represents a potentially severe security threat that involves the insertion of malicious modifications into the design or manufacturing process of the processor. This kind of trojan can compromise the integrity, security, and functionality of the processor in various ways. Here are some aspects of how a hardware trojan might manifest in a processor:

1. **Data Manipulation:** A hardware trojan in a processor could manipulate data as it passes through different stages of execution. This might lead to unauthorized access, data corruption, or the extraction of sensitive information.

2. **Instruction Manipulation:** Trojans might alter the instructions executed by the processor, leading to unexpected and potentially malicious operations. This could compromise the overall functionality of the processor and the systems it powers.

3. **Performance Impact:** Hardware trojans could introduce subtle changes that impact the performance of the processor. This might include slowing down specific operations or degrading overall processing speed.

4. **Backdoor Insertion:** One of the severe consequences of a hardware trojan in a processor is the potential introduction of a hidden backdoor. This backdoor could provide unauthorized access to the system, allowing external entities to control or manipulate the processor.

5. **Security Key Leakage:** Processors often handle encryption and decryption tasks, and a hardware trojan might be designed to leak cryptographic keys. This compromises the security of encrypted communications and data.

6. **Denial of Service:** Trojans could be programmed to trigger a denial of service by disrupting the normal operation of the processor. This might render the entire system or specific functionalities inoperable.

1.11.2 Memory

A hardware trojan in memory represents a serious security concern where malicious modifications are introduced into the design or manufacturing process of memory components. Memory devices, such as RAM (Random Access Memory) or non-volatile memory, play a critical role in storing and retrieving data in electronic systems. Here are some potential implications of a hardware trojan in memory:

1. **Data Corruption or Manipulation:** A hardware trojan in memory could manipulate stored data, leading to corruption or unauthorized access. This might result in the compromise of sensitive information stored in the memory.

2. **False Readings:** Trojans may introduce errors in memory read operations, providing false or altered data to the processor. This can impact the accuracy and reliability of the information retrieved from the memory.

3. **Data Leakage:** Trojans might be designed to leak sensitive data from the memory, compromising the confidentiality of stored information. This could include personal data, encryption keys, or other critical data.

4. **Denial of Service:** Introducing a hardware trojan into memory could lead to disruptions in memory access or cause the memory to become unusable. This could result in a denial of service, affecting the normal operation of the entire system.

5. **Altered Access Control:** Hardware trojans could modify the access control mechanisms of the memory, allowing unauthorized entities to read, write, or manipulate data stored in the memory.

6. **Persistent Malicious Code:** In non-volatile memory (such as Flash memory), trojans might embed persistent malicious code that survives power cycles. This could lead to the execution of unauthorized instructions or actions every time the system boots.

7. **Compromised Security Protocols:** Memory is often involved in storing security-related information and cryptographic keys. Trojans targeting memory could compromise the security protocols implemented in the system.

1.11.3 I/O

One potential target of hardware trojans could be the I/O (Input/Output) pins of a device. I/O pins are crucial for communication between the IC and external components or systems. Here are some ways I/O pins might be affected by hardware trojans:

1. **Data Manipulation:** A trojan could alter the data being transmitted through the I/O pins, leading to data corruption or unauthorized access.

2. **Signal Manipulation:** Hardware trojans might modify the signal characteristics on the I/O pins, affecting the integrity of communication protocols and potentially causing communication errors.

3. **Key Leakage:** Trojans may be designed to leak cryptographic keys or sensitive information through the I/O pins, compromising the security of the system.

4. **Functionality Alteration:** I/O pins are used to control various functions of a device. Hardware trojans could change the functionality of these pins, leading to unexpected behavior.

5. **Denial of Service:** Trojans might be programmed to disrupt or disable the I/O functionality, leading to a denial of service or rendering the device inoperable.

6. **Backdoor Insertion:** A trojan could introduce a hidden backdoor accessible through specific I/O configurations, allowing unauthorized access to the system.

7. **Sensor Manipulation:** If the I/O pins are connected to sensors, trojans might manipulate sensor readings, leading to incorrect data being processed by the system.

1.11.4 Power supply

A hardware trojan in the power supply of a system represents a unique and potentially devastating threat. The power supply unit (PSU) is a critical component responsible for providing the necessary electrical power to the various components of a system. A trojan affecting the power supply can have far-reaching consequences on the entire system's functionality and security. Here are some potential implications:

1. **Voltage Manipulation:** A hardware trojan in the power supply may alter the voltage levels supplied to different components. Incorrect voltage can lead to erratic behavior, malfunctions, or permanent damage to sensitive electronic components.

2. **Brownouts or Blackouts:** Trojans might induce intentional voltage drops (brownouts) or complete power shutdowns (blackouts), causing disruptions to the normal operation of the system. This can lead to data loss, system crashes, or denial of service.

3. **Frequency Manipulation:** The trojan may manipulate the frequency of the power supplied to the system. Deviations in frequency can impact the synchronization of components, potentially leading to performance degradation or malfunctions.

4. **Electromagnetic Interference (EMI):** Hardware trojans might introduce intentional electromagnetic interference in the power supply, affecting the electromagnetic compatibility of the system and potentially disrupting nearby electronic devices.

5. **Power-Related Attacks on Security Mechanisms:** Trojans could target the power supply to conduct power analysis attacks, which involve analyzing power consumption patterns to extract sensitive information like cryptographic keys. This poses a threat to the security of the system.

6. **Remote Triggering of Power-Related Vulnerabilities:** In the context of remote attacks, trojans in the power supply might exploit vulnerabilities remotely, leading to malicious actions such as power cycling, shutdowns, or manipulation of power-related functions.

7. **Circuitry Manipulation:** Trojans may tamper with the internal circuitry of the power supply, allowing for backdoor access or control by external entities. This could potentially compromise the overall security of the system.

1.11.5 Clock

A hardware trojan in the clock, often referred to as a clock trojan, can pose a significant threat to the proper functioning and security of a digital system. The clock is a fundamental component in electronic systems, providing synchronization for various operations. Here are some potential implications of a hardware trojan in the clock:

1. **Clock Frequency Manipulation:** A clock trojan might manipulate the frequency of the system clock. This can lead to performance issues, disrupt timing-sensitive operations, or create synchronization problems between different components.
2. **Clock Skewing:** Trojans could introduce intentional delays or skewing in the clock signal. This can affect the alignment of signals, leading to misbehavior or unexpected outcomes in the operation of the digital system.
3. **Clock Gating Control:** Hardware trojans might manipulate the control signals for clock gating, affecting the power consumption of the system. Unauthorized control of clock gating could lead to inefficient power usage or unintended activation/deactivation of specific circuitry.
4. **Introduction of Jitter:** Trojans may introduce jitter into the clock signal, causing irregularities in the timing of operations. Jitter can impact the reliability of communication protocols and introduce vulnerabilities.
5. **Clock Domain Crossing Violations:** Intentional violations of clock domain boundaries can be introduced by trojans, leading to issues related to data integrity and synchronization between different clock domains within the system.
6. **Frequency Modulation Attacks:** Sophisticated trojans might employ frequency modulation attacks, dynamically changing the clock frequency during specific intervals. This can make detection challenging and potentially evade traditional security measures.
7. **Synchronization Attacks:** Clock trojans might aim to desynchronize multiple components within a system, disrupting the coordinated operation of these components and potentially causing malfunctions.
8. **Cryptographic Timing Attacks:** In systems employing cryptographic algorithms, clock trojans might be designed to exploit timing variations introduced by clock manipulation. This could potentially lead to the extraction of sensitive cryptographic information.

1.12 Effects of hardware Trojans

Different effects of a HT on the device that depends on the adversary possibilities and intentions. The following, are some of those effects.

1.12.1 Change function

original circuit functions might be altered depending on adding or removing several instructions. That leads to an improper calculations under specific conditions, compromising the main system operations. For example, a miscalculated airstrike that leads to hit a completely wrong target.

1.12.2 Reduce reliability

downgrading system performance and rendering it more vulnerable to side-channel attacks are the most intended goals by an adversary. For example, Trojan may increase the power consumption, causing a faster battery discharge to interrupt the circuit operation.

1.12.3 Leak information

keys and plaintexts of cryptographic circuits are the most targeted information to be leak. An adversary could add a comparator Trojan that enables the key leakage whenever a certain input or sequence of outputs is set.

1.12.4 Denial of service (DoS)

The entire circuit might be put out of service. For instance, much load on a military radar system may lead it to not detect some specific threats.

1.13 Comparison of some hardware Trojans attacks

Different hardware Trojan attacks can vary significantly in their techniques, goals, and potential impact. Here is a comparison of some common types of hardware Trojan attacks.

1.13.1 Logic Trojans vs Analog Trojans

- Logic Trojans: These involve inserting additional logic gates or modifying existing ones to achieve a specific malicious function. Logic Trojans are often digital in nature and can be triggered by specific conditions.

- Analog Trojans: Analog Trojans typically target the analog components of a circuit, such as resistors and transistors, to subtly alter the behavior of the circuit. They can be harder to detect due to the continuous nature of analog signals.

1.13.2 Design-Level Trojans vs. Piggybacking

- Design-Level Trojans: These involve making modifications at the gate or netlist level, often requiring a deep understanding of the target design. Design-level Trojans can be strategically placed to avoid detection.

- Piggybacking: Involves adding Trojan functionality within existing IP cores or components. Piggybacking can leverage legitimate functionality to mask malicious behavior.

1.13.3 Fault-based Trojans vs. Physical Attacks

- Fault-based Trojans: These Trojans exploit faults or errors in the circuit, such as induced glitches or timing violations, to trigger malicious behavior. They can be activated under specific conditions.

- Physical Attacks: Involve direct manipulation of the hardware, such as doping changes or optical masking during the fabrication process. Physical attacks can be more challenging to implement but can have a significant impact.

1.13.4 Supply Chain Attacks vs. Stealth Techniques

- Supply Chain Attacks: These involve introducing Trojans during the manufacturing process, either at the foundry or fabrication plant. This type of attack can be orchestrated at various stages of the supply chain.
- Stealth Techniques: Focus on making the Trojans difficult to detect. Techniques such as camouflaging and probabilistic activation are designed to evade traditional testing and analysis methods.

