Ministry of Higher Education and Scientific Research

وزارة التعليم العالي والبحث العلمي

Badji Mokhtar Annaba University

Faculty of Technology

Department of Computer Science

جامعة باجي مختار – عنابـــة

كـليـة التكنـولـوجيـا

قســم الاعلام الالـي

Thesis

Submitted in partial fulfillment of requirements for the degree of

Doctoral Third Cycle

Major: Computer Science

Minor: Embedded Systems

By :

GUECHI BILLEL

Entitled:

Security of Heterogeneous Systems on Chip

Thesis defended on **14/12/2025** In front of the examination committee composed of :

N°	Name & First Name	Grade	Institution	Quality
01	**Ghoualmi-Zine Nassira**	Prof.	Badji Mokhtar University-Annaba	President
02	**Redjimi Mohammed**	Prof.	University 20 août 1955 - Skikda	Supervisor
03	**Khadir Mohamed Tarek**	Prof.	Badji Mokhtar University-Annaba	Examiner
04	**Belmeguenai Aissa**	Prof	University 20 août 1955 - Skikda	Examiner
05	**Khettatba Mourad**	MCA	Badji Mokhtar University-Annaba	Examiner
06	**Zeghida Djamel**	MCA	University 20 août 1955 - Skikda	Examiner

Academic year: 2024/2025

Ministère de l'enseignement Supérieur et de la recherche Scientifique

وزارة التعليم العالي والبحث العلمي

Université Badji Mokhtar – Annaba

Faculté de Technologie

Département Informatique

جامعة باجي مختار – عنابـــة

كـليــة التكنـولـوجيـا

قســم الاعلام الالـي

Thèse

Présentée pour obtenir le diplôme de

Doctorat Troisième Cycle

Filière : Informatique

Spécialité : Informatique Embarquée

Par :

GUECHI BILLEL

Thème :

Sécurisation des Systèmes Hétérogènes sur puce

Thèse soutenue le **14/12/2025** devant le jury composé de :

N°	Nom et prénom	Grade	Etablissement	Qualité
01	**Ghoualmi-Zine Nassira**	Prof.	Université Badji Mokhtar-Annaba	Président
02	**Redjimi Mohammed**	Prof.	Université 20 août 1955 - Skikda	Rapporteur
03	**Khadir Mohamed Tarek**	Prof.	Université Badji Mokhtar-Annaba	Examinateur
04	**Belmeguenai Aissa**	Prof.	Université 20 août 1955 - Skikda	Examinateur
05	**Khettatba Mourad**	MCA	Université Badji Mokhtar-Annaba	Examinateur
06	**Zeghida Djamel**	MCA	Université 20 août 1955 - Skikda	Examinateur

Année universitaire : 2024/2025

Dedication

To my beloved parents, whose unwavering support and prayers have guided me through every step of this journey.

To my brothers and sisters, for their constant encouragement and belief in my potential.

To my fiancee Meriem, whose love and patience have been my strength throughout this process.

Declaration of Authorship and Ownership

I, **Billel Guechi**, hereby declare that this thesis titled, *"Security of Heterogeneous Systems On Chip "* and the work presented in it are my own. This thesis has been submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at *[Badji Mokhtar University Annaba]*, and it represents my original work except where otherwise indicated by reference.

I certify that the research in this thesis has not been previously submitted for any degree or professional qualification at any other institution. The intellectual property contained within this work belongs to me, **Billel Guechi**, unless otherwise stated.

Date: September 2024 Signature: _____

Billel Guechi

Acknowledgements

First and foremost, I would like to express my deepest gratitude to Allah, the Most Merciful, for blessing me with the ability, strength, and endurance to complete this PhD journey. Without His divine guidance, none of this would have been possible.

I would like to extend my heartfelt thanks to my supervisor, Pr. Mohammed Redjimi, for his invaluable guidance, constant support, and insightful advice throughout the course of my research. His mentorship has been instrumental in shaping this work, and I am forever grateful for his patience and encouragement.

To my parents, thank you for your endless love, sacrifices, and prayers. You have been my pillars of strength, and your unwavering belief in me has kept me going even during the toughest times.

To my brothers and sisters, thank you for your support and understanding, always pushing me forward with your words of encouragement.

To my fiancée Meriem, your love and constant encouragement have been my anchor through this journey. Your patience and understanding, especially during the challenging moments, have meant the world to me.

Finally, I would like to thank all my friends, colleagues, and anyone who has contributed to my academic and personal growth during this journey. This accomplishment is a collective achievement, and I am truly grateful to each one of you.

Thank you all.

Billel Guechi
September 2024

Abstract

The increasing complexity and globalization of integrated circuit (IC) supply chains have introduced critical vulnerabilities to hardware systems, particularly System-on-Chip (SoC) architectures. These architectures, due to their intricate designs and external dependencies, are highly susceptible to hardware Trojans (HTs). Hardware Trojans represent a significant security threat, capable of compromising data integrity, system functionality, and user privacy. This dissertation presents two key contributions to address this challenge.

First, a passive method for detecting externally activated hardware Trojans is introduced. This method is particularly effective in the context of radio receivers, where external signals can be used to activate hidden Trojan circuits. By leveraging passive side-channel monitoring techniques, this approach identifies anomalous patterns without interfering with system operation, improving detection accuracy for Trojans that activate only in response to specific external stimuli.

The second contribution is the development of a hardware security module (HSM) designed to protect data integrity through encryption. This HSM incorporates robust encryption mechanisms to guard against unauthorized access or data manipulation, ensuring that sensitive data remains intact and secure. The module integrates real-time encryption with integrity-checking, offering comprehensive protection against hardware and software-level tampering.

Key results show moderate success for both contributions. The passive detection method demonstrated a medium level of accuracy in identifying externally activated Trojans in controlled radio receiver environments, with some challenges in minimizing false positives. Similarly, the hardware security module achieved a reasonable balance between performance overhead and data integrity protection, making it suitable for certain medium-critical applications but with room for further optimization in high-performance systems.

These findings contribute to the detection of hardware Trojans and securing SoC designs, providing a foundation for future improvements in hardware security solutions.

Keywords: globalization, integrated circuits (IC), system-on-chip (Soc), hardware Trojans (HTs), hardware security module (HSM), externally activated hardware Trojan, encryption.

الملخص

التعقيد المتزايد وعولمة دورة انتاج الدوائر المتكاملة **(IC)** قد أدخلا ثغرات خطيرة على الأنظمة الالكترونية، وخصوصًا على هيكلة الأنظمة على الرقاقة **.(SoC)** هذه الهيكلة، بسبب تصاميمها المعقدة واعتمادها على مصادر خارجية، تكون عرضة بشكل كبير لهجمات أحصنة طروادة المادية **.(HTs)** تمثل أحصنة طروادة المادية تهديدًا كبيرًا على الأمان، حيث يمكنها اختراق سلامة البيانات، ووظائف النظام، وخصوصية المستخدمين. هذه الأطروحة تقدم مساهمتين رئيسيتين لمواجهة هذا التحدي.

أولاً، يتم تقديم طريقة للكشف عن أحصنة طروادة المفعلة خارجيًا. هذه الطريقة فعالة بشكل خاص في سياق أجهزة الاستقبال اللاسلكية، حيث يمكن استخدام الإشارات الخارجية لتفعيل الدوائر المخفية لأحصنة طروادة. من خلال الاستفادة من تقنيات مراقبة القنوات الجانبية السلبية، تستطيع هذه الطريقة تحديد الأنماط الشاذة دون التدخل في تشغيل النظام، مما يحسن دقة الكشف عن أحصنة طروادة التي تتفعل فقط استجابة لمثيرات خارجية محددة.

المساهمة الثانية هي تطوير وحدة أمان مادية **(HSM)** مصممة لحماية سلامة البيانات من خلال التشفير. تدمج هذه الوحدة آليات تشفير قوية للحماية من الوصول غير المصرح به أو التلاعب بالبيانات، مما يضمن بقاء البيانات الحساسة سليمة وآمنة. تدمج وحدة التشفير مع فحص السلامة، مما يوفر حماية شاملة ضد التلاعب المادي وبرمجيات النظام.

أظهرت النتائج الرئيسية نجاحًا مقبولا لكلتا المساهمتين. حيث أظهرت طريقة الكشفدقة متوسطة في تحديد أحصنة طروادة المفعلة خارجيًا في بيئات أجهزة استقبال لاسلكية مُتحكم بها، مع بعض التحديات في تقليل الإنذارات الكاذبة. وبالمثل، حققت وحدة الأمان المادية توازنًا معقولًا بين العبء على الأداء وحماية سلامة البيانات، مما يجعلها مناسبة لبعض التطبيقات متوسطة الأهمية، مع مجال لتحسين الأداء في الأنظمة عالية الأداء.

تساهم هذه النتائج في الكشف عن أحصنة طروادة المادية وتأمين تصميمات الأنظمة على الرقاقة، مما يوفر أساسًا للتحسينات المستقبلية في حلول الأمان المادي.

الكلمات المفتاحية: العولمة، الدوائر المتكاملة**(IC)** ، أنظمة على رقاقة**(SoC)** ، أحصنة طروادة المادية**(HTs)** ، وحدة أمان مادية**(HSM)** ، أحصنة طروادة مفعلة خارجيًا، التشفير.

Résumé

La complexité croissante et la mondialisation des chaînes d'approvisionnement de circuits intégrés (IC) ont introduit des vulnérabilités critiques dans les systèmes matériels, en particulier les architectures de systèmes sur puce (SoC). Ces architectures, en raison de leurs conceptions complexes et de leur dépendance à des sources externes, sont particulièrement vulnérables aux Chevaux de Troie matériels (HTs). Les Chevaux de Troie matériels représentent une menace de sécurité importante, capable de compromettre l'intégrité des données, le bon fonctionnement des systèmes et la confidentialité des utilisateurs. Cette thèse présente deux contributions principales pour relever ce défi.

Tout d'abord, une méthode passive de détection des Chevaux de Troie matériels activés de manière externe est introduite. Cette méthode est particulièrement efficace dans le contexte des récepteurs radio, où des signaux externes peuvent être utilisés pour activer des circuits cachés de Troie. En exploitant des techniques de surveillance passive des canaux auxiliaires, cette approche identifie des schémas anormaux sans interférer avec le fonctionnement du système, améliorant ainsi la précision de la détection des Trojans qui ne s'activent qu'en réponse à des stimuli externes spécifiques.

La deuxième contribution est le développement d'un module de sécurité matérielle (HSM) conçu pour protéger l'intégrité des données par le biais du chiffrement. Ce HSM intègre des mécanismes de chiffrement robustes pour se prémunir contre l'accès non autorisé ou la manipulation des données, garantissant que les données sensibles restent intactes et sécurisées. Le module intègre un chiffrement en temps réel avec des vérifications d'intégrité, offrant une protection complète contre les altérations tant au niveau matériel que logiciel.

Les résultats clés montrent un succès modéré pour les deux contributions. La méthode de détection passive a démontré un niveau de précision moyen dans l'identification des Trojans activés de manière externe dans des environnements contrôlés de récepteurs radio, avec quelques défis pour minimiser les faux positifs. De même, le module de sécurité matérielle a atteint un équilibre raisonnable entre la surcharge de performance et la protection de l'intégrité des données, le rendant adapté à certaines applications de criticité moyenne, avec une marge d'optimisation pour les systèmes à haute performance.

Ces conclusions contribuent à la détection des Chevaux de Troie matériels et à la sécurisation des conceptions de SoC, fournissant une base pour des améliorations futures dans les solutions de sécurité matérielle.

Mots-clés: mondialisation, circuits intégrés (IC), système sur puce (SoC), Chevaux de Troie matériels (HTs), module de sécurité matérielle (HSM), Cheval de Troie activé de manière externe, chiffrement.

Table Of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overall Context Of The Research

Research on hardware Trojans focuses on understanding, detecting, mitigating, and preventing malicious modifications to hardware components. Hardware Trojans are unauthorized alterations to a hardware design, often implanted during the manufacturing process or through the supply chain. These alterations can compromise the security, functionality, and reliability of electronic devices.

Hardware Trojans can vary significantly in their complexity and impact. They might be designed to leak sensitive information [Guechi and Redjimi, 2023], disrupt operations, or provide backdoor access to a system. They can be classified based on their physical characteristics (e.g., additional circuits) or their activation and effect mechanisms (e.g., always-on or triggered by specific conditions) [Tehranipoor and Koushanfar, 2016].

The paper also aims at identifying the possible threat scenarios regarding hardware Trojan, and the are known to be state sponsored attacks, industrial espionage and criminal actions. The threat which is much more severe in industrial and civil infrastructures, military applications and consumer electronics.

Substandard hardware may result to major disasters or have devastating impacts on security . There are some approaches which are under progress to find out the hardware Trojan. These methods can be broadly categorized into destructive and non-destructive approaches:

- **Destructive:** Such methods as optical inspection or de-layering which involve the physical remove the chip to get an understanding of even that much deeper.
- **Non-destructive:**Techniques which can be used are side-channel attacks, functional testing method and formal verification.These approaches are trying to reach for detecting abnormities in the chip's behaviour while still leaving the chip intact.

To defend against hardware Trojans, designers are working on what is called scan-based design-for-testability These include the DfT methodologies [Wang et al., 2022], and the possibilities to use such approaches are being considered in relation to two categories of products: These include such as: the consideration of security measures into the design

phase, by employing hardware based security modules and also having strong supply chain verification protocols. It becomes significant to ensure that supply chain is maintained adequately.

To counter hardware Trojans, researchers are developing scan-based design-for-testability (DfT) methodologies . These include incorporating security features into the design phase, using hardware-based security modules, and implementing robust supply chain verification protocols. Ensuring the integrity of the supply chain is crucial. Research in this area focuses on techniques like secure design practices, provenance tracking, and establishing trusted foundries to minimize the risk of Trojan insertion during manufacturing.

Beyond detection, developing strategies to mitigate the impact of hardware Trojans is essential. This includes designing systems that can tolerate or quickly recover from Trojan-triggered faults, as well as real-time monitoring and adaptive responses.

Overall, research on hardware Trojans is multidisciplinary, involving fields such as electrical engineering, computer science, cybersecurity, and supply chain management. The goal is to evaluate Systems on Chip (SoCs), especially the software-hardware interface (internal chip attacks that may originate from malicious software and/or hardware: viruses, Trojans, spyware, hardware viruses), that ensure the security and integrity of hardware systems in an increasingly interconnected and vulnerable world.

1.2 Motivation Of The Research

The motivation behind research in the area of hardware Trojans is driven by the critical need to ensure the security, integrity, and reliability of electronic systems. As technology continues to advance and integrate into every aspect of modern life, the risks associated with hardware vulnerabilities become increasingly significant.

One major motivation is national security concerns. Hardware Trojans can compromise the functionality and confidentiality of military and defense systems, leading to potential national security threats (Figure 1.1). Ensuring the integrity of defense equipment is paramount to prevent espionage and sabotage. Additionally, critical infrastructure such as power grids, communication networks, and transportation systems are essential to national security. A compromised hardware component in these systems can cause widespread disruptions and pose severe risks to public safety.

The economic impact of hardware Trojans is another significant motivator. Hardware Trojans can be used to steal intellectual property, leading to substantial financial losses for companies and undermining competitive advantage. Furthermore, ensuring a secure supply chain is critical for preventing the insertion of Trojans during the manufacturing process. Compromised hardware can lead to costly recalls and damage to brand reputation.

Consumer protection is also a crucial factor. With the increasing use of personal devices and IoT, hardware Trojans can be used to access sensitive personal data. Ensuring the security of consumer electronics protects users' privacy and trust. Additionally, malicious

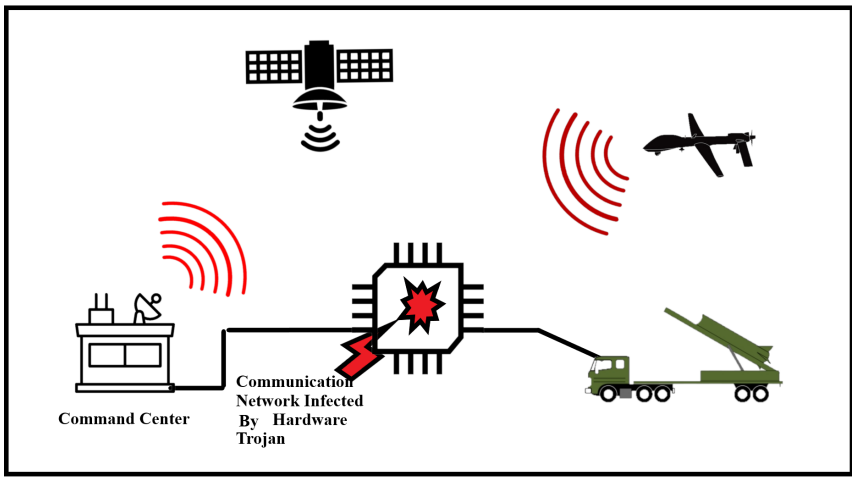


Figure 1.1: Hardware Trojan Threats In Military Systems.

hardware can cause devices to malfunction, leading to safety hazards. Ensuring the integrity of consumer products is essential for user safety and confidence.

Technological advancements also drive the need for hardware Trojan research. Modern electronic systems are highly complex, making it challenging to detect and mitigate hardware Trojans. Research in this area aims to develop advanced detection and prevention techniques. The rapid growth of IoT devices increases the potential attack surface for hardware Trojans. Ensuring the security of these interconnected devices is critical to maintaining a secure network ecosystem.

Regulatory and compliance requirements further motivate this research. Compliance with industry standards and regulations necessitates robust security measures to prevent hardware Trojans. Research helps develop methodologies that meet these standards. Additionally, governments are increasingly recognizing the need for secure hardware and are enacting regulations that require the implementation of security measures against hardware Trojans.

Finally, advances in attack techniques highlight the need for ongoing research. The threat landscape is constantly evolving, with attackers developing new techniques to insert and activate hardware Trojans. Continuous research is necessary to stay ahead of these evolving threats and develop effective countermeasures. Modern hardware Trojans can be highly sophisticated, making them difficult to detect using traditional methods. Research aims to innovate new detection and mitigation strategies to address these sophisticated threats.

In conclusion, the motivation behind hardware Trojan research is multifaceted, encompassing national security, economic stability, consumer protection, technological advancement, regulatory compliance, and the evolving nature of threats. Ensuring the security and integrity of electronic systems in this context is crucial for maintaining trust and safety in an increasingly connected world.

1.3 Problematic In The Hardware Trojan Research

The problematic in the hardware Trojan research area revolves around several key challenges that complicate the detection, mitigation, and prevention of these malicious alterations in hardware systems (Table 1.1). One significant issue is the difficulty in detecting hardware Trojans due to their sophistication and stealth. Modern Trojans can mimic legitimate circuitry and remain undetected for long periods, and they can be very small, activating only under specific conditions. This complexity makes standard testing and inspection techniques inadequate. Additionally, the diverse nature of hardware Trojans means there is a lack of standardized detection methods that can address all types. Current testing techniques are often limited and may not be comprehensive enough to identify deeply embedded or conditionally triggered Trojans.

Challenge	**Statistic**	**Source**
Sophistication and Stealth	Only about 30% of hardware Trojans are detected using current methods.	DARPA Hardware Security Report
Size and Activation Mechanisms	40% of hardware Trojans can be smaller than 1% of the total chip area.	Semiconductor Research Corporation (SRC)
Diverse Nature of Trojans	Over 50% of hardware security experts see the lack of standardized techniques as a major barrier.	Industry Research
Limited Testing Techniques	60% of companies consider their current testing methods inadequate for detecting Trojans.	IEEE Survey
Complex and Global Supply Chains	Global semiconductor supply chain involves over 50 countries and numerous vendors.	Global Semiconductor Supply Chain Report
Insider Threats	Insider threats account for approximately 20% of supply chain breaches.	National Counterintelligence and Security Center (NCSC) Report
Cost of Detection and Mitigation	Implementing comprehensive security measures can increase chip production costs by up to 30%.	McKinsey & Company Study
Resource Limitations in SMEs	SMEs have less than 10% of the budget required for advanced hardware security measures.	National Institute of Standards and Technology (NIST) Report
Rapid Evolution of Trojans	New types of hardware Trojans are developed approximately every 18 months.	Security Researchers' Notes
Integration with Emerging Technologies	IoT devices are expected to reach 30 billion by 2030, expanding the attack surface.	IoT Device Growth Statistics
Lack of Comprehensive Regulations	Less than 25% of countries have specific regulations addressing hardware security.	Global Hardware Security Regulations Report
Compliance Burdens	Companies spend up to 15% of their annual IT budget on compliance with varying regulations.	IT Budget Compliance Report
Need for Multidisciplinary Approaches	Over 70% of academic papers in this field involve cross-disciplinary research.	Cross-disciplinary Research in Hardware Security
Integration of Security in Design	Only 35% of hardware designers consistently integrate security considerations from the design phase onwards.	Electronic Design Automation (EDA) Consortium Survey

Table 1.1: Challenges and Statistics in Hardware Trojan Research Area.

Supply chain vulnerabilities further exacerbate the problem. The complexity and global nature of supply chains make it challenging to ensure integrity at every stage of manufacturing and assembly, with potential insider threats adding another layer of risk [Jain et al., 2021]. Economic and resource constraints also play a role, as implementing comprehensive security measures can be costly, and smaller companies may lack the necessary resources and expertise. The rapid evolution of attack techniques means

that attackers are continuously developing new and more sophisticated Trojans, staying ahead of current detection and prevention methods (Figure 1.2). Emerging technologies like IoT, AI, and quantum computing introduce additional vectors for hardware Trojans, complicating the security landscape.

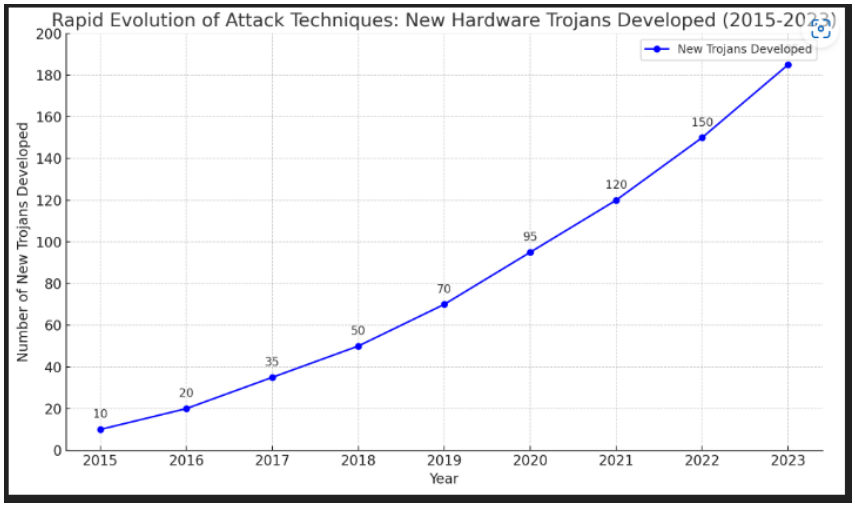


Figure 1.2: Estimation Of The Rapid Evolution Of hardware Trojan.

Regulatory and compliance challenges add another layer of complexity. There is often a lack of comprehensive regulatory frameworks specifically addressing hardware security and Trojans, leading to inconsistent security practices. Companies operating in multiple jurisdictions face the burden of complying with varying regulations. Lastly, addressing hardware Trojans effectively requires multidisciplinary approaches, integrating efforts from electrical engineering, computer science, cybersecurity, and supply chain management. Ensuring security from the design phase onwards requires a shift in mindset and practice among hardware designers and manufacturers.

