TAL TECH

DOCTORAL THESIS

On the Use of Defensive Schemes for Hardware Security

Mohammad Eslami

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY
TALLINN 2024

TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
53/2024

On the Use of Defensive Schemes for Hardware Security

MOHAMMAD ESLAMI

TAL TECH PRESS

TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies
Department of Computer Systems

The dissertation was accepted for the defense of the degree of Doctor of Philosophy in Information and Communication Technologies on 7 July 2024

Supervisor: Prof. Dr. Samuel Pagliarini,
Department of Computer Systems, School of Information Technologies,
Tallinn University of Technology
Tallinn, Estonia

Co-supervisor: Dr. Tara Ghasempouri,
Department of Computer Systems, School of Information Technologies,
Tallinn University of Technology
Tallinn, Estonia

Opponents: Prof. Dr. Kaveh Razavi,
Department of Information Technology and Electrical Engineering,
Federal Institute of Technology (ETH) Zurich
Zurich, Switzerland

Prof. Dr. Philippe Maurine,
Laboratory of Computer Science, Robotics and Microelectronics of Montpellier (LIRMM),
University of Montpellier
Montpellier, France

Defence of the thesis: 4 October 2024, Tallinn

Declaration:
Hereby I declare that this doctoral thesis, my original investigation and achievement, submitted for the doctoral degree at Tallinn University of Technology, has not been submitted for any academic degree elsewhere.

Mohammad Eslami _____
signature

European Union
European Regional Development Fund
Investing in your future

ISSN 2585-6898 (publication)
ISBN 978-9916-80-196-3 (publication)
ISSN 2585-6901 (PDF)
ISBN 978-9916-80-197-0 (PDF)
DOI https://doi.org/10.23658/taltech.53/2024
Printed by Koopia Niini & Rauam

Eslami, M. (2024). *On the Use of Defensive Schemes for Hardware Security* [TalTech Press]. https://doi.org/10.23658/taltech.53/2024

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
53/2024

Kaitseskeemid riistvara turvalisuse tagamiseks

MOHAMMAD ESLAMI

TAL TECH
KIRJASTUS

Contents

List of Publications 7

Author's Contributions to the Publications 8

Abbreviations 9

1 Introduction 11
1.1 The Significance of Integrated Circuits in Our Lives 11
1.2 Threats in the Integrated Circuit Supply Chain 14
1.3 Countermeasures 16
1.4 Contributions and Outline of the Thesis 19

2 Background 21
2.1 Life cycle of an Integrated Circuit 21
2.1.1 Design 21
2.1.2 Fabrication 23
2.1.3 Testing and Packaging 25
2.1.4 Distribution 25
2.2 Security Vulnerabilities of Integrated Circuits 25
2.2.1 Hardware Trojan as a Major Security Risk 25
2.3 Countermeasures against Hardware Trojans 27
2.3.1 Detection Techniques 28
2.3.2 Design for Hardware Trust 31

3 Reusing Verification Assertions for Security Purposes 34
3.1 Assertions as Hardware Trojan Detectors 34
3.2 Binding the Assertions to the Main Design 35
3.3 Security Coverage 36
3.4 OpenTitan - A Case Study 38
3.5 Optimizing the Assertion List 39
3.6 Experimental Results 41
3.6.1 Optimizing the Assertions 42

4 Enhancing IC Security by Embedding Online Checkers during Physical Synthesis 45
4.1 Limitations of the Concept of Reusing Verification Assertions as Security Checkers 46
4.2 Adding Online Monitors during Physical Synthesis 46
4.2.1 Generation of Online Monitors for the Back-end Phase 47
4.2.2 Embedding Online Monitors into the Layout 48
4.2.3 ECO Flow 49
4.3 Experimental Results 50
4.3.1 Impact of Adding Online Monitors on SC 51
4.3.2 Impact of Adding Online Monitors on PPA 52
4.3.3 Comparison of the Presented Work with Other Techniques 55

5 SALSy: Security-Aware Layout Synthesis 59
5.1 Security Assessment Scheme 60
5.2 SALSy Techniques 61
5.2.1 Benchmarks 62

5.2.2 Open-source PDK 62
5.2.3 Countermeasures against FSP/FI 63
5.2.4 Countermeasures against HT Insertion 66
5.3 Scores for Open-source PDK and Comparisons 67
5.4 Silicon Validation of SALSy 68
5.4.1 Implementation for Commercial Process Design Kits 69
5.5 Results 72
5.5.1 Pre-silicon Results 72
5.5.2 Post-fabrication Results 73
5.6 SALSy Versus Other Techniques 76

6 Conclusions and Future Directions 78

List of Figures 81

List of Tables 82

References 83

Acknowledgements 95

Abstract 96

Kokkuvõte 97

Appendix 1 99

Appendix 2 107

Appendix 3 119

Appendix 4 137

Curriculum Vitae 153

Elulookirjeldus 155

List of Publications

The present Ph.D. thesis is based on the following publications.

I M. Eslami, T. Ghasempouri and S. Pagliarini, "Reusing Verification Assertions as Security Checkers for Hardware Trojan Detection," *In Proceedings of the 2022 23rd International Symposium on Quality Electronic Design (ISQED)*, Santa Clara, CA, USA, 2022, pp. 1-6, DOI: https://doi.org/10.1109/ISQED54688.2022.9806292

II M. Eslami, J. Knechtel, O. Sinanoglu, R. Karri, and S. Pagliarini, "Benchmarking Advanced Security Closure of Physical Layouts: ISPD 2023 Contest," *In Proceedings of the 2023 International Symposium on Physical Design (ISPD '23)*, New York, NY, USA, 2023, pp. 256–264. DOI: https://doi.org/10.1145/3569052.3578924

III M. Eslami, T. Ghasempouri and S. Pagliarini, "SCARF: Securing Chips with a Robust Framework against Fabrication-time Hardware Trojans," in IEEE Transactions on Computers, pp. 1-15, 2024. DOI: https://doi:10.1109/TC.2024.3449082

IV M. Eslami, T. Perez, and S. Pagliarini, "SALSy: Security-Aware Layout Synthesis," *In arXiv*, under review for IEEE Transactions on Dependable and Secure Computing, 2024. DOI: https://doi.org/10.48550/arXiv.2308.06201

Other related publications

V J. Knechtel, M. Eslami, et al. "Trojan Insertion versus Layout Defenses for Modern ICs: Red-versus-Blue Teaming in a Competitive Community Effort," accepted for publication in IACR Transactions on Cryptographic Hardware and Embedded Systems, pp. 1-41, 2025.

Author's Contributions to the Publications

I In Publication I, I was the main author and proved the idea of reusing the verification assertions as security checkers by investigating the security properties of verification assertions among 40 different IPs of the OpenTitan SoC using Cadence formal tools. I also prepared the results and wrote the manuscript.

II In Publication II, I was the main author, and I co-organized a global security contest. My role in the project involved designing various types of hardware Trojans and creating a fully automated flow for Trojan insertion into the submitted layouts by participants. I also prepared the figures and wrote the manuscript.

III In Publication III, I was the main author and developed a methodology to add online checkers to digital designs at the physical synthesis stage to protect against fabrication-time attacks. I also obtained the results, prepared the figures and wrote the manuscript.

IV In Publication IV, I was the main author and utilized various techniques during the physical synthesis step to protect the layout, resulting in a secure 65nm CMOS chip. I conducted experiments, tested the chip outputs, and analyzed results. Moreover, I prepared the figures and wrote the manuscript.

Abbreviations

3PIP	Third-Party Intellectual Property
ABV	Assertion Based Verification
AI	Artificial Intelligence
AO	Always On
ASIC	Application-Specific Integrated Circuit
BEOL	Back-End-Of-the-Line
BMC	Bounded Model Checking
CAD	Computer-Aided Design
CED	Concurrent Error Detection
CTS	Clock Tree Synthesis
DF	Degrading Factor
DfHT	Design for Hardware Trust
DFT	Design For Test
DMR	Dual Modular Redundancy
DRC	Design Rule Check
ECO	Engineering Change Order
FEOL	Front-End-Of-the-Line
FI	Fault Injection
FIB	Focused Ion Beam
FPGA	Field-Programmable Gate Array
FSP	Front-Side Probing
GDS	Graphic Data System
HDL	Hardware Description Language
HT	Hardware Trojan
IC	Integrated Circuit
IoT	Internet of Things
I/O	Input/Output
IP	Intellectual Property
IR	Instruction Register
ISPD	International Symposium on Physical Design
OEM	Original Equipment Manufacturer
PDK	Process Design Kit
PO	Primary Output
POS	Product Of Sum
PPA	Power-Performance-Area
PSL	Property Specification Language
RE	Reverse Engineering
RIE	Reactive Ion Etching
RTL	Register-Transfer Level
SALSy	Security-Aware Layout Synthesis
SC	Security Coverage
SoC	System on Chip
SOP	Sum Of Product
SPV	Security Path Verification
STA	Static Timing Analysis
SVA	SystemVerilog Assertion

TSMC	Taiwan Semiconductor Manufacturing Company
UV	UltraViolet

1 Introduction

The impact of digitalization has brought about significant changes in our daily lives, affecting the way we communicate, work, and interact as a society [1]. Integrated Circuits (ICs), or simply, computer chips, are a crucial component in modern electronics and have played a fundamental role in this technological revolution [2].

ICs have been essential in driving the technological advancements we experience today. These complex microelectronic components, which consist of an array of transistors on a small silicon substrate, form the foundation for the development and operation of a wide range of devices, from smartphones to sophisticated computing systems that power our daily lives [2, 3]. Their ability to condense complex functionalities into compact dimensions has been crucial in the exponential growth of computational power, leading to innovations that permeate every aspect of our interconnected world.

The fabrication of ICs involves a thorough and elaborate manufacturing process, **compulsorily** conducted in controlled cleanroom environments to ensure precision and reliability [4]. Scaling emerges as a major trend in IC fabrication, with the shrinking of transistors and the concentration of functionalities into a single chip. This not only enhances the overall performance of electronic devices but also contributes to a reduction in physical footprint, paving the way for the development of smarter and more powerful gadgets.

Within the domain of the IC supply chain, design houses occupy a key role. These entities are responsible for designing and developing the complex architecture of these microelectronic marvels, utilizing their engineering expertise and creativity to create the blueprint for future technologies. However, the complexity of IC fabrication often leads design houses to collaborate with specialized foundries for the actual manufacturing process. The most prominent players in this landscape are the Taiwan Semiconductor Manufacturing Company (TSMC), Intel, and Samsung [5]. These companies are well known for their cutting-edge facilities and extensive knowledge of semiconductor manufacturing.

Many design houses opt to outsource the manufacturing of their ICs to specialized foundries such as TSMC. This is due to the high cost of establishing and maintaining an in-house fabrication facility, which can easily exceed billions of dollars [6]. Even large companies like Apple prefer to outsource their chip fabrication to take advantage of the latest technologies and facilities offered by these foundries [7]. Outsourcing enables these companies to concentrate on their core strengths, such as design, while leveraging the expertise of external partners, like TSMC, in fabrication. This approach enables businesses to optimize efficiency and reduce costs.

The collaborative model between design houses and foundries, which is referred to as the fabless model, allows for flexibility, efficiency, and cost-effectiveness, all of which are crucial elements in navigating the dynamic landscape of semiconductor technology. By partnering with foundries, design houses can avoid the substantial capital investment required for establishing and maintaining in-house fabrication facilities, while still benefiting from the latest technologies and facilities.

1.1 The Significance of Integrated Circuits in Our Lives

ICs are a key part of technology advancements. They have changed the way we live and helped make progress in many areas. Their small size, excellent performance, and flexibility have allowed them to be used in many different ways. This has led to new developments in communication, computing, healthcare, transportation, and entertainment. This section discusses the essential role ICs play in our modern lives and how they help drive progress.

ICs are founded on the principles of integration, which involve the interconnection of

millions to billions of electronic components on a single silicon substrate. The miniaturization of transistors, guided by the famous Moore's Law [8], has led to rapid increases in computational power. At the same time, improvements in fabrication techniques have allowed for complex functionalities to be integrated within smaller and smaller devices [9]. From the simple beginnings of the transistor to the complex designs of modern microprocessors and system-on-chips (SoCs), ICs have evolved significantly. This evolution has made ICs a powerful force that drives the digital age.

ICs are found everywhere and play a crucial role in many industries due to their unique capabilities and functions. In communication, ICs form the backbone of wireless networks, enabling smooth connectivity and allowing data exchange on a global scale [10]. In computing, ICs power various devices that drive productivity and innovation, from smartphones and laptops to supercomputers and data centers [11]. In healthcare, ICs contribute to significant advancements in medical imaging, diagnostics, and treatment, transforming patient care and improving examination results [12]. Furthermore, ICs are vital in transportation systems, enhancing safety, efficiency, and sustainability through innovations in automotive electronics, avionics, and navigation systems [13]. Lastly, ICs enrich our leisure and entertainment experiences by powering gaming consoles, audiovisual equipment, and digital media platforms that captivate audiences around the world.

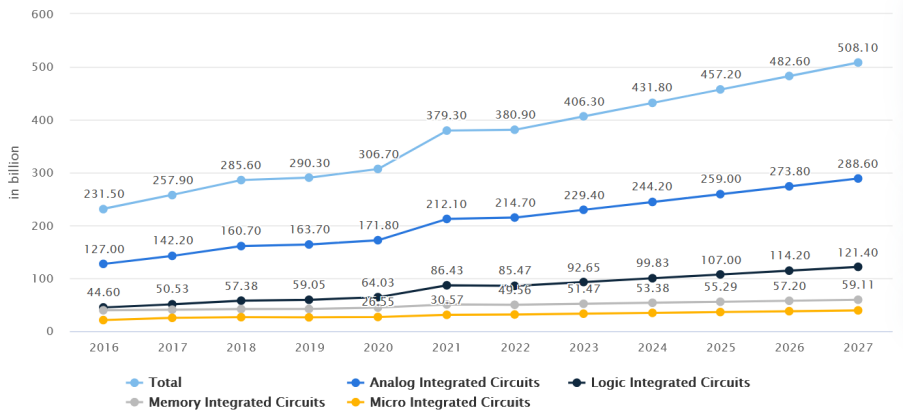


Figure 1: The estimated sales volume of the ICs based on the estimated end-market product volume (from [14])

The impact of ICs goes beyond just technological advancements, as they significantly influence societal dynamics, economic landscapes, and cultural norms. The widespread use of ICs has made information and resources more accessible, bridging geographical gaps and promoting global interconnectedness. Additionally, ICs have driven economic growth and innovation, creating new industries, job opportunities, and paths for entrepreneurship. As shown in Figure 1, the rapid increase in IC production is still expected to grow in the coming years [14]. This growth is driven by the increasing demand for electronic devices in various sectors, including consumer electronics, automotive, healthcare, and industrial applications. Moreover, the ongoing advancements in IC technology, such as the development of smaller, more powerful, and energy-efficient chips, are also contributing to the growth of the IC market. According to recent market research [14], shown in Figure 1, the sales volume of ICs is estimated to reach a value of over 508 billions by 2027.

Looking to the future, ICs hold enormous potential, with emerging technologies such as Artificial Intelligence (AI), the Internet of Things (IoT), and quantum computing set to

further expand the frontiers of possibility [15, 16]. Numerous companies are striving to capture a larger portion of the market by increasing their influence in the semiconductor industry.

The semiconductor industry is a significant player in the global economy, with IC fabrication and the sale of fabricated chips generating substantial revenue. Figure 2 presents the market shares of major semiconductor companies in the year 2022. However, this distribution is likely to change in the upcoming years, as only a few of these companies possess the capability to fabricate advanced ICs with feature sizes below 10nm [17]. This future also presents challenges that must be addressed, including ethical considerations, environmental sustainability, and ensuring equitable access to technological benefits. As those responsible for guiding technological progress, it is our duty to address these challenges thoughtfully and work towards creating a future where ICs contribute to a more sustainable, ethical, and equitable world.

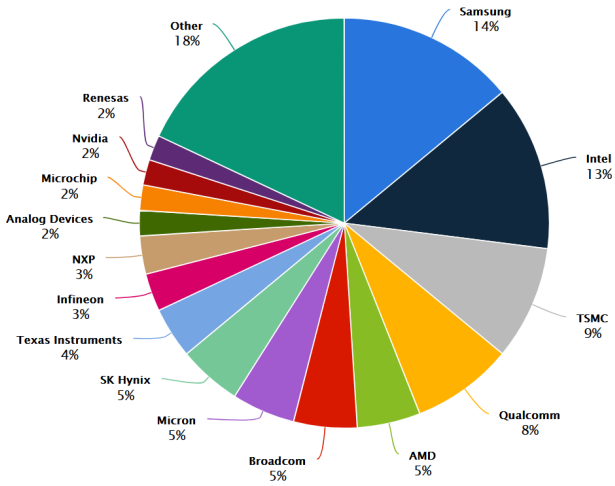


Figure 2: The market share of the selected leading brands of the total market size in the year 2022 (from [14])

It should be noted that in the context of this thesis, the term IC will almost always refer to Application-Specific Integrated Circuit (ASIC) unless otherwise specified. ASICs are a crucial class of ICs designed for specific applications, providing optimized performance and efficiency due to their custom-tailored design. To provide clarity and context, brief descriptions of various types of chips are presented in the following paragraphs.

Custom ASICs are specifically designed for a particular application or function. They offer unparalleled performance and efficiency because their design is optimized for specific tasks. Despite their high initial development costs, ASICs can achieve lower per-unit costs in high-volume production due to their fixed functionality once fabricated.

Field-Programmable Gate Arrays (FPGAs) provide flexibility and reprogrammability, allowing for post-manufacturing hardware configuration. This makes FPGAs ideal for rapid prototyping and for applications that require frequent updates. However, their general-purpose nature typically results in lower performance and higher power consumption compared to custom ASICs [18].

Microcontrollers combine a processor, memory, and peripherals on a single chip, optimized for control-oriented tasks. These chips are widely used in embedded systems for

applications such as sensor interfacing, motor control, and other low-power functions. Microcontrollers offer a balanced trade-off between flexibility and performance, capable of running software programs tailored to specific tasks.

1.2 Threats in the Integrated Circuit Supply Chain

While outsourcing the fabrication of ICs to specialized foundries offers numerous advantages, it also introduces a spectrum of threats and challenges that design houses must contend with [19]. In Figure 3, the overall life cycle of an IC, from conception to market distribution, is illustrated within a fabless model. The process begins at the design house, where the design phase is performed. During this phase, engineers meticulously define all specifications and intricacies of the IC, either by performing the digital design in its entirety or by incorporating Intellectual Properties (IPs) from Third-Party Intellectual Property (3PIP) vendors. The outcome of the design phase is a Graphic Data Stream (GDS) file, which encapsulates all components and interconnections of the design. This file is subsequently sent to the foundry for fabrication, where the chip is manufactured according to the design house's selected technology. Thereafter, the fabricated chip is forwarded to another facility for packaging. During the test and packaging phase, bare dies are packaged accordingly, and preliminary tests are conducted to ensure that the chips are free from defects. Finally, the chip is made available to the end-user through market distribution.

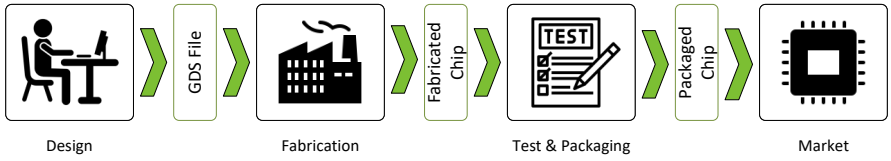


Figure 3: Different phases of IC's life cycle from design to market.

Once a chip is sent for fabrication, it moves beyond the direct oversight of the design house, which makes it difficult to monitor and ensure the integrity of the manufacturing process. This lack of oversight creates potential risks such as the insertion of Hardware Trojans (HTs) [20], Reverse Engineering (RE) [21], IP piracy [22], overproduction [23], counterfeiting [24], and broader supply chain security concerns. Even after verification and distribution in the market, chips remain susceptible to end-user threats such as Fault Injection (FI) [25], probing [26], and microarchitectural side-channel attacks [27]. Moreover, geopolitical factors can make these challenges even more complicated, adding uncertainties to the supply chain dynamics.

HTs refer to malicious modifications done to the IC during its life cycle before distribution in the market without the knowledge of the original designer [28]. The intention behind incorporating HTs comes from diverse entities, including dissatisfied employees, malicious third-party contractors, state-sponsored actors, competitors, and cybercriminals. Each group operates with distinct motivations that span from data theft and sabotage to spying, financial gain, competitive advantage, and the pursuit of political or ideological goals [20, 29].

RE is another challenging threat to the security of ICs, and is referred to as the process of analyzing and understanding the design, functionality, and composition of ICs without permission from the owner. A malicious user or a rogue element in the foundry can extract the gate-level netlist of a design, analyze it, and comprehend the functionality of the IC

[30, 31, 32]. This extracted information can be later used for IP piracy, IC counterfeiting, or as a guide for HT insertion by the adversary.

Another risk to the security of the ICs is IP piracy [22]. This method allows for the illegal and unauthorized use, reproduction, distribution, or exploitation of the IP without permission or proper licensing from the rightful owner. Such actions undermine the originality and exclusivity of the design house's IPs and can potentially harm the financial investments and competitive advantage of the design house.

Overproduction is the act of manufacturing more IC units than what is contractually required [23]. This surplus production often occurs without the knowledge or authorization of the owner. Overproduction can lead to a variety of risks and challenges, including the infiltration of counterfeit or unauthorized components into the market, which may lack the rigorous quality controls of legitimate products.

IC counterfeiting refers to the illegal manufacturing and distribution of fake ICs which are designed to look and function like genuine electronic components [24]. These counterfeit ICs are produced using substandard or inferior materials to mimic the design and performance of legitimate ICs made by established semiconductor manufacturers. This unlawful practice poses significant risks to the electronics industry since fake ICs can be used in various electronic devices, potentially compromising their security, reliability, and performance.

Concerning end-user threats, FI and probing represent two prominent security risks capable of compromising the integrity and confidentiality of ICs. These attacks aim to exploit the physical or logical vulnerabilities of the IC, with the objective of extracting sensitive information, altering the circuit's behavior, or causing malfunctions [25, 26, 33].

FI involves intentionally introducing errors or malfunctions into the IC's operation by manipulating its environment, such as voltage, temperature, or clock signals [33]. Attackers can use various FI techniques, including laser, electromagnetic, and voltage glitching, to induce transient or permanent faults in the circuit. These faults can lead to unintended behavior, such as bypassing security mechanisms, corrupting data, or revealing cryptographic keys.

Probing, on the other hand, involves physically accessing the IC's internal signals or data using specialized equipment, such as microprobes or Focused Ion Beams (FIBs) [26]. Attackers can use probing to extract sensitive information, such as cryptographic keys or proprietary data, or to modify the circuit's behavior by tampering with its internal connections. Probing attacks can be particularly effective against ICs with weak physical security measures, such as inadequate shielding or encapsulation.

Moreover, microarchitectural attacks, such as Spectre [34] and Meltdown [35], are a class of security vulnerabilities that exploit the microarchitectural features of modern processors, such as speculative execution, out-of-order execution, and cache hierarchies. These attacks can enable unauthorized access to sensitive data, including passwords, encryption keys, and personal information, stored in the memory of a computer system.

Spectre is a vulnerability that exploits speculative execution, a technique used by modern processors to improve performance by executing instructions before they are known to be needed. Spectre can trick a processor into executing malicious code that indirectly accesses sensitive data from the memory of other applications or the operating system.

Meltdown is a vulnerability that exploits the out-of-order execution and cache hierarchies of modern processors. Meltdown can enable an attacker to bypass the memory isolation mechanisms of the operating system and access sensitive data from the memory of other applications or the kernel.

In addition to the mentioned threats, geopolitical factors and international tensions add layers of complexity to the risks associated with outsourcing IC fabrication. Changes in trade policies, geopolitical disputes, or disruptions in diplomatic relations between countries can impact the supply chain and manufacturing processes [36]. Such uncertainties may lead to delays in the delivery of fabricated ICs, disrupting production schedules and affecting the timely release of products to the market.

Supply chain security risks can become more severe due to the absence of direct control over the manufacturing process [37]. Semiconductor supply chains have a global reach and involve numerous suppliers and subcontractors. If any of these entities encounter a security breach, it can result in the insertion of malicious components or unauthorized alterations during the fabrication process. This can have significant implications, ranging from compromised functionality to potential security breaches.

Due to these persistent threats, the semiconductor industry faces staggering financial losses, amounting to several billions of dollars annually [38]. Once the chip is sent for fabrication, the loss of oversight underscores the importance of addressing and mitigating the inherent risks associated with outsourcing IC production. Hence, design houses must implement robust security measures to safeguard against the mentioned threats.

1.3 Countermeasures

Countermeasures against the introduced threats are crucial for maintaining the integrity and security of the chips. The most common classes of techniques against these threats include HT detection [39, 40], fingerprinting [41, 42], and Design for Hardware Trust (DfHT) techniques [43] including hardware obfuscation [44], cryptography [45], watermarking [46, 47, 48], and split manufacturing [49, 50, 51, 52].

In addressing the challenges HT poses, both HT detection and DfHT techniques can be used to protect ICs. HT detection methods aim to identify potential HTs after the chip has been delivered. They employ various approaches, categorized into destructive and non-destructive methods. Conversely, DfHT methods aim to prevent HT insertion during IC fabrication or to ease their detection. The main difference is that HT detection methods are applied after IC fabrication, while DfHT methods require considering security measures in the IC design before fabrication.

Regarding RE, IP piracy, and IC counterfeiting, many techniques have been introduced in prior art. It is worth noting that methods developed to tackle one of these issues often provide solutions to the others as well. One of these methods is IC fingerprinting [41, 42], which is a process used to uniquely identify and authenticate individual ICs or chips based on their inherent physical characteristics and minor variations that occur during the fabrication process. These variations, which are impossible to clone or replicate, derive from factors such as process imperfections, environmental conditions, and material properties.

Watermarking [46, 47, 48] is another measure against IP piracy and IC counterfeiting, and it involves the incorporation of a distinct signature into an IP core, constructing the watermarked IP in a manner that preserves its original functionality. The key characteristic is that this process does not alter the core's intended operation. Upon the completion of the chip fabrication process, the IP owner can preserve it and extract its signature using pre-defined activation parameters. This extraction serves as a means to validate the lawful utilization of their IP core within the SoC by comparing it with the initially embedded signature. It is crucial for watermarking to be easily embeddable and verifiable without imposing significant overhead or succumbing to potential attacks.

Hardware obfuscation, [44, 53, 54, 55, 56, 57], is another security technique that pro-

tects ICs mainly by preventing RE in electronic systems. It involves intentionally making the design, functionality, or inner workings of a hardware component difficult to understand or interpret, thereby increasing the complexity and time required to reverse engineer a design. This technique can be implemented through various methods, such as logic obfuscation, layout obfuscation, gate-level obfuscation, functional obfuscation, and key-based obfuscation. These methods aim to modify the circuit's logic gates, alter the physical layout, add redundant or dummy gates, hide the actual functionality, or implement a secret key within the hardware design. By employing these techniques, design houses can prevent potential attackers and safeguard their IPs. However, it is essential to recognize that obfuscation is not a foolproof method and may still be bypassed by skilled attackers with sufficient resources and time. Examples of obfuscations schemes that have been 'broken' are SARLock [58], Anti-SAT [59].

While countermeasures like logic locking aim to prevent IP piracy and IC overproduction, watermarking primarily concentrates on verifying the legality of IP use. In light of numerous copyright violation incidents over the past two decades [60], having a unique identifier for each IP is considered essential for claiming ownership. The distinctive signature provided by watermarking facilitates a robust mechanism for IP owners to assert their rights and authenticate the rightful use of their IP within complex SoCs.

Cryptography provides a means to protect sensitive data, such as encryption keys, passwords, and personal information, from unauthorized access, modification, or disclosure [45]. In the context of hardware security, cryptography is used to implement various security mechanisms, such as secure boot, secure storage, and secure communication, to ensure the integrity, confidentiality, and authenticity of the hardware and the data it processes.

While cryptography is fundamental for securing data and communications, it is insufficient as a standalone solution for hardware security. Physical access to hardware can enable attackers to bypass cryptographic protections through side-channel attacks, fault injection, or direct probing. Additionally, vulnerabilities in the implementation of cryptographic algorithms and poor key management practices can undermine the effectiveness of cryptographic measures.

Split manufacturing [49] is a technique used to protect IP and prevent RE during the fabrication phase. This method involves dividing the IC manufacturing process into multiple stages, each performed by a different foundry or fabrication facility. By doing so, no single foundry has access to the complete IC design, making it significantly more difficult for an unauthorized party to reverse engineer or counterfeit the IC.

In split manufacturing, the goal is to divide the design into front-end-of-line (FEOL) and back-end-of-line (BEOL) processes. The FEOL process involves the fabrication of transistors and other active devices, while the BEOL process focuses on the creation of interconnects and metal layers. By separating these processes, the foundry responsible for the FEOL process has access only to the transistor-level design, while the foundry handling the BEOL process receives a partially fabricated IC with no information about the underlying transistor structure. This division of information helps maintain the confidentiality of the IP and reduces the risk of unauthorized replication or reverse engineering.

However, IC split manufacturing also presents challenges, such as increased coordination and communication between foundries, potential yield loss due to process mismatches, and the need for compatible sizes of metal layers and vias between the FEOL and BEOL processes. Despite these challenges, IC split manufacturing is a promising approach for enhancing the security and protection of IPs in the semiconductor industry.

In recent years, there has been a growing focus on security closure within the hardware

security community [61, 62, 63, 64, 65, 66, 67, 68, 69, 70]. Security closure refers to the procedure of verifying that the security measures and countermeasures integrated into a hardware system align with the intended security objectives, while simultaneously managing the trade-offs between PPA limitations [67]. This process is similar to timing closure, which aims to verify that the system meets the desired timing specifications under PPA constraints.

To promote research and development in the area of security closure, several academic conferences and workshops have been organizing security-focused design contests. For example, the International Symposium on Physical Design (ISPD) has been organizing the ISPD Hardware Security Contest in recent years [70, 71]. The contest aims to provide a platform for researchers and practitioners to showcase their innovative solutions for hardware security and to evaluate the effectiveness of their solutions against a set of benchmark circuits and attack scenarios.

The ISPD Hardware Security Contest 2022 and 2023 have attracted a large number of submissions from around the world, highlighting the growing interest and importance of security closure in the hardware security community. The contest results and the research papers presented at the conferences provide valuable insights and guidance for the design and implementation of secure hardware systems [61, 62, 63, 64, 65, 66, 68, 69]. This thesis has been shaped by these contests: in 2022, as a contest participant, I have secured a third place award after a 4-way tie between the top teams. In 2023, I have organized the contest in collaboration with my supervisor and external collaborators from New York University.

In addition to the aforementioned techniques, there are alternative methods such as Assertion Based Verification (ABV) primarily employed for verification and dependability objectives [72]. For instance, ABV is utilized to ascertain whether the design is free of bugs or resilient against faults. Although not primarily intended for security purposes, the similarity in effects between faults and security threats, such as when an HT is activated, suggests that these approaches could be repurposed for security applications [73, 74].

In general, ABV is a widely used functional verification methodology that employs formal properties, known as assertions, to validate the correctness of a digital circuit design. As shown in Figure 4, ABV is an essential part of modern design and verification flows, as it helps to improve the quality and efficiency of the verification process by providing a systematic and automated way to check the design's functional behavior.

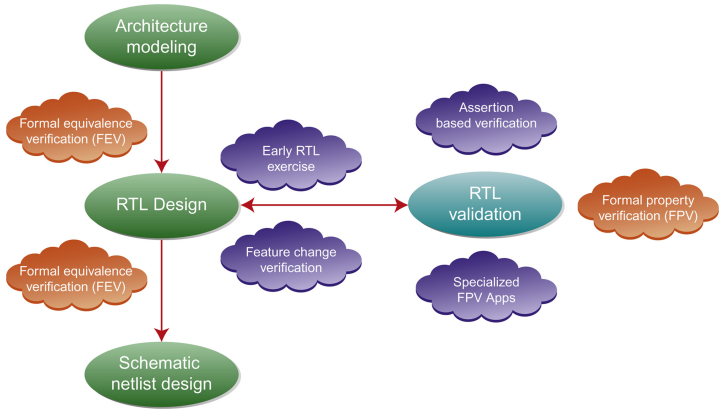


Figure 4: Different verification methods as part of modern design and verification flows (from [75]).

Assertions are formal statements written in a Hardware Description Language (HDL)

or a specialized assertion language, such as Property Specification Language (PSL) or SystemVerilog Assertion (SVA) [76]. They describe the expected behavior of a design under specific conditions, such as data dependencies, timing constraints, or protocol compliance. By embedding these assertions into the design or testbench, engineers can monitor the design's behavior during simulation and automatically detect any violations of the specified properties.

It is important to highlight that the mentioned countermeasures may exhibit efficacy against multiple threats. However, for the sake of brevity and clarity, the most common applications for each technique have been outlined.

1.4 Contributions and Outline of the Thesis

The core of this thesis is the investigation of methods and countermeasures against threats in the post-design stage, with particular emphasis on mitigating risks associated with HTs after the design phase. It involves several techniques in different phases of the IC design to enhance the security of ICs. Figure 5 presents the overall structure of the thesis.

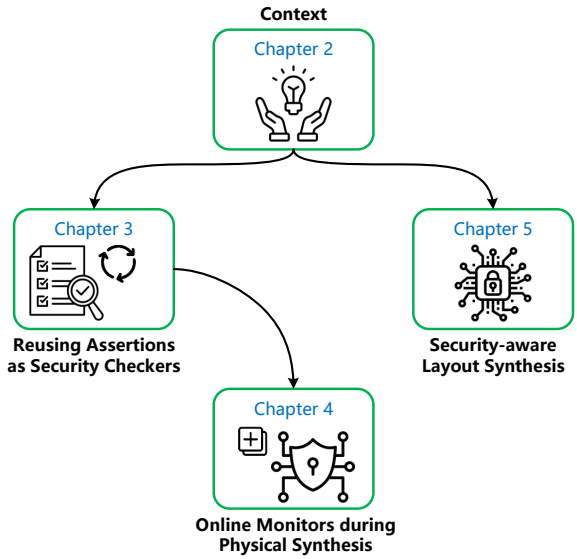


Figure 5: Structure of the thesis.

- **Chapter 2** This chapter offers a comprehensive exploration of advanced IC fabrication processes, along with an examination of various threats encountered throughout the IC life cycle and the latest countermeasures, with a specific emphasis on methods targeting HTs. It begins by providing an overview of the IC life cycle, detailing the steps involved from silicon to IC and encompassing aspects such as design and fabrication. Following this, the chapter explores the most critical security vulnerabilities of ICs, with a focus on HTs as a significant security concern. Finally, it surveys the existing countermeasures found in the current literature against HTs.

- **Chapter 3** This chapter introduces a novel approach to enhance the security of digital designs against HTs by repurposing verification assertions. The chapter explains how

assertions can be transformed into online monitors for efficient HT detection, and introduces a security metric and an assertion selection methodology. The chapter presents a comprehensive analysis of experimental results, demonstrating the adaptability and scalability of the method by applying over 100 assertions to a diverse set of IPs within the OpenTitan SoC. The chapter concludes by emphasizing the practicality and flexibility of the proposed detection solution, which is independent of the specific activation mechanisms of HTs, offering an adaptable security enhancement for digital designs.

- **Chapter 4** The main focus of this chapter is to present a comprehensive approach to enhancing IC security throughout the design process, particularly in the back-end stage. The chapter builds upon the previously introduced method for repurposing verification assertions as security checkers at the front-end phase of IC design. To further improve security, the chapter proposes a novel technique for incorporating online monitors during physical synthesis, providing an additional layer of protection at the back-end phase. The back-end flow can be considered as a complementary approach to the front-end method, but both techniques can also be employed independently, depending on user preferences and specific requirements, offering a more customizable and adaptable solution to enhance IC security.

- **Chapter 5** This chapter introduces Security-Aware Layout Synthesis (SALSy), a novel methodology for designing ICs with inherent security considerations, similar to the established practice of balancing PPA metrics and security, known as security closure. SALSy is a proactive strategy at the back-end phase that enhances IC security against fabrication-time and post-fabrication adversarial attacks, including HT insertion, FI, and probing. The methodology has been validated through a silicon-based demonstration, confirming its compatibility and effectiveness with a commercial Process Design Kit (PDK) and library. SALSy achieves this enhanced security with only a minimal impact on power consumption, thus maintaining a balanced trade-off between security and efficiency.

- **Chapter 6** As the last part of this thesis, this chapter serves as a summary of the main results and ideas presented in the study. It also suggests future research to improve IC security and the proposed methods, discussing possible ways for further development in the field.

2 Background

ICs have become a fundamental component of modern-day electronics, enabling the miniaturization and enhanced performance of various devices. The fabrication of ICs involves intricate processes that require advanced technology and precision engineering. This chapter provides a detailed overview of the complexities of IC fabrication and the efforts that have been made to enhance the security of these ICs against possible threats and attacks.

2.1 Life cycle of an Integrated Circuit

The life cycle of an IC involves several key stages. It begins with the design phase, where engineers create detailed schematics and layouts. This is followed by fabrication, where the IC is manufactured in a cleanroom environment using complex processes like photolithography. After fabrication, the IC undergoes testing and packaging to ensure it meets quality standards. Once packaged, the IC is distributed and integrated into electronic devices.

2.1.1 Design

The journey from idea to chips begins at the design stage, where the goal of fabricating the IC is established, and the relevant system specifications or requirements are set to meet user demands. Subsequently, engineers at the design house carefully define the implementation details of the chip. The design process is generally divided into two stages: front-end and back-end [77]. Each stage includes several steps that collectively contribute to the comprehensive design and fabrication of an IC.

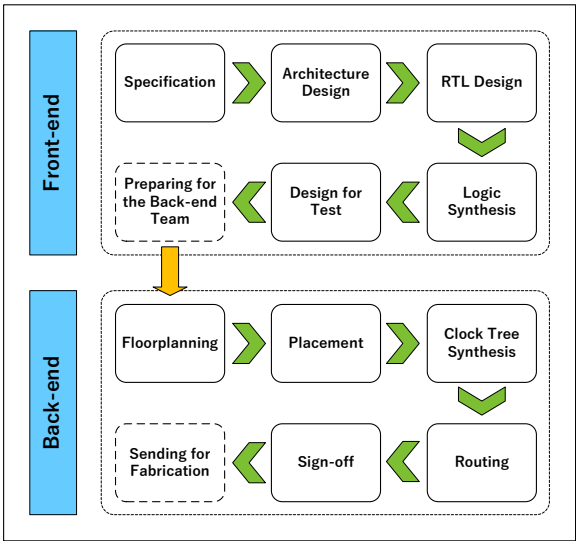


Figure 6: An overall view of different steps in IC design

As shown in Figure 6, each of the front-end and the back-end phases are divided into several main steps. In the following, more details are provided about each of the front-end and back-end procedures.

Front-end: The front-end phase focuses on the functional aspects of the IC. The primary goal of front-end design is to create a high-level description of the IC's behavior that meets the desired Power, Performance, and Area (PPA) requirements. The outcome of the front-

end phase is a gate-level netlist, which is passed to the back-end engineers for further implementation steps, and it involves the following steps:

- Specification: The first step in front-end design is to define the functional requirements, performance goals, and constraints of the IC. Moreover, the number of the pins and the packaging type is considered in this step. These specifications serve as a guideline for the subsequent design stages and ensure that the final product meets the intended purpose. At this stage, it is crucial to consider the interaction between hardware and software components to ensure that the IC is optimized for both hardware functionality and software compatibility, leading to a more efficient and effective design.

- Architecture Design: Based on the specifications, a high-level architecture is developed to meet the PPA requirements. This step involves defining the overall structure, organization, and functionality of the system or chip being developed. Additionally, designing interface and communication is performed in this step, which defines the interfaces between modules and creates the communication protocols and data transfer mechanisms, including the selection of interconnect protocols and the definition of buses. Domain separation between digital and analog components is also addressed, along with the establishment of clock domains and power domains to ensure proper operation and power management.

- Register-Transfer Level (RTL) Design: The IC's functionality is described using an HDL, such as Verilog or VHDL, at the RTL. This level of abstraction allows designers to focus on the data flow and control logic without worrying about the underlying gate-level implementation.

- Logic Synthesis: The RTL design is converted into a gate-level netlist, which consists of logic gates and their interconnections. During logic synthesis, the design is optimized for PPA metrics considering the target technology and fabrication process.

- Design for Test (DFT): Test structures and methodologies, such as scan chains, are implemented in this step to facilitate testing and debugging of the fabricated IC. This step involves a set of techniques to measure the reliability of the IC.

Back-end: Back-end design, also known as physical design, focuses on the physical implementation of the IC. This stage involves the use of Computer-Aided Design (CAD) tools to develop a digital representation of the IC [78]. It begins with transforming the gate-level netlist into a layout that meets the geometric and electrical constraints imposed by the fabrication process. The primary goal of back-end design is to create a manufacturable IC layout with optimal performance, power, and area. The outcome of the design phase is a GDS file, which contains all the necessary information for the subsequent fabrication stages. The back-end design stage includes the following main steps:

- Floorplanning: In this step, the overall outline of the IC is defined, including the placement of major blocks, Input/Output (I/O) pads, and power distribution networks. This step sets the foundation for the subsequent placement and routing stages.

- Placement: The exact location of standard cells (pre-designed logic gates) and macro blocks (larger functional units, such as memory or processors) within the IC layout is determined during placement. The primary objective of placement is to minimize the interconnect length and optimize the IC's performance, power, and area.

- Clock Tree Synthesis (CTS): A balanced clock distribution network is generated at this step to ensure that clock signals reach all parts of the IC with optimal skew and delay. This step is critical for synchronizing the IC's operation and optimizing its performance.

- Routing: Once the CTS is completed, the interconnections between standard cells and macro blocks are created using metal layers. Routing must adhere to design rules and optimize for signal integrity, power, and performance.

- Sign-off: Final checks and analyses, such as Static Timing Analysis (STA), power analysis, and reliability analysis, are performed during sign-off to ensure that the design meets all specifications and is ready for fabrication.

2.1.2 Fabrication

Silicon is the most commonly used material for IC fabrication due to its abundant availability, chemical stability, and semiconductor properties. The use of silicon as a base material can be traced back to the invention of the transistor in 1947 by John Bardeen, Walter Brattain, and William Shockley [79]. In the subsequent years, the development of silicon-based ICs revolutionized the electronics industry, leading to the rapid advancement of technology.

The fabrication of ICs is a complex process that involves several stages, including wafer preparation, photolithography, etching, doping, metallization, and dicing. Each stage is critical in ensuring the overall performance and reliability of the final product. An overview of IC fabrication flow is illustrated in Figure 7.

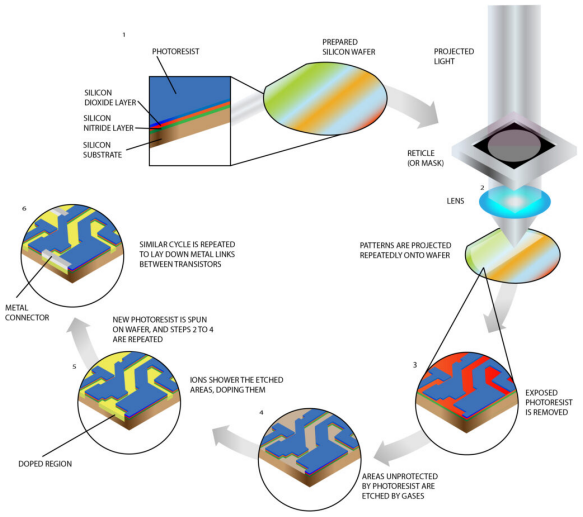


Figure 7: Overall view of IC fabrication flow (from [80])

Wafer Preparation: In order for silicon to be suitable for use in computer chips, it undergoes a rigorous purification process to achieve a purity level of less than one foreign atom per billion atoms. The purified silicon is first melted and then pulled to create a solid, resulting in a single, continuous, and unbroken crystal lattice structure in the form of a cylindrical ingot.

As depicted in Figure 8, this ingot is subsequently sliced into thin, circular surfaces called *wafers*. Engineers use these silicon wafers as the substrate for semiconductor devices.

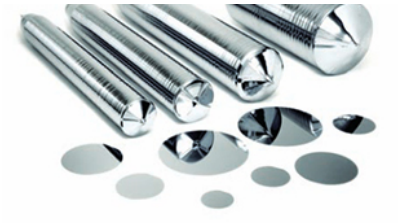


Figure 8: Silicon ingots (top) and wafers (bottom) of different diameters (from [81])

Photolithography: Photolithography is a process used to transfer the circuit pattern from the design phase onto the silicon wafer. The principle of photolithography hinges on replicating a structure delineated on a lithographic mask onto a light-sensitive resist, previously coated on a substrate. This process offers two options: utilizing either positive resist or negative resist. The procedure of using positive resist involves the following steps:

- Cleaning and Deposition of Metal Film: The silicon wafer is thoroughly cleaned to remove any impurities or contaminants that may affect the fabrication process. Afterward, a metal film is deposited on the substrate.
- Coating: The wafer is coated with a light-sensitive material called photoresist, which undergoes chemical changes when exposed to UltraViolet (UV) light. This coating is commonly heated for 30 minutes between 60 and 100 °C.
- Exposure: The circuit pattern is projected onto the photoresist-coated wafer using a mask aligner, which exposes the photoresist to UV light in the desired pattern. Thus, the structure of the mask is imaged on the resist and causes photochemical changes therein.
- Development: The exposed photoresist is chemically developed, resulting in a patterned layer that serves as a template for the subsequent etching and doping processes.

Etching: Etching is the process of selectively removing layers of material from the silicon wafer to create the desired circuit structure. This process can be performed using wet chemical etching or dry etching techniques, such as plasma etching or Reactive Ion Etching (RIE) [82]. The choice of etching technique depends on the specific material being etched and the desired level of precision.

Doping: Doping is the process of intentionally introducing impurities, known as dopants, into the silicon wafer to alter its electrical properties. During this process, high-energy ions are accelerated and implanted into the wafer, altering its conductivity and creating regions of n-type or p-type semiconductor material essential for device functionality. Common dopants include boron, phosphorus, and arsenic, which are used to create p-type and n-type semiconductors [83]. Doping can be achieved through various techniques, such as diffusion, ion implantation, and epitaxy.

Metallization: Metallization is a process that involves depositing metal layers onto the surface of a silicon wafer to create interconnects and contact pads, which are crucial for linking various components of an IC. Techniques such as sputtering or electroplating are employed to deposit metals like aluminum, copper, or tungsten onto the wafer.

Dicing: Once the fabrication of individual ICs on the wafer is complete, the wafer undergoes dicing to be cut into individual chips. Dicing is typically performed using specialized

cutting tools, such as diamond saws or laser systems, to precisely cut along predefined lines, known as scribe lines, that separate the chips. The separated chips are then picked up and forwarded to the packaging facilities for further testing and integration into electronic systems.

2.1.3 Testing and Packaging

After the fabrication process is completed, the chips are subjected to electrical testing to verify the correct functioning. Once the functional dies have been identified, they are separated from the wafer and packaged as individual semiconductor devices. The final stage in IC fabrication is packaging, which involves encapsulating the silicon chip in a protective casing and connecting it to external components. Packaging helps to ensure the mechanical stability, electrical performance, and thermal management of the IC [84].

2.1.4 Distribution

After successful packaging and testing, the chips are shipped to distributors, who act as intermediaries between the manufacturers and Original Equipment Manufacturers (OEMs). Distributors often maintain relationships with multiple manufacturers, which enables them to offer a diverse portfolio of chip products to answer the varying needs of OEMs. Finally, OEMs integrate these packaged chips into their products, such as computers, smartphones, and other electronic devices, which are ultimately sold to end-users through retail channels.

2.2 Security Vulnerabilities of Integrated Circuits

As previously mentioned, the process of bringing an IC to market involves numerous complex stages, often spanning multiple countries for various design, fabrication, packaging, and testing phases. At each stage, various threats can compromise the security of the ICs. Although a rogue engineer could manipulate the IC during different design stages, design houses typically have good control over their staff and can prevent such irregularities to a large extent unless they use infected 3PIP [85]. Attacks during the testing and packaging phases primarily involve false test report generation and the use of lower-quality materials in packaging and bonding. Furthermore, these attacks are more likely to be discovered by the design house upon IC delivery.

The threats persist even after the chip is distributed in the market, with attackers potentially attempting to reverse engineer the chip for various purposes. However, a significant threat lies in the foundry, where IC fabrication takes place, and where there is no control over what happens to the chip. This is particularly concerning because most recognized threats to IC security, such as HT insertion, RE, IP piracy, and IC counterfeiting, can occur individually or simultaneously during the fabrication process. Consequently, this thesis focuses on threats in the post-design stages, especially those within the foundry.

Among all the threats in the fabrication phase, HT insertion is more likely to remain undetected due to its stealthy nature and has received significant attention in recent years within academia. It is worth noting that implementing techniques to countermeasure HTs may also be effective against other security attacks. Further details about this will be provided in the following sections.

2.2.1 Hardware Trojan as a Major Security Risk

HTs are malicious modifications or additions that are intentionally made to the design, layout, or functionality of an IC in order to compromise the security, reliability, or performance of electronic systems. These modifications are typically inserted during the design and fabrication stages of the IC's lifecycle, often without the knowledge of the

original designers. An HT imposes a significant threat to any hardware design intended for deployment in critical operations.

HTs are typically composed of *trigger* and *payload* parts. The trigger is an optional part of the HT and is referred to as a particular input sequence, temperature, or voltage level at which the HT can be activated. If an HT does not include the trigger, it is called an "always-on" HT. However, the trigger part is typically designed to only activate in extremely rare conditions, in order to make the detection of HT difficult.