1.13.5 Side-Channel Attacks vs. Time-based Activation

- Side-Channel Attacks: Involve analyzing unintended emanations from a device, such as power consumption or timing variations, to deduce information about the Trojan's behavior.
- Time-based Activation: Specifies a particular time or number of clock cycles for the Trojan to activate. This can make the Trojan more challenging to detect, as it may remain dormant until specific conditions are met.

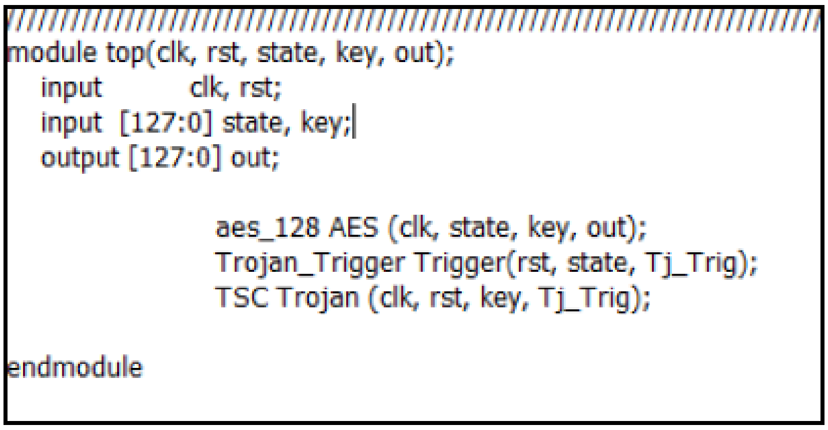
1.13.6 Environmental Activation vs. Parameter Variation

- Environmental Activation: Involves triggering the Trojan based on external conditions such as temperature or voltage levels. This can be used to make the Trojan behavior context-dependent.
- Parameter Variation: Modifies parameters of components, such as resistor values or transistor sizes, to subtly alter the behavior of the circuit. Parameter variation can be harder to detect due to its subtle nature.

Understanding the characteristics of different hardware Trojan attacks is crucial for developing effective detection and mitigation strategies. Combining various countermeasures, including testing, formal verification, and supply chain security measures, is often necessary to enhance the overall resilience of hardware systems against Trojan threats.

1.14 Case study (Trojan in an AES-128)

The Trust-Hub (https://www.trust-hub.org), sponsored by the USA National Science Foundation (NSF). It is a source of benchmarks infected by different types of Trojans at diverse abstraction levels in order to support the security community to compare the behavior of Trojan-free and Trojan-infected circuits to evaluate HT detection methods. A set of benchmarks are available, considering different categories in the Trojan taxonomy.



```
////////////////////////////////////////////////////////////////////////
module top(clk, rst, state, key, out);
   input        clk, rst;
   input  [127:0] state, key;
   output [127:0] out;

                  aes_128 AES (clk, state, key, out);
                  Trojan_Trigger Trigger(rst, state, Tj_Trig);
                  TSC Trojan (clk, rst, key, Tj_Trig);

endmodule
```

Figure 1.19: Architecture of the Trojan-infected AES-T2000 benchmark

The AEST2000 benchmark consists of a Trojan surrounding an AES128 encrypting block with the purpose of leaking its secret key. This HT is triggered after detecting a sequence of 4 input plaintexts. The payload is responsible for generating a significant current leakage indicating the actual state of each bit of the secret key. An attacker with access to the manufactured device can therefore exploit its current leakage to have access to the key. The code in (see Figure 1.19) illustrates the whole process. The library aes128 is the original AES circuit, whilst Trojan_Trigger and TSC Trojan are respectively the trigger and the payload circuit.

The Trojan classification according to the Trojan taxonomy is:

- Insertion phase: Design
- Abstraction level: Register Transfer level
- Activation mechanism: Internally conditionally triggered
- Effects: Leak information
- Location: Processor
- Physical characteristics: Functional

1.15 Conclusion

In conclusion, this chapter has provided an in-depth exploration of the general concepts surrounding Hardware Trojans. It commenced with an introduction to the subject, followed by a definition and an examination of the threats posed by Hardware Trojans in various domains, such as the IC market, SoC designers, foundries, IP vendors, and end users.

The discussion then delved into the critical fields affected by Trojans, including military, banking and finance, nuclear centers, and drilling rigs. A thorough classification and taxonomy of Trojans were introduced, drawing upon contributions from various research studies. Furthermore, the activation mechanisms of Hardware Trojans were explored, including the "Always ON" and "Internally triggered" methods.

The chapter proceeded to investigate Trojan activation methods, distinguishing between region-free and region-aware activation. It also delved into the diverse payloads of Hardware Trojans, encompassing data leakage, denial of service (DoS), unauthorized access, functionality alteration, and triggered malware activation. Operation-based payload classification was introduced, categorizing payloads based on whether they generate new operations, modify control interfaces, or alter data interfaces for current operations.

The architecture and design of Hardware Trojans were thoroughly examined, covering various types such as combinational, sequential, analog/RF, piggybacking, and fault-based Trojans. The chapter also explored the potential locations of Hardware Trojans within a system, including processors, memory, I/O, power supply, and clocks.

A comprehensive overview of the effects of Hardware Trojans was presented, discussing their potential to change function, reduce reliability, leak information, and cause denial of service (DoS) incidents. A comparative analysis of different Hardware Trojan attacks was provided, contrasting logic Trojans with analog Trojans, design-level Trojans with piggybacking, fault-based Trojans with physical attacks, supply chain attacks with stealth techniques, and side-channel attacks with time-based activation.

The chapter concluded with a practical case study involving a Trojan in an AES-128, offering a real-world illustration of the discussed concepts. In summary, this chapter has laid a strong foundation for understanding the multifaceted nature of Hardware Trojans, their activation mechanisms, diverse payloads, architectural considerations, and potential impact on different systems and industries.

Chapter 2

Hardware Trojan Insertion/Detection Principle Approaches and Tools

2.1 Introduction

In the contemporary landscape of electronic systems and integrated circuits, the increasing complexity and interconnectedness have given rise to unprecedented challenges in ensuring the security and trustworthiness of these critical components. One particularly insidious threat that has garnered significant attention is the insertion of Hardware Trojans (HTs) - clandestine modifications in the hardware design or manufacturing process that compromise the functionality, reliability, or security of the targeted system. As electronic devices play an integral role in our daily lives, ranging from critical infrastructure to personal communication devices, the detection and prevention of Hardware Trojans have become paramount in safeguarding the integrity of these systems.

This chapter explores the sophisticated domain of Hardware Trojans, specifically focusing on the insertion and detection approaches employed in countering this sophisticated threat.The chapter checks not only the techniques employed by adversaries to stealthily introduce Trojans into hardware but also the evolving strategies and technologies designed to identify and mitigate these malicious insertions.Central to this exploration is a comprehensive examination of the software and hardware materials utilized in both the perpetration and defense against Hardware Trojans.

2.2 Insertion Approaches

HT insertion is considered possible at any design phase of an integrated circuit since not all chip third-party suppliers are to be trusted. The first crucial stage in embedded systems design is to gather and analyze the product requirements and turn them into specifications. This phase is the most trusted since suspicious constraints are in the form of text, which eases the process of debugging and checking without any advanced test mechanism.

Whereas, during the design phase, register-transfer level (RTL) is a design abstraction that models a synchronous digital circuit in terms of the flow of digital signals (data) between hardware registers and the logical operations performed on those signals. Register-transfer-level abstraction is used in hardware description languages (HDLs) like Verilog and VHDL to create high-level representations of a circuit, from which lower-level representations and ultimately actual wiring can be derived. For that, untrusted third-party IPs and codes might be used, which would put the entire design at risk.

Logic synthesis is a process by which an abstract specification of desired circuit behavior, typically at register transfer level (RTL), is turned into a design implementation in terms of logic gates, typically by a computer program called a synthesis tool. Common examples of this process include the synthesis of designs specified in hardware description languages, including VHDL and Verilog. Some synthesis tools generate bitstreams for programmable logic devices such as PALs or FPGAs, while others target the creation of ASICs. Besides Place and route is composed of two steps: placement and routing. The first step, placement, involves deciding where to place all electronic components, circuitry, and logic elements in a generally limited amount of space.

This is followed by routing, which determines the exact design of all the wires needed to connect the components. This step must implement all the desired connections while following the rules and limitations of the manufacturing process. At this level of design all related libraries, tools, standard cells, and third-party hard IPs are to be properly certified, so that this step of design would be considered trusted.

Verification step is intended to check that a product, service, or system meets a set of design specifications. In the development phase, verification procedures involve performing special tests to model or simulate a portion, or the entirety, of a product, service, or system, then performing a review or analysis of the modeling results. In the post-development phase, verification procedures involve regularly repeating tests devised specifically to ensure that the product, service, or system continues to meet the initial design requirements, specifications, and regulations as time progresses. It is a process that is used to evaluate whether a product, service, or system complies with regulations, specifications, or conditions imposed at the start of a development phase. In the contrary with the previous phases, verification step typically performed in the local foundry, which qualify it to be trusted since the use of certified tools and testbenches.

At the fabrication phase, third-party mask shops have access to genuine stream files (GDSII, OASIS), which might be an easy access to system applications and mess with genuine design. The most common attack at the assembly and package phase is to modify authentic hardware components during chip integration and replace them by malicious ones.

In the end, the post-silicon test and validation phase will be safer if it is performed locally in the concerned factory or by another fully certified company instead of outsourcing from an untrusted third-party one, and this phase is considered the last to detect Trojans before delivering the IC for deployment.