In conclusion, the problematic in the hardware Trojan research area is multifaceted, involving technical, economic, and regulatory challenges. Overcoming these obstacles requires continuous innovation in detection and prevention techniques, comprehensive security practices across supply chains, sufficient economic investment, and the development of standardized regulatory frameworks. Addressing these challenges is critical to ensuring the security and integrity of electronic systems in a world increasingly reliant on complex and interconnected technologies.

1.4 Research Methodology

After clearly framing the problem and research objectives, we have attempted to develop a security evaluation approach. In our research, we have adopted a dual-approach methodology to address the problem of hardware Trojan detection and data security. The two primary approaches are as follows:

1. **Hardware Trojan Externally Activated Detection Technique:**

- **Objective:** This approach aims to detect externally activated hardware Trojans that could compromise the integrity and functionality of hardware systems.
- **Method:** We utilize a combination of anomaly detection algorithms and signal analysis to identify unusual patterns that may indicate the presence of a Trojan. By monitoring external activation signals and comparing them with expected behavior, we can pinpoint deviations that suggest malicious alterations.

2. **Using a Hardware Security Module (HSM) to Protect Data:**
   - **Objective:** This approach focuses on enhancing data security by encrypting sensitive information using a Hardware Security Module.
   - **Method:** The HSM provides a secure environment for cryptographic operations, ensuring that data is encrypted before transmission or storage. We implement strong encryption algorithms within the HSM, protecting data from unauthorized access and tampering even if a Trojan is present in the hardware.

This methodology leverages both detection (Chapter 4) and prevention (Chapter 5) strategies to address the multifaceted challenges posed by hardware Trojans, enhancing the overall security of hardware systems.

Scientific Productions

The scientific papers that were realized during this thesis are:

- Guechi, B., & Redjimi, M. (2020, October). Hardware Trojan Detection in Heterogeneous Systems on Chip. In The Proceedings of the Third International Conference on Smart City Applications (pp. 1105-1116). Cham: Springer International Publishing.
- Guechi, B., & Redjimi, M. (2023). Hardware Security Module Cryptosystem Using Petri Net. Indonesian Journal of Electrical Engineering and Informatics (IJEEI), 11(2), 494-502.

1.5 Organization Of The manuscript

The manuscript is structured into five distinct chapters (Figure 1.3), each addressing critical aspects of hardware trojans and their mitigation strategies.

Chapter 1, "Introduction," lays the groundwork for the manuscript by providing a comprehensive overview of the research context, motivation, problematic areas within the hardware trojan domain, and the adopted research methodology.

Chapter 2, "General Concepts of Hardware Trojans," serves as the foundational section, providing an in-depth exploration of the fundamental principles underlying hardware trojans. It encompasses topics such as the definition of hardware trojans, the threats they pose, critical fields affected by trojans, taxonomy, classification, activation mechanisms, and their architectural design.

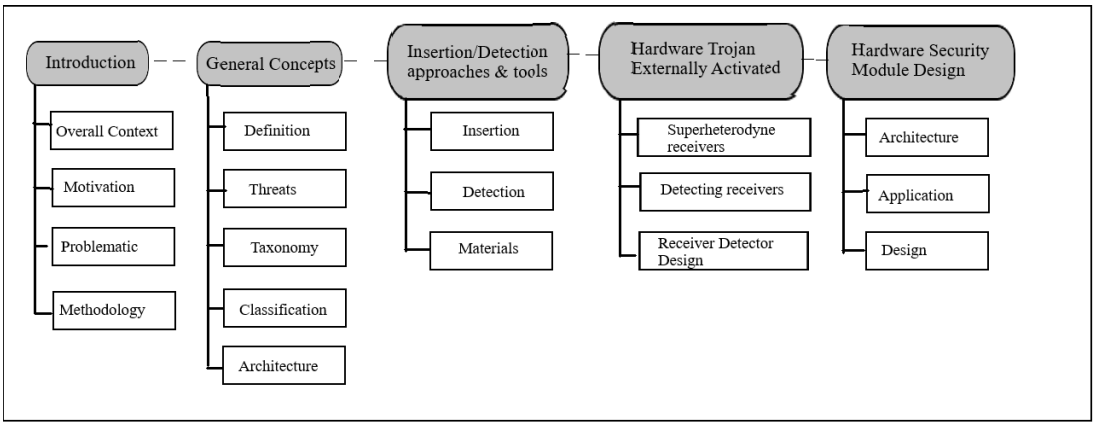


Figure 1.3: Organization Of The Manuscript.

Chapter 3, "Hardware Trojan Insertion/Detection Principle Approaches and Tools," delves into the methodologies and tools employed in both the insertion and detection of hardware trojans. This chapter meticulously examines insertion approaches, detection techniques, and materials utilized in hardware trojan mitigation.

Chapter 4, "Hardware Trojan Externally Activated Detection Technique," shifts focus to a specific detection method involving response examination of the hardware to specific external signals or triggers. It elucidates the principles of superheterodyne radio receivers, detecting such receivers, and the methodology for designing receiver detectors.

Chapter 5, "Hardware Security Module To Counter Hardware Trojans," introduces hardware security modules (HSMs) as a potent countermeasure against hardware trojans. It delineates the architecture, applications, advantages, and characteristics of HSMs, presenting a detailed design tailored to mitigate hardware trojan threats effectively.

Overall, these chapters provide a comprehensive framework for understanding, detecting, and mitigating the risks posed by hardware trojans in modern electronic systems.

Chapter 2

General Concepts of Hardware Trojans

2.1 Introduction

This chapter is about 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 most concepts from available literature on hardware Trojans.

2.2 Definition

A hardware Trojan is a malicious modification of the circuitry of an Integrated Circuit (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 premade 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.

2.3 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.

2.3.1 IC market model

In the context illustrated in Figure 2.1, 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 2.1).

Figure 2.1 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).

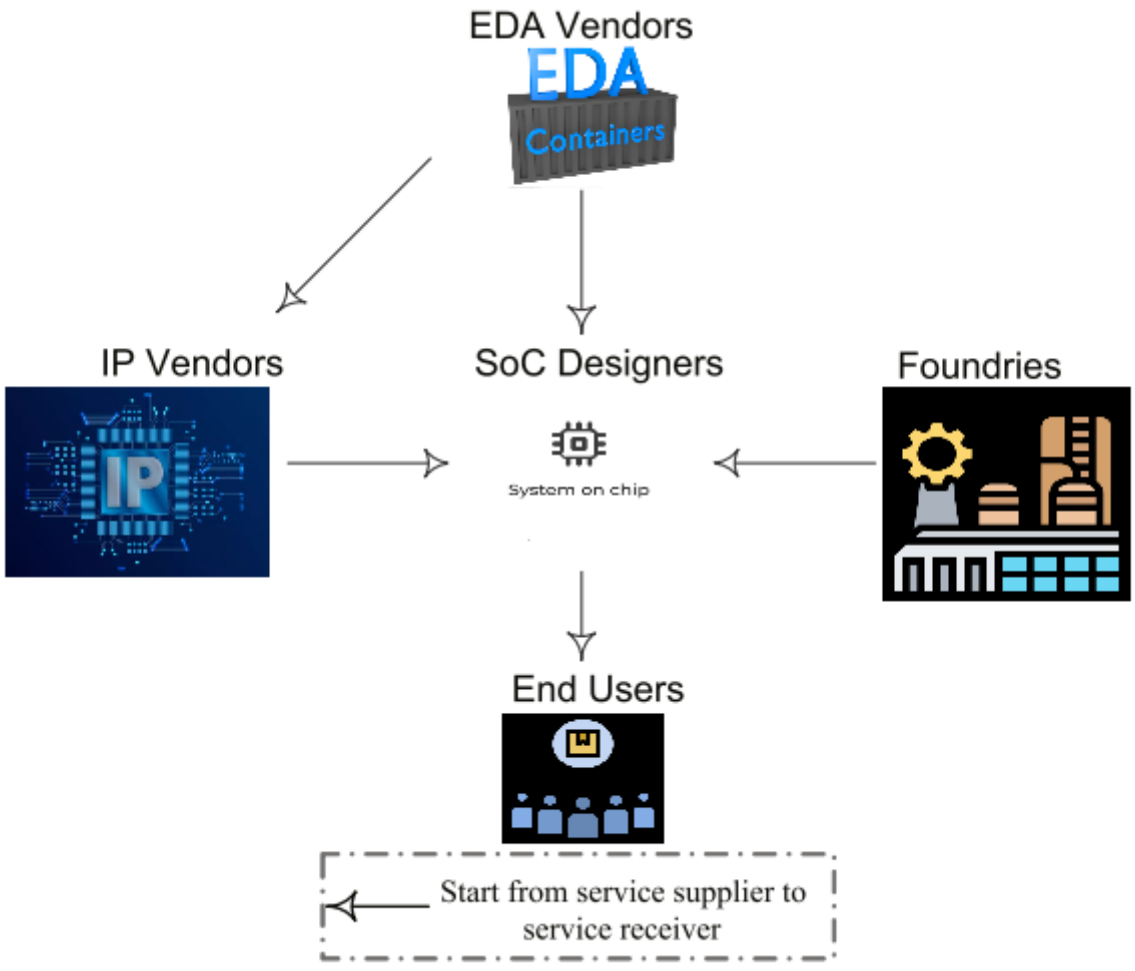


Figure 2.1: IC market model

2.3.2 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.

2.3.3 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 pre-silicon 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.

2.3.4 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.

2.3.5 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, 2012].

2.4 Critical fields affected by Trojans

2.4.1 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.

2.4.2 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.

2.4.3 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 these 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.

2.5 Trojan Taxonomy and Classification

This section is dedicated to present a 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.

2.5.1 Banga et al. classification

In their study [Banga et al., 2017], 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.

2.5.2 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 2.2 according to their physical, activation, and action characteristics.

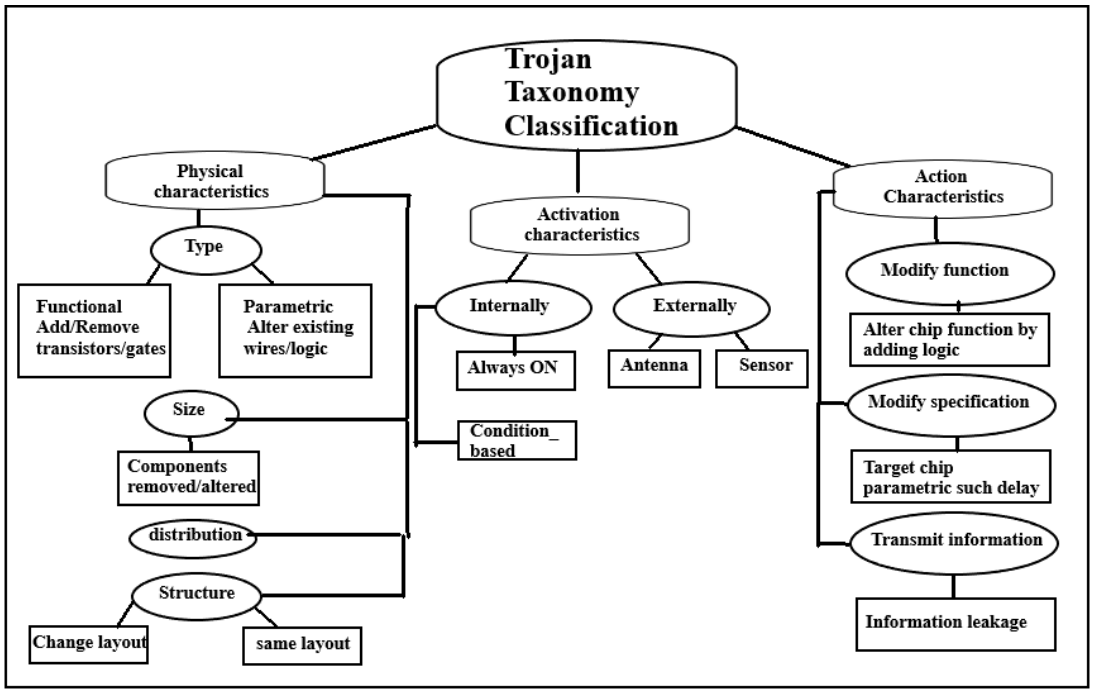


Figure 2.2: Taxonomy of Trojans

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 2.2: 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 2.2 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.

2.5.3 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. Figure2.3 illustrates the Bug-Based HT impact on a target circuit.

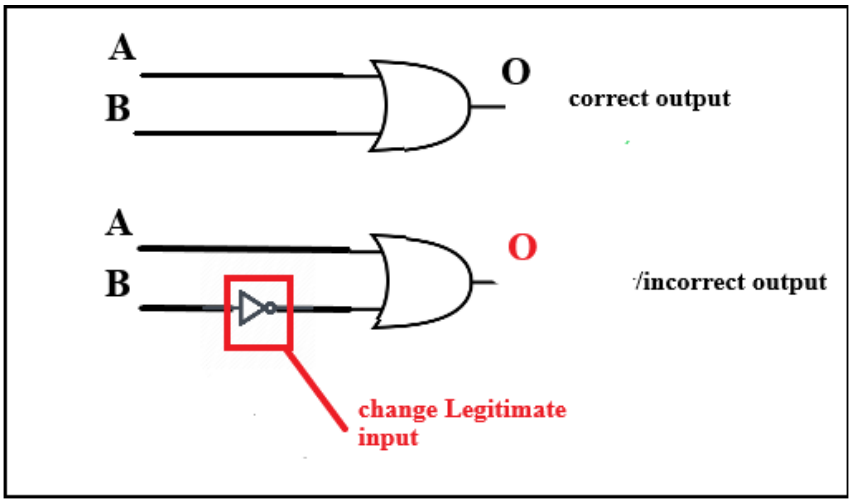


Figure 2.3: Bug-Based HT using logic inverter

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 2.4 , 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.

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

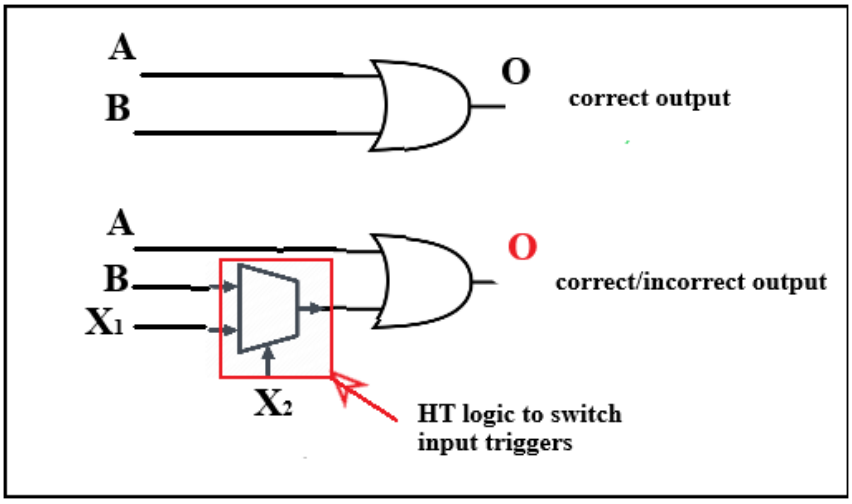


Figure 2.4: Parasite-Baset HT using logic Multiplexer to trigger malicious input.

2.5.4 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 2.5 , 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.

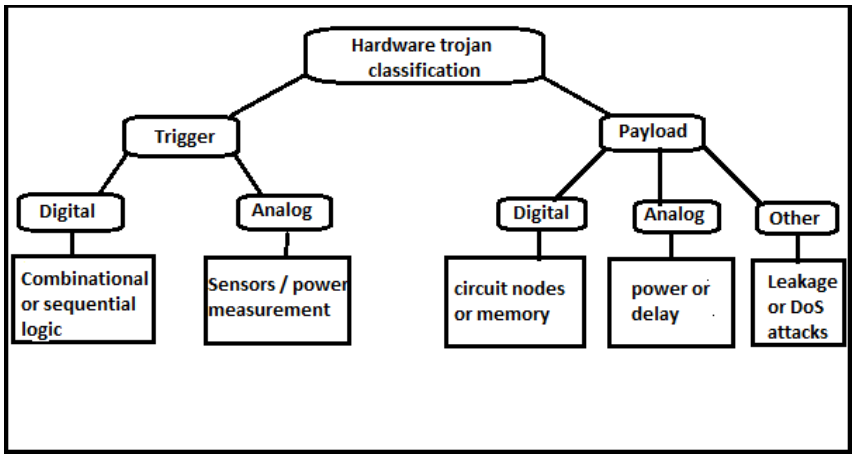


Figure 2.5: Analog/Digital HT classification

2.5.5 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 2.6).

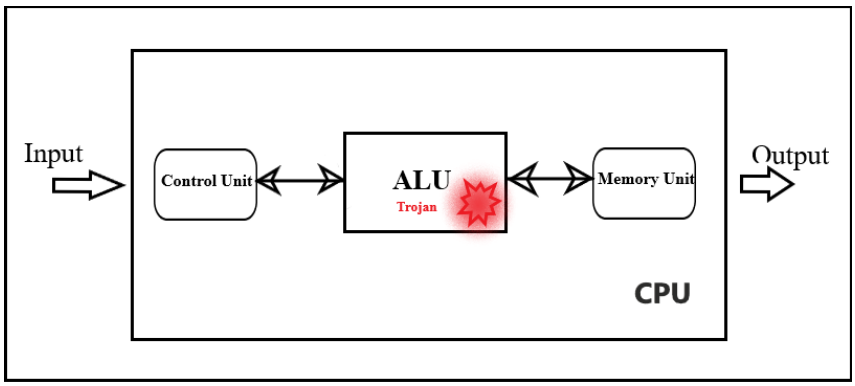


Figure 2.6: Ip-level HT

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 2.7).

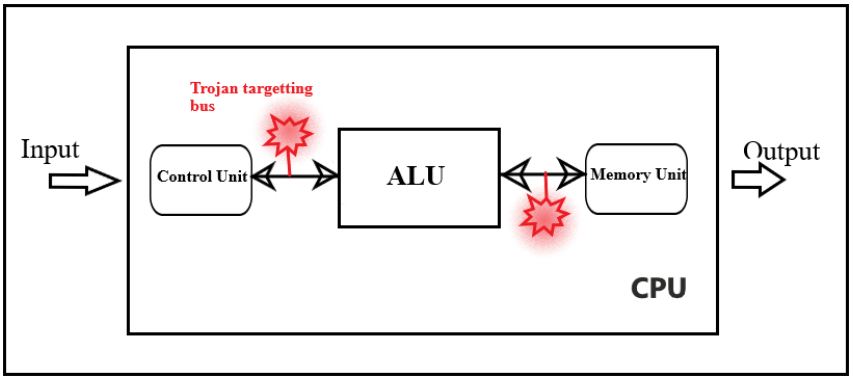


Figure 2.7: Bus-level HT

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 2.8).

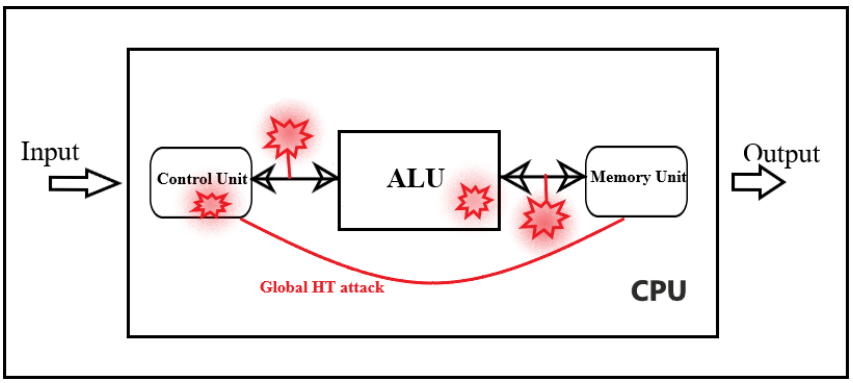


Figure 2.8: SoC-level HT

2.6 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.

2.6.1 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.

2.6.2 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.

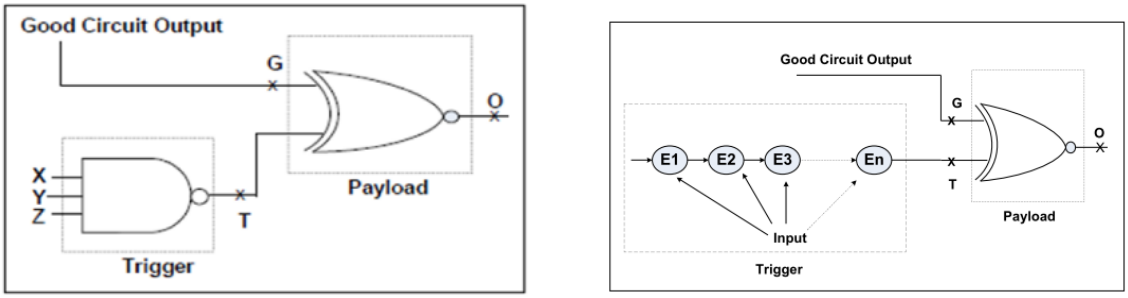
Compared to sequential trigger mechanisms that may rely on specific sequences of inputs over time, combinational triggers are more instantaneous, making them potentially harder to detect during functional testing. Researchers and security analysts need to develop advanced methods to identify and counter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

Figure 2.9.(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.

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



(a) Combinational trigger mechanism

(b) Sequential trigger mechanism

Figure 2.9: Combinational and Sequential trigger mechanism. From "Preventive Techniques for Hardware Trojans" by Das, M.K. ,Masaryk University, Faculty of Informatics, December 2016, Master's Thesis, P.7 .

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 2.9.(b).

Figure 2.10 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 2.9.(b)) .

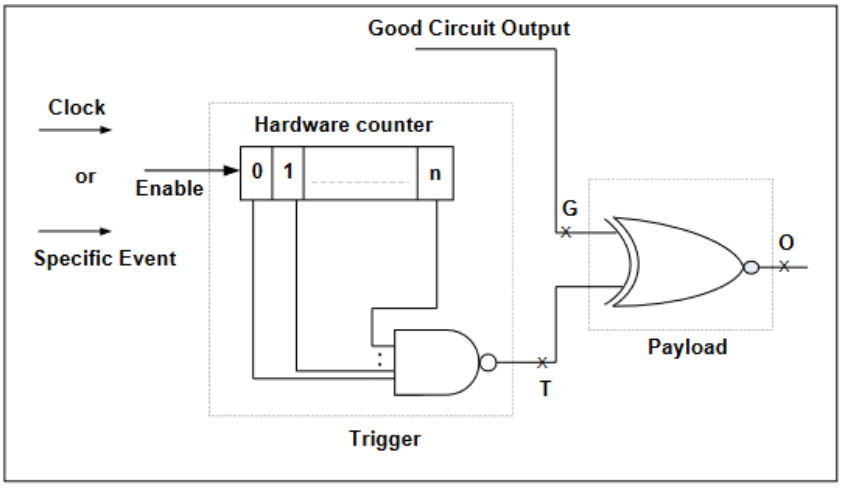


Figure 2.10: Hardware counter triggering Trojan. From "Preventive Techniques for Hardware Trojans" by Das, M.K. ,Masaryk University, Faculty of Informatics, December 2016, Master's Thesis, P.9 .

2.7 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.

2.7.1 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. 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].

2.7.2 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.

2.8 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:

[3]ATPG s an electronic design automation method 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 (https://www.fpgakey.com/wiki/details/68)

2.8.1 Data Leakage

The hardware Trojan could be designed to leak sensitive information or data from the affected system (Figure 2.11). This could involve transmitting confidential data to an external entity without the knowledge of the system owner.