The payload is the part of an HT that is responsible for carrying out the malicious action once the trigger condition is satisfied. This action can result in various forms of harm, including data theft, denial of service, or unauthorized access to the system. As long as the trigger condition is not met, the circuit operates normally, making it difficult to detect the presence of the HT. When the payload becomes activated, the malicious behavior is executed. This stealthy nature of HTs, where they remain dormant until the payload is activated, makes their detection particularly challenging.

A malicious foundry can incorporate three categories of HTs into an IC layout: additive, substitution, and subtractive [86]. Additive HTs involve introducing additional circuit components and/or wiring into an existing design. Substitution HTs necessitate the removal of logic to accommodate extra HT circuit components and/or wiring within an existing circuit design. Lastly, subtractive HTs include the removal of circuit components and/or wiring to modify the behavior of an existing circuit design. This thesis focuses on evaluating the vulnerability of a circuit layout to additive HT attacks due to their significant impact on system behavior, their detectability through changes in different characteristics of the design, and the extensive body of research that provides a solid foundation for further study.

Figure 9 presents an HT taxonomy based on the trigger and payload types of additive HTs. The triggers can be created by adding, modifying, or removing hardware components within an IC and can be either digital or analog. An ideal trigger for an HT is identified by its key characteristics, which include a small size that requires minimal additional circuit components, stealth that demands rare events for activation, and controllability that enables easy activation by attackers but not by defenders or during normal operation. Previously demonstrated triggers exhibit a range of characteristics, from large and stealthy (requiring many additional gates) to small and easily triggered. Sophisticated HTs are typically small, stealthy, and controllable.

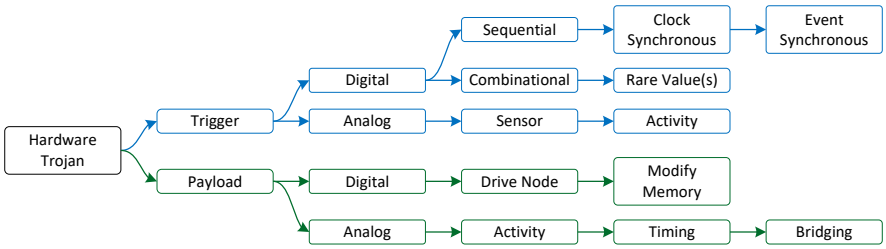


Figure 9: HT Taxonomy based on trigger and payload types (from [87])

On the other hand, both analog and digital payloads exist, with various effects such as information leakage, alteration of the IC's internal state, or rendering the system unusable

through a denial-of-service attack. Regardless of the effect, the payload mechanism must establish a connection to or near a target, which can be a *security-critical* component within the IC design.

HTs pose a substantial security threat due to their complex nature and the challenges they present in detection. Traditional testing and verification methods, such as functional verification and design rule checking, are often inadequate for detecting HTs. This is because HTs are designed to be stealthy, remaining dormant until activated by a specific, often complex, trigger. The relatively simple test vectors used in conventional testing are unlikely to activate these triggers.

Moreover, HTs are designed to make only subtle changes to the IC's behavior, which can be difficult to detect using traditional fault models. These models are intended to identify accidental faults, such as manufacturing defects, and are not equipped to detect intentional, malicious modifications like HTs.

The task of detecting HTs requires a unique set of test vectors that can activate the target fault. This can be a challenging and time-consuming process, particularly for cyclic sequential designs. Furthermore, the increasing complexity and size of modern ICs exacerbate the detection process, making it even more difficult to identify and isolate these malicious modifications. Therefore, while test vectors and fault models are essential tools in IC testing, they may not be sufficient for detecting HTs, and more advanced, specialized methods are needed.

Another issue is that HTs can be designed to be stealthy and adaptable, making them suitable for various malicious activities. For example, an HT can be programmed to leak sensitive information, disable critical system functions, or create a backdoor for unauthorized access, depending on the attacker's objectives.

Furthermore, since ICs are used in various electronic systems, a single compromised IC can have far-reaching consequences. For instance, an HT in a widely used microprocessor could affect millions of devices, leading to significant security and privacy breaches.

Lastly, the globalized nature of the semiconductor industry increases the risk of unauthorized access to the IC design or manufacturing process, making it easier for adversaries to insert HTs. This global supply chain presents numerous vulnerabilities that attackers can exploit and highlights the need for robust security measures throughout the IC life cycle. Due to these rising concerns, protecting measures should be added to ICs before sending them for fabrication in order to mitigate the risks posed by HTs.

2.3 Countermeasures against Hardware Trojans

As mentioned in the previous section, HTs pose a significant threat to ICs, as they are embedded at the hardware level, which makes software-level countermeasures inadequate for addressing the risks they present. The detection of HTs in hardware designs is a complex task, primarily due to the absence of a golden version or a known-good reference for comparison during the verification process.

In principle, an effective method for detecting an HT would be to activate it and observe its effects. However, this approach is challenging, as an HT's type, size, and location are generally unknown, and its activation is likely to be a rare event. Consequently, an HT can remain hidden during the normal operation of the chip and only becomes active when the specific triggering condition is met. This stealthy nature of HTs necessitates the development of advanced detection and mitigation strategies at the hardware level to ensure the security and reliability of ICs.

To minimize the risk of HTs, researchers have been exploring different methods in recent years. These methods are mainly classified into DfHT techniques and HT detection

techniques, which are approaches used to enhance the security and reliability of ICs. An overview of these techniques is shown in Figure 10.

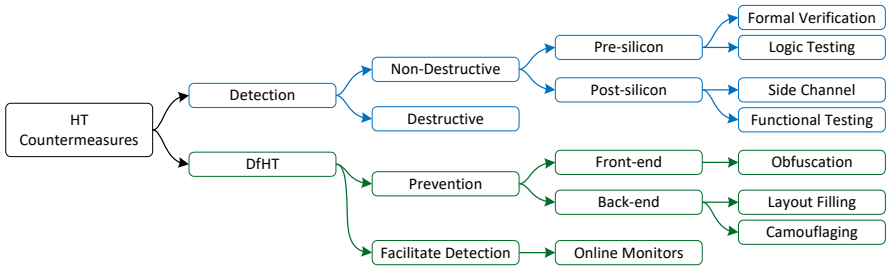


Figure 10: An overview of different protection methods against HT (adopted from [88])

2.3.1 Detection Techniques

HT detection is the most widely adopted approach by researchers to tackle HTs [89]. The primary goal of this method is to verify the integrity of existing designs and fabricated ICs without requiring additional circuitry. These techniques are categorized into destructive and non-destructive methods.

Destructive techniques typically involve reverse engineering the IC by depackaging and obtaining images of each layer to reconstruct the golden behavior of the chip. This approach has the potential to provide very high assurance that any malicious modification in the IC can be detected. However, it comes with significant drawbacks, such as high cost and time consumption, taking several weeks or months for an IC of reasonable complexity. Furthermore, at the end of this invasive process, the IC cannot be used, and the information obtained is limited to a single IC sample. An overview of the process of delayering an IC is presented in Figure 11.

It is important to note that reverse engineering modern complex chips is a labor-intensive and error-prone task. Obtaining the entire chip structure through RE may necessitate the use of tens of ICs, as depackaging and delayering procedures can introduce unintended errors in the RE process. Nonetheless, employing destructive RE on a limited number of samples may be appealing for acquiring the characteristics of a golden batch of SoCs. However, destructive method is proven to be the only effective approach among HT detection techniques in practice.

On the other hand, non-destructive methods, as their name indicates, aim to detect HTs without causing harm to the IC. Some of these methods are performed in the pre-silicon stage, where the design has not yet been sent for fabrication. These techniques are primarily used to validate 3PIPs purchased from third-party vendors [91, 92]. In this stage, the design house retains control over the circuit and has the ability to simulate and observe internal signals to detect potential malicious behavior. The main existing pre-silicon detection methods include formal verification and logic testing.

Formal verification methods [91, 92, 93, 94, 95, 96, 97, 98, 99] involve creating mathematical models of the circuit design and its specifications, and then using automated tools, such as theorem provers or model checkers, to exhaustively analyze the design for any discrepancies or violations of the specified properties. This process can help detect HTs, as well as other design errors or vulnerabilities in 3PIPs, that might be missed by traditional simulation-based testing methods.

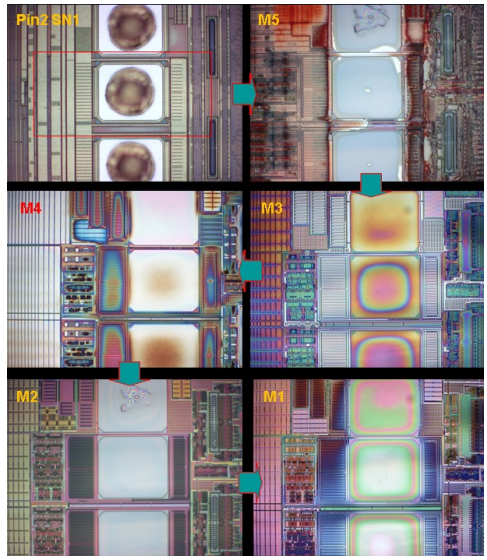


Figure 11: The process of delayering an IC by removing each layer of die (from [90])

Some common formal verification techniques used in HT detection include property checking [93], equivalence checking [94], model checking [95], and information flow [96]. While traditionally applied to software systems to uncover security bugs and enhance test coverage, these methods have also proven effective in verifying the trustworthiness of 3PIP [91, 92].

Property checking involves verifying whether a given circuit design satisfies specific safety or security properties, such as the absence of unauthorized information flow or the presence of proper access control mechanisms. Equivalence checking compares the original circuit design with a trusted version or a higher level of abstraction to ensure they exhibit the same functionality, helping identify any malicious modifications. Model checking explores all possible states of a circuit design to verify that it adheres to the specified properties and does not contain any unintended behavior. Information flow analysis is used to analyze and track the flow of sensitive data within a digital circuit design. The primary goal of information flow analysis is to ensure that confidential information does not leak to unauthorized parts of the circuit or external entities, which could be a result of malicious modifications or HTs.

In [97], a model-checking technique is introduced to formally verify the presence of malicious modifications in 3PIP caused by HTs. This method is based on the concept of Bounded Model Checking (BMC). BMC generates reports detailing the sequence of input patterns that violate specific defined properties. The main advantage of this approach is its feasibility to extract the triggering condition of the HT from these reported input patterns. Another approach, as presented in [98], focuses on formally verifying unauthorized information leakage in 3PIPs. However, due to the challenge of space explosion, these approaches are constrained by the limited processing capability of model checking. While these techniques offer a promising solution for HT detection, they each encounter specific challenges and limitations [99].

Another category of pre-silicon approaches is logic testing [100, 101, 102] , which focuses on analyzing the functional behavior of a digital circuit design to identify potential malicious modifications. In this approach, test patterns or vectors are applied to the circuit's

inputs, and the corresponding outputs are observed and compared with the expected results. The primary objective of logic testing is to activate any hidden or rare trigger conditions associated with HTs, thereby exposing their malicious payloads. While some works distinguish logic testing from functional analysis based on the type of input patterns used (specific patterns for logic testing and random patterns for functional analysis), both methods share the same fundamental concept of applying inputs and observing outputs. Consequently, this thesis considers them as part of the same category.

A technique is introduced in [100], which identifies suspicious nets with weak input-to-output dependency. This approach is based on the observation that an HT is triggered under extremely rare conditions, which assumes the logic implementing the trigger circuit nearly unused or inactive during normal operation. The authors proposed a metric to find those nearly unused logic by quantifying the degree of controllability of each input net on its output function. This metric is computed by applying random input patterns and measuring the number of output transitions. If the threshold for a net is lower than a predefined one, it is flagged as suspicious. However, this technique has significant limitations, such as producing a large number of false-positive results and not providing any method to verify if the suspicious signals perform malicious operations. Another approach presented in [101] demonstrates how to design HTs that can evade [100] by distributing the trigger vector over multiple clock cycles.

The authors of [102] presented a technique to identify potential triggering inputs of an HT. The proposed technique is based on the observation that input ports of the triggering circuit of an HT remain inactive during normal operation. It performs functional simulation of the IP with random input patterns and traces the activation history of the input ports in the form of Sum-Of-Product (SOP) and Product-Of-Sum (POS). It then identifies redundant inputs by analyzing the SOPs and POSs, which are unactivated during functional simulation. These redundant input signals are potential triggering inputs of a HT. However, this method also produces a large number of false-positive results due to incomplete functional simulation and unactivated entries belonging to normal operations.

Pre-silicon techniques can be used effectively in many applications. However, the main challenge in HT detection arises when HTs are inserted during the fabrication process. In this scenario, the design house receives only the fabricated chip, making it impossible to observe all internal signals.

Post-silicon HT detection schemes are employed after the chip fabrication process. As depicted in Figure 10, these techniques can be categorized into two main classes: side channel and functional testing.

Side-channel analysis approaches [103, 104] aim to detect HTs by measuring various circuit parameters, such as delay, power (static and dynamic), temperature, and electromagnetic radiation. These methods exploit the side effects caused by additional circuits or activities resulting from HT trigger/payload activation. However, most detection techniques rely on the availability of “golden ICs” (HT-free ICs) for comparison to identify HT-infected ICs.

The authors in [103] demonstrate the use of side-channel profiles such as power consumption and electromagnetic emanation for HT detection. They generate power signature profiles from a small set of ICs randomly selected from a batch of manufactured ICs, which serve as golden chips. After profiling, the golden chips undergo rigid destructive RE to compare them against the original design. If found to be HT-free, the ICs are accepted as genuine, and their profiles serve as power templates. The remaining ICs are then tested efficiently and non-destructively by applying the same stimuli and building their power profiles. These profiles are compared using statistical techniques, such as principal component

analysis, against the templates obtained from the golden chips.

While side-channel analysis methods may achieve some success in detecting HTs, they face challenges in providing high coverage for every gate or net and extracting the abnormal side-channel signals of HTs in the presence of process and environmental variations. As IC feature sizes shrink and the number of transistors increases, growing process variations can easily mask the small side-channel signals induced by low-overhead and rarely triggered HTs. The authors in [104] proposed a backside imaging method based on filler cell patterns in the IC layout, which does not require a golden chip. However, the comparison between simulated and measured optical images still suffers from manufacturing process variations. Additionally, capturing clear images at higher resolutions is time-consuming.

Functional testing techniques [105, 106] aim to activate HTs by applying test vectors and comparing the responses with the correct results. The effectiveness of these techniques relies on the availability of a golden response. Although this approach may seem similar to manufacturing tests used for detecting manufacturing defects, conventional manufacturing tests using functional, structural, or random patterns have limited success in detecting HTs [107]. Skilled adversaries can design HTs that are activated under extremely rare conditions, allowing them to evade detection during the manufacturing test process.

To address this challenge, researchers in [105] and [106] have developed test pattern generation methods to trigger rarely activated nets and improve the probability of observing HT effects from primary outputs. However, due to the vast number of logical conditions in a circuit, it is impractical to enumerate all conditions of a real design. Moreover, HTs that transmit information via nonfunctional means, such as through an antenna or by modifying the specification, can evade detection by functional tests [108]. These limitations highlight the need for more advanced and sophisticated functional test techniques to effectively detect HTs.

In summary, destructive methods involve physically dissecting or altering the chip to analyze its internal structure, while nondestructive methods rely on non-invasive techniques to inspect the chip's functionality and behavior without causing any physical damage. Both approaches have their advantages and limitations, and their selection depends on various factors, including the specific application, available resources, and desired level of detection accuracy.

2.3.2 Design for Hardware Trust

As mentioned in the previous section, detecting a smartly-designed HT with a small size remains a significant challenge using existing techniques. As depicted in Figure 10, DfHT approaches aim to address this issue by incorporating additional logic to either facilitate the detection of HTs [109, 110, 111] or to prevent an adversary from inserting an HT in the first place [91, 92, 95, 96, 104, 112, 113]. Although it is impossible to achieve complete prevention against HT insertion in practice, research efforts have concentrated on limiting available chip resources to make it extremely difficult for adversaries to exploit them for the insertion of malicious logic [91, 92, 95, 96].

To facilitate HT detection, prior works try to add different redundancies to the design before sending it for fabrication in order to verify if the IC is HT-free after it is delivered. One of the most common redundancies is adding **online monitors** to the design. Online monitoring [114, 115, 116, 117, 118] is an effective approach to increase trust in hardware systems concerning HT attacks, as triggering all types and sizes of HTs during pre-silicon and post-silicon tests is exceptionally challenging.

These techniques have been widely employed for enhancing reliability and dependability, with a focus on Concurrent Error Detection (CED) methods. CED techniques introduce

redundancy through parity codes or hardware duplication and incorporate a dedicated checker to identify any discrepancies or errors [114]. Moreover, these techniques can be repurposed to identify unwanted changes caused by HTs [115, 116]. The authors in [119] and [116] have proposed leveraging a diverse range of 3PIP vendors to mitigate the impact of HTs. The work presented in [119] suggests verifying the integrity of a design by comparing multiple 3PIPs with an alternative untrusted design that serves a similar function. Meanwhile, [116] utilizes some constraints to prevent collusion among 3PIPs sourced from the same vendor to enhance the security of the design.

Another group of approaches utilizes a distributed software scheduling protocol to establish a trustworthy system resilient to HT activation in a multicore processor [117, 118]. These methods seek to mitigate the impact of HTs by coordinating the allocation and execution of tasks across multiple cores.

Furthermore, online monitoring techniques can utilize existing or supplemental on-chip structures to monitor chip behaviors [120, 121] or operating conditions, such as transient power [122, 123] and temperature [110]. Upon detection of any irregularities, the chip can be disabled or bypassed to ensure reliable operation, even though with some performance overhead. The authors in [124] propose a design for an on-chip analog neural network that can be trained to distinguish trusted from untrusted circuit functionality based on measurements obtained via on-chip measurement acquisition sensors.

The second category of DfHT techniques comprises HT prevention methods. As the name suggests, these approaches aim to prevent HT insertion by employing various techniques at different stages of IC design. In order to insert targeted HT, attackers usually need to gain an understanding of the design's functionality first. Since HT attacks are mostly performed in a location other than the design house, the attacker typically accomplishes HT insertion by reverse engineering the circuit to identify its intended functionality.

The process of reverse engineering a circuit can be both time-consuming and complex, as it requires analyzing the circuit's structure and behavior to understand its purpose and potential vulnerabilities. However, since an attacker has access to advanced CAD tools and the same PDK used by the design house, the threat of reverse engineering the design and, consequently, HT insertion remains. Therefore, preventive techniques are used to make the insertion of HTs as difficult as possible and minimize the risk of successful attacks. Depending on the implementation phase where the defensive technique is applied, these approaches can be further classified into front-end and back-end techniques.

Front-end engineers may employ various obfuscation techniques, such as logic locking [125, 126], to safeguard the IP of the design. The primary goal of logic obfuscation is to hide the original functionality of a design by incorporating several locking schemes into the original design. These locking circuits reveal the correct function only when the correct key is applied. This can make it more difficult for attackers to insert HTs without knowing the right input vectors.

Combinational logic obfuscation can be achieved by using XOR/XNOR gates at specific locations in a design [127]. In sequential logic obfuscation, additional states are introduced to a finite state machine to hide its functional states [106]. Some techniques also suggest the insertion of reconfigurable logic for logic obfuscation [128, 129]. The design operates correctly only when the reconfigurable circuits are correctly configured by the design house or end-user.

On the other hand, the focus of the back-end phase is to improve the security of the layouts during the physical synthesis stage. The HT prevention techniques in the back-end phase include layout filling [130, 131, 132] and camouflaging [133, 134, 135, 136, 137].

Layout filling is a technique aimed at restricting available resources, such as gaps and

free routing tracks, to prevent adversaries from inserting malicious logic [130, 131, 132]. During the back-end phase, CAD tools are often unable to completely fill the area with regular standard cells. Consequently, unused spaces are typically occupied by filler cells or decap cells, which serve no functional purpose. The main purpose of filler cells is to occupy the empty spaces in the layout, which can help to improve the overall density and reduce the risk of manufacturing defects, while decaps are utilized to manage peak current in the chip, particularly in areas with significant instantaneous power.

As a result, one method for attackers to insert HTs into a circuit layout involves replacing filler cells and, to some extent, decaps. Removing these nonfunctional cells minimally impacts electrical parameters, causing the presence of maliciously inserted cells challenging to detect. By occupying these spaces with functional cells, dummy vias, or other design elements, the attacker's ability to exploit the layout for inserting HTs is significantly reduced. This approach makes it more challenging for an attacker to find suitable locations to insert malicious logic without being detected or causing functional issues in the design.

An approach called BISA is presented in [130], and it involves filling empty spaces with functional filler cells during layout design. Subsequently, these cells are interconnected to establish combinational circuitry that can be tested later. Any failure detected during subsequent testing indicates that a functional filler has been substituted by a potential HT. The general BISA insertion flow includes preprocessing (gathering detailed information about the standard cell library), identifying unused spaces, placing BISA cells, and routing BISA cells.

Camouflaging [133, 134, 135, 136, 137] is a layout-level obfuscation technique that aims to create indistinguishable layouts for different gates by adding dummy contacts and faking connections between the layers within a camouflaged logic gate. It involves replacing standard logic gates with functionally equivalent but visually different gates to make it challenging to identify the actual logic function implemented in the design. By camouflaging the gates, attackers are hindered from extracting a correct gate-level netlist of a circuit from the layout through imaging different layers. As a result, the original design is protected from the insertion of targeted HTs. The authors in [138] employed a similar dummy contact approach and developed a set of camouflaging cells. These camouflaging cells can further enhance the security of the design by making it more difficult for attackers to identify the actual functionality of the gates.

3 Reusing Verification Assertions for Security Purposes

This chapter presents a novel approach to improve the security of digital designs by reusing verification assertions, particularly in the context of HT detection. It demonstrates that by transforming existing verification assets, one can create efficient security mechanisms capable of detecting HTs. The process by which assertions are leveraged as online monitors is explained, and a security metric alongside an assertion selection methodology employing the advanced capabilities of the Cadence JasperGold Security Path Verification (SPV) tool [139] is introduced.

Moreover, this chapter presents a comprehensive analysis of experimental outcomes, by applying over 100 assertions to a diverse collection of IPs within the OpenTitan SoC [140]. This demonstrates the presented method's adaptability and scalability to circuits of industry-relevant sizes. The chapter concludes by proving the practicality of this detection solution, emphasizing its independence from the specific activation mechanisms of HTs, thereby offering an adaptable security enhancement to digital designs [73].

As discussed in Section 2.3.1, several approaches exist to facilitate HT detection, with online monitors being the most common technique. Online monitoring techniques rely on embedding checker circuits in different locations of the design to catch unwanted behavior. One approach for building these checker circuits is using assertions, which describe the expected behavior of the circuit and help detect deviations between the intent and actual behavior. However, although online monitoring techniques offer high detection coverage, they impose significant overheads on the circuit. Recent efforts have aimed to decrease these overheads while maintaining maximum detection coverage, but the trade-off remains unfavorable [141].

In this chapter, I propose a methodology for selecting and reusing assertions written by verification engineers for functional verification purposes to achieve security goals, such as HT detection. This approach leverages existing design knowledge, which is often underutilized after the verification process. A new metric called **Security Coverage (SC)** is presented to evaluate the efficiency of online checkers in detecting HTs while considering the imposed overhead on the circuit. This metric helps automate the removal of assertions that are not helpful for HT detection and eliminates the need for detailed knowledge about the design for the engineer who is responsible for ensuring the security of the circuit.

3.1 Assertions as Hardware Trojan Detectors

This section explores the possibility of reusing assertions to detect HTs. To investigate this, the `B19-T500` benchmark from Trust-Hub [142] is selected, which is a Trojan-inserted version of the `B19` circuit from the `ITC'99` benchmark suite [143]. The Trust-Hub benchmarks, with their small sizes and rare triggering conditions, provide a suitable starting point for validating the effectiveness of HT detection schemes [142]. Consequently, these HTs remain hidden during standard verification checks [144]. In my initial study, I utilize these benchmarks to explain and evaluate my proposed approach. However, the goal is to extend this concept and apply it to more complex and realistic circuits.

The ideal characteristics for an assertion to be considered an effective security checker are:

1. It has a minimal overhead on the circuit once synthesized, as many assertions may be required in complex designs for high detection coverage.

2. It has a broad scope that captures high-level behavior, rather than focusing on local signals.

The assertions that satisfy the above conditions are referred to as **top-level assertions**. To illustrate this concept, a set of assertions that satisfy these conditions have been manually written for detecting the HTs inside `B19_T500`. The `B19` benchmark consists of four copies of the `Viper` processor, with an HT circuit embedded within each processor. The HT is triggered by a counter that counts specific vectors and resets with other specific input vectors. If the counter value falls between `3'b100` and `3'b110`, the HT is activated. The payload of the HT manipulates the bits of the Instruction Register (IR) of the embedded `Viper` processor, thereby altering the functionality of the circuit [142].

Although the easiest way to detect this embedded HT would be to write assertions to check the IR bits directly, this approach is not practical for two reasons: First, in a realistic scenario, the HT locations are unknown. Second, this style of assertion writing does not describe any system-level behavior, violating the second condition of being a good security checker. Moreover, as the defender is unable to anticipate the specific trigger for an HT, writing an assertion such as "`(IR == 3'b110 => alert)`" is not a practical solution.

Table 1 presents the top-level assertions considered for detecting HTs in the `B19_T500` circuit, written in PSL. These assertions check the correctness of transactions between the memory and processor by comparing the contents of the IR with the Store Operation (OP_STORE) and the Read Operation (OP_READ) instructions. The first two assertions presented in Table 1 check for invalid write operations in the memory, while the subsequent ones perform similar checks for read operations. Considering the first assertion (ASR_1), if the IR does not contain the OP_STORE operation, then the write signal (wr) must not be asserted. Similar checks are performed using the other assertions presented in the table. For more information on the memory access mechanism in the `Viper` processor, the reader is referred to its documentation [145]. Simulation results demonstrate that these assertions can effectively detect the HT inserted in the `B19_T500` benchmark, with further discussion on their effectiveness provided in Section 3.6.

Table 1: Considered assertions for detecting HTs on B19-T500 *benchmark*

Name	Assertion definition
ASR_1	`assert always {(!(IR == OP_STORE)) -> (!wr)};`
ASR_2	`assert always {(IR == OP_STORE) -> (wr)};`
ASR_3	`assert always {(!(IR == OP_READ)) -> (!rd)};`
ASR_4	`assert always {(IR == OP_READ) -> (rd)};`

3.2 Binding the Assertions to the Main Design

Simulation provides valuable insights into the incorrect behavior of a circuit and its internal values. However, it does not offer sufficient information about the design's PPA characteristics. Consequently, it is impossible to assess the quality of the assertions using simulations alone. To obtain accurate PPA reports, the design must be synthesized.

As mentioned before, PSL and SystemVerilog are the most popular languages for describing assertions, but they are not directly synthesizable. To address this issue, the MBAC tool [146] is used to convert PSL and SystemVerilog assertions into a synthesizable Verilog format. This allows the PPA results to be obtained after synthesis.

Once the synthesizable code is generated from the assertions, it can be bound to the main circuit to evaluate the effectiveness of the assertions based on the overheads imposed on the circuit. For this purpose, first, the main circuit without the assertions is synthesized,

and reports on the maximum clock frequency, power, and area are obtained. Then, the original circuit with the bound assertions is synthesized, effectively turning the assertion into an embedded online checker. Finally, the results of the two syntheses are compared to evaluate the overheads of the assertions.

3.3 Security Coverage

Despite having information about the PPA results for each assertion, it is not possible to make a definitive decision regarding their effectiveness. While the exact cost of these assertions is known, there is a lack of knowledge about their success in HT detection. Therefore, a new evaluation scheme is needed to balance the costs and the benefits. In this context, I propose a new metric for assessing assertions based on security properties.

To achieve this, every node n in the design is categorized into either the set of covered nodes C or the set of vulnerable nodes V. Initially, all the nodes are considered to be vulnerable. As indicated in Equation 1, in order to consider a node as covered, there must be a functional path between the node and the output of any assertion. Therefore, merely having a connection between the node and the output of the assertion is not enough, and conventional methods, such as extracting the input cone(s), are not applicable in this case.

$$n \in \begin{cases} C & \text{if a functional path exists between } n \text{ and any } assertion_k \\ V & \text{if no functional path exists between } n \text{ and every } assertion_k \end{cases} \quad (1)$$

Where k denotes the identifier number of the assertion. Therefore, the SC metric for a design *Des* is defined as follows:

$$SC_{(Des)} = \frac{|C|}{|C| + |V|} \quad (2)$$

Where $|C|$ and $|V|$ are the number of the covered and vulnerable nodes, respectively. In other words, The SC is defined as the ratio of nodes covered by the assertion to the total nodes within the design.

Figure 12 shows an example of a design with two integrated assertions, namely Assr_1 and Assr_2. The assertion circuit, which includes multiple logic gates upon synthesis and integration with the original design, is depicted as a single rectangle in Figure 12 for the sake of simplicity. It is important to note that adding the assertion circuit introduces a new Primary Output (PO) to the design, which can be used to verify if the assertion rises. To calculate the SC for the entire design, Equation 2 is applied. This equation can also be used to determine the SC for each individual assertion, allowing for a comparative analysis of their security properties.

In this scenario, the SC for each assertion is computed by finding the number of covered nodes (C). As shown in Figure 12b, nodes 1, 2, 3, 6, 7, 9, 12, 13, and 16 (highlighted in green) have at least one functional path to Assr_1. Similarly, the covered elements for Assr_2 (highlighted in blue) include nodes 2, 3, 4, 5, 7, 8, 10, and 17, as depicted in Figure 12c. Therefore, the SC values for Assr_1 and Assr_2 are 50% and 44.44%, respectively. If both assertions are bound to the design at the same time, the overall SC for the design can be increased to 77.78%.

It is important to note that some nodes may appear in multiple subsets of covered nodes for different assertions, as exemplified by nodes 2, 3, and 7. Conversely, certain nodes may not be covered by any assertions, such as nodes 11, 14, 15, and 18. Furthermore, the proposed approach is not limited to combinational designs, as sequential elements

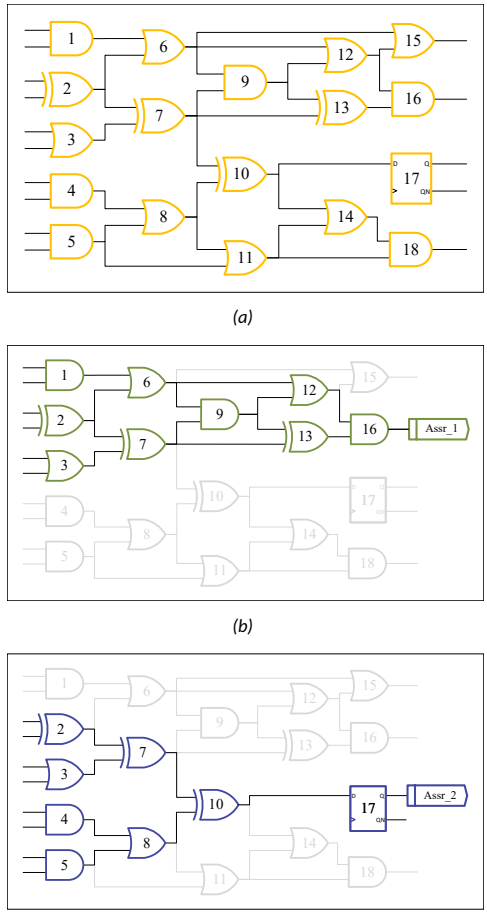


Figure 12: An example of a) original design, b) nodes covered by bound assertion Assr_1, and c) nodes covered by bound assertion Assr_2 (from [88])

such as flip flops (node 17 in Figure 12) are also taken into account as covered elements if there is a path between them and the assertion.

Hence, if any node in the design is the location of the payload of an HT, an assertion that can be reached by that node can detect the malicious logic. A higher number of nodes reachable from the original circuit to the assertion output indicates better coverage in HT detection for that assertion.

To obtain SC for each assertion, the circuit is first synthesized with the assertion bound to it. Then, all the nodes in the synthesized netlist are extracted using a tool that generates a list containing all nodes inside the netlist for further analysis. The generated list is then submitted to the SPV tool. This tool is primarily used for taint analysis [139], checking if design parts are securely isolated. With modifications, this tool can be used to calculate the SC by creating a list of node pairs (origin, destination), where all circuit nodes are possible origin nodes, and the destination node is the output of an assertion. To check the existence of functional paths between pairs, the SPV tool creates properties for each pair and attempts to prove the absence of a functional path or provide counterexamples. Once

the analysis is completed, SC is calculated using Equation 2.

With the SC information, the effectiveness of assertions can be evaluated in terms of security. This metric can be used to perform trade-off analysis to help users decide which assertions best suit their needs. It is important to note that not all circuits require 100% SC. For instance, if certain sensitive parts of the circuit have already been identified and only those need to be secured, covering those parts may be sufficient to meet user requirements.

3.4 OpenTitan - A Case Study

In Section 3.1, the use of custom assertions as security checkers for detecting HTs was studied. However, writing top-level assertions for security purposes is time-consuming and challenging to generalize. The main contribution of the approach presented in this chapter is to ***reuse existing verification assertions*** as security checkers.

For this study, I used OpenTitan, an open-source project featuring an embedded RISC-V-based processor and IPs from various vendors. The project comes with functional assertions for different IPs, making it an ideal candidate for evaluating the suitability of these assertions as security checkers. Obtaining realistic designs and verification assets from the industry is often challenging, so OpenTitan stands out as a valuable resource for this research.

I selected the Register Top modules of each IP, which control transactions between the IP and the bus, grant access to read/write requests for IP registers, and have a unique error generation mechanism for invalid addresses. Since the Register Top modules of different IPs share the same assertions, they provide a good comparison among experiments. A set of selected assertions is shown in Table 2, with a total of 108 different assertions studied on 35 individual IPs of the OpenTitan SoC. A brief description of each assertion is presented as follows:

- wePulse: This assertion ensures that once the `reg_we` signal goes high (indicating a write enable condition), it must go low in the subsequent clock cycle. This can be used to validate that `reg_we` is only briefly asserted for a single clock cycle, ensuring that the write enable signal is properly managed and does not stay high across multiple cycles.

- rePulse: Similar to the previous assertion, this assertion ensures that when the `reg_re` signal is asserted (indicating a read enable condition), it must be deasserted in the following clock cycle. This verifies that `reg_re` is only briefly activated for a single clock cycle, ensuring proper management of the read enable signal and preventing it from remaining high across multiple cycles

- reAfterRV: This assertion ensures that whenever there is a rising edge on either the read enable (`reg_re`) or write enable (`reg_we`) signals, the signal `tl_o` should be asserted in the subsequent clock cycle. This helps in verifying that the `tl_o` signal is correctly managed and activated following an enable condition (either read or write).

- en2addrHit: This assertion ensures that whenever either the write enable (`reg_we`) or read enable (`reg_re`) signals are active, the `addr_hit` signal must be one-hot encoded. This helps in verifying that the address hit signal is correctly managed and represents a valid single address hit when either read or write operations are enabled.

To obtain the SC of each assertion, the same flow as explained in the previous sections is used: MBAC translation followed by assertion binding and synthesis. The nodes from the synthesized netlist are then fed into the SPV tool to calculate the SC.

Table 2: Considered assertions for Register Top modules of different IPs in OpenTitan SoC

Name	Assertion definition
wePulse	`assert property (@(posedge clk_i) disable iff ((!rst_ni) !== 1'b0) $rose(reg_we) |=> !(reg_we));`
rePulse	`assert property (@(posedge clk_i) disable iff ((!rst_ni) !== 1'b0) $rose(reg_re) |=> !(reg_re));`
reAfterRv	`assert property (@(posedge clk_i) disable iff ((!rst_ni) !== 1'b0) $rose(reg_re || reg_we) |=> tl_o);`
en2addrHit	`assert property (@(posedge clk_i) disable iff ((!rst_ni) !== 1'b0) ((reg_we || reg_re) $onehot0(addr_hit)));`

3.5 Optimizing the Assertion List

Manually checking assertions to determine if they are top-level is a time-consuming process, questioning the efficiency of the proposed approach. To address this, a methodology is presented to help users select efficient security checkers from available assertions based on their needs. This step is necessary since not all functional assertions are useful for security purposes.

Figure 13 presents an automated flow for the assertion selection process. The first step in this process involves selecting an assertion from a candidate list, which consists of assertions that can be synthesized. Assertions with a simulation-based nature that cannot be synthesized will be filtered out at this stage. The selected assertion is then converted into synthesizable logic and integrated into the design. The prepared design is used for overhead evaluation, where synthesis is performed with the assertion, and PPA metrics are compared against the values obtained from the original design.

After evaluating the overheads and ensuring they meet user-defined requirements, the next step is to calculate the SC. This is accomplished by using Equation 2 and employing the SPV tool. If the assertion passes the overhead evaluation and achieves the desired SC, it is added to the final list of assertions. However, if the assertion fails to meet the criteria (e.g., unacceptable overheads or insufficient SC), an alternative assertion from the candidate list is considered (if available). The decision in this step can be based on either the individual SC of the assertion or the overall SC threshold specified by the user for the entire design. Once the flow is completed, the generated netlist with the embedded assertion(s) is considered finalized.

As a case study, the `alert_handler` IP from the OpenTitan SoC is selected, which contains several assertions, and I demonstrate how to create a list of security checkers from these assertions using the proposed methodology. In the first step, a candidate list of 13 different assertions is created, all predefined by OpenTitan developers to ensure the design's functionality remains as intended.

Since the final security checker list is based on user needs, two different strategies are defined for selecting appropriate candidates: **fixed-threshold** and **dynamic-threshold**. It is important to note that defining strategies is completely flexible and depends on the desired level of security and acceptable PPA overheads. In the following, the proposed optimization flow for the defined strategies is explained.

Fixed-Threshold Strategy: In this strategy, a fixed threshold margin in terms of SC is set

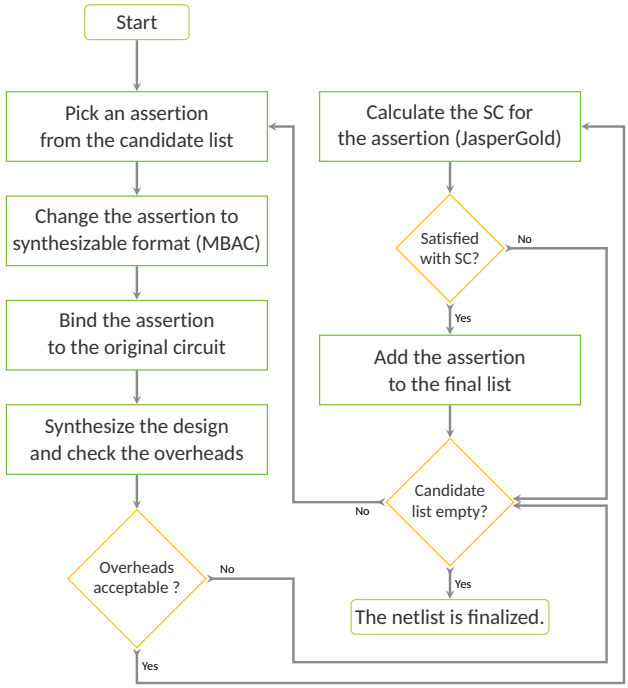


Figure 13: Optimization flow for selecting the assertions to be used as security checkers (adopted from [73])

for the overheads. If the overheads of a selected assertion exceed the defined threshold, it is immediately discarded, and another assertion is picked from the candidate list. This process continues until all assertions in the candidate list are evaluated. For example, if the performance overhead for a given assertion is X (percent), the SC should be at least 10X (percent). It should be noted that the number 10 is a configurable parameter that can be set by the user based on their specific requirements and the characteristics of the design under analysis. In this study, it has been empirically observed that this value works well for achieving a balance between the SC and the introduced overheads.

Dynamic-Threshold Strategy: In this strategy, the threshold margin for overheads is first adjusted based on the average overheads. A similar threshold is then set for the SC, taking into account the impact of each assertion on the overall SC. Consequently, each assertion is evaluated dynamically by considering both its overheads and its impact on the overall overheads and SC. For instance, the maximum PPA overhead for a given assertion should not exceed twice the average PPA overhead. For SC, only assertions that have a positive impact on the overall SC of the circuit compared to other assertions are chosen.

Unlike the previous strategy, where assertions are assessed individually, the dynamic strategy performs comparisons between competing assertions. Since looking at SC results for individual assertions does not provide information about their positive impact, this strategy selects only assertions that perform better than average.

The first strategy is simple and easy to implement but requires the user to define a constant (i.e., 10) for the threshold. The second strategy does not require such a constant but needs a sufficient number of assertions to define average overhead and coverage. The next section shows how the dynamic strategy can be more effective than its fixed-threshold counterpart. However, more complex strategies can be defined, which can be explored in

future work.

3.6 Experimental Results

This section presents the experimental results, encompassing PPA overheads and SC values achieved for various designs, as detailed in earlier sections. Cadence Genus, with the target cell library being a commercial 65nm CMOS library, is employed for all the experiments conducted and reported here.

Figure 14 illustrates the normalized PPA overheads for the considered assertions in the `B19-T500` benchmark. As depicted in this figure, three assertions (*ASR_1*, *ASR_2*, and *ASR_4*) have zero area overhead. The highest overheads are observed in *ASR_2* and *ASR_1*, which cause the circuit to consume 9% more power when bound. Furthermore, the timing overhead for all assertions is below 6%. It is important to note that, according to the heuristic, normalized numbers lower than one fall within the noise margin and do not impact performance

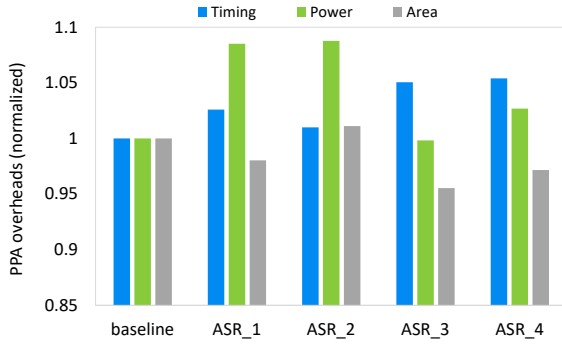


Figure 14: PPA overheads imposed by different assertions on the B19-T500 benchmark from Trust-Hub (from [73])

Table 3 presents the SC calculated for the same assertions in the `B19-T500` benchmark. The second and third columns present the total number of nodes and the number of covered nodes for a given assertion bound to the design, respectively. The fourth column indicates the percentage of the SC. Since the total nodes encompass both the nodes in the original design and those associated with the assertion, the total node count varies for each assertion.

As shown, proposed assertions cover an average of 6.8% of the total nodes in the circuit. This means that they can detect HTs within their covered areas, regardless of how rarely the HTs are triggered and what impacts they might have on the circuit. This is one of the primary advantages of the presented method, as the user no longer needs to worry about activating rare HTS.

Figure 15 illustrates the SC obtained for different IP Register Top modules of the OpenTitan SoC. The highest SC is 4.77% for the `nmi_gen_reg_top` module. However, the majority of IPs have a SC value below 1%, which does not make them good candidates for being security checkers. This is primarily because these assertions only perform small interface checks and do not describe the top-level behavior of the circuit. Instead, they cover only some local nodes, resulting in low SC for the entire circuit. This highlights the importance of the optimization step in the proposed methodology to avoid selecting unnecessary assertions that have minimal impact on HT detection.

Table 3: SC for the synthesized assertions bound to B19−T500 benchmark

Assertion name	**Total nodes**	**Covered nodes**	**Security Coverage** (%)
ASR_1	5014	315	6.28 (%)
ASR_2	5062	304	6.01 (%)
ASR_3	4916	367	7.47 (%)
ASR_4	4944	369	7.46 (%)
Average	**4984**	**339**	**6.80** (%)

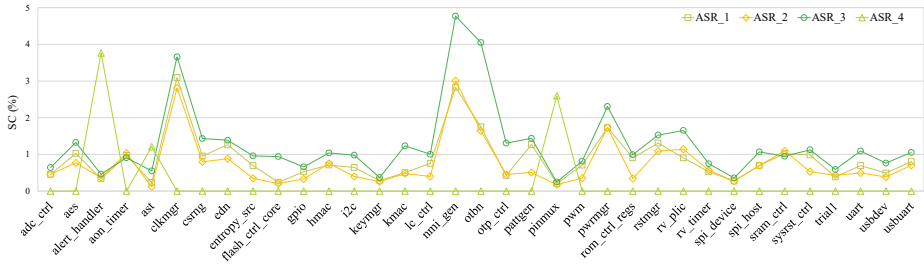


Figure 15: SC percentage for the Register Top modules of OpenTitan IPs (from [88])

3.6.1 Optimizing the Assertions

This section presents a practical experiment using the optimization flow shown in Figure 13. Previously, two strategies for selecting appropriate assertions were defined in Section 3.5, and now more details are provided about the assertion selection procedure.

Fixed-threshold strategy: The PPA overhead results for the assertion candidate list of `alert_handler` IP are shown in Figure 16a. As shown in this figure, the maximum overhead is due to the timing degradation of the `ah_asr_8` assertion (2.99%), while the minimum overhead is from the `ah_asr_3` and `ah_asr_4` assertions with a value of 0.75%. The next step is to check the SC, which should be at least ten times higher than the maximum overhead for each assertion. Figure 16b shows the SC results obtained from each assertion using the SPV tool. For clarity, numbers are associated with the assertion names. Based on these results, `ah_asr_12` and `ah_asr_13` can be disregarded since they do not meet the required SC condition, and the remaining candidates are considered as the final security checkers. Although 15% of the assertions were removed based on this strategy, defining smarter strategies can improve the effectiveness of the final list. Therefore, a second strategy (dynamic threshold) on the same candidate list is defined to achieve better efficiency.

Dynamic-threshold strategy: For the first condition of this strategy, the average overhead for all assertions is calculated to be 1.79%. Consequently, all candidates pass this condition since they have less than twice the average overhead in all cases (Figure 16a). However, for the second condition, referring to the SC results alone is insufficient, as they do not provide a basis for comparison. Therefore, an additional step is required to select the security checkers.

For this purpose, the assertions are arranged in descending order of SC, starting from the highest (`ah_asr_11`) to the lowest (`ah_asr_12`). Beginning with `ah_asr_11`, the assertion with the next highest number (`ah_asr_10` in the first round) is added, and the SC for the newly formed set of assertions is calculated. This process is repeated until the

lowest number is added to the list.

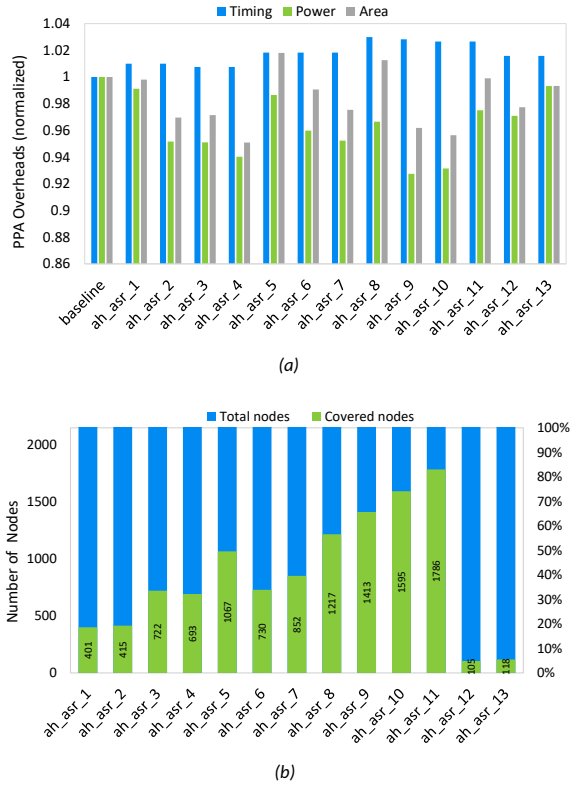


Figure 16: The figures for a) PPA overheads imposed by different assertions, b) Number of covered nodes for the individual assertion of `alert_handler` IP (from [73])

Figure 17 shows the SC numbers for each set of assertions. The assertion numbers are included in the set names to identify the effect of the added assertion in each step. For example, `ah_asr_11_10` represents a set starting from `ah_asr_11` (the highest coverage) and ending with `ah_asr_10` (the last assertion added), while `ah_asr_11_9` includes `ah_asr_11`, `ah_asr_10`, and `ah_asr_9`. A moving average trend-line is added to the figure to aid in selecting the best assertions. Since the moving average trend-line has a period of 2, it enables a good comparison between the SC of the newly added assertion in each stage and the two previous assertions. If the SC obtained after adding an assertion crosses the moving average trend, it indicates a noticeable difference.

Referring to the second condition of Dynamic-threshold Strategy and Figure 17, only three assertions have SC numbers that cross the moving average trend-line (`ah_asr_5`, `ah_asr_7`, and `ah_asr_13`), and they can be added to the final list. Additionally, the `ah_asr_11` assertion is added to the final list due to having the highest SC.

Compared to the SC numbers of different Register Top modules of various IPs (Figure 15), the SC numbers of different assertions in the `Alert Handler` IP are relatively higher (Figure 16b). This is primarily because the assertions written for this specific IP describe a top-level behavior of the design, rather than checking only local signals and interfaces.

These two examples demonstrate that different strategies can be defined based on user needs, making the presented approach flexible. Furthermore, one of the advantages of

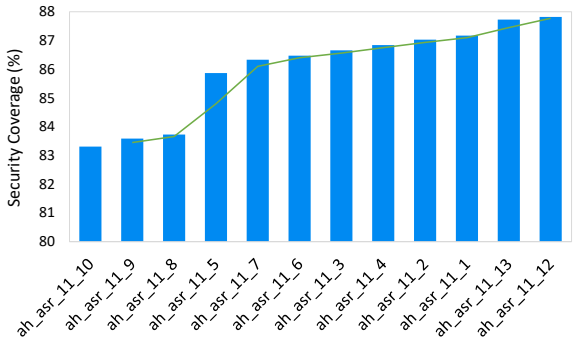


Figure 17: SC percentage for different assertion sets of `Alert Handler` IP (from [73])

this work compared to current approaches is its simplicity and lack of complex procedures. For instance, the work presented in [141] supports HT detection with flexible overheads, but it requires substantial effort and complicated steps. In contrast, my approach uses commercial tools that are widely available, increasing the portability and scalability of the presented work.