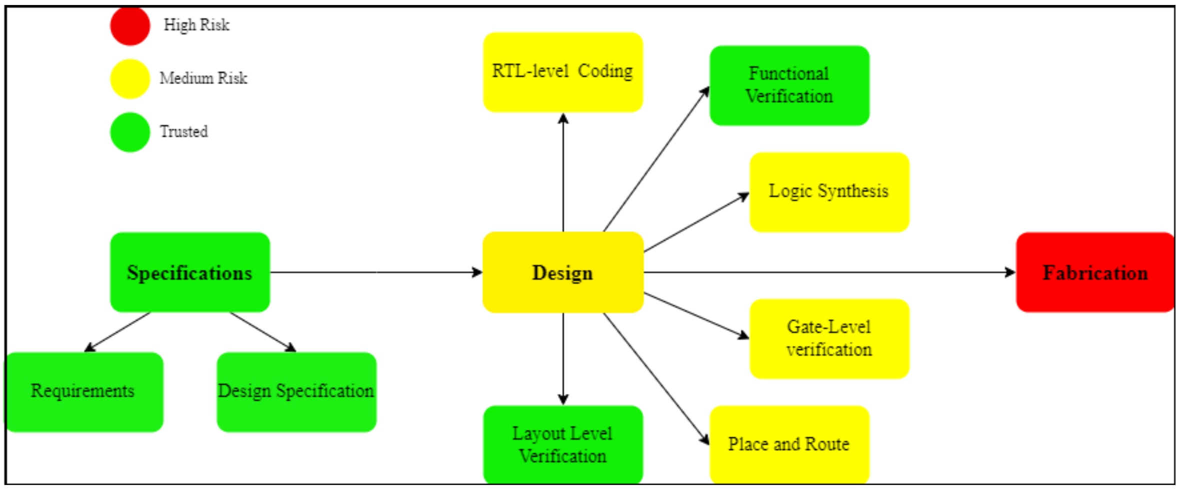


Figure 2.1: Hardware trojan insertion in circuit design phases

To assess the risks associated with Hardware trojans (HTs), various studies, such as [?] , have presented taxonomies. These taxonomies abstract different categories related to the architecture, effects, and insertion of Trojans in Integrated Circuits (ICs).

2.2.1 Insertion phase

This classification delineates various stages within the IC design process where a potential adversary might be situated. The subsequent analysis outlines the susceptibilities to Trojan insertion at each phase:

Specification: An adversary could intentionally establish weak requirements for the system. This could compromise design reliability, rendering the device susceptible to the leakage of sensitive information.

Design: Even if the entire design is conducted in-house, the use of untrusted tools, libraries, third-party IPs, and standard cells may adversely impact it. For example, untrusted tools might introduce additional circuitry to create backdoors in the authentic design. If any aspect of the design phase is outsourced, a Trojan could be directly incorporated into the hardware description files of the genuine circuit.

Fabrication: A foundry, mask shop, or their staff members who are not trustworthy may gain entry to authentic circuit components, enabling them to anticipate the circuit's functioning and possible uses. This makes the design susceptible to the insertion or deletion of components. Furthermore, modifying the physical characteristics of the circuit, such as sizes and channel doping concentration levels, can significantly increase the susceptibility of the circuit to attacks based on faults.

Assembly and package: The IC is enclosed in a protective case, and the packaged chip is assembled on a PCB with other hardware. An adversary may introduce malicious hardware components around the genuine design to induce malfunctions or increase leakages.

Post-silicon test: During the testing phase, an adversary can no longer modify the genuine circuit structure. However, the test setup, programs, or reports may be altered to mask potential Trojan effects. Additionally, as the final step in the IC design process, this phase represents the last opportunity for original designers to detect Trojans before the deployment stage.

2.2.2 Abstraction Level

The abstraction level pertains to the potential manipulation of the design in the event that an adversary gains access to sensitive files across various abstraction levels. The ensuing examination highlights opportunities for HT insertion at each abstraction level.

System level: A Hardware Trojan (HT) can involve modifications to functional specifications, protocols, interfaces, and constraints within the authentic design. If an adversary operates at the system level, they might introduce ambiguous specifications to gain control over confidential data transmitted by the manufactured device. For example, during the specification phase, an adversary could manipulate the specifications of true random number generators (TRNG), causing them to operate predictably under specific conditions known only to the HT owner. Such alterations have the potential to significantly compromise the reliability of secure systems relying on these architectures, enabling attackers to access sensitive information.

Development environment level: Tools and scripts that are not trusted may contain concealed functionalities, causing designers to create circuits that are compromised by Trojans. Moreover, untrusted simulation tools and testbenches have the potential to obscure Hardware Trojan (HT) effects. Any untrustworthy third-party vendor could introduce Trojans at this particular level.

Register-transfer level: A Hardware Trojan (HT) can manifest as straightforward alterations in authentic Register-Transfer (RT) level codes or constraint files. An adversary may manipulate circuit functions to induce significant outcomes, such as failures in cryptographic blocks. Potential sources for HT insertion at this level include attackers during the design phase or an untrusted supplier of code.

Gate level: The inclusion or removal of one or more gates within the initial netlist is classified as a gate-level Hardware Trojan (HT). The standard delay format (SDF) files, which encompass system timing data, can also be altered to modify timing checks, constraints, and delays, concealing the effects of the HT. Adversaries operating during the gate design phase and third-party vendors possess the capability to introduce Trojans at this specific level.

Transistor level: The introduction of additional transistors can substantially raise leakages, providing attackers with insights into the internal states of security-focused circuits. Additionally, the incorporation of transistors may be employed to extend critical path delays, causing malfunctions in the circuit. Possible origins of Trojans at this level include adversaries during the design phase or the utilization of untrustworthy tools, libraries, and models.

Physical layout level: The initial parameters of circuit components remain susceptible even following layout generation. An example is an attacker manipulating original masks, modifying transistor dimensions like lengths, widths, or channel doping concentrations. Additionally, resizing wires can cause malfunctions and increased leakages. Adversaries operating at both the design and fabrication levels, as well as third-party mask shops, have the capability to modify the original layout and introduce such Trojans.

2.2.3 Some employed strategies in the insertion of HTs

In emulating the strategies employed by adversaries in the insertion of hardware Trojan horses (HTH), Alkabani and Koushanfar categorized the necessary components into three groups: trigger, storage, and driver. A trigger serves to activate the intended HTH, and following the trigger event, the ensuing action can be recorded in either memory or a sequential circuit. The driver is responsible for executing the action prompted by the trigger. Based on this classification, Alkabani and Koushanfar propose a systematic approach for embedding hardware Trojans into integrated circuits (ICs) through pre-synthesis manipulation of the circuit's structure [Alkabani and Koushanfar, 2008]. This model addresses trust concerns related to Intellectual Property (IP) cores, especially when multiple cores from various vendors are utilized. In an abstracted view of the design process, the Trojan designer formulates a high-level design description to ascertain the computation model of the circuit that can be represented by a finite-state machine (FSM).

Tiago D. Perez and Samuel Pagliarini have presented a Hardware Trojan Insertion in Finalized Layouts [Perez and Pagliarini, 2023]. Their work provides the first-ever documentation of how effortlessly an HT can be incorporated into a finalized layout. This is achieved by introducing an insertion framework based on the engineering change order flow. To validate their findings, they have constructed an ASIC prototype using 65-nm CMOS technology, featuring four trojaned cryptocores. In each core, a side-channel HT is inserted with the goal of leaking the cryptokey over a power channel.

Yu Liu et al. have presented a wireless cryptographic IC containing two hardware Trojans capable of leaking the encryption key [Liu et al., 2017]. Using silicon measurements from 40 chips fabricated in Taiwan Semiconductor Manufacturing Company's (TSMC's) 0.35-μ m technology, they demonstrate the operation of two hardware Trojans. These Trojans are designed to leak the secret key of a wireless cryptographic integrated circuit (IC) that includes an Advanced Encryption Standard (AES) core and an ultrawideband (UWB) transmitter (TX). Their impact is carefully concealed within the transmission specification margins allowed for process variations. These hardware Trojans evade detection by production testing methods for both the digital and analog parts of the IC, and they do not violate the transmission protocol or any system-level specifications. However, an informed adversary who knows what to look for in the transmission power waveform can retrieve the 128-bit AES key, leaked with every 128-bit ciphertext block sent by the UWB TX.

Exurville et al., have described the structure of created HTs and how they are inserted at layout level in FPGA [Ngo et al., 2015b]. The attack scenario involves an untrusted ASIC foundry. When the tape-out database (GDS file) is received, the perpetrator introduces a Hardware Trojan (HT) before the fabrication process. To minimize the impact of the HT on the authentic circuit, it is crucial to insert the HT while preserving the original placement and routing of the target circuit. For simulating HT insertion on ASICs, maintaining identical placement and routing between the golden circuit and the HT-infected circuit on FPGAs is essential. Consequently, the only disparities between the two lie in the logic and interconnect used by the HT.

To insert an HT without altering the routing, the following steps are executed within the Xilinx framework:

1. Synthesize, translate, map, place and route the original circuit, which, in this case, is an AES-128 block cipher.
2. Extract the Native Circuit Description (NCD) file containing all the circuit, placement, and routing details of the original circuit (the golden model).
3. Utilize the FPGA Editor tool to open the NCD file and manually or through a script, insert the HT in unused LUTs and Slices of the FPGA.
4. Generate bit files for both the original and infected circuits using FPGA Editor.

This method ensures that the placement and routing remain consistent in both the golden and HT-infected circuits, facilitating the proof-of-concept of the HT attack at the ASIC layout level, where the FPGA fabric is considered as an ASIC.

Yang et al. have developed an Analog malicious hardware trojan [Yang et al., 2016b]. They demonstrate how a fabrication-time attacker can utilize analog circuits to create a hardware attack that is both small (requiring as little as one gate) and stealthy (needing an unlikely trigger sequence before affecting a chip's functionality). In the open spaces of an already placed and routed design, a circuit is constructed using capacitors to siphon charge from nearby wires during transitions between digital values. When the capacitors fully charge, they execute an attack that compels a victim flip-flop to assume a desired value. The attack is weaponized into a remotely-controllable privilege escalation by connecting the capacitor to a controllable wire and selecting a victim flip-flop that stores the privilege bit for their processor. This attack is implemented in an OR1200 processor, and a chip is fabricated as part of the process.

Cruz et al. have presented A Machine Learning Based Automatic Hardware Trojan Attack Space Exploration and Benchmarking Framework [Cruz et al., 2022]. They introduce MIMIC, a pioneering machine learning-guided framework for automated Trojan insertion. This framework has the ability to generate a substantial and targeted set of valid Trojans for a given design by emulating the characteristics of a small group of known Trojans. While existing tools can automatically insert Trojan instances using fixed Trojan templates, they lack the capability to analyze known Trojan attacks to create new instances that precisely capture the threat model. MIMIC operates in two primary steps: (1) it examines the structural and functional features of existing Trojan populations in a multi-dimensional space to train machine learning models and generate numerous "virtual Trojans" for the specified design, (2) subsequently, it integrates them into the design by aligning their functional/structural properties with suitable nets of the internal logic structure.