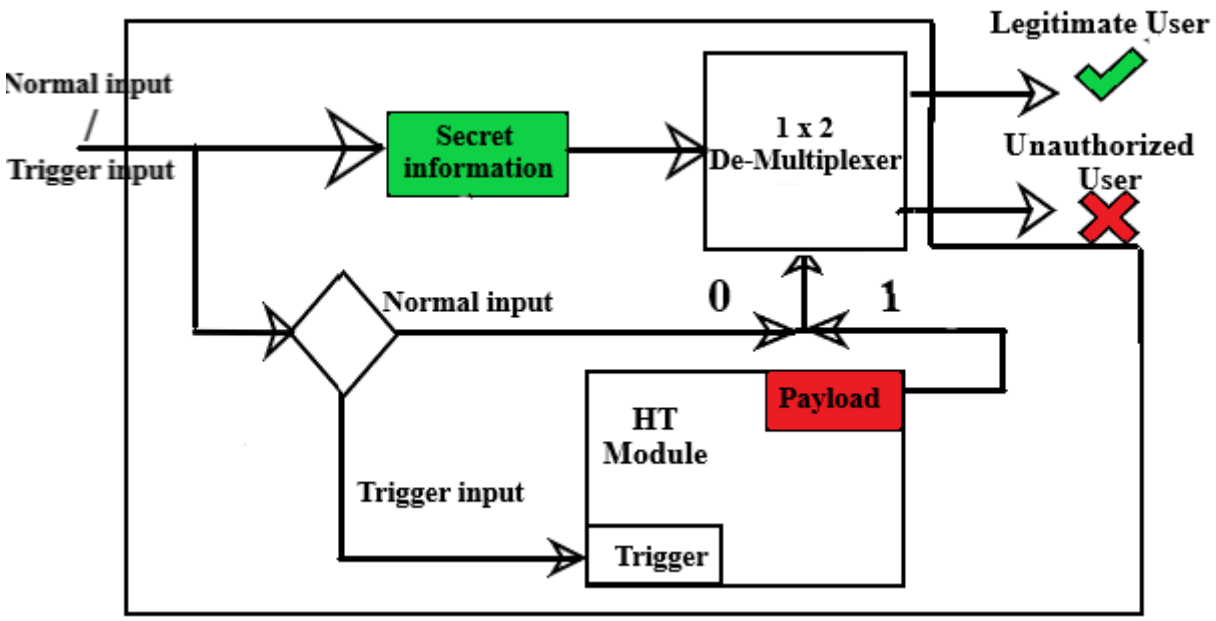


Figure 2.11: Information leakage hardware Trojan.

2.8.2 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.

2.8.3 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.

2.8.4 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.

2.8.5 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.

2.9 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 2.12 shows the classification of payload logic based on the operation performed when Hardware Trojan is activated.

2.9.1 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

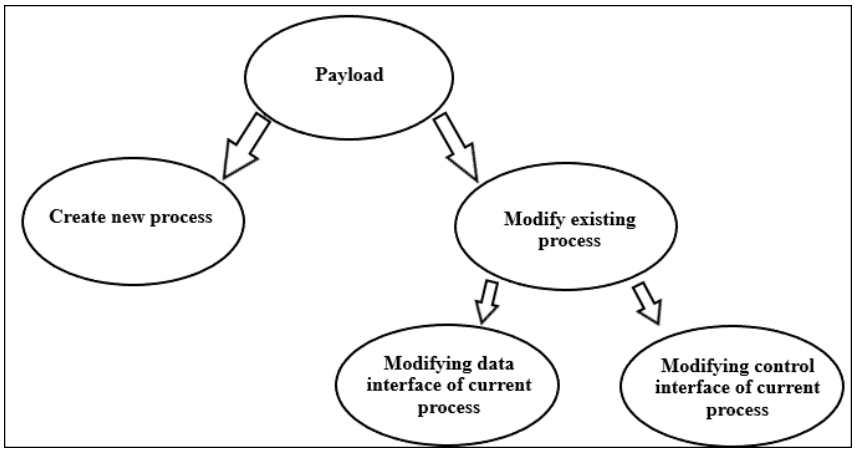


Figure 2.12: Operation Based payload classification.

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.

2.9.2 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.

2.9.3 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 en-

ables 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.

2.10 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 2.13).

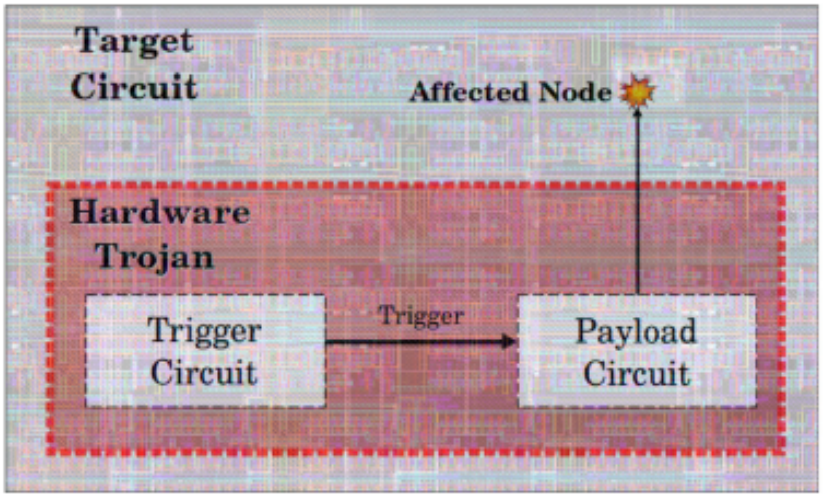


Figure 2.13: Architecture of a Trojan inserted on a target circuit. From "Testing Techniques for Detection of Hardware Trojans in Integrated Circuits of Trusted Systems." by Guimaraes, L.A. , Université Grenoble Alpes, 2017, Phd Thesis, P.8 .

2.10.1 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 2.14). 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.

2.10.2 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.

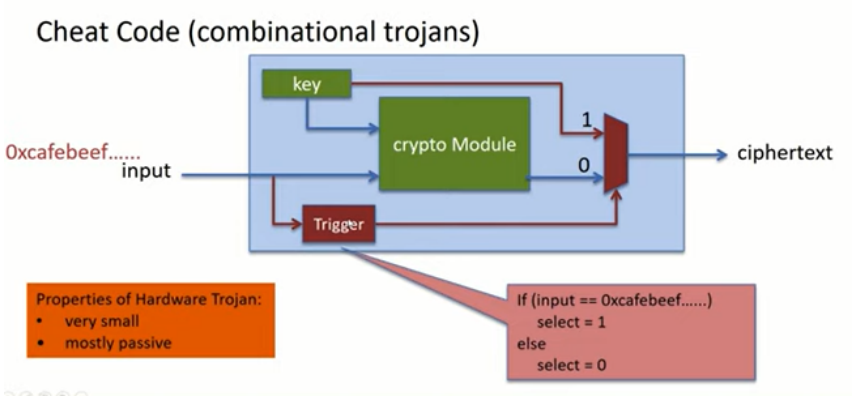


Figure 2.14: Combinational Trojan. From "Hardware Security lecture slides", Indian Institue of Technology Madras.

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 2.15. 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).

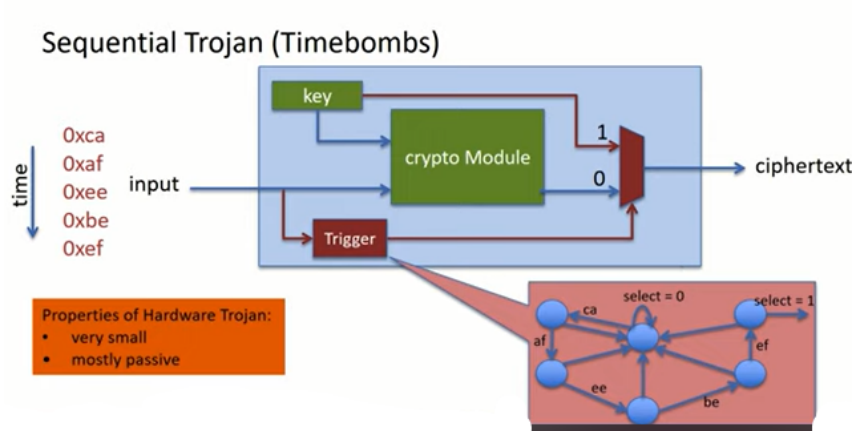


Figure 2.15: Sequential Trojan (Timebombs). From "Hardware Security lecture slides", Indian Institue of Technology Madras.

2.10.3 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 2.16).

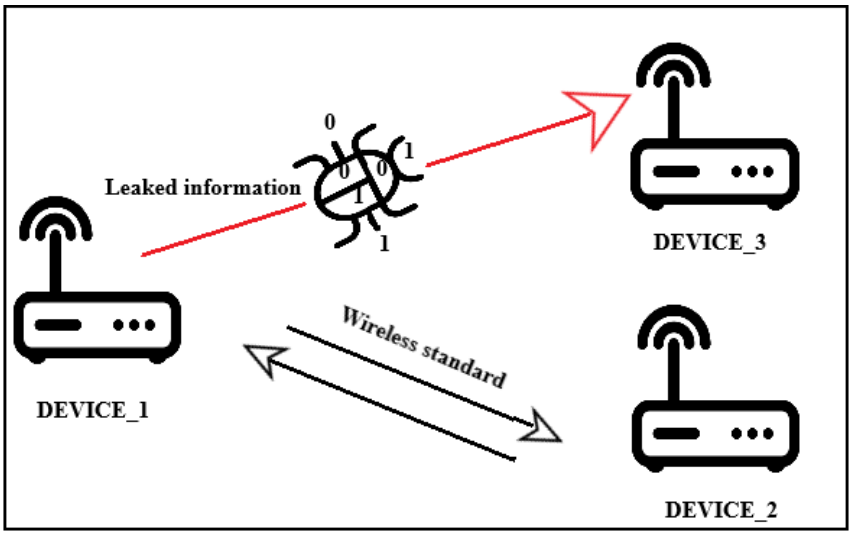


Figure 2.16: Threat model

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 2.17. 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.

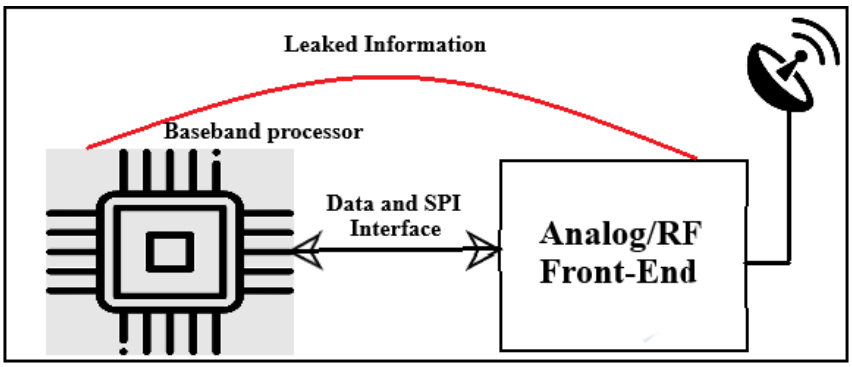


Figure 2.17: Amplitude modulating hardware Trojans.

2.10.4 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.

2.10.5 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.

2.11 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.

2.11.1 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.

2.11.2 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.

2.11.3 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.

2.11.4 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.

2.11.5 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.

2.12 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.

2.12.1 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.

2.12.2 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.

2.12.3 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.

2.13 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.

2.13.1 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.

2.13.2 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.

2.13.3 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.

2.13.4 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.

2.13.5 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.

2.13.6 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.

The comparison table (Table 2.1) outlines three types of hardware Trojan attacks: combinational, sequential, and parametric. Combinational Trojans are triggered by specific input combinations, instantly altering circuit functionality, with moderate detection difficulty through logic testing. Sequential Trojans require a specific event sequence to activate, conditionally modifying control logic, and are harder to detect due to their complex triggers. Parametric Trojans subtly change parameters like timing or power, continuously degrading performance with very high detection difficulty, often requiring side-channel analysis. Each type varies in invasiveness, activation method, and impact, highlighting the challenges in securing hardware against diverse Trojan threats.

Characteristic	**Type 1: Combinational Trojan**	**Type 2: Sequential Trojan**	**Type 3: Parametric Trojan**
Trigger Mechanism	Rare event in combinational logic	Specific sequence of events	Subtle change in parameters
Payload	Alter data path or logic function	Modify control logic	Degrade performance subtly
Invasiveness	High	Medium	Low
Activation	Instant	Requires specific conditions	Continuous
Detection Difficulty	Moderate	High	Very High
Impact	Direct alteration	Conditional alteration	Gradual performance degradation
Example	XOR gate insertion	State machine modification	Threshold voltage alteration
Countermeasures	Logic testing	Temporal analysis	Side-channel analysis

Table 2.1: Comparison of Hardware Trojan Attack Types

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.

2.14 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.

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 following code 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.

```
1    module top(clk, rst, state, key, out);
2    input       clk,rst;
3    input       [127:0] state, key;
4    output      [127:0] out;
5
6            aes_128 AES(clk, state, key, out);
7            Trojan_Trigger Trigger(rst, state, Tj_Trig);
8            TSC Trojan (clk, rst, key, Tj_Trig);
9    endmodule
```

Listing 2.1: Architecture of the Trojan-infected AES-T2000 benchmark

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

2.15 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 3

Hardware Trojan Insertion/Detection Principle Approaches and Tools

3.1 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.

3.2 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.

During the design phase, register-transfer level (RTL) serves as an abstraction that represents a synchronous digital circuit by describing how digital signals (data) move between hardware registers and the logical operations applied to them. This abstraction is utilized in hardware description languages (HDLs) such as Verilog and VHDL to create high-level circuit models, from which lower-level designs and physical wiring are developed. However, the use of untrusted third-party IPs and code during this process can compromise the security of the entire design.

Logic synthesis is a process by which an abstract specification of desired circuit behaviour, 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 [Simons, 2022]. 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. Place and route are 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.

The 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 modelling 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. Contrary to the previous phases, the verification step is typically performed in the local foundry, which qualifies it to be trusted for the use of certified tools and test benches.

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.

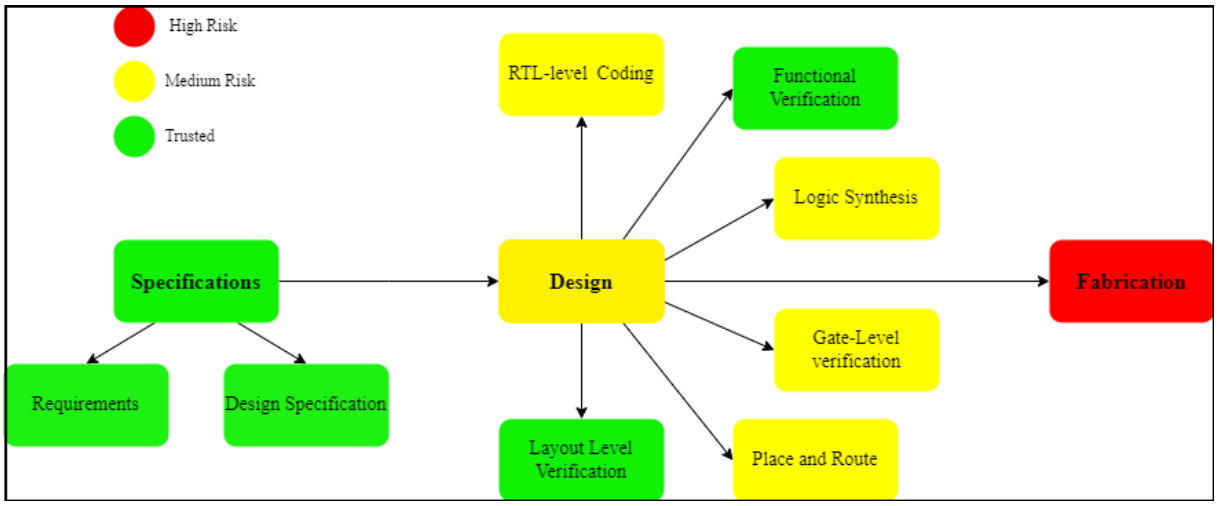


Figure 3.1: Hardware trojan insertion in circuit design phases

To assess the risks associated with Hardware trojans (HTs), various studies, such as [Perez and Pagliarini, 2023], have presented taxonomies. These taxonomies abstract different categories related to the architecture, effects, and insertion of Trojans in Integrated Circuits (ICs).

3.2.1 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.

3.2.2 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.

3.2.3 HT mounting outside the IC

Mounting Hardware Trojans (HT) externally to the Integrated Circuit (IC) is simpler, requiring surface mounting on existing components. This external placement offers easier installation compared to embedding HTs within the IC. Successful mounting of an HT outside the IC relies on accessing the target device for a specific time period. Although not as integrated as internal placement, various circuit elements connect to the board. Detecting these circuits separately from elements mounted during design is considered challenging. In cases like communication lines, where circuit elements might not be present, they are often completely covered with coating. Camouflaging the HT by covering it further complicates detection.

In [Kinugawa et al., 2019] survey, pertains to an HT that is capable of being installed on the peripheral circuits and wiring of Integrated Circuits (ICs). This Hardware Trojan is associated with the risk of information leakage via electromagnetic waves. Unintentional electromagnetic emissions can lead to information leakage. However, intentional electromagnetic interference can be employed to control the timing and scope of this information leakage. Consequently, this example of a threat can be viewed as a novel challenge that encompasses both emission security and disturbance security within the Electromagnetic Compatibility (EMC) field.

In Figure 3.2, the HT is composed of MOSFET and short wiring. This Hardware Trojan (HT) is installed on a board or connecting cable, and upon receiving the trigger pattern, it releases leaked data. This is achieved by utilizing the wiring pattern on the board and the line connected to the device as an antenna, referred to as the unintentional antenna.

The discussed Hardware Trojan functions similarly to a mixer circuit that multiplies two signals. Initially, the HT produces an amplitude modulation (AM) signal by multiplying a target signal within the device, intended for leakage, with a signal emitted from the external environment acting as a carrier signal. Following this, the AM signal is transmitted outside the device via an "unintended antenna." Finally, by demodulating the transmitted signal, the internal device signal can be recovered.

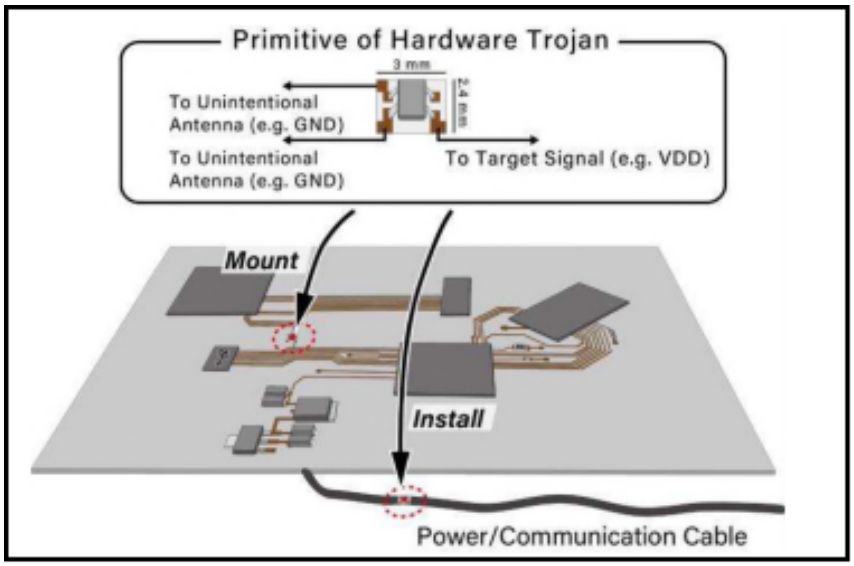


Figure 3.2: HT mounted outside of the IC. From "Survey of Hardware Trojan Threats and Detection" by Yuichi, H. , Shinichi, K. , International Symposium on Electromagnetic Compatibility – EMC Europe, 2020.

3.2.4 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.

3.3 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 Figure 3.3.

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 de-metallization using Chemical Mechanical Polishing (CMP)[1], followed by Scanning Electron Microscope (SEM) image reconstruction and analysis [Shi et al., 2019]. The non-destructive methods

[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.

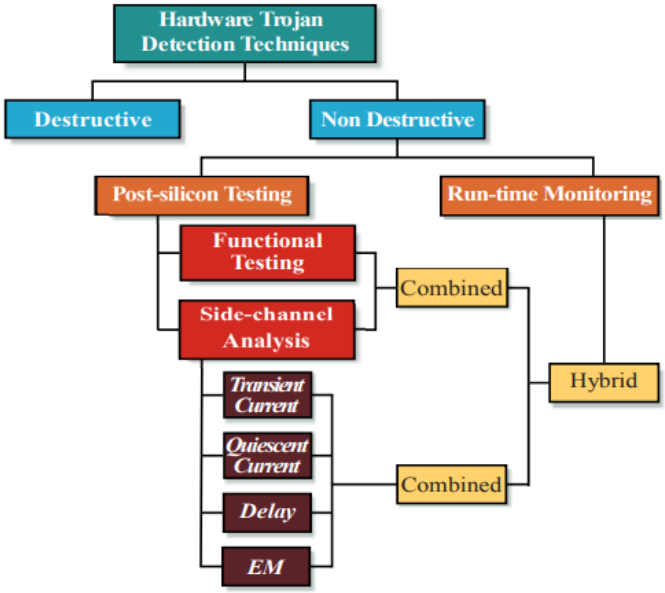


Figure 3.3: Classification of hardware Trojan Detection Techniques. From "Testing Techniques for Detection of Hardware Trojans in Integrated Circuits of Trusted Systems." by Guimaraes, L.A. , Université Grenoble Alpes, 2017, Phd Thesis, P.16 .

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.

3.3.1 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 fabrication 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.

3.3.2 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.

3.3.3 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.

3.4 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).

3.4.1 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.

3.4.2 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.

3.4.3 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. The system's model and specification are formulated in 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 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.

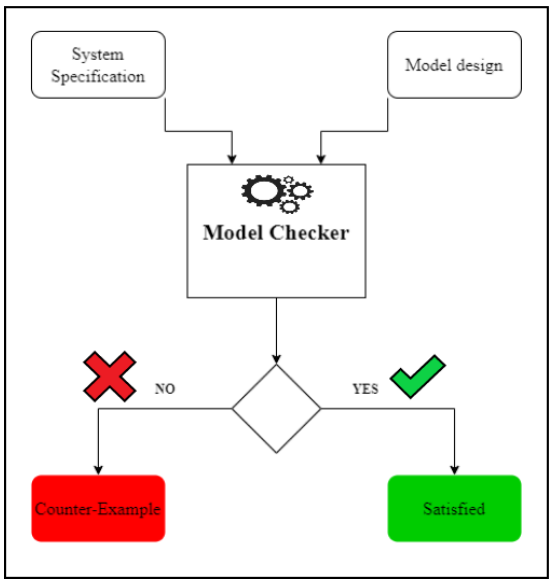


Figure 3.4: Model checker

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 [Bình, 2023]. 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

3.4.4 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.

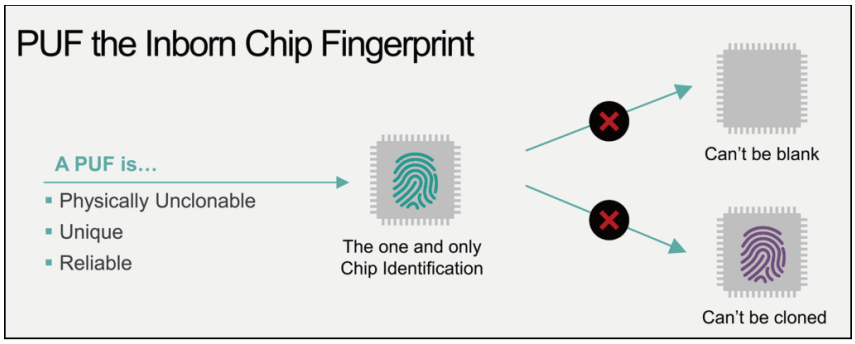


Figure 3.5: PUF-based Security IP Solution. From "https://www.pufsecurity.com/technology/puf/", Visited on September 2024.