4 Enhancing IC Security by Embedding Online Checkers during Physical Synthesis

This chapter presents a comprehensive approach to enhance IC security throughout the design process in the back-end stage. In the previous chapter, a systematic method for converting existing verification assets into effective security checkers was introduced by repurposing verification assertions at the front-end phase of IC design [73]. To further enhance security, I propose a novel technique for incorporating online monitors during physical synthesis, offering an additional layer of protection at the back-end phase [88].

While integrating online monitors is not a new concept, most previous works have focused on introducing these checkers during the front-end phase of design. In contrast, this work takes a different approach by directly incorporating the checkers into the layout during the back-end phase, while still taking into account the front-end inserted assertions. This strategy allows for a more comprehensive security solution that spans both front-end and back-end design phases [88]. Integrating security checkers in the back-end design phase provides unique benefits, as the design is close to its final form at this stage. One such advantage is the ability to achieve more efficient area utilization, as the precise location of each design element is determined during the physical synthesis process.

Although this back-end methodology can be considered as a complementary approach to the front-end method, both techniques can be employed independently, depending on user preferences and specific requirements. This flexibility allows for a more customizable and adaptable solution to enhance IC security.

As deeply discussed in Section 2, the IC production process consists of multiple stages, as depicted in Figure 18. Front-end engineers convert the high-level design description into a gate-level netlist through logic synthesis. This netlist is then passed to the back-end team, where engineers modify it according to specific constraints such as area, power, and timing. The resulting layout is sent to a foundry for fabrication. Upon receiving the chip, specific tests are performed to verify its functionality.

As shown in Figure 18, the assumption of this work is that the foundry is considered an untrusted facility where an adversary (e.g., rogue engineer) may be present. The design and test stages, including the design house and test house, are assumed to be trusted. Defensive techniques are implemented in the design by front-end and back-end teams before sending it for fabrication, aiming to counter fabrication-time attacks.

It is also assumed that the attacker within the foundry has the capability to insert sophisticated, small, and rarely activated HTs that can evade side-channel analysis, logic testing, and simple forms of chip inspection such as electrical testing. The attacker has access to the target technology's PDK and advanced commercial CAD tools. This work focuses on functional HTs that alter the chip's functionality, allowing their effects to be observed by comparing internal signals with the expected ones.

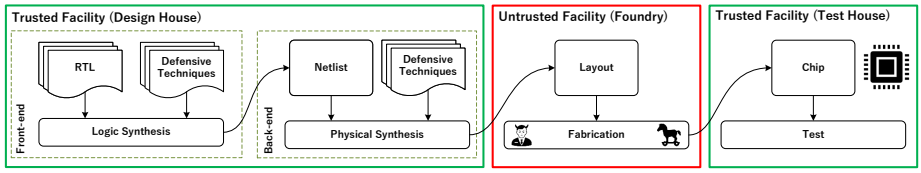


Figure 18: Different stages of IC design: The design house and the test house are considered trusted, while the foundry is assumed to be untrusted.

4.1 Limitations of the Concept of Reusing Verification Assertions as Security Checkers

The primary focus of the work presented in Chapter 3 is introducing of a new metric (i.e., SC) for assessing the security properties of the generated security checkers. However, it is crucial to acknowledge that while achieving higher SC numbers for various assertions might suggest improved design security, ensuring design security involves more than just relying on SC metrics, even when they are in high ranges (e.g., exceeding 80%). There are several challenges associated with SC, and some significant ones include:

1. **Ineffectiveness of verification assertions at runtime:** Functional assertions can cover various security properties by ensuring the expected behavior of the circuit. However, they may not be precise enough to thoroughly cover the negative or unexpected behavior of a circuit under attack. Although the SPV tool is used to calculate taint propagation coverage, the effectiveness of detecting HT behavior by the synthesized assertions at runtime is not guaranteed.

2. **Scalability concerns:** The requirement to "bind the assertion" and synthesize the entire design for characterizing overheads may present scalability challenges for larger designs. The resource-intensive nature of this process could restrict its practicality for more extensive and complex circuits, making it difficult to efficiently apply this approach to large-scale projects.

3. **Assertion availability:** The approach focuses on reusing existing assertions instead of generating new ones. However, a potential challenge arises when no suitable assertion with acceptable SC is found for a given design. This situation was illustrated by the varying SC numbers obtained for different assertions in the design in Section 3.6. In such cases, alternative security measures may need to be considered to ensure adequate protection.

To address the aforementioned challenges and limitations, it is crucial to integrate an additional layer of security into the design. However, it is important to note that this additional layer does not necessitate significant changes to the existing circuit design and fabrication processes. Instead, the goal is to develop a security solution that can be seamlessly incorporated within the current design flows, ensuring both practicality and effectiveness in enhancing the overall security of the system. This supplementary measure can help enhance the overall security of the system and provide more comprehensive protection against potential threats.

4.2 Adding Online Monitors during Physical Synthesis

Figure 19 presents a simplified view of the layout of a block within an IC. The green polygons represent standard cells, which are later interconnected through various metal layers to establish the logical function of the design. Due to fabrication complexities, especially in modern process nodes, achieving 100% density with a layout entirely filled with standard cells is impractical. Consequently, gaps are present in the layout, highlighted in red in Figure 19. These gaps can potentially be exploited by an adversary for inserting malicious logic (i.e., HTs) [20, 87, 130, 147].

Although these gaps are typically filled with filler cells or decap cells before being sent to fabrication, these cells lack functionality and are not connected to the design's logic. In the case of decap cells, they can be removed, but this may have a slight impact on the overall design. The proposed approach takes advantage of these gaps and available resources to

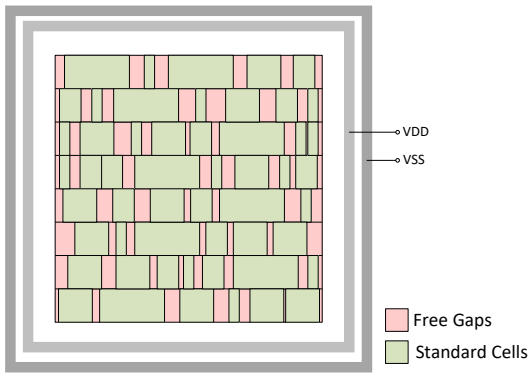


Figure 19: Illustrative example of a block layout within a chip (from [88])

insert online checkers into the design. Doing this not only adds an extra security layer to the design but also limits the adversary's ability to insert malicious logic by increasing the density and congestion in the layout.

While the presented method can be employed independently, it is used as a complementary approach alongside reusing assertions for detecting HTs to address its limitations. It is important to note that this technique leverages the Engineering Change Order (ECO) capabilities of the CAD tools for inserting online monitors into the layout. ECO features enable engineers to make last-minute modifications to the existing layout, such as adding or removing components and changing connections. By utilizing ECO features, alterations to the original layout with each added online monitor are minimized, which provides optimal overheads compared to front-end approaches. This makes the presented approach enhance security while maintaining design efficiency.

4.2.1 Generation of Online Monitors for the Back-end Phase

The online monitors provide an additional layer of protection for nodes that are not covered by assertions. To implement this protection strategy, a Dual Modular Redundancy (DMR) scheme is used, as depicted in Figure 20. In the left image (Figure 20a), a segment of a design is shown, where the covered nodes (10 and 17) are highlighted in green, and the vulnerable or uncovered nodes (11, 14, and 18) are highlighted in red. To create an online monitor for this design, an exact duplicate of the uncovered gates, with the same equivalent gates from the library (i.e., the same gate type and drive strength), is first generated. Then, the output of the duplicated part is compared with the output of the original part by XORing these two signals, as illustrated in Figure 20b. Consequently, implementing an online monitor for this segment results in seven new covered nodes (11, D11, 14, D14, 18, D18, and V18) being added to the previously covered nodes (10 and 17). In this figure, the duplicated nodes are labeled with the prefix "D", while the voter nodes are labeled with the prefix "V". For instance, node D11 is the duplicated version of node 11, and V18 is the voter which compares the output of node 18 with D18.

It is important to note that the output of node V18 is utilized only for security purposes and has a minimal impact on the performance of the design. This is because it only adds some wire capacitance to node 18, which is a negligible effect in most cases.

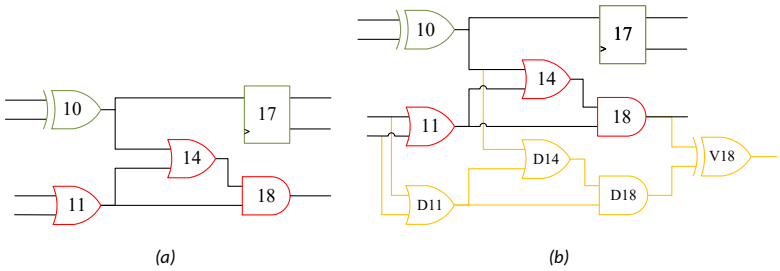


Figure 20: An example of a) design before adding online monitors, and b) design with the protection logic (D11, D14, and D18 as the duplicates and V18 as the voter) to protect the uncovered gates (11, 14, and 18)

4.2.2 Embedding Online Monitors into the Layout

The complete flow for incorporating online monitors during physical synthesis is illustrated in Figure 21. It consists of four primary steps, which are explained below:

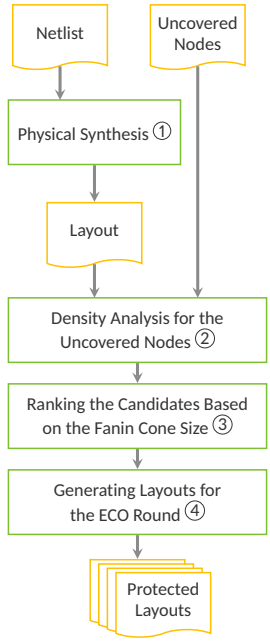


Figure 21: An overall flow of integrating online monitors during physical synthesis (from [88])

1) Physical Synthesis: The initial step in the comprehensive flow involves using a physical synthesis tool to convert the netlist into a layout. This netlist includes the original design and all specified assertions associated with it. The process consists of several stages such as placement, clock tree synthesis, and routing. Physical synthesis provides important details such as the precise placement of standard cells, the physical arrangement of the clock, and the structure of interconnections concerning wire length and the utilization of each available metal layer. In essence, physical synthesis provides insights into the spatial configuration of the design. It is important to note that this time-consuming step is only performed once for each design, making the proposed approach compatible with

industry-standard flows without requiring any modifications to them.

2) Density Analysis for the Uncovered Nodes: The resulting layout, combined with the report of uncovered nodes from the SPV tool, serves as the input to a developed analytical tool. The goal here is to identify uncovered nodes with available gaps around them, which can accommodate the online checker responsible for protecting the respective node. If there is no space available around the uncovered node, the online checker might be placed at a distance, leading to increased resource utilization and degradation of the PPA parameters of the design. This step is critical in the proposed flow to prevent such situations, minimizing layout modifications, ensuring compatibility with the ECO flow, and reducing overheads. In contrast to front-end protection schemes that only provide estimated overheads, this approach offers a distinct advantage by determining the actual overheads accurately. It is important to note that the radius of searching for an area around each node is adjustable and can be changed based on the design size and density.

3) Ranking the Candidates Based on the Fanin Cone Size: The candidates generated by the density analysis tool are subjected to a ranking process. As previously mentioned, the preference is to place the online checker for a group of interconnected uncovered nodes rather than individual gates. This choice provides benefits in terms of area, power, and routing resources. Consequently, this ranking system prioritizes subsets of candidate gates with larger input cones, optimizing the overall efficiency of the protection scheme. It is important to note that during this cone analysis, only candidates with a cone size[1] of 2 or greater are considered to enhance the efficiency of the proposed approach.

4) Generating Layouts for the ECO Round: In the final step, protected layouts are generated. To accomplish this, a tool is developed that takes the ranked list of candidates (generated in the previous step) and incrementally integrates the online checkers into the layout. Specifically, one online monitor is added to the design at a time, and a new layout containing the added monitor is generated. This process is repeated from the top of the ranked list to the end. Consequently, if there are *n* candidates (the nodes suitable for protection by online checkers) in the ranked list, *n* different layout files are created in an iterative manner. Each protected layout file, such as *Layout1*, contains the online monitor from *Candidate1*; *Layout2* contains the online monitors for *Candidate1* and *Candidate2*, and so on. It is important to note that these protected layouts are generated to be used along with the ECO flow, and the finalized layout, which includes all potential online checkers and is intended for fabrication, is obtained after the completion of the entire ECO rounds.

Since each new protected layout only adds one online monitor compared to the previous one, the user has the option to keep or discard the added online monitor. This incremental method helps to carefully integrate online monitors into the design while efficiently managing the resources needed for each addition. Additionally, it allows for a detailed evaluation of the impact on the PPA parameters of the design.

4.2.3 ECO Flow

As described, the proposed approach efficiently inserts online checkers and generates protected layouts, prioritizing area and resource utilization. This approach can be further enhanced by introducing a timing-aware element to the ECO flow. Hence, a new metric is introduced called the Degrading Factor (DF), which is set at 25% of the total positive setup slack of the design before the addition of online checkers. This DF serves as a threshold parameter for deciding whether to keep or discard protected layouts based on their impact on timing.

[1]Cone size refers to the number of gates or logic elements that have a direct or indirect influence on a particular node in the circuit.

To accomplish this, the PPA numbers of the initial layout are first stored before online monitors are integrated. The ECO flow starts by choosing the first protected layout, which includes the highest-ranked online monitor from the cone analysis, and then calculates the PPA numbers for this modified layout. Next, the total setup slack number is compared with the previous layout (the one without the newly added monitor). If the slack number is negative or if the difference between the new slack and the previous one exceeds the DF, the ECO flow rejects the newly added layout as it worsens the timing beyond user constraints and moves on to the next one. This process continues until all online monitors are integrated or there is no more slack available for the new logic. Furthermore, various checks are carried out at each ECO round to ensure compliance with design rules and prevent issues that may arise from implementing the new layout.

4.3 Experimental Results

This section presents the experimental results of integrating online monitors for different IPs of OpenTitan.

Figure 22 presents the calculated SC percentages for individual assertions within three selected IPs. These IPs have been chosen since the SC for each assertion is significantly higher compared to the figures obtained from the initial experiment in Section 3.6. The average SC is 85.83% for the selected assertions in the `alert_handler` (Figure 22a), 46.03% for the selected assertions in the `alert_handler_esc` (Figure 22b), and 38.35% for the selected assertions in the `flash_phy_rd` module (Figure 22c).

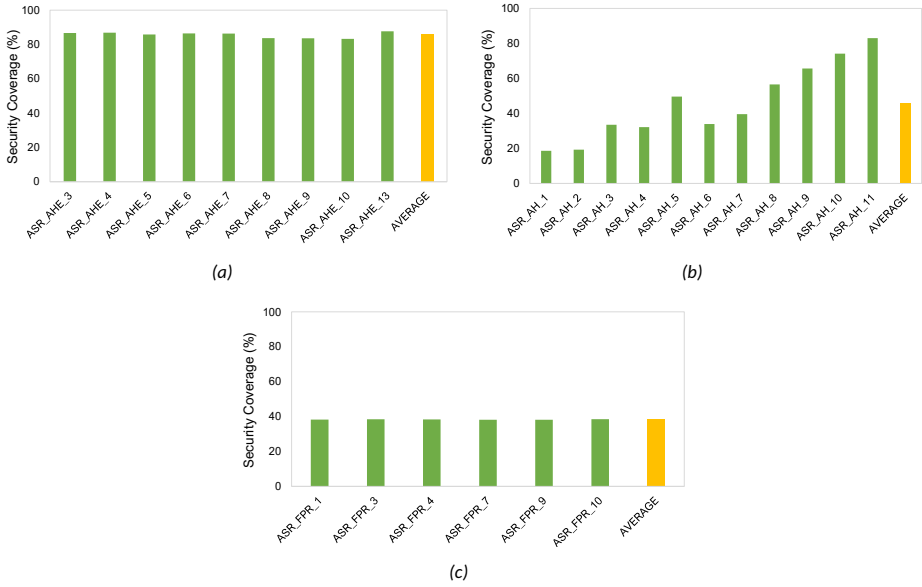


Figure 22: Calculated SC percentage for the different assertions in selected IPs of OpenTitan: a) alert_handler_esc, b) alert_handler, and c) flash_phy_rd (from [88])

To assess the efficiency of integrating online monitors during the back-end phase, five different IPs were chosen from all IPs studied earlier. Two of these IPs were selected for security reasons: `alert_handler_esc_timer` has the highest SC, and `keymgr_reg_top` has the lowest SC. The other three IPs were chosen based on their sizes: `ast_reg_top` is the largest design, `flash_ctrl_core_reg_top` has an average size, while the smallest

one is `nmi_gen_reg_top`. It is important to note that all these IPs are selected among 40 different IPs that already have some assertions to maintain the concept of repurposing existing assertions. This work does not introduce new assertions or modify existing ones across different IPs. The calculated SC for all these IPs is previously shown in Figure 22 and Section 3.6.

For all the results reported in this section, the Cadence suite is used: logical synthesis is performed by Genus, while physical synthesis is carried out by Innovus. The formal tool for performing taint analysis is JasperGold SPV, as mentioned earlier. The target technology node is a commercial 65nm CMOS one.

4.3.1 Impact of Adding Online Monitors on SC

To evaluate the impact of integrating online monitors on the security properties of each design, the same evaluation scheme given by the SC equation presented in Section 3.3 is used. In this equation, the total covered nodes (C) in the numerator now include both the nodes previously covered by the assertion and the newly covered nodes introduced by the online monitor.

Table 4 presents the results regarding the impact of added online monitors on the security of the considered designs. In this table, the first column denotes the IP name, while the second column enumerates the instances in each IP. The third column indicates the SC before the integration of online monitors. These values are obtained by binding all available assertions to each IP and analyzing the coverage using the SPV tool. The fourth and fifth columns represent the increase in SC specifically due to the online monitors and SC after adding the online monitors, respectively. As indicated, the lowest increase in SC is 0.43% for the `alert_handler_esc_timer` IP.

Table 4: The impact of adding online monitors on the security of selected IPs

IP Name	Instances	SC Before	SC Added	SC Total	# NCbM	# AM	# IM	# TM	Preventing Factor
`alert_handler_esc_timer`	1404	87.82%	0.43%	88.25%	6	1	0	1	Density
`ast_reg_top`	7048	2.49%	17.45%	19.94%	1382	183	7	190	Density
`flash_ctrl_core_reg_top`	7048	1.38%	16.6%	17.98%	954	105	264	369	Timing
`keymgr_reg_top`	4611	0.91%	9.98%	10.89%	490	55	86	141	Timing
`nmi_gen_reg_top`	214	3.53%	33.58%	37.11%	92	14	0	14	Density

NCbM: Nodes Covered by Monitors, **AM:** Added Monitors, **IM:** Ignored Monitors, **TM:** Total Monitors

This is mainly because of the IP's existing extensive coverage of nodes, making it difficult to identify suitable candidates that pass all stages of the online monitor insertion flow (as shown in Figure 21). This means finding a group of uncovered connected nodes with adequate space around them is challenging, given the limited total number of uncovered nodes. However, the increased SC does not exclusively signify the added security to each design. The introduced logic for inserting online monitors also occupies the gaps in the layout and utilizes the routing resources that could potentially be exploited by an attacker.

In column 6, the total number of covered nodes is displayed after the inclusion of online monitors. These covered nodes encompass those not covered by the assertions, as well as the new redundant logic added to form the online checker. Columns 7 and 8 show the number of applied and ignored online monitors in the IP, respectively. Column 9 indicates the number of online monitors that can be generated for each design after conducting density and cone analysis (Figure 21). The total number of online monitors equals the number of individual protected layouts produced for the ECO flow. It is important to note that not all generated online monitors can be integrated into the design due to timing constraints.

The final column identifies the factor that limits the addition of more online monitors. If all available online monitors are successfully integrated into the design, it indicates that no additional monitors can be generated, primarily due to the high density around the uncovered nodes. On the other hand, if some online monitors remain unembedded in the design (excluding those exceeding the DF), it is mainly because the design's timing resources have been exhausted, necessitating their exclusion. More details about the PPA restrictions are discussed in the next section.

4.3.2 Impact of Adding Online Monitors on PPA

The baseline layout for each IP is set up such that the design density ranges between 60% and 65%. This configuration provides a positive setup slack of approximately 10% of the clock period[2] for each design. Since online monitors introduce new logic that can affect the design's timing, the 10% margin enables the use of positive slack for integrating these monitors.

In Table 5, a comparison of various PPA metrics before and after the implementation of online monitors for the selected IPs is presented. The first column lists the names of the IPs, while the subsequent two columns provide details regarding the area and placement characteristics of the layouts. The second column displays the total area for each design, and the third column represents the placement density. It is noteworthy that the smallest design, `nmi_gen_reg_top`, exhibited a significant increase in area parameters following the addition of online monitors. This can be attributed to the fact that the size of the added logic became comparable to the overall design, consequently impacting the cell area, which denotes the space on a chip occupied by logic cells.

The fourth column represents the total power consumption for each IP in the study. The IP denoted as `nmi_gen_reg_top` exhibited the largest increase in power consumption due to its small size. The fifth and sixth columns present the timing characteristics of the design. While the hold slack remains relatively constant across all designs, the setup slack undergoes notable changes for most designs, attributed to the impact of redundant logic in different timing paths. The two designs with the preventing factor of timing, `flash_ctrl_core_reg_top` and `keymgr_reg_top`, display the most significant decrease in setup slack.

The last column represents the total wire length for each design. These metal wires are utilized to connect different parts of the design. The increase in the total wire length suggests that the design has become more congested, limiting the free routing resources available to be utilized by an adversary.

To obtain a more comprehensive understanding of the effects of incorporating each online monitor on the design characteristics, I have analyzed various PPA results at the end of each successful ECO round, where a new online monitor is integrated into the design. This thorough examination enables monitoring of the individual impacts on different attributes throughout the iterative ECO process. Figure 23 demonstrates the deterioration of setup slack after each successful ECO round, with the vertical axis representing the total setup slack time in nanoseconds (ranging from the worst to the best slack time) and the horizontal axis depicting the progression of ECO rounds.

As mentioned earlier, the ECO flow begins with the layout containing the assertions, which is labeled as the "Baseline" in Figure 23. As shown in the figure, some rounds have minimal impact on the timing, while others significantly degrade the total setup slack. In certain cases, the slack may even improve due to the heuristics of the physical synthesis

[2]A clock period is the time it takes for a clock signal to complete one full cycle. This is the time between two consecutive rising edges or falling edges of the clock signal.

Table 5: The impact of adding online monitors on the PPA metrics of selected IPs

IP Name	Total Area (μm^2)	Placement Density	Total Power (mW)	Setup Slack (ns)	Hold Slack (ns)	Total Wire Length (μm)
`alert_handler_esc_timer` (before)	3692.88	63.94%	1.46	0.328	0.133	22305.40
`alert_handler_esc_timer` (after)	3700.80	64.07%	1.47	0.328	0.133	22351.75
Difference	+0.21%	+0.20%	+0.68%	0.00%	0.00%	+0.21%
`ast_reg_top` (before)	28644.48	61.46%	9.93	0.287	0.084	353799.80
`ast_reg_top` (after)	30848.76	66.18%	10.26	0.091	0.07	397612.20
Difference	+7.69%	+7.68%	+3.32%	-68.29%	-16.67%	+12.38%
`flash_ctrl_core_reg_top` (before)	14243.40	64.25%	4.94	0.210	0.181	148407.90
`flash_ctrl_core_reg_top` (after)	15542.28	70.11%	5.13	0.007	0.186	170461.90
Difference	+9.12%	+9.12%	+3.85%	-96.67%	+2.76%	+14.86%
`keymgr_reg_top` (before)	18325.80	62.68%	8.11	0.278	0.152	186978.10
`keymgr_reg_top` (after)	19062.72	65.20%	8.29	0.000	0.151	199832.90
Difference	+4.02%	+4.02%	+2.22%	-100%	-0.66%	+6.87%
`nmi_gen_reg_top` (before)	769.68	61.16%	0.23	0.118	0.178	4841.51
`nmi_gen_reg_top` (after)	918.00	72.94%	0.27	0.043	0.179	6077.74
Difference	+19.27%	+19.26%	+17.39%	-63.56%	+0.56%	+25.53%

tool. However, the degradation does not exceed the DF, which is set at 25% of the total setup slack. It is important to note that this parameter can be adjusted based on the user's preferences. For example, if set to lower values, online monitors causing a sudden decrease in the total setup slack (e.g., the online monitor added in round 81 in Figure 23b) will be discarded, resulting in a more smooth overall trend for setup slack decrease.

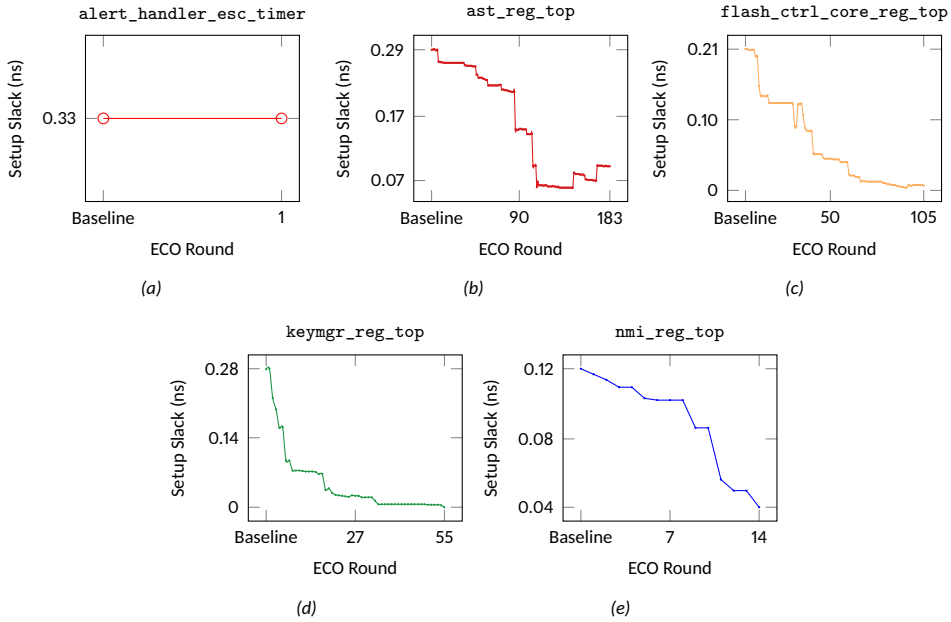


Figure 23: Changes in setup slack after each round of adding the online monitors for different IPs (from[88])

Figure 24 shows the progressive increase in wire length for each metal layer in the protected layouts. The wire length, measured in μm, is depicted on the vertical axis, and the horizontal axis identifies the protected layout for each round. Similar to Figure23, the term *Baseline* denotes the layout that includes assertions but lacks online monitors. Despite the varying number of metal stacks in different target technologies, newer technologies generally offer ten or more metal layers, and a higher utilization of upper metal layers suggests heightened congestion in the design. As a defender, the emphasis is on the greater use of upper metal layers, signifying an overall rise in congestion in the protected layouts. With the exception of `alert_handler_esc_timer`, where only a single online checker was available to be added, a steady trend of increased wire length in metal layers M3-M6 is noticeable for all designs in Figure 24. This trend reduces the available free routing resources for potential adversaries.

Figure 25 illustrates the layout views, showcasing the placement configuration before and after the integration of online monitors. Each row presents a pair of images, where the left image represents the cell placement of the layout before integrating online monitors, and the right image corresponds to the final protected layout after successfully completing all ECO rounds.

Similarly, Figure 26 showcases the layout views, including the routed view before and after the integration of online monitors. Each row also presents a pair of images, with the left image in each pair representing the routed view of the layout before integrating online

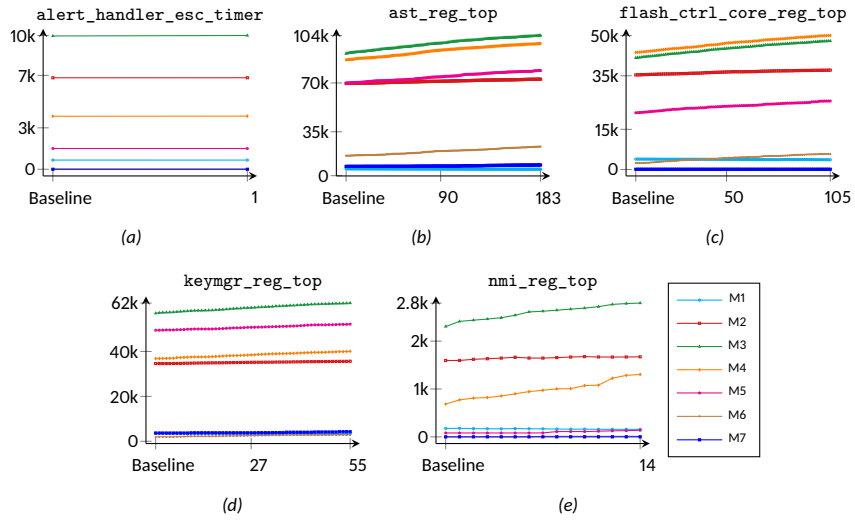


Figure 24: The impact of adding online monitors on the length of different metal layers in the protected layouts for different IPs (from [88])

monitors, and the right image corresponding to the final protected layout after successfully completing all ECO rounds.

Comparing the layout images on the right with those on the left in Figure 25 and Figure 26, reveals that the overall placement and routing configuration of the layouts remained unchanged, even for larger designs. This underscores the more efficient utilization of resources, a key advantage of the presented approach in adding online monitors during physical synthesis compared to similar works conducted in the front-end phase of IC design.

4.3.3 Comparison of the Presented Work with Other Techniques

Table 6 presents a comparison of the proposed method with various detection and DfHT techniques. The first column outlines the specific technique or category, while the second column references relevant works in that category. The third column categorizes the technique as detection, DfHT, or a combination of both. The subsequent column describes the chip design stage where the method is applied for protection. Column 5 identifies the location of the potential attacker, and column 6 indicates whether the technique can also be used to prevent HTs. The final two columns offer a concise summary of the advantages and drawbacks of the techniques, respectively.

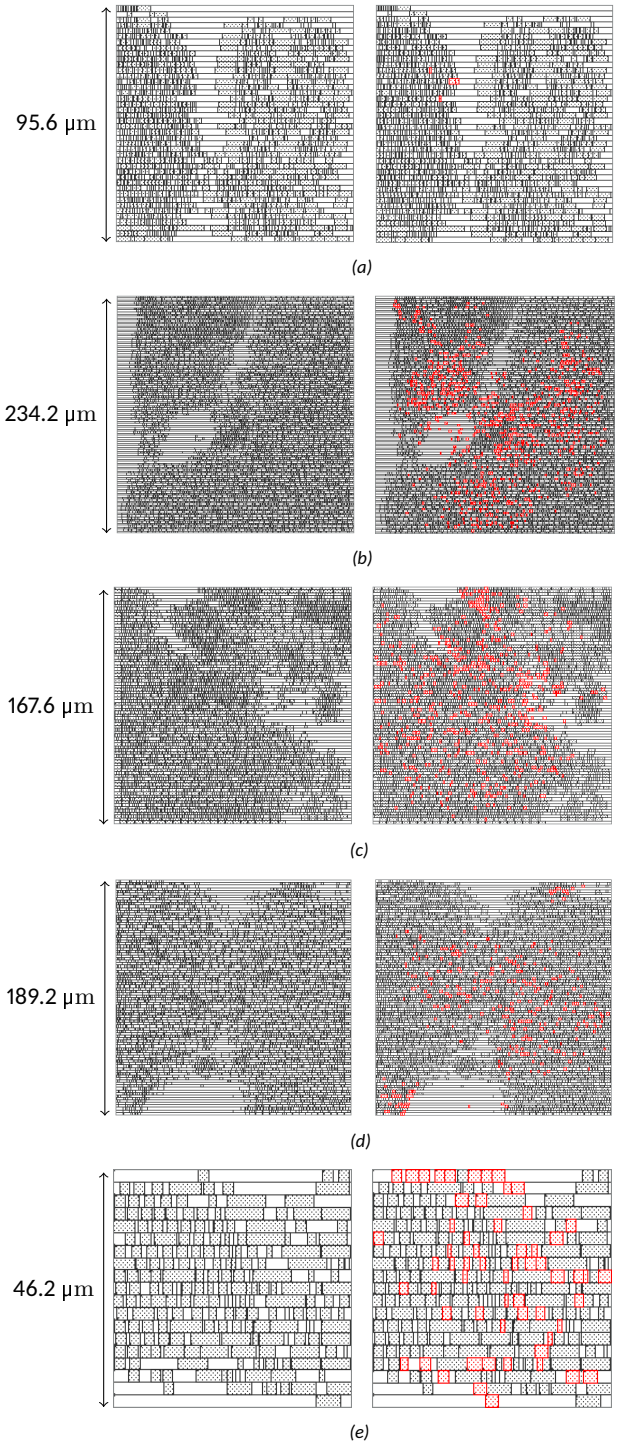


Figure 25: The layout view of selected IPs, whereas the left images in each row represent the placement configuration before adding online monitors, while the right one represents the view of each design after integrating the online monitors: a) `alert_handler_esc_timer`, b) `ast_reg_top`, c) `flash_ctrl_core_reg_top`, d) `keymgr_reg_top`, and e) `nmi_reg_top` (from [88])

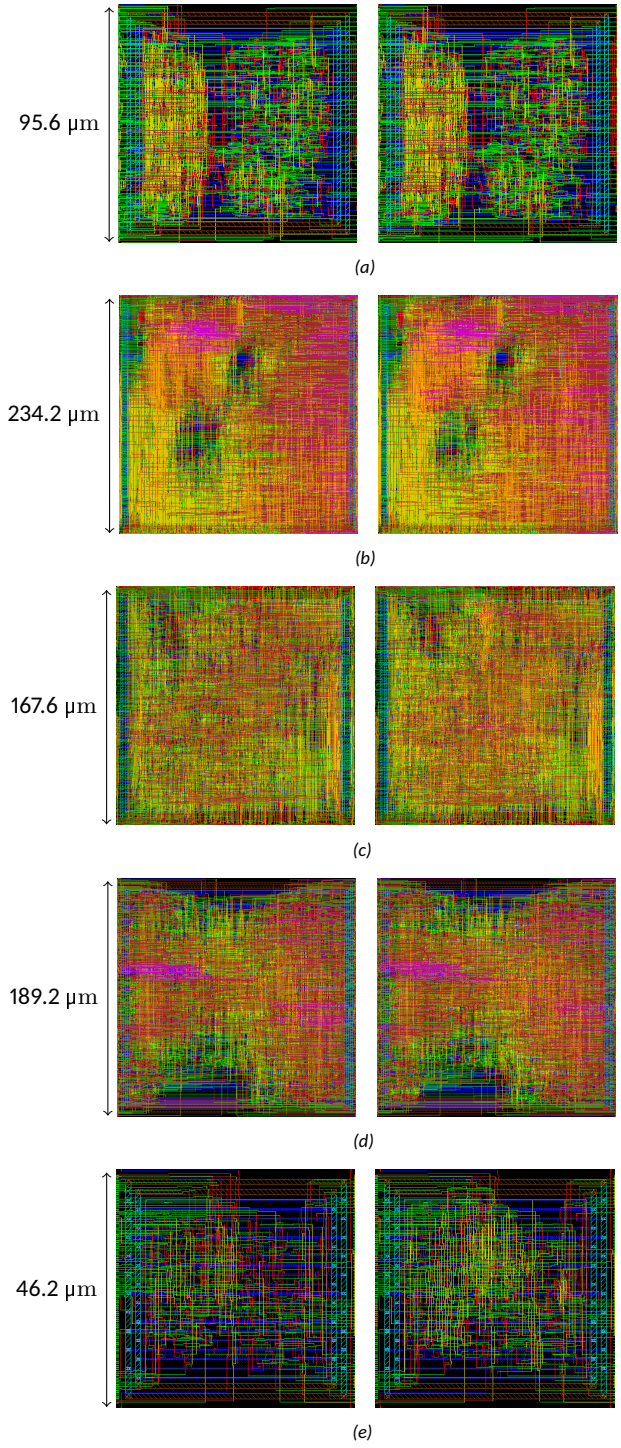


Figure 26: The layout view of selected IPs, whereas the left images in each row represent the routed view before adding online monitors, while the right one represents the view of each design after integrating the online monitors: a) `alert_handler_esc_timer`, b) `ast_reg_top`, c) `flash_ctrl_core_reg_top`, d) `keymgr_reg_top`, and e) `nmi_reg_top` (from [88])

Table 6: Comparison of Different Detection and DfHT Techniques

Technique	Refs	Detection or DfHT	Design Stage	Attacker Location	Prevents HT?	Pros.	Cons.
IC Fingerprinting, Delay Measurement	[103, 148, 149, 150]	Detection	Test	Des. house, Foundry	No	Nearly zero overheads, Non-destructive method	Needs reverse engineering for attributes of the Golden chip, Confusion with environmental and process variation effects
Logic Testing	[151, 152]	Detection	Test	Des. house, Foundry	No	Very fast technique, Can be fully automatic	All input combinations are not covered to activate HTs, Ineffective when dealing with HTs with sequential triggers
BISA, Layout Filling	[130, 131, 132]	DfHT	Back-end	Foundry	Yes	Replacement of filler cells with the functional ones, Independent testing circuit from the original one	Design becomes unroutable in high occupation ratios, Considerable overheads
Selective Placement	[65, 67, 68]	DfHT	Back-end	Foundry	Yes	Lowest overheads among HT prevention techniques	Risk of violating the critical paths in the routing stage, Timing degradation
TPAD	[141]	Both	Front-end, Back-end*	Des. house, Foundry	Yes	Zero false positives, Applicable to FPGA	Considerable overheads, Risk of degradation
This work		Both	Front-end, Back-end	Foundry	Yes	Reusing verification assets, Low PPA overheads	Achieving high density for some designs is challenging

* This technique is mainly applied in the front-end stage. To prevent CAD tool attacks, the design is split into two parts, each processed by different CAD tools independently. The final layouts from these tools are later merged in the back-end stage for being set to the foundry for fabrication.

5 SALSy: Security-Aware Layout Synthesis

This chapter presents a new methodology called Security-Aware Layout Synthesis (SALSy), which enables the design of ICs with inherent security considerations [69]. This approach is similar to the well-established practice of balancing PPA metrics and security, a concept that is referred to as security closure.

However, unlike PPA metrics, commercial layout synthesis tools do not offer any direct settings or options for security. Therefore, the task of SALSy is to work within the constraints and capabilities of these tools to indirectly achieve security properties in the final layouts. This involves modifying and adapting the existing algorithms and heuristics for placement, routing, CTS, and other physical synthesis tasks to make them more security-aware and resilient to various attacks and threats. Therefore, SALSy is a proactive strategy at the back-end phase that enhances the security of ICs against both fabrication-time and post-fabrication adversarial acts, including HT insertion, FI, and probing.

This methodology has been validated through a silicon demonstration, confirming its compatibility and effectiveness with a commercial PDK and library. SALSy achieves this enhanced security enhancement with only a minimal impact on power consumption, thus maintaining a balanced trade-off between security and PPA.

As discussed in detail in Section 2, in the fabless model, foundries are regarded as untrusted entities as design houses lack ownership or oversight over them. Consequently, IC design houses must prioritize safeguarding their designs (layouts) against potential threats originating from these untrusted foundries [153, 154].

Moreover, beyond fabrication-time attacks, numerous other threats exist. Once a finalized IC becomes available to malicious end-users on the open market, it becomes susceptible to attacks such as fault injection [21, 25]. In fault injection attacks, adversaries attempt to compromise the chip's security by introducing various faults into its operation.

Another post-fabrication attack is probing, where attackers seek unauthorized access to a chip's internal data through physical probing techniques [26, 155]. Typically, this attack aims to extract sensitive information like cryptographic keys or proprietary data, posing significant risks, particularly in critical or dependable applications [25].

To address these concerns, hardware security researchers have pursued the concept of security closure [61, 62, 63, 64, 65, 67, 68, 70]. This approach involves accepting certain overheads in terms of PPA to implement heightened security measures. The goal is to minimize vulnerabilities and potential attack surfaces, aiming to create trustworthy and resilient ICs capable of withstanding potential security breaches while ensuring reliable performance.

The methodology presented in this chapter outlines a comprehensive approach to achieving security closure, encompassing various techniques. The proposed flow (SALSy) is designed to be adaptable to layouts of any size, type, or technology. SALSy presents a generic and comprehensive approach to enhancing the security of IC designs during the physical synthesis stage. The proposed methodology addresses multiple security threats, including HTs, FI, and probing attacks. SALSy has been validated through the prototyping of a chip using a commercial 65nm CMOS technology, demonstrating its effectiveness and compatibility with current industry practices.

In addition to the development and validation of SALSy, this work also provides a comparative analysis of the utilization of commercial libraries and PDKs with open-source alternatives for security research. The analysis highlights the limitations and constraints associated with using open-source PDKs. To further facilitate and promote security research in the IC design community, this work also provides publicly accessible scripts that can be used to thoroughly verify and validate the techniques outlined in the study. These

scripts are designed to operate within a commercial physical synthesis tool, ensuring their compatibility and relevance to the current industry standards and practices.

5.1 Security Assessment Scheme

Researchers have introduced various metrics to evaluate the complexity of inserting HTs into a specific layout [71, 87]. This work uses the scoring framework presented in [71] to assess security. This framework is chosen because it considers a variety of threats (HT insertion, FI, and probing) instead of focusing on just one specific threat. Furthermore, to reflect the real challenges faced by an engineer during physical synthesis, design quality (i.e., PPA) is also taken into account in the final scores. Consequently, the overall score is a function of both *design quality* and *security*, as shown in Equation 3.

$$Score = DesignQuality \times Security \tag{3}$$

Where $DesignQuality$ consists of a weighted distribution of power, performance (in terms of clock frequency), area, and routing quality, and $Security$ consists of equally weighted metrics for HT insertion, FI, and probing. It should be noted that in this evaluation scheme, Front-Side Probing (FSP) is considered a proxy for FSP and FI.

The security scores for FSP/FI are determined by identifying a set of sensitive (security-critical) cells and their related interconnecting wires. These cells, known as ***cell assets***, and the related wires, called ***net assets***, are used to calculate a metric known as the ***exposed area***. This metric is calculated for each set of cell and net assets in each design and represents any spatial area that can be accessed from the top through the metal stack. An example of an exposed area is depicted in Figure 27. In this illustration, the red-marked cell areas indicate the exposed regions, making them susceptible to FI or probing attacks due to the lack of protection from other elements (i.e., metal wires) on the front side.

Figure 27: Example of exposed area (highlighted in red) for cell assets (from [71]).

To determine the HT-related portion of the Security score, an ***exploitable region*** metric is established. This metric defines a set of continuous placement sites[3] that are either i) free, ii) occupied by filler cells or non-functional cells, or iii) unconnected cells. When the number of these continuous placement sites reaches a minimum threshold of 20, they are identified as an exploitable region. Furthermore, free routing tracks around the exploitable region(s) are also considered. The motivation behind this is that an adversary needs both placement and routing resources to successfully insert an HT. Consequently, there should

[3]A placement site refers to a predetermined valid position within a layout where a cell can be legally positioned. These placement sites are typically determined by factors such as the standard cell height and the contacted poly pitch.

be enough gaps in the layout or some logic that can be easily removed to accommodate the HT.

The baseline layouts, before applying any security closure techniques, have a default score of 1. Layout modifications that improve design quality and/or security would be scored within the range of $[0, 1)$, while poor modifications would be scored within the range of $(1, \infty]$.

To derive each component of the scoring formula (i.e., power) outlined in Equation 3, first the relative metric for the baseline layout should be calculated. Subsequently, the corresponding metric for the modified (secured) layout can be obtained. The score for each specific component is then computed by dividing the values of the secured version by those of the baseline version. The final score is obtained by assigning relative weights to each element and summing them. Hence, Equation 3 can be expanded as follows:

$$Score = \overbrace{\left(0.1 \times \frac{(des\_p\_total)_{\text{s}}}{(des\_p\_total)_{\text{bl}}} + 0.3 \times \frac{(des\_perf)_{\text{s}}}{(des\_perf)_{\text{bl}}} + 0.3 \times \frac{(des\_area)_{\text{s}}}{(des\_area)_{\text{bl}}} + 0.3 \times \frac{(des\_issues)_{\text{s}}}{(des\_issues)_{\text{bl}}}\right)}^{\textit{DesignQuality}} \times \left(\overbrace{\frac{1}{2} \times \left(0.5 \times \frac{(fsp\_fi\_ea\_c)_{\text{s}}}{(fsp\_fi\_ea\_c)_{\text{bl}}} + 0.5 \times \frac{(fsp\_fi\_ea\_n)_{\text{s}}}{(fsp\_fi\_ea\_n)_{\text{bl}}}\right)}^{\textit{Security (fsp/fi)}} + \overbrace{\frac{1}{2} \times \left(0.6 \times \frac{(ti\_sts)_{\text{s}}}{(ti\_sts)_{\text{bl}}} + 0.4 \times \frac{(ti\_fts)_{\text{s}}}{(ti\_fts)_{\text{bl}}}\right)}^{\textit{Security (ti)}}\right) \tag{4}$$

In this equation, des_p_total, des_perf, des_area, and des_issues represent the power, performance concerning timing violations (if any), area, and Design Rule Checks (DRCs) respectively. The terms $fsp\_fi\_ea\_c$ and $fsp\_fi\_ea\_n$ indicate the exposed area of the cell assets and the exposed area of the net assets, respectively, and ti_sts and ti_fts terms denote the exploitable regions and available routing resources (free tracks) of exploitable regions.

5.2 SALSy Techniques

This section introduces different techniques used in SALSy, the primary contribution of this study. In this section, two perspectives of SALSy will be presented: one is the pre-silicon view, compatible with open-source PDKs, and the other is the post-silicon view, more aligned with commercial PDKs. Consequently, the results obtained from a real fabricated chip implementing SALSy concepts will be provided in the following section. Comparing to open-source PDKs is crucial as it enables this work to be assessed relative to others within the framework outlined in [71].

An outline of the employed techniques and their respective sequence is depicted in Figure 28. Notably, not all techniques suitable for an open-source PDK can be applied in an actual tapeout. The color scheme adopted in the figure designates green-colored rectangles to indicate techniques fully compatible with commercial PDKs.

The techniques outlined in steps 1 through 6 are primarily used to increase the design's security against FSP/FI attacks. The last two steps, on the other hand, are mainly focused on eliminating exploitable regions to protect the design against HT insertion. To help the user determine whether the design has achieved the desired level of security, two checkpoints are included. The first checkpoint is placed after the completion of Edge Cell Placement in step 4. If the user is satisfied with the enhanced security against FSP/FI at this point, they can choose to skip steps 5 and 6. Otherwise, the techniques in steps 5 and 6 can be applied to further improve the design's security against these types of attacks. The second

checkpoint is placed after step 8, and is used to evaluate the enhanced security against HT insertion. If the security scores for the TI analysis meet the user's requirements, the layout can be considered final. If not, steps 7 and 8 can be repeated until the TI scores reach the threshold defined by the user.

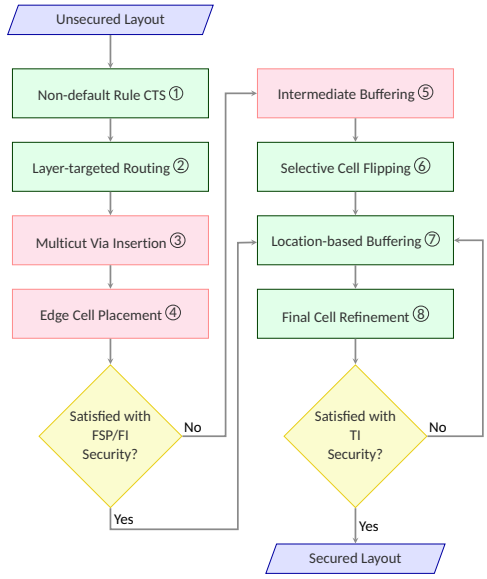


Figure 28: SALSy framework. Red boxes highlight techniques that are not feasible for the tapeout. Green boxes highlight techniques that can be used in both open-source PDKs and in the tapeout. (from [69])

5.2.1 Benchmarks

The selected benchmarks for the open-source experiment predominantly comprise crypto cores, including CAST, Camellia, MISTY, PRESENT, OpenMSP430_1, three versions of AES, SEED, TDEA, OpenMSP430_2, and SPARX [156, 157, 158].

5.2.2 Open-source PDK

Consistent with comparable academic efforts, the selected PDK/standard cell library in [71] is the Nangate 45nm Open Cell Library [159], given its unrestricted availability. The metal stacks taken into account are 6M and 10M, contingent upon the benchmark's complexity.

It is worth emphasizing that the scoring formula prioritizes a delicate balance between security and PPA, as outlined in Equation 4. Regarding the design aspect, while customized implementation scripts were utilized for each benchmark, it is important to note that these scripts primarily focused on traditional parameter exploration in physical synthesis and will not be extensively elaborated upon. The subsequent discussion will primarily center around the security aspects. Moreover, given that the scoring formula accounts for distinct metrics concerning front-side probing/fault injection (*fsp_fi*) and HT insertion (*ti*), the relevant SALSy techniques will be explained separately.