2.3 Detection Techniques

Hardware Trojan detection is a critical aspect of ensuring the security and reliability of integrated circuits and electronic systems. The primary goal of hardware Trojan detection is to identify the presence of such malicious entities within a chip and prevent their adverse impact on system performance. Several detection techniques and methodologies have been developed to address the growing sophistication of Hardware Trojans.

The methods essentially assess the deviations induced by Hardware Trojans (HTs) on the system's behavior or seek potential HT profiles. In order to achieve this, designers need to be familiar with at least one specific parameter from the authentic device or define a target HT model for detection. If the deviation observed in the assessed parameter of a design under Trojan test (DUTT) surpasses an acceptable margin, the DUTT is categorized as being infected by a Trojan. A scheme derived from earlier surveys and research outlines the primary classifications of testing methods employed for identifying Trojans [put figure here].

The majority of current methods focus on detecting Trojans in manufactured integrated circuits (ICs) and presume the presence of a gate-level golden netlist. There have been limited studies addressing Trojan detection at higher-level design descriptions, such as the register transfer level IP. It's important to highlight that there is currently no singular, universally effective technique that can be employed to detect all types of Trojans.

Trojan detection methods can be categorized into two primary types: destructive and non-destructive [Tehranipoor and Wang, 2011]. Destructive techniques, involve employing a subset of manufactured integrated circuits (ICs) that undergo demetallization using Chemical Mechanical Polishing (CMP)[1], followed by Scanning Electron Microscope (SEM) image reconstruction and analysis [search for suitable ref]. The non-destructive methods can be categorized into two primary types: non-invasive and invasive. Non-invasive techniques preserve the original design without any alterations, whereas invasive techniques involve modifying the design to incorporate specific features aimed at detecting Trojans.

2.3.1 Invasive Trojan Detection Technique

The invasive methods for detecting Trojans can be further categorized into two groups: one focuses on preventing the insertion of Trojans during the design or fabri-

[1]Chemical mechanical polishing (CMP) is a planarization technique that was developed for semiconductor applications in the late 1980s and early 1990s. During this period, the number of metal layers increased dramatically and device topographies began to exhibit features that inhibited conformal deposition and gap fill by photoresist, metal, and insulator films. For more info visit: https://www.mks.com/n/chemical-mechanical-polishing, accessed 04 February 2024.

cation of an integrated circuit (IC), while the other enables the detection of inserted Trojans through post-manufacturing testing.

In [Vashistha et al., 2022], The proposed framework employs self-testing, advanced imaging, and image processing with machine learning to detect hardware Trojans inserted by untrusted foundries. They incorporate on-chip test structures with minimal power, delay, and silicon area overheads. The central aspect of the framework involves on-chip golden circuit design, allowing the generation of authentic samples for image-based Trojan detection through self-testing. A fundamental aspect of this framework involves the creation of the golden circuit (GC), which is designed to generate trusted on-chip cell images for training machine learning algorithms. This is accomplished by populating the remaining vacant spaces with logic cells from the original circuit design after the Place and Route process. These additional cells are interconnected in a tree structure to attain full test coverage. Due to their limited number, these cells can be easily verified through logic testing to ensure the absence of inserted Trojans. Once authenticated, these cells serve as on-chip golden cells, acting as training samples for the image analysis-based Trojan detection.

In [Supon and Rashidzadeh, 2021], An innovative tamper-resistant approach is introduced to capture the electromagnetic signature of integrated circuits, enhancing their security. In this proposed method, the unused metal and polysilicon layers are maximally employed as internal magnetic probes to monitor the device's signature, simultaneously restricting attackers' access to layout resources for routing Hardware Trojans. In the proposed approach, following the routing of the main circuit, the unused metal and polysilicon layers are filled with minimum feature wires. These wires can be placed in various ways, either randomly or with the metals configured as inductive sensors. Opting for inductive sensors offers distinct advantages over random distribution. These sensors serve as internal magnetic probes capable of capturing the induced signature of the chip when specific sections of the main circuit are activated. By the end of this process, no resources remain for connecting Trojan circuits. If an attacker attempts to modify the magnetic probes' design or alter the main circuit placement to gain access to resources for inserting and routing a Trojan, it will alter the IC's signature. Additionally, the probes undergo separate testing to identify any changes in their properties.

In [Banga and Hsiao, 2009b], A design method named VITAMIN is introduced, relying on the inversion of the supply voltage for alternate logic levels within an integrated circuit (IC). In this technique, the logic behavior of a gate, functioning with an inverted supply voltage, undergoes inversion during the test mode. Consequently, the activity of a seldom activated Trojan circuit is heightened, facilitating its detection through a comparison of the power profiles among different ICs.

In [Aslam et al., 2020], They propose a hardware Trojan triggered by threshold voltage, operating within the 0.45V to 0.998V threshold voltage range. It consumes ultra-low power (10.5nW) and maintains stealthiness, with an area overhead as low as 1.5% for a 28 nm technology node. The hardware Trojan detection sub-scheme introduces a lightweight, threshold voltage-aware sensor with a detection sensitivity of 0.251mV/nA. Using fixed and dynamic ring oscillator-based sensor segments, the proposal includes precise measurements of frequency and delay variations in response to threshold voltage shifts in a PMOS transistor. Lastly, the FPGA security scheme is strengthened by online transistor dynamic scaling (OTDS) to address hardware Trojan impact through run-time tolerant circuitry capable of identifying critical gates with worst-case drain current degradation.

2.3.2 Non-invasive Trojan Detection Technique

In non-invasive Trojan detection methods, Trojans are identified by comparing the behavior of the test IC with either the golden IC instance or a golden functional model. These techniques can be categorized into two primary types: run-time and test-time methods. Run-time techniques utilize an online monitoring system to identify suspicious activities during in-field operation, whereas test-time techniques focus on detecting Trojan-infected chips before their deployment.

Run-time, non-invasive Trojan detection approaches:

Run-time Trojan detection techniques involve monitoring and analyzing system behavior in real-time to identify potential malicious activities or the presence of Trojans during active operation. Several techniques are employed for this purpose:

In [Ngo et al., 2015a], the focus is on detecting advanced Hardware Trojans using a run-time detection approach. They aim to identify high-level and critical behavioral invariants, checking them during the circuit operation. The properties to be verified are described using Assertion and Property Specification Language (PSL). Subsequently, a Hardware Property Checker (HPC) is developed and integrated into the IC to verify these properties in real-time.

In [Cui et al., 2014], High-level synthesis for run-time hardware Trojan detection and recovery has been presented, They suggest establishing design guidelines to aid in run-time Trojan detection and swift recovery by exploring the diversity of untrusted third-party IP cores. Using these design guidelines, they demonstrate an optimization strategy aimed at reducing the implementation cost, specifically in terms of the number of distinct IP cores utilized in the implementation.

In [Li et al., 2022], they have presented Partial Reconfiguration for Run-time Memory Faults and Hardware Trojan Attacks Detection. The solution primarily relies on a centralized security detection controller for partially reconfigurable inspection content and a static memory wrapper to manage access conflicts and buffer testing cells. They demonstrate that a field programmable gate array prototype of the proposed framework can achieve the detection of 16 memory faults and three types of Hardware Trojans with one reconfigurable partition. This results in a 12.7% reduction in area and a 2.9% decrease in power overhead compared to a static implementation.

In [Amornpaisannon et al., 2024], a Secure Run-Time Hardware Trojan Detection Using Lightweight Analytical Models has been presented. They concentrate on hardware Trojans affecting the processor's performance. Current advanced detectors employ complex machine-learning models that monitor hardware counters for anomalies. These models necessitate a dedicated off-chip processor and extensive training for each target processor. In this study, they present a lightweight solution that utilizes data from a single reference run to accurately identify whether a Trojan is impeding processor performance across various CPU configurations, eliminating the need for new profiles. To achieve this, they employ an analytical model based on the application's inherent microarchitecturally independent characteristics. These models predict expected microarchitectural events across different processor configurations without requiring reference values for each application-hardware configuration pair. By comparing predicted values to actual hardware events, one can promptly identify unexpected application slowdowns, which are key indicators of many hardware Trojans.

Test-time, non-invasive Trojan detection approaches:

Test-time, non-invasive Trojan detection approaches refer to methods employed to identify potential hardware Trojans during the testing phase of integrated circuits without causing any physical or operational harm. These approaches focus on examining the characteristics and behavior of the circuits before deployment. There are two primary categories of testing approaches for detecting Trojans: those centered on logic testing and those focused on measuring side-channel parameters like power and delay. Test-time techniques offer the advantage of no hardware overhead, contrasting with run-time methods. However, a drawback is the need for a "golden" (Trojan-free) manufactured IC or functional model. Run-time methods often come with notable performance and power overhead, yet they serve as the ultimate defense, ensuring 100% confidence in computed results. Here are some common techniques in this category:

In [Yang et al., 2022], they have introduced a golden-free multidimensional self-referencing technique that utilizes side-channel signatures in both the time and frequency domains. They significantly expand Trojan coverage and enhance detection confidence. The article presents a fully automated detection framework with systematic methodologies for generating tests, extracting signatures, processing signals, calculating thresholds, and making decisions based on metrics. They effectively enable the synergistic self-referencing approach. Finally, they assess the proposed technique using a comprehensive hardware measurement setup involving 96 Trojan-inserted test chips.

In [Huang et al., 2022], The article introduces a technique based on ring oscillators to enhance both wire and net coverage. The circuit under test is partitioned into numerous blocks, each incorporating a ring oscillator for detecting hardware Trojans. Additionally, a path tracking algorithm is outlined to optimize path assignments.

In [Elkanishy et al., 2019], This study introduces a compact supervisory circuit that classifies the operation of a Bluetooth (BT) System on Chip (SoC) at low frequencies. It achieves this by monitoring the input power and the radio frequency (RF) output of the BT chip through an envelope detector. The concept involves cost-effective fabrication of an envelope detector, power supply current monitor, and classification algorithm on a customized low-frequency integrated circuit in a trusted legacy technology. When unexpected behavior is detected, the supervisory circuit has the capability to cut off power to the BT SoC. In this initial phase, they prototype the supervisory circuit using readily available components. Simple yet descriptive features are extracted from the RF output signal envelope. Subsequently, machine learning (ML) models are trained to classify various BT operation modes, such as BT advertising and transmit modes.