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 3.6 .

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.

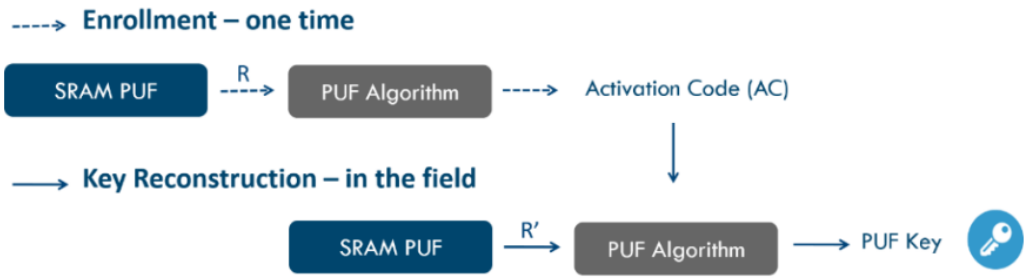


Figure 3.6: Error correction techniques for cryptographic key reconstruction. From "https://www.pufsecurity.com/technology/puf/", Visited on September 2024.

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 bi-stable 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.

3.4.5 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.

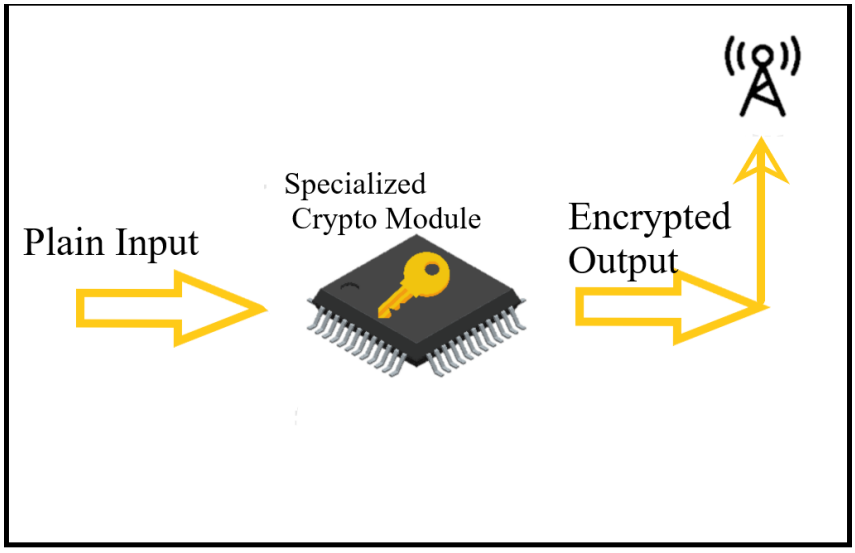


Figure 3.7: Hardware-based Encryption using Specialized Crypto Module.

Key characteristics of hardware-based encryption include:

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

3.4.6 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:

- **Cryptographic Functions:** TPMs have built-in cryptographic functions, such as generating and storing keys securely, performing digital signatures, and executing

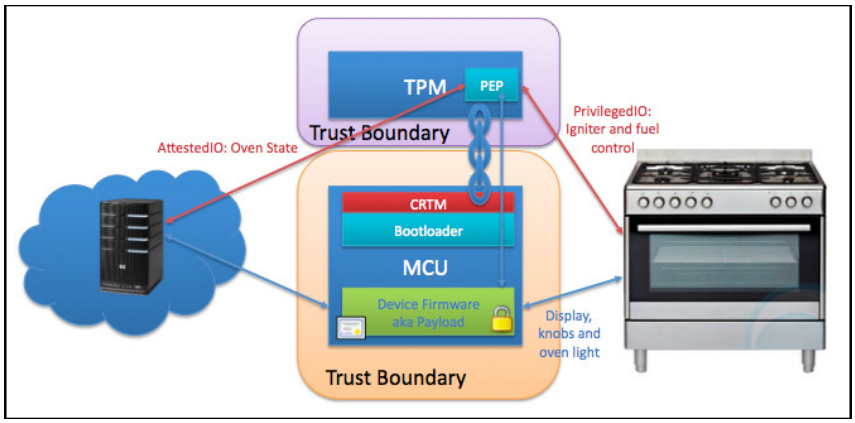


Figure 3.8: Trusted Platform Module (TPM) In Oven system. From "https://embeddedcomputing.com/", "Using a trusted platform module and trusted brokered IO as the foundation of IoT security", visited on October 2024.

hash functions. These capabilities enable the TPM to support various security applications.

- **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.
- **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.
- **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.
- **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.
- **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.
- **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.
- **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.

A Review of Some Studies on The Trusted Platform Module (TPM)

A model delineating the structure of a standard for Trusted Computing (TC), specifically focusing on the Trusted Platform Module (TPM), which is an architecture with numerous potential implementations [Muñoz and Fernandez, 2020]. The architecture of TPM includes cryptographic functions that verify the utilization of authentic hardware and software platforms, along with defining operations for executing trusted tasks. Certain versions of TPM are designed exclusively for secure storage of private keys. Despite its potential, this technology is frequently underutilized. The first step of this methodology is presented in the form of a security pattern in this paper. An effort has been made to offer a reasonable amount of information in the pattern description, targeting the requirements of application designers. The pattern is depicted in an abstract form, independent of implementation details, yet articulated with sufficient detail and precision for the utilization by designers.

In [Kim and Kim, 2019], Hybrid Implementation of Trusted Platform Module (HTPM) has been presented. Since hardware-based Trusted Platform Module (TPM) encounters various inherent issues, including significantly low performance, susceptibility to off-chip security vulnerabilities, and a lack of responsiveness in incident handling. As we approach the era of Quantum computing, it becomes imperative to offer cryptography functions that are Quantum-Resistant (QR) without compromising performance. A TPM solution based on software delivers superior performance, on-chip security, and agility in incident response. Nevertheless, it lacks hardware-backed protection and essential features such as secure key storage, resilience against side-channel attacks, and genuine random number generation, among other capabilities. The HTPM operates as a completely dual-mode TPM, allowing end-users complete autonomy to select either a hardware TPM mode or a software TPM mode based on their requirements. In this study they have conducted and furnish a comprehensive risk analysis of the proposed HTPM to demonstrate the most effective ways to address security challenges in implementing the HTPM. Lastly, they have presented a performance analysis of their proposition to showcase significant enhancements in cryptographic operations.

The growing frequency of cyber threats targeting Supervisory Control and Data Acquisition (SCADA) and automation systems within the Industrial Internet of Things (IIoT) and Industry 4.0 era has sparked apprehensions regarding the imperative to safeguard critical infrastructures and manufacturing facilities. The shift towards increased interconnection and interoperability has heightened the susceptibility of these systems. This vulnerability is particularly pronounced in the presence of widely used legacy standard protocols, which expose data to external networks. The objective of the paper [Tidrea et al., 2019] is to demonstrate the benefits of incorporating Trusted Platform Modules (TPMs) into automation/SCADA systems, emphasizing security enhancements. Two approaches are suggested to verify the authenticity of transmitted messages. The study also provides measurements regarding the additional time latency introduced by the proposed concept.

3.4.7 Dual and Redundant Hardware Designs

Dual and redundant hardware designs refer to an approach in system architecture where critical components or subsystems are duplicated, and there is a built-in redundancy to ensure continued operation in the event of a hardware failure. This redundancy is implemented to enhance system reliability, availability, and fault tolerance. In a dual and redundant hardware design, if one component fails, the redundant counterpart can seamlessly take over, minimizing downtime and maintaining system functionality. This approach is commonly employed in safety-critical systems, mission-critical applications, and industries where uninterrupted operation is crucial. The redundancy provides a safeguard against hardware failures and contributes to overall system resilience.

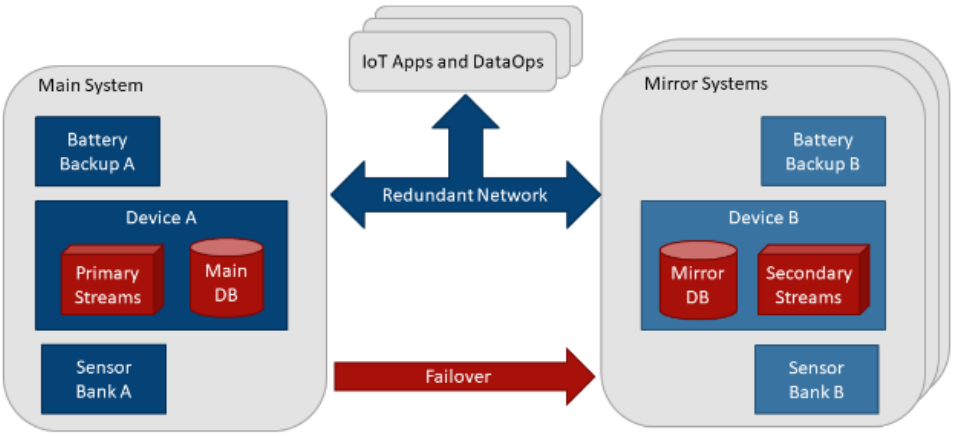


Figure 3.9: Disaster Recovery for Embedded IoT Edge Devices.

Implementing redundant hardware designs or dual-die configurations can provide a backup mechanism that helps mitigate the impact of a hardware Trojan by allowing the system to switch to a trusted component. Here are some key features of this technique:

- **Fault Tolerance:** Dual and redundant hardware designs are implemented to enhance fault tolerance. If one component fails, the redundant counterpart can seamlessly take over, ensuring continued operation.
- **Reliability:** The redundancy in hardware components contributes to improved system reliability. The likelihood of a complete system failure is reduced, as there are backup components ready to assume control in case of a failure.
- **Availability:** By having duplicate hardware components, dual and redundant designs contribute to increased system availability. The system can continue functioning even during hardware failures, minimizing downtime.
- **Hot Swapping:** Some designs allow for "hot swapping" of components, meaning that faulty hardware can be replaced or repaired without shutting down the entire system. This is crucial for maintaining continuous operation.
- **Safety-Critical Applications:** Dual and redundant hardware designs are often used in safety-critical applications, such as aviation, medical devices, and nuclear systems, where system failure could have severe consequences.
- **Automatic Failover:** The system is designed to automatically switch to the redundant hardware if a failure is detected, without requiring manual intervention. This automatic failover mechanism enhances system responsiveness.
- **Scalability:** Depending on the specific design, the redundancy can be scalable to meet the requirements of different applications. Additional redundant components can be added for increased reliability.
- **Monitoring and Diagnostics:** Dual and redundant hardware designs often include monitoring and diagnostic features to detect potential issues before they lead to a failure. This proactive approach helps in preventive maintenance.
- **Cost:** While the initial cost of implementing dual and redundant hardware designs may be higher, the potential cost savings from avoiding downtime and system failures can justify the investment, especially in critical applications.
- **Industry Standards:** Various industries may have specific standards and regulations for implementing dual and redundant hardware designs, especially in sectors where safety and reliability are paramount. Adherence to these standards is crucial for certification and compliance.

the implementation of dual and redundant hardware designs stands as a key strategy to fortify critical systems against potential failures, ensuring a resilient and dependable operation that aligns with the stringent requirements of safety-critical industries.

A Review of Some Studies on Dual and Redundant Hardware designs

An Architecture Design of Distributed Redundant Flight Control Computer Based on Time-Triggered Buses for UAVs have been presented In [Zhang and Zhao, 2021]. This architecture introduces novel design approaches for UAV flight control, including distributed fault-tolerant management, Byzantine fault-tolerant design utilizing dual-core self-monitoring, and an open/integrated method for processing and fusing information from multiple sensors airborne. Leveraging the characteristics of the redundant Flight Control Computer (FCC) based on time-triggered buses, a model for distributed task scheduling and communication is established. An optimal static scheduling and real-time analysis algorithm, grounded in a search tree, is then proposed for these distributed tasks. Subsequently, an analysis and validation of the real-time performance and reliability of the FCC are conducted. The results of the verification demonstrate that, in comparison to the centralized FCC architecture relying on the event-triggered mechanism, the suggested UAV FCC architecture exhibits superior task schedulability and system scalability. Furthermore, it demonstrates enhanced task reliability under equivalent redundancy configurations, indicating its potential to furnish a distributed, synchronized, fault-tolerant, and redundantly reconfigurable technology platform for future UAV FCCs.

The paper [Amin and Mahmood-ul Hasan, 2022] introduces a Unified Fault-Tolerant Control System (UFTCS) designed for Air-Fuel Ratio (AFR) control in Spark Ignition (SI) Internal Combustion (IC) engines, incorporating advanced analytical and hardware redundancies. The analytical redundancy, denoted as the Hybrid Fault-Tolerant Control System (HFTCS), integrates both active and passive components. The active part employs Lookup Tables (LTs), while the passive part features a robust proportional feedback controller with a high-gain fuel throttle actuator. To address the critical issue of engine shutdown resulting from the simultaneous failure of any two sensors or a single actuator, an advanced hardware redundancy protocol, Modified Triple Modular Redundancy (MTMR), is proposed for sensors, and Dual Redundancy (DR) is suggested for actuators to prevent engine tripping. Simulation results using MATLAB/Simulink demonstrate the UFTCS's robustness to sensor faults under normal and noisy conditions. Probabilistic reliability analysis of various hardware redundancy schemes further validates the superior overall reliability of the UFTCS. Finally, a comparative analysis with existing AFR control systems highlights its enhanced performance.

Safety-critical computing systems, found in domains like avionics or space, necessitate specific safety measures based on deployment criticality. One challenge encountered by these systems involves transient hardware failures. To address potential issues, a common approach is to introduce redundancy, such as utilizing two cores simultaneously running the same program. However, this redundancy alone does not mitigate all potential failures, like Common Cause Failures (CCF), where a single fault impacts both cores uniformly (e.g., a voltage droop). If both redundant cores share identical states during a fault, the possibility of CCF arises, as the fault can affect both cores in a similar manner. In [Bas et al., 2022], SafeDM is introduced, a Diversity Monitor in hardware that quantifies the diversity of each redundant processor to ensure that CCF will not be overlooked, all without the necessity of deploying lockstepped cores. Data and instruction diversity are computed separately by SafeDM, employing distinct techniques suitable for each case. Integration of SafeDM is carried out in a RISC-V FPGA space MPSoC from Cobham

Gaisler, where its effectiveness is demonstrated through extensive benchmarking, incurring minimal area and power overheads. In summary, an effective hardware solution in SafeDM is provided for quantifying diversity in cores engaged in redundant execution.

3.4.8 Anti-tamper Packaging

Hardware anti-tamper packaging involves the use of specialized materials, mechanisms, and technologies to safeguard electronic devices and components against unauthorized access, tampering, or reverse engineering. This is particularly important in applications where the security and integrity of the hardware are critical, such as in defense systems, secure communication devices, and other sensitive electronic equipment. Employing anti-tamper packaging techniques, such as coatings or encapsulation, can make it more challenging for attackers to physically access and tamper with the hardware.

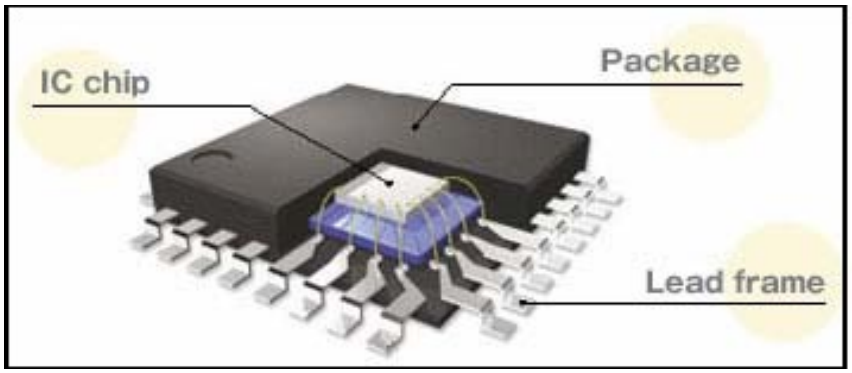


Figure 3.10: IC packaging process. adapted from electronics packaging tutorial.

Here are some common features and strategies used in hardware anti-tamper packaging:

- **Encapsulation and Potting:** Components or entire devices may be encapsulated or potted in a resilient material to make physical access difficult. This can prevent unauthorized individuals from tampering with or extracting sensitive information from the hardware.
- **Tamper-Evident Seals:** Special seals or coatings are applied to the hardware, and any attempt to breach the packaging results in visible damage or alteration. This provides a clear indication that tampering has occurred.
- **Anti-tamper Coatings:** Applying coatings that are resistant to probing, tampering, or chemical attacks can enhance the overall resistance of the hardware to unauthorized access.
- **Secure Enclosures:** Using robust and secure enclosures with complex locking mechanisms adds an additional layer of protection against physical tampering.
- **Self-Destructive Features:** Some anti-tamper packaging incorporates self-destructive features that trigger when tampering is detected, rendering the hardware inoperable or destroying sensitive data.
- **Intrusion Detection Systems:** Implementing sensors or systems that can detect physical tampering and trigger alerts or protective measures.

- **Unique Identifiers:** Embedding unique identifiers, such as serialized codes or cryptographic keys, into the hardware can help verify its authenticity and detect any attempts at tampering.
- **Secure Boot and Authentication:** Implementing secure boot processes and authentication mechanisms at the hardware level can prevent unauthorized access and ensure that only authenticated software can run.
- **Environmental Sensors:** Integrating sensors that monitor environmental conditions, such as temperature, humidity, or radiation, can help detect attempts to compromise the hardware.
- **Shielding Against Side-Channel Attacks:** Employing materials and techniques to protect against side-channel attacks, such as power analysis or electromagnetic analysis, which could be used to extract sensitive information.

Anti-tampering design encompasses a range of methods aimed at thwarting, identifying, or addressing physical assaults on a chip. It stands as an essential facet of chip security, executed through physical layout, digital RTL, or a blend of both. Typically manifesting as a protective shield, anti-tamper designs combat diverse attack forms, like data scrambling during writing or identifying glitches in circuitry. Consequently, the implementation of anti-tampering design is paramount in safeguarding sensitive information against the three primary tampering attack categories: **invasive, semi-invasive, and non-invasive.**

Invasive attacks initiate with the dismantling of the chip packaging and elimination of the passivation layer. Subsequently, through physical scrutiny, hackers can acquire access to the circuitry designs and security expertise. Alternatively, they may manipulate signals on the exposed chip to induce erroneous behavior or disrupt regular operations.

Semi-invasive attacks involve deliberately causing electrical malfunctions within a chip and subsequently observing the resulting consequences.

Non-invasive attacks are passive in nature, where hackers observe the regular functioning of a chip without making alterations or directly influencing any aspect.

Thus, anti-tampering design encompasses all the approaches to combat the diverse forms of tampering attacks.

Anti-tampering Techniques

1. Tamper resistance depends on limiting the physical access to a chip, making tampering a more challenging and time-consuming task. Standard tamper-resistance methods include physical packaging technologies, metal shielding, inherent physical security, post-masking, and controlling access to prevent unauthorized entry. Furthermore, the security function "random dummy read" can prevent repetitive reads at the same address, incorporating address and data bus scrambling techniques to add complexity and bewilder potential hackers.

2. Tamper detection involves the capability to identify and perceive tampering attempts. In any secure design, it is crucial to incorporate anti-tamper circuitry that conducts health checks and detects irregularities in power supply as well as instances of fault injection.

3. Tamper response pertains to the actions taken upon detecting tampering. Examples of potential tamper responses encompass triggering alarm signals (interrupts), deactivating or disabling a device, and eliminating or erasing critical memory space.

4. Tamper evidence involves generating visible indicators that persist when tampering takes place, enabling authorized personnel to determine whether tampering has occurred or not.

3.5 Conclusion

In summary, this chapter extensively covers the realm of hardware Trojan insertion and detection in integrated circuits. It begins with an introduction followed by a detailed exploration of various insertion approaches, including the insertion phase, abstraction levels, external mounting of hardware Trojans, and strategies employed in their insertion. The chapter then delves into detection techniques, categorizing them into invasive and non-invasive methods, and discussing the advantages and disadvantages associated with each. Additionally, it explores software and hardware materials used in hardware Trojan mitigation, encompassing simulation and testing, deductive verification, formal verification, Physically Unclonable Functions (PUFs), hardware-based encryption, Trusted Platform Module (TPM), dual and redundant hardware designs, and anti-tamper packaging. This comprehensive examination provides a solid foundation for understanding the principles, approaches, and tools in the complex landscape of hardware Trojan insertion and detection.

Chapter 4

Hardware Trojan Externally Activated Detection Technique

4.1 Introduction

The emergence of hardware Trojans poses a formidable threat to the integrity and security of electronic systems. These insidious malicious implants, when activated externally via radio waves, can clandestinely manipulate the functionality of integrated circuits, giving rise to potential breaches, espionage, and sabotage. Recognizing the critical need to fortify our defenses against such sophisticated attacks, researchers and security experts have delved into the realm of radio wave analysis as a potent tool for detecting and mitigating externally triggered hardware Trojans.

Unlike traditional software-based threats, hardware Trojans operate at the physical level of electronic components, making their detection a complex and challenging task. The utilization of radio waves as a means of external activation adds an additional layer of intricacy, as conventional security measures may fall short in identifying these covert manipulations. This necessitates the exploration and development of specialized techniques capable of dissecting radio wave emissions to unveil the presence of hardware Trojans and thwart their potential malicious activities.

This exploration delves into the cutting-edge methodologies employed by researchers to detect hardware Trojans activated through radio waves. From leveraging electromagnetic side-channel analysis to advanced signal processing techniques, the arsenal of tools available for uncovering these covert threats is expanding rapidly. By comprehensively understanding and implementing these techniques, researchers aim to bolster the resilience of critical systems against the pervasive risk posed by externally activated hardware Trojans.

This study navigates through the intricacies of hardware Trojan detection, shedding light on the significance of radio wave analysis in fortifying our defense mechanisms. As we embark on this journey into the depths of hardware security, it becomes apparent that staying ahead in the race of hardware security requires constant innovation and a keen understanding of the multifaceted challenges presented by the fusion of hardware vulnerabilities and radio wave exploitation.

4.2 Superheterodyne Radio Receiver: What it is and How it Works

Invented in 1918 to overcome the issues of lack of selectivity, superhet designs have been at the centre of radio communications technology for nearly 100 years, and only recently are other topologies taking over. Despite this the superheterodyne radio is still used in many applications and the RF design techniques used are still applicable in many radio communications applications.

The superhet radio works by using a variable frequency local oscillator and feeding the incoming signals and the local oscillator into an RF mixer to convert the signals to a fixed frequency intermediate frequency. By having a fixed frequency amplification and filtering stages it enables the radio to remove unwanted signals more effectively than other forms like the TRF (Tuned Radio Frequency) sets or even regenerative radios that were used particularly in the early days of radio.

4.2.1 Superheterodyne radio applications and usage

Superheterodyne (superhet) radios are a common type of radio architecture widely used in various applications. The most important use cases are:

- **Aviation and Maritime Communication:** Superheterodyne radios play a crucial role in aviation and maritime communication systems. They are utilized in aircraft radios, ship communication systems, and control towers, providing a robust and effective means of communication in these critical settings.
- **Military and Defense Applications:** Superhet radios are employed in military and defense applications due to their reliability and selectivity. They are utilized for communication between military units, intelligence gathering, and other defense-related activities.