5.2.3 Countermeasures against FSP/FI

1) Non-default Rule Clock Tree Synthesis: The core idea of this strategy is to modify the default rules for CTS[4] in order to protect a broader range of assets by widening the clock distribution wires. It is worth noting that CTS routing consumes fewer resources compared to signal routing. Therefore, CTS wires can be significantly widened, often several times more than signal wires. As illustrated in Figure 30a, the enlarged clock tree can significantly cover more exposed areas under it. Often, the quality of the CTS is improved by using non-default rules.

2) Layer-targeted Routing: Recall that the exposed area metric, pertaining to both cells and nets, denotes the area of assets directly accessible from the front side. In the initial step, the objective is to shield the net assets under other non-asset nets to safeguard them against FSP/FI, as outlined in Algorithm 1. To achieve this, the lowest available metal layers[5] are exclusively assigned to the net assets (line 3). It is important to note that the minimum width for routing these asset-related wires is utilized to hide them under other nets (line 5).

Algorithm 1 Layer-targeted Routing Algorithm

1: $net\_assets \leftarrow List\_of\_net\_assets$
2: $other\_nets \leftarrow List\_of\_other\_nets$
3: $prf\_lays\_assets \leftarrow [M2, M3]$
4: $prf\_lays\_others \leftarrow [M4, M5, M6]$
5: $width\_for\_assets \leftarrow width(M2)$ ▷ This value is the minimum width according to the library
6: $width\_for\_others \leftarrow width(M2) \times 2$
7: **foreach** net **in** net_assets **do**
8: **assign** *prf_lays_assets* **to** *route_layer*
9: **assign** *width_for_assets* **to** *width_rule*
10: **end for**
11: **route** *net_assets* **with** *width_rule* **in** *route_layer*
12: **if** $(route\_err)$ **then**
13: **route** *net_assets* **with** *default_rules*
14: **end if**
15: **foreach** net **in** $other\_nets$ **do**
16: **assign** *prf_lays_others* **to** *route_layer*
17: **assign** *width_for_others* **to** *width_rule*
18: **end for**
19: **route** *other_nets* **with** *width_ruler* **in** *route_layer*
20: **if** $(route\_err)$ **then**
21: **route** *other_nets* **with** *default_rules*
22: **end if**

Subsequently, all remaining non-asset nets are designated to be routed using higher metal layers (line 4). Additionally, it is tried to opt for a wider width, different from the

[4]CTS is a crucial phase in the design workflow of digital ICs, involving the creation of a network of clock branches to efficiently distribute the clock signal across the entire circuit. By carefully constructing the network, observing delay balancing, and managing skew, timing can be improved, and power consumption can be reduced. CTS routing typically takes priority over signal routing, which is leveraged for security purposes.

[5]Metal layers are organized into a metal stack, with the lower layers typically being thinner.

default width, to enhance the chance of covering net and cell assets (line 6). If the routing tool encounters difficulties in routing the nets with the adjusted width or in the preferred metal layer, it will attempt to route them using the default width and default metal layers (lines 12-14, 20-22). It is important to note that for the physical synthesis tool employed, routing constraints are considered soft constraints, meaning that the tool will make every effort to adhere to the constraints. However, if it faces challenges, the constraints may be relaxed.

As an example, in Algorithm 1, M2 and M3 layers are dedicated exclusively to routing the net assets (line 3), while the higher metal layers M4-M6 are allocated for routing non-asset nets (line 4). In this scenario, the width of the non-asset nets is set to be twice as wide as that of the net assets (line 6). However, this value can be adjusted if additional resources become available[6]. Upon implementing this technique, congestion significantly increases, which enables more cell assets and net assets to be protected against FSP/FI, as depicted in Figure 30b.

3) Multi-cut Via Insertion: In an IC's metal stack, a vertical connection known as a *via* enables the connection between different metal layers. Typically, the physical synthesis tool optimizes routing resource utilization and minimizes congestion by employing the minimum number and smallest size of vias available for connections. However, the proposed strategy aims to deliberately increase congestion on the top of the cell assets to improve their coverage. To achieve this, the insertion of multi-cut vias between the M1 and M2 layers allows for a larger metal piece to be routed over the cell assets, leading to improving coverage, as illustrated in Figure 29. The decision to use multi-cut vias exclusively between the M1 and M2 layers is deliberate, as it avoids affecting the resources in higher metal layers, which are reserved for signal routing.

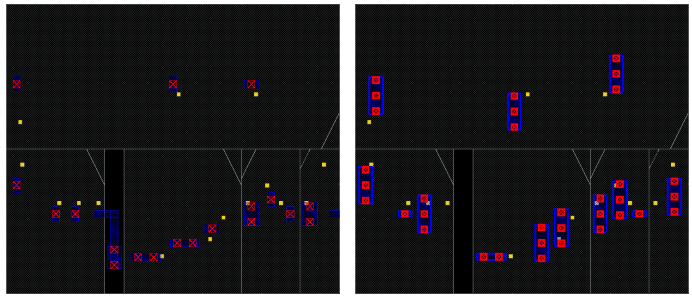


Figure 29: Using the default rules for via insertion (left) and multi-cut via insertion (right) to increase the coverage of cell assets (from [69])

4) Edge Cell Placement: In certain benchmarks, it has been observed that net assets encompass lengthy wires that extend from IO pins to their respective sinks (highlighted in green in Figure 30c). To address this, a technique is employed where the sink cell linked to the IO pins through net assets is moved to the nearest feasible position to their driver. This replacement substantially reduces the length of the net assets, increasing the chance of being covered by other nets on the upper layers since shorter nets typically exhibit fewer turns and jogs.

5) Intermediate Buffering: The aforementioned technique for shortening net assets is effective only for wires connected to IO pins, leaving other net assets vulnerable due

[6] In real ICs, the number of metal layers depends on the technology and metal stack agreed upon with the foundry. Current technologies often provide 10 or more layers.

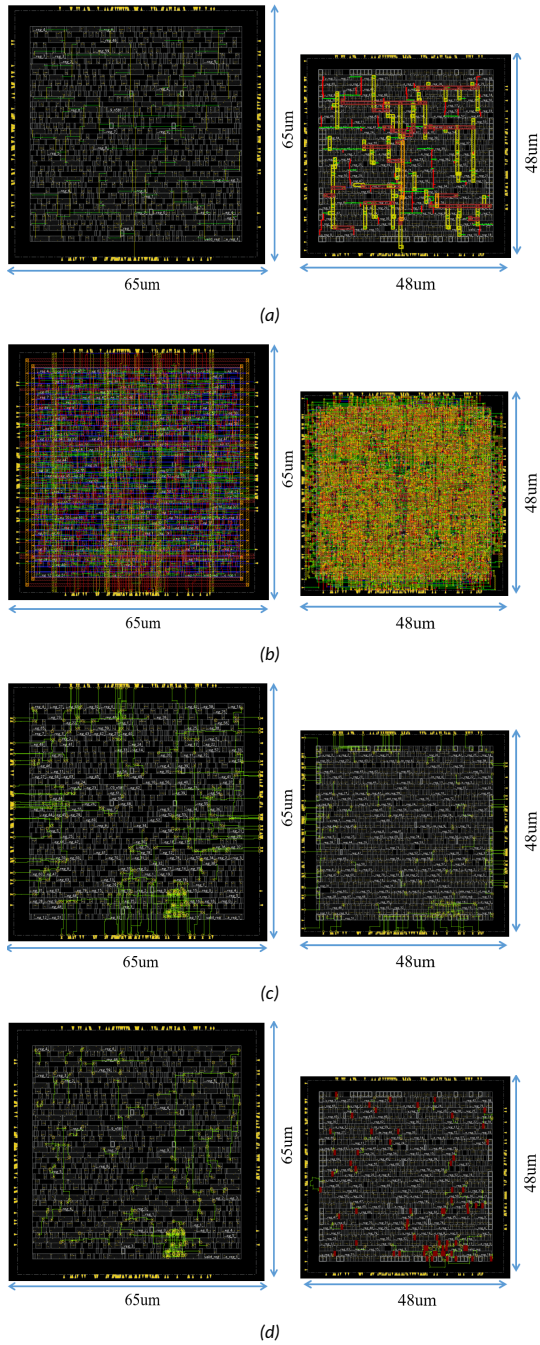


Figure 30: Different techniques used in SALSy (from [69]). The design on the left is always the BL variant, and the design on the right is always the SEC variant. a) Non-default Rule CTS, b) Increased congestion by applying Layer-targeted Routing, c) Edge Cell Placement for shortening the long net assets (highlighted in green), and d) Reducing the length of the net assets (highlighted in green) by applying Intermediate Buffering technique (added buffers appear in red).

to their extended length. When both the driver and sink are placed within the core area (the region containing all cells), addressing this challenge becomes essential. In such cases, buffers are inserted between the driver and sink to shorten the length of these lengthy net assets, as depicted in Figure 30d. It is important to note that buffer insertion can significantly impact the circuit's timing and power consumption. Therefore, this technique is implemented iteratively with multiple checkpoints. If the insertion of a buffer leads to a timing violation, the buffer can be removed, reverting the circuit to its previous state without the violation.

6) Selective Cell Flipping: In certain scenarios, the exposed area of net assets can be notably reduced by changing the orientation of the cell (i.e., flipping it over the Y axis). This makes the physical synthesis tool automatically re-route the nets connected to the flipped cell, thereby enhancing the chance of covering the net asset beneath other nets, as illustrated in Figure 31. It is important to emphasize that this technique is executed during the final stages of the proposed methodology, and only net assets with the most substantial exposed area are targeted for this adjustment.

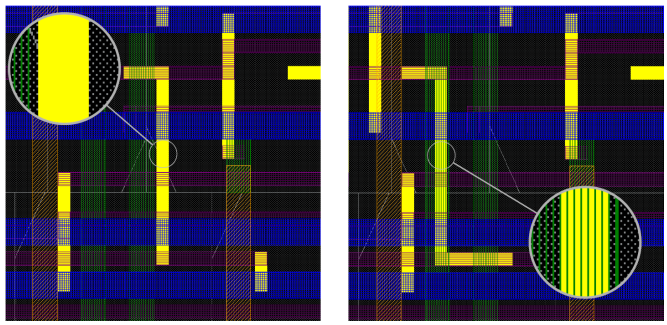


Figure 31: An example of covering a net asset by flipping the cell (from [69]): The exposed area (solid yellow regions in the left image) is totally covered by the nets in the upper metal layer(s) after the net is re-routed (right image)

5.2.4 Countermeasures against HT Insertion

This section outlines the techniques employed against HT insertion. The concept of an *exploitable region* is revisited here, which is defined as a set of continuous gaps, filler cells, disconnected cells, or non-functional cells, and could be exploited by an adversary for inserting malicious logic. Since HT components must connect to the existing design, the availability of routing resources is also a consideration in determining such regions. However, the primary focus here is on eliminating free placement sites. This is mainly because if all the gaps are eliminated, there would be no space for HT logic to be placed in the design. Consequently, available routing resources become irrelevant, as there would be no HT cells to connect to the original design.

7) Location-based Buffering: Despite reducing the design area to maximize the density of the core area, some gaps may still exist, creating large exploitable areas. Given that a continuous gap exceeding 20 placement sites qualifies as an exploitable region, a script has been developed to identify such regions and introduce buffers to either fill these gaps or reduce them to fewer than 20 sites. It is important to note that the insertion of buffers can incur additional power consumption and possibly affect timing. Nonetheless, the balance between enhanced security and the impact on PPA is considered advantageous for this specific technique.

8) Final Cell Refinement: In particular cases, the insertion of buffers may be unsuccessful due to insufficient routing resources within the congested regions. Additionally, even if successful, it could lead to the creation of timing violations. To address this, efforts are made to mitigate any remaining vulnerable areas by incrementally adjusting the adjacent cells. This straightforward method can be implemented through algorithmic strategies as outlined in the [65] framework or, in scenarios with a limited number of instances, manually by a physical design engineer.

Figure 32 illustrates the successful elimination of all exploitable regions within a design through the adoption of the mentioned techniques. It is important to clarify that eliminating all exploitable regions in the layout does not necessarily mean that there are no gaps or empty spaces left. Rather, it implies that any remaining gaps are smaller than the predefined threshold for an exploitable region.

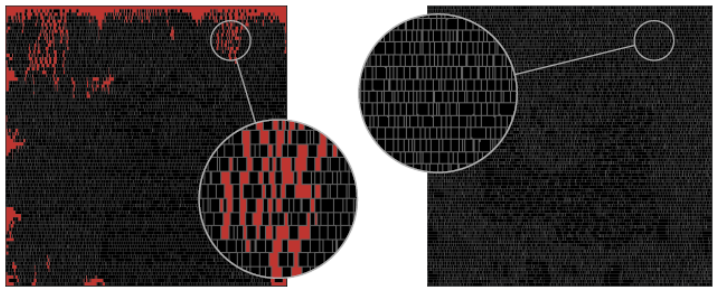


Figure 32: An example of a) design with exploitable regions (highlighted in red), and b) design with zero exploitable regions using the proposed techniques (from [69])

5.3 Scores for Open-source PDK and Comparisons

This section presents the benchmark scores achieved by employing the specified methods on the considered open-source PDK. Additionally, the results are compared with those from other studies aimed at improving the security of the same benchmark layouts. To facilitate this comparison, data from teams that participated in the security closure contest associated with the ISPD'22 conference have been compiled. A detailed version of the contest's logistics and framework is provided in [71].

The scoring equation presented in Equation 3 is designed to normalize results relative to baseline measures, attributing equal significance to both design quality and security considerations. Nevertheless, due to the presence of a multiplicative relationship between these two parts of the score, a hypothetical zero score in security—though impossible in reality—would result in a total score of zero.

Participants in the competition eventually sensed that implementing a singular metallic shield atop the entire layout effectively neutralized all security vulnerabilities. This approach is equivalent to employing a complete metal layer as a sacrificial layer. While this method does result in violations of DRC, the scoring formula regrettably does not impose penalties for such infractions: given that the security metric is nullified, the design aspect of Equation 3 becomes irrelevant. In conclusion, the ISPD'22 contest ended with several teams achieving a perfect score of zero. The final scores are outlined in Table 7. SALSy techniques are denoted as team 'K.'

It is worth mentioning that the proposed solution involving a sacrificial metal layer lacks any substantial value in practice. Its efficacy is limited to fulfilling the criteria for contest scoring and does not offer real protection against the threats under consideration. A clearer

Table 7: Overall scores of the participating teams

Benchmarks / Teams	J	N	O	E	L	A	Q	**K**
AES_1	0.764	0.025	0.000	0.000	0.271	0.000	0.000	**0.000**
AES_2	1.687	0.054	0.000	0.000	0.324	0.000	0.000	**0.000**
AES_3	1.332	0.000	0.000	0.000	0.295	0.000	0.000	**0.000**
Camellia	0.676	0.000	0.000	0.000	0.281	0.000	0.000	**0.000**
CAST	1.687	0.000	0.000	0.000	0.300	0.000	0.000	**0.000**
MISTY	3.178	0.000	0.000	0.000	0.254	0.000	0.000	**0.000**
OpenMSP430_1	0.841	0.000	0.000	0.000	0.344	0.000	0.000	**0.000**
PRESENT	0.629	0.000	0.000	0.000	0.319	0.000	0.000	**0.000**
SEED	2.203	0.000	0.000	0.000	0.207	0.000	0.000	**0.000**
TDEA	0.596	0.003	0.000	0.000	0.246	0.000	0.002	**0.000**
OpenMSP430_2	1.031	0.000	0.000	0.000	0.822	0.000	0.000	**0.000**
SPARX	0.476	0.000	0.000	0.000	0.262	0.000	0.000	**0.000**

understanding of the overheads imposed by the presented techniques can be obtained from Table 8. When considering only the design component of Equation 3, it is evident that SALSy scores are quite competitive. These findings, coupled with an assessment of the feasibility of adapting the proposed techniques for use with a commercial PDK, have prompted me to proceed with a tapeout, which is elaborated upon in the subsequent section.

Table 8: Design quality scores of the participating teams

Benchmarks / Teams	J	N	O	E	L	A	Q	**K**
AES_1	0.995	0.713	0.447	0.475	0.527	0.519	1.347	**0.481**
AES_2	3.737	0.702	0.425	0.458	0.539	0.509	0.817	**0.461**
AES_3	2.689	1.059	0.473	0.498	0.566	0.541	1.171	**0.523**
Camellia	0.753	0.746	0.398	0.420	0.470	0.418	0.960	**0.530**
CAST	1.663	0.851	0.412	0.409	0.463	0.439	0.908	**0.495**
MISTY	5.009	0.753	0.418	0.396	0.457	0.417	1.559	**0.458**
OpenMSP430_1	0.756	0.656	0.406	0.440	0.490	0.469	1.025	**0.632**
PRESENT	0.752	0.693	0.359	0.427	0.465	0.446	1.009	**0.306**
SEED	1.917	0.892	0.416	0.442	0.418	0.442	0.924	**0.522**
TDEA	0.750	0.846	0.459	0.526	0.534	0.524	0.808	**0.584**
OpenMSP430_2	0.995	0.777	0.464	0.543	0.524	0.570	0.848	**0.608**
SPARX	0.753	0.663	0.397	0.420	0.422	0.404	1.047	**0.509**

5.4 Silicon Validation of SALSy

In the prior section, various methodologies to enhance IC security were explained, anchored by specific evaluation of benchmark circuits. However, these techniques were designed for open-source PDKs. In contrast, industry-utilized commercial PDKs possess a greater level of complexity than their academic counterparts. Therefore, using commercial PDKs can increase design complexity and introduce certain practical limitations. These limitations may arise from factors such as compatibility issues and the need for specific design rules and guidelines to be followed. Consequently, in order to demonstrate the gaps and limitations of open-source PDKs for rigorous security closure assessment and offer solutions to address these issues, I decided to fabricate a chip incorporating the mentioned security features. This hands-on approach allows the community to gain valuable insights into the challenges and potential solutions when working with open-source PDKs in the context of secure chip design.

In designing the chip, the first step is to adapt the scoring system to the commercial library, which enables the evaluation of the security features of the chip using the same metrics from [71]. Next, it is needed to decide which designs to be included in the tapeout.

A small chip size (1 mm^2) has been chosen that can accommodate four designs arranged as eight blocks: four secured versions (SEC) and four baseline versions (BL). Having a pair of each benchmark on the same chip makes it possible to fairly evaluate and compare each block's security and design quality.

The selection of designs aims to contain a range of complexities and sizes, with Camellia, CAST, PRESENT, and SEED as the final candidates. The chip's floorplan, depicted in Figure 33, uses color differentiation for various blocks, while the core area is reserved for the comparison and control unit. To ensure a fair comparison, all BL version benchmarks maintain the density from the open-source experiment. In contrast, the density of the SEC versions varies as different SALSy techniques are applied. Additionally, separate power domains were created for each block, facilitating the activation of a single block at a time. This ensures the remaining blocks are powered down, allowing for accurate power consumption measurements for each block individually.

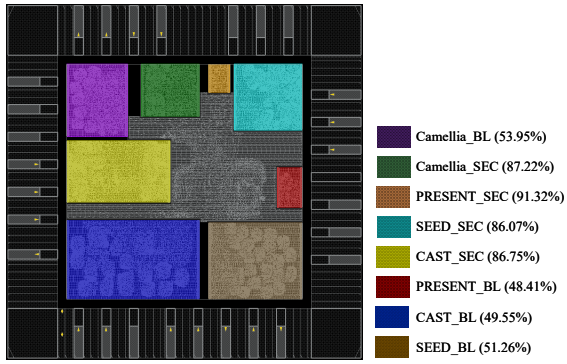


Figure 33: Floorplan view of the chip including eight blocks and density of each block (from [69])

Figure 34 presents a microscope view of the fabricated chip. The effectiveness of the proposed techniques has been validated across all four benchmarks that were demonstrated on this chip. It is important to emphasize that the presented methodology is generally applicable to any design with different functionality or scale. A notable advantage of this approach is its independence from prior knowledge of the design, as it operates at the layout stage. It is proposed that SALSy enables the possibility of assigning security closure to a separate design team, as no specific design details or characteristics are required for enhancing security. The interface between this team and the traditional physical synthesis team would be a straightforward list of assets. This separation allows for a more focused and efficient approach to security improvement during the design process.

5.4.1 Implementation for Commercial Process Design Kits

A crucial factor in enhancing the scores for HT and FSP/FI is to increase the design's density. By doing so, the possibility of HT insertion decreases due to the reduction in gaps, and more cell and net assets can be protected against FI/FSP as a result of increased wire congestion. Therefore, all designs are shrunk as much as possible before applying any specific technique (notice the smaller size of the SEC variants compared to the BL variants in Figure 33 and the remarkably high-density levels achieved for the SEC variants). In the following text, more details about the chip implementation are provided. It is worth mentioning that several scripts for the implementation flow have been released in a GitHub repository [160]. This sets the presented work apart from previous security closure attempts, which are nearly impossible to reproduce.

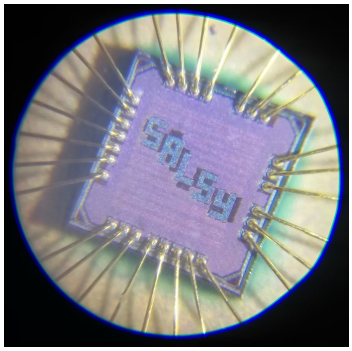


Figure 34: Microscope view of the fabricated 1mm^2 chip (from [69])

1) Non-default Rule CTS: This method can also be applied to commercial libraries but with certain limitations on the maximum wire width. The foundries set these limitations to ensure the designs remain compatible with their processing capabilities. Consequently, the enlargement of clock wires cannot be as extensive as in the open-source experiment; however, a moderate increase from the standard width is feasible. Despite its reduced efficacy relative to the open-source experiment, this technique is still employed due to its minimal impact on power consumption and other performance metrics.

2) Layer-targeted Routing: In a manner similar to the presented approach in the open-source experiment, the routing strategy delineated in Algorithm 1 is employed. The difference lies in the utilization of commercial libraries, which offer a more detailed and defined set of parameters to guarantee the accuracy of design rules and verification. Consequently, as the design's complexity increases, a more significant number of violations appear for various reasons, all in the service of ensuring the chip's quality and reliability throughout the manufacturing process. Therefore, achieving a high density (exceeding 90% for the specified 65nm technology) presents significant challenges. It becomes unfeasible to route every asset or non-asset net within their optimal metal layers as previously done in the open-source experiment. Nonetheless, despite the necessity to route some asset nets through the upper metal layers, this method continues to effectively cover a notable portion of the exposed areas of the assets.

3) Multicut Via Insertion: This method was the first one that had to be abandoned due to the strict constraints in the commercial PDK. While employing multi-cut vias for pin connections is a theoretical possibility, it leads to systemic DRC violations once the vias are connected to the wires. Given the significant challenge of addressing numerous DRC violations, adopting this method in the chip was not practical. However, this strategy may be applicable and worth reconsidering for an alternate commercial library/PDK. Typically, multi-cut vias are utilized in power routing rather than signal routing, which was the novel usage that was explored.

4) Edge Cell Placement: An important distinction between the open-source experiment and an actual tapeout is that each design was treated as a separate chip in the open-source experiment, while all the designs must be placed together on one chip in this case. This integration significantly restricts the flexibility in assigning IO pin locations for each design. These locations are typically determined during the top-level floorplanning stage, where decisions are made about which side of the block (left, right, bottom, or top) the IO pins should be placed. Taking the *PRESENT_SEC* block as an example, as depicted in Figure 33, putting the IO pins on the block's bottom edge is advantageous due to its closeness to the

centrally located control and comparison unit. This choice results in routing with fewer issues and avoids unnecessary resource utilization, leading to a more optimized floorplan.

The constraints on pin placement present a challenge for incorporating this technique into the chip's design. As illustrated in Figure 35, the closeness of the IO pins to the net assets, highlighted in white, offers only a limited space. This spatial limitation makes it infeasible to position all connected cells adjacent to their respective driver/sink pins, as doing so would lead to excessive congestion, making the design unroutable. Consequently, this technique is unsuitable for the current floorplan configuration. Nevertheless, it is important to note that this constraint does not prevent the application of this technique in different chip designs. For instance, in an open-source experiment, cells were successfully placed near their corresponding IO pins due to the square shape of the block and the availability of space around all four sides of the design (Figure 30).

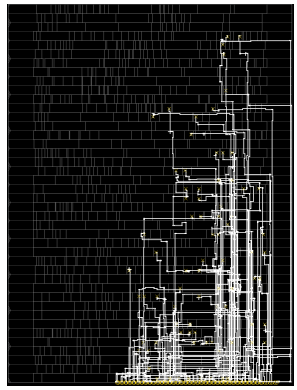


Figure 35: A design (PRESENT) with most of the IO pins on the bottom side and the net assets (highlighted in white) connected to their relative IO pin (from [69])

5) Selective Cell Flipping: This method is applicable both in the context of chip implementation and open-source experiment and presents no particular limitations. Nevertheless, its utilization is limited due to the inherent manual nature of this technique. The primary goal is to maintain a comprehensive and automated methodology, avoiding reliance on selective methods, thereby guaranteeing the comprehensive applicability of the presented work.

6) Intermediate Buffering: As mentioned in the previous section, buffer insertion can have undesirable effects on the timing and power of the design. In the open-source experiment, such issues were only considered as a negative factor in the final score to penalize the teams. However, in the actual chip, any single issue that violates the timing of the design (e.g., setup time, hold time) is considered unacceptable. Therefore, the timing closure of the design must be perfect, and the trade-off between timing issues and enhanced security is only possible as long as the timing slack remains positive. Due to this reason, this technique was replaced with a smarter buffer insertion algorithm, which is explained in the following text.

7) Location-based Buffering: As mentioned previously, the buffer insertion strategy is modified such that it can focus on filling the continuous gaps in the design, rather than shortening long net assets. This change transformed the buffer insertion technique into a location-based algorithm that targets exploitable regions instead of searching for long net assets. The sinks and drivers of the added buffers are selected from nearby cells to minimize the negative impact on the timing of the design, as illustrated in Figure 36.

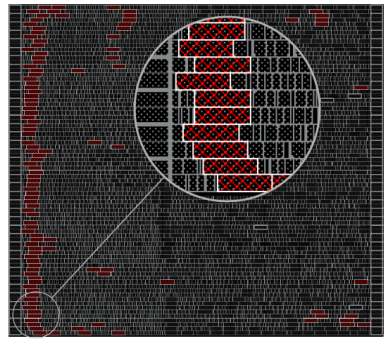


Figure 36: Added buffers (highlighted in red) using the smart algorithm to eliminate exploitable regions (from [69])

8) Final Cell Refinement: Similar to the open-source experiment, this technique is applied in the very late stages of chip implementation as a manual fix. If any exploitable region remains that can be eliminated with a few cell movements, this method can be used. However, it is decided to minimize the use of this technique in the chip implementation for two reasons: i) if an exploitable region is eliminated simply by moving cells, an adversary can potentially revert the changes to create enough space for their malicious logic, making this effort less effective in a realistic scenario; ii) it conflicts with the goal of creating an automated flow.

5.5 Results

This section presents the practical results of the chip design and measurement procedures. The Cadence suite was employed throughout the physical implementation phase, targeting a commercial 65nm CMOS technology. The results are categorized into pre-silicon and post-silicon parts. The former encompasses the data from the final layout sent for fabrication, including the block area and density. The latter part includes the actual chip metrics measured, such as power consumption.

5.5.1 Pre-silicon Results

As previously mentioned, the evaluation methods use the scoring system referenced in [71], and further details on each metric are explained below. The final scores of the presented methodology are depicted in Table 9, which clearly shows that the highest score belongs to HT insertion. This highlights the efficacy of SALSy as a preventive measure within a practical PDK environment. On the other hand, the expected lower scores in power are due to the security-enhancing buffer insertions. The power consumption consistently exceeded the baseline figures, as shown by the values exceeding 1.0 across the table. However, it is crucial to acknowledge the unavoidable trade-off between increased security and associated overheads. The power metrics in Table 9 are based on physical synthesis estimates, with exact figures detailed in Section 5.5.2.

This table demonstrates a significant decrease in the number of *exploitable regions* in the secured version for all benchmarks. This decline is especially notable, with the complete elimination of such regions in the Camellia and PRESENT benchmarks. Additionally, the CAST and SEED benchmarks show reductions of 95.3% and 90.3%, respectively. In terms of FSP/FI evaluation, the benchmarks show diverse results. The PRESENT benchmark stands out with a notable 43% reduction in the *exposed area* compared to the baseline, whereas

the CAST benchmark shows a more modest improvement of 18.5%.

Table 9: Final scores of SALSy for four different benchmarks

Metrics / Benchmarks			Camellia	CAST	PRESENT	SEED
Design Quality	DRC	des_issues	0.000	0.000	0.000	0.000
	PPA	des_perf	0.000	0.000	0.000	0.000
		des_p_total	1.184	1.072	1.161	1.041
		des_area	0.686	0.606	0.597	0.627
	Overall	des	0.467	0.419	0.439	0.417
Security	Trojan Insertion	ti_sts	0.000	0.015	0.000	0.026
		ti_fts	0.000	0.079	0.000	0.169
	Overall	ti	0.000	0.047	0.000	0.097
	FSP/FI	fsp_fi_ea_c	0.842	0.797	0.293	0.762
		fsp_fi_ea_n	0.624	0.833	0.568	0.835
	Overall	fsp_fi	0.733	0.815	0.430	0.799
Final score		OVERALL	0.171	0.181	0.094	0.187

To offer a thorough understanding of the relationship between each step of SALSy and the resulting scores, individual scores for the PRESENT benchmark are presented in Table 10 after applying each technique. This table shows that the Layer-targeted Routing technique has the most significant effect on the *fsp_fi* and overall scores, due to its considerable impact on increasing congestion. On the other hand, the Location-based Buffer Insertion technique has the most substantial impact on enhancing the *ti* score, as it drastically reduces the number of gaps in the layout. Notably, the overall trend of score improvement, as displayed in Table 10, remains consistent for all other three benchmarks.

5.5.2 Post-fabrication Results

This section details the measurement results obtained from the actual chip. The testing environment, as shown in Figure 37, consists of several components: a controller responsible for serial communication, input feeding, output reading, and data analysis; a power supply; a frequency generator providing a fast clock; and a precise measuring unit for assessing the chip's power consumption under various scenarios. The experiments are conducted on 20 packaged chips, selected from a total of 100 fabricated chips.

Verifying the Chip Functionality:

Before commencing power measurements, it is crucial to verify the proper functioning of the chips and their respective blocks. To achieve this, a Python script was developed to systematically activate each block at the target frequency while simultaneously ensuring the accuracy of the output data. All chips were found to be functional, allowing power measurements to be proceeded with. It is important to note that the target frequency for all blocks is 100 MHz, while the clock frequency for the comparison and control unit is set to 1 MHz. Additionally, a fast 100 MHz reference clock is generated by an external frequency generator, as depicted in Figure 37. The reader is reminded that total power consists of dynamic and static (leakage) power, which will be reported separately. The dynamic power results are reported at 100 MHz.

Leakage Power Measurement: Once the chip's functionality is verified, the next step involves measuring its power consumption. Initially, the Always On (AO) leakage power is evaluated. This type of power consumption refers to the power consumed by the circuit when it is in an idle or standby mode, but still powered on. This power consumption is caused by the leakage current that flows through the transistors and other components of the circuit or system, even when they are not actively switching or processing data.

Table 10: The changes in the scores of PRESENT benchmark after applying SALSy techniques in each step

Metrics / Steps			Non-default Rule CTS	Layer-targeted Routing	Location-based Buffer Insertion	Final Cell Refinement
Design Quality	DRC	des_issues	0.000	0.000	0.000	0.000
	PPA	des_perf	0.000	0.000	0.000	0.000
		des_p_total	1.018	1.138	1.159	1.161
		des_area	0.597	0.597	0.597	0.597
	Overall	des	0.404	0.434	0.436	0.439
Security	Trojan Insertion	ti_sts	0.010	0.011	0.005	0.000
		ti_fts	0.116	0.117	0.071	0.000
	Overall	ti	0.063	0.064	0.038	0.000
	FSP/FI	fsp_fi_ea_c	0.913	0.318	0.315	0.298
		fsp_fi_ea_n	0.985	0.586	0.583	0.568
	Overall	fsp_fi	0.949	0.452	0.449	0.430
Final score		OVERALL	0.204	0.112	0.106	0.094

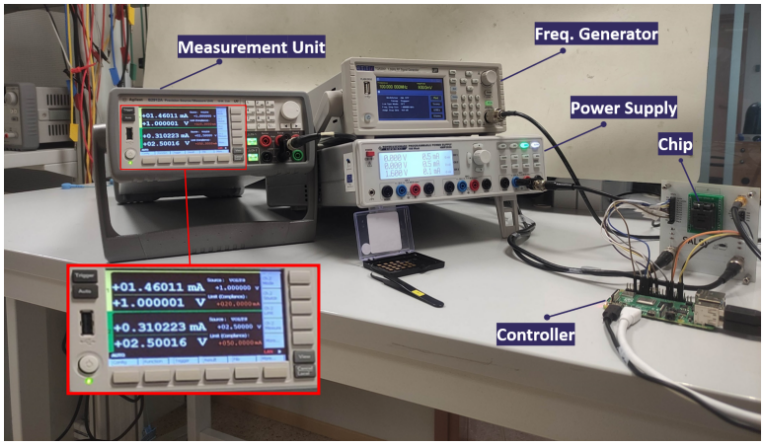


Figure 37: The testing environment for the fabricated chip

During this assessment, the inputs to the chip must be set and kept constant, without any changes or transitions. This allows the capturing of the baseline power consumption while the chip remains idle, without any specific operations underway.

Upon completing the AO leakage power assessment, the subsequent step involves evaluating leakage power for each specific block. This is accomplished by sequentially activating a single block, employing a customized configuration of input signals tailored for that block's voltage island. Each block is equipped with power switches, enabling selective activation or deactivation.

This precise power domain isolation significantly improves measurement accuracy by preventing power sharing with other blocks. Similar to the AO leakage measurement, no clock or other signals are toggled during this process. Consequently, this method accurately determines the power consumed by each individual block in isolation, providing insight into their specific power characteristics.

Figure 38 presents the results for leakage power. It is evident that different chips display unique power signatures, which can be attributed to process variation. These variations are inherent in the semiconductor fabrication process and can result in variations in power consumption among individual chips. The observed differences in leakage power emphasize the importance of process variation in chip manufacturing and underscore the necessity for comprehensive testing and analysis of power characteristics in real-world chip deployments.

Regarding static power, the overheads are on average 1.72%, 1.66%, 15.89%, and 7.24% for the PRESENT, SEED, Camellia, and CAST benchmarks, respectively.

Dynamic Power Measurement: The dynamic power measurement test is performed to evaluate the energy usage of each design block when operational. This is accomplished by sequentially activating each block and supplying them with the necessary inputs (i.e., plain text) at a clock frequency of 100MHz. These inputs are either derived from an internal register bank within the chip or provided by the host controller via the UART protocol. The results of this experiment are illustrated in Figure 39.

Across all benchmarks, the average overhead for dynamic power consumption remains below 3%. Specifically, the PRESENT, SEED, Camellia, and CAST benchmarks exhibit overheads of 0.79%, 0.86%, 2.02%, and 1.96%, respectively.

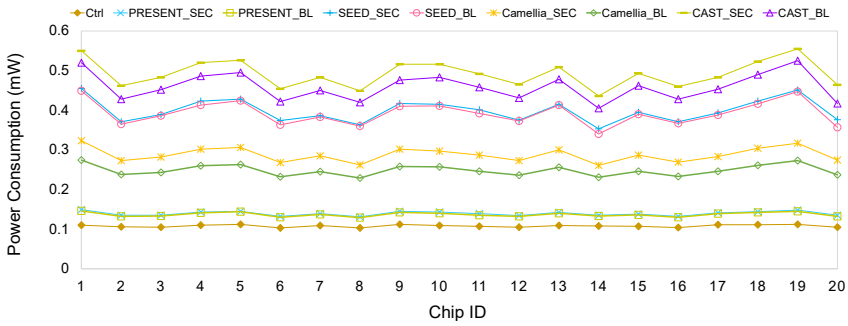


Figure 38: The measured leakage power (in mW) for 20 fabricated chips (from [69])

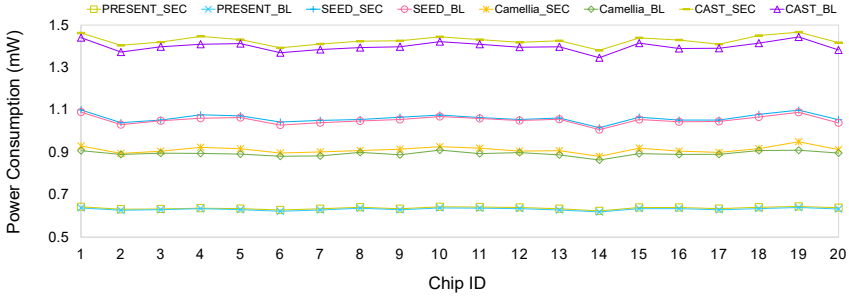


Figure 39: The measured dynamic power (in mW) for 20 fabricated chips (from [69])

5.6 SALSy Versus Other Techniques

SALSy is designed to counter post-design attacks, with its effectiveness measured using metrics from [71]. Nevertheless, it is suggested that redefining certain metrics may enhance the realism of the evaluation, thereby making the findings more applicable to industry practices. For instance, the threshold of 20 continuous gaps for the exploitable area might be overly optimistic, given that small HTs occupy only a few placement sites [161]. Additionally, in addressing FSP/FI threats, the goal is to protect the design. In an ideal scenario, attempts by attackers to compromise sensitive data, such as drilling holes (milling), should render the chip inoperable due to damage to protective nets above the sensitive ones. Consequently, simplistic defense strategies like covering the entire core area with a large metal plate should not be considered viable solutions, as such measures would not prevent attacks.

Furthermore, the scripted nature of SALSy reflects a deliberate decision that aligns with the scalability requirements of the industry, extending to even the most advanced technology nodes. While all proposed techniques can be readily adapted to sub-65nm technology, there are additional constraints for more advanced nodes. For instance, Non-default Rule CTS and Layer-targeted Routing rely on selecting wider wires, but there may be limited width options available. In older technologies, foundry recipes were more forgiving, allowing for virtually any width between 1X and 20X. However, in modern FinFET technology, available widths may be more discrete, such as 1X, 1.4X, 1.6X, and above. Thus, what was previously a continuum of valid widths has now become a discrete set.

While the concept of designing a security-aware place and route engine may hold academic appeal, implementing such a system could face scalability challenges when

applied to real-world, large-scale chip designs. Opting for a scripted approach with SALSy underscores its flexibility, enabling adaptation to diverse design sizes and complexities, including those featuring memory or analog macros. Notably, dealing with macros presents its own set of challenges, as security engineers have limited control over regions around macros due to high wire congestion. Consequently, employing techniques like Location-based Buffering in these areas poses challenges. However, it is worth noting that this issue also poses difficulties for potential attackers, as inserting malicious logic (i.e., HTs) in such congested layout regions is inherently challenging.

Additionally, standard deviation values for the leakage power consumption of both the baseline and secured versions of each block were calculated. For instance, the minimum values of standard deviation are 5.41×10^{-3} and 5.49×10^{-3} for the baseline and secured versions of the PRESENT benchmark, respectively. Conversely, the highest values of standard deviation are 3.46×10^{-2} for the baseline and 3.39×10^{-2} for the secured versions of the CAST benchmark. These results underscore that the SALSy approach demonstrates sensitivity to process variation comparable to that of the conventional security-unaware flow.

In addition, SALSy has been benchmarked against the most closely related existing works to provide readers with a clearer understanding of the significant differences in power, timing, area, and density promoted by SALSy. As illustrated in Table 11, this work stands out as the sole research that has verified the proposed techniques through practical implementation in silicon. In contrast, all other works aim for security closure, with some encountering various issues related to the utilization of limited PDKs/libraries. The ☺ symbol denotes improvement, the ☹ symbol signifies a decline in the metrics introduced, and the 😐 symbol indicates negligible changes after applying the individual technique. The term 'N/A' is used when the authors have not reported a metric. It is worth noting that an increase in density is considered beneficial as it enhances the security of the design in the face of the threats under consideration.

Table 11: Comparison of this work (SALSy) with the previous techniques

Ref.	Technique	Implications				Validated?
[155]	Internal Shielding	Power ☹	Timing ☹	Area ☹	Density ☺	✗
[66]	TroMUX	Power ☹	Timing ☹	Area 😐	Density ☺	✗
[130]	BISA	Power ☹	N/A	Area 😐	Density ☺	✗
[132]	Layout Filling	Power ☹	Timing ☹	Area 😐	Density ☺	✗
[67]	DEFense Framework	Power 😐	Timing ☹	Area 😐	Density ☺	✗
[65]	ASSURER	Power 😐	N/A	Area ☺	Density ☺	✗
[162]	T-TER	Power 😐	Timing 😐	Area 😐	Density 😐	✗
[68]	GDSII-Guard	Power ☹	Timing ☹	Area 😐	Density 😐	✗
	This thesis (SALSy)	Power ☹	Timing ☺	Area ☺	Density ☺	✓

6 Conclusions and Future Directions

In the modern era of technology, high-performance ICs have become essential across a wide range of applications, from AI and the IoT to automotive and aerospace sectors. However, the fabrication of these advanced ICs demands significant investment, leading to the globalization of the IC supply chain. This globalization, in turn, introduces various potential threats, including HTs, IP piracy, counterfeiting, and reverse engineering. This thesis addressed these challenges by developing novel methods and techniques to design ICs with inherent security considerations.

Chapter 3 introduced a novel approach to enhancing the security of digital designs through the reuse of verification assertions, specifically for HT detection. The chapter explained how assertions can be leveraged as online monitors and introduces a security metric and an assertion selection methodology utilizing the advanced capabilities of the Cadence JasperGold SPV tool. A comprehensive analysis of experimental outcomes demonstrated the adaptability and scalability of this method to industry-relevant circuits. The chapter concluded by highlighting the practicality of this detection solution, which is independent of the specific activation mechanisms of HTs, thus offering a versatile security enhancement for digital designs.

Chapter 4 presented a comprehensive method to enhance IC security throughout the back-end design process. It proposed a novel technique for incorporating online monitors during physical synthesis, adding an extra layer of protection at the back-end phase. While the integration of online monitors is not new, this work's unique contribution lies in directly incorporating checkers into the layout during the back-end phase, considering front-end inserted assertions. This strategy offers a more comprehensive security solution that spans both design phases. The flexibility of employing either technique independently allows for a customizable and adaptable solution to enhance IC security. Future work could focus on optimizing this methodology to reduce area and power overheads and incorporating path awareness for better utilization of positive setup slack time.

Chapter 5 introduced Security-Aware Layout Synthesis (SALSy), a new methodology enabling IC design with inherent security considerations. SALSy balances PPA metrics, and security. However, since commercial layout synthesis tools lack direct settings for security, SALSy adapts existing algorithms for placement, routing, CTS, and other tasks to make them security-aware. This strategy enhances IC security against both fabrication-time and post-fabrication adversarial acts, including HT insertion, FI, and probing. The methodology has been validated through a silicon-based demonstration, confirming its compatibility and effectiveness with a commercial PDK and library. SALSy achieves enhanced security with minimal impact on power consumption, maintaining a balance between security and efficiency. Although effective against HT insertion, future research will focus on improving defenses against fault injection and probing attacks, automating selective techniques, and introducing new evaluation metrics for a more comprehensive assessment.

This thesis has made significant contributions to the field of IC security, particularly in HT detection, enhancing security in the IC design process, and developing security-aware layout synthesis. The proposed methods and techniques, validated through extensive experimentation and simulation, have proven effective in mitigating various security threats. These findings provide a potent framework for developing more secure ICs and guiding future research in this critical area. By integrating security considerations into every phase of IC design, this work lays the groundwork for creating more secure ICs, essential for the technological advancements of the modern era.

In conclusion, the innovative approaches developed in this thesis offer a comprehensive and adaptable suite of solutions for enhancing IC security. They address the growing need

for secure protection mechanisms in the face of an increasingly globalized and vulnerable supply chain. As technology continues to grow, the insights and methodologies presented here will serve as a crucial foundation for ongoing advancements in hardware security, ensuring that the future of technology remains secure and resilient.

List of Figures

1 The estimated sales volume of the ICs based on the estimated end-market product volume (from [14]) 12
2 The market share of the selected leading brands of the total market size in the year 2022 (from [14]) 13
3 Different phases of IC's life cycle from design to market. 14
4 Different verification methods as part of modern design and verification flows (from [75]). 18
5 Structure of the thesis. 19
6 An overall view of different steps in IC design 21
7 Overall view of IC fabrication flow (from [80]) 23
8 Silicon ingots (top) and wafers (bottom) of different diameters (from [81]) .. 24
9 HT Taxonomy based on trigger and payload types (from [87]) 26
10 An overview of different protection methods against HT (adopted from [88]) 28
11 The process of delayering an IC by removing each layer of die (from [90]) .. 29
12 An example of a) original design, b) nodes covered by bound assertion Assr_1, and c) nodes covered by bound assertion Assr_2 (from [88]) 37
13 Optimization flow for selecting the assertions to be used as security checkers (adopted from [73]) 40
14 PPA overheads imposed by different assertions on the B19-T500 benchmark from Trust-Hub (from [73]) 41
15 SC percentage for the Register Top modules of OpenTitan IPs (from [88]) ... 42
16 The figures for a) PPA overheads imposed by different assertions, b) Number of covered nodes for the individual assertion of `alert_handler` IP (from [73]) 43
17 SC percentage for different assertion sets of `Alert Handler` IP (from [73]) 44
18 Different stages of IC design: The design house and the test house are considered trusted, while the foundry is assumed to be untrusted. 45
19 Illustrative example of a block layout within a chip (from [88]) 47
20 An example of a) design before adding online monitors, and b) design with the protection logic (D11, D14, and D18 as the duplicates and V18 as the voter) to protect the uncovered gates (11, 14, and 18) 48
21 An overall flow of integrating online monitors during physical synthesis (from [88]) 48
22 Calculated SC percentage for the different assertions in selected IPs of OpenTitan: a) alert_handler_esc, b) alert_handler, and c) flash_phy_rd (from [88]) 50
23 Changes in setup slack after each round of adding the online monitors for different IPs (from[88]) 54
24 The impact of adding online monitors on the length of different metal layers in the protected layouts for different IPs (from [88]) 55
25 The layout view of selected IPs, whereas the left images in each row represent the placement configuration before adding online monitors, while the right one represents the view of each design after integrating the online monitors: a) `alert_handler_esc_timer`, b) `ast_reg_top`, c) `flash_ctrl_core_reg_top`, d) `keymgr_reg_top`, and e) `nmi_reg_top` (from [88]) 56

26 The layout view of selected IPs, whereas the left images in each row represent the routed view before adding online monitors, while the right one represents the view of each design after integrating the online monitors: a) `alert_handler_esc_timer`, b) `ast_reg_top`, c) `flash_ctrl_core_reg_top`, d) `keymgr_reg_top`, and e) `nmi_reg_top` (from [88]) 57
27 Example of exposed area (highlighted in red) for cell assets (from [71]). 60
28 SALSy framework. Red boxes highlight techniques that are not feasible for the tapeout. Green boxes highlight techniques that can be used in both open-source PDKs and in the tapeout. (from [69]) 62
29 Using the default rules for via insertion (left) and multi-cut via insertion (right) to increase the coverage of cell assets (from [69]) 64
30 Different techniques used in SALSy (from [69]). The design on the left is always the BL variant, and the design on the right is always the SEC variant. a) Non-default Rule CTS, b) Increased congestion by applying Layer-targeted Routing, c) Edge Cell Placement for shortening the long net assets (highlighted in green), and d) Reducing the length of the net assets (highlighted in green) by applying Intermediate Buffering technique (added buffers appear in red). 65
31 An example of covering a net asset by flipping the cell (from [69]): The exposed area (solid yellow regions in the left image) is totally covered by the nets in the upper metal layer(s) after the net is re-routed (right image). 66
32 An example of a) design with exploitable regions (highlighted in red), and b) design with zero exploitable regions using the proposed techniques (from [69]) 67
33 Floorplan view of the chip including eight blocks and density of each block (from [69]) 69
34 Microscope view of the fabricated 1mm^2 chip (from [69]) 70
35 A design (PRESENT) with most of the IO pins on the bottom side and the net assets (highlighted in white) connected to their relative IO pin (from [69]) 71
36 Added buffers (highlighted in red) using the smart algorithm to eliminate exploitable regions (from [69]) 72
37 The testing environment for the fabricated chip 75
38 The measured leakage power (in mW) for 20 fabricated chips (from [69]) .. 76
39 The measured dynamic power (in mW) for 20 fabricated chips (from [69]) . 76

List of Tables

1 Considered assertions for detecting HTs on B19-T500 benchmark 35
2 Considered assertions for Register Top modules of different IPs in OpenTitan SoC 39
3 SC for the synthesized assertions bound to B19−T500 benchmark 42
4 The impact of adding online monitors on the security of selected IPs 51
5 The impact of adding online monitors on the PPA metrics of selected IPs ... 53
6 Comparison of Different Detection and DfHT Techniques..... 58
7 Overall scores of the participating teams 68
8 Design quality scores of the participating teams 68
9 Final scores of SALSy for four different benchmarks..... 73
10 The changes in the scores of PRESENT benchmark after applying SALSy techniques in each step 74
11 Comparison of this work (SALSy) with the previous techniques 77

References

[1] I. Bojanova, "The digital revolution: What's on the horizon?," *IT Professional*, vol. 16, no. 1, pp. 8–12, 2014.

[2] IEEE Digital Reality, "The impacts that digital transformation has on society," last accessed on Apr. 03, 2024. Available at: `https://digitalreality.ieee.org/publications/impacts-of-digital-transformation`.

[3] Y.-Q. Lv, Q. Zhou, Y.-C. Cai, and G. Qu, "Trusted integrated circuits: The problem and challenges," *Journal of Computer Science and Technology*, vol. 29, pp. 918–928, Sep 2014.