In [Gayatri et al., 2020], they have performed System Level Hardware Trojan Detection Using Side-Channel Power Analysis and Machine Learning. This paper suggests the detection of Hardware Trojans at the system level in the AES-256 decryption algorithm implemented in the Atmel XMega Controller (Target Board). The approach combines side-channel power analysis and machine learning. The ChipWhisperer-Lite board is employed for power analysis. Power traces from both the golden algorithm (Hardware Trojan-free) and Hardware Trojan-infected algorithms are acquired and utilized to train the machine learning model following the 80/20 rule. The resulting machine learning model achieves an accuracy range of 97%-100% for all the inserted Trojans.

The Side Channel Analysis method involves detecting Hardware Trojans by measuring circuit parameters and comparing them with the golden circuit (trojan-free circuit). In this approach, the detection relies on measuring the path delay caused by Hardware Trojans in a digital circuit. However, if the impact of the path delay is minimal, detection becomes challenging. In [Hasan et al., 2020] addresses two key aspects. Firstly, it aims to increase the path delay, and secondly, it explores an alternative method to overcome the challenge of implausible differences between delays in the golden and trojan-infected circuits. The entire experiment is carried out in Cadence, utilizing a library with a feature size of 45nm. For the first part, combined sweeping is employed. In the second part, it is demonstrated that trojans can be detected by measuring the slope of the observed delay.

2.3.3 Advantages/disadvantages of invasive/non-invasive detection techniques

Invasive Hardware Trojan Detection Techniques:

Advantages

1. **Higher Sensitivity:** Invasive techniques often involve physical modifications or probing, providing higher sensitivity to detect subtle hardware Trojans that may be missed by non-invasive methods.

2. **Direct Physical Examination:** Invasive techniques allow for a direct physical examination of the integrated circuit, making it easier to identify and analyze modifications at the hardware level.

3. **Potential for Root Cause Analysis:** Invasive methods can facilitate a more detailed analysis, aiding in identifying the root cause of a Trojan and understanding its behavior.

Disadvantages

1. **Destructive Nature:** Invasive methods can be destructive, rendering the tested hardware unusable. This is a significant drawback if post-analysis or forensic examination of the hardware is desired.
2. **Costly and Time-Consuming:** Invasive techniques often require specialized equipment and expertise, making them more costly and time-consuming compared to non-invasive methods.
3. **Ethical and Legal Concerns:** The invasive nature of these techniques raises ethical and legal concerns, especially if the hardware being tested is valuable or if the testing involves proprietary technology.

Non-Invasive Hardware Trojan Detection Techniques:

Advantages

1. **Preservation of Hardware:** Non-invasive methods preserve the integrity of the hardware being tested, allowing it to remain functional after the detection process.
2. **Lower Cost:** Non-invasive techniques are generally more cost-effective as they do not require specialized equipment for physical alterations or probing.
3. **Applicability to Operational Systems:** Non-invasive methods can be applied to operational systems without the need for disrupting their functionality, making them suitable for real-world scenarios.

Disadvantages

1. **Lower Sensitivity:** Non-invasive techniques may have lower sensitivity compared to invasive methods, potentially missing subtle or well-hidden hardware Trojans.
2. **Indirect Detection:** Non-invasive methods often rely on indirect indicators such as power consumption or electromagnetic emissions, which may not provide a complete picture of the hardware.
3. **Potential for False Positives/Negatives:** Non-invasive techniques may be prone to false positives or negatives, especially if the Trojans are designed to evade detection by these methods.
4. **Limited Insight into Trojan Behavior:** Non-invasive techniques may provide limited insight into the behavior of detected Trojans compared to invasive methods.

In practice, a combination of both invasive and non-invasive techniques may be employed to achieve a more comprehensive and accurate hardware Trojan detection strategy. The choice depends on factors such as the specific characteristics of the hardware, the desired level of sensitivity, ethical considerations, and the resources available for testing.

2.4 Software and Hardware Materials in HT Mitigation

Mitigating hardware Trojans, which are malicious modifications or additions to electronic circuits during the manufacturing process, requires a combination of software and hardware materials. The principal validation methods for complex systems are simulation, testing, deductive verification, and Formal verification (Model checking).

2.4.1 Simulation and testing

Conducting experiments prior to field deployment is common in both simulation and testing processes. Simulation is executed on a system's abstraction or model, while testing involves the actual product. In both approaches, signals or inputs are typically introduced at specific points within the system, and the resulting outputs are observed at other points. Although these methods are often cost-effective in identifying numerous errors, thoroughly examining all potential interactions and pitfalls through simulation and testing is rarely possible.

2.4.2 Deductive verification

Deductive verification methods employ axioms and proof rules to establish the accuracy of systems. Initially, these proofs were manually crafted. Over time, researchers recognized that software tools could be employed to propose different approaches for advancing from the current proof stage. Deductive verification offers the benefit of being applicable to the analysis of systems with infinite states. While there is some automation possible for this task, it is important to note that, even if the property under verification is valid, there is no restriction on the potential time or memory resources required to discover a proof.

Deductive verification is a labor-intensive procedure that is typically carried out by individuals with expertise in logical reasoning. As a result, it is primarily employed in critical systems, particularly those with high sensitivity such as security protocols.

2.4.3 Formal Verification:

Formal verification techniques use mathematical methods to prove the correctness of hardware designs. They can help identify and eliminate potential security vulnerabilities, including those introduced by hardware Trojans. The primary techniques for validating hardware systems include simulation, testing, and formal verification. Both simulation and testing entail conducting experiments prior to the system's deployment in real-world settings. Simulation is carried out on the design of the hardware system, while testing is executed on its actual implementation. These approaches are frequently employed as economical means of identifying numerous errors. Nevertheless, achieving a comprehensive examination of all potential behaviors in hardware systems through simulation and testing methods is rarely possible.

Formal verification is employed, specifically model checking, to assess if a hardware system meets specified properties. Unlike simulation and testing approaches, model checking enables the examination of the system's behavior for all conceivable inputs. The goal is to assist hardware designers in developing transparent and accurate systems with assured functionality. Organizations such as Cadence Design Systems, IBM, and Mentor Graphics create tools that facilitate what is known as functional verification, aiming for identical objectives. They employ the Assertion-Based Verification (ABV) methodology to promptly identify errors in hardware design.

Temporal Logic

Temporal logic is a formal system used in the field of computer science and formal verification to reason about and specify the temporal aspects of system behavior. Temporal logic provides a way to express statements and properties about the order and timing of events in a system over time. There are two main branches of temporal logic: Linear Temporal Logic (LTL) and Computation Tree Logic (CTL). Both are used to specify and verify temporal properties, but they have different approaches and applications.

Formal specification comprises one or more characteristics that can be represented as formulas in either Linear Temporal Logic (LTL) or Computational Tree Logic (CTL).

1. **Linear Temporal Logic (LTL):** LTL deals with linear sequences of time, expressing properties about the future along a single path. It includes temporal operators such as "next" (X), "until" (U), "always" (G), and "eventually" (F). For example, the LTL formula $G(p \rightarrow Xq)$ states that "it is always the case that if p holds, then in the next state, q will eventually hold."

2. **Computation Tree Logic (CTL):** CTL is designed to reason about branching time, where multiple possible paths of execution exist. CTL introduces existential and universal path quantifiers, allowing the specification of properties that hold along some or all possible paths. CTL includes operators such as "AX" (for "for all next"), "EX" (for "there exists next"), "AG" (for "always globally"), and "EF" (for "eventually in the future").

Temporal logic is commonly used in formal verification methods, especially in model checking, where it helps specify and verify system properties over time. Model checking tools use temporal logic formulas to automatically explore all possible states of a system and check whether the specified properties hold in each state or along each possible path of execution.

The application of temporal logic extends beyond formal verification and is also used in areas like real-time systems, concurrent programming, and specification of reactive systems, where understanding and reasoning about the temporal aspects of system behavior are crucial.

Model Checking

Model checking is a formal verification technique used to automatically check whether a given system or model satisfies a set of specified properties. It involves systematically exploring the entire state space of a system to verify if certain desired properties hold or if there are any violations. Model of the system and its specification are formulated in some precise mathematical language. To this end, the problem is formulated as a task in logic, namely to check whether a structure satisfies a given logical formula. This general concept applies to many kinds of logic and many kinds of structures. A simple model-checking problem consists of verifying whether a formula in the propositional logic is satisfied by a given structure. Model checking is widely employed in various domains, including hardware and software systems, communication protocols, and concurrent systems.

The model checker typically employs a thorough exploration of the finite state space of the system to ascertain the truth or falsity of a given specification, which represents a property of the system. If the system does not meet a specified property, the model checker generates a counterexample that illustrates an incorrect behavior. This problematic sequence offers valuable insights into comprehending the underlying cause of the failure, along with crucial hints for resolving the issue.

With adequate resources, the model checker will invariably conclude with either an affirmative or negative response. Additionally, it can be executed through algorithms that demonstrate reasonable efficiency.

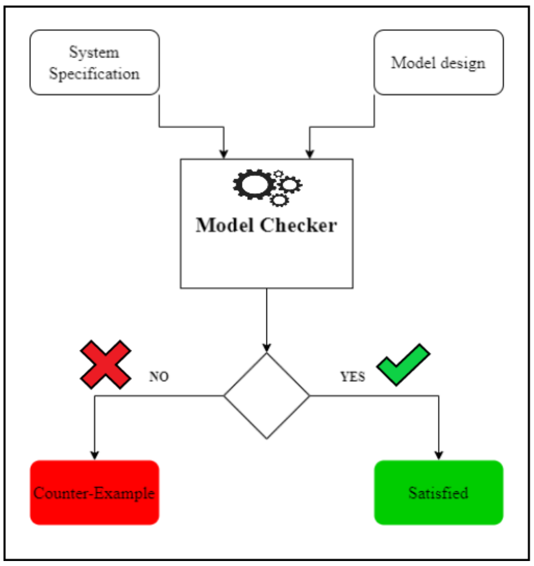


Figure 2.2: Model checker

Applying model checking to a design consists of several tasks:

1. **Modeling:** The initial step involves transforming the system's design into an abstract model that is compatible with a model checking tool. The abstract model aims to remove extraneous details from the design, and while automated conversion is feasible in certain instances, human guidance and assistance are often necessary, especially in cases where it is challenging to automate (such as in software design).