4.2.2 Superheterodyne receiver How it works

As mentioned above in paragraph two of section 4.2, describes Superhet way of working. To achieve the overall superhet radio topology, there are several techniques and technologies that are involved within the receiver:

- **Overall theory:** The fundamental idea and radio frequency (RF) architecture of the superheterodyne radio center around a mixing process, allowing the translation of signals from one frequency to another. The RF input denotes the input frequency, the local oscillator represents the locally generated oscillator signal, and the intermediate frequency, positioned between RF and audio frequencies, characterizes the output frequency. Within a mixer the instantaneous amplitude of the two input signals (f1 and f2) is multiplied and this results in signals at the output of frequencies of (f1 + f2) and (f1 - f2). This enables an incoming frequency to be translated down to a fixed frequency where it can be effectively filtered. Varying the frequency of the local oscillator enables the receiver to be tuned to different frequencies.

- **Image Response:** A critical concern in the superheterodyne radio system is the issue of image response. The intermediate frequency stages may receive signals on two distinct frequencies, and RF tuning is employed to eliminate one while accepting the other. The presence of image signals can lead to undesirable interference, potentially obscuring desired signals if both coincide within the intermediate frequency section. In economical radios, harmonics of the local oscillator may follow different frequencies, resulting in diverse heterodyne effects as the receiver is tuned. Effective rejection of image interference is essential for achieving optimal performance in a radio receiver.

- **Block diagram:** The comprehensive block diagram of the superheterodyne receiver illustrates the fundamental blocks employed in the receiver. This basic representation provides insight into the overall functioning of the radio. In more advanced radio systems, extra blocks are incorporated into the fundamental block diagram. These additional blocks might include extra demodulators or additional circuit components within the local oscillator, depending on the desired level of detail. Additionally, certain superheterodyne radios may feature two or more conversions to enhance performance across various aspects.

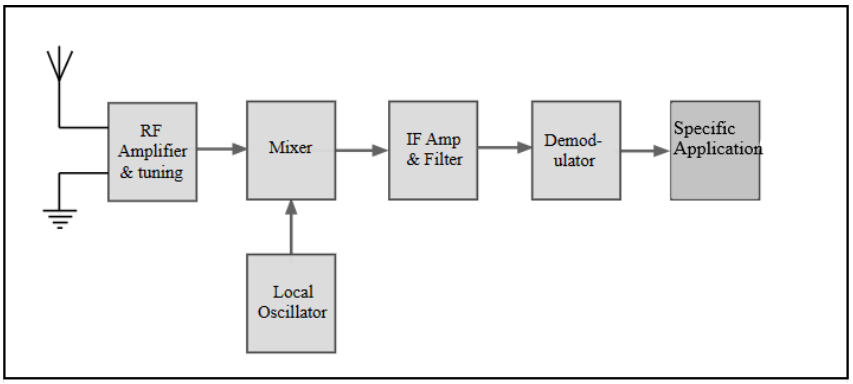


Figure 4.1: Block diagram of a basic superheterodyne receiver. From "https://www.electronics-notes.com/", "Superheterodyne Receiver Theory & Principles", Visited on November 2024.

The signal received by the antenna enters the receiver and goes through a mixer. The mixer has three ports: signal, local oscillator, and intermediate frequency (IF). The signal is appropriately directed to the signal port, which is specifically designed to accommodate lower-level signals compared to the local oscillator (LO) port. A different signal generated locally, commonly referred to as the local oscillator (LO), is introduced into the mixer's second port, where it combines with the original signal through a mixing process. The mixer operation involves multiplying the instantaneous amplitudes of the two signals. The mixer's non-linear behavior results in the creation of signals at frequencies equivalent to the sum and difference of the initial signals.

Numerous mixers are referred to as balanced, indicating that the two incoming signals are either absent or significantly minimized in the output. The mixer's output then proceeds to the intermediate frequency (IF) stages, where the signal undergoes amplification

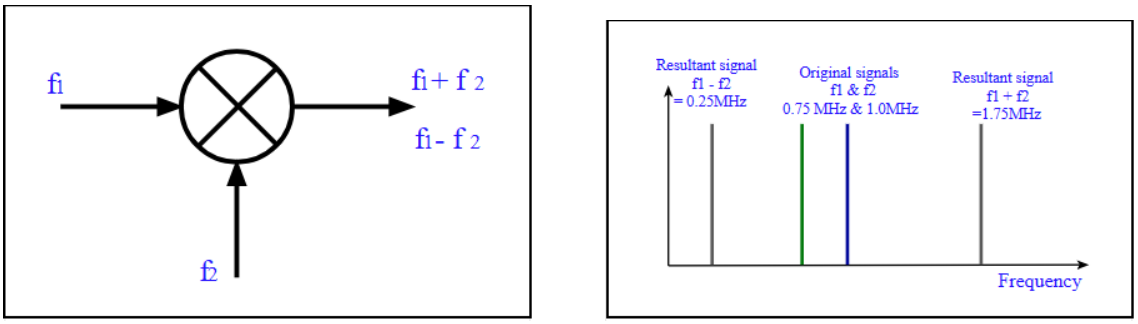


Figure 4.2: Mixing two RF signals signals at the sum and difference frequencies are produced. From "https://www.electronics-notes.com/", "Superheterodyne Receiver Theory & Principles", Visited on November 2024.

and filtration. Signals converted to frequencies within the IF filter's passband are allowed to pass through and are additionally amplified by the amplifier stages. Signals outside the filter's passband are rejected.

Adjusting the receiver's tuning involves a straightforward process of altering the local oscillator frequency. This modification affects the incoming signal frequency, allowing signals to be converted down and pass through the filter. Examining a concrete example can elucidate the process. Consider two signals, one at 1.0 MHz and another at 1.1 MHz, to observe how this functions in practice.

Assuming the intermediate frequency (IF) filter is centered at 0.25 MHz and the local oscillator is configured at 0.75 MHz, the mixer generates two signals from the 1.0 MHz signal: one at 0.25 MHz and another at 1.75 MHz. Naturally, the 1.75 MHz signal is rejected, while the one at 0.25 MHz successfully traverses the IF stages. Regarding the 1.1 MHz signal, it produces frequencies at 0.35 MHz and 1.85 MHz. Since both fall outside the IF filter's bandwidth, only the signal from the 1.0 MHz source passes through the IF stages. If the local oscillator frequency is increased by 0.1 MHz to 0.85 MHz, the 1.1 MHz signal will generate frequencies at 0.25 MHz and 1.95 MHz. Consequently, the 1.1 MHz signal, resulting in the 0.25 MHz frequency after mixing, will successfully pass through the filter. Meanwhile, the 1.0 MHz signal will produce frequencies of 0.15 MHz and 1.85 MHz, both of which will be rejected. In this manner, the receiver functions as a variable frequency filter, and the tuning is achieved by adjusting the local oscillator frequency within the superheterodyne receiver.

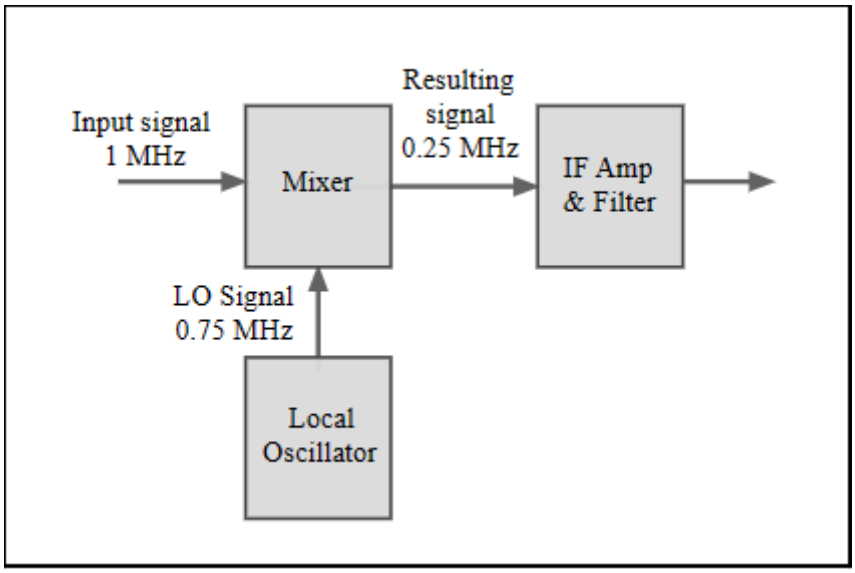


Figure 4.3: The basic principle of the superheterodyne radio using a mixer to convert the frequency of the incoming signal. From "https://www.electronics-notes.com/", "Superheterodyne Receiver Theory & Principles", Visited on November 2024.

4.2.3 Drawbacks Of Superheterodyne Receiver

The superheterodyne receiver design offers several advantages, but it is not without drawbacks. Here are some common drawbacks associated with superheterodyne receivers:

- **Image Frequency Interference:** Superheterodyne receivers can be susceptible to image frequency interference. This occurs when undesired signals at a frequency equal to the sum or difference of the local oscillator frequency and the desired signal frequency interfere with the reception.
- **Image Rejection Requirements:** Achieving high image rejection often requires careful design and precision in component values. This can increase the overall complexity of the receiver and may demand more precise manufacturing processes.
- **Sensitivity to Component Variations:** The performance of a superheterodyne receiver is sensitive to variations in component values, such as those caused by temperature changes or manufacturing tolerances. This sensitivity can affect the overall stability and performance of the receiver.

Local Oscillator Leakage

A major downside to surperhet architecture is that when down-converting an RF band to IF a Local Oscillator (LO) is used. This local oscillator is tuned to a frequency such that when mixed with the incoming RF signal, the desired RF band is down-converted to the fixed IF band. In all of these receivers, there is inevitable reverse leakage, and therefore some of the **local oscillator power actually couples back through the input port and radiates out of the antenna** (Figure 4.4). This Leakage signal constitutes the key factor of our approach (Section 4.4) used to detect such receivers that are used as triggers for implanted hardware Trojan in ICs.

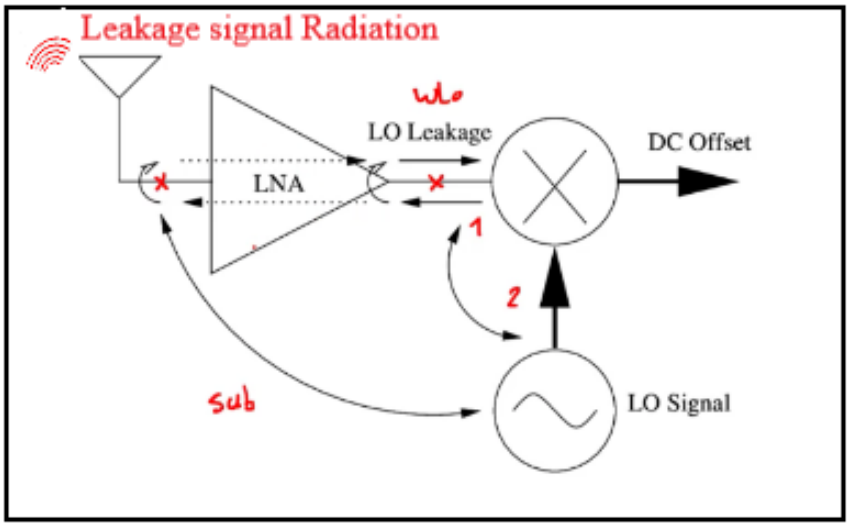


Figure 4.4: Local Oscillator Leakage Signal radiates out of the antenna.

4.3 Detecting Superheterodyne Receivers

Progress in electronics and RF design has led to the development of smaller, more affordable, and widely accessible radio receivers. While these innovative devices open the door to numerous applications, there is also a potential misuse for initiating dormant Hardware Trojan implanted in integrated circuit (IC). A strategy to indirectly identify potential externally activated HT involves locating the radio receiver, thereby reducing this threat. Radio receivers utilize various high-frequency signals that inadvertently emit into the environment, causing unintended electromagnetic emissions. Detecting these emissions offers a means to identify the presence of radio receivers.

The fact that superheterodyne receivers emit robust sinusoidal signals from their local oscillator (LO) is widely acknowledged. Previous studies have proven that these emissions can be identified through the use of periodograms [Wild and Ramchandran, 2005]. In this method, a signal that may include unintended emissions is sampled, and its periodogram is calculated. Each bin is then evaluated against a threshold, and the detector registers a positive outcome if this threshold is surpassed.

Unintentional emissions can provide insights into the internal state of an electronic device as shown in [Sekiguchi and Seto, 2008], and radio receivers are no exception. Contrary to other device types, radio receivers exhibit high sensitivity to faint stimulation signals. Sending a recognized stimulation signal to a receiver can alter its unintended emissions predictably. Detecting radio receivers is achievable by analyzing and comparing their unintended emissions with the applied stimulation signal. This method is known as **stimulated emissions** detection. It bears similarities to harmonic detection methods that expose the electronic device to a potent stimulation and search for "reflected" harmonics generated by interactions with nonlinear electronic components. Because the suggested technique alters the intended signals within the device, it can employ a lower-power stimulation, operate over a greater distance, and experience fewer false alarms compared to harmonic detection.

4.4 Methodology for Designing Receivers Detector

To showcase our research [Guechi and Redjimi, 2021], we implemented a Hardware Trojan attack scenario. We simulated an AES-128 encryption circuit using Modelsim software and uploaded it to an FPGA Nexys 2 Spartan3E-500. The Trojan was integrated as an external component, specifically a 315MHz radio receiver, designed to trigger a malicious function within the encryption circuit, ultimately disrupting the encryption process entirely. Under normal circumstances, the encrypted result is sent from the FPGA to an external microcontroller equipped with a radio receiver to capture the encrypted output. However, if the Trojan activates the malicious function, the expected encrypted output will instead be received as plain text.

4.4.1 Trojan implementation

In the Trojan scenario, radio receiver module is employed to serve as a trigger, initiating a harmful function that halts the encryption process entirely. This receiver is activated by a radio transmitter under the control of the NodeMCU3[1] microcontroller. Once the signal is received by the radio receiver, it activates a section of the circuit on the FPGA, which consequently interrupts the encryption process, allowing the passage of plain text instead of encrypted data (Figure 4.5).

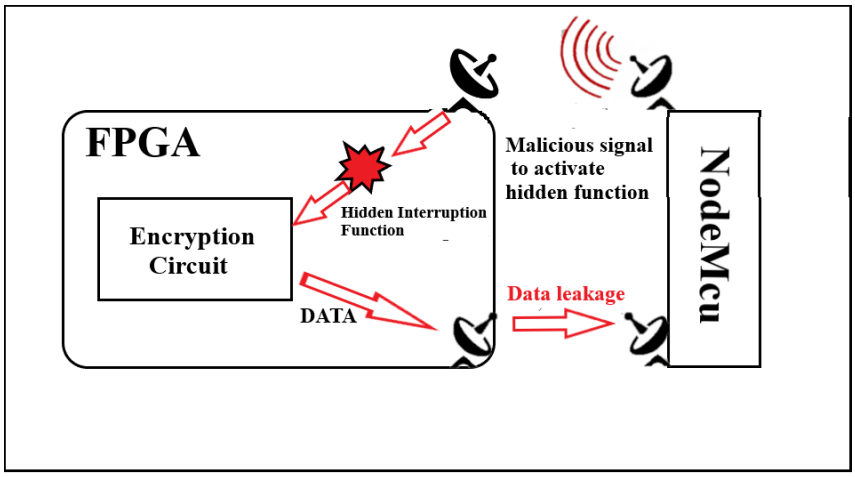


Figure 4.5: Trojan Implementation Scenario.

Interfacing Spartan 3E-500 Nexys 2 With Radio Receiver/Transmitter

Interfacing an FPGA with a 315 MHz radio receiver module involves several steps, including understanding the communication protocol of the radio receiver, configuring the FPGA to process the received signals, and implementing the necessary logic in the FPGA. Here's a general outline of the process:

[1]The NodeMCU (Node MicroController Unit) is an open-source software and hardware development environment built around an inexpensive System-on-a-Chip (SoC) called the ESP8266. It contains the crucial elements of a computer: CPU, RAM, networking (WiFi), and even a modern operating system and SDK.(https://www.make-it.ca/nodemcu-details-specifications/)

1. **Understand the Radio Receiver Module:** Before interfacing the FPGA with the radio receiver module, it's essential to understand the module's output characteristics and communication protocol (Table 4.1). Most 315 MHz radio receiver modules output a digital signal that represents the demodulated data received over the air.

   Key Characteristics to Determine:

   - **Power Supply Requirements:** Voltage and current requirements.
   - **Data Output Format:** Digital signal format (e.g., pulse width modulation, on-off keying).
   - **Signal Timing:** Timing characteristics of the output data (e.g., baud rate, pulse width).

2. **Connect the Radio Receiver Module to the FPGA**

   - **Power Connections:** Connect the Vcc pin of the radio receiver module to the 3.3V pin on the Spartan 3E-500 Nexys 2 board (Pin J4-1). And Connect the GND pin of the radio receiver module to the ground pin on the Spartan 3E-500 Nexys 2 board (Pin J4-2).
   - **Data Connection:** Connect the Data Output pin of the radio receiver module to one of the general-purpose I/O (GPIO) pins on the Spartan 3E-500 Nexys 2 board (e.g., Pin J1-1).

3. **Design the FPGA Interface**

   - **Clock Management:** Ensure the FPGA clock frequency is suitable for processing the incoming data.
   - **Input Conditioning:** Synchronize and filter the incoming signal.
   - **Data Decoding:** Implement the logic to decode the incoming data based on the modulation scheme used by the radio receiver.

4. **Implement the FPGA Code**: Verilog code provided in detail (Appendix A.1). This code is designed to interface with a 315 MHz radio receiver module and decode the incoming On-Off Keying (OOK) signal[2].

[2]On–off keying (OOK) denotes the simplest form of amplitude-shift keying (ASK) modulation that represents digital data as the presence or absence of a carrier wave. In its simplest form, the presence of a carrier for a specific duration represents a binary one, while its absence for the same duration represents a binary zero.(https://dbpedia.org/page/On-off_keying)

Parameter	Value/Range
working current	50mA(9vdc)
Principal of work	Superhet (VCO, PLL)
Modulation	OOK/ASK
Working band	315MHz; 433.92 MHZ(customize service available)
Operation Temperature	-20°C to 75°C
Bandwidth	200KHZ
Sensitivity	-110dBm (50Ω)
Modulation rate	<10Kbps
Decoding form	PT2272
Antenna length	18cm(Tx) 24cm(Rx)
Emission Distance	2km(Open area without interference)
Working Output Modes	Unlocked, interlocked, self-locked

Table 4.1: RF Receiver/Transmitter Link Kit Datasheet.

Interfacing the Spartan 3E-500 Nexys 2 FPGA board with a 315 MHz radio transmitter involves same steps as interfacing with the receiver except that the Data Input pin of the radio transmitter module is going to be driven by the PIO pin on the Spartan 3E-500 Nexys 2 board (e.g., Pin J1-1). Verilog code provided in detail (Appendix A.2).

Interfacing The Microcontroller NodeMCU with Radio Receiver/Transmitter

Interfacing a NodeMCU microcontroller with a 433 MHz(we have chosen this frequency to avoid the interference with 315MHz RX/TX Modules) radio receiver and transmitter involves connecting the appropriate GPIO pins of the NodeMCU to the radio modules and writing the firmware to handle the transmission and reception of data.

Connection Overview

1. **Transmitter Connections:** connect the Vcc pin of the transmitter module to the 3.3V pin on the NodeMCU. And connect the GND pin to the GND pin on the NodeMCU. Finally Connect the Data pin of the transmitter module to a GPIO pin on the NodeMCU (e.g., D1).

2. **Receiver Connections:** Connect the Vcc pin of the receiver module to the 3.3V pin on the NodeMCU. And connect the GND pin to the GND pin on the NodeMCU. Finally connect the Data pin of the receiver module to a GPIO pin on the NodeMCU (e.g., D2).

Steps to Implement

1. **Install MicroPython:** Ensure that MicroPython is installed on your NodeMCU. You can use tools like esptool.py to flash MicroPython firmware onto your NodeMCU.

2. **Upload the Code:** Use an IDE like **Thonny** or a tool like **ampy** to upload the transmitter and receiver scripts to your NodeMCU.

In order to stop the encryption process on the FPGA, we use the transmitter on the NodeMCU microcontroller to send "stop_encryption" String to the FPGA (Appendix A.4). Once this Sting is received we send the content of register on the FPGA that contains the Raw Data to the NodeMCU microcontroller using the code in (Appendix A.2)

4.4.2 Detection Technique

Radio receivers utilize the superheterodyne receiver design. In this architecture, the RF signal is shifted to a fixed lower intermediate frequency (f_{IF}). To achieve this conversion, a local oscillator (LO) is employed. The LO operates at a frequency such that when combined with the incoming RF signal, the desired RF band is converted down to the fixed IF band. In all of these receivers, there is inevitable reverse leakage, and therefore some of the local oscillator power actually couples back through the input port and radiates out of the antenna [Wild and Ramchandran, 2005].

Our research methodology involves quietly listening for the local oscillator frequency f_{LO} signal without emitting any signal itself, thus functioning as a passive detector. The periodogram detector used search for sinusoidal f_{LO} emissions. According to [Stagner, 2013] A perfect periodogram detector calculates:

$$S_x(f) = \frac{1}{M} \sum_{n=0}^{M-1} |x(n) \exp(-j2\pi nf)| \tag{4.1}$$

1. $S_x(f)$: This represents the power spectral density (PSD) of the signal $x(n)$ at frequency f.

2. $\frac{1}{M}$: This is a normalization factor where M is the number of samples in the signal. It ensures that the computed PSD is averaged over the length of the signal.

3. $\sum_{n=0}^{M-1}$: This is a summation over n, where n ranges from 0 to $M-1$. Essentially, it means we are summing over all the samples of the signal $x(n)$.

4. $|\cdot|$: This denotes the magnitude of the complex value inside. In this context, it computes the magnitude of the complex number resulting from the product of $x(n)$ and the exponential term.

5. $x(n)$: This is the value of the discrete-time signal at the n-th sample.

6. $\exp(-j2\pi nf)$: This is a complex exponential term, where:

   - j is the imaginary unit ($j^2 = -1$).
   - $2\pi nf$ represents the angular frequency component, with f being the frequency and n being the sample index. The negative sign in the exponent indicates a shift in the frequency domain.

Putting it all together:

- The term $x(n)\exp(-j2\pi nf)$ is essentially a Fourier transform operation. It shifts the signal $x(n)$ into the frequency domain at frequency f.
- The magnitude $|x(n)\exp(-j2\pi nf)|$ computes the strength of the signal at that specific frequency f.
- The summation $\sum_{n=0}^{M-1}$ adds up these magnitudes over all n samples.
- The factor $\frac{1}{M}$ averages the summed magnitudes over the length of the signal.

It evaluates each bin $Sx(f)$ against a predetermined threshold, and the detector considers the signal satisfactory if one or more bins surpass the threshold. The periodogram is approximated using the Fast Fourier Transform (FFT). The detector seeks out peaks within the periodogram. As local oscillator signals manifest as peaks, each point on the periodogram undergoes comparison with a predetermined threshold. If at least one point surpasses this threshold, a detection event is triggered.

To demonstrate how to figure out the local oscillator (LO) signal of a radio receiver using the given formula (4.2) for power spectral density (PSD), let's go through the process step-by-step. The goal is to identify the frequency of the LO signal within the received signal $x(n)$.