[4] S. A. Campbell, "An introduction to microelectronic fabrication," in *Fabrication Engineering at the Micro and Nanoscale (3rd Edition)*, Oxford University Press, 2008.

[5] Yole Intelligence, "High-end performance packaging 2023," last accessed on Apr. 03, 2024. Available at: `https://www.yolegroup.com/product/report/high-end-performance-packaging-2023`.

[6] F. Alan, "TSMC's new 2nm chip production fab will cost it how much?," last accessed on Apr. 03, 2024. Available at: `https://www.phonearena.com/news/tsmc-to-spend-fortune-on-2nm-production-fab_id140626`.

[7] MacRumors, "Apple books nearly 90% of tsmc's 3nm production capacity for this year," last accessed on Apr. 03, 2024. Available at: `https://www.macrumors.com/2023/05/15/apple-tsmc-3nm-production-capacity/`.

[8] G. Moore, "Cramming more components onto integrated circuits," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 82–85, 1998.

[9] S. Sze and K. K. Ng, *Physics of Semiconductor Devices*. John Wiley & Sons, Ltd, 3rd ed., 2006.

[10] A. S. Tanenbaum and D. J. Wetherall, *Computer Networks*. Prentice Hall Press, 5th ed., 2010.

[11] J. L. Hennessy and D. A. Patterson, *Computer Architecture, Sixth Edition: A Quantitative Approach*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 6th ed., 2017.

[12] N. Smith and A. Webb, *Introduction to Medical Imaging: Physics, Engineering and Clinical Applications*. Cambridge Texts in Biomedical Engineering, Cambridge University Press, 2010.

[13] N. Zaman, *Automotive Electronics Design Fundamentals*. Springer International Publishing, 2015.

[14] Statista, "Integrated circuits - worldwide," last accessed on Apr. 03, 2024. Available at: `https://www.statista.com/outlook/tmo/semiconductors/integrated-circuits/worldwide`.

[15] International Roadmap for Devices and Systems, "Semiconductors and artificial intelligence," last accessed on Apr. 03, 2024. Available at: `https://irds.ieee.org/topics/semiconductors-and-artificial-intelligence`.

[16] ElectronicsHub, "Integrated circuits-types , uses," last accessed on Apr. 03, 2024. Available at: https://www.electronicshub.org/integrated-circuits-types-uses/.

[17] C. A. Johan, G. Dieter, H. Guido, H. Denis, and H. Arndt, "How does the semiconductor industry landscape look today?," last accessed on Apr. 03, 2024. Available at: https://www.kearney.com/industry/technology/article/-/insights/how-does-the-semiconductor-industry-landscape-look-today.

[18] I. Kuon and J. Rose, "Measuring the gap between fpgas and asics," in *Proceedings of the 2006 ACM/SIGDA 14th International Symposium on Field Programmable Gate Arrays*, p. 21–30, 2006.

[19] A. Matthew, "Supply chain threats against integrated circuits," last accessed on Apr. 03, 2024. Available at: https://www.intel.com/content/dam/www/central-libraries/us/en/documents/2022-08/supply-chain-threats-against-integrated-circuits-whitepaper-july2020.pdf.

[20] T. D. Perez and S. Pagliarini, "Hardware trojan insertion in finalized layouts: From methodology to a silicon demonstration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 7, pp. 2094–2107, 2023.

[21] S. E. Quadir, J. Chen, D. Forte, N. Asadizanjani, S. Shahbazmohamadi, L. Wang, J. Chandy, and M. Tehranipoor, "A survey on chip to system reverse engineering," *J. Emerg. Technol. Comput. Syst.*, vol. 13, apr 2016.

[22] S. M. Saeed, A. Zulehner, R. Wille, R. Drechsler, and R. Karri, "Reversible circuits: Ic/ip piracy attacks and countermeasures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2523–2535, 2019.

[23] F. Koushanfar, "Integrated circuits metering for piracy protection and digital rights management: An overview," in *Proceedings of the 21st Edition of the Great Lakes Symposium on Great Lakes Symposium on VLSI*, GLSVLSI '11, (New York, NY, USA), p. 449–454, Association for Computing Machinery, 2011.

[24] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1207–1228, 2014.

[25] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proceedings of the IEEE*, vol. 100, no. 11, pp. 3056–3076, 2012.

[26] H. Wang, D. Forte, M. M. Tehranipoor, and Q. Shi, "Probing attacks on integrated circuits: Challenges and research opportunities," *IEEE Design & Test*, vol. 34, no. 5, pp. 63–71, 2017.

[27] A. Javeed, C. Yilmaz, and E. Savas, "Microarchitectural side-channel threats, weaknesses and mitigations: A systematic mapping study," *IEEE Access*, pp. 48945–48976, 2023.

[28] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, vol. 43, no. 10, pp. 39–46, 2010.

[29] C. Helfmeier, D. Nedospasov, C. Tarnovsky, J. S. Krissler, C. Boit, and J.-P. Seifert, "Breaking and entering through the silicon," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, (New York, NY, USA), p. 733–744, Association for Computing Machinery, 2013.

[30] B. Lippmann, A.-C. Bette, M. Ludwig, J. Mutter, J. Baehr, A. Hepp, H. Gieser, N. Kovač, T. Zweifel, M. Rasche, and O. Kellermann, "Physical and functional reverse engineering challenges for advanced semiconductor solutions," in *Proceedings of the 2022 Conference & Exhibition on Design, Automation & Test in Europe*, DATE '22, (Leuven, BEL), p. 796–801, European Design and Automation Association, 2022.

[31] R. S. Rajarathnam, Y. Lin, Y. Jin, and D. Z. Pan, "ReGDS: A reverse engineering framework from GDSII to gate-level netlist," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 154–163, 2020.

[32] R. Torrance and D. James, "The state-of-the-art in ic reverse engineering," in *Cryptographic Hardware and Embedded Systems - CHES 2009* (C. Clavier and K. Gaj, eds.), (Berlin, Heidelberg), pp. 363–381, Springer Berlin Heidelberg, 2009.

[33] M. Eslami, B. Ghavami, M. Raji, and A. Mahani, "A survey on fault injection methods of digital integrated circuits," *Integration*, vol. 71, pp. 154–163, 2020.

[34] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *2019 IEEE Symposium on Security and Privacy (SP)*, pp. 1–19, 2019.

[35] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown: Reading kernel memory from user space," in *27th USENIX Security Symposium (USENIX Security 18)*, pp. 973–990, 2018.

[36] J. Simpson, "Asml halts hi-tech chip-making exports to china reportedly after us request," last accessed on May 08, 2024. Available at: `https://www.theguardian.com/technology/2024/jan/02/asml-halts-hi-tech-chip-making-exports-to-china-reportedly-after-us-request`.

[37] European Union Agency for Cybersecurity, "ENISA threat landscape 2023," last accessed on Apr. 03, 2024. Available at: `https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023`.

[38] Pricewaterhouse Coopers, "Global economic crime and fraud survey 2020," last accessed on May. 20, 2024. Available at: `https://www.pwc.com/gx/en/services/forensics/economic-crime-survey.html`.

[39] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[40] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty, "Towards trojan-free trusted ics: Problem analysis and detection scheme," in *2008 Design, Automation and Test in Europe*, pp. 1362–1365, 2008.

[41] X. Chen, G. Qu, and A. Cui, "Practical IP watermarking and fingerprinting methods for ASIC designs," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, 2017.

[42] H. Huang, A. Boyer, and S. B. Dhia, "The detection of counterfeit integrated circuit by the use of electromagnetic fingerprint," in *2014 International Symposium on Electromagnetic Compatibility*, pp. 1118–1122, 2014.

[43] Y. Jin, E. Love, and Y. Makris, *Design for Hardware Trust*, pp. 365–384. New York, NY: Springer New York, 2012.

[44] M. Yasin, J. Rajendran, and O. Sinanoglu, "Trustworthy hardware design: Combinational logic locking techniques," Springer, Cham, 2019.

[45] O. Harrison and J. Waldron, "Practical symmetric key cryptography on modern graphics hardware," in *Proceedings of the 17th Conference on Security Symposium*, p. 195–209, 2008.

[46] A. B. Kahng, J. Lach, W. H. Mangione-Smith, S. Mantik, I. L. Markov, M. Potkonjak, P. Tucker, H. Wang, and G. Wolfe, "Watermarking techniques for intellectual property protection," in *Proceedings of the 35th Annual Design Automation Conference*, DAC '98, (New York, NY, USA), p. 776–781, Association for Computing Machinery, 1998.

[47] M. Khan and S. Tragoudas, "Rewiring for watermarking digital circuit netlists," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 7, pp. 1132–1137, 2005.

[48] M. Lewandowski, R. Meana, M. Morrison, and S. Katkoori, "A novel method for watermarking sequential circuits," in *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, pp. 21–24, 2012.

[49] T. D. Perez and S. Pagliarini, "A survey on split manufacturing: Attacks, defenses, and challenges," *IEEE Access*, vol. 8, pp. 184013–184035, 2020.

[50] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?," in *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1259–1264, 2013.

[51] A. Sengupta, M. Nabeel, J. Knechtel, and O. Sinanoglu, "A new paradigm in split manufacturing: Lock the feol, unlock at the beol," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 414–419, 2019.

[52] Y. Wang, P. Chen, J. Hu, and J. J. Rajendran, "The cat and mouse in split manufacturing," in *Proceedings of the 53rd Annual Design Automation Conference*, DAC '16, (New York, NY, USA), Association for Computing Machinery, 2016.

[53] K. Zamiri Azar, H. Mardani Kamali, H. Homayoun, and A. Sasan, "Threats on logic locking: A decade later," in *GLSVLSI '19: Proceedings of the 2019 on Great Lakes Symposium on VLSI*, p. 471–476, 2019.

[54] Y. Xie and A. Srivastava, "Delay locking: Security enhancement of logic locking against IC counterfeiting and overproduction," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2017.

[55] M. Yasin and O. Sinanoglu, "Evolution of logic locking," in *2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 1–6, 2017.

[56] J. Sweeney, V. Mohammed Zackriya, S. Pagliarini, and L. Pileggi, "Latch-based logic locking," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 132–141, 2020.

[57] Y. Liu, M. Zuzak, Y. Xie, A. Chakraborty, and A. Srivastava, "Strong Anti-SAT: Secure and effective logic locking," in *2020 21st International Symposium on Quality Electronic Design (ISQED)*, pp. 199–205, 2020.

[58] M. Yasin, B. Mazumdar, J. J. V. Rajendran, and O. Sinanoglu, "Sarlock: Sat attack resistant logic locking," in *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 236–241, 2016.

[59] Y. Xie and A. Srivastava, "Anti-sat: Mitigating sat attack on logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 2, pp. 199–207, 2019.

[60] S. D. Hampton and A. J. Bailey, "Intellectual property case filing trends over the last decade," last accessed on May. 20, 2024. Available at: `https://www.hamptonip.com/articles/post/intellectual-property-case-filing-trends-over-the-last-decade/`.

[61] V. Gohil, H. Guo, S. Patnaik, and J. Rajendran, "Attrition: Attacking static hardware trojan detection techniques using reinforcement learning," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, p. 1275–1289, 2022.

[62] X. Wei, J. Zhang, and G. Luo, "Rethinking ic layout vulnerability: Simulation-based hardware trojan threat assessment with high fidelity," in *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 163–163, 2024.

[63] J. Bhandari, L. Mankali, M. Nabeel, O. Sinanoglu, R. Karri, and J. Knechtel, "Beware your standard cells! on their role in static power side-channel attacks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–14, 2024.

[64] F. Wang, Q. Wang, L. Alrahis, B. Fu, S. Jiang, X. Zhang, O. Sinanoglu, T.-Y. Ho, E. F. Y. Young, and J. Knechtel, "Trolloc: Logic locking and layout hardening for ic security closure against hardware trojans," 2024.

[65] G. Guo, H. You, Z. Tang, B. Li, C. Li, and X. Zhang, "Assurer: A ppa-friendly security closure framework for physical design," in *2023 28th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 504–509, 2023.

[66] F. Wang, Q. Wang, B. Fu, S. Jiang, X. Zhang, L. Alrahis, O. Sinanoglu, J. Knechtel, T.-Y. Ho, and E. F. Young, "Security closure of ic layouts against hardware trojans," in *Proceedings of the 2023 International Symposium on Physical Design*, ISPD '23, (New York, NY, USA), p. 229–237, Association for Computing Machinery, 2023.

[67] J. Knechtel, J. Gopinath, J. Bhandari, M. Ashraf, H. Amrouch, S. Borkar, S.-K. Lim, O. Sinanoglu, and R. Karri, "Security closure of physical layouts iccad special session paper," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2021.

[68] X. Wei, J. Zhang, and G. Luo, "Gdsii-guard: Eco anti-trojan optimization with exploratory timing-security trade-offs," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2023.

[69] M. Eslami, T. Perez, and S. Pagliarini, "Salsy: Security-aware layout synthesis," 2023. Available at: `https://arxiv.org/abs/2308.06201v2`.

[70] M. Eslami, J. Knechtel, O. Sinanoglu, R. Karri, and S. Pagliarini, "Benchmarking advanced security closure of physical layouts: Ispd 2023 contest," in *Proceedings of the 2023 International Symposium on Physical Design*, ISPD '23, (New York, NY, USA), p. 256–264, Association for Computing Machinery, 2023.

[71] J. Knechtel, J. Gopinath, M. Ashraf, J. Bhandari, O. Sinanoglu, and R. Karri, "Benchmarking security closure of physical layouts: Ispd 2022 contest," ISPD '22, (New York, NY, USA), p. 221–228, Association for Computing Machinery, 2022.

[72] Y. Li, W. Wu, L. Hou, and H. Cheng, "A study on the assertion-based verification of digital ic," in *2009 Second International Conference on Information and Computing Science*, vol. 2, pp. 25–28, 2009.

[73] M. Eslami, T. Ghasempouri, and S. Pagliarini, "Reusing verification assertions as security checkers for hardware trojan detection," in *2022 23rd International Symposium on Quality Electronic Design (ISQED)*, pp. 1–6, 2022.

[74] K. Alatoun, B. Shankaranarayanan, S. M. Achyutha, and R. Vemuri, "Soc trust validation using assertion-based security monitors," in *2021 22nd International Symposium on Quality Electronic Design (ISQED)*, pp. 496–503, 2021.

[75] E. Seligman, T. Schubert, and M. V. A. K. Kumar, "Chapter 1 - formal verification: from dreams to reality," in *Formal Verification (Second Edition)*, pp. 1–23, 2023.

[76] A. T. Sonny and S. Lakshmiprabha, "Ovl, psl, sva: Assertion based verification using checkers and standard assertion languages," in *2013 International Conference on Advanced Computing and Communication Systems*, pp. 1–4, 2013.

[77] F. Semiconductor, "A guide to ic design flow," last accessed on Jul. 11, 2024. Available at: `https://fpt-semiconductor.com/blogs/a-guide-to-ic-design-flow/`.

[78] C. Mead and L. Conway, *Introduction to VLSI Systems*. Addison-Wesley series in computer science and information processing, Addison-Wesley, 1980.

[79] C. E. Shannon, "A mathematical theory of communication," *Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[80] Techovedas, "10 fabrication steps to build a semiconductor chip," last accessed on Apr. 03, 2024. Available at: `https://techovedas.com/10-fabrication-steps-to-build-a-semiconductor-chip/`.

[81] SUMCO, "About silicon wafers," last accessed on Apr. 03, 2024. Available at: `https://www.sumcosi.com/english/products/about.html`.

[82] M. Madou, *Fundamentals of Microfabrication: The Science of Miniaturization, Second Edition*. Taylor & Francis, 2002.

[83] R. Pierret, *Advanced Semiconductor Fundamentals*. Modular series on solid state devices, Prentice Hall, 2003.

[84] J. Lau, *Ball Grid Array Technology*. Electronic packaging and interconnection series, McGraw-Hill, 1995.

[85] D. Wang, "The golden age of ic design in taiwan," last accessed on May. 20, 2024. Available at: `https://nextrendsasia.org/the-golden-age-of-ic-design-in-taiwan/`.

[86] J. Vosatka, *Introduction to Hardware Trojans*, pp. 15–51. 2018.

[87] T. Trippel, K. G. Shin, K. B. Bush, and M. Hicks, "Icas: an extensible framework for estimating the susceptibility of ic layouts to additive trojans," in *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1742–1759, 2020.

[88] M. Eslami, T. Ghasempouri, and S. Pagliarini, "Scarf: Securing chips with a robust framework against fabrication-time hardware trojans," *IEEE Transactions on Computers*, pp. 1–14, 2024.

[89] S. Bhunia and M. Tehranipoor, *Hardware Security - A Hands-On Learning Approach*. Elsevier, 2019.

[90] IC Failure Analysis Lab, "Delayering / parallel lapping," last accessed on Apr. 03, 2024. Available at: `https://icfailureanalysis.com/delayering-parallel-lapping`.

[91] Y. Jin, B. Yang, and Y. Makris, "Cycle-accurate information assurance by proof-carrying based signal sensitivity tracing," in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 99–106, 2013.

[92] X. Guo, R. G. Dutta, Y. Jin, F. Farahmandi, and P. Mishra, "Pre-silicon security verification and validation: A formal perspective," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2015.

[93] M. Qin, W. Hu, D. Mu, and Y. Tai, "Property based formal security verification for hardware trojan detection," in *2018 IEEE 3rd International Verification and Security Workshop (IVSW)*, pp. 62–67, 2018.

[94] G. Shrestha and M. S. Hsiao, "Ensuring trust of third-party hardware design with constrained sequential equivalence checking," in *2012 IEEE Conference on Technologies for Homeland Security (HST)*, pp. 7–12, 2012.

[95] A. Biere, A. Cimatti, E. Clarke, M. Fujita, and Y. Zhu, "Symbolic model checking using sat procedures instead of bdds," in *Proceedings 1999 Design Automation Conference (Cat. No. 99CH36361)*, pp. 317–320, 1999.

[96] A. C. Myers and B. Liskov, "A decentralized model for information flow control," in *Proceedings of the Sixteenth ACM Symposium on Operating Systems Principles*, SOSP '97, (New York, NY, USA), p. 129–142, Association for Computing Machinery, 1997.

[97] J. Rajendran, V. Vedula, and R. Karri, "Detecting malicious modifications of data in third-party intellectual property cores," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2015.

[98] J. Rajendran, A. M. Dhandayuthapany, V. Vedula, and R. Karri, "Formal security verification of third party intellectual property cores for information leakage," in *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*, pp. 547–552, 2016.

[99] A. Nahiyan, M. Sadi, R. Vittal, G. Contreras, D. Forte, and M. Tehranipoor, "Hardware trojan detection through information flow security verification," in *2017 IEEE International Test Conference (ITC)*, pp. 1–10, 2017.

[100] A. Waksman, M. Suozzo, and S. Sethumadhavan, "Fanci: identification of stealthy malicious logic using boolean functional analysis," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, (New York, NY, USA), p. 697–708, Association for Computing Machinery, 2013.

[101] J. Zhang, F. Yuan, and Q. Xu, "Detrust: Defeating hardware trust verification with stealthy implicitly-triggered hardware trojans," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, (New York, NY, USA), p. 153–166, Association for Computing Machinery, 2014.

[102] J. Zhang, F. Yuan, L. Wei, Y. Liu, and Q. Xu, "Veritrust: Verification for hardware trust," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 7, pp. 1148–1161, 2015.

[103] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar, "Trojan detection using ic fingerprinting," in *2007 IEEE Symposium on Security and Privacy (SP '07)*, pp. 296–310, 2007.

[104] B. Zhou, R. Adato, M. Zangeneh, T. Yang, A. Uyar, B. Goldberg, S. Unlu, and A. Joshi, "Detecting hardware trojans using backside optical imaging of embedded watermarks," in *Proceedings of the 52nd Annual Design Automation Conference*, DAC '15, (New York, NY, USA), Association for Computing Machinery, 2015.

[105] M. Banga and M. S. Hsiao, "A novel sustained vector technique for the detection of hardware trojans," in *2009 22nd International Conference on VLSI Design*, pp. 327–332, 2009.

[106] R. S. Chakraborty and S. Bhunia, "Security against hardware trojan through a novel application of design obfuscation," in *2009 IEEE/ACM International Conference on Computer-Aided Design - Digest of Technical Papers*, pp. 113–116, 2009.

[107] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware trojan attacks: Threat analysis and countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, 2014.

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

[109] J. Aarestad, D. Acharyya, R. Rad, and J. Plusquellic, "Detecting trojans through leakage current analysis using multiple supply pad I_{ddq}s," *IEEE Transactions on Information Forensics and Security*, vol. 5, no. 4, pp. 893–904, 2010.

[110] D. Forte, C. Bao, and A. Srivastava, "Temperature tracking: An innovative run-time approach for hardware trojan detection," in *2013 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 532–539, 2013.

[111] F. Stellari, P. Song, A. J. Weger, J. Culp, A. Herbert, and D. Pfeiffer, "Verification of untrusted chips using trusted layout and emission measurements," in *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 19–24, 2014.

[112] C. Sturton, M. Hicks, D. Wagner, and S. T. King, "Defeating uci: Building stealthy and malicious hardware," in *2011 IEEE Symposium on Security and Privacy*, pp. 64–77, 2011.

[113] C. Cadar, D. Dunbar, and D. Engler, "Klee: unassisted and automatic generation of high-coverage tests for complex systems programs," in *Proceedings of the 8th USENIX Conference on Operating Systems Design and Implementation*, OSDI'08, (USA), p. 209–224, USENIX Association, 2008.

[114] S. Mitra and E. McCluskey, "Which concurrent error detection scheme to choose ?," in *Proceedings International Test Conference 2000 (IEEE Cat. No.00CH37159)*, pp. 985–994, 2000.

[115] O. Keren, I. Levin, and M. Karpovsky, "Duplication based one-to-many coding for trojan hw detection," in *2010 IEEE 25th International Symposium on Defect and Fault Tolerance in VLSI Systems*, pp. 160–166, 2010.

[116] J. Rajendran, H. Zhang, O. Sinanoglu, and R. Karri, "High-level synthesis for security and trust," in *2013 IEEE 19th International On-Line Testing Symposium (IOLTS)*, pp. 232–233, 2013.

[117] D. McIntyre, F. Wolff, C. Papachristou, and S. Bhunia, "Trustworthy computing in a multi-core system using distributed scheduling," in *2010 IEEE 16th International On-Line Testing Symposium*, pp. 211–213, 2010.

[118] C. Liu, J. Rajendran, C. Yang, and R. Karri, "Shielding heterogeneous mpsocs from untrustworthy 3pips through security-driven task scheduling," in *2013 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, pp. 101–106, 2013.

[119] T. Reece, D. B. Limbrick, and W. H. Robinson, "Design comparison to identify malicious hardware in external intellectual property," in *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications*, pp. 639–646, 2011.

[120] G. Bloom, B. Narahari, and R. Simha, "Os support for detecting trojan circuit attacks," in *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, pp. 100–103, 2009.

[121] J. Dubeuf, D. Hély, and R. Karri, "Run-time detection of hardware trojans: The processor protection unit," in *2013 18th IEEE European Test Symposium (ETS)*, pp. 1–6, 2013.

[122] S. Narasimhan, W. Yueh, X. Wang, S. Mukhopadhyay, and S. Bhunia, "Improving ic security against trojan attacks through integration of security monitors," *IEEE Design & Test of Computers*, vol. 29, no. 5, pp. 37–46, 2012.

[123] Y. Jin and D. Sullivan, "Real-time trust evaluation in integrated circuits," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1–6, 2014.

[124] Y. Jin, D. Maliuk, and Y. Makris, "Post-deployment trust evaluation in wireless cryptographic ics," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 965–970, 2012.

[125] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *DAC Design Automation Conference 2012*, pp. 83–89, 2012.

[126] R. S. Chakraborty and S. Bhunia, "Harpoon: An obfuscation-based soc design methodology for hardware protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 10, pp. 1493–1502, 2009.

[127] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," in *2008 Design, Automation and Test in Europe*, pp. 1069–1074, 2008.

[128] A. Baumgarten, A. Tyagi, and J. Zambreno, "Preventing ic piracy using reconfigurable logic barriers," *IEEE Design & Test of Computers*, vol. 27, no. 1, pp. 66–75, 2010.

[129] J. B. Wendt and M. Potkonjak, "Hardware obfuscation using puf-based logic," in *Proceedings of the 2014 IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '14, p. 270–277, IEEE Press, 2014.

[130] K. Xiao and M. Tehranipoor, "Bisa: Built-in self-authentication for preventing hardware trojan insertion," in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 45–50, 2013.

[131] K. Xiao, D. Forte, and M. Tehranipoor, "A novel built-in self-authentication technique to prevent inserting hardware trojans," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 33, no. 12, pp. 1778–1791, 2014.

[132] P.-S. Ba, S. Dupuis, M. Palanichamy, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, "Hardware trust through layout filling: A hardware trojan prevention technique," in *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 254–259, 2016.

[133] J. Rajendran, M. Sam, O. Sinanoglu, and R. Karri, "Security analysis of integrated circuit camouflaging," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, (New York, NY, USA), p. 709–720, Association for Computing Machinery, 2013.

[134] R. P. Cocchi, J. P. Baukus, L. W. Chow, and B. J. Wang, "Circuit camouflage integration for hardware ip protection," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–5, 2014.

[135] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 517–532, 2020.

[136] R. P. Cocchi, J. P. Baukus, L. W. Chow, and B. J. Wang, "Circuit camouflage integration for hardware IP protection," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–5, 2014.

[137] M. Li, K. Shamsi, T. Meade, Z. Zhao, B. Yu, Y. Jin, and D. Z. Pan, "Provably secure camouflaging strategy for IC protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 8, pp. 1399–1412, 2019.

[138] Y. Bi, P.-E. Gaillardon, X. S. Hu, M. Niemier, J.-S. Yuan, and Y. Jin, "Leveraging emerging technology for hardware security - case study on silicon nanowire fets and graphene symfets," in *2014 IEEE 23rd Asian Test Symposium*, pp. 342–347, 2014.

[139] Cadence Design Systems, Inc., "Jasper spv app," last accessed on May. 20, 2024. Available at: `https://www.cadence.com/en_US/home/tools/system-design-and-verification/formal-and-static-verification/jasper-gold-verification-platform/security-path-verification-app.html`.

[140] "Opentitan documentation index," last accessed on May. 24, 2024. Available at: `https://opentitan.org/book/doc/introduction.html`.

[141] T. F. Wu, K. Ganesan, Y. A. Hu, H.-S. P. Wong, S. Wong, and S. Mitra, "Tpad: Hardware trojan prevention and detection for trusted integrated circuits," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 35, no. 4, pp. 521–534, 2016.

[142] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmarks development," in *IEEE 31st Intl. Conf. on Computer Design (ICCD)*, pp. 471–474, 2013.

[143] F. Corno, M. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE Des. Test Cmput.*, vol. 17, no. 3, pp. 44–53, 2000.

[144] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, "Benchmarking of Hardware Trojans and Maliciously Affected Circuits," *J. of Hardw. and Sys. Security*, vol. 1, no. 1, pp. 85–102, 2017.

[145] W. Cullyer, "Implementing high integrity systems: the VIPER microprocessor," *IEEE Aerospace and Electronic Systems Magazine*, vol. 4, no. 6, pp. 5–13, 1989.

[146] M. Boulé and Z. Zilic, "Automata-based assertion-checker synthesis of PSL properties," *ACM TDAES*, vol. 13, no. 1, pp. 1–21, 2008.

[147] A. Hepp, T. Perez, S. Pagliarini, and G. Sigl, "A pragmatic methodology for blind hardware trojan insertion in finalized layouts," ICCAD '22, 2022.

[148] Y. Jin and Y. Makris, "Hardware trojan detection using path delay fingerprint," in *2008 IEEE International Workshop on Hardware-Oriented Security and Trust*, pp. 51–57, 2008.

[149] I. Exurville, L. Zussa, J.-B. Rigaud, and B. Robisson, "Resilient hardware trojans detection based on path delay measurements," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 151–156, 2015.

[150] X.-T. Ngo, I. Exurville, S. Bhasin, J.-L. Danger, S. Guilley, Z. Najm, J.-B. Rigaud, and B. Robisson, "Hardware trojan detection by delay and electromagnetic measurements," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 782–787, 2015.

[151] S. Dupuis, P.-S. Ba, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, "New testing procedure for finding insertion sites of stealthy hardware trojans," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 776–781, 2015.

[152] N. Lesperance, S. Kulkarni, and K.-T. Cheng, "Hardware trojan detection using exhaustive testing of k-bit subspaces," in *The 20th Asia and South Pacific Design Automation Conference*, pp. 755–760, 2015.

[153] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

[154] W. Hu, C.-H. Chang, A. Sengupta, S. Bhunia, R. Kastner, and H. Li, "An overview of hardware security and trust: Threats, countermeasures, and design tools," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 6, pp. 1010–1038, 2021.

[155] H. Wang, Q. Shi, A. Nahiyan, D. Forte, and M. M. Tehranipoor, "A physical design flow against front-side probing attacks by internal shielding," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 2152–2165, 2020.

[156] T. Sugawara, N. Homma, T. Aoki, and A. Satoh, "Asic performance comparison for the iso standard block ciphers," in *JWIS*, 2007.

[157] O. Girard et al., "openmsp430 at opencores.org," 2021. Available at: `https://opencores.org/projects/openmsp430`.

[158] M. Hicks et al., "Mit-ll common evaluation platform (cep) at github.com," 2021. Available at: `https://github.com/mit-ll/CEP`.

[159] SILVACO, "Nangate freepdk45 open cell library." Available at: `http://www.nangate.com/?page_id=2325`.

[160] "Salsy repository." Available at: `https://github.com/Centre-for-Hardware-Security/SALSy`.

[161] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "A2: Analog malicious hardware," in *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 18–37, 2016.

[162] T. Trippel, K. G. Shin, K. B. Bush, and M. Hicks, "T-ter: Defeating a2 trojans with targeted tamper-evident routing," in *Proceedings of the 2023 ACM Asia Conference on Computer and Communications Security*, ASIA CCS '23, (New York, NY, USA), p. 746–759, Association for Computing Machinery, 2023.

Acknowledgements

In expressing my deepest gratitude, I first pay tribute to the Iranian heroes and martyrs who have demonstrated unyielding resistance in the face of adversity, enduring immense hardships and making selfless sacrifices, even giving up their precious lives. Their unwavering courage and dedication stand as a testament to the fact that true success often emerges from great struggles, and that perseverance against all odds can pave the way for a more prosperous and harmonious world. It is because of their sacrifices that I have been able to pursue my studies in peace, and their enduring legacy of resilience continues to inspire and motivate me daily, reminding me that even in the darkest times, hope and determination can lead to triumph.

I extend my heartfelt appreciation to my advisor, Prof. Dr. Samuel Pagliarini, whose guidance, support, and encouragement have been invaluable throughout my academic journey. Your wisdom, insight, and unwavering belief in my potential have significantly contributed to the successful completion of this thesis.

My boundless gratitude goes to my beloved wife. Your unwavering support, understanding, and sacrifices have been the cornerstone of my success. Your love, patience, and encouragement have sustained me through the most challenging times and have made this achievement possible. I am deeply grateful for your constant presence and dedication.

I also extend my heartfelt thanks to my parents and my sister. Your continuous support and belief in me have been a constant source of strength and motivation.

I wish to acknowledge my cherished friends on the 5th floor of the ICT building: Zain, Levent, Malik, Muayad, Mojtaba, Mahdi, Reza, Sharjeel, and others. Your camaraderie, advice, and the many unforgettable moments we shared have made this journey not only fruitful but also immensely enjoyable. Your friendship and support have been invaluable, and I am grateful to have such remarkable friends and colleagues.

I would also like to express my gratitude to my co-advisor, Dr. Tara Ghasempouri, for her contribution to this work.

Moreover, I am deeply grateful to the welcoming community of Tallinn, who has embraced me for nearly four years. Your warmth, kindness, and support have made my stay enriching, memorable, and filled with opportunities for growth, providing me with a nurturing environment to pursue my studies.

I also acknowledge the generous financial support provided by the European Commission through the European Social Fund in the context of the project “ICT programme,” by the Estonian Research Council grant “MOBERC35,” and by HARNO (Grant No. 11.4-1/23/1). Your support has been instrumental in enabling my research and studies, and I am thankful for your investment in my academic pursuits.

Lastly, I would like to express my appreciation to my colleagues and the esteemed faculty members of the Computer Systems Department at TalTech for their assistance, collaboration, and contributions throughout my research. Your support, camaraderie, and shared expertise have made this journey a rewarding and enriching experience that I will always cherish.

Thank you all.

Abstract
On the Use of Defensive Schemes for Hardware Security

The digitalization era has profoundly transformed daily life, driven in part by the crucial role of Integrated Circuits (ICs) in modern electronics. These microelectronic components, central to devices ranging from smartphones to advanced computing systems, have been pivotal in technological advancements. However, the globalization of IC fabrication introduces significant security risks such as Hardware Trojans (HTs), intellectual property theft, counterfeiting, and reverse engineering.

This thesis addresses these security challenges by developing innovative methodologies for designing ICs with inherent security features. The first contribution is a novel approach that reuses verification assertions for HT detection. By leveraging the capabilities of the Cadence JasperGold Security Path Verification tool, assertions are transformed into online monitors, supported by a security metric and an assertion selection process. Experimental results, applied to diverse IPs within the OpenTitan System-on-Chip, validate the adaptability and scalability of this approach.

The second contribution enhances IC security during the back-end design phase by embedding online monitors directly into the layout, considering front-end inserted assertions. This dual-phase strategy, applied during physical synthesis, provides a comprehensive and flexible security solution that can be customized based on specific requirements.

The thesis further introduces Security-Aware Layout Synthesis (SALSy), a methodology that integrates security considerations into standard layout synthesis processes. SALSy adapts placement, routing, and clock tree synthesis algorithms to be security-aware, thereby protecting ICs against HT insertion, fault injection, and probing attacks. Validated through silicon-based implementations, SALSy demonstrates effectiveness with minimal impact on power consumption.

Overall, this thesis makes significant contributions to IC security by proposing methods for HT detection, improving security in the IC design process, and developing security-aware layout synthesis techniques. These solutions have been rigorously tested and proven to mitigate various security threats, providing a robust framework for future IC designs. By embedding security at every stage of the design process, this work lays the foundation for creating more secure and resilient technological advancements in an increasingly globalized supply chain.

Kokkuvõte
Kaitseskeemid riistvara turvalisuse tagamiseks

Digitaalajastu on sügavalt muutnud igapäevaelu, seda osaliselt tänu integreeritud ahelate olulisele rollile kaasaegses elektroonikas. Need mikroelektroonilised komponendid, mis on olulised nii nutitelefonide kui ka arenenud arvutisüsteemide jaoks, on olnud tehnoloogiliste edusammude juures võtmerollis. Siiski toob IC-de globaalne tootmine kaasa olulisi turvariske, nagu riistvaralised troojalased, intellektuaalomandi vargus, võltsimine ja pöördprojekteerimine.

Käesolev doktoritöö tegeleb nende turvariskidega, arendades välja uuenduslikke metoodikaid IC-de projekteerimiseks sisemiste turvafunktsioonidega. Esimene panus on uudne lähenemine, mis kasutab riistvaralise troojalase avastamiseks verifitseerimisväiteid uuesti. Kasutades Cadence JasperGold Security Path Verification tööriista võimalusi, muudetakse väited võrgupõhisteks monitorideks, toetudes turvameetrile ja väidete valiku protsessile. Eksperimentaalsed tulemused, mis kohaldati erinevatele intellektuaalomanditele OpenTitan System-on-Chip raames, kinnitavad selle lähenemisviisi kohandatavust ja mastaapsust.

Teine panus tugevdab IC-de turvalisust tagumise disainifaasi raames, sisestades otse paigutusse võrgupõhised monitorid, arvestades eesmise faasi sisestatud väiteid. See kahefaasiline strateegia, mida rakendatakse füüsilise sünteesi käigus, pakub terviklikku ja paindlikku turvalahendust, mida saab kohandada vastavalt spetsiifilistele nõudmistele.

Töö tutvustab lisaks turvateadlikku paigutuse sünteesi (SALSy), metodoloogiat, mis integreerib turvaküsimused tavapärastesse paigutuse sünteesi protsessidesse. SALSy kohandab paigutuse, marsruutimise ja kellapuu sünteesi algoritme, et muuta need turvateadlikuks, kaitstes IC-sid riistvaralise troojalase sisestamise, rikke sissepritse ja sondimise rünnakute eest. Ränipõhiste rakenduste kaudu valideeritud SALSy demonstreerib efektiivsust minimaalse mõjuga energiatarbimisele.

Kokkuvõttes teeb käesolev doktoritöö märkimisväärseid panuseid IC-de turvalisusse, pakkudes meetodeid riistvaralise troojalase avastamiseks, parandades turvalisust IC-de projekteerimise protsessis ja arendades turvateadlikke paigutuse sünteesi tehnikaid. Need lahendused on põhjalikult testitud ja tõestanud oma efektiivsust erinevate turvariskide vähendamisel, pakkudes tugevat raamistikku tulevastele IC-de projekteerimistele. Turvalisuse integreerimine igasse projekteerimise etappi loob aluse turvalisemate ja vastupidavamate tehnoloogiliste edusammude saavutamiseks üha globaliseeruvas tarneahelas.

Appendix 1

I

M. Eslami, T. Ghasempouri and S. Pagliarini, “Reusing Verification Assertions as Security Checkers for Hardware Trojan Detection,” In Proceedings of the 2022 23rd International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, 2022, pp. 1-6.
DOI: https://doi.org/10.1109/ISQED54688.2022.9806292

2022 23rd International Symposium on Quality Electronic Design (ISQED) | 978-1-6654-9466-3/22/$31.00 ©2022 IEEE | DOI: 10.1109/ISQED54688.2022.9806292

Reusing Verification Assertions as Security Checkers for Hardware Trojan Detection

Mohammad Eslami, Tara Ghasempouri and Samuel Pagliarini
Department of Computer Systems
Tallinn University of Technology (TalTech)
Tallinn, Estonia
Email: {FirstName.LastName@taltech.ee}

***Abstract*—Globalization in the semiconductor industry enables fabless design houses to reduce their costs, save time, and make use of newer technologies. However, the offshoring of Integrated Circuit (IC) fabrication has negative sides, including threats such as Hardware Trojans (HTs) – a type of malicious logic that is not trivial to detect. One aspect of IC design that is not affected by globalization is the need for thorough verification. Verification engineers devise complex assets to make sure designs are bug-free, including assertions. This knowledge is typically not reused once verification is over. The premise of this paper is that verification assets that already exist can be turned into effective security checkers for HT detection. For this purpose, we show how assertions can be used as online monitors. To this end, we propose a security metric and an assertion selection flow that leverages Cadence JasperGold Security Path Verification (SPV). The experimental results show that our approach scales for industry-size circuits by analyzing more than 100 assertions for different Intellectual Properties (IPs) of the OpenTitan System-on-Chip (SoC). Moreover, our detection solution is pragmatic since it does not rely on the HT activation mechanism.**

***Index Terms*—Hardware Trojans, Hardware Security, Security Coverage, Verification**

I. Introduction and Background

Over the last decades, the Integrated Circuit (IC) industry has experienced significant changes in the fabrication process due to globalization. This globalization has made the companies define new strategies to reduce their costs in the IC supply chain. Hence, today, it is very unlikely for a design house to fabricate a circuit. Instead, the fabrication process is outsourced to third-party vendors. This opens an opportunity for an attacker to replace some parts of the original circuit with altered ones, or even to remove some logic in the design process [1]. This threat is generally referred to as Hardware Trojans (HTs). Hardware Trojans are known as one of the greatest threats against the trustworthiness of ICs, and they have raised serious concerns about the reliability and security of digital designs [2]. If an IC is utilized in a product/system while Trojan(s) remain there, it may lead to functionality change, reliability degradation/denial of service, and information or data leakage [3].

Unfortunately, typical test and verification tasks are not sufficient to detect HTs that are inserted during the fabrication stage [1] (i.e., fabrication-time attacks). Many accepted approaches exist to enable HT detection [2], [4]–[8]. In these approaches, it is tried to detect the HTs either by disruptive methods or non-disruptive ones. In disruptive methods, the IC is being de-masked and if necessary, de-layered. Then it goes through the investigation using electron microscopes and special measurement equipment to check if it is the same as designed [4]. Non-disruptive methods focus on catching the unwanted behavior of the circuit using analytical or formal methods. Mostly used techniques in non-disruptive methods include side-channel analysis and online monitoring. Side-channel analysis techniques are based on the concept of checking the different parametric characteristics of the circuit such as power and timing and looking for a deviation from the expected signatures to detect Trojans [5], [6]. The major drawback of the side-channel analysis technique is that the obtained results from the analysis can be confused with the process variation. Some approaches try to enhance the detection probability of this technique by applying Automatic Test Pattern Generation (ATPG) or dummy flip-flop insertion [2], [7].

On the other hand, online monitoring techniques rely on embedding checker circuits in different locations of the design to catch the unwanted behavior of the system [8]–[10]. One of the popular approaches for building these checker circuits is to use assertions [11]. Assertions precisely describe the expected behavior of the circuit and help to check if there is a deviation between the intent and the actual behavior of the design using simulation or formal methods [11]. Assertion-Based Verification (ABV) techniques are mostly implemented by writing assertion checkers in Property Specification Language (PSL) [12] or SystemVerilog [13].

Although online monitoring techniques offer a high detection coverage, they impose significant overheads on the circuit. In recent years, it is tried to decrease the overheads while keeping the detection coverage at the maximum level, but still, the trade-off between the detection coverage and the imposed overheads is unfavorable [10].

Since most of the time and efforts spent on the design and verification processes do not contribute to achieving HT detection, a considerable amount of design knowledge (i.e. test benches, coverage, metrics, assertions) is generated and then not re-utilized. In this paper, we propose a methodology for selecting the assertions that have already been written by verification engineers (for functional verification) and explain how to reuse them for security purposes (i.e., HT detection).

Reusing the available data seems to be a wiser choice rather than spending similar (or more) time and resources on complex detection schemes. For this purpose, we present a new metric called *security coverage* to evaluate the efficiency of online checkers in detecting Trojans considering the amount of overhead imposing on the circuit. This metric helps to remove assertions that are not helpful for HT detection from the circuit in an automated fashion while there is no need for detailed knowledge of the circuit.

The remainder of the paper is organized as follows: In Section II, we explain the prerequisites for converting assertions to security checkers and introduce a new security coverage metric. The effectiveness of the selected checkers is studied in Section III. Section IV explains a methodology to optimize the security checker list, and experimental results are presented in Section V. Finally, Section VI concludes the paper.

II. Assertions as Trojan Detectors

In this section, we answer the question *can assertions be devised for detecting Trojans?* For this purpose, we study the B19-T500 benchmark from Trust-Hub [14] which is a Trojan-inserted version of the B19 circuit from the ITC'99 benchmark suite [15]. Trust-Hub benchmarks provide an opportunity for developers to verify the effectiveness of their HT detection schemes since the Trojans can be considered as a representative for real ones due to their small sizes and rare triggering conditions [16]. This implies the HTs remain hidden during standard verification checks [17].

More precisely, the B19 benchmark consists of four copies of the Viper processor, and the Trojan circuit is located inside each Viper processor. The Trojan is triggered by a counter which counts a specific vector and resets with another specific vector. If the counter gets a value between 3'b100 and 3'b110, the Trojan is triggered. The Trojan payload modifies the bits of the Instruction Register (IR) of the embedded Viper processor and changes the functionality of the circuit [14].

A. Prerequisite of good Security Checker

The ideal conditions for assertions to be considered as good security checkers are: 1) they should not impose significant overheads on the circuit once they are synthesized (many of them may be needed in complex designs to reach the high detection coverage), and 2) they should not be limited in scope. Assertions that only check some local signals (i.e., checking if one of the specific bits of a register is 0 or 1) are rarely interesting. Instead, assertions that capture a high-level behavior are preferred. From now on, we call these assertions "top-level assertions".

To clarify this concept, we have defined a set of manually written assertions that satisfy the aforementioned conditions. While the easiest way for detecting the B19-T500 Trojan is to write some assertions for checking the IR bits directly, this style is not practical for two reasons: First, in a realistic scenario, we do not have any information about the locations of HTs. Even taking this fact into account, thousands (or more) of this type of assertion would be needed for covering all the necessary checks of individual signals, even for small circuits. Furthermore, this style of assertion writing violates the second condition of being a good security checker: it does not describe any notion of a system-level behavior.

TABLE I
Considered assertions for detecting Trojans on B19-T500 benchmark

Name	Assertion definition
ASR_1	`assert always {(!(IR == OP_STORE)) -> (!wr)};`
ASR_2	`assert always {(IR == OP_STORE) -> (wr)};`
ASR_3	`assert always {(!(IR == OP_READ)) -> (!rd)};`
ASR_4	`assert always {(IR == OP_READ) -> (rd)};`

Table I contains the top-level assertions considered for detecting Trojans in the B19-T500 circuit. They have been written in PSL. As shown in the table, the assertions check the correctness of transactions between the memory and processor. The first two assertions check the violation (invalid access) of the write operation in the memory while the next ones do the same check for the read operation. For more details about the mechanism of the memory access in the Viper processor, the reader is referred to its documentation [18]. Our simulation results show that our assertions can effectively detect the Trojan inserted in the B19-T500 benchmark. The effectiveness of these assertions is discussed later in Section V.

B. Binding the assertions to the main design

While simulation provides useful information about the incorrect behavior of the circuit and different internal values, it is not sufficient for determining the performance characteristics of the design such as power, timing, and area. Hence, it is impossible to qualify the assertions via simulations only. Instead, the design should be synthesized, and exact performance reports being taken into consideration. As mentioned earlier, PSL and SystemVerilog are the most commonly used languages for describing assertions, but these assertions are not directly synthesizable. Therefore, we have to transform these assertions to a synthesizable format such that the performance results can be obtained after synthesis. For this purpose, we use the MBAC tool [19] to convert PSL and SystemVerilog assertions to a synthesizable Verilog format.

After generating a synthesizable code from the assertions, they can be bound to the main circuit so that we can evaluate the effectiveness of the assertions based on the overheads imposed on the circuit. For this purpose, we first synthesize the main circuit without the assertions and obtain the maximum clock frequency, power, and area reports. Later, the original circuit and the bound assertions is once again synthesized. At this point, the assertion goes from a verification asset to an embedded online checker. Finally, these results of the two syntheses are compared to evaluate the performance of the assertions.

C. Security Coverage

Despite having information about the performance results for each assertion, we cannot still reach a complete decision

regarding the effectiveness of a given assertion. In other words, although we know exactly the cost of these assertions, we are not aware of what we achieve. So, a new evaluation scheme is needed to build a trade-off between the costs and the achievements. In this sense, we propose a new metric for the assessment of the assertions considering the security properties. The general idea is to synthesize the original design along with the assertion circuit and check if there are any **functional paths**[1] from the individual nodes in the main design to the output of the assertion circuit. Cadence JasperGold Security Path Verification (SPV), our tool of choice, performs proofs to find the existence of functional paths between the design nodes and the assertion outputs, and existence of such paths means that the origin nodes are covered by the assertion. Therefore, if the origin node in the design is the location of the payload of a Trojan, an assertion that can be reached by that node can detect the malicious logic. A higher number of nodes reachable from the original circuit to the assertion output represents better coverage in Trojan detection for that assertion. Based on these explanations, we define our security coverage metric as follows:

$$Security\ Coverage = \frac{Number\ of\ reachable\ nodes}{Number\ of\ total\ nodes} \quad (1)$$

To obtain the security coverage for each assertion, first, the circuit is synthesized while the assertion is bound to it. Then, all the nodes in the synthesized netlist have to be extracted for further analysis. For this purpose, a tool is developed which receives a netlist as an input and generates a list containing all nodes inside it. This list is then submitted to SPV. This tool is mainly used to check if a part of a design is securely isolated from the other parts, usually referred to as taint analysis (e.g., test if features of a processor should not be accessible except in the test mode), but with some changes in its initial configuration, it can be utilized to calculate the security coverage we need. For this aim, we create a list of pairs of nodes in the format *origin*, *destination*, where all nodes in the circuit are possible origin nodes and a destination node is the output of an assertion. Then, the list of pairs of nodes is fed into the SPV tool to check if there is a functional path between them. In this stage, the inner engines of the SPV tool create properties for each pair and try to prove that there is no functional path for the property or to provide some counterexamples for the opposite condition. After finishing the analysis, the security coverage can be calculated using equation 1. A Tool Command Language (TCL) script is used to automate the process.

Now, the needed information for evaluating the effectiveness of the assertions can be taken into account for qualifying them regarding the security aspects. The security coverage can be utilized to drive a trade-off analysis and help the user to decide which assertions are suitable for his/her purposes. It should be noted that not all the circuits need 100% of security coverage; for example, if some sensitive parts of the circuit are already identified and are the only parts needed to be secured, covering those parts is sufficient to satisfy the user demands.

[1]A functional path is one that can be exercised with a combination of valid inputs. This is in contrast with structural paths or timed paths (STA) that might not be reachable.

III. OpenTitan - a case study

In the previous section, we studied our own defined assertions to prove that they can detect Trojans and be seen as security checkers. But, this practice is hard to generalize: writing such top-level assertions is significantly time-consuming and hard to achieve. Moreover, the main contribution of this work is precisely to reuse the assertions that already exist for verification purposes, instead of generating new ones. Hence, in this section we study the evaluation of different assertions written for verifying the OpenTitan System-on-Chip (SoC) [20], to check if they can be used as security checkers.

OpenTitan is an open-source project consisting of a RISC-V-based processor and IPs from different vendors [20]. It also includes functional assertions for different Intellectual Properties (IPs) which makes it a remarkable candidate for our study[2].