   The behavioral modeling describes how the circuit should behave. For these reasons, behavioral modeling is considered highest abstraction level as compared to data-flow or structural models. The VHDL synthesizer tool decides the actual circuit implementation. The VHDL behavioral model is widely used in test bench design, since the test bench design doesn't care about the hardware realization

2. **Specification:** The specification is typically presented in a logical formalism, often employing temporal logic to articulate the progression of system behavior over time. Key concerns in specification include:

   - **Consistency:** Is the provided specification free of contradictions?
   - **Completeness:** Does the provided specification encompass all the properties that the system is required to fulfill?
     These matters are not explicitly tackled within the realm of model checking.

3. **Verification:** In an ideal scenario, verification is entirely automated. Nevertheless, practical implementation frequently requires human involvement. An example of such manual intervention is the examination of verification outcomes. In instances of a negative result, the user receives an error trace, aiding the designer in pinpointing the source of the error. Addressing the error may involve modifying the system and reapplying the model checking algorithm. An error trace can also result from a false negative, that is from:

   - incorrect modeling of the system.
   - incorrect formalization of the specification.
   - inconsistent specification.

     A final possibility is that the verification task will fail to terminate normally, due to the size of the model, which is too large to fit into the computer memory

2.4.4 Physically Unclonable Functions (PUFs)

PUFs stand for Physically Unclonable Functions. PUFs are electronic components or systems that leverage the inherent physical variations in manufacturing processes to create unique and unpredictable identifiers for individual devices. These identifiers are difficult to clone or replicate, providing a form of hardware-based security.

Because of variations in the deep submicron manufacturing process, each transistor within an integrated circuit possesses slightly distinct physical characteristics. These variances result in minor yet quantifiable discrepancies in electronic properties like transistor threshold voltages and gain factor. As these process variations are beyond full control during manufacturing, the physical attributes of the devices cannot be duplicated or replicated.

Leveraging these inherent variations, PUFs prove highly beneficial as distinctive identifiers for individual integrated circuits (ICs). Achieved through the IC's internal circuitry, these minute variations are translated into a digital pattern of 0s and 1s, exclusive to that particular chip and consistently reproducible. This pattern serves as a "silicon fingerprint," analogous to the biometric characteristics found in humans.

The application of this technology spans from employing Physical Unclonable Functions for IoT security, capitalizing on its cost-effectiveness and adaptable implementation for significant advantages, to utilizing Physical Unclonable Functions in Aerospace & Defense. This demonstrates the technology's capability to provide the utmost level of security.

The Advantages of a Physical Unclonable Function

Devices, especially those integrated into the Internet of Things (IoT), necessitate keys for safeguarding their data, intellectual property (IP), and functionalities. These keys may be embedded onto the devices either by the device manufacturers, also known as OEMs, or at an earlier stage by a chip vendor. When chip vendors furnish pre-provisioned chips, they enhance the overall worth of the product offered to OEMs. Alternatively, if OEMs opt for self-provisioning, they usually acquire chips at a lower cost.

Regardless of whether the duty of provisioning cryptographic keys falls on the chip vendor or the device manufacturer, it is always a non-trivial undertaking. Introducing confidential keys into chips demands a reliable factory, introduces additional expenses and intricacies to the manufacturing process, and imposes restrictions on flexibility. This intricacy can be circumvented by generating the keys internally within the chip, utilizing either an internal random number generator (RNG) or a PUF.

Nevertheless, generating a key poses only one aspect of the challenge. This is because securely storing keys on devices is also far from straightforward. Storing secret keys in non-volatile memory (NVM) is not a viable option, as NVM is susceptible to hardware attacks. With hardware attacks that enable adversaries to access NVM content becoming more prevalent, relying on unprotected key storage becomes impractical. Thus, there is a demand for alternative secure key storage solutions. One possible approach involves incorporating a secure element into the device. However, introducing additional hardware also brings heightened complexity and cost. A silicon PUF, on the other hand, offers a secure means of storing cryptographic keys without the necessity of adding extra hardware.

Typical Use Cases for a Physical Unclonable Function in IoT Devices

- **Key Vault:** The most widely recognized application of PUF technology involves generating and safeguarding the cryptographic root key for a device. The cryptographic root key, produced by the PUF, eliminates the need for key injection and is impervious to duplication from one device to another. This is because the key is never stored; instead, it is reconstructed from the unique silicon fingerprint of the device each time it is required. Given that this fingerprint varies for every chip, there is no conceivable method for an attacker to replicate a key from one device to another.

- **Firmware IP Protection:** Imagine an IoT device holding sensitive information that demands protection—perhaps valuable intellectual property containing confidential trade secrets or measurement data that is either privacy-sensitive or crucial to the system. In such cases, the device necessitates a secure vault. Within this secure vault, any form of data can be securely stored and inherently linked to the device's hardware. Achieving this is straightforward with a PUF, wherein all sensitive data is encrypted using a key derived from the PUF root key.

- **Edge-to-Cloud Security:** Establishing a secure channel between an IoT device and the cloud, utilizing a public key infrastructure like a Transport Layer Security (TLS) connection with a cloud service, involves the exchange of certificates between the device and the cloud. These certificates serve to authenticate the entities to one another. Generating a certificate to authenticate a device entails creating a public/private key pair derived from the PUF fingerprint.

PUF Processing Algorithms

As mentioned earlier, the implementation of Physically Unclonable Functions (PUFs) necessitates processing algorithms to transform the silicon fingerprint into a cryptographic root key. This is crucial because the silicon fingerprint exhibits slight variations across different measurements due to inherent process differences, and external factors like ambient conditions (e.g., temperature and power supply) also impact electronic properties. Consequently, an effective PUF implementation must convert this potentially noisy fingerprint into a stable and genuinely random sequence of 0s and 1s to qualify as a cryptographic key. To achieve this, most PUF implementations employ two main processes:

1. Error correction, ensuring consistency in the derived key across multiple PUF measurements.

2. Privacy amplification, converting the fingerprint into a completely random string.

Error Correction

Error correction methods employed in the reconstruction of cryptographic keys involve two distinct phases: enrollment and reconstruction. During the one-time enrollment phase, the PUF response is correlated with a codeword from an error-correcting code. Details of this correlation are retained in the activation code (AC), also known as "helper data." The AC is designed in a way that it imparts no information about the key, and it can be stored off-chip since it lacks sensitivity. Although the AC must be accessible by PUF algorithms, any alteration, whether malicious or not, will hinder key reconstruction. Importantly, the AC remains valid exclusively for the chip on which it was originally generated.

Whenever the device requires the confidential PUF key, a fresh PUF measurement, complete with noise, is conducted. Subsequently, the PUF key, devoid of noise, is derived from the activation code (AC) and this newly obtained PUF response. This step is referred to as the reconstruction phase. Both the enrollment and reconstruction phases are depicted in Figure 2.3 .

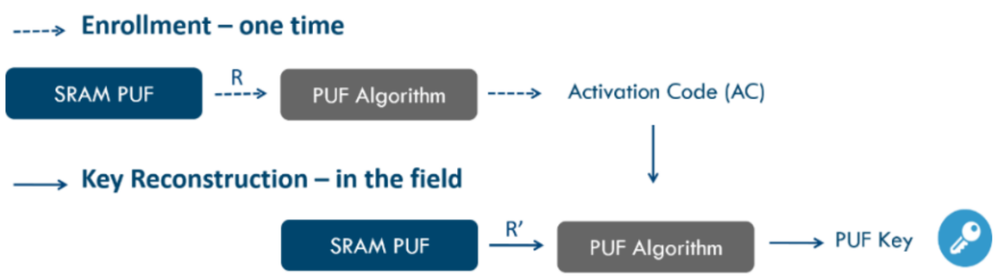


Figure 2.3: Error correction techniques for cryptographic key reconstruction

Privacy Amplification

Confidential keys offer security by virtue of their complete randomness and consequent unpredictability. While physical measurements like PUF responses exhibit a substantial degree of randomness, they often fall short of being uniformly random. Privacy amplification algorithms come into play to produce uniformly random keys. This is achieved, for instance, by hashing a substantial amount of data with ample entropy, resulting in a random string comprising 128 or 256 bits.

What Are the Challenges When Implementing a Physical Unclonable Function?

If Physically Unclonable Functions (PUFs) truly serve as reliable anchors of trust for devices, one might wonder why every chip vendor and original equipment manufacturer (OEM) doesn't adopt their own PUF implementations. The reason is that discovering and commercializing new types of PUFs is a challenging task. Identifying chip elements exhibiting the required behavior to create a device fingerprint involves extensive research before the actual productization phase begins. The process entails conducting millions of measurements under diverse conditions, accounting for silicon aging, to define the parameters necessary for error correction and privacy amplification algorithms. Typically, the journey of productizing a novel PUF implementation demands years of dedicated research and development efforts.

Furthermore, it's worth noting that numerous PUF implementations, including those already existing and available in the market, often necessitate substantial modifications to the chip hardware. The integration of such PUFs results in alterations to the manufacturing process, either by augmenting the number of masks essential for production or by mandating non-standard processing steps. This inevitably raises the overall cost of incorporating PUF technology into chips, thereby undermining the cost-effectiveness advantage associated with the use of PUFs, as discussed earlier.

Typical CMOS Implementations of Physical Unclonable Functions

Fortunately, these challenges can be easily addressed by both chip vendors and OEMs through the utilization of standard CMOS implementations of PUFs, seamlessly integrating them into their devices without necessitating any alterations to the chip hardware. The two PUF examples outlined below only require IP licensing and can be deployed on existing hardware. **The SRAM PUF** is designed for IoT platforms, like microcontrollers, and can be incorporated as software IP by an OEM. Chip vendors looking to incorporate this PUF into their products can opt for either soft- or hardware-IP versions. **The Butterfly PUF** caters to FPGA platforms, commonly employed in military, government, and aerospace applications, and can be implemented within the programmable fabric.