Step-by-Step Process

1. Acquire the Signal: Obtain a discrete-time sample of the received signal $x(n)$ from the radio receiver. This signal will likely contain multiple frequency components, including the LO signal.
2. Understand the Received Signal: The received signal $x(n)$ can be expressed as:

$$x(n) = s(n) + \cos(2\pi f_{LO} n + \phi) \tag{4.2}$$

   where $s(n)$ is the actual signal of interest, f_{LO} is the frequency of the local oscillator, and ϕ is the phase.
3. Compute the Power Spectral Density: Use the given formula (4.2) to compute the PSD of the received signal.
4. Steps to Implement the PSD Calculation:
   - Choose the Length M: Determine the number of samples M to use for the PSD calculation.
   - Frequency Range: Decide the range of frequencies over which you want to compute the PSD. Typically, this range will be from 0 to the Nyquist frequency (half the sampling rate).
   - Frequency Range: Decide the range of frequencies over which you want to compute the PSD. Typically, this range will be from 0 to the Nyquist frequency (half the sampling rate).

- Loop Over Frequencies: For each frequency f in the chosen range:
  (a) Compute the complex exponential term $\exp(-j2\pi nf)$.
  (b) Multiply each sample $x(n)$ by the corresponding exponential term.
  (c) Take the magnitude of the product.
  (d) Sum the magnitudes over all samples n from 0 to $M-1$.
  (e) Normalize by $\frac{1}{M}$.
- Identify the LO Signal Frequency: The LO signal will manifest as a prominent peak in the PSD at its frequency f_{LO}. The PSD value will be significantly higher at this frequency compared to others due to the presence of the cosine term in the signal.

We give a conceptual Python implementation to practically illustrate our detection technique (see Appendix A.5):

4.4.3 Results and Discussion

The primary circuit responsible for the encryption operation has been implemented and configured on the FPGA. Prior to this, a ModelSim simulation was conducted multiple times using various input values to verify the circuit's normal behavior. To facilitate radio communication between the FPGA and the microcontroller, a radio receiver and transmitter are utilized. The output from the encryption circuit has been successfully received (see appendix A.3).

The encryption process was successfully disrupted using the microcontroller's radio transmitter. The signal was received by the FPGA via the radio receiver, which triggered a malicious function that halted the entire encryption process. Regarding the detection of the leakage power from the super heterodyne receiver, it is important to determine the power transmitted from the super heterodyne receiver to our radar (Figure 4.6).

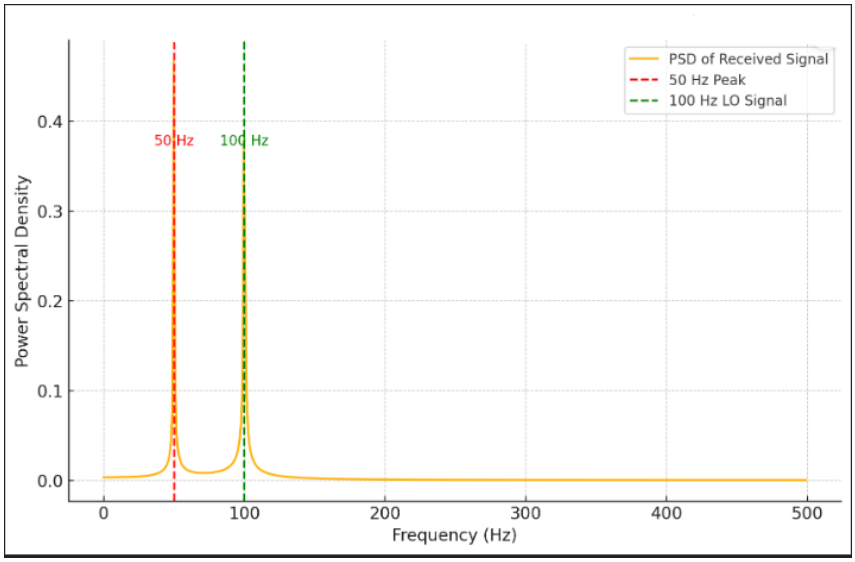


Figure 4.6: Power Spectral Density Of Received Signal with Peaks Highlighted.

As we see in the previous (Figure 4.6), we notice two highlighted peaks: the first one at 50 Hz, which corresponds to the signal of interest $s(n)$. This peak represents the strength of the sinusoidal component at 50 Hz within the signal. The signal of interest was a sine wave at 50 Hz, hence the prominent peak at this frequency. The second peak is at 100 Hz, which corresponds to the local oscillator (LO) signal. This peak represents the strength of the cosine component at 100 Hz. This is the frequency of the LO signal that was added to the signal of interest.

However, our method used to detect the local oscillator (LO) signal by examining the power spectral density (PSD) has some drawbacks:

1. **Noise Sensitivity:** The presence of noise in the signal can obscure the peaks corresponding to the signal of interest and the LO signal. This can make it difficult to accurately identify the LO frequency.

2. **Resolution Limitation:** The resolution of the PSD is limited by the number of samples (M). A higher number of samples provides better frequency resolution but requires more computation and memory.

3. **Computational Complexity:** Computing the PSD using the periodogram approach involves a Fourier transform, which can be computationally intensive for large signals.

4. **Spectral Leakage:** Spectral leakage occurs when the signal contains components that are not integer multiples of the frequency resolution, causing spreading of power across multiple frequency bins.

Demonstrations highlight the practical challenges and limitations of using the PSD method to detect the LO signal in a radio receiver (see Appendix A.6).

4.5 Comparaison to existing approaches

The table below shows a detailed comparison to the closest existing approaches to our approach for Hardware Trojan detection regarding multiple aspects:

Table 4.2: Comparison between the existing approaches for Hardware Trojan Detection

Aspect	**Gayatri et al. (2020)**	**Guechi & Redjimi (2021)**	**Elkanishy et al. (2019)**
Detection domain	Side-channel / parametric features (power, delay, switching activity)	Test-time and functional statistical data	Side-channel (power / EM)
Golden reference requirement	Required (directly or indirectly)	Not required	Typically required
Training data	Labeled Trojan-free and Trojan-infected samples	No labeled data required	Labeled data needed
Sensitivity to Trojans	High for parametric and power-active Trojans	High for Trojans affecting feature correlations	High sensitivity to leakage changes
Robustness to stealth Trojans	Limited when Trojan minimizes parametric impact	Improved robustness if correlations are disturbed	Lower if Trojan is low-power or dormant
Scalability	Moderate due to training and feature extraction overhead	High; scalable to large SoCs using test data	Moderate due to training
Localization capability	Limited (feature-level inference)	Detection only; no physical localization	Moderate (via leakage features)

From the previous table we notice that:

Elkanishy et al. (2019) employ supervised machine-learning techniques on side-channel leakage to identify Trojan-infected circuits, with high sensitivity to detect Trojan that has power-active feature but at the cost of golden-reference dependence and significant measurement overhead. In contrast, we propose a golden-chip-free unsupervised detection approach that leverages correlation-based feature selection across heterogeneous SoC components, enabling scalable and cost-effective detection of stealthy hardware Trojans without requiring labeled training data.

Gayatri et al. (2020) rely on supervised machine-learning techniques applied to side-channel and parametric features, ensuring accuracy in the presence of two principle requirements labeled training data and golden circuit free from any sort of intrusion. In contrast, our approach which is a golden-chip-free unsupervised detection based on

correlation-based feature selection, enabling scalable and cost-effective detection of stealthy hardware Trojans in heterogeneous systems-on-chip.

4.6 Conclusion

In this chapter, we have had an in-depth discussion about externally activated hardware Trojan. Our conversation covered specified mechanism by which these Trojans can be triggered from outside sources, this mechanism relies on the local oscillator radiation of radio receivers to detect its presence, the potential risks they pose to system security, and the latest detection and prevention strategies. This detailed exploration provided a comprehensive understanding of the complexities and challenges associated with combating these types of hardware threats.

We then addressed the detection of superheterodyne receivers, presenting a methodology for designing an effective detection system. Our contributions include the development and implementation of a main circuit and the introduction of a Trojan to rigorously test the detection capabilities. Through this, we demonstrated a detection technique that identifies the presence of superheterodyne receivers.

Our results demonstrate that there are specific constraints that need to be respected. These constraints are crucial for ensuring the accuracy and reliability of our findings. Ignoring them could lead to imperfection in the detection approach.

Chapter 5

Hardware Security Module To Counter Hardware Trojans

5.1 Introduction

Hardware Security Modules (HSMs) play a crucial role in safeguarding sensitive data and critical systems against a pervasive threat: hardware trojans. As our digital infrastructure becomes increasingly interconnected and reliant on complex hardware components, the risk of malicious actors implanting undetectable backdoors or manipulations within these components grows. Hardware trojans pose a significant challenge because they can bypass traditional software-based security measures, potentially compromising the integrity and confidentiality of systems.

In this landscape, HSMs emerge as a formidable defense mechanism. Essentially, HSMs are specialized hardware devices designed to securely manage, process, and store cryptographic keys and sensitive data. They provide a trusted execution environment that is isolated from the main system, making them resistant to tampering and attacks, including those orchestrated by hardware trojans.

By offloading cryptographic operations to dedicated hardware, HSMs mitigate the risk of key exposure and unauthorized access. Furthermore, HSMs often incorporate robust security features such as tamper detection mechanisms, secure boot processes, and physical shielding to prevent unauthorized manipulation or extraction of sensitive information.

In the context of countering hardware trojans, HSMs serve as a critical line of defense by ensuring the integrity and authenticity of cryptographic operations, even in the presence of compromised or malicious hardware components. Their role extends beyond merely securing data; they also provide a foundation of trust that underpins the entire security infrastructure, bolstering confidence in the integrity of digital transactions, communications, and systems. As organizations confront the evolving threat landscape, leveraging HSMs as a cornerstone of their security architecture becomes increasingly imperative to thwart the insidious threat posed by hardware trojans.

5.2 Hardware Security Modules (HSMs)

Hardware Security Modules (HSMs), are used in various applications to ensure the safeness of critical systems such as banking, military communications, networking. In this section ....

5.2.1 What is a Hardware Security Module?

Hardware Security Modules (HSMs) are specialized hardware devices designed to manage, generate, and safeguard cryptographic keys and perform cryptographic operations securely. These modules provide a trusted execution environment that is isolated from the main system, ensuring the confidentiality, integrity, and availability of sensitive data and cryptographic operations. HSMs are commonly used in various industries, including finance, healthcare, government, and military communications, to protect critical assets and meet compliance requirements.

5.2.2 How Do Hardware Security Modules Work?

Encryption, which involves making sensitive information unreadable to anyone without proper authorization, serves as the fundamental function of an HSM. Additionally, secure decryption and message authentication are integral aspects of HSM operations.

Randomly generated values play a crucial role in encryption by serving as the foundation for creating encryption keys. With these keys, decrypting sensitive information becomes possible with just one step. Therefore, it's imperative to store encryption keys in a secure environment.

Hardware security modules (HSMs) produce and retain encryption keys utilized across different devices. They possess unique hardware components to generate entropy and produce top-notch random keys. In larger organizations, multiple HSMs might be utilized simultaneously rather than relying on a single unit. Regardless of the number of HSMs in operation, implementing an efficient, centralized key management system that aligns with external regulations and robust internal security protocols enhances both security and compliance.

5.2.3 Hardware Security Module Architecture

HSMs often incorporate features that make them tamper-proof or tamper-resistant. For instance, they might exhibit visible indications of logging and alerting, or they might cease functioning if tampering is detected. Certain HSMs may even erase keys upon detecting any tampering activity. These modules are typically encased in packaging that is either tamper-resistant, tamper-evident, or tamper-responsive. They contain one or multiple cryptoprocessor chips or a module containing a combination of chips to deter bus probing and tampering.

HSMs are typically capable of clustering to achieve high availability, especially considering their common inclusion in mission-critical infrastructure like online banking applications or military communication system. Certain hardware security modules sup-

port business continuity and comply with the high-availability standards of data center settings. This can include features such as field-replaceable components or dual power supplies, ensuring continuous availability even in the face of disasters.

Certain HSMs possess the ability to internally run custom modules developed in languages like C, Java, .NET, or other programming languages. This capability can be beneficial for organizations requiring execution of business logic or specialized algorithms within a trusted environment. Advanced hardware security modules of the next generation are often equipped to handle tasks like running and loading Commercial Off-The-Shelf (COTS) software, operating systems, and other complex operations without necessitating extensive reprogramming or customization.

5.2.4 Hardware Security Module Applications

Any software application that utilizes digital keys has the option to utilize a hardware security module (HSM). Typically, the decision to employ an HSM is based on the potential serious consequences that would arise from the compromise of these keys. In essence, digital keys must hold significant value to warrant their generation and management within a hardware security module.

The key functions of an HSM are as follows:

- **Key Generation and Storage:** HSMs generate cryptographic keys using secure random number generators (RNGs) and store them within their secure hardware enclave. These keys are safeguarded against unauthorized access and tampering, ensuring their confidentiality and integrity.
- **Cryptographic Operations:** HSMs perform a wide range of cryptographic operations, including encryption, decryption, digital signatures, and key management functions. These operations are executed within the secure environment of the HSM, safeguarding sensitive data and cryptographic processes from external threats.
- **Secure Access Control:** HSMs enforce strict access controls to prevent unauthorized users or applications from accessing sensitive cryptographic keys and operations. Access to the HSM is typically restricted to authorized entities through authentication mechanisms such as passwords, cryptographic tokens, or biometric authentication.
- **Tamper Detection and Response:** HSMs incorporate physical security features such as tamper-evident seals, sensors, and anti-tampering mechanisms to detect and respond to unauthorized access or tampering attempts. In the event of a potential breach, HSMs can initiate protective measures such as zeroization (secure deletion) of sensitive data to prevent its compromise.
- **Secure Communication:** Many HSMs support secure communication protocols to establish trusted channels between hardware components, applications, and external systems. By encrypting data in transit and verifying the authenticity of communication endpoints, HSMs mitigate the risk of eavesdropping, tampering, and impersonation attacks.

Overall, HSMs provide a trusted execution environment for cryptographic operations, ensuring the confidentiality, integrity, and availability of sensitive data and cryptographic processes. By leveraging HSMs, organizations can enhance their security posture, protect against evolving threats, and maintain compliance with regulatory requirements.

5.2.5 Comparing Hardware Security Modules (HSMs) with Trusted Execution Environments (TEEs) and Trusted Platform Modules (TPMs)

A trusted execution environment (TEE) is a secure enclave established within the main processor of a computer system. Its purpose is to guarantee the integrity and confidentiality of data and code contained within the TEE.

A trusted platform module (TPM) is a specialized chip integrated and soldered into the motherboard, making it challenging to access its confidential keys and evident upon any attempt. This physical installation serves to establish a hardware-based foundation of trust within the computing system. While TPMs typically do not increase computational capabilities, they may provide basic functionalities like generating random keys or encrypting small data sets.

In contrast, a hardware security module (HSM) stores encryption keys separately from the operating system. While there may be some similarities among trusted execution environments (TEEs), trusted platform modules (TPMs), and HSMs, they are distinct and offer differing advantages. Similar to TPMs, HSMs reveal evidence of physical tampering, but they typically offer greater levels of security compared to both TPMs and TEEs.

Certain individuals contend that hardware security modules (HSMs) no longer require reliance on physical tamper protection and proprietary hardware designs. Instead, they propose leveraging the security features of trusted execution environments (TEEs) to establish a "soft HSM" or virtual hardware security module. For instance, Google's Cloud HSM is marketed as a cloud-based hardware security module, representing a fully virtualized service rendition of the traditional HSM.

In summary:

- Trusted Execution Environments (TEEs) provide a broad, integrated processing environment as a component of the chipset.
- Trusted Platform Modules (TPMs) offer modest processing capabilities, along with the ability to measure the boot sequence and other elements, serving as a physical foundation for trust. They are an affordable built-in component.
- Hardware Security Modules (HSMs) represent the pinnacle of security for processing sensitive data, managing or storing secret keys, and executing cryptographic operations. While they are typically pricier and external devices, advancements in cloud technology can make them more cost-effective and scalable.

5.2.6 Advantages and Characteristics of Hardware Security Modules

The primary advantages of hardware security modules include safeguarding against physical access, ensuring the secure handling of key material, reliable key generation, and providing a secure execution environment.

Integrated Circuits (ICs) cannot be fully shielded from external threats. However, HSMs employ various protective measures to deter both external attacks and physical tampering. These measures often include voltage and temperature sensors, resin-embedded chips, and drill protection foil.

For instance, in the event of an attacker trying to drill into an HSM device, whether by forcibly breaking the casing or utilizing substances like acid or ice to corrode its layers, sensors promptly detect the intrusion, activate an alarm, and execute any predefined countermeasures specified in the configuration, such as key deletion.

Keys are valuable only when they are both random and securely safeguarded; otherwise, they become vulnerable to attackers' guesses. In traditional IT systems, generating secure keys is challenging due to reliance on conventional commands that operate based on if-then conditions. Regrettably, skilled attackers can exploit knowledge of the "if" or input data for a given command to predict the "then" or output data.

HSMs overcome this challenge by creating genuinely random keys. They achieve this by capturing data from random physical phenomena nearby, such as atmospheric noise or atomic decay processes, to generate unpredictable values as the foundation for random keys.

Crucially, a hardware security module produces, stores, and employs these keys for performing signatures, encryptions, and other cryptographic operations—all within the protected confines of the HSM's secure environment.

As the cryptographic operation keys remain contained within the HSM, the environment offers unparalleled defense against logical attacks, making theft virtually impossible. Certain hardware security modules go a step further in safeguarding users against Trojans and insider threats by furnishing a secure execution environment for user applications. Within these setups, the entire application is developed and run within the secure confines of the HSM.

5.2.7 Optimal Strategies for Leveraging HSMs

Highlighted here are the paramount advantages and functionalities of hardware security modules to contemplate:

- **Key Management:** Implement robust key management practices to maximize the security of cryptographic keys stored within the HSM. This includes proper key generation, rotation, and storage procedures.

- **Integration with Applications:** Integrate HSMs seamlessly with your applications and systems. This may involve using APIs provided by the HSM vendor or implementing cryptographic libraries that support HSM integration.
- **Performance Optimization:** Optimize performance by carefully configuring the HSM and tuning your applications to minimize latency and maximize throughput. Consider factors such as encryption algorithms, key sizes, and cryptographic operations.
- **Regular Maintenance and Updates:** Regularly maintain and update HSMs to patch vulnerabilities and ensure they remain secure against emerging threats. Follow best practices for firmware updates and security patch management.
- **Strong random number generation:** Any HSM must have the capability to randomly generate numbers or pseudo-random number generation in order to support other cryptographic functions.
- **Tamper-resistance:** The HSM should delete all sensitive data or "zeroize" itself should it detect any abnormal electrical activity, physical penetration, unusual temperature, or other signs of tampering. This stops a successful attacker from retrieving the secret keys once they have gained physical access [avinetworks, 2024].

Certainly, hardware security modules come with several drawbacks, primarily revolving around cost, which varies based on the required levels of security and functionality.

5.3 Hardware Security Module Design To Counter Hardware Trojan Threat

Hardware Security Modules (HSMs) are meticulously designed cryptographic devices engineered to mitigate the risks posed by hardware trojans, which are malicious alterations or insertions within integrated circuits. HSMs typically integrate multiple layers of security mechanisms to safeguard against such threats. These defenses often include tamper-resistant hardware components, such as secure elements and physically unclonable functions (PUFs), which make it exceedingly difficult for attackers to manipulate or compromise the device. Additionally, HSMs employ stringent manufacturing and testing processes to detect and prevent any unauthorized modifications during production. Through these robust design strategies, HSMs offer a fortified barrier against hardware trojans, ensuring the integrity and confidentiality of sensitive cryptographic operations and data.

Our approach involves the creation of a Hardware Security Module featuring an encrypt/decrypt engine grounded in Petri net principles for algorithm modulation. We ensure the integrity of system components through the adoption of a Secure System-on-Chip (SoC) architecture, thereby offering physical protection. Data flow is comprehensively secured through the utilization of the encrypt/decrypt engine. We have detailed the entire system implementation and presented a case study [Guechi and Redjimi, 2023].

Research in various domains has extensively explored the utilization of HSMs for deploying security services, including but not limited to network infrastructure [Lesjak et al., 2016], security for agricultural sensors [Kloibhofer et al., 2020], and the protection of biomedical data [Canim et al., 2011].

5.3.1 Mathematical Background Of The Cryptosystem

Petri nets serve as a graphical and mathematical framework applied across various domains, encompassing the modeling and examination of discrete-event systems, concurrent systems, fault-tolerant systems, and numerous other applications. As a mathematical formalism, they enable the establishment of state equations, algebraic equations, and additional mathematical representations that govern system behavior.

A Petri net is essentially a weighted directed graph comprising two distinct node types: places (depicted as circles) and transitions (illustrated as bars). These elements are interconnected by directed arcs, extending either from places to transitions or from transitions to places, with each arc annotated with a weight. The state of the net, known as marking and denoted by M, indicates the number of tokens present in each place, with M(p) representing the number of tokens in place p. A transition becomes enabled when the number of tokens in each input place meets or exceeds the weight specified on the input arc connected to that transition, denoted as w. To traverse a specific path within the net, the transition must be enabled, which entails removing a token from each input place and adding a token to each output place in accordance with the weight specified on the associated arc. In our context, we have opted to assign a uniform weight of 1 to the entire graph to simplify the implementation of the algorithm in its hardware manifestation.

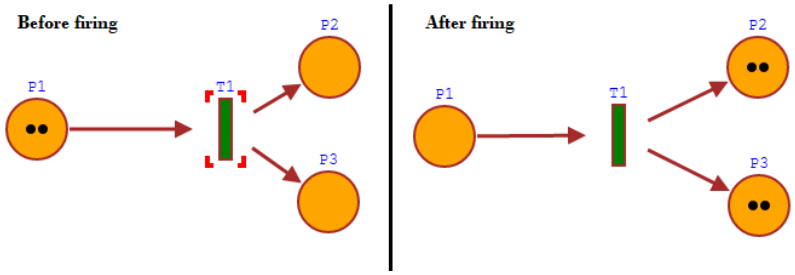


Figure 5.1: Transition Firing operation.

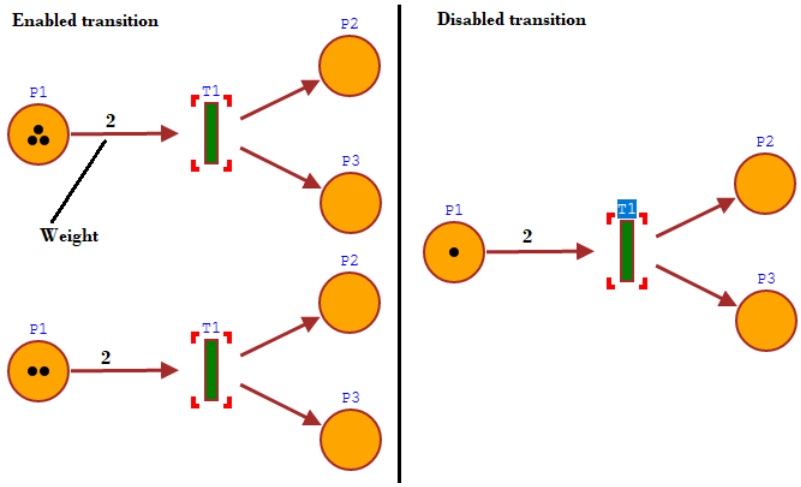


Figure 5.2: Enabled VS Disabled Transitions.