For this purpose, the Register Top modules of each IP are chosen. This module controls the transactions between the IP and the bus, and is responsible for granting access to read/write requests for IP registers. Moreover, it has a unique error generation mechanism for writes and reads that target addresses that are not represented within the register list [20]. Since the Register Top modules of different IPs include the same assertions, it provides a good comparison among the experiments. A set of selected assertions is shown in Table II. In total, 108 different assertions are studied on 35 individual IPs in OpenTitan SoC to demonstrate that the obtained results are comparable to a realistic example and our approach is scalable to industry-size circuits.

To obtain the security coverage of each assertion, the same flow as explained in the previous section is used: MBAC translation, assertion binding, and then synthesis. Finally, the nodes from the synthesized netlist are fed into the SPV tool to calculate the security coverage.

IV. Optimizing the assertion list

Manually checking the assertions to see if they are top-level or not is a very time-consuming process and it questions the efficiency of the proposed approach. Hence, a decision flow is needed to wisely choose the assertions suitable for the security aims. For this purpose, we present a methodology to help the user only pick efficient security checkers from the available assertions based on his/her needs. This is a necessary step since all the functional assertions are not suitable for security purposes. The overall flow of this methodology is shown in Fig. 1. The first step is to create a list of candidates containing the assertions which can be recognized to be synthesized along with the original circuit, and pick one. Then it has to be converted to a synthesizable format (step 2) to be bound to the main design (step 3). After the synthesis process, different

[2]Some assertions have simulation-based nature and cannot be synthesized (i.e., assertions checking whether a signal is *X* or not).

TABLE II
CONSIDERED ASSERTIONS FOR REGISTER TOP MODULES OF DIFFERENT IPS IN OPENTITAN SOC

Assertion name	Assertion definition
wePulse	`assert property (@(posedge clk_i) disable iff ((!rst_ni) !== 1'b0) $rose(reg_we) |=> !(reg_we));`
rePulse	`assert property (@(posedge clk_i) disable iff ((!rst_ni) !== 1'b0) $rose(reg_re) |=> !(reg_re));`
reAfterRv	`assert property (@(posedge clk_i) disable iff ((!rst_ni) !== 1'b0) $rose(reg_re || reg_we) |=> tl_o);`
en2addrHit	`assert property (@(posedge clk_i) disable iff ((!rst_ni) !== 1'b0) ((reg_we || reg_re) |-> $onehot0(addr_hit)));`

performance reports are generated to help the user decide if the overheads are acceptable or not (step 4). The margin threshold for the overheads can be defined by the user based on his/her needs, and if the overheads for the selected assertion go beyond the defined boundaries, that assertion is put away and another one is picked from the candidate list. Otherwise, the circuit nodes are extracted and fed into the SPV tool to obtain security coverage (step 5). Finally, the user can decide to add this assertion to the security checker list based on the trade-off between the overheads and the security coverage or to select another one from the candidate list.

A. Optimization flow for selecting the assertions

In this part, we choose Alert Handler IP from the OpenTitan SoC [20], which contains several assertions and we show how to form a list of security checkers among these assertions. At the first step, a candidate list including 13 different assertions is created. These assertions are predefined by the OpenTitan developers and their main objective is to make sure that the functionality of the circuit will remain the same as its intent. Since the final security checker list is defined based on the user needs, we defined two different strategies for selecting the appropriate candidates: *fixed-threshold* and *dynamic-threshold* strategies. It should be noted that defining the strategies is completely flexible and depends on the amount of security needed in the cost of performance reduction. In the following, we explain the proposed optimization flow for our defined strategies.

Fixed-threshold strategy: *If the maximum performance overhead for each assertion is X (percent), the security coverage should be at least 10X (percent).*

This strategy defines fixed thresholds for the overheads and/or security coverage of each assertion, and removes the assertions that violate these thresholds from the candidate list. To follow the strategy rules, different overheads of each assertion should be obtained first, and after calculating the security coverage, the maximum overhead (area, power, or timing) goes under comparison. Based on this strategy, only 2 out of 13 assertions of the Alert Handler IP are removed from the candidate list.

Dynamic-threshold strategy: *The maximum performance overhead for each assertion should not exceed twice the average performance overheads. For the security coverage, we only pick those assertions that have a positive impact on the overall coverage of the circuit compared to other assertions.*

In contrast to the previous strategy, where assertions are assessed individually, the dynamic strategy performs comparisons between competing assertions. We follow the same procedure to obtain the performance overheads for the first condition of this strategy, but by looking at the security coverage results for individual assertions, no information can be obtained regarding the positive impact of the assertions. Instead, this strategy selects only assertions that fare better than average.

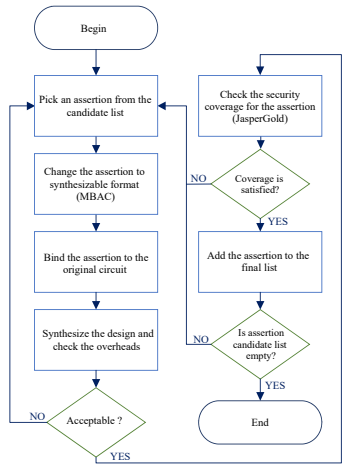


Fig. 1. Proposed flow for optimizing the assertion list

The first strategy, while simple and easy to implement, requires the user to define a constant (10) for the threshold. The second strategy requires no such constant, but a sufficient number of assertions is needed for defining what average overhead and coverage look like. More details are provided in the next section, where we show how the dynamic strategy can be more effective than its fixed threshold counterpart. Nevertheless, more convoluted strategies can be defined and this remains as future work.

V. EXPERIMENTAL RESULTS

In this section, we present experimental results including performance overheads and security coverage obtained for different circuits as explained in previous sections. For all experiments reported here, we have used Cadence Genus and our target cell library is a commercial 65nm library.

A. B19-T500 benchmark from Trust-Hub

Fig. 2 shows the normalized numbers of timing, power, and area overheads for the assertions considered for the B19-T500 benchmark. As shown in this figure, while the area overhead

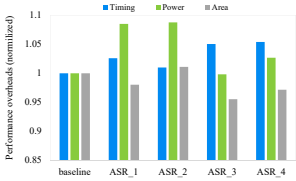


Fig. 2. Performance overheads imposed by different assertions of B19-T500 benchmark from Trust-Hub

for three of the assertions (*ASR_1, ASR_2,* and *ASR_4*) is zero, the maximum overheads belong to *ASR_2* and *ASR_1* respectively, which consume approximately 9% more power than the original circuit. Also, the timing overhead is less than 6% for all of the assertions. It should be noted that the normalized numbers lower than 1 are within the noise margin of the circuit and do not have any effect on the performance.

Table III shows the security coverage calculated for the same assertions in the B19-T500 benchmark. As shown in this table, our assertions cover on average 6.8% of the total nodes in the circuit, which means that they can catch the Trojans in their covered areas, regardless of how rare the Trojans are triggered and what impacts they would cause to the circuit. This is one of the main advantages of our method such that the user has no more to be concerned about activating the rare Trojans.

B. OpenTitan SoC

Fig. 3 depicts the security coverage obtained for different IP Register Top modules of OpenTitan SoC. As shown in this figure, the highest number for security coverage is 4.77% for the *nmi_gen_reg_top* module, and the security coverage percentage for the majority of IPs is less than 1, which does not represent a good candidate for being a security checker. This is mainly because these types of assertions are only performing small interface checks, and do not satisfy the condition of describing the top-level behavior of the circuit. Instead, they only cover some local nodes which leads to a low security coverage for the whole circuit nodes. This justifies the need for the optimization step in our proposed methodology to avoid selecting unnecessary assertions that do not have a considerable impact on Trojan detection.

C. Selecting the Assertions

In this part, we present a practical experiment using the optimization flow as shown in Fig. 1. We defined two strategies for selecting the proper assertions in Section IV, and in the following, we provide more details about the procedure of assertion selection.

TABLE III
CONVERTED ASSERTIONS TO SYNTHESIZABLE FORMAT USING MBAC TOOL

Assertion name	**Total nodes**	**Covered nodes**	**Security Coverage** (%)
ASR_1	5014	315	6.28 (%)
ASR_2	5062	304	6.01 (%)
ASR_3	4916	367	7.47 (%)
ASR_4	4944	369	7.46 (%)
Average	**4984**	**339**	**6.80** (%)

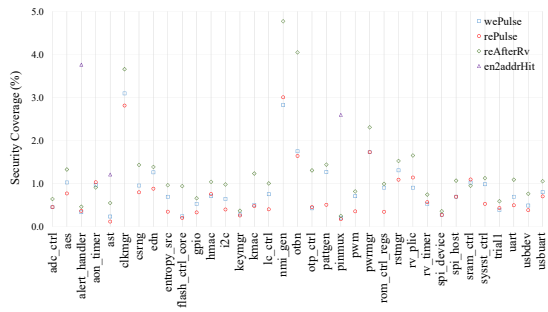


Fig. 3. Security coverage percentage for the Register Top modules of OpenTitan SoC IPs

Fixed-threshold strategy: The performance results for the assertion candidate list are shown in Fig. 4. As shown in this figure, the maximum number for the overheads belongs to timing degradation of *ah_asr_8* assertion (2.99%), while the minimum overhead belongs to *ah_asr_3* and *ah_asr_4* assertions with the number of 0.75%. At the next step, they should be checked for the security coverage which makes it to be at least 10 times higher than the maximum overhead for each assertion. Fig. 5 shows the security coverage results obtained from each assertion using the SPV tool. For better understanding, we simply associated numbers with the assertion names. Based on these results, we can ignore the *ah_asr_12,* and *ah_asr_13* since they do not satisfy the required security coverage condition and consider the other candidates as the final security checkers. Although 15% of the assertions were removed based on this strategy, defining smarter strategies can enhance the effectiveness of the final list. Hence, second strategy is defined on the same candidate list for achieving more efficiency.

Dynamic-threshold strategy: For the first condition of this strategy, we should calculate the average overhead for all the assertions which is 1.79%. Hence, all the candidates are passed since they have less than twice the average overhead in all the cases (Fig. 4). But for the second condition, it is not sufficient to refer to the security coverage results since they do not have any notion of comparison to each other. Hence, an extra step is required to select the security checkers. For this purpose, we organize the assertions from the highest security coverage (*ah_asr_11*) to the lowest one (*ah_asr_12*). Starting from the *ah_asr_11*, we add the assertion with the next highest number (*ah_asr_10* in the first round) and calculate the security coverage for the new set of assertions we made. Then, the next highest number is added to the existing set, and this process is repeated until the lowest number is added to the list.

Fig. 6 represents the security coverage numbers for each set of assertions. We include the assertion numbers in naming the set of assertions to identify the effect of added assertion in each step. For example, *ah_asr_11_10* represents a set of assertions that starts from *ah_asr_11* (the highest coverage) and ends with *ah_asr_10* (the last assertion added to the list), and *ah_asr_11_9* includes the assertions *ah_asr_11*, *ah_asr_10*, and *ah_asr_9*. Furthermore, a moving average trend-line is

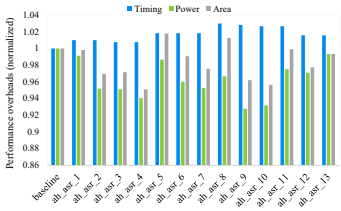


Fig. 4. Performance overheads imposed by different assertions of Alert Handler IP

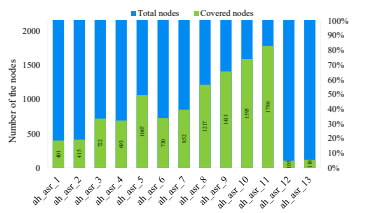


Fig. 5. Number of covered nodes for the individual assertions of Alert Handler IP

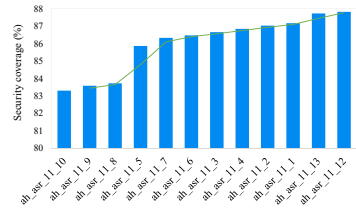


Fig. 6. Security coverage percentage for different assertion sets of Alert Handler IP

added to this figure to help for choosing the best assertions. Since the period of the moving average trend-line is 2, it can make a good comparison between the security coverage of the newly added assertion in each stage, and the number for the two previous assertions. Therefore, if the security coverage obtained after adding an assertion crosses the moving average trend, it can be realized that a noticeable difference has happened. Returning to the second condition of Strategy 2 and from Fig. 6, we can see that security coverage numbers of only three assertions have crossed the moving average trend-line (*ah_asr_5, ah_asr_7,* and *ah_asr_13*). Hence, they can be added to the final list. Moreover, the *ah_asr_11* assertion is added to the final list since it has the highest security coverage.

In contrast with the results of the Register Top modules of different IPs (Fig. 3), the security coverage numbers of different assertions in Alert Handler IP are relatively higher (Fig. 5). This is mainly because the assertions written for this specific IP are describing a top-level behavior of the design, rather than checking only local signals and interfaces.

As shown in these two examples, different strategies can be defined based on user needs which makes the presented approach flexible. Moreover, one of the advantages of our work comparing with the current approaches is the simplicity of using it without complex procedures. For example, the presented work in [10] supports Trojan detection with flexible overheads, but it requires a lot of effort and complicated steps. In contrast, we use commercial tools that are available to the community, thus increasing the portability and scalability of the presented work.

VI. Conclusion

In this paper, we presented a new methodology for using verification assertions as security checkers. The security coverage, our proposed metric for assessing the effectiveness of assertions in Trojan detection, abstracts the notion of a Trojan trigger and focuses on the effect of the payload.

We examined our methodology on case studies from the Trust-Hub benchmarks and the OpenTitan SoC with more than 100 assertions to show the scalability of our work for the industry-size circuits. Moreover, we showed how defining a smart strategy can enhance the assertion selection process. In the future, we will focus on automating these strategies to enhance the current methodology.

Acknowledgment

This work was partially supported by the EU through the European Social Fund in the context of the project "ICT programme". It was also partially supported by the Estonian Research Council grant "MOBERC35".

References

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

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

[3] T. Perez, M. Imran, P. Vaz, and S. Pagliarini, "Side-Channel Trojan Insertion - a Practical Foundry-Side Attack via ECO," in *IEEE ISCAS*, 2021, pp. 1–5.

[4] S. Jha and S. K. Jha, "Randomization Based Probabilistic Approach to Detect Trojan Circuits," in *11th IEEE High Assurance Sys. Eng. Symp.*, 2008, pp. 117–124.

[5] L. N. Nguyen, B. B. Yilmaz, M. Prvulovic, and A. Zajic, "A Novel Golden-Chip-Free Clustering Technique Using Backscattering Side Channel for Hardware Trojan Detection," in *IEEE HOST*, 2020, pp. 1–12.

[6] Y. Huang, S. Bhunia, and P. Mishra, "Scalable Test Generation for Trojan Detection Using Side Channel Analysis," *IEEE TIFS*, vol. 13, no. 11, pp. 2746–2760, 2018.

[7] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A Novel Technique for Improving Hardware Trojan Detection and Reducing Trojan Activation Time," *IEEE TVLSI*, vol. 20, no. 1, pp. 112–125, 2012.

[8] S. R. Hasan, C. A. Kamhoua, K. A. Kwiat, and L. Njilla, "Translating circuit behavior manifestations of hardware Trojans using model checkers into run-time Trojan detection monitors," in *IEEE AsianHOST*, 2016, pp. 1–6.

[9] R. S. Chakraborty, S. Pagliarini, J. Mathew, S. R. Rajendran, and M. N. Devi, "A Flexible Online Checking Technique to Enhance Hardware Trojan Horse Detectability by Reliability Analysis," *IEEE Trans. on Emerging Topics in Computing*, vol. 5, no. 2, pp. 260–270, 2017.

[10] T. F. Wu, K. Ganesan, Y. A. Hu, H.-S. P. Wong, S. Wong, and S. Mitra, "TPAD: Hardware Trojan Prevention and Detection for Trusted Integrated Circuits," *IEEE TCAD*, vol. 35, no. 4, pp. 521–534, 2016.

[11] H. D. Foster, A. C. Krolnik, and D. J. Lacey, *Assertion-Based Design*, 2nd ed. Springer-Verlag, 2005.

[12] "IEEE Standard for Property Specification Language (PSL)," *IEEE Std 1850-2010 (Revision of IEEE Std 1850-2005)*, pp. 1–182, 2010.

[13] "IEEE Standard for SystemVerilog–Unified Hardware Design, Specification, and Verification Language," *IEEE Std 1800-2017 (Revision of IEEE Std 1800-2012)*, pp. 1–1315, 2018.

[14] Trust-Hub benchmarks. [Online]. Available: https://trust-hub.org

[15] F. Corno, M. Reorda, and G. Squillero, "RT-level ITC'99 benchmarks and first ATPG results," *IEEE Des. Test Cmput.*, vol. 17, no. 3, pp. 44–53, 2000.

[16] H. Salmani, M. Tehranipoor, and R. Karri, "On design vulnerability analysis and trust benchmarks development," in *IEEE 31st Intl. Conf. on Computer Design (ICCD)*, 2013, pp. 471–474.

[17] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, "Benchmarking of Hardware Trojans and Maliciously Affected Circuits," *J. of Hardw. and Sys. Security*, vol. 1, no. 1, pp. 85–102, 2017.

[18] W. Cullyer, "Implementing high integrity systems: the VIPER microprocessor," *IEEE Aerospace and Electronic Systems Magazine*, vol. 4, no. 6, pp. 5–13, 1989.

[19] M. Boulé and Z. Zilic, "Automata-based assertion-checker synthesis of PSL properties," *ACM TDAES*, vol. 13, no. 1, pp. 1–21, 2008.

[20] OpenTitan Documentation Index. [Online]. Available: https://docs.opentitan.org/doc

Appendix 2

II

M. Eslami, J. Knechtel, O. Sinanoglu, R. Karri, and S. Pagliarini, “Benchmarking Advanced Security Closure of Physical Layouts: ISPD 2023 Contest,” In Proceedings of the 2023 International Symposium on Physical Design (ISPD ’23), New York, NY, USA, 2023, pp. 256–264. DOI: https://doi.org/10.1145/3569052.3578924

Benchmarking Advanced Security Closure of Physical Layouts

ISPD 2023 Contest

Mohammad Eslami*
mohammad.eslami@taltech.ee
Tallinn University of Technology
Estonia

Johann Knechtel*
johann@nyu.edu
New York University Abu Dhabi
UAE

Ozgur Sinanoglu
ozgursin@nyu.edu
New York University Abu Dhabi
UAE

Ramesh Karri
rkarri@nyu.edu
New York University
USA

Samuel Pagliarini
samuel.pagliarini@taltech.ee
Tallinn University of Technology
Estonia

ABSTRACT

Computer-aided design (CAD) tools traditionally optimize "only" for power, performance, and area (PPA). However, given the wide range of hardware-security threats that have emerged, future CAD flows must also incorporate techniques for designing secure and trustworthy integrated circuits (ICs). This is because threats that are not addressed during design time will inevitably be exploited in the field, where system vulnerabilities induced by ICs are almost impossible to fix. However, there is currently little experience for designing secure ICs within the CAD community.

This contest seeks to actively engage with the community to close this gap. The theme is security closure of physical layouts, that is, hardening the physical layouts at design time against threats that are executed post-design time. Acting as security engineers, contest participants will proactively analyse and fix the vulnerabilities of benchmark layouts in a blue-team approach. Benchmarks and submissions are based on the generic DEF format and related files.

This contest is focused on the threat of Trojans, with challenging aspects for physical design in general and for hindering Trojan insertion in particular. For one, layouts are based on the ASAP7 library and rules are strict, e.g., no DRC issues and no timing violations are allowed at all. In the alpha/qualifying round, submissions are evaluated using first-order metrics focused on exploitable placement and routing resources, whereas in the final round, submissions are thoroughly evaluated (red-teamed) through actual insertion of different Trojans.

CCS CONCEPTS

• **Security and privacy → Security in hardware**; • **Hardware → Physical design (EDA)**.

*Both authors contributed equally to this research.

ISPD '23, March 26–29, 2023, Virtual Event, USA

ACM ISBN 978-1-4503-9978-4/23/03…$15.00
https://doi.org/10.1145/3569052.3578924

KEYWORDS

Hardware Security; Physical Design; Security Closure; Contest; Hardware Trojans; ASAP7;

ACM Reference Format:
Mohammad Eslami, Johann Knechtel, Ozgur Sinanoglu, Ramesh Karri, and Samuel Pagliarini. 2023. Benchmarking Advanced Security Closure of Physical Layouts: ISPD 2023 Contest. In *Proceedings of the 2023 International Symposium on Physical Design (ISPD '23), March 26–29, 2023, Virtual Event, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3569052.3578924

1 INTRODUCTION

This paper presents the second contest on *security closure* of physical layouts, i.e., on the challenge of hardening the physical layouts of integrated circuits (ICs) at design time against hardware-security threats that are executed post-design time.

Bringing this topic to the physical-design community is important for multiple reasons. First, many threats for hardware security, like Trojan insertion or side-channel attacks, are directly targeting for vulnerabilities of the physical layouts. Second, threats that are not mitigated during design-time are almost impossible to fix later on; ICs are unlike patchable software. Third, even if efforts are taken toward secure design at higher abstraction layers (like logic synthesis), such efforts may be undermined later on by, e.g., power, performance, and area (PPA) optimization, thus becoming futile without dedicated support for security closure at layout level.

This paper is organized as follows. We outline the theme, general approach, and some logistics in this Sec. 1. In Sec. 2, we discuss the scope and background for the contest and outline tasks as well as possible directions for solving them. In Sec. 3 we describe the implementation and evaluation of the contest in detail. The contest website [15] provides further information; importantly, all benchmarks and results will remain online there after the contest concludes, to stimulate further interest from the community.

1.1 Theme and Context

Securing the omnipresent information technology is an important but tough endeavour that requires efforts all the way from software down to the hardware. For the design, manufacturing, and deployment of ICs, there are numerous companies and partners involved within complex and world-wide supply chains. ICs run through many hands, where some of those may be acting with malicious

intent. Furthermore, once ICs are deployed in the field, an even larger attack surface arises. Most relevant for the physical-design community is that computer-aided design (CAD) tools traditionally optimize "only" for PPA, whereas modern CAD flows should also incorporate techniques for secure IC design. See also, e.g., [9, 13, 16, 17] for further reading.

This contest is part of the International Symposium on Physical Design (ISPD) 2023. Participants will focus on securing the physical layout of ICs. Acting as security engineers, participants will iteratively and proactively evaluate and fix the vulnerabilities of IC layouts at design-time.

The threat to consider in this contest—Trojan insertion—represents a scenario with clear relation to physical design for defending against. Further, the contest scope is well constrained, thereby easing the ramp-up for participants new to hardware security.

1.2 Objective and General Approach

The objective of this contest is the following. Implement physical-design measures to proactively harden layouts against post-design Trojan insertion, conducted during mask generation or manufacturing. See Sec. 2.1 for more details on this threat scenario.

To achieve this objective, participants would want to, e.g., revise placement and routing such that insertion of Trojan trigger components as well as routing trigger and payload components becomes difficult, all while accounting for the impact on design checks and PPA by the defense approach. Given that there are different, possibly competing metrics to be considered for design quality and security closure at once, some machine learning-based guidance could be promising here. In any case, there is no single, right or wrong approach toward that end—it is up to the participants' creativity and skills to come up with the best defense schemes.

Participants can work on any physical-design platform of their choice, be it commercial tools, prominent open-source tools like *OpenROAD* [11], or custom in-house tools.[1] In any case, before devising or even implementing some defense measures, participants would want to i) understand the scope in general and the threat in particular (Sec. 2), ii) understand the way the threat is considered and scored for this contest (Sec. 3.4), and iii) be as creative as possible for security closure while not "re-inventing the wheel" for core CAD algorithms and design techniques.

1.3 Logistics

This contest is open to students of all levels (undergrads, graduates, and/or post-graduates) as well as practitioners from industry, with prizes limited to academic participants.

The benchmarks are physical layouts of various crypto cores. We provide all relevant files along with the layouts. See Sec. 3.3 and the contest website [15] for more details.

The scoring employs a weighted function considering security metrics as well as design-quality metrics. There are also constraints to be considered, importantly that no design rule check (DRC) and no timing violations are allowed. See Sec. 3.4 for more details.

[1] However, the use of commercial tools, in particular Cadence Innovus, is recommended for quicker ramp-up. We have implemented and thoroughly evaluated a related reference design flow (see Sec. 3.3.3, also including modifications as needed for the library of choice, ASAP7 [4] (see Sec. 3.2).

There is an alpha/qualifying round, where we provide a set of benchmarks early on. (Even before that, we release some sample benchmarks.) All participants that submit, for each benchmark, some valid solution(s) providing any improvement for the overall score, move on to the final round. There, we ramp up the challenge; while the alpha round considers only first-order metrics for Trojan insertion, the final round considers actual Trojan insertion for more thorough and realistic assessment of the security of the layouts submitted by participants. During both the alpha and final rounds, results and current rankings are shared regularly for those participants who opt in; this is meant to spur the contest throughout the whole timeline which spans several months.

Participants can interact with the organizers through a dedicated mailing list. We also publish questions and answers (Q&As) regularly on the website [15].

The final results, rankings, and awards will be first announced at ISPD and then published on the contest website [15] as well. Cash prizes and award plaques will be given to the top three teams. Top teams are encouraged to disseminate their results and means for security closure further with the community, but that is not a requirement for participation.

2 BACKGROUND

2.1 Hardware Security

2.1.1 Overview. Due to the changes in the business model of manufacturing ICs, it is rarely the case anymore that circuits are designed and fabricated by the same entity [23]; companies outsource the fabrication process to third parties. This outsourcing brings new challenges to the security and trustworthiness of ICs since there is significantly less control and oversight during the fabrication process.

The main threats during the fabrication stage include Trojan insertion, IP piracy, and illegal overproduction [5, 19, 22]. For example, IP piracy refers to copying and illegal sale or reuse of intellectual design property extracted from the chip during fabrication. For hardware security in general, there is a wide range of other threats as well, including physical attacks to retrieve sensitive information in the field [13].

Hardware Trojan is a generic term for malicious modifications to a design, either via addition, subtraction, or replacement of the existing logic. A rogue engineer in a concerned third-party company could, potentially, manipulate the intended behavior of a chip, either in parts or in the whole batch of the production line (Fig. 1). The malicious intent might be to disrupt functionality, leak information, damage the chip, or reduce the performance by increasing the power consumption or temperature, etc. [12].

2.1.2 Hardware Trojans. As indicated, a Hardware Trojan may leak sensitive information (i.e. secret key used in a crypto core) or change the original behavior of the design so that it could destroy the chip during operation. An activation mechanism, also called trigger, is typically based on specific and rare combinational and/or sequential conditions. Once the trigger condition is met, the Trojan main's circuitry, also called payload, performs its malicious operation.

In case a Trojan has a significant impact on the PPA or/and on the functionality, or in case its size is relatively large, it might be

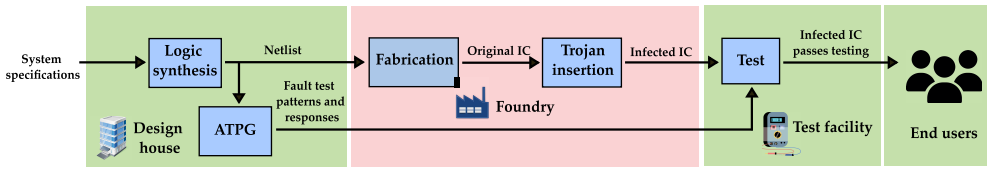


Figure 1: The simplified IC supply chain from design to end-user, including the threat model for this contest. The foundry is considered untrusted, thus marked red; an adversary would insert some Trojan during fabrication itself. Reused from [18] with permission.

easy to detected. Hence, an adversary should smartly design and insert Trojans so that they remain hidden during testing and normal operation [10, 27].

Hardware Trojans are difficult to identify using traditional post-manufacturing testing that uses functional, structural, or random patterns [7, 26]. This is because the generation and application of manufacturing tests aim to find flaws or unacceptably wide changes in device parameters that lead to divergence from functional or parametric specifications. Such tests are, however, not well suited to determine malicious but rare, additional functionality brought on by Trojans or to determine alterations in circuit behavior brought on by random unusual events. When seeking to adopting a post-silicon test/validation method to reliably detect Trojans, there are a number of significant hurdles.[2] Thus, other approaches like design-time security closure against Trojan insertion are important [25].

2.1.3 Security Closure Against Hardware Trojans. As outlined, Trojan detection and diagnostic approaches have certain limitations, including the identification of uncommon nodes, process variation, and measurement error. Thus, ICs should be devised with some self-protection awareness if these methods are to be made more effective. The primary modes for Trojan prevention, at the moment, include obfuscation, layout filling, split manufacturing, and insertion of self-testing circuitry [20, 25, 26]. Techniques for bridging layout gaps with functional logic are suggested to make Trojan detection easier and decrease the possibility of Trojan insertion [25, 26].

2.2 Threat Model

For this year's contest, we consider Hardware Trojans inserted at the post-layout stage as the threat to be tackled. Adversaries have access to all technology details and cell libraries used by the victim to create the layout because they are familiar with the foundry's manufacturing process. However, we assume that adversaries are unaware of the specific timing/power limitations, clock domains, input/output pin functionality, or high-level functionality of the victim's design, while they are knowledgeable about IC design and have access to contemporary EDA tools.

[2]First, an adversary can come up with a large number of possible Trojan types and realize excessively many different versions of all shapes and sizes. Thus, Trojans differ greatly in terms of their structural and functional characteristics, and also because of their covert nature, it can be very difficult to activate some unknown Trojans and observe their results. Therefore, deterministic and even exhaustive testing methods seem to be impractical. Process variation and measurement noise also present themselves as significant obstacles to observing Trojan effects in physical parameters, such as delay and supply current. Increased process variation in advanced technologies in particular can conceal Trojan-related effects in physical parameters.

This threat model is compatible with state-of-the-art work, in particular with recently demonstrated Trojan insertion in actual silicon [21].

3 IMPLEMENTATION AND EVALUATION

3.1 Platform

3.1.1 Tool Flow for Participants. Recall that participants are free to use any physical-design tools of their choice, be they commercial, open-source, or in-house tools. Still, as this is the second contest in a row, it is more advanced and demanding, also because the ASAP7 library is used (see Sec. 3.2). Thus, we recommend that participants employ the suggested tools and the provided reference design flow for quicker ramp-up.

3.1.2 Backend for Organizers. The evaluation backend is based on commercial design tools, including Cadence Genus, Innovus, Conformal, etc. The actual evaluation, the parsing of reports, the computation of metrics and scores, and the file management tasks are all implemented using *tcl* and *bash* scripting.

The backend is implemented as a daemon. It supports parallel processing of various submissions from different teams, and further supports parallel processing of all calls to commercial tools for individual submissions. This implementation approach significantly reduces the runtime. The workflow is as follows:

(1) Initialize. Global runtime variables, like all the uniform resource locators (URLs) for the participants' private submissions sites are retrieved and memorized, enabling faster access later on. Local work and backup folders are initialized. This step supports 'testing' versus 'production' modes; the daemon is run separately twice on our server to support both modes in parallel.
(2) Download of new submissions. Any new submissions are downloaded and queued for evaluation, considering the current overall workload of the backend and the currently ongoing runs (if any) of the concerned participants. For fairness, all participating teams are given some upper limits for parallel processing.
(3) Evaluation. Once some submissions pass the queue, evaluation is started, and an email notification is sent to the concerned participants. The evaluation itself is conducted in multiple steps, with basic design checks done first and more complicated evaluation next and in parallel. All these valuation steps are implemented in *tcl* and *bash* scripts and can be easily updated, without the need to revise the backend daemon itself.

(4) Upload of new results. Once the evaluation is done, a follow-up email notification is sent, which also contains the scores and important messages, like current processing status (for all other submissions) and errors or warnings observed for the particular submission.

Steps (2)–(4) are repeated periodically and automatically by the daemon. All daemon procedures also feature status monitoring and logging for robust processing. An exemplary screenshot of the daemon is shown in Fig. 2.

While specifically developed and tailored for this contest, the backend scripts are also written with some flexibility in mind. We are releasing all backend scripts at [14] as these can be helpful to the community for implementation of other contests.

We did not release the backend scripts to the participants during the contest itself. Doing so would not have provided any benefit to the participants, as all the important evaluation and scoring scripts are already released along with the benchmarks. Thus, participants are able to run all important scripts independently at their end, to help implementing and debugging their ideas.

3.1.3 Frontend for Participants. With the outlined workflow for the backend, we require some frontend for exchange of submission and result files. With reliability and world-wide availability in mind, we opt for *Google Drive* as web frontend.

All registered teams are provided access to their dedicated Google Drive folder. Teams may upload submission files anytime, upon which new files are automatically downloaded by our backend for evaluation. Results are returned into the teams' respective benchmark folders. Results will include the overall score but also report and log files as generated by our backend, to provide participants with more detailed insights.

3.2 ASAP7 PDK and Library

One of the significant changes in the present edition of the contest is the adoption of a PDK that is much more modern. The ASAP7 PDK [6], originally developed by the team from Arizona State and ARM, is likely the most complete PDK developed by academia. The same team also provides a standard cell library [24] for their PDK, one that utilizes FinFET transistors and is properly characterized. The many files provided do resemble a commercial PDK, including multi-Vth cells, extraction decks, DRC decks, and so on.

For this contest, a few modifications were made to the library, mainly to ensure that participants could use different physical-synthesis tools and versions with ease. Some details are given next.

A few complex via rules were dropped, while still maintaining the major features of the technology. In tandem, we have also added new design rules that were not part of the original technology setup; we added those to create interesting and challenging scenarios for the participants to work around. One of the most significant addition has been the notion of colored metals, i.e., metal layers that would be fabricated using more than one mask. This departure from the original setup of the ASAP7 PDK introduces some challenges that are common in the first generation of FinFET technologies.[3] We have also introduced max density rules for all metal layers; this is meant to discourage the participants from adopting some simple shield-based solutions to protect their layouts.

More details can be obtained from the technology LEF file provided in the benchmarks release [15], or directly from our repository containing the modified ASAP7 technology and the reference design flow [4]. Note that all significant changes are annotated as comments.

3.3 Benchmarks

3.3.1 Overview. We consider 6 different crypto cores as benchmarks: AES128, Camellia, CAST, MISTY, SEED, and SHA256. These cores have different sizes and complexities, ensuring different difficulty levels across the benchmarks.

For all benchmarks, we first pass the RTL description [1–3] to the logic synthesis tool, Cadence Genus, followed by the physical implementation using Cadence Innovus. For each benchmark, we use custom timing constraints while the same ASAP7 library is used for all benchmarks. For logical and physical synthesis, we have refrained from utilizing optimization on purpose, namely to keep some margin for the participants to work with. In other words, we enable participants to explore trade-offs between security and design metrics. The vanilla scripts for logical and physical synthesis have been made available early on [4, 15], to help the participants adapt to the ASAP7 library. More details for the physical design are provided in Sec. 3.3.3.

The benchmark release includes the post-route Verilog netlist and DEF file. It also contains the design database, the SDC timing files, the reports for the evaluation and scoring of the baseline layout, the evaluation and scoring scripts, and list of *cell assets*.

Cell assets are selected manually from all flip-flops (FFs) to represent potential locations, like key registers, that some Trojan could connect to for its trigger and/or payload. Note that, for the alpha round, participants are informed about all the assets, but there is no specific evaluation or scoring related to assets. In the final round, specific Trojans will be targeting at subsets of those assets. In any case, all assets must be maintained by participants.

3.3.2 Sample Benchmark: SHA256. Early on, a sample benchmark, SHA256, was provided for the participants as a warm-up design to help them get familiar with the ASAP7 library as well as adapt to strict rules and constraints of the contest. We include this sample benchmark along with the other benchmarks for both alpha and final rounds since the same strategy and implementation flow are used for the sample and the final benchmarks.

Fig. 3 shows layout details for the SHA256 benchmark.

3.3.3 Implementation Flow. Next, we provide more details about the physical implementation. The script for the sample benchmark SHA256 is made available early on at [4]. It can be used by participants for other benchmarks, requiring only minor customizations for different designs. This script includes the following steps:

(1) **Defining Globals:** Global variables like the version of the tool used, the design technology, and the number of available processor cores are set, along with all the paths for the

[3] In more mature technologies, where two metal shapes of the same layer are drawn side by side, the minimum distance between them depends on the width and parallel-run length of the shapes. In technologies where coloring is considered, the minimum distance changes also depending on whether the metals are in the same color or in different colors.

```
ISPD23 -- 1)
ISPD23 -- 1)  Checking team folder "_test_TalTech" (Google team folder ID "1b2vTk-1V7GAs2ARlLQt-u-2WMip6zBTq") for new submission files ...
ISPD23 -- 1)  Checking team folder "_test" (Google team folder ID "1ssu33v9fAXssAPXnqXLbQ-fkDFzxzIDc") for new submission files ...
ISPD23 -- 1)  Download new submission file "aes.zip" (Google file ID "1BOjH4FSW5OkgRW4ChMhQMDDrO0vPcp6I") into dedicated folder "/data/nyu_projects/ISPD23/data/alpha/_test/aes/downloads/downloads_16
Downloading aes.zip -> /data/nyu_projects/ISPD23/data/alpha/_test/aes/downloads/downloads_1675075835/aes.zip
Downloaded 1BOjH4FSW5OkgRW4ChMhQMDDrO0vPcpI at 17.4 MB/s, total 39.0 MB
ISPD23 -- 1)   Unpacking zip file "aes.zip" into dedicated folder "/data/nyu_projects/ISPD23/data/alpha/_test/aes/downloads/downloads_1675075835" ...
ISPD23 -- 1)
ISPD23 -- 1) Done
ISPD23 --
ISPD23 -- 2) Start evaluation processing of newly downloaded submission files, if any ...
ISPD23 -- 2)  Time: Mon Jan 30 14:50:45 +04 2023
ISPD23 -- 2)  Time stamp: 1675075845
ISPD23 -- 2)
ISPD23 -- 2)  [ _test_TalTech ]: Currently 0 run(s) ongoing, 0 more run(s) queued, and would be allowed to start 6 more run(s).
ISPD23 -- 2)  [ alpha -- _test        -- aes      -- 1675075835 ]: Start processing within dedicated work folder "/data/nyu_projects/ISPD23/data/alpha/_test/aes/work/downloads_1675075835" ...
ISPD23 -- 2)  [ alpha -- _test        -- aes      -- 1675075835 ]:  Send out email about processing start ...
ISPD23 -- 2)  [ _test        ]: Currently 1 run(s) ongoing, 0 more run(s) queued, and would be allowed to start 5 more run(s).
ISPD23 -- 2)  [ alpha -- _test        -- aes      -- 1675075835 ]:  Init work folder ...
ISPD23 -- 2)  [ alpha -- _test        -- aes      -- 1675075835 ]:  Basic checks ...
ISPD23 -- 2)  [ alpha -- _test        -- aes      -- 1675075835 ]:   Assets check ...
ISPD23 -- 2)  [ alpha -- _test        -- aes      -- 1675075835 ]:   Pins check ...
ISPD23 -- 2)  [ alpha -- _test        -- aes      -- 1675075835 ]:  Assets check passed.
ISPD23 -- 2)  [ alpha -- _test        -- aes      -- 1675075835 ]:   Pins check passed.
ISPD23 -- 2)  [ alpha -- _test        -- aes      -- 1675075835 ]:  All basic checks passed.
ISPD23 -- 2)  [ alpha -- _test        -- aes      -- 1675075835 ]:  Starting LEC design checks ...
ISPD23 -- 2)  [ alpha -- _test        -- aes      -- 1675075835 ]:  Starting Innovus design checks ...
ISPD23 -- 2)
ISPD23 -- 2) Done
```

Figure 2: Screenshot of the backend daemon working in 'testing' mode. Listed is the download, initial design checks, and start of commercial tools for detailed evaluation, all for one submission of the AES128 benchmark.

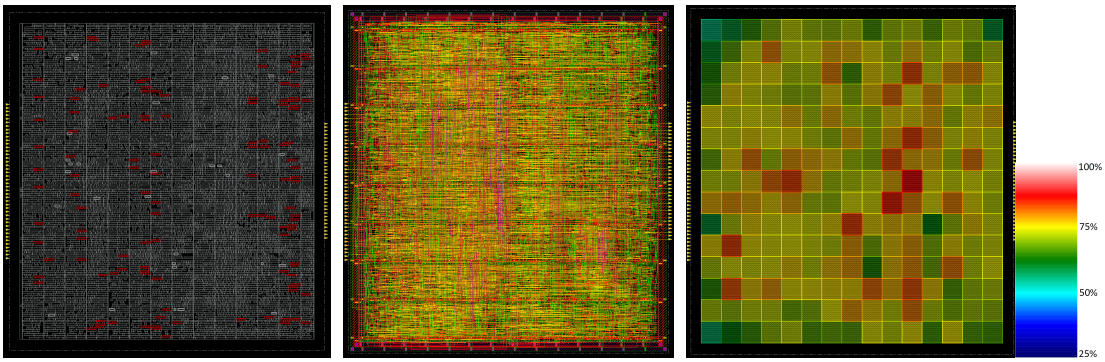


Figure 3: Layout details for the SHA256 benchmark. (Left) placement with cell assets highlighted in red. Note that input pins are placed on the left side, whereas output pins are placed on the right side. (Middle) routing layers. (Right) cell density map.

initial netlist, libraries, LEF files, and timing constraint files. Participants are not allowed to use different library files.

(2) **Floorplanning:** The size of the floorplan should be defined properly based on which benchmark is being implemented. Participants are free to define the floorplan and aspect ratio. At the same time, power planning (see also further below) has to be accounted for, including parameters for ring spacing, ring offset, ring size, stripe frequency and stripe-to-stripe distance. These parameters are fixed for all designs; participants have to maintain the same floorplan/powerplan strategy for fairness.

(3) **Pin Assignment:** The location of IO pins can influence the quality of the design. Thus, we place and constrain all input pins to the left side of the designs and all the output pins on the right side.

(4) **Power Distribution Network:** The ASAP7 PDK and library are rather restrictive regarding how power stripes can be defined. There are only a few combinations of metal layers, width, spacing, and offset that can generate a coherent power network with adequate via arrays. Taking this into account, the core rings are specified to be routed using M6 and M7 metal layers. For the standard-cell rails, follow pins appear in both M1 and M2 in what is called "stapled style." Finally, the vertical and horizontal stripes are specified to be routed using M3 and M4 metal layers, respectively. Once all these parameters are set, the power distribution network (PDN) is routed and power vias are generated. Participants should not modify this power distribution strategy, except for adjusting it to smaller/larger floorplans.

(5) **Place and Route:** First, standard cells are placed in the core area. If the floorplan is too small to fit all standard cells, participants should revisit floorplanning and resize the floorplan accordingly. Once the placement of standard cells is passed, the clock is ready to be distributed (clock tree synthesis, CTS). After performing CTS, the design is ready to be routed. Routing is one of the last steps in physical design and takes typically the largest share of the implementation time. Note that, after routing, it is very likely that some violations occur, due to different reasons. Solving these violations, especially those related to pin access, which can become very challenging for dense layouts, is also part of the challenge put forward in this contest.

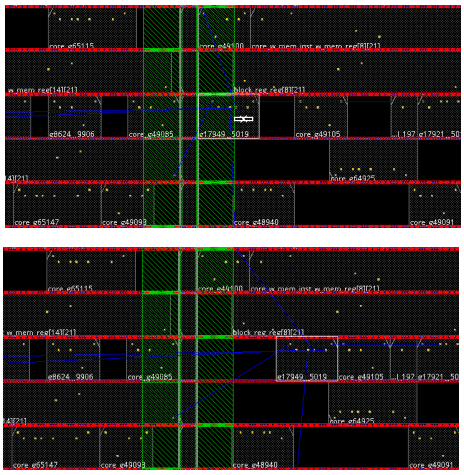


Figure 4: An example for fixing pin access DRC violations. (Top) DRC violation around a power stripe. (Bottom) Fixing the violation by moving some instances.

(6) **Generating Reports, Exporting Final Layout:** Once the design passed all necessary checks and verification, it is ready to be exported. Furthermore, post-route reports for PPA, etc., can be generated. Participants are required to generate DEF files and post-route netlist files for submission.

As indicated, with the reference design flow [4], some DRC violations are *expected* especially for pin access around the power stripes. These violations can be easily fixed manually, by moving the standard cells away from the power stripes (Fig. 4). For larger designs, a semi-automated approach to detect and fix these violations might be devised by the participants.

3.3.4 Alpha, Final Rounds Benchmarks. After the warm-up phase, 5 crypto cores (AES128, Camellia, CAST, MISTY, and SEED) are added to the sample benchmark (SHA256). As indicated, the implementation flow for logical and physical synthesis of these benchmarks is the same as for the sample benchmarks. The only differences are using specific timing constraints and different floorplan sizes for each design.

As mentioned before, the benchmarks exhibit different levels of complexity, size, and density; benchmarks can be classified into categories from 'easy' to 'difficult.' The cell density maps for selected benchmarks are shown in Fig. 5. There, red marks high-density areas, while green and blue marks low-density areas. Note that the layouts underlying for Fig. 5 are not in the same scale; thus, different grid sizes are used for comparable representation. Details for layout dimensions and grid sizes are given in Table 1.

Another parameter/metric for the layout complexity is routing congestion and utilization in each metal layer. Routed layouts are shown in Fig. 6. For example, for AES128 (left-most subfigure), more pink areas indicate that this required is more utilized as it required more routing within the top metal layer (pink = M7).

Table 1: Specification of the benchmark layouts

Benchmarks	Dimensions (µm)	Density Grid (# Rows)
AES128	822.44 × 822.44	14 × 14
Camellia	158.24 × 158.24	8 × 8
CAST	293.24 × 293.24	14 × 14
MISTY	174.44 × 174.44	10 × 10
SEED	206.84 × 206.84	10 × 10
SHA256	190.64 × 190.64	11 × 11

3.4 Metrics and Scoring

For security evaluation, we consider two sets of metrics and phases for scoring this year. In the first phase, i.e., for the alpha/qualifying round, submissions are evaluated using first-order metrics. In the second phase, i.e., for the final round, we extend these simple metrics with results for actual Trojan insertion.

3.4.1 Design Metrics, First-Order Security Metrics (Alpha Round). For evaluating the quality of the design, PPA metrics (power; worst negative slack, WNS; area) are considered. For evaluating the resilience, security metrics describe the layout resources remaining for Trojan insertion. (Thus, participants should reduce unused resources, i.e., open placement sites and free routing tracks, as much as possible for better scoring.) Both security and PPA metrics are evaluated over the baseline to obtain the scoring.

Next, the metrics are categorized.

(1) Security – *sec*
  (a) Trojan insertion – *sec_ti*
    (i) Placement sites of exploitable regions (ers)
      - Max # of sites across all ers – *sec_ti_sts_max*
      - Median # of sites across all ers – *sec_ti_sts_med*
      - Total # of sites across all ers – *sec_ti_sts_sum*
    (ii) Routing resources of exploitable regions (ers)
      - Total # of free tracks across all ers – *sec_ti_fts_sum*
      - Note that, for each exploitable region, free tracks are summed up across all metal layers.
(2) Design quality – *des*
  (a) Power
    - Total power – *des_pwr_tot*
  (b) Performance
    - Worst neg. slack, setup timing req. – *des_prf_WNS_set*
    - Worst neg. slack, hold timing req. – *des_prf_WNS_hld*
  (c) Area
    - Total die area (*not* standard cell area) – *des_ara_die*

3.4.2 Actual Trojan Insertion (Final Round). For each design, we attempt actual insertion of different Trojans. The possible outcomes—from the participant's perspective as defenders—would be 'fail' if the Trojan insertion is successful, 'partial pass' if the insertion succeeds but does compromise timing of the design, and 'full pass' if the insertion fails, e.g., induces some DRC violations. Note that this scoring can be further augmented with other metrics.

For the task of Trojan insertion, we use an ECO-based flow similar to that proposed in [8]. We use three different types of Trojans that: (i) leak information, (ii) modify the output value of FFs, and (iii) over-consume power. For the first two types, the target is chosen from the cell assets. The third type can be connected to

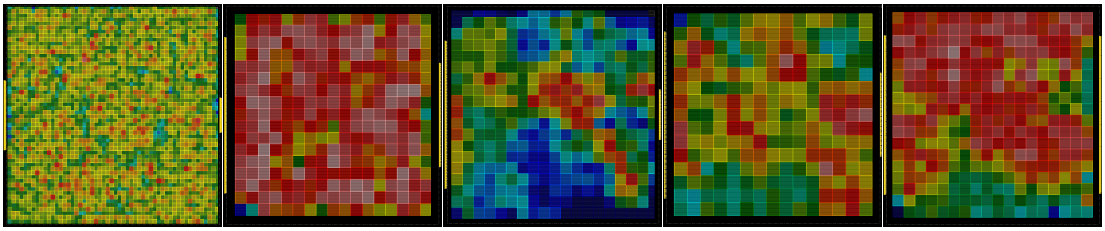


Figure 5: Cell density maps for AES128, Camellia, CAST, MISTY, and SEED (left to right). See Fig. 3 for legend.

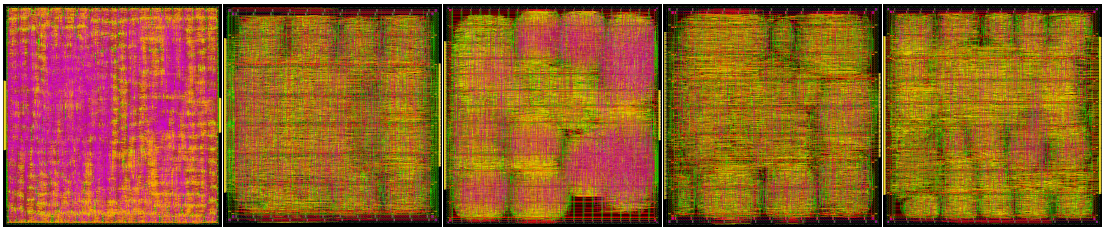


Figure 6: Routed layouts for AES128, Camellia, CAST, MISTY, and SEED (left to right). The same number of metal layers is used for all benchmarks. The highest metal in the stack is M7, represented in pink color.

any location of the design since it is only dependent on the clock and the triggering condition [8]. Furthermore, we consider different versions for each type, as in varying number of triggering bits and number of the payload bits.