- **SRAM PUF:** The performance of an SRAM cell is influenced by the variance in the threshold voltages of its transistors. Even minimal differences in these voltages get magnified, driving the SRAM cell into one of its two stable states. Consequently, its PUF behavior is significantly more consistent than the inherent threshold voltages, establishing it as the simplest and most reliable method for employing transistor threshold voltages to construct an identifier.
- **Butterfly PUF**: The Butterfly PUF is built on the concept of establishing formations within the FPGA matrix that exhibit characteristics akin to an SRAM cell during the initialization phase. A Butterfly PUF cell is a cross-coupled bistable circuit that can be induced into an unstable state before stabilizing into one of the two possible stable states.

A Physical Unclonable Function, abbreviated as PUF, stands as a highly beneficial security component for both chip vendors and OEMs. The cryptographic key generated and securely established by a PUF serves as a foundational element of trust for a device. It forms the basis for successful applications in safeguarding keys, data, intellectual property (IP), and establishing secure connections with the cloud or other devices.

Discovering and commercializing a Physical Unclonable Function (PUF) in electronic circuits demands extensive research and development efforts spanning several years. Fortunately, there are existing PUF implementations that are readily accessible for utilization. Examples include SRAM PUFs and Butterfly PUFs, which do not necessitate extra hardware and are founded on standard CMOS processes, enabling seamless integration into chips and devices at a minimal cost. These silicon-proven PUFs effectively address the challenges of key provisioning for manufacturers, providing a robust foundation of trust for any device.

A Review of Some Studies on Physical Unclonable Functions (PUFs)

The concept of Physical Unclonable Functions (PUFs) has been a focal point in numerous studies, reflecting a significant interest and scholarly attention. Researchers from various disciplines have delved into the exploration of PUFs, investigating their properties, applications, and security implications. The prevalence of studies on PUFs attests to their relevance in the realm of hardware security and cryptographic systems. These collective research efforts contribute to a comprehensive understanding of PUFs, highlighting their potential and challenges across a spectrum of contexts.

A complete review of physical unclonable functions (pufs) and its applications in iot environment has been presented In[Yadav et al., 2022]. The topic of the paper is a comprehensive exploration of physical unclonable functions (PUFs). The paper delves into various studies on PUFs, advocating for their utilization over conventional security mechanisms and conducting a thorough comparison across multiple dimensions. It introduces the classification of PUFs into strong and weak categories. The paper further discusses the implementation of authentication schemes for nodes, servers, routers, and network gateways in a network communication scenario. It elucidates the communication procedure through an architectural framework. Addressing security challenges faced by smart devices due to potential attacks is a significant aspect covered. Lastly, the paper reviews emerging concepts and advancements in the field of physical unclonable functions.

In [Ning et al., 2020], Physical unclonable function: architectures, applications and challenges for dependable security has been deeply discussed. The physical design of PUF is thought to be easily produced but challenging or nearly impossible to replicate due to variations in the manufacturing process. Despite this, a substantial community of analysts perceives that hardware-based PUFs have opened avenues for achieving reliable security. This research thoroughly examines the architecture, applications, requirements, and challenges of PUFs for security solutions. To present the literature comprehensively, the authors have introduced a taxonomy that categorizes PUFs into two main groups: non-silicon and silicon-based PUFs. Currently, there is no all-encompassing survey that systematically compares the usability of memory-based PUFs with analogue/mixed-signal-based PUFs, which are considered more suitable than their counterparts. Similarly, the study outlines network-specific application scenarios in wireless sensor networks, wireless body area networks, and the Internet of Things, categorizing PUFs as strong, weak, and controlled. Additionally, the authors identify potential limitations in PUF structures and outline open research challenges to achieve desired security levels.

Strong PUFs are susceptible to machine learning and modeling attacks.A solution is suggested where the challenges of a Strong PUF are encrypted to eliminate the linear challenge-response correlation exploitable by potential attacks In[Vatajelu et al., 2019]. In this scenario, a Weak PUF with a ZeroBit Error Rate generates the encryption key, ensuring that each PUF instance exhibits a distinct, nonlinear correlation between challenges and responses. Two implementations of this solution are introduced, and their robustness against machine learning attacks is showcased.

2.4.5 Hardware-based Encryption

Hardware-based encryption is a method of securing data through the use of dedicated cryptographic hardware components. In this approach, encryption and decryption processes are offloaded to specialized hardware modules, providing a higher level of security and often better performance compared to software-based encryption.

Key characteristics of hardware-based encryption include:

1. **Dedicated Hardware Components:** Hardware-based encryption relies on specialized hardware components, such as cryptographic processors or integrated circuits, designed specifically for handling encryption algorithms.

2. **Accelerated Processing:** The dedicated hardware is optimized for cryptographic operations, leading to faster encryption and decryption compared to software-based approaches. This is particularly beneficial in scenarios where real-time processing or high-throughput is essential.

3. **Secure Key Storage:** Hardware-based encryption often includes secure key storage mechanisms, protecting cryptographic keys from unauthorized access or extraction. This helps in preventing key compromise, a critical aspect of maintaining data security.

4. **Tamper Resistance:** Hardware-based encryption solutions may incorporate tamper-resistant features, making it difficult for attackers to physically access or manipulate the cryptographic components. This enhances the overall security of the system.

5. **Random Number Generation:** Some hardware-based encryption systems include dedicated modules for random number generation, crucial for the generation of secure cryptographic keys and initialization vectors.

6. **Cryptographic Algorithms Support:** Hardware-based encryption supports a variety of cryptographic algorithms, including symmetric key algorithms (e.g., AES) and asymmetric key algorithms (e.g., RSA), allowing for flexibility in implementation.

7. **Integration with Secure Elements:** Hardware-based encryption is often integrated into secure elements or modules, providing a secure environment for sensitive operations and key management.

Applications of hardware-based encryption can be found in various fields, including data storage devices (e.g., self-encrypting drives), network communication devices, and secure communication modules in embedded systems. This approach is valued for its ability to provide robust security measures, especially in situations where the protection of sensitive information is paramount.

A Review of Some Studies on Hardware-based encryption

A resilient architecture for the hardware implementation of the advanced encryption standard (AES) algorithm is introduced In [Masoumi, 2019], featuring high efficiency and resistance against power analysis attacks. By selecting an appropriate topology, the FPGA implementation's resource requirements for the AES algorithm have been minimized. Furthermore, an innovative approach, combining a randomized SBox with a modified Boolean masking technique, eliminates the correlation between the Hamming distance of sensitive data and the algorithm's power consumption on the target platform. The robustness of the proposed outer masking, a modified version of the existing first-order Boolean masking scheme, is assessed through Welch's t-test statistical analysis and experimental results. Additionally, the effectiveness of the internal randomization technique within the SBox module relies on randomization in the underlying composite field $GF(2^4)^2$.

Scalable and Efficient Hardware Architectures for Authenticated Encryption in IoT Applications has been presented In [Khan et al., 2021]. Three generic implementation strategies (unrolled, round-based, and serialized) are proposed for developing highly efficient hardware architectures. These methods are suitable for all authenticated encryption schemes and are characterized by their lightweight and swift nature, as opposed to traditional public key encryption methods. Ascon serves as an illustration of the three outlined strategies: 1) The unrolled architecture achieves throughputs of 766.9 Mb/s (Ascon-128) and 1389.2 Mb/s (Ascon-128a), making it suitable for high-throughput IoT applications. 2) The round-based architecture achieves TP-to-area ratios of 0.153 (Ascon-128) and 0.244 (Ascon-128a), surpassing state-of-the-art results by 73.8% and 40.2%, respectively. 3) A novel serialized implementation technique is introduced, processing the substitution-box (S-box) in a multiple-bit-per-cycle fashion, a departure from the conventional one-bit-per-cycle approach. The two-bits-per-clock-cycle implementation increases throughput by 230.8% with only 36.8% additional hardware area. These strategies enable scalability in the number of rounds (round-based) and bits-per-clock-cycle (serialized), addressing varying requirements in throughput and area, as demonstrated in smart city IoT applications.

In [Azzaz et al., 2020], A novel and efficient strategy has been proposed to enhance the security of biometric models, specifically fingerprint templates, against potential attacks. The suggested design relies on the Vernam stream cipher, with a hardware-based key generator. The devised cryptosystem incorporates a multi-scroll chaotic system known for its extensive key space, capable of generating N×N grid multi-scroll attractors with favorable chaotic dynamic behavior. The hardware implementation involves describing the Euler method using VHDL. Experimental results on a Field-Programmable Gate Array (FPGA) confirm the efficacy of the developed architecture, achieving a well-balanced trade-off between hardware resources and performance. Furthermore, security analysis demonstrates the robustness of the designed encryption algorithm against statistical, brute force, and entropy attacks. Consequently, this solution emerges as a lightweight security measure, particularly beneficial in various embedded applications, especially for securing biometric authentication systems.

2.4.6 Trusted Platform Module (TPM)

A Trusted Platform Module (TPM) is a specialized hardware component that provides a secure foundation for the establishment of trust in computing environments. It is designed to enhance the security of systems by providing a range of cryptographic functions and capabilities.

The Trusted Platform Module (TPM) encompasses fundamental elements crucial for enhancing the security of computing environments. This specialized hardware component integrates cryptographic functions, secure storage, and a root of trust to establish and maintain system integrity. TPM facilitates secure boot processes, remote attestation, and the sealing/unsealing of sensitive data, providing hardware-based security resistant to various attacks. Standardized by organizations like the Trusted Computing Group, TPM plays a pivotal role in safeguarding systems through its key features:

1. **Cryptographic Functions:** TPMs have built-in cryptographic functions, such as generating and storing keys securely, performing digital signatures, and executing hash functions. These capabilities enable the TPM to support various security applications.

2. **Secure Storage:** TPMs include a secure storage area, often referred to as the Platform Configuration Registers (PCR), where sensitive information like cryptographic keys and measurements of system components can be securely stored.

3. **Root of Trust:** TPM serves as a root of trust, providing a secure starting point for system integrity. It helps in establishing and maintaining trustworthiness throughout the boot process and during system operation.

4. **Secure Boot:** TPM can be utilized in a secure boot process to ensure that only authorized and unaltered software components are loaded during system initialization. This helps prevent the execution of malicious code during startup.

5. **Remote Attestation:** TPM supports remote attestation, allowing a system to prove its integrity to a remote entity. This is particularly useful in scenarios where one system needs to verify the trustworthiness of another system in a network.

6. **Sealing and Unsealing:** TPMs enable the sealing of data to a specific set of system states, ensuring that sensitive information can only be accessed when the system is in a predefined, trusted state. Unsealing can then be performed when the system is in the expected state.