The formal representation of Petri net is as follows: $PN = (P, T, F, W, M0), P = p1, p2, p3, \ldots, pn$ is a finite set of places, $T = t1, t2, t3, \ldots, tn$ is a finite set of transitions, $F \subseteq (P \times T) \cup (T \times P)$is a set of arcs, W: $F \rightarrow \{1, 2, 3, \ldots, n\}$ is a weight function, $M_0 : P \rightarrow \{0, 1, 2, 3, \ldots, n\}$ is the initial marking. The structure $N = (P, T, F, W)$ is the petri net structure without initial marking, and when there is an initial marking it would be denoted by $(N, M0)$.

A Petri net is a 5-tuple PN=⟨P,T,F,W,M⟩ where:
P={p1,p2,p3,...,pn} is a finite set of places,
T={t1,t2,t3,...,tn} is a finite set of transitions,
$F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs,
W: $F \rightarrow \{1, 2, 3, \ldots, n\}$ is a weight function,
M0:$P \rightarrow \{0, 1, 2, 3, \ldots, n\}$ is the initial marking,
P $\cap T = \emptyset$ and T $\cap P = \emptyset$.

Table 5.1: The formal definition of Petri net

5.3.2 Secure SoC Architecture

The protected SOC module guarantees the integrity of components, thereby offering physical safeguarding for confidential keys and sensitive firmware [Haj-Yahya et al., 2019]. Utilizing buffers within the internal RAM prevents the exposure of secret keys and interim values during cryptographic processes. This is facilitated by employing a secure boot loader to ensure the initiation from authentic OS or firmware possessing appropriate system privileges. This OS configures the memory management unit (MMU) to exclusively grant access to buffers in the internal RAM to secure system processes with requisite system privileges. The secure ROM is pre-programmed, granting users complete freedom to initialize it with a master key, which is later utilized to derive the secret key during runtime. This arrangement complicates unauthorized access to the key since it's not directly stored in the ROM. In the internal RAM, specific buffers are designated to uphold the confidentiality of secret keys and interim values during cryptographic oper-

ations. During boot-up, the OS configures the MMU to assign the appropriate access privileges to authorized system processes for interacting with these buffers.

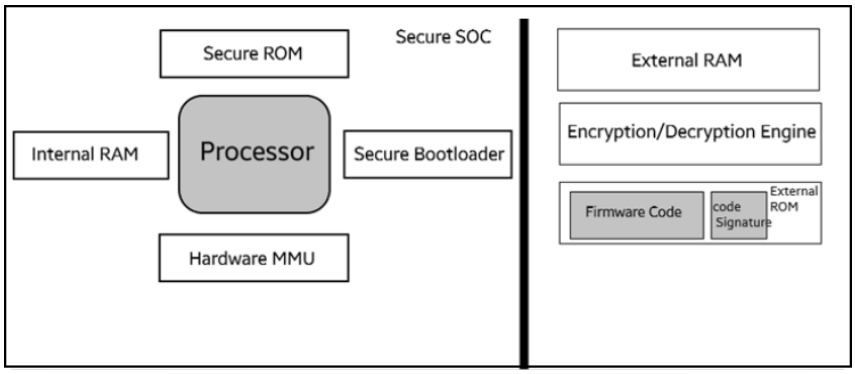


Figure 5.3: Chip Components are protected inside Secure SoC.

During initialization, the secure boot loader verifies the firmware to ensure booting from the authentic version, as it houses crucial code responsible for configuring access permissions to the internal RAM. This verification process entails the utilization of the firmware code's code signature module and the public key for code verification.

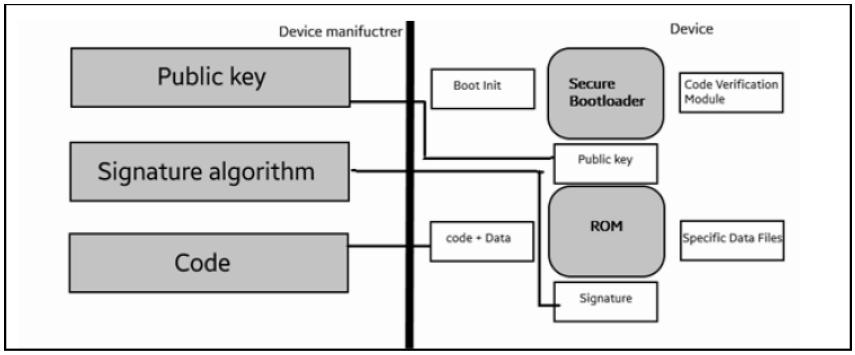


Figure 5.4: Code Signing Operation.

5.3.3 Cryptosystem Design and Implementation

Designing a cryptosystem based on Petri nets involves leveraging the dynamic and concurrent nature of Petri nets to create a secure and efficient cryptographic framework. Petri nets can model complex systems with concurrent activities and transitions, making them suitable for cryptographic applications where multiple operations need to be performed simultaneously. In this cryptosystem, cryptographic algorithms and protocols are represented as transitions, while data and cryptographic keys are represented as tokens in the Petri net. The transitions represent cryptographic operations such as encryption, decryption, key generation, and key exchange, while the places in the Petri net hold the intermediate states of the cryptographic process. By designing the Petri net structure appropriately and defining the transitions and places to enforce security properties such as confidentiality, integrity, and authentication, a robust and reliable cryptosystem can be developed. Additionally, the Petri net model allows for analysis and verification of the cryptosystem's security properties, aiding in the detection and mitigation of potential

vulnerabilities. Overall, leveraging Petri nets for cryptosystem design offers a versatile and effective approach to developing secure cryptographic solutions.

Regarding the encryption/decryption mechanism utilized for securing sensitive data transmitted to and from the device, our cryptosystem employs a Petri net framework to generate a sophisticated random number, serving as a private key for our engine. This process involves leveraging the Petri net's state markings following specific transitions, known exclusively to us as SOC manufacturers. Consequently, we ensure that solely encrypted data traverses the bus to and from the engine, rendering futile any attempt by unauthorized users to monitor the bus. Furthermore, our proposed Petri net comprises six transitions and six places, with flexibility for adaptation based on user requirements or security protocol enhancements.

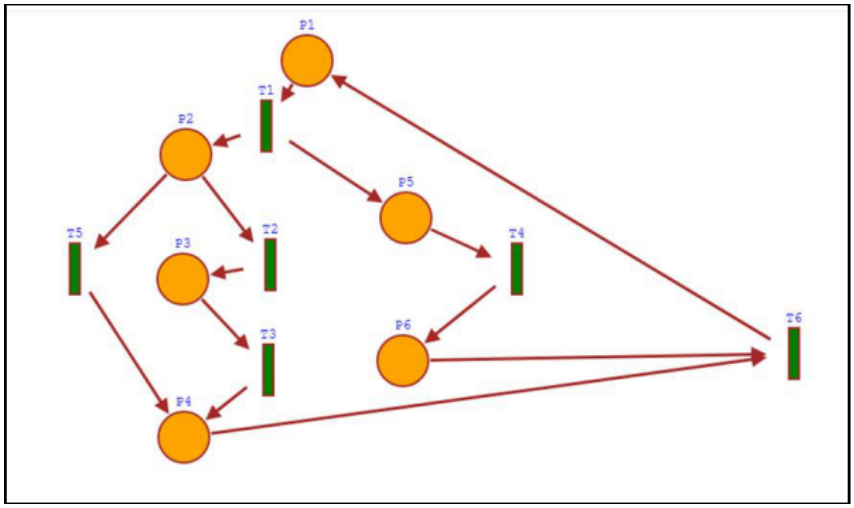


Figure 5.5: The suggested Petri net used to generate a private key.

The algorithm used to generate the private key for Encrypt/Decrypt operations is as follows:

Algorithm to generate the private key

Input: The master key
Output: The private key

BEGIN

- turn the master key into its ASCII code then convert it to binary code and divide it into a sequence of 8 bits each, we get this sequence S= S0, S1, S2,. . . . . .
- Initialize the marking of the Petri net with S: place P1 with S0, place P2 with S1 . . . , while the remaining places are set to zero.
- Fire the enabled transitions N time.
- Finally take the resulting marking of the Petri net as a private key.

END

The algorithm used to Encrypt/Decrypt sensitive data is as follows:

Algorithm for the data encryption

Input: Data
Output: Encrypted Data

BEGIN

- Turn data into its ASCII code then convert it to binary code.
- Make the XOR operation of the result with the previously generated private key.
- Divide the resulting sequence into 8 bits each and get its corresponding ASCII code which will be the cypher text.

END

For decryption operations, we substitute only the plain data with the encrypted data, and we perform the same steps to get back the plain data. Finally, all this work has been wired into hardware, and to demonstrate the circuit in action, LOGISIM is used for simulation purposes (see appendix A.6)

In the hardware representation of the cryptosystem, each register corresponds to a place in the petri net and each comparator corresponds to a transition. During SOC setup, the user will input values into our circuit to initialize the secure ROM Figure 5.6. This includes setting the desired master key value and specifying the number of times, denoted as N, that the counter will trigger the circuit transitions. Once the execution finishes, the resulting private key is stored in the registers. Additionally, 8-bit registers were utilized solely for simulation purposes; however, users have the flexibility to employ larger registers based on their security policies and system storage needs.

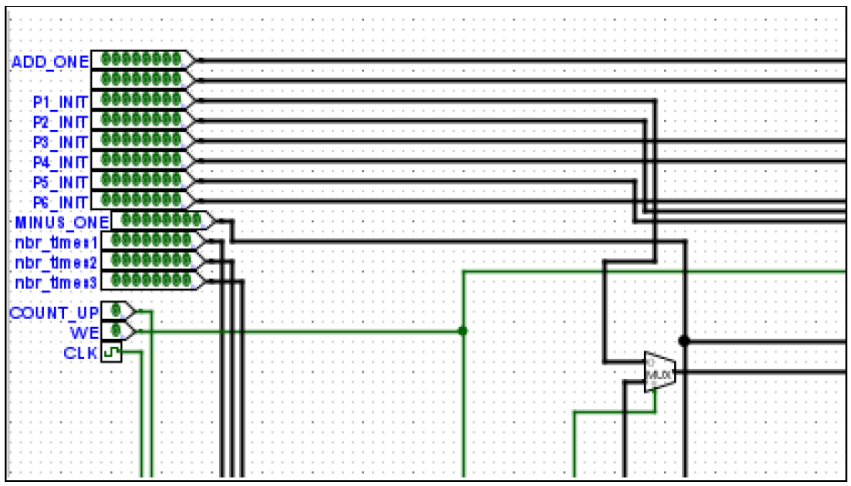


Figure 5.6: System Inputs.

5.3.4 Results and Discussion

To comprehensively illustrate the outcomes, capabilities, and limitations of our Secure System On Chip (SOC), we introduce a detailed case study. In this scenario, we employ the term "alpha" as the master key, utilized to derive both the private key and the plaintext message "hit the target," which represents sensitive data intended for transmission to an advanced military command post. Upon inputting the master key "alpha" into the algorithm designated for private key generation, we obtain a specific 64-bit key, considering N as 10 in this instance (though we can adjust N to a larger value as necessitated by security requirements and system performance considerations):
PK = 0000000000000000010000000101111101110000011101100110000100000001

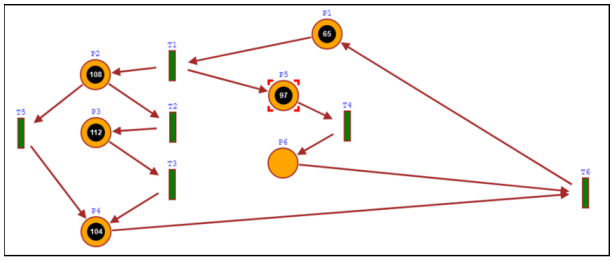


Figure 5.7: Master key Initialization.

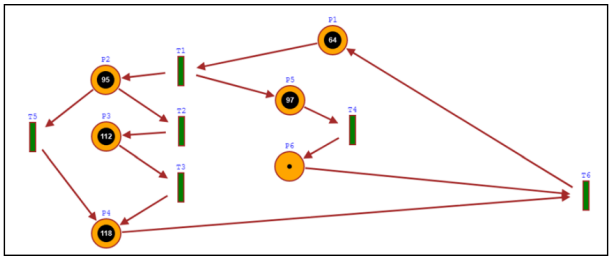


Figure 5.8: Generated Private Key

After converting the intended message (hit the target) into ASCII code and subsequently into binary code, the resulting sequence is as follows:
01101000011010010111010001110100011010000110010101110100011000010111001001101
101110100

After performing the XOR operation on the two preceding outcomes and obtaining the associated ASCII code, we arrive at the following ciphertext:
hitthe4> STX (DC1) (EOT) u

The system performance has been tested depending on the number of firing Petri net transitions (N) and the length of the master key Figure 5.9.

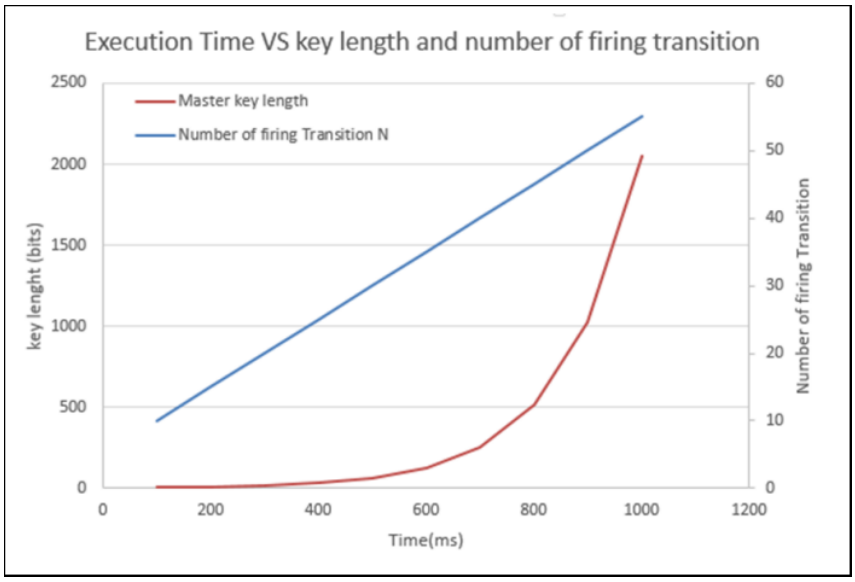


Figure 5.9: Execution Time VS key length and number of firing transition.

The system demonstrates its strength through the utilization of a highly robust encryption/decryption system rooted in Petri net, a methodology exclusive to SOC manufacturers. Consequently, deciphering the encryption would pose a formidable challenge for any potential attacker. Moreover, to safeguard against key exposure, secret keys are not directly stored on the chip; instead, they are dynamically generated during execution. The integrity of the SOC firmware is meticulously maintained through code signing, ensuring booting from authentic firmware. Nonetheless, some performance limitations arise, notably in execution time, attributed to the cyclic encryption algorithm's requirement for multiple rounds to generate intricate encryption keys. Additionally, the extended key length necessitates substantial storage resources, a critical concern in embedded systems.

5.4 Comparison to existing approaches

The table below resumes a brief comparison between our approach and the existing approaches in terms of mitigation for Hardware Trojans.

Table 5.2: Comparison between the existing approaches for Hardware Trojan Mitigation

Aspect	**Kloibhofer et al., 2020**	**Guechi & Redjimi,. 2023**	**Canim et al., 2011**
Mitigation objective	Prevent cyber and software attacks through asset isolation	Limit the impact of hardware Trojans at runtime	Protect sensitive medical data
Primary threat addressed	Key extraction, firmware manipulation, network attacks	Data leakage and corruption caused by hardware Trojans	Software attacks, unauthorized access
Protection after Trojan activation	Not supported	Supported through enforced security controls	Not supported
Data confidentiality under HT	Not guaranteed	Preserved via mandatory encryption	Not guaranteed
Data integrity under HT	Not guaranteed	Preserved via runtime integrity verification	Not guaranteed
Suitability for untrusted supply chains	Low	High	Low

Kloibhofer et al. mitigate security threats by isolating cryptographic assets within a hardware security module, effectively preventing software and network-based attacks. However, a trustworthy hardware platform is needed and it is vulnerable to Hardware Trojans. In contrast, our proposed approach in the image of HT-aware HSM architecture that mitigates Trojan impact at runtime by enforcing encryption, integrity, and controlled data access, so confidentiality and integrity are preserved even when malicious hardware is present.

Canim et al. employ cryptographic hardware to mitigate unauthorized access and protect biomedical data confidentiality and integrity, trusted hardware platform is needed. In contrast, in our proposed approach the hardware itself may be compromised, HT-aware HSM architecture that mitigates Trojan impact at runtime by enforcing encryption, integrity, and controlled data access even after Trojan activation has been proposed.

5.5 Conclusion

This chapter delves into the intricacies of Hardware Security Modules (HSMs). It begins with an introduction to HSMs, explaining their purpose, functionality, and architecture. The chapter highlights the various applications of HSMs and contrasts them with Trusted Execution Environments (TEEs) and Trusted Platform Modules (TPMs), outlining their advantages and unique characteristics. Furthermore, it discusses optimal strategies for utilizing HSMs effectively.

We have explored both hardware and software strategies to enhance security in System-on-Chip (SoC) and embedded systems. Besides we have Developed an HSM featuring a secure SoC architecture to safeguard the security of circuit components and maintain data integrity. The HSM's encryption and decryption engine ensures protection, utilizing the Petri net modulation technique for its encryption/decryption algorithm. Our method is highly secure for protecting the secret keys used in encryption and decryption operations, as we do not store these keys directly in the HSM's storage. Instead, we generate them during execution using our engine, which employs the Petri net algorithm as a secret key generator.

In summary, this chapter shifts focus to the design of HSMs aimed at countering hardware Trojan threats. It provides a detailed mathematical background of the cryptosystem, followed by a discussion on secure System-on-Chip (SoC) architecture and the implementation of the cryptosystem. The chapter concludes with the presentation of results and a discussion on the effectiveness of these security measures.

Conclusion and perspectives

This research aimed to evaluate the security of systems on chip (SoCs), especially the software-hardware interface, regarding any suspicious attacks. For that, we have conducted a deep study on how these attacks work and how they affect the integrity of such systems and hamper their normal function. Besides, this doctoral thesis has explored the main critical issues posed by hardware Trojans in integrated circuits, and it showed potential threats to security and whole system performance. An in-depth investigation of existing studies to detect and deter hardware Trojans has been conducted. The globalization of supply chains has made hardware Trojans a growing concern about the complexity of hardware systems.

Limitations and effectiveness of the existing detection methods have been highlighted, especially in terms of the ability to cover all sorts of anomalies that would threaten such hardware systems. Considering the highlighted issue, we have shown a technique to detect hardware Trojans externally activated, and we have demonstrated our work by emulating the attack on an FPGA using a Microcontroller, radio receivers and transmitters to perform the attack. Mitigation has also been taken into consideration, and for that, we have shown a new methodology that leverages the advanced Hardware Security Module (HSM) that consists of a powerful cryptosystem to secure the whole data flow and to prevent sensitive information from being leaked by unauthorized third-party users.

Results demonstrated the effectiveness of our detection/mitigation approaches in comparison to existing approaches. However, hardware Trojans are becoming more stealthy and dynamic, and no single detection strategy can be fully effective in revealing such a threat. Besides, hardware Trojans could hurt the system at multiple emplacements ranging from complete denial of service to causing the system to shut down completely. Due to the sophisticated nature of hardware Trojans, effective detection and mitigation require a combination of advanced techniques in order to be able to put hardware Trojans' harmful effects away from the system on chip (SoC).

As hardware Trojans continue to evolve in complexity and stealth, future research and development efforts must focus on creating more advanced and adaptive detection and mitigation strategies. One promising direction involves **machine learning and artificial intelligence (AI)**, which can analyze large datasets of circuit behavior and side-channel signals to identify subtle deviations caused by Trojans. AI-driven models, particularly those using deep learning, could enhance detection accuracy by learning patterns that traditional methods might miss.

In addition, **hardware-embedded security features** will play a key role in future designs. By incorporating security mechanisms directly into the design and manufacturing

process, it will be possible to create tamper-resistant hardware that can autonomously detect anomalies in real-time. **Blockchain technology** may also offer a potential solution for ensuring supply chain integrity by providing transparent and immutable tracking of hardware components throughout the manufacturing process.

Moreover, **formal verification techniques** need further refinement to scale with the increasing complexity of modern integrated circuits. Combining formal methods with probabilistic and statistical analysis can help cover a wider range of potential Trojan insertion points while reducing false positives.

On the mitigation front, **runtime monitoring** and **self-healing architectures** could detect Trojan activation during operation and respond by isolating or disabling affected components, minimizing the impact on system performance. Finally, establishing **global security standards and certification processes** for hardware manufacturing will be crucial in creating a more secure ecosystem, where trusted and verified practices become the norm across all stages of production.

In conclusion, the future of hardware Trojan detection and mitigation will require a multi-faceted approach, integrating advanced technologies, security-by-design principles, and industry-wide collaboration to stay ahead of increasingly sophisticated threats.

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.

[Amin and Mahmood-ul Hasan, 2022] Amin, A. A. and Mahmood-ul Hasan, K. (2022). Unified fault-tolerant control for air-fuel ratio control of internal combustion engines with advanced analytical and hardware redundancies. *Journal of Electrical Engineering & Technology*, 17(3):1947–1959.

[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.

[avinetworks, 2024] avinetworks (2024). What are Hardware Security Modules? Definition & FAQs — Avi Networks — avinetworks.com. `https://avinetworks.com/glossary/hardware-security-modules/`. [Accessed 10-04-2024].

[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 et al., 2017] Banga, A., Kumar, P., and Alazab, M. (2017). Hardware trojan detection in a combinational and sequential circuits. In *2017 IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, pages 1–5. IEEE.

[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.

[Bas et al., 2022] Bas, F., Benedicte, P., Alcaide, S., Cabo, G., Mazzocchetti, F., and Abella, J. (2022). Safedm: a hardware diversity monitor for redundant execution on non-lockstepped cores. In *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 358–363.

[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.

[Bình, 2023] Bình, N. (2023). Vhdl Code For Full Adder Using Behavioral Modeling? Trust The Answer - vi-magento.com — vi-magento.com. `https://vi-magento.com/vhdl-code-for-full-adder-using-behavioral-modeling-trust-the-answer/`. [Accessed 01-03-2024].

[Canim et al., 2011] Canim, M., Kantarcioglu, M., and Malin, B. (2011). Secure management of biomedical data with cryptographic hardware. *IEEE Transactions on Information Technology in Biomedicine*, 16(1):166–175.

[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.

[Guechi and Redjimi, 2021] Guechi, B. and Redjimi, M. (2021). *Hardware Trojan Detection in Heterogeneous Systems on Chip*, page 1105–1116. Springer International Publishing.

[Guechi and Redjimi, 2023] Guechi, B. and Redjimi, M. (2023). Hardware security module cryptosystem using petri net. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, 11(2):494–502.

[Haj-Yahya et al., 2019] Haj-Yahya, J., Wong, M. M., Pudi, V., Bhasin, S., and Chattopadhyay, A. (2019). Lightweight secure-boot architecture for risc-v system-on-chip. In *20th International Symposium on Quality Electronic Design (ISQED)*, pages 216–223. IEEE.

[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.

[Jain et al., 2021] Jain, A., Zhou, Z., and Guin, U. (2021). Survey of recent developments for hardware trojan detection. In *2021 ieee international symposium on circuits and systems (iscas)*, pages 1–5. IEEE.

[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.

[Kim and Kim, 2019] Kim, Y. and Kim, E. (2019). htpm: Hybrid implementation of trusted platform module. page 3–10. Association for Computing Machinery.

[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.

[Kinugawa et al., 2019] Kinugawa, M., Fujimoto, D., and Hayashi, Y. (2019). Electromagnetic information extortion from electronic devices using interceptor and its countermeasure. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(4):62–90.