An exemplary Trojan with a 16-bit trigger (utilizing the original design) and a 5-bit payload is outlined in Fig. 7. As shown, the Trojan requires only few additional instances which makes it a practical and relevant example; it would likely be hard to spot by conventional Trojan detection. The Trojan's impact on timing is depicted in Fig. 8. The bars in the top subfigure show the number of paths with corresponding timing slack before inserting the Trojan, while the bottom subfigure shows the paths after inserting the Trojan. As shown, the Trojan does not have a considerable impact on the timing, further hindering its detection.

3.4.3 Scoring. Next, we provide more details for the calculation of first-order metrics and scores. The most important settings and considerations are as follows:

(1) Timing checks, logical equivalence, PDN checks, as well as DRC checks, are all hard constraints, i.e., must be met.
   - Thus, as timing checks are based on WNS, only positive slack values are accepted and considered for scoring.

(2) Not considered for scoring are further design checks, like checks for placement and routing issues like dangling wires etc.; see [15] for more details on these checks.
   - Thus, participants can neither improve nor worsen their scores by fixing or worsening those design checks.
   - However, all checks are considered as soft constraints with a margin of +10 issues—any deviation above these margins is considered as a fail.

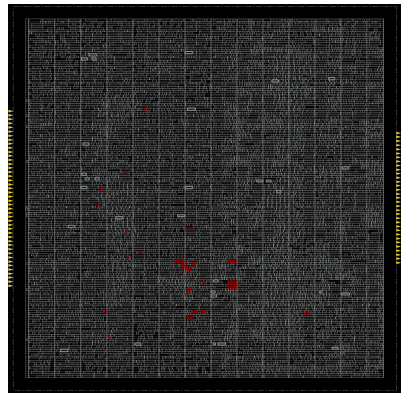


Figure 7: An exemplary Trojan inserted into the SHA256 benchmark. Additional components are highlighted in red.

(3) All metrics are normalized to their respective nominal baseline values, obtained from the provided benchmark layouts. A submission that improves on some metric will be scored a related value between 0 and 1, whereas a deteriorated layout be scored a value greater than 1.
   - For positive WNS values, this means to compute 'baseline_WNS' / 'submission_WNS.'
   - For all other metrics 'm', this means to compute 'submission_m' / 'baseline_m.'

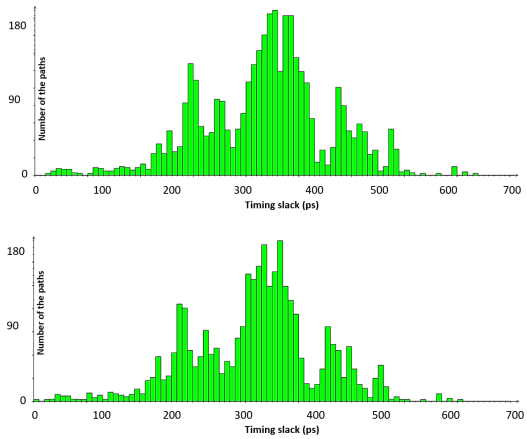


Figure 8: Impact of the inserted Trojan circuitry on timing for the SHA256 benchmark. Distribution of timing paths before Trojan insertion (top) versus after (bottom).

(4) Such normalized scoring is more sensitive to deterioration than it is to improvements. This is on purpose; the main objective is to further improve the layouts, not deteriorate them, so deterioration for any metric(s) should have a relatively large detrimental impact on the overall score.

The actual score calculation is shown below.

$$\begin{aligned} score &= (sec + des)/2 \\ &= ((1/2 \times sec\_ti\_sts + 1/2 \times sec\_ti\_fts) + (des)/2) \end{aligned} \quad (1)$$

with the calculation of score components detailed next, where s refers to the secured/submitted layout and b to the baseline layout.

(1) Trojan insertion – sec_ti
  (a) 50% weighted: placement sites of exploitable regions (sec_ti_sts)
    - 50% weighted: $score(sec\_ti\_sts\_sum) = sec\_ti\_sts\_sum(s)/sec\_ti\_sts\_sum(b)$
    - 33.3% weighted: $score(sec\_ti\_sts\_max) = sec\_ti\_sts\_max(s)/sec\_ti\_sts\_max(b)$
    - 16.6% weighted: $score(sec\_ti\_sts\_med) = sec\_ti\_sts\_med(s)/sec\_ti\_sts\_med(b)$
  (b) 50% weighted: routing resources of exploitable regions (sec_ti_fts)
    - $score(sec\_ti\_fts\_sum) = sec\_ti\_fts\_sum(s)/sec\_ti\_fts\_sum(b)$
(2) Design quality – des
  (a) 33.3% weighted: power (des_pwr)
    - $score(des\_pwr\_tot) = des\_pwr\_tot(s)/des\_pwr\_tot(b)$
  (b) 33.3% weighted: performance (des_prf)
    - 50% weighted: ($des\_prf\_WNS\_set$)
    - 50% weighted: ($des\_prf\_WNS\_hld$)
  (c) 33.3% weighted: area (des_ara)
    - $score(des\_ara\_die) = des\_ara\_die(s)/des\_ara\_die(b)$

3.5 Constraints

For a submission to be considered valid, all the following constraints have to be respected:

- Submissions cannot incorporate trivial defenses. Specifically, filler, decap, and tap cells are scrubbed and thus considered as free placement sites for evaluation of exploitable regions.
- Submissions must meet setup, hold timing checks using the provided SDC files for timing analysis.
- Submissions must have 0 DRC violations.
- Participants must maintain the overall functional equivalence of the underlying design. However, participants are free to revise (parts of) the design implementation, as long as this constraint and the next one are met.
- Participants must maintain the assets, i.e., sensitive components, which are declared along with each benchmark. More specifically, cells declared as assets cannot be removed or restructured. However, participants are free to revise the physical design of assets as well as other logic in general.
- Participants cannot design custom cells; only those cells defined in the provided LIB/LEF files can be utilized.
- Participants cannot revise the metal layers/metal stack.
- Participants must include a clock tree in their submission but are free to revise its implementation, as long as other constraints are met.
- Participants must follow the PDN recipe provided in the reference flow. The PDN structure's stripes are checked for dimensions, area, and locations.
- Submissions must maintain the general IO pin placement. More specifically, pins must remain placed at the left or right side assigned in the baseline layout, but actual pin locations (along the y-axis) can be revised.

4 CONCLUSION

The threat of hardware Trojans has been studied for more than two decades now; yet, this field is still actively researched. On the defensive side, the community lacks commonly adopted approaches for both detecting and preventing Trojans. On the offensive side, there are still doubts about the practicality of Trojans and the real capabilities of such adversaries. This contest, with its focus on advanced security closure against Trojans, is thus an important activity. We seek to educate the physical-design community about (i) hardware security in general, (ii) the key role which CAD tools play toward providing secure and trustworthy ICs, and (iii) the concept of red-team-blue-team security evaluation.

ACKNOWLEDGEMENTS

This work was supported in parts by the Center for Cyber Security (CCS) at NYU New York/Abu Dhabi (NYU/NYUAD) and by the EC through the European Social Fund in the context of the project "ICT programme". The organizers also are thankful for the support of the ISPD Organizing Committee.

REFERENCES

[1] 2007. *Cryptographic Hardware Project.* http://www.aoki.ecei.tohoku.ac.jp/crypto/

[2] 2012. *Freecores: tiny_aes128 implementation.* https://github.com/freecores/tiny_aes

[3] 2013. *Hardware implementation of the SHA-256.* https://github.com/secworks/sha256

[4] 2022. *Reference Design for a modified version of ASAP7nm.* https://github.com/Centre-for-Hardware-Security/asap7_reference_design

[5] Swarup Bhunia, Miron Abramovici, Dakshi Agrawal, Paul Bradley, Michael S. Hsiao, Jim Plusquellic, and Mohammad Tehranipoor. 2013. Protection Against Hardware Trojan Attacks: Towards a Comprehensive Solution. *Des. Test* 30, 3 (2013), 6–17. https://doi.org/10.1109/MDT.2012.2196252

[6] Lawrence T. Clark, Vinay Vashishtha, Lucian Shifren, Aditya Gujja, Saurabh Sinha, Brian Cline, Chandarasekaran Ramamurthy, and Greg Yeric. 2016. ASAP7: A 7-nm finFET predictive process design kit. *Microelectronics Journal* 53 (2016), 105–115. https://doi.org/10.1016/j.mejo.2016.04.006

[7] Vasudev Gohil, Satwik Patnaik, Hao Guo, Dileep Kalathil, and Jeyavijayan (JV) Rajendran. 2022. DETERRENT: Detecting Trojans Using Reinforcement Learning. In *Proc. Des. Autom. Conf.* 697–702. https://doi.org/10.1145/3489517.3530518

[8] Alexander Hepp, Tiago Perez, Samuel Pagliarini, and Georg Sigl. 2022. A Pragmatic Methodology for Blind Hardware Trojan Insertion in Finalized Layouts. In *Proc. Int. Conf. Comp.-Aided Des.* https://doi.org/10.1145/3508352.3549452

[9] W. Hu, C. H. Chang, A. Sengupta, S. Bhunia, R. Kastner, and H. Li. 2020. An Overview of Hardware Security and Trust: Threats, Countermeasures and Design Tools. *Trans. Comp.-Aided Des. Integ. Circ. Sys.* (2020). https://doi.org/10.1109/TCAD.2020.3047976

[10] Susmit Jha and Sumit Kumar Jha. 2008. Randomization Based Probabilistic Approach to Detect Trojan Circuits. In *High Assur. Sys. Eng. Symp.* 117–124. https://doi.org/10.1109/HASE.2008.37

[11] A. B. Kahng and T. Spyrou. 2021. The OpenROAD Project: Unleashing Hardware Innovation. In *Proc. GOMACTech.* https://theopenroadproject.org

[12] Ramesh Karri, Jeyavijayan Rajendran, Kurt Rosenfeld, and Mohammad Tehranipoor. 2010. Trustworthy Hardware: Identifying and Classifying Hardware Trojans. *Computer* 43, 10 (2010), 39–46. https://doi.org/10.1109/MC.2010.299

[13] J. Knechtel. 2021. Hardware Security for and beyond CMOS Technology. In *Proc. Int. Symp. Phys. Des.* https://doi.org/10.1145/3439706.3446902

[14] J. Knechtel. 2022–2023. *Backend Daemon Scripts.* https://github.com/DfX-NYUAD/backend_daemon_gdrive

[15] J. Knechtel et al. 2022–2023. *Advanced Security Closure of Physical Layouts.* https://wp.nyu.edu/ispd23_contest/

[16] J. Knechtel, J. Gopinath, J. Bhandari, M. Ashraf, H. Amrouch, S. Borkar, S.-K. Lim, O. Sinanoglu, and R. Karri. 2021. Security Closure of Physical Layouts. In *Proc. Int. Conf. Comp.-Aided Des.* https://doi.org/10.1109/ICCAD51958.2021.9643543

[17] J. Knechtel, E. B. Kavun, F. Regazzoni, A. Heuser, A. Chattopadhyay, D. Mukhopadhyay, S. Dey, Y. Fei, Y. Belenky, I. Levi, T. Güneysu, P. Schaumont, and I. Polian. 2020. Towards Secure Composition of Integrated Circuits and Electronic Systems: On the Role of EDA. In *Proc. Des. Autom. Test Europe.* https://doi.org/10.23919/DATE48585.2020.9116483

[18] Nimisha Limaye, Nikhil Rangarajan, Satwik Patnaik, Ozgur Sinanoglu, and Kanad Basu. 2022. PolyWorm: Leveraging Polymorphic Behavior to Implant Hardware Trojans. *Trans. Emerg. Top. Comp.* 10, 3 (2022), 1443–1455. https://doi.org/10.1109/TETC.2021.3090060

[19] Eric Love, Yier Jin, and Yiorgos Makris. 2012. Proof-Carrying Hardware Intellectual Property: A Pathway to Trusted Module Acquisition. *Trans. Inf. Forens. Sec.* 7, 1 (2012), 25–40. https://doi.org/10.1109/TIFS.2011.2160627

[20] Satwik Patnaik, Mohammed Ashraf, Ozgur Sinanoglu, and Johann Knechtel. 2019. A Modern Approach to IP Protection and Trojan Prevention: Split Manufacturing for 3D ICs and Obfuscation of Vertical Interconnects. *Trans. Emerg. Top. Comp.* 9 (2019), 1815–1834. Issue 4. https://doi.org/10.1109/TETC.2019.2933572

[21] Tiago Perez and Samuel Pagliarini. 2022. Hardware Trojan Insertion in Finalized Layouts: From Methodology to a Silicon Demonstration. *Trans. Comp.-Aided Des. Integ. Circ. Sys.* (2022). https://doi.org/10.1109/TCAD.2022.3223846

[22] Nikhil Rangarajan, Satwik Patnaik, Johann Knechtel, Shaloo Rakheja, and Ozgur Sinanoglu. 2022. *The Next Era in Hardware Security.* Springer. https://doi.org/10.1007/978-3-030-85792-9

[23] Mohammad Tehranipoor and Farinaz Koushanfar. 2010. A Survey of Hardware Trojan Taxonomy and Detection. *Des. Test* 27, 1 (2010), 10–25. https://doi.org/10.1109/MDT.2010.7

[24] Vinay Vashishtha, Manoj Vangala, and Lawrence T. Clark. 2017. ASAP7 predictive design kit development and cell design technology co-optimization: Invited paper. In *Proc. Int. Conf. Comp.-Aided Des.* 992–998. https://doi.org/10.1109/ICCAD.2017.8203889

[25] Fangzhou Wang, Qijing Wang, Bangqi Fu, Shui Jiang, Xiaopeng Zhang, Lilas Alrahis, Ozgur Sinanoglu, Johann Knechtel, Tsung-Yi Ho, and Evangeline F. Y. Young. 2023. Security Closure of IC Layouts Against Hardware Trojans. In *Proc. Int. Symp. Phys. Des.* https://doi.org/10.1145/3569052.3571878

[26] Kan Xiao and Mohammed Tehranipoor. 2013. BISA: Built-in self-authentication for preventing hardware Trojan insertion. In *Proc. Int. Symp. Hardw.-Orient. Sec. Trust.* 45–50. https://doi.org/10.1109/HST.2013.6581564

[27] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester. 2016. A2: Analog Malicious Hardware. In *Proc. Symp. Sec. Priv.* https://doi.org/10.1109/SP.2016.10

Appendix 3

III
M. Eslami, T. Ghasempouri and S. Pagliarini, “SCARF: Securing Chips with a Robust Framework against Fabrication-time Hardware Trojans,” *in IEEE Transactions on Computers*, pp. 1-15, 2024. DOI: https://doi: 10.1109/TC.2024.3449082

SCARF: Securing Chips with a Robust Framework against Fabrication-time Hardware Trojans

Mohammad Eslami, *Graduate Student Member, IEEE,* Tara Ghasempouri, *Member, IEEE,* Samuel Pagliarini, *Member, IEEE*

***Abstract*—The globalization of the semiconductor industry has introduced security challenges to Integrated Circuits (ICs), particularly those related to the threat of Hardware Trojans (HTs) – malicious logic that can be introduced during IC fabrication. While significant efforts are directed towards verifying the correctness and reliability of ICs, their security is often overlooked. In this paper, we propose a comprehensive framework that integrates a suite of methodologies for both front-end and back-end stages of design, aimed at enhancing the security of ICs. Initially, we outline a systematic methodology to transform existing verification assets into potent security checkers by repurposing verification assertions. To further improve security, we introduce an innovative methodology for integrating online monitors during physical synthesis – a back-end insertion providing an additional layer of defense. Experimental results demonstrate a significant increase in security, measured by our introduced metric, Security Coverage (SC), with a marginal rise in area and power consumption, typically under 20%. The insertion of online monitors during physical synthesis enhances security metrics by up to 33.5%. This holistic framework offers a comprehensive defense mechanism across the entire spectrum of IC design.**

***Index Terms*—IC Design, ASIC, Hardware Trojan Horse, Verification, Assertions, Online Checkers, DfHT.**

I. Introduction

THE fabrication of an Integrated Circuit (IC) is mostly performed in a fabless fashion, a model in which the fabrication of an IC is performed in other places rather than *inside* the design house. Globalization has commonly led to the widespread adoption of this model by most companies, primarily driven by the impracticality of establishing proprietary fabrication facilities that require substantial financial investments in the order of billions of dollars. Notably, even industry leaders such as Apple opt to outsource the fabrication of their chips to external entities [1]. This strategic approach highlights the economic sensibility of relying on specialized foundries for chip production, especially for advanced node technologies, allowing design houses to channel resources more efficiently towards innovation and design endeavors.

Nevertheless, while this fabless model offers numerous advantages, it also comes with a tangible dark side. When the chip is sent to the foundry for fabrication, there will be no guarantee that the returned chip precisely aligns with the initial specifications (i.e., it can be modified inside the foundry). Even small modifications to the chip's design can compromise its security and pose potential risks to human life or lead to financial losses. This modification is commonly referred to as Hardware Trojan (HT), which is a malicious alteration, addition, or subtraction to the original design with the intent to compromise its integrity [2], [3]. The goal of such manipulations can range from information leakage, functionality change, performance degradation, or the deliberate reduction of the chip's useful lifespan [4], [5]. An HT is composed of two parts, namely, the trigger and the payload. Trigger, as its name indicates, is the activation mechanism of the HT, while the payload is the saboteur function that is executed when the trigger is activated. The trigger part can be designed to activate under specific temporal and/or temperature conditions or through a combination of certain inputs or internal signals of the chip. A sophisticated attacker strategically sets the trigger to activate only under extremely rare conditions, thus ensuring that the HT remains undetected by conventional detection schemes during normal chip operation [2], [6].

Therefore, design companies should proactively implement measures to safeguard their chips against fabrication-time inserted HTs [7]–[10]. Moreover, the importance of this research cannot be emphasized enough. Historically, the primary focus during the verification stage within design houses has been on detecting and fixing bugs. Security, which is often overlooked in hardware development in favor of reliability and dependability, must no longer be neglected, given the surge in hardware attacks [11].

Given this scenario, our focus is primarily on enhancing HT detection by incorporating defensive techniques at different stages of chip design, **both** in the front-end and back-end. This approach is rarely observed in prior works. In this paper, first, we show how the knowledge generated by verification engineers, specifically assertions, can be leveraged as valuable security assets during the front-end stage. These data are usually utilized to prove that a design is bug-free, and not used again. Repurposing this knowledge for security purposes has the potential to yield substantial time and effort savings.

However, relying solely on the reuse of verification data

Manuscript received 19 February 2024; revised 8 August 2024; accepted 14 August 2024. This work was supported in part by the EU through the European Social Fund in the context of the project "ICT programme". The work of Mohammad Eslami was supported by the HARNO under Grant 11.4-1/23/1. Recommended for acceptance by X. Jia. *(Corresponding author: Mohammad Eslami.)*

Mohammad Eslami and Tara Ghasempouri are with the Department of Computer Systems, Tallinn University of Technology (TalTech), 12618 Tallinn, Estonia (e-mail: mohammad.eslami@taltech.ee; tara.ghasempouri@taltech.ee).

Samuel Pagliarini is with the Department of Computer Systems, Tallinn University of Technology (TalTech), 12618 Tallinn, Estonia, and also with the ECE Department, Carnegie Mellon University, 15213, Pittsburgh, PA USA (e-mail: samuel.pagliarini@taltech.ee).

Digital Object Identifier 10.1109/TC.2024.3449082

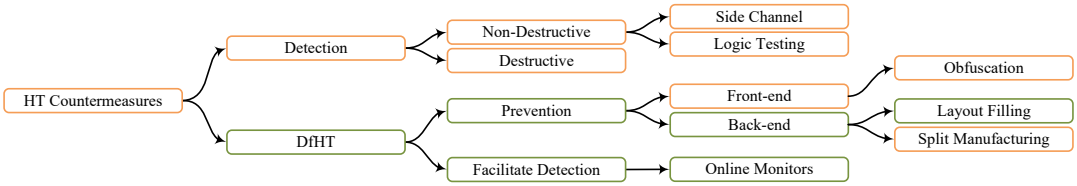


Fig. 1. An overview of the HT protection methods. The techniques used in this work are colored in green.

may not suffice to attain high levels of security. Therefore, we introduce a complementary technique aimed at enhancing the security of digital designs, which is the incorporation of online checkers during the back-end stage of physical synthesis. The insertion of the online monitors serves a dual purpose: not only do they aid in HT detection, but they also act as functional layout fillers. By utilizing the free resources within the layout, such as gaps and routing resources, these monitors contribute to restricting the available resources that an adversary might exploit for inserting HTs. Nonetheless, adding an online checker is a form of redundancy that may pose significant design overheads (in terms of timing, area, power, and performance). Yet, by proposing a smart timing- and area-aware technique, and performing the checker insertion by leveraging the Engineering Change Order (ECO) features of commercial CAD tools, we considerably minimize the overheads in our flow.

This smart technique seeks to make a balance between heightened security measures and the optimization of resource utilization, thereby enhancing the overall robustness of digital designs. Our overall methodology for reusing the verification data for security purposes, as the first contribution of this work, is described in our prior work [12]. In this paper, we present an in-depth description of how we make use of back-end stage techniques to augment our prior research on IC security.

The remaining sections of the paper are organized as follows: Section II provides background information on various techniques against hardware attacks during fabrication time and reviews related works. Section III explores the primary motivation behind this work, focusing on transforming verification assertions into security checkers. In Section IV, technical details regarding the implementation flow of adding online checkers during physical synthesis are explained. Experimental results are presented in Section V. The paper concludes with Section VI.

II. Background

Due to the rising concerns regarding hardware attacks, design houses try to protect their chips, particularly during the post-design stages where the environment is not trusted, and there is limited oversight in the chip fabrication process. To address this particular oversight, programs such as the Microelectronics Quantifiable Assurance (MQA) have started to add traceability to devices and systems. In tandem, security concerns have also brought the concept of a "Zero Trust" model, in which all parties, tools, and assets thereof are considered potentially untrusted. These concepts co-exist with techniques to improve trust; An overview of the different protection techniques is shown in Fig. 1. These techniques are based on the concept of either detecting HTs on the fabricated chip or preventing the insertion of HTs through Design for Hardware Trust (DfHT) [13] approaches.

A. Detection Techniques

Once an adversary introduces an HT at fabrication time, detection becomes exceptionally challenging [14]. Effective detection techniques must be used after fabrication to counter this threat. These techniques are mainly based on inspecting the fabricated chip to ensure that the chip is HT-free. Detection methods are performed either in a destructive or non-destructive fashion. In destructive methods, the chip is depackaged and each layer is separately inspected using highly advanced methods to check if any logic is added (or removed) by an adversary [15].

However, this approach has significant drawbacks. Firstly, it involves significant time and cost. Secondly, the analyzed chip is rendered entirely non-operational after inspection, leading to its destruction. Consequently, this method is limited to analyzing random samples and is not viable for inspecting entire lots of chips.

Therefore, non-destructive methods were introduced to analyze the fabricated chips before being distributed in the market [16]–[22]. These methods are mainly classified into side-channel analysis and logic testing.

Side-channel analysis techniques concentrate on observing physical attributes such as power consumption, path delay, or electromagnetic emanations [16]–[19]. These attributes are then compared to those of *golden chips*, which are assumed to be HT-free.

Logic testing involves applying test stimuli to evaluate chips, comparing the responses with expected ones precomputed through simulation [20]–[22]. Detection relies on detecting changes in chip functionality during testing, and aims to activate potential HTs within a limited test time. To avoid detection, attackers create static HTs activated by extremely rare conditions.

B. DfHT Techniques

In DfHT approaches, the concept is to embed additional protection logic in order to either aid in detecting the HTs

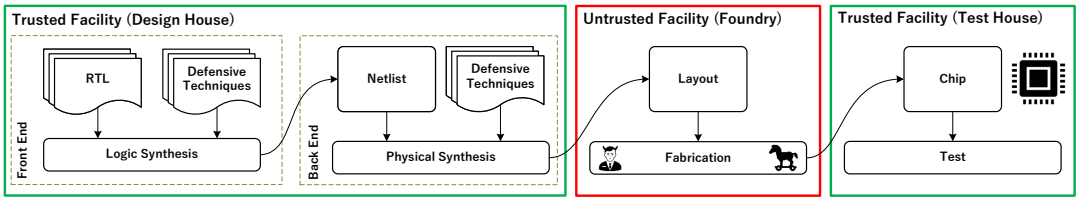


Fig. 2. Different stages of IC design: The design house and the test house are considered trusted, while the foundry is assumed to be untrusted.

[23]–[25] or to prevent an adversary from inserting an HT altogether [26]–[32]. Although complete prevention against HT insertion remains impossible in practice, efforts have focused on strategically limiting available chip resources, making it exceedingly difficult for adversaries to exploit them for malicious logic insertion [29]–[32].

When referring to the process of designing an IC, the tasks are typically divided into two stages: front-end and back-end [33]. The front-end stage focuses on the initial specification and creation of the chip's architecture. This involves tasks such as specifying the functionality, creating a high-level design, and simulating the behavior of the design. Hardware Description Languages (HDLs) like Verilog or VHDL are commonly used in this stage. Once the front-end design is complete, the process transitions to the back-end. In this stage, the focus is on transforming the design into a physical layout that can be manufactured, and it includes tasks like clock tree synthesis, place and route, timing closure, and physical verification.

Due to the distinct characteristics of the front-end and back-end stages, the approaches employed for HT prevention vary between the two stages. Front-end engineers may deploy different obfuscation techniques, such as logic locking [26], [27], to protect the Intelectual Property (IP) of the design. In these methods, the design is secured with additional keys, initiating normal operation only when the correct sequence of keys is applied. While these techniques were not developed for HT prevention, it is understood that obfuscating the design against IP theft perhaps makes the design less evident for an adversary attempting to insert an HT.

Conversely, prevention methods applied in the back-end stage focus on protecting the chip layout from potential attackers within the foundry. One approach is layout filling which is aimed at restricting available resources, such as gaps and free routing tracks, to prevent adversaries from inserting malicious logic [29]–[31]. Another strategy, known as split manufacturing, involves fabricating one part of the chip in one foundry and the remainder in a second foundry [34]. Despite its promise, practical challenges arise, including finding two companies with compatible manufacturing technology and managing complex handshakes between the foundries.

In the domain of DfHT, another research track focuses on developing methods to enhance HT detection [35], [36]. For this purpose, different types of checkers can be integrated into the design to sense irregularities and raise awareness in case of incorrect behavior. These checkers contribute to side-channel analysis by serving as thermal sensors to magnify thermal activity, or act as security checkers by introducing redundancy to the design. This approach aligns with strategies proposed for enhancing reliability against faults. In some works, such as [14], alternative methods like functional testing and side-channel analysis are also considered for facilitating HT detection. However, in our classification, we view side-channel analysis and functional testing as primary methods for HT detection rather than simply facilitating detection.

C. Threat Model

The chip production process involves several stages, as shown in Fig. 2. Front-end engineers transform the high-level design description into a gate-level netlist through logic synthesis. This netlist is then handed to the back-end team, where engineers adjust it based on specific constraints such as area, power, and timing. Defensive techniques can be incorporated into the design by either the front-end or back-end team to protect against potential threats. The resulting layout is sent to a foundry for fabrication. After the chip is received, specific tests are performed on it to verify its functionality.

In our threat model, we assume the foundry to be an untrusted facility, where a potential adversary (e.g., a rogue engineer) may be present. This encompasses fabrication-time attacks, similar to those explored in prior research, where malicious modifications to the design are introduced during the IC manufacturing [37]. Conceptually, a malicious foundry can incorporate three types of HTs into the layout of an IC: additive, substitution, and subtractive HTs. Additive HTs involve introducing additional circuit components and/or wiring into the existing design. Substitution HTs require removing logic that is replaced by HT circuit components and/or wiring. Subtractive HTs involve removing circuit components and/or wiring to alter the behavior of the existing circuit design. In this work, we *only* focus on additive HT attacks due to their significant impact on system behavior, their detectability through changes in various design characteristics, and the extensive body of research on this HT type. Design and test stages, including the design house and test house, are assumed to be trusted. Front-end and back-end teams implement defensive techniques in the design before sending it for fabrication, aiming to counter fabrication-time attacks.

Furthermore, we assume the attacker within the foundry has the capability to insert sophisticated, small, and rarely activated HTs that can evade side-channel analysis, logic testing, and simple forms of chip inspection. The attacker,

$$\forall\ n \in \{nodes_{(Des)}\},\ n = \begin{cases} C & \text{if } functional\ path\ exists\ between\ (n)\ and\ (assertion) \\ V & \text{if } functional\ path\ does\ not\ exist\ between\ (n)\ and\ (assertion) \end{cases} \quad (1)$$

due to his/her location, has access to the target technology's Process Design Kit (PDK) and advanced commercial CAD tools. Specifically, we focus on functional HTs that alter the chip's functionality, allowing their effects to be observed by comparing internal signals with the expected ones.

D. Related Works

1) Logic Locking: As mentioned before, logic locking is a key-based technique used to protect the intellectual property of ICs by obfuscating their functionality. However, this technique has limitations [38] and has been subject to numerous attacks that can discover the key value. Attacks such as boolean Satisfiability (SAT) attacks, removal attacks, and bypass attacks, have demonstrated that logic locking can be vulnerable, allowing attackers to uncover the key and compromise the security of the design. A logic locking technique that relies on a secret key to control circuit functionality is presented in [26]. The secret key becomes crucial for proper circuit operation, serving as an authentication mechanism. Another methodology is presented in [27] to protect gate-level IPs, providing obfuscation and authentication. This approach modifies the state-transition function and internal-circuit structure, allowing normal operation with a predefined enabling sequence or key. However, it may have limitations against structural analysis-based attacks as it does not explicitly modify the state space of the existing Finite State Machine (FSM) [27].

2) Layout Filling: In [29], the authors propose a method of populating unused spaces with functional cells, creating an independent combinational circuit for post-fabrication testing against cell modifications. Nevertheless, a challenge lies in achieving a high occupation ratio while keeping the design routable. To overcome this limitation, [31] gives priority to filling gaps that could potentially be exploited for HT insertion. However, this strategy may lead to alterations in the initial routing, posing a risk of violating critical paths due to rerouting.

Addressing the limitation of user control over placement, [39] proposes a placing refinement technique in which large unused layout spaces are segmented. Despite this refinement, there remains a vulnerability where attackers can reverse the refinement, creating optimal zones for their malicious logic. In a different approach presented in [40], a selective placement method is employed, where sensitive logic is positioned in denser layout areas, leading to places gaps around less sensitive regions.

Another attempt to mitigate layout vulnerabilities is discussed in [41], where the authors aim to reduce large gaps by shifting cells and utilizing the ECO features of commercial CAD tools. However, this technique often results in worsened timing, leading to negative timing slack in most cases.

3) Online Monitors: These techniques have been used widely for reliability and dependability, focusing on Concurrent Error Detection (CED) techniques, which introduce redundancy through parity codes or hardware duplication, complemented by a dedicated checker [42]. The approach in [36] focuses on combining special CED techniques and selective programmability to protect digital systems against HT attacks. However, this method imposes considerable area and power overheads on the design.

III. Reusing Verification Assertions as HT Detector

The foundation of this work originates from the utilization of assertions as HT detectors, as discussed in [12]. A hardware assertion is a statement or condition commonly specified in HDLs that defines a certain expected behavior or property of a digital circuit. This concept gains significance, particularly when extensive time and effort are invested in the verification of designs to ensure their integrity and the absence of bugs. The generated verification assets (i.e., assertions), which are often no longer utilized after the verification process, can be repurposed for security applications, specifically for the detection of HTs. The main challenge lies in the absence of a technique to synthesize the assertions as checkers, as well as a metric that identifies suitable assertions for use as HT detectors. To address this gap, we introduced a novel metric termed Security Coverage (SC) that evaluates the security effectiveness of verification assertions [12].

A. Security Coverage

As the name indicates, SC focuses on assessing the security qualifications of existing assertions. It is defined as the ratio of nodes covered by the assertion to the total nodes within the design. Each node (n) in the design is defined as either covered (C) or vulnerable (V), as indicated in Eq. 1.

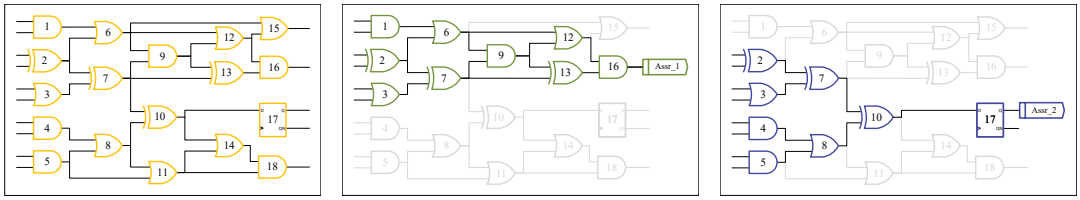
Therefore, the SC of a design is calculated as follows:

$$SC_{(Des)} = \frac{|\bigcup_{i=1}^{k} C_i|}{|\bigcup_{i=1}^{k} C_i| + |\bigcup_{i=1}^{k} V_i|} \quad (2)$$

Where k denotes the total number of the assertions, and C_i and V_i are the covered and vulnerable nodes, respectively.

An illustrative example representing a design with two integrated assertions is presented in Fig. 3. To calculate the SC for the entire design, Eq. 2 is applied. Furthermore, this equation can be employed to determine the SC for each individual assertion, enabling a comparative analysis of their security properties. Considering the two assertions in this scenario, the SC for each assertion is computed by finding the number of covered nodes (C). Examining Fig. 3(b), nodes 1, 2, 3, 6, 7, 9, 12, 13, and 16 (highlighted in green) have at least one functional path[1] to Assr_1. Similarly, the covered

[1]A functional path refers to a path within a circuit that can be traversed using a combination of valid inputs. This contrasts with non-functional paths that are unreachable with a given set of inputs.



(a) (b) (c)

Fig. 3. An example of a) original design, b) nodes covered by bound assertion Assr_1, and c) nodes covered by bound assertion Assr_2

elements for Assr_2 (highlighted in blue) encompass nodes 2, 3, 4, 5, 7, 8, 10, and 17, as depicted in Fig. 3(c). It is worth noting that certain nodes may appear in multiple subsets of covered nodes for different assertions, exemplified by nodes 2, 3, and 7, while some other nodes may not be covered by any assertions (nodes 11, 14, 15, and 18).

Nevertheless, several challenges restrict the calculation and use of SC for assertions in a design:

1) **Synthesis Limitations:** The majority of verification assertions are written to identify irregularities during simulation, and they cannot be synthesized. Consequently, they cannot be bound to the design.
2) **Functional Path Requirement:** To consider a node as covered, there must be a functional path between the node and the output of the assertion. Hence, the existence of a connection between the node and the output of the assertion is not sufficient, and traditional methods, such as extracting the input cone(s), cannot be used.
3) **Overheads:** It is crucial to acknowledge that individual assertions introduce specific overheads to the design. It is also the case that combined assertions introduce overheads that are not the sum of their individual overheads. In evaluating the effectiveness of each assertion, it is necessary to consider these factors to allow for a comprehensive evaluation of the trade-off space between security and the associated design overheads.

The synthesis limitations problem is the main challenge since it affects other problems. The most widely used languages for describing assertions are PSL and SystemVerilog, but the use of standardized languages does not imply that they are synthesizable – some assertions are clearly only meant for simulation purposes and have to be filtered out. To address this, we utilize the MBAC tool [43] to convert PSL and/or SystemVerilog assertions into a synthesizable Verilog format.

Once the assertions are transformed into synthesizable code, they are integrated with the main circuit for an assessment of their effectiveness and the overhead they introduce. To address the second challenge and obtain the SC for assertions, we use the Cadence JasperGold Security Path Verification (SPV) tool [44]. This tool allows us to perform a proof analysis for verifying the existence of functional paths between desired nodes in the design, also known as taint analysis. It should be noted that JasperGold could have been replaced by any other tool capable of doing taint analysis, academic or commercial. In our scenario, the considered origin nodes are all nodes in the design; the destination node is the assertion output. The existence of a functional path between any node in the design (origin) and the assertion output (destination) means that the node is covered by the assertion (see Fig. 3).

In other words, if a covered node's value is maliciously manipulated (e.g., due to an inserted HT activating its payload), the assertion can detect it, functioning as an online monitor. This concept shares similarities with verification schemes, where assertions aim to identify irregularities in the design. However, the key difference is that verification assertions are only necessary until the logical synthesis step. Verification engineers mainly seek to correct the potential mistakes made by the design team, and they can achieve their objectives through simulation, ensuring the functionality of the design. Once they confirm that the design is bug-free, assertions are no longer needed since the design's functionality and specification remain constant in subsequent chip design steps. In contrast, security engineers must consider potential threats during fabrication, and assertions that have transformed into online monitors must remain with the design until the chip is fabricated.

The third challenge involves quantifying the overheads introduced by each assertion on the design. While simulation offers insights into the circuit's incorrect behavior and internal values, it falls short in determining crucial performance characteristics like power, timing, and area. To address this limitation, the design undergoes multiple synthesis runs, and precise performance reports are generated. Initially, the original circuit is synthesized without the assertions, providing maximum clock frequency, power, and area reports. Subsequently, the circuit, now integrated with the bound assertions functioning as embedded security checkers, undergoes synthesis again. The evaluation is based on a comparative analysis of results from the two synthesis processes, ensuring a comprehensive assessment of assertion performance in terms of design overhead.

Once the SC and the overheads of each assertion are known, we can decide if the assertion is worth to be kept or not. However, this cannot be performed manually since investigating the trade-off between SC and overheads for each assertion is a time-consuming task. Hence, an automation flow is needed to select the assertions. The prerequisite of automation flow is defining a strategy to only pick efficient assertions in terms of security and overheads. This efficiency can be defined such that the assertion has more security properties, imposes less

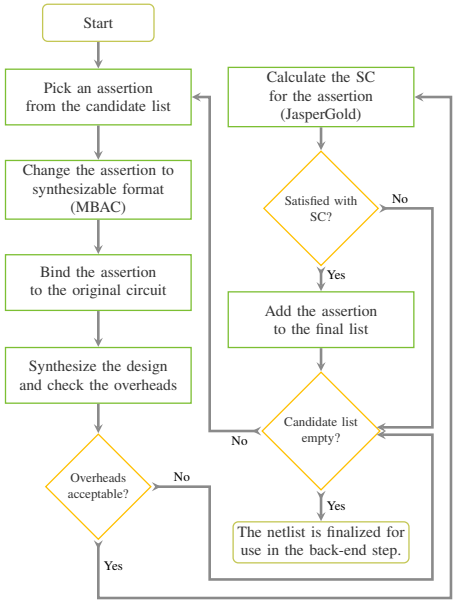


Fig. 4. Optimization flow for selecting the assertions to be used as security checkers

overheads on the design, or a balance between both.

Fig. 4 illustrates a proposed automated flow for the assertion selection process. The initial step involves selecting an assertion from the candidate list, which contains assertions with synthesizable potential, excluding those with a simulation-based nature that have already been filtered out. The chosen assertion is then converted into synthesizable logic and integrated into the design. This prepared design enables overhead evaluation, where synthesis is performed with the assertion to compare various metrics (e.g., power, performance, and area (PPA)) against the values obtained from the original design.

After assessing the overheads and confirming their compatibility with user-defined requirements, the subsequent step involves calculating the SC. This is achieved by employing Eq. 2 and utilizing the JasperGold SPV tool. If the assertion passes both the overhead evaluation and achieves the desired SC, it is included in the final list of assertions. Otherwise (e.g., if the overheads are regarded as unacceptable or the SC falls short of the user's expectations), an alternative assertion from the candidate list is considered (if any). It should be noted that the decision in this step can be based on either the SC of the assertion individually or the overall SC threshold, which is specified by the user for the whole design. Once the flow is completed, the generated netlist with the embedded assertion(s) is considered finalized. This finalized netlist will be used in the back-end stage for further security improvements through the insertion of online monitors.

B. Evaluation of the Security Properties of Verification Assertions

In a practical case study assessing the security properties of verification assertions, we conducted an SC calculation for predefined assertions in various IPs of the OpenTitan System on Chip (SoC). Given that OpenTitan, an open-source silicon root of trust project, is developed and maintained by a community of experienced engineers, it is recognized as a highly reliable and suitable case study for evaluating the effectiveness of our proposed framework. The initial experiment focused on selecting four distinct assertions present in 35 different Register Top modules of OpenTitan, resulting in a set of 140 assertions. This initial analysis indicates a selection based on commonality across different IPs, disregarding specific characteristics or overheads.

The SC of each assertion in different IPs are depicted in Fig 5. As shown in this figure, the calculated SC percentages for each assertion are less than 5%, which may raise concerns, even when overlooking the associated overheads. However, it is crucial to recognize that a lower SC does not necessarily imply inadequate security properties, as some applications prioritize safeguarding specific security-sensitive components rather than the entire design.

The SC of an assertion is influenced by various factors, with a crucial one being the nature of how the assertion is formulated for verification. Specifically, if an assertion is defined to concentrate only on local signals, such as checking whether specific bits of a register are 0 or 1, it is generally considered less impactful. On the contrary, assertions that are not restricted to narrow scopes and instead describe high-level behaviors are preferable. Consequently, for users prioritizing a higher SC for the entire design over assertions safeguarding specific design regions, the emphasis should be on selecting assertions covering larger design parts. The initial experiment did not prioritize assertions based on high-level descriptions but focused on finding synthesizable assertions present in multiple IPs. However, as depicted in Fig. 5, the SC results are generally unremarkable. This underscores the need for further analysis when choosing assertions, rather than blind selection.

In the subsequent experiment, we investigate the SC of assertions uniquely generated for deployment in specific IPs, causing them to be unsuitable for use in other IPs. This experiment demands more effort, as synthesizable assertions undergo a preliminary analysis before binding into the design. Consequently, not only unsynthesizable assertions are excluded from consideration, but also assertions focused on checking local signals are filtered out.

Figure 6 illustrates the calculated SC percentages for individual assertions within three chosen IPs. As shown in this figure, the SC for each assertion is notably higher compared to the figures obtained from the initial experiment (Fig. 5). The average SC stands at 85.83% for the selected assertions in the ALERT_HANDLER (Fig. 6(a)), 46.03% for the selected assertions in the ALERT_HANDLER_ESC (Figure 6(b)), and 38.35% for the selected assertions in the FLASH_PHY_RD module (Figure 6(c)).

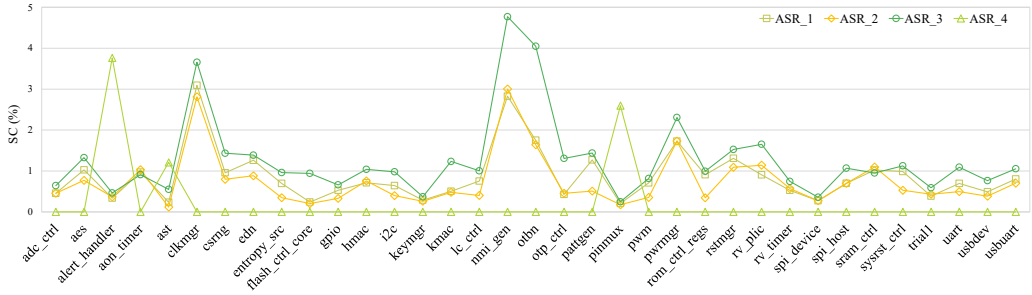
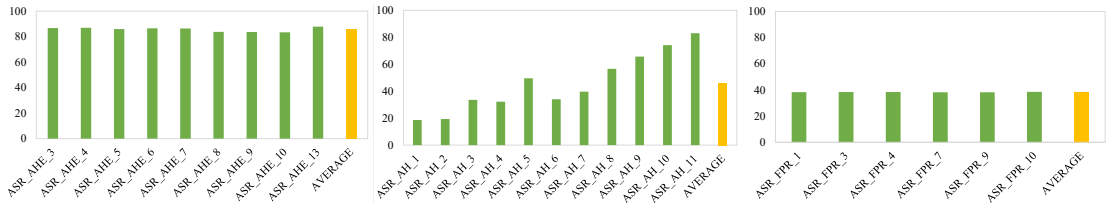


Fig. 5. SC percentage for different IPs of OpenTitan



(a) (b) (c)

Fig. 6. Calculated SC percentage for the different assertions in selected IPs of OpenTitan: a) alert_handler_esc, b) alert_handler, and c) flash_phy_rd

C. Challenges

While achieving higher SC numbers for different assertions (e.g., in the ALERT_HANDLER_ESC IP as shown in Fig. 6(a)) may suggest improved design security, it is crucial to acknowledge that ensuring design security goes beyond relying solely on SC metrics, even with high percentages (e.g., exceeding 80%). There are notable challenges associated with SC, and some key ones include:

1) Ineffectiveness of verification assertions at runtime: While functional assertions can cover various security properties by ensuring the expected behavior of the circuit, they may lack the precision to thoroughly cover the negative or unexpected behavior of a circuit under attack. Although the SPV tool is employed to calculate taint propagation coverage, the effectiveness of detecting HT behavior by the synthesized assertions at runtime is not guaranteed.
2) Scalability concerns: The necessity to "bind the assertion" and synthesize the entire design for characterizing overheads may pose scalability challenges for larger designs. The resource-intensive nature of this process could limit its practicality for more extensive and complex circuits.
3) Assertion availability: The methodology aims to reuse existing assertions rather than generating new ones. However, a potential challenge arises if no suitable assertion with acceptable SC is found for a given design. This situation is exemplified by the SC numbers obtained for different assertions, as depicted in Fig. 5.

To overcome these challenges, an additional layer of security must be incorporated into the design to address the limitations associated with reusing the verification assertions as online monitors.

IV. Enhancing the Security by Adding Online Monitors during the Physical Synthesis

In this section, we introduce a novel methodology to incorporate online monitors into the layout during the physical synthesis flow. While prior works have explored the idea of integrating online monitors, most efforts have concentrated on introducing these checkers during the front-end stage of design. In our work, we take a different approach by directly incorporating the checkers into the layout while considering front-end inserted assertions.

Figure 7 provides a simplified view of the layout of a block of an IC. The green polygons represent standard cells that are later interconnected through various metal layers, establishing the logical function of the design. Due to fabrication complexities, particularly in modern process nodes, achieving 100% density where the layout is entirely filled with standard cells, is impractical. Thus, gaps are present in the layout, highlighted in red in Fig. 7. These gaps can be later used by an adversary for inserting his/her malicious logic (i.e., HTs) [29].

Despite these gaps being filled with filler cells before being sent to fabrication, since these filler cells lack functionality and are not connected to the original design, they can be easily removed by the attacker inside the foundry. Our methodology leverages these gaps and available resources to insert online checkers into the design. By doing this, not only do we add an extra security layer to the design, but also we limit the adversary to put malicious logic by increasing the density and congestion in the layout.

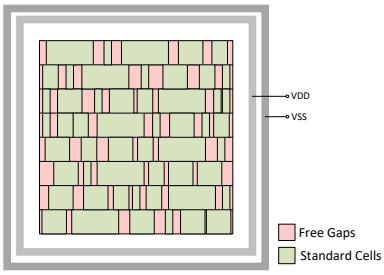


Fig. 7. Illustrative example of a block layout within a chip

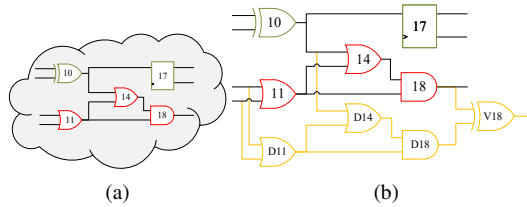


Fig. 8. An example of a) design before adding online monitors, and b) design with the protection logic (D11, D14, and D18 as the duplicates and V18 as the voter) to protect the uncovered gates (11, 14, and 18)

Although this methodology can also be used independently with minor modifications in the flow, we use it as a complementary methodology along with reusing the assertion for detecting HTs to cover its shortcomings. It should be noted that we leverage the ECO features of the CAD tools for inserting the online monitors into the layout[2]. This use of ECO minimizes alterations to the original layout with each added online monitor, ensuring optimal overheads in comparison to front-end approaches.

A. Online Monitors

As mentioned earlier, the online monitors act as an extra protection layer for the nodes not covered by assertions. To create this protection strategy, we use a Dual Modular Redundancy (DMR) scheme, as shown in Fig. 8. The left image (Fig. 8(a)) illustrates a subpart of a design represented in Fig. 3, in which the covered nodes (10 and 17) are highlighted in green, whereas the vulnerable/uncovered nodes (11, 14, and 18) are highlighted in red. To construct an online monitor for this subset of the design, first, we duplicate the uncovered gates with exactly the same equivalent gates from the library (i.e., the same gate type, and same drive strength). Then, we compare the output of the duplicated part with the output of the original part by XORing these two signals, as depicted in Fig. 8(b). Therefore, adding an online monitor for this part adds seven new covered nodes (11, D11, 14, D14, 18, D18, and V18) to the previously covered nodes (10 and 17).

It is important to highlight that an adversary may attempt to substitute the online monitors by a HT. However, the online monitors are always on and therefore contribute to the power profile of the circuit. An HT, on the other hand, should have a stealthy trigger, which is not compatible with an always-on functionality. An HT that promotes a noticeable change in the chip's power profile is likely detectable.

A greater number of connected uncovered gates is desirable, as all of these connected gates require only one XOR gate as a voter. However, limitations inherent in the physical synthesis flow may pose challenges to implementing DMR for all uncovered gates, even if they form a cone, as illustrated in Fig. 8. Further details on this aspect are elaborated in the subsequent section.