7. **Hardware-Based Security:** As a dedicated hardware component, TPM is resistant to many software-based attacks. It is tamper-resistant and provides a higher level of security compared to purely software-based solutions.

8. **Standardized Specifications:** TPM specifications are standardized by organizations such as the Trusted Computing Group (TCG), ensuring interoperability and compatibility across different hardware and software platforms.

TPMs are commonly found in various computing devices, including laptops, desktops, and servers. They play a crucial role in bolstering the security of systems by providing a foundation for secure boot processes, cryptographic operations, and protection of sensitive information.

Bibliography

[Alkabani and Koushanfar, 2008] Alkabani, Y. and Koushanfar, F. (2008). Extended abstract: Designer's hardware trojan horse. In *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 82–83.

[Amornpaisannon et al., 2024] Amornpaisannon, B., Diavastos, A., Peh, L.-S., and Carlson, T. E. (2024). Secure run-time hardware trojan detection using lightweight analytical models. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 43(2):431–441.

[Aslam et al., 2020] Aslam, S., Jennions, I. K., Samie, M., Perinpanayagam, S., and Fang, Y. (2020). Ingress of threshold voltage-triggered hardware trojan in the modern fpga fabric–detection methodology and mitigation. *IEEE Access*, 8:31371–31397.

[Azzaz et al., 2020] Azzaz, M. S., Tanougast, C., Maali, A., and Benssalah, M. (2020). An efficient and lightweight multi-scroll chaos-based hardware solution for protecting fingerprint biometric templates. *International Journal of Communication Systems*, 33(10):e4211.

[Banga and Hsiao, 2008] Banga, M. and Hsiao, M. S. (2008). A region based approach for the identification of hardware trojans. In *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 40–47. IEEE.

[Banga and Hsiao, 2009a] Banga, M. and Hsiao, M. S. (2009a). A novel sustained vector technique for the detection of hardware trojans. In *2009 22nd international conference on VLSI design*, pages 327–332. IEEE.

[Banga and Hsiao, 2009b] Banga, M. and Hsiao, M. S. (2009b). Vitamin: Voltage inversion technique to ascertain malicious insertions in ics. In *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 104–107.

[Bhunia et al., 2014] Bhunia, S., Hsiao, M. S., Banga, M., and Narasimhan, S. (2014). Hardware trojan attacks: Threat analysis and countermeasures. *Proceedings of the IEEE*, 102(8):1229–1247.

[Cruz et al., 2022] Cruz, J., Gaikwad, P., Nair, A., Chakraborty, P., and Bhunia, S. (2022). A machine learning based automatic hardware trojan attack space exploration and benchmarking framework. In *2022 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pages 1–6. IEEE.

[Cui et al., 2014] Cui, X., Ma, K., Shi, L., and Wu, K. (2014). High-level synthesis for run-time hardware trojan detection and recovery. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6.

[Elkanishy et al., 2019] Elkanishy, A., Badawy, A.-H. A., Furth, P. M., Boucheron, L. E., and Michael, C. P. (2019). Supervising communication soc for secure operation using machine learning. In *2019 IEEE 62nd International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 582–585.

[Gayatri et al., 2020] Gayatri et al., Mitra, M. P. (2020). System level hardware trojan detection using side-channel power analysis and machine learning. In *2020 5th International Conference on Communication and Electronics Systems (ICCES)*.

[Ghandali et al., 2016] Ghandali, S., Becker, G. T., Holcomb, D., and Paar, C. (2016). A design methodology for stealthy parametric trojans and its application to bug attacks. In *Cryptographic Hardware and Embedded Systems–CHES 2016: 18th International Conference, Santa Barbara, CA, USA, August 17-19, 2016, Proceedings 18*, pages 625–647. Springer.

[Hasan et al., 2020] Hasan, M. M., Baul, S., Hashem, M. B., and Rahman, H. (2020). Hardware trojan detection using slope of path delay trend: Combination of clock and dc sweep. In *2020 11th International Conference on Electrical and Computer Engineering (ICECE)*, pages 145–148.

[Huang et al., 2020] Huang, Z., Wang, Q., Chen, Y., and Jiang, X. (2020). A survey on machine learning against hardware trojan attacks: Recent advances and challenges. *IEEE Access*, 8:10796–10826.

[Huang et al., 2022] Huang et al., Hsiao, C. (2022). A security method of hardware trojan detection using path tracking algorithm. *J Wireless Com Network*, 81.

[Jacob et al., 2014] Jacob, N., Merli, D., Heyszl, J., and Sigl, G. (2014). Hardware trojans: current challenges and approaches. *IET Computers & Digital Techniques*, 8(6):264–273.

[Jha and Jha, 2008] Jha, S. and Jha, S. K. (2008). Randomization based probabilistic approach to detect trojan circuits. In *2008 11th IEEE High Assurance Systems Engineering Symposium*, pages 117–124. IEEE.

[Khan et al., 2021] Khan, S., Lee, W.-K., and Hwang, S. O. (2021). Scalable and efficient hardware architectures for authenticated encryption in iot applications. *IEEE Internet of Things Journal*, 8(14):11260–11275.

[King et al., 2008] King, S. T., Tucek, J., Cozzie, A., Grier, C., Jiang, W., and Zhou, Y. (2008). Designing and implementing malicious hardware. *Leet*, 8:1–8.

[Li et al., 2022] Li, Y., Chen, L., Wang, J., and Gong, G. (2022). Partial reconfiguration for run-time memory faults and hardware trojan attacks detection. In *2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 173–176.

[Liu et al., 2017] Liu, Y., Jin, Y., Nosratinia, A., and Makris, Y. (2017). Silicon demonstration of hardware trojan design and detection in wireless cryptographic ics. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(4):1506–1519.

[Masoumi, 2019] Masoumi, M. (2019). A highly efficient and secure hardware implementation of the advanced encryption standard. *Journal of Information Security and Applications*, 48:102371.

[Ngo et al., 2015a] Ngo, X. T., Danger, J.-L., Guilley, S., Najm, Z., and Emery, O. (2015a). Hardware property checker for run-time hardware trojan detection. In *2015 European Conference on Circuit Theory and Design (ECCTD)*, pages 1–4.

[Ngo et al., 2015b] Ngo, X.-T., Exurville, I., Bhasin, S., Danger, J.-L., Guilley, S., Najm, Z., Rigaud, J.-B., and Robisson, B. (2015b). Hardware trojan detection by delay and electromagnetic measurements. pages 782–787.

[Ning et al., 2020] Ning, H., Farha, F., Ullah, A., and Mao, L. (2020). Physical unclonable function: architectures, applications and challenges for dependable security. *IET Circuits, Devices & Systems*, 14(4):407–424.

[Perez and Pagliarini, 2023] Perez, T. D. and Pagliarini, S. (2023). Hardware trojan insertion in finalized layouts: From methodology to a silicon demonstration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 42(7):2094–2107.

[Qu and Yuan, 2014] Qu, G. and Yuan, L. (2014). Design things for the internet of things—an eda perspective. In *2014 IEEE/ACM international conference on Computer-Aided Design (ICCAD)*, pages 411–416. IEEE.

[Skorobogatov,] Skorobogatov, S. Hardware assurance and its importance to national security (2012).

[Subramani et al., 2020] Subramani, K. S., Helal, N., Antonopoulos, A., Nosratinia, A., and Makris, Y. (2020). Amplitude-modulating analog/rf hardware trojans in wireless networks: Risks and remedies. *IEEE Transactions on Information Forensics and Security*, 15:3497–3510.

[Supon and Rashidzadeh, 2021] Supon, T. M. and Rashidzadeh, R. (2021). On-chip magnetic probes for hardware trojan prevention and detection. *IEEE Transactions on Electromagnetic Compatibility*, 63(2):353–364.

[Tehranipoor and Wang, 2011] Tehranipoor, M. and Wang, C. (2011). *Introduction to hardware security and trust*. Springer Science & Business Media.

[Vashistha et al., 2022] Vashistha, N., Lu, H., Shi, Q., Woodard, D. L., Asadizanjani, N., and Tehranipoor, M. M. (2022). Detecting hardware trojans using combined self-testing and imaging. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 41(6):1730–1743.

[Vatajelu et al., 2019] Vatajelu, E. I., Natale, G. D., Mispan, M. S., and Halak, B. (2019). On the encryption of the challenge in physically unclonable functions. In *2019 IEEE 25th International Symposium on On-Line Testing and Robust System Design (IOLTS)*, pages 115–120.

[Wang et al., 2008] Wang, X., Tehranipoor, M., and Plusquellic, J. (2008). Detecting malicious inclusions in secure hardware: Challenges and solutions. In *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 15–19. IEEE.

[Wolff et al., 2008] Wolff, F., Papachristou, C., Bhunia, S., and Chakraborty, R. S. (2008). Towards trojan-free trusted ics: Problem analysis and detection scheme. In *Proceedings of the conference on Design, automation and test in Europe*, pages 1362–1365.

[Yadav et al., 2022] Yadav, A., Kumar, S., and Singh, J. (2022). A review of physical unclonable functions (pufs) and its applications in iot environment. *Ambient Communications and Computer Systems: Proceedings of RACCCS 2021*, pages 1–13.

[Yang et al., 2016a] Yang, K., Hicks, M., Dong, Q., Austin, T., and Sylvester, D. (2016a). A2: Analog malicious hardware. In *2016 IEEE symposium on security and privacy (SP)*, pages 18–37. IEEE.

[Yang et al., 2016b] Yang, K., Hicks, M., Dong, Q., Austin, T., and Sylvester, D. (2016b). A2: Analog malicious hardware. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 18–37.

[Yang et al., 2022] Yang, S., Hoque, T., Chakraborty, P., and Bhunia, S. (2022). Golden-free hardware trojan detection using self-referencing. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 30(3):325–338.

[Zhang and Qu, 2014] Zhang, J. and Qu, G. (2014). A survey on security and trust of fpga-based systems. In *2014 International Conference on Field-Programmable Technology (FPT)*, pages 147–152. IEEE.

[Zhang et al., 2015] Zhang, J., Yuan, F., Wei, L., Liu, Y., and Xu, Q. (2015). Veritrust: Verification for hardware trust. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(7):1148–1161.