[Kloibhofer et al., 2020] Kloibhofer, R., Kristen, E., and Davoli, L. (2020). Lorawan with hsm as a security improvement for agriculture applications. In *Computer Safety, Reliability, and Security. SAFECOMP 2020 Workshops: DECSoS 2020, DepDevOps 2020, USDAI 2020, and WAISE 2020, Lisbon, Portugal, September 15, 2020, Proceedings 39*, pages 176–188. Springer.

[Lesjak et al., 2016] Lesjak, C., Bock, H., Hein, D., and Maritsch, M. (2016). Hardware-secured and transparent multi-stakeholder data exchange for industrial iot. In *2016*

IEEE 14th International Conference on Industrial Informatics (INDIN), pages 706–713. IEEE.

[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.

[Muñoz and Fernandez, 2020] Muñoz, A. and Fernandez, E. B. (2020). Tpm, a pattern for an architecture for trusted computing. New York, NY, USA. Association for Computing Machinery.

[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.

[Sekiguchi and Seto, 2008] Sekiguchi, H. and Seto, S. (2008). Proposal of an information signal measurement method in display image contained in electromagnetic noise emanated from a personal computer. In *2008 IEEE Instrumentation and Measurement Technology Conference*, pages 1859–1863.

[Shi et al., 2019] Shi, Q., Vashistha, N., Lu, H., Shen, H., Tehranipoor, B., Woodard, D. L., and Asadizanjani, N. (2019). Golden gates: A new hybrid approach for rapid hardware trojan detection using testing and imaging. In *2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 61–71.

[Simons, 2022] Simons, M. (2022). What is Link library in VLSI?; Davidgessner — davidgessner.com. `https://www.davidgessner.com/essay-types-tips/what-is-link-library-in-vlsi/`. [Accessed 20-02-2024].

[Skorobogatov, 2012] Skorobogatov, S. (2012). Hardware assurance and its importance to national security. *Available On-line:¡ http://www. cl. cam. ac. uk/sps32/secnews. html*.

[Stagner, 2013] Stagner, C. (2013). *Detecting and locating electronic devices using their unintended electromagnetic emissions*. Phd thesis, Missouri University Of Science and Technology.

[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 Koushanfar, 2016] Tehranipoor, M. and Koushanfar, F. (2016). A survey of hardware trojan taxonomy and detection. *IEEE Design & Test of Computers*, (01):1–1.

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

[Tidrea et al., 2019] Tidrea, A., Korodi, A., and Silea, I. (2019). Cryptographic considerations for automation and scada systems using trusted platform modules. *Sensors*, 19(19).

[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., 2022] Wang, W., Wang, X., Wang, J., Xiong, N. N., Cai, S., and Liu, P. (2022). Ensuring cryptography chips security by preventing scan-based side-channel attacks with improved dft architecture. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 52(3):2009–2023.

[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.

[Wild and Ramchandran, 2005] Wild, B. and Ramchandran, K. (2005). Detecting primary receivers for cognitive radio applications. In *First IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks, 2005. DySPAN 2005.*, pages 124–130.

[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.

[Zhang and Zhao, 2021] Zhang, X. and Zhao, X. (2021). Architecture design of distributed redundant flight control computer based on time-triggered buses for uavs. *IEEE Sensors Journal*, 21(3):3944–3954.

Appendix A

Appendix

A.1 FPGA Code Implementation To Use Radio Receiver.

```
module RadioReceiverInterface (
input wire clk,          // System clock
input wire reset,        // System reset
input wire data_in,      // Data from the radio receiver module
output reg [7:0] data    // Decoded data output
);

parameter IDLE = 2'b00, DATA = 2'b01;
reg [1:0] state;
reg [15:0] count;
reg [7:0] shift_reg;

always @(posedge clk or posedge reset) begin
if (reset) begin
state <= IDLE;
count <= 16'b0;
shift_reg <= 8'b0;
data <= 8'b0;
end else begin
case (state)
IDLE: begin
if (data_in == 1'b1) begin
state <= DATA;
count <= 16'b0;
end
end

DATA: begin
count <= count + 1'b1;
if (data_in == 1'b0) begin
if (count > THRESHOLD) begin
shift_reg <= {shift_reg[6:0], 1'b1};
end else begin
shift_reg <= {shift_reg[6:0], 1'b0};
end
count <= 16'b0;
end

```

```
39      if (shift_reg[7:0] == 8'hFF) begin
40      data <= shift_reg;
41      state <= IDLE;
42      end
43      end
44
45      default: state <= IDLE;
46      endcase
47      end
48      end
49      endmodule
```

Listing A.1: Verilog code to interface with radio receiver module

A.2 FPGA Code Implementation To Use Radio Transmitter

```
1       module RadioTransmitter (
2       input wire clk,           // System clock
3       input wire reset,         // System reset
4       output wire data_out      // Data output to radio transmitter
5       );
6
7       reg [23:0] counter;       // Counter to divide the clock
8       reg data_reg;             // Register to hold the data output
9
10      // Set the desired frequency for the square wave (example: 1 kHz)
11      // Assuming a 50 MHz clock on the Nexys 2
12      localparam TARGET_FREQ = 1000;
13      localparam CLOCK_FREQ = 50000000;
14      localparam COUNT_MAX = CLOCK_FREQ / (2 * TARGET_FREQ);
15
16      always @(posedge clk or posedge reset) begin
17      if (reset) begin
18      counter <= 24'b0;
19      data_reg <= 1'b0;
20      end else begin
21      if (counter >= COUNT_MAX - 1) begin
22      counter <= 24'b0;
23      data_reg <= ~data_reg;  // Toggle data output
24      end else begin
25      counter <= counter + 1'b1;
26      end
27      end
28      end
29
30      assign data_out = data_reg;
31
32      endmodule
```

Listing A.2: Verilog code to interface with radio transmitter module

A.3 Code To Capture Raw Data (Data Without Encryption)

```
from machine import Pin
import time

# Configure the GPIO pin for the receiver
receiver = Pin(4, Pin.IN)  # D2 pin

def receive_signal():
# Example: read the received signal
return receiver.value()

def main():
last_signal = None
while True:
signal = receive_signal()
if signal != last_signal:
print("Received Raw data:", signal)
last_signal = signal
time.sleep(0.01)  # Small delay to debounce

if __name__ == "__main__":
main()
```

Listing A.3: Python Code used on the microcontroller to capture data sent from the FPGA

A.4 Code To cause the encryption process to stop

```
from machine import Pin
import time

# Configure the GPIO pin for the transmitter
transmitter = Pin(5, Pin.OUT)  # D1 pin

def send_signal(data):
for bit in data:
if bit == '1':
transmitter.value(1)
else:
transmitter.value(0)
time.sleep(0.01)  # Adjust timing as needed

def str_to_binary(data):
return ''.join(format(ord(char), '08b') for char in data)

def main():
message = "stop_encryption" #String to be sent the FPGA
binary_message = str_to_binary(message)

while True:
send_signal(binary_message)
time.sleep(1)  # Send message every second

if __name__ == "__main__":
main()
```

Listing A.4: Python Code used on the microcontroller to cause the encryption process on the FPGA to stop

A.5 Conceptual Python Implementation To Practically Illustrate Our Detection Technique

```
import numpy as np
import matplotlib.pyplot as plt

def compute_psd(x, M, sampling_rate):
freqs = np.fft.fftfreq(M, 1/sampling_rate)
psd = np.zeros(M)

for k, f in enumerate(freqs):
exp_term = np.exp(-1j * 2 * np.pi * np.arange(M) * f /
    sampling_rate)
psd[k] = np.abs(np.sum(x[:M] * exp_term)) / M

return freqs, psd

# Parameters
sampling_rate = 1000  # Example: 1000 samples per second
M = 1024              # Number of samples
t = np.arange(M) / sampling_rate

# Simulate received signal: s(n) + LO signal
f_LO = 100           # Local oscillator frequency (Hz)
s = np.sin(2 * np.pi * 50 * t)  # Example signal of interest at 50
    Hz
lo_signal = np.cos(2 * np.pi * f_LO * t)
x = s + lo_signal

# Compute PSD
freqs, psd = compute_psd(x, M, sampling_rate)

# Plot PSD
plt.plot(freqs[:M//2], psd[:M//2])  # Plot only positive frequencies
plt.xlabel('Frequency (Hz)')
plt.ylabel('Power Spectral Density')
plt.title('Power Spectral Density of Received Signal')
plt.show()
```

Listing A.5: Python Code To Detect Local Oscillator Signal

A.6 Demonstrations highlight the practical challenges and limitations of using the PSD method to detect the LO signal in a radio receiver

Here are the plots (Figure A.1) demonstrating the drawbacks of the PSD method for detecting the local oscillator signal:

1. **PSD with Noise:** The noise raises the baseline level, making it harder to distinguish the peaks at 50 Hz (signal of interest) and 100 Hz (LO signal). The peaks are less prominent due to the noise interference.

2. **High-Resolution PSD:** The peaks at 50 Hz and 100 Hz are sharper and more distinct compared to the lower resolution plot. Higher resolution allows for better discrimination of closely spaced frequency components but requires more computational resources.

3. **PSD with Spectral Leakage:** The peak at 52.5 Hz spreads power across adjacent frequency bins, demonstrating spectral leakage. This spreading can obscure other frequency components and distort the PSD, making it challenging to accurately identify the LO signal.

These plots illustrate the practical challenges of using the PSD method for signal analysis, particularly in noisy environments, with limited resolution, and when dealing with spectral leakage.

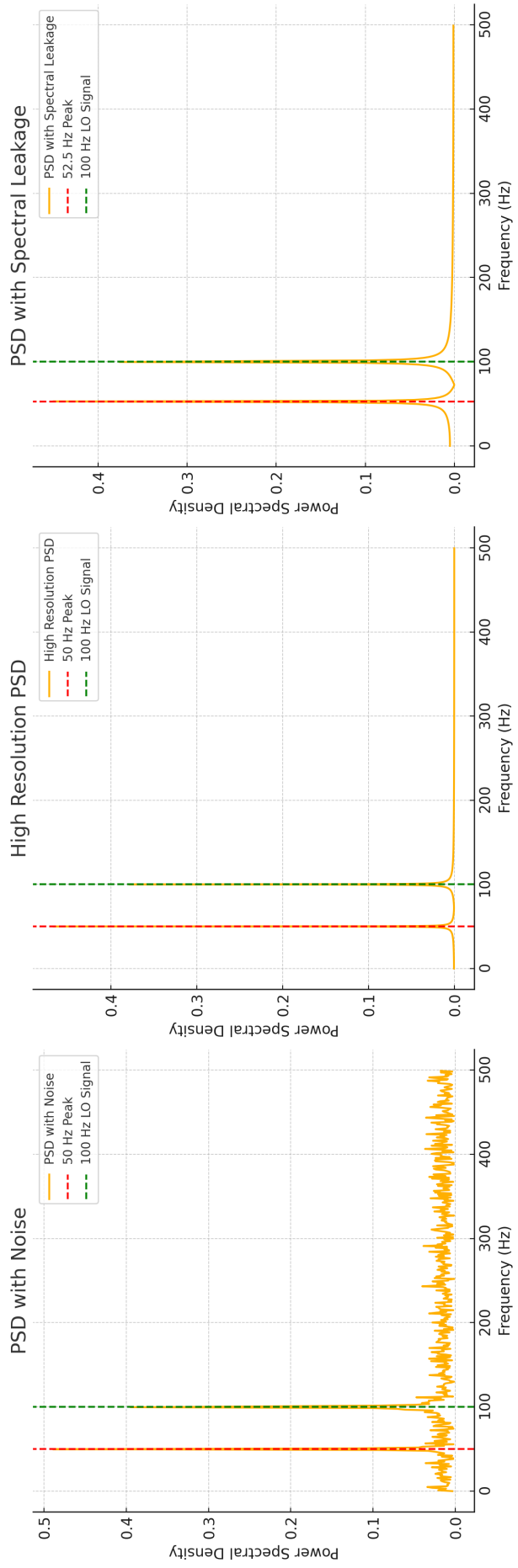


Figure A.1: drawbacks of the PSD method for detecting the local oscillator signal.

A.7 Hardware Representation of the Key Generator

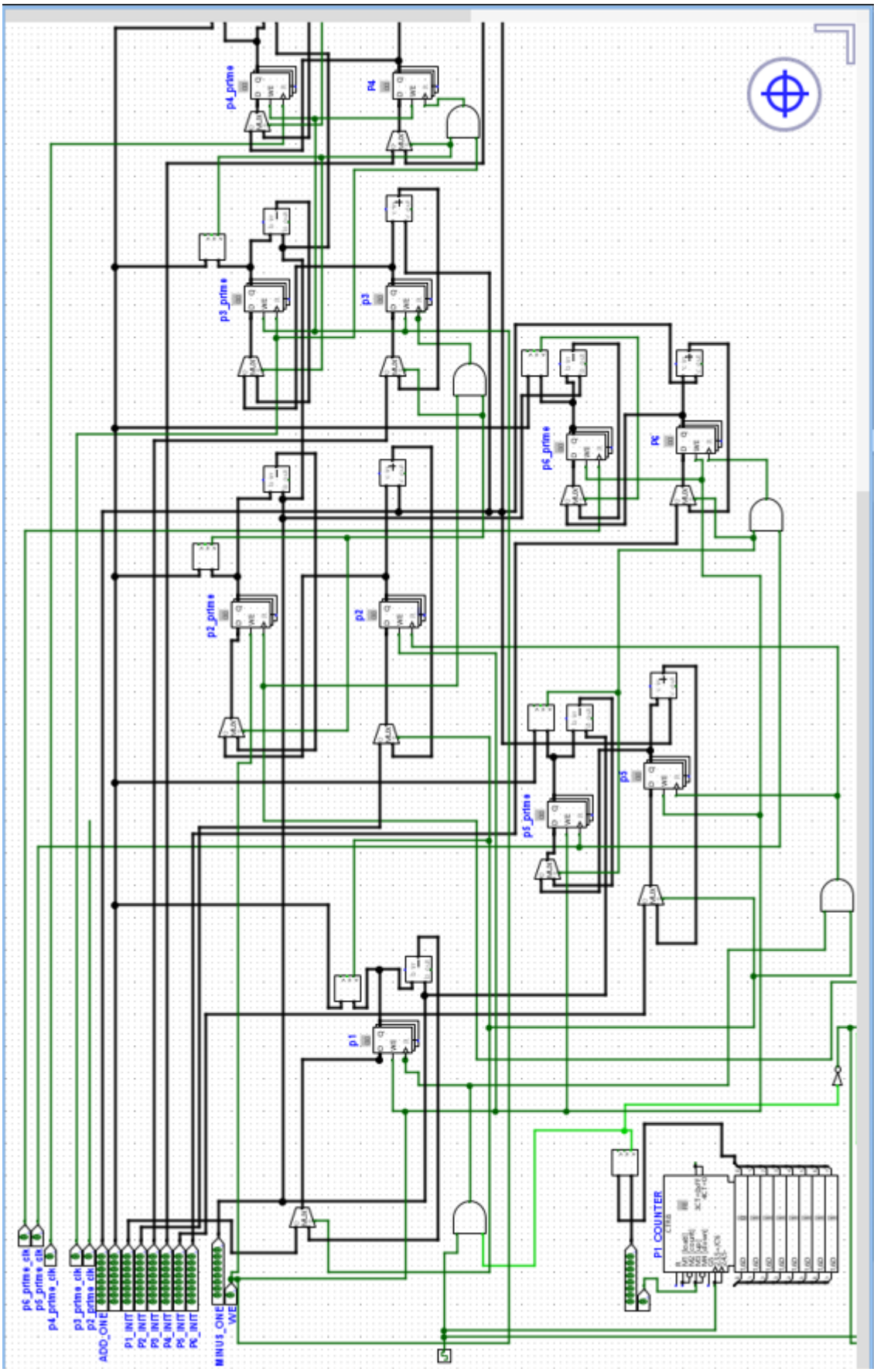
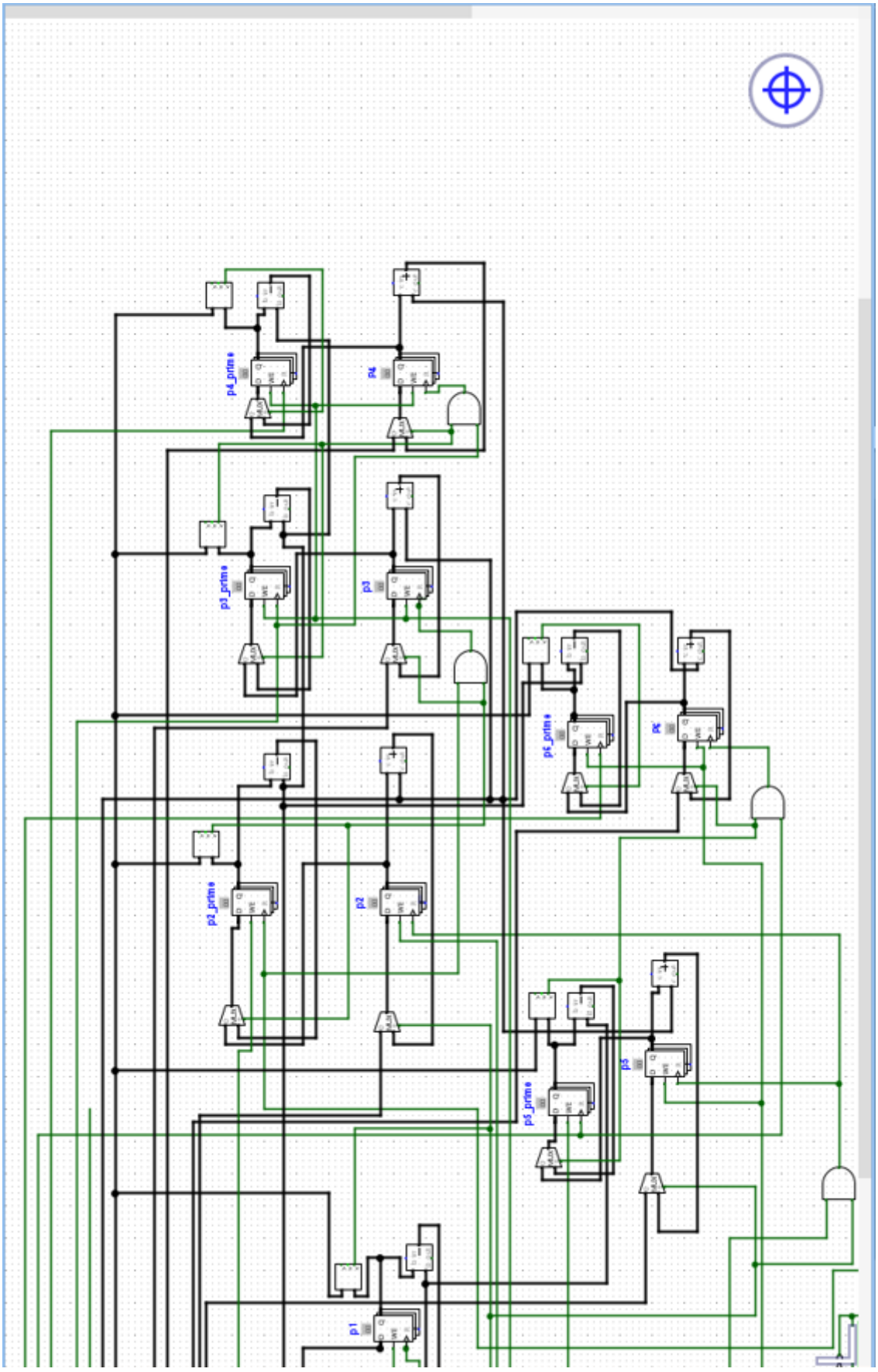
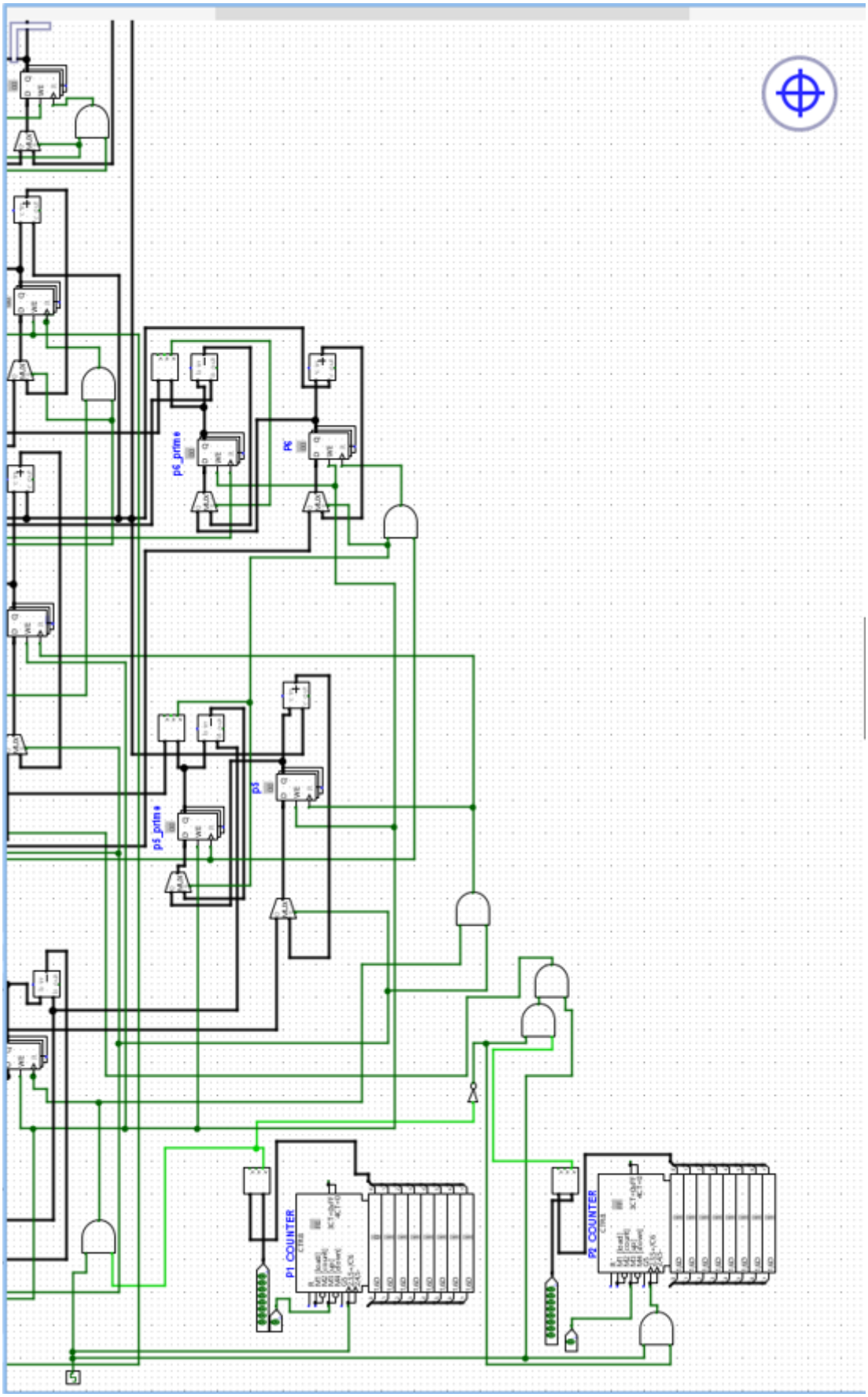


Figure A.2: Hardware Representation of the Key Generator


p4_prime
p4
p3_prime
p3
p6_prime
p6
p2_prime
p2
p5_prime
p5
p1


p6_prime
P6
p5_prime
p5
P1_COUNTER
P2_COUNTER