[2]ECO features in CAD tools allow engineers to make last-minute modifications to the existing layout, such as adding or removing components, changing connections, etc. These changes may be necessary due to factors such as updated specifications, errors discovered in the initial design, or other requirements that emerge during the design phase.

B. Embedding Online Monitors into the Layout

The comprehensive flow for integrating online monitors during the physical synthesis flow is outlined in Fig. 9. It involves four major steps that are detailed as follows:

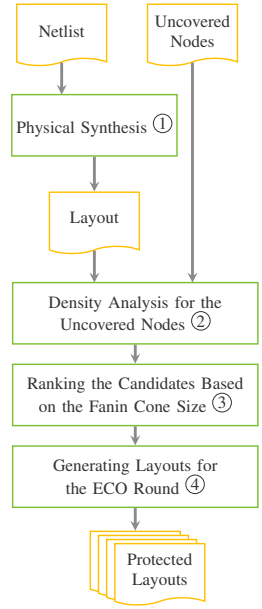


Fig. 9. An overall flow of integrating online monitors during the physical synthesis

1) Physical Synthesis: The first step starts with turning the netlist into a layout using a physical synthesis tool. Here, the netlist includes the main design and all selected assertions bound to it. This step involves several stages such as placement, clock tree synthesis, and routing. Through physical synthesis, vital information such as the precise placement of standard cells, the physical distribution of the clock, and the structure of interconnections in terms of wire length or the utilization of each available metal layer is obtained. Essentially, physical synthesis provides insight into the spatial

configuration of the design. It should be noted that this time-consuming step is performed only once for each design.

2) Density Analysis for the Uncovered Nodes: The resulting layout, along with the report of uncovered nodes from the SPV tool, becomes the input to our developed analytical tool. The objective here is to identify the uncovered nodes with available gaps around them, suitable for housing the online checker tasked with safeguarding the respective node. If there is no space available around the uncovered node, the online checker might be placed at a far distance from it, leading to higher resource utilization and degradation of the PPA parameters of the design. This is a crucial step in our flow to prevent this from happening in order to minimize layout modifications. In other words, the density analysis provides an adequate basis for embedding the online monitors for future ECO rounds. This makes the presented methodology different from front-end protection schemes where the actual overheads are often ignored. It should be noted that the scope of the searching area around each node is adjustable and can be changed according to the design size and density. A larger search area increases the processing time because more potential locations must be evaluated, and it can also lead to increased use of routing resources, which can negatively impact the overall performance of the design due to longer interconnect delays and higher power consumption. Therefore, the balance between the size of the search area and the associated overhead must be carefully managed to optimize both the detection capabilities and the design performance.

3) Ranking the Candidates Based on the Fanin Cone Size: The generated candidates from the density analysis tool undergo a ranking process. As emphasized earlier, our preference is to place the online checker for a group of interconnected uncovered nodes rather than individual gates. This choice offers advantages in terms of area, power, and routing resources. Therefore, our ranking system assigns higher priority to subsets of candidate gates with larger input cones, optimizing the overall efficiency of the protection scheme. It should be noted that in our cone analysis, we only consider the candidates with a cone size of 2 or greater to improve the efficiency of our methodology.

4) Generating Layouts for the ECO Round: The final step involves the generation of protected layouts. To facilitate this, we developed a tool that takes the ranked list of candidates (generated in the previous step) and integrates the online checkers into the layout incrementally. More specifically, one online monitor is added to the design at a time, and a new layout containing the added monitor is generated. This process is repeated from the top of the ranked list to the end. Therefore, if there are n candidates (the nodes suitable to be protected by online checkers) in the ranked list, we create n different layout files in an iterative fashion. Each protected layout file, such as *Layout1*, contains the online monitor from *Candidate1*; *Layout2* contains the online monitors for *Candidate1* and *Candidate2*, and so forth. It should be noted that these protected layouts are generated to be used along with the ECO flow, and the finalized layout, which includes all potential online checkers and is intended for fabrication, is derived after the conclusion of the ECO round.

Since the difference between each protected layout and its predecessor is the addition of only one online monitor, the user can decide whether to keep or discard the introduced online monitor. This incremental approach ensures a controlled integration of online monitors into the design while precisely managing the resources utilized for adding each online monitor. Moreover, it allows for a fine-grained assessment of the impact on the PPA parameters of the design.

C. ECO Flow

As described, the insertion of online checkers and the generation of protected layouts are efficiently conducted in our methodology, prioritizing area and resource utilization. We enhance this approach by introducing a timing-aware element to the ECO flow. We introduce a metric called the Degrading Factor (DF), set at 25% of the total positive slack of the design before the addition of online checkers. This DF serves as a deterministic parameter for deciding whether to keep or discard protected layouts due to the impact on the timing.

For this purpose, different PPA numbers of the initial layout are stored before incorporating online monitors. The ECO flow starts by selecting the first protected layout, including the highest-ranked online monitor from the cone analysis, and calculates the PPA numbers for this modified layout. Subsequently, the total slack number is compared with the previous layout (the one excluding the newly added monitor). If the slack number is negative or the difference between the new slack and the previous one exceeds the DF, the ECO flow discards the newly added layout since it degrades the timing beyond user constraints and proceeds to the next one. This process continues until all online monitors are integrated, or there is no more slack available for the new logic. Additionally, different checks are performed at each ECO round to ensure compliance with various design rules and avoid issues arising from the application of the new layout.

V. Experimental Results

This section presents the experimental results of integrating online monitors for five different IPs of OpenTitan. Two IPs were chosen for security reasons: ALERT_HANDLER_ESC_TIMER has the highest SC, and KEYMGR_REG_TOP has the lowest SC. Other three IPs were chosen based on their sizes: AST_REG_TOP is the largest design, FLASH_CTRL_CORE_REG_TOP has an average size, while the smallest one is NMI_GEN_REG_TOP. It should be noted that all these IPs are selected among the ones that already have some assertions, to maintain the concept of repurposing existing assertions. We do not introduce new assertions or modify existing ones across different IPs. The calculated SC for all these IPs is previously illustrated in Fig. 5 and Fig. 6.

We use the Cadence suite for all results herein reported: logical synthesis is performed by Genus, while the physical synthesis is performed by Innovus. The formal tool for performing the taint analysis is JasperGold SPV, as mentioned earlier. Our target technology node is a commercial 65nm CMOS one.

TABLE I
THE IMPACT OF ADDING ONLINE MONITORS ON THE SECURITY OF SELECTED IPS

IP Name	Instances	SC Before	SC Total	SC Added	# Nodes Covered by Monitors	# Applied Monitors	# Ignored Monitors	# Total Monitors	Preventing Factor
ALERT_HANDLER_ESC_TIMER	1404	87.82%	88.25%	0.43%	6	1	0	1	Density
AST_REG_TOP	7048	2.49%	19.94%	17.45%	1382	183	7	190	Density
FLASH_CTRL_CORE_REG_TOP	7048	1.38%	17.98%	16.6%	954	105	264	369	Timing
KEYMGR_REG_TOP	4611	0.91%	10.89%	9.98%	490	55	86	141	Timing
NMI_GEN_REG_TOP	214	3.53%	37.11%	33.58%	92	14	0	14	Density

A. Impact of Adding Online Monitors on SC

To assess the impact of integrating online monitors on the security properties of each design, we employ the evaluation scheme outlined in Eq. 2. The total covered nodes (C) in the numerator of Eq. 2 now encompass both the nodes previously covered by the assertion and the newly covered nodes introduced by the online monitor. It should be noted that since the output of the assertion or the voter of the online monitors can only be changed by an abnormality in the circuit's expected behavior, the chance of a false positive is eliminated. Consequently, the results obtained from the embedded checkers and assertions are always considered as true positive.

Table I presents the results concerning the impact of added online monitors on the security of the considered designs. In this table, the first column denotes the IP name, while the second column enumerates the instances in each IP. The third column indicates the SC before the integration of online monitors. These values are obtained by binding all available assertions to each IP and analyzing the coverage using the SPV tool. The fourth and fifth columns represent the SC after adding the online monitors and the increase in SC specifically due to the online monitors, respectively. As indicated, the lowest increase in SC is 0.43% for the ALERT_HANDLER_ESC_TIMER IP. This is mainly because this IP already has a high number of covered nodes, and finding suitable candidates that pass all steps of the online monitor insertion flow (as shown in Fig. 9) is challenging. In other words, finding a set of uncovered connected nodes with sufficient space around them is more difficult since the total number of uncovered nodes is limited. However, the increased SC is not the only benefit we achieve. The introduced logic for inserting online monitors also fills the gaps in the layout and utilizes routing resources, which positively impacts security by reducing potential exploitation opportunities for attackers.

In column 6, the number of total covered nodes is presented after the addition of online monitors. These covered nodes include the nodes not covered by the assertions and the new redundant logic added to form the online checker. Columns 7 and 8 represent the number of applied online monitors in the IP and the number of ignored ones. Column 9 represents the number of online monitors that can be generated for each design after performing density and cone analysis (Fig. 9). The total number of online monitors is equal to the number of individual *protected layouts* generated for the ECO flow. It should be noted that not all generated online monitors can be integrated into the design due to timing restrictions.

The final column indicates the factor preventing the addition of more online monitors. If all available online monitors can be successfully integrated into the design, it means that there are no more online monitors that could be generated, mainly because of the high density around the uncovered nodes. In contrast, if some online monitors are left unembedded in the design (excluding those that exceed the DF), it is mainly because the timing resources of the design are exhausted, and they have to be ignored. More details about the PPA restrictions are discussed in the next part.

B. Impact of Adding Online Monitors on PPA

The initial layout (baseline) for each IP is configured such that each design has a density ranging between 60% to 65%. By doing this, we can preserve a positive setup slack of approximately 10% of the clock period for each design. Given that the online monitors introduce new logic to the design, influencing its timing, this 10% margin allows the utilization of positive slack for integrating online monitors.

Table II presents various PPA metrics before and after adding the online monitors to the selected IPs. The first column denotes the IP names. The next two columns illustrate the area and placement characteristics of the layouts, where the second column represents the total area for each design, and the third column represents the placement density. It should be noted that the total area reported in this table refers to the total **cell area**, and not the overall physical area of the block/chip. Among all designs, NMI_GEN_REG_TOP experienced the most significant increase in area parameters, as it is the smallest design, and even a few online monitors make the size of the added logic comparable to the overall design size.

The fourth column represents the total power consumed by each IP, with NMI_GEN_REG_TOP again showing the largest increase among the IPs due to its small size. The fifth and sixth columns present the timing characteristics of the design. While the hold slack remains relatively constant for all designs, the setup slack undergoes considerable changes for most of the designs, as the redundant logic might impact the timing of the design based on its location in different timing paths of the design. Consistent with Table I, the two designs with the preventing factor of timing (FLASH_CTRL_CORE_REG_TOP and KEYMGR_REG_TOP) exhibit the highest decrease in setup slack.

The last column represents the total metal wire length for each design. These metal wires are utilized to connect different parts of the design (signal routing), distribute power, or propagate the clock. As previously mentioned, the adversary

TABLE II
THE IMPACT OF ADDING ONLINE MONITORS ON THE PPA METRICS OF SELECTED IPS

IP Name	Total Area (μm^2)	Placement Density	Total Power (mW)	Setup Slack (ns)	Hold Slack (ns)	Total Wire Length (μm)
ALERT_HANDLER_ESC_TIMER (before)	3692.88	63.94%	1.46	0.328	0.133	22305.40
ALERT_HANDLER_ESC_TIMER (after)	3700.80	64.07%	1.47	0.328	0.133	22351.75
Difference	+0.21%	+0.20%	+0.68%	0.00%	0.00%	+0.21%
AST_REG_TOP (before)	28644.48	61.46%	9.93	0.287	0.084	353799.80
AST_REG_TOP (after)	30848.76	66.18%	10.26	0.091	0.07	397612.20
Difference	+7.69%	+7.68%	+3.32%	-68.29%	-16.67%	+12.38%
FLASH_CTRL_CORE_REG_TOP (before)	14243.40	64.25%	4.94	0.210	0.181	148407.90
FLASH_CTRL_CORE_REG_TOP (after)	15542.28	70.11%	5.13	0.007	0.186	170461.90
Difference	+9.12%	+9.12%	+3.85%	-96.67%	+2.76%	+14.86%
KEYMGR_REG_TOP (before)	18325.80	62.68%	8.11	0.278	0.152	186978.10
KEYMGR_REG_TOP (after)	19062.72	65.20%	8.29	0.000	0.151	199832.90
Difference	+4.02%	+4.02%	+2.22%	-100%	-0.66%	+6.87%
NMI_GEN_REG_TOP (before)	769.68	61.16%	0.23	0.118	0.178	4841.51
NMI_GEN_REG_TOP (after)	918.00	72.94%	0.27	0.043	0.179	6077.74
Difference	+19.27%	+19.26%	+17.39%	-63.56%	+0.56%	+25.53%

not only needs gaps to place malicious logic but also requires available routing resources to connect the malicious logic to other parts of the design. The increase in the total wire length of the design suggests that the design has become more congested, and the free routing resources are now more limited for use by the adversary.

To provide a more detailed insight into the impact of adding each online monitor to the design characteristics, we have extracted various PPA results at the end of each ECO round, where the new online monitor is successfully integrated into the design. This detailed breakdown allows us to observe individual impacts on different characteristics throughout the iterative ECO process. Fig. 10 illustrates the degradation in the setup slack after each successful ECO round. The vertical axis represents the total setup slack time in nanoseconds, ranging from the worst to the best slack time, while the horizontal axis depicts the progression of ECO rounds.

As previously mentioned, the ECO flow initiates with the layout containing the assertions, referred to as the "Baseline" in Fig. 10. As depicted in this figure, some rounds have a negligible impact on the timing, while others considerably degrade the total setup slack. There are also cases in which the slack is improved, attributed to the heuristics of the physical synthesis tool. However, this degradation does not exceed the DF, which is set at 25% of the total setup slack. It is worth noting that this parameter can be adjusted based on the user's preferences. For instance, if set to lower values, online monitors causing a sudden decrease in the total setup slack (e.g., the online monitor added in round 81 in Fig. 10(b)) will be discarded, resulting in a smoother overall trend for setup slack decrease.

Fig. 11 illustrates the progressive increase in wire length for each metal layer in the protected layouts. The vertical axis represents the wire length in μm, while the horizontal axis denotes the protected layout in each round. Similar to Fig. 10, the "Baseline" term refers to the layout containing assertions without online monitors. Although the number of available metal stacks varies in different target technologies, with new technologies typically offering ten or more metal layers, higher consumption of upper metal layers indicates increased congestion in the design. As a defender, our focus is on higher consumption of upper metal layers, reflecting the overall increase in congestion in the protected layouts. Excluding ALERT_HANDLER_ESC_TIMER, where only one online checker was available to be added, we observe a consistent trend of increased wire length in metal layers M3-M6 for all designs in Fig. 11, thereby reducing the available free routing resources for potential adversaries.

The visual representations of the layout, including placement configuration and the routed view of designs before and after the integration of online monitors, are illustrated in Fig. 12. In each row, the left image pair showcases the cell placement, while the right one displays the routed view. Specifically, the left image in each pair represents the layout before the addition of online monitors, while the right image corresponds to the final protected layout after the successful completion of all ECO rounds. Upon comparing the images on the right with those on the left, it is evident that the overall placement and routing configuration of the layouts remained unchanged, even for larger designs. This highlights the more efficient utilization of resources, which is a key advantage of our methodology in adding online monitors during the physical synthesis compared to similar works performed in the front-end step of chip design.

C. SCARF Versus Other Techniques

Table III provides a comparative analysis of our framework with other detection and DfHT approaches. The first column indicates the specific technique or category, while the second column references works in that category. The third column categorizes the technique into detection or DfHT (or both). The subsequent column details the chip design stage where the method is applied for protection. Column 5 specifies the location of the potential attacker, and column 6 indicates if the technique can also be employed to prevent HTs. The last two columns briefly outline the advantages and disadvantages of the techniques, respectively.

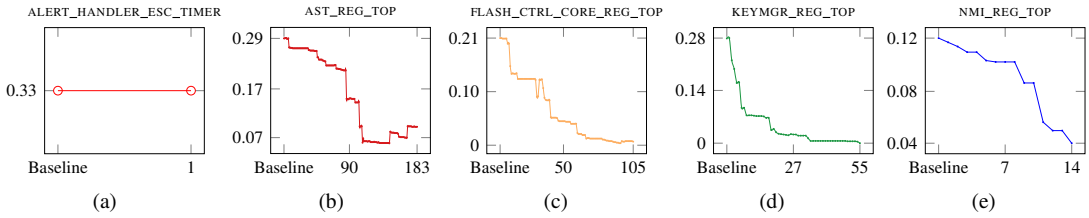


Fig. 10. Changes in setup slack after each round of adding the online monitors for different IPs

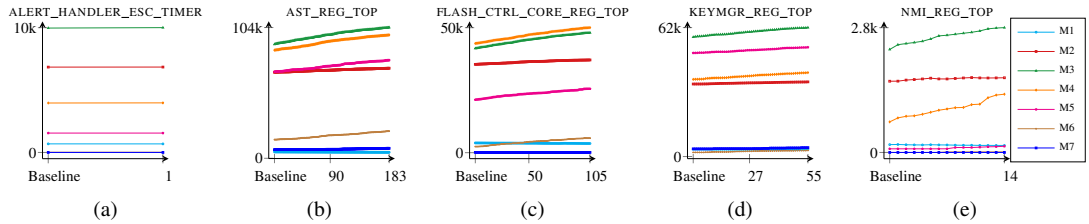


Fig. 11. The impact of adding online monitors on the length of different metal layers in the protected layouts for different IPs

TABLE III
COMPARISON OF DIFFERENT DETECTION AND DFHT TECHNIQUES

Technique	Refs	Detection or DfHT	Design Stage	Attacker Location	Prevents HT?	Pros.	Cons.
IC Fingerprinting, Delay Meas.	[16]–[19]	Detection	Test	Des. house, Foundry	No	Nearly zero overheads, Non-destructive method	Needs reverse engineering for attributes of the Golden chip, Confusion with environmental and process variation effects
Logic Testing	[21] [22]	Detection	Test	Des. house, Foundry	No	Very fast technique, Can be fully automatic	All input combinations are not covered to activate HTs, Ineffective when dealing with HTs with sequential triggers
BISA, Layout Filling	[29]–[31]	DfHT	Back-end	Foundry	Yes	Replacement of filler cells with the functional ones, Independent testing circuit from the original one	Design becomes unroutable in high occupation ratios, Favors layouts with large continuous gaps
Selective Placement	[39]–[41]	DfHT	Back-end	Foundry	Yes	Lowest overheads among HT prevention techniques	Risk of violating the critical paths in the routing stage, Timing degradation
TPAD	[36]	Both	Front-end, Back-end*	Des. house, Foundry	Yes	Zero false positives, Applicable to FPGA	Considerable overheads, Risk of degradation
SCARF	**This work**	Both	Front-end, Back-end	Foundry	Yes	Reusing verification assets, Low PPA overheads	Achieving high density for some designs is challenging

* This technique is mainly applied in the front-end stage. To prevent CAD tool attacks, the design is split into two parts, each processed by different CAD tools independently. The final layouts from these tools are later merged in the back-end stage for being set to the foundry for fabrication.

VI. CONCLUSION

In this paper, we presented a defensive framework aimed at enhancing the security of chips against fabrication-time attacks. The first part of the presented framework focused on the front-end stage, where we demonstrated the reuse of verification assertions by transforming them into security checkers. However, some security checkers may not be optimally effective, particularly those designed primarily for debugging purposes. To address this limitation, we proposed a novel methodology involving the insertion of online checkers during the back-end stage of the physical synthesis.

Although we designed the back-end methodology as a complementary approach to the front-end one, both techniques can be used independently based on user preferences and requirements. Our experimental results demonstrate that adding online monitors results in a modest increase of less than 20% in area, power, and placement density in the worst case. Simultaneously, it enhances the SC by 33.58% in the best case. Notably, our methodology utilizes only positive slack time during the ECO flow for adding online monitors, ensuring that it does not degrade the overall timing of the design.

While our framework has demonstrated promising results, there is still room for improvement. Future work could focus on embedding various optimizations to reduce the imposed

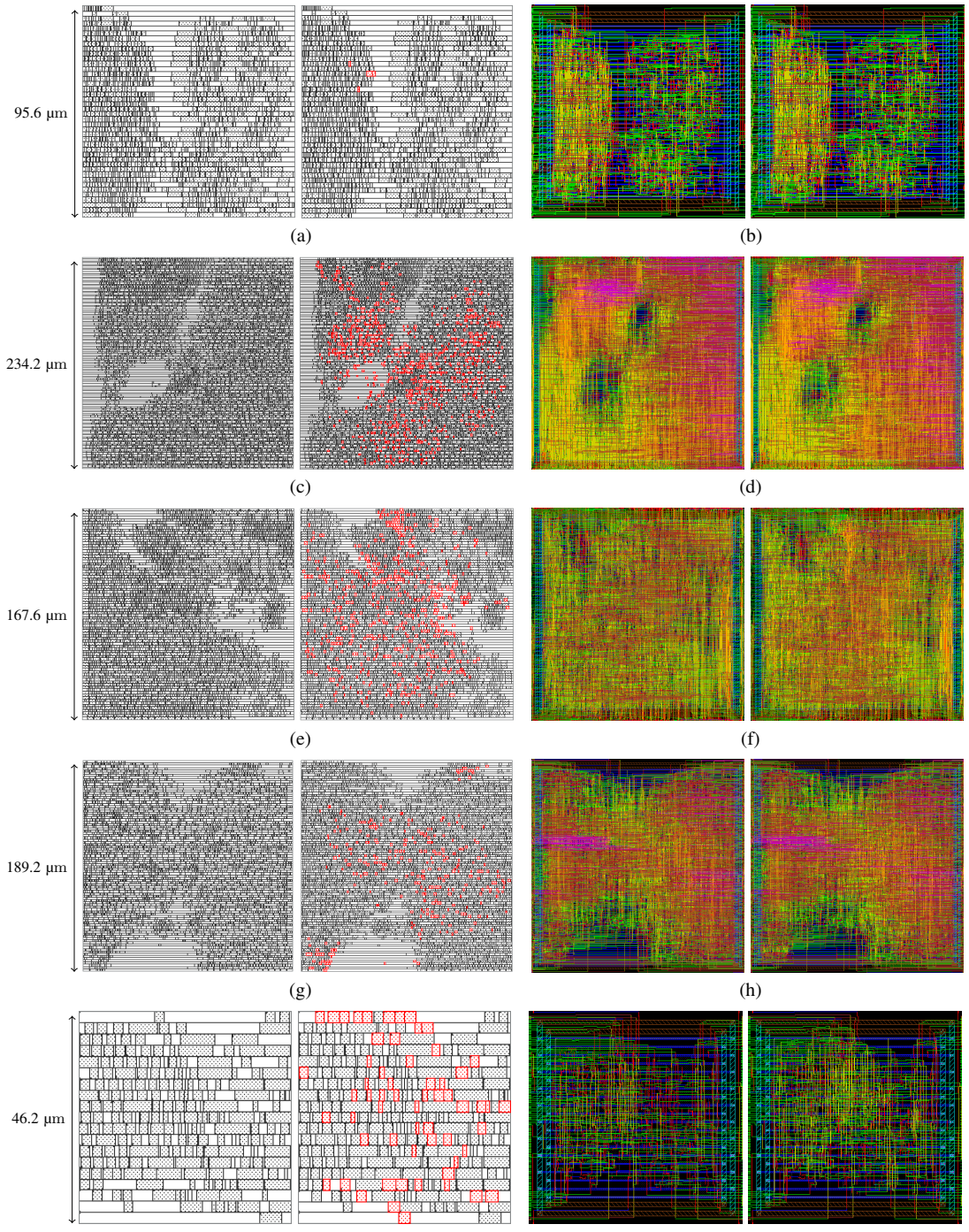


Fig. 12. The layout view of the IPs before and after adding online monitors, whereas the left pair of images in each row represents the placement configuration, while the right one represents the routed view of each design: a, b) ALERT_HANDLER_ESC_TIMER, c, d) AST_REG_TOP, e, f) FLASH_CTRL_CORE_REG_TOP, g, h) KEYMGR_REG_TOP, and i, j) NMI_REG_TOP

area and power, as well as incorporating path awareness into the flow to make better use of positive slack time.

REFERENCES

[1] C. Ting-Fang, "Tsmc to triple u.s. chip investment to $40bn to serve apple, others," 2022. [Online]. Available: https://asia.nikkei.com/Business/Tech/Semiconductors/TSMC-to-triple-U.S.-chip-investment-to-40bn-to-serve-Apple-others

[2] S. Bhunia et al., "Hardware trojan attacks: Threat analysis and countermeasures," *Proc. IEEE*, pp. 1229–1247, 2014.

[3] M. Tehranipoor et al., "A survey of hardware trojan taxonomy and detection," *Des. Test*, p. 10–25, 2010.

[4] C. Helfmeier et al., "Breaking and entering through the silicon," in *Proc. Comp. Comm. Sec.*, 2013, p. 733–744.

[5] S. E. Quadir et al., "A survey on chip to system reverse engineering," *J. Emerg. Technol. Comput. Syst.*, pp. 1–34, 2016.

[6] R. Karri et al., "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, pp. 39–46, 2010.

[7] W. Hu et al., "An overview of hardware security and trust: Threats, countermeasures, and design tools," *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, vol. 40, no. 6, pp. 1010–1038, 2021.

[8] M. Rostami et al., "A primer on hardware security: Models, methods, and metrics," *Proc. IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

[9] S. Bhunia et al., "Protection against hardware trojan attacks: Towards a comprehensive solution," *Des. Test*, pp. 6–17, 2013.

[10] V. Gohil et al., "Deterrent: Detecting trojans using reinforcement learning," in *Proc. Des. Autom. Conf.*, 2022, pp. 697–702.

[11] "Enisa threat landscape 2023." [Online]. Available: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023

[12] M. Eslami et al., "Reusing verification assertions as security checkers for hardware trojan detection," in *Proc. Int. Symp. Qual. Elec. Des.*, 2022, pp. 1–6.

[13] Y. Jin et al., *Design for Hardware Trust*. New York, NY: Springer New York, 2012, pp. 365–384.

[14] K. Xiao et al., "Hardware trojans: Lessons learned after one decade of research," *Trans. Des. Autom. Elec. Sys.*, pp. 1–23, 2016.

[15] C. Bao et al., "On reverse engineering-based hardware trojan detection," *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, pp. 49–57, 2016.

[16] D. Agrawal et al., "Trojan detection using ic fingerprinting," in *Proc. Symp. Sec. Priv.*, 2007, pp. 296–310.

[17] Y. Jin et al., "Hardware trojan detection using path delay fingerprint," in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2008, pp. 51–57.

[18] I. Exurville et al., "Resilient hardware trojans detection based on path delay measurements," in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2015, pp. 151–156.

[19] X.-T. Ngo et al., "Hardware trojan detection by delay and electromagnetic measurements," in *Proc. Des. Autom. Test Europe*, 2015, pp. 782–787.

[20] F. Wolff et al., "Towards trojan-free trusted ics: Problem analysis and detection scheme," in *Proc. Des. Autom. Test Europe*, 2008, pp. 1362–1365.

[21] S. Dupuis et al., "New testing procedure for finding insertion sites of stealthy hardware trojans," in *Proc. Des. Autom. Test Europe*, 2015, pp. 776–781.

[22] N. Lesperance et al., "Hardware trojan detection using exhaustive testing of k-bit subspaces," in *Proc. Asia South Pac. Des. Autom. Conf.*, 2015, pp. 755–760.

[23] H. Salmani et al., "A novel technique for improving hardware trojan detection and reducing trojan activation time," *Trans. VLSI Syst.*, pp. 112–125, 2012.

[24] B. Zhou et al., "Cost-efficient acceleration of hardware trojan detection through fan-out cone analysis and weighted random pattern technique," *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, pp. 792–805, 2016.

[25] R. S. Chakraborty et al., "On-demand transparency for improving hardware trojan detectability," in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2008, pp. 48–50.

[26] J. Rajendran et al., "Security analysis of logic obfuscation," in *Proc. Des. Autom. Conf.*, 2012, pp. 83–89.

[27] R. S. Chakraborty et al., "Harpoon: An obfuscation-based soc design methodology for hardware protection," *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, vol. 28, no. 10, pp. 1493–1502, 2009.

[28] J. Rajendran et al., "Security analysis of integrated circuit camouflaging," in *Proc. Comp. Comm. Sec.*, 2013, p. 709–720.

[29] K. Xiao et al., "Bisa: Built-in self-authentication for preventing hardware trojan insertion," in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2013, pp. 45–50.

[30] ——, "A novel built-in self-authentication technique to prevent inserting hardware trojans," *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, pp. 1778–1791, 2014.

[31] P.-S. Ba et al., "Hardware trust through layout filling: A hardware trojan prevention technique," in *Proc. Comp. Soc. Symp. VLSI*, 2016, pp. 254–259.

[32] K. Xiao et al., "Efficient and secure split manufacturing via obfuscated built-in self-authentication," in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2015, pp. 14–19.

[33] M. Rostami et al., "Hardware security: Threat models and metrics," in *Proc. Int. Conf. Comp.-Aided Des.*, 2013, pp. 819–823.

[34] T. D. Perez et al., "A survey on split manufacturing: Attacks, defenses, and challenges," *IEEE Access*, pp. 184 013–184 035, 2020.

[35] C. Bao et al., "Temperature tracking: Toward robust run-time detection of hardware trojans," *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, pp. 1577–1585, 2015.

[36] T. F. Wu et al., "Tpad: Hardware trojan prevention and detection for trusted integrated circuits," *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, pp. 521–534, 2016.

[37] T. Trippel et al., "Ical: an extensible framework for estimating the susceptibility of ic layouts to additive trojans," in *Proc. Symp. Sec. Priv.*, 2020, pp. 1742–1759.

[38] H. M. Kamali et al., "Advances in logic locking: Past, present, and prospects," Cryptology ePrint Archive, Paper 2022/260, 2022.

[39] G. Guo et al., "Assurer: A ppa-friendly security closure framework for physical design," in *Proc. Asia South Pac. Des. Autom. Conf.*, 2023, p. 504–509.

[40] J. Knechtel et al., "Security closure of physical layouts iccad special session paper," in *Proc. Int. Conf. Comp.-Aided Des.*, 2021, pp. 1–9.

[41] X. Wei et al., "Gdsii-guard: Eco anti-trojan optimization with exploratory timing-security trade-offs," in *Proc. Des. Autom. Conf.*, 2023, pp. 1–6.

[42] S. Mitra et al., "Which concurrent error detection scheme to choose ?" in *Proc. Int. Test Conf.*, 2000, pp. 985–994.

[43] M. Boulé et al., "Automata-based assertion-checker synthesis of psl properties," *ACM Trans. Des. Autom. Electron. Syst.*, 2008.

[44] Cadence Design Systems, Inc., "Jasper spv app," 2024. [Online]. Available: https://cadence.com/en_US/home/tools/system-design-and-verification/formal-and-static-verification/jasper-gold-verification-platform/security-path-verification-app.html

Mohammad Eslami (Graduate Student Member, IEEE) received his M.S. degree in computer engineering from the Shahid Bahonar University of Kerman, Kerman, Iran, in 2018. Currently, he is pursuing his doctoral studies at the Centre for Hardware Security, Tallinn University of Technology (TalTech), Tallinn, Estonia. His research interests primarily revolve around hardware security, with a particular focus on physical design automation and secure ASIC design.

Tara Ghasempouri (Member, IEEE) received the PhD degree from University of Verona, Italy in 2016. Currently, she is a Senior Researcher at Tallinn University of Technology (TalTech) in Tallinn, Estonia. Her fields of interest are Hardware verification and Hardware security. At the moment, she leads projects concerned with security of digital system as well as self-driving vehicle. She started collaboration with Taltech by receiving Mobilitas+ postdoctoral grant. She had research visitor position at the University of Delft, The Netherlands and TU Munich, Germany.

Samuel Pagliarini (Member, IEEE) received the PhD degree from Telecom ParisTech, Paris, France, in 2013. He has held research positions with the University of Bristol, Bristol, UK, and with Carnegie Mellon University, Pittsburgh, PA, USA. From 2019 to 2023, he led the Centre for Hardware Security at Tallinn University of Technology in Tallinn, Estonia. He is currently a professor at Carnegie Mellon University, Pittsburgh, PA, USA.

Appendix 4

IV

M. Eslami, T. Perez, and S. Pagliarini, “SALSy: Security-Aware Layout Synthesis,” *In arXiv*, under review for IEEE TDSC, 2024.
DOI: https://doi.org/10.48550/arXiv.2308.06201

SALSy: Security-Aware Layout Synthesis

Mohammad Eslami, *Graduate Student Member, IEEE,* Tiago Perez, *Graduate Student Member, IEEE,* Samuel Pagliarini, *Member, IEEE*

***Abstract*—Integrated Circuits (ICs) are the target of diverse attacks during their lifetime. Fabrication-time attacks, such as the insertion of Hardware Trojans (HTs), can give an adversary access to privileged data and/or the means to corrupt the IC's internal computation. Post-fabrication attacks, where the end-user takes a malicious role, also attempt to obtain privileged information through means such as fault injection and probing. Taking these threats into account and at the same time, this paper proposes a methodology for Security-Aware Layout Synthesis (SALSy), such that ICs can be designed with security in mind in the same manner as power-performance-area (PPA) metrics are considered today, a concept known as security closure. Furthermore, the trade-offs between PPA and security are considered and a chip is fabricated in a 65nm CMOS commercial technology for validation purposes – a feature not seen in previous research on security closure. Measurements on the fabricated ICs indicate that SALSy promotes a modest increase in power in order to achieve significantly improved security metrics.**

***Index Terms*—Hardware Security, Integrated Circuits, Layout Synthesis, Hardware Trojan, Fault Injection, Probing.**

I. Introduction

GLOBALIZATION of the Integrated Circuit (IC) supply chain has brought several benefits to IC vendors, including increased efficiency, cost savings, and access to specialized fabrication. The IC supply chain, today, is a complex network of entities involved in the processes of designing, manufacturing, testing, distributing, and marketing ICs. The limited availability of advanced silicon manufacturing sites, i.e., foundries, is the embodiment of the process of globalization. With the cost to establish a foundry in the range of billions of dollars, only a few companies are left in the competition for cutting-edge chip manufacturing. Hence, IC design companies can avoid these capital expenses by outsourcing the fabrication process and instead concentrating on their core skills, such as designing their specific ICs and the systems around them [1]. This arrangement in which IC design companies are fabless has been the norm for many years, and it is sustained by the significant investments semiconductor foundries make in R&D.

However, globalization has also created new security challenges. In the fabless model, the foundries are considered untrusted since design houses have no ownership or oversight claims over them. Hence, IC design houses should seek to protect their designs (layouts) against potential adversaries located in the untrusted foundries [2]–[5]. Such adversaries could perform IP theft, IC overproduction, many forms of reverse engineering, and also compromise the IC's functionality or reliability [6], [7]. Untrusted foundries may introduce malicious hardware, known as Hardware Trojan (HT), into the IC design. HTs can compromise device functionality or security [8], [9]. For instance, a foundry may introduce a backdoor into an IC that allows an attacker to remotely control the device, steal sensitive data, or inject malicious code [10].

Furthermore, there are many other threats beyond fabrication-time attacks. The finalized IC, once being available to a malicious end-user from the open market, may be targeted by an adversary through fault injection [7], [11], [12]. In this type of attack, the adversary tries to compromise the security of the chip by injecting different types of faults into it.

Another post-fabrication time attack is probing, in which the attacker tries to gain unauthorized access to the internal data of a chip by performing physical probing [13], [14]. This attack is mostly performed to extract sensitive data from within the chip, such as cryptographic keys or other proprietary information. Such attacks are even more relevant in dependable/critical applications [12].

With these concerns in mind, the notion of security closure [15] has been pursued by hardware security researchers. It involves accepting certain overheads in terms of power-performance-area (PPA) to achieve heightened security measures, which aim to minimize vulnerabilities and potential attack surfaces. The primary objective is to create trustworthy and robust ICs capable of withstanding potential security breaches and ensuring their reliable performance.

This paper presents a methodology for security closure consisting of different techniques. Our proposed flow, Security-Aware Layout Synthesis (SALSy), is generic and can be adopted in any layout, regardless of size, type, or technology. The main contributions of this work are as follows:

- Providing a generic approach for enhancing the security of designs during physical synthesis against multiple threats: (i) HTs, (ii) fault injection, and (iii) probing.
- Prototyping a chip in a commercial 65nm CMOS technology to validate SALSy in silicon.
- Comparing the use of commercial libraries and Process Design Kits (PDKs) with open-source ones in order to highlight the limitations and restrictions of using open-source PDKs for security research.
- Making the scripts readily accessible to the public to empower the research community to comprehensively verify and validate the techniques presented in this study. Furthermore, the scripts operate within a commercial

This work was partially supported by the EU through the European Social Fund in the context of the project "ICT programme". M. Eslami was also supported by HARNO (Grant No. 11.4-1/23/1).

M. Eslami, T. D. Perez, and S. Pagliarini are with the Department of Computer Systems, Centre for Hardware Security, Tallinn University of Technology (TalTech), 12618, Tallinn, Estonia (e-mail: mohammad.eslami@taltech.ee; diadamiperez@gmail.com; samuel.pagliarini@taltech.ee)

Samuel Pagliarini is also with the ECE Department at Carnegie Mellon University, Pittsburgh, PA (e-mail: pagliarini@cmu.edu)

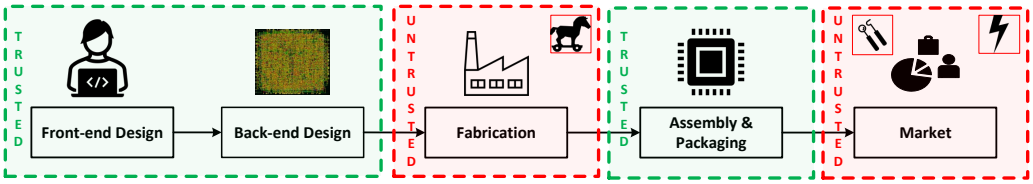


Fig. 1. Different threats in the IC supply chain: HTs are a fabrication-time threat, FSP/FI are post-fabrication threats.

physical synthesis tool, assuring that SALSy is compatible with current industry practices.

The rest of the paper is organized as follows: Section II provides background on major threats against the ICs after being sent for fabrication, along with previous efforts aimed at mitigating post-design-time attacks. In Section III, the details and different techniques we used to enhance the security of the designs using open-source PDKs are explained. Section IV points out the differences between commercial PDKs and open-source ones. The experimental results from the chip are presented in Section V, followed by a discussion in Section VI. Finally, Section VII concludes the paper.

II. Background

The IC supply chain encompasses various stages, from design and manufacturing to distribution and deployment, as depicted in Fig. 1. Each stage presents unique security challenges and potential threats.

Once the chip is sent for fabrication, the design team has no control over it. Hence, considerations must be taken into account to protect the chip against mentioned threats in the fabrication and post-fabrication stages. In this work, we focus on three major post-design threats: ***Hardware Trojans, Fault Injection,*** and ***Probing.*** In the following, we provide more details about each of these threats.

A. Hardware Trojans

As mentioned, a HT is a malicious modification to an IC that can cause harm to the system it is embedded in and can remain undetected for long periods of time. Its purpose is to modify the behavior of the IC in a way that benefits the attacker, harms the user, leaks sensitive information, or causes the IC to fail under specific conditions [8], [16]. HTs can be introduced into an IC in several ways, such as by modifying the layout of the design or by manipulating the fabrication process to introduce defects [17]. The HT's activation mechanism is called 'trigger' – the event or condition that initiates the attack. The trigger could be a specific input, a particular sequence of operations, or even a specific date or time [9].

The specialized literature classifies HTs into two main categories: functional and parametric. Functional HTs alter the functionality of a circuit, while parametric HTs change the circuit's performance parameters. This work focuses primarily on additive functional HTs. Additive HTs involve the insertion of additional malicious components into an IC. Conceptually, the HT logic can replace filler cells in a finalized layout [18], [19].

B. Fault Injection

Fault injection has emerged as a potent attack vector that adversaries can exploit to compromise the security and reliability of ICs [11]. Fault injection attacks involve intentionally disturbing digital circuits to disrupt their regular operation, manipulate data, or bypass security mechanisms [20], [21]. Adversaries can exploit various fault injection techniques and target specific vulnerabilities to achieve their malicious objectives [11], [12]. By tampering with supply voltages, clocks, or even electromagnetic fields, attackers can induce faults that lead to system failures, unauthorized access, or information leakage. Circuits that implement cryptographic operations are particularly sensitive to such attacks. Mechanisms exploited in fault injection attacks include (but are not limited to):

Timing Manipulation: Fault injection attacks often target the precise timing of digital circuits, exploiting vulnerabilities in clock signals, synchronization mechanisms, or critical timing paths to cause errors or disrupt the intended operation.

Power and Voltage Manipulation: Tinkering with power supply levels, injecting voltage spikes, or inducing power glitches can lead to circuit malfunction.

C. Probing

This type of attack involves physical access and measurement of internal signals within a digital circuit [13], [22]. Adversaries employ various methods, such as utilizing oscilloscopes, logic analyzers, or even direct physical contact, to capture and analyze signals. This technique allows attackers to extract sensitive information, such as encryption keys, proprietary algorithms, or critical data, by bypassing traditional security mechanisms. Probing is often performed in two ways:

Front-side Probing: This method allows unauthorized access to the internal data of an IC by making electrical connections with specific points on the surface of the chip (metal interconnects and active components) using specialized equipment like microprobes or needles. By reading the internal signals, an attacker can potentially extract sensitive information, including cryptographic keys or proprietary data.

Back-side Probing: This method involves physically accessing the back or underside of the chip to obtain internal data, requiring the removal of the protective packaging to expose the silicon die. This more invasive and destructive technique enables direct access to the chip's internal circuitry, providing valuable insights into its behavior.

Front-side probing and fault injection threats share similarities, as both require physical access to the IC. Combating

one often involves addressing the other, as their countermeasures overlap, including secure layout design. From this point onward, we will refer to these threats as one under the abbreviation **FSP/FI**.

D. Related Works

While numerous works focus on mitigating post-design-time attacks through methods like shielding to protect specific chip areas, only a limited number of studies propose defensive techniques targeting multiple threats during the physical synthesis phase – the final step before chip fabrication. Notably, the vast majority of proposed techniques have not been validated in silicon. This is particularly true for HT prevention techniques.

Defensive techniques proposed against HT insertion are mainly based on the idea of increasing the design's density, thus reducing the available space within the design where an attacker could potentially embed malicious logic.

In [23], a locking technique is introduced during physical synthesis to enhance security and bridge layout gaps. Despite its resilience to various attacks, its practicality is debatable as meeting timing constraints remains uncertain for many benchmarks. In [24], authors propose populating unused spaces with functional cells, creating an independent combinational circuit for post-fabrication testing against cell modifications. A limitation of this approach is achieving a high occupation ratio while maintaining the design routable. To address this limitation, [25] prioritizes filling gaps that could be used for HT insertion. However, this approach may alter initial routing, potentially defrauding critical paths due to rerouting.

In [26], another selective approach places sensitive logic in denser layout areas and steers gaps around less sensitive regions. However, the user's control over placement can be limited, particularly when using commercial CAD tools. In [27], authors address this by suggesting a placing refinement technique to segment large unused layout spaces. Yet, this approach faces a significant vulnerability where attackers can reverse the refinement, creating optimal zones for their malicious logic. In [28], authors attempt to cut the large gaps in the layout by shifting the cells and leveraging Engineering Change Order (ECO) features of commercial CAD tools. However, this technique worsens the timing in most cases leaving the design with negative timing slack.

Countermeasures against FSP/FI are principally based on the idea of covering security-sensitive elements with different metal layers since the attacker usually uses laser or ion beams to attack the chip. In [26], a method is introduced to route sensitive wires beneath regular ones and widen non-sensitive wires. While this incurs substantial overhead in routing resources, it does not assure wire protection, especially against advanced attack techniques. To tackle this, [27] proposes maximizing sensitive element coverage by rerouting nets and utilizing available design tracks. However, this approach's applicability is uncertain, particularly as the work uses an open-source PDK.

Shielding offers another avenue to safeguard sensitive elements. In [29], guard wires are introduced to shield sensitive nets, yet the focus remains on fabrication-time attacks. Alternatively, [14] presents an anti-probing physical approach via added steps in the synthesis flow. However, this method's efficiency faces challenges with increased sensitive element numbers, and it entails area and power overheads.

E. Threat Model

A vital step to implementing countermeasures against different attacks is identifying and focusing on the exact attacker's capabilities, intentions, and limitations – i.e., establishing a coherent threat model. As shown in Fig. 1, we assume all design stages, meaning front-end and back-end, are performed in a trusted environment. We also assume the packaging-related activities are performed in a trusted setting.

Still referring to Fig. 1, the first threat we consider in this work is that of a fabrication-time HT insertion. We assume that a rogue engineer inside the foundry has the ability to insert HTs into the finalized layout, and he/she is capable of using any type of advanced CAD tool. Since the adversary works for the foundry, he has access to the PDK. In our threat model, we specifically focus on additive HTs. We exclude other HT types, such as parametric ones, from our scope of interest. Parametric HTs differ in that they do not introduce new logic for the trigger and/or the payload part, leaving the adversary with less control of what data to leak or corrupt. Similar arguments can be made about negative (substitution) HTs, which also make for a less controllable malicious logic.

When the already packaged chip reaches the market as a ready-to-use product, we consider the end-user to be a potential adversary. In line with the specialized literature, we assume he/she is capable of using advanced means such as laser or focused ion beam generators, as well as precise measurement devices to perform fault injection and/or probing. In both cases, the goal of the attacker is to extract sensitive information out of the chip, either by injecting different types of fault or by creating a cavity through milling to expose the sensitive nets, followed by depositing a conductor in the cavity to form a contact pad on the chip's surface. In this case, an attacker is able to probe the created pad to extract the sensitive data. It should be noted that for the probing, we only consider front-side probing.

F. Security Assessment

Different metrics have been introduced by researchers to assess the complexity of inserting HTs into a specific physical layout [30]–[32]. For evaluating the security, in this work, we use the scoring framework introduced in [31]. The reason for using this framework is that it takes a variety of threats (FSP, FI, and HT insertion) rather than considering only one specific threat. Moreover, to mimic the real challenges that a security engineer would face during physical synthesis, design quality (i.e., PPA) is considered in the final scores as well. Hence, the overall score is a function of *design quality* and *security*, as shown in Eq. 1.

$$Score = DesignQuality \times Security \tag{1}$$

Where *DesignQuality* consists of a weighted distribution of power, performance (in terms of clock frequency), area,

Fig. 2. Example of exposed area (highlighted in red) for cell assets (adopted from [31]).

and routing quality, and *Security* consists of equally weighted metrics for FSP/FI and HT insertion.

To obtain the security scores for FSP/FI, first, a set of sensitive (security-critical) cells and their related interconnections (wires) are considered for each design. The cells are termed as ***cell assets*** and the wires are called ***net assets***. After that, a so-called ***exposed area*** metric is calculated for each set of cell and net assets in each design. The ***exposed area*** refers to any spatial area of those cell/net assets, whether continuous or fragmented, that is reachable through the metal stack from the top. An example of an exposed area is shown in Fig. 2.

For the HT-related portion of the Security score, an ***exploitable region*** metric is defined as the set of continuous placement sites[1] that are either i) free ii) occupied by filler cells or non-functional cells, or iii) unconnected cells. The criterion for identifying continuous placement sites as an ***exploitable region*** is met when the number of the sites reaches a minimum threshold of 20. Moreover, free routing tracks around the exploitable region(s) are also considered. The main idea is that an adversary needs placement resources **and** routing resources to insert an HT successfully. Hence, there should be enough gaps in the layout or some logic that can be easily removed.

The baseline layouts, prior to applying any security closure techniques, have a default score of 1. The modified layouts that improve design quality and/or security would be scored between $[0, 1)$, whereas poor modifications would be scored between $(1, \infty]$.

To obtain each element of the scoring formula (i.e., power) in Eq. 1, first, the relative metric for the baseline layout is calculated. Then, the same number for the modified (secured) layout is obtained. By dividing the values of the secured version by the values of the baseline version, the score for that specific component is computed. Then, by adding the relative weights to each element, the final score is obtained. Therefore, Eq. 1 can be expanded as shown in Eq. 2.

In this equation, *des_p_total*, *des_perf*, *des_area*, and *des_issues* denote the power, performance in terms of timing violations (if any), area, and Design Rule Checks (DRCs) respectively. The terms *ti_sts* and *ti_fts* represent the exploitable regions and available routing resources (free tracks) of exploitable regions, whereas *fsp_fi_ea_c* and *fsp_fi_ea_n* indicate the exposed area of the cell assets, and the exposed area of the net assets, respectively.

[1]A placement site represents a predefined valid location where a cell is legally placed; placement sites are typically a function of the standard cell height and the contacted poly pitch.

III. Security-Aware Layout Synthesis

In this section, we present SALSy, the main contribution of this work. We will further present two views of SALSy, one pre-silicon view that is compatible with open-source PDKs and one post-silicon view that is more realistic and therefore corresponds to commercial PDKs. We will also show results that are collected from a real fabricated chip that implements the SALSy concepts. Nevertheless, the comparison to open-source PDKs is important because it allows us to compare our work to others under the framework established in [31].

An overview of the used strategies and their relative order is shown in Fig. 3. It can be seen that not all techniques that are adequate for an open-source PDK can be used in an actual tapeout. The adopted color scheme indicates that, where green-colored rectangles are utilized to mark techniques that are fully compatible with commercial PDKs.

Benchmarks: The benchmarks chosen for the open-source experiment mostly consist of crypto cores (CAST, Camellia, MISTY, PRESENT, OpenMSP430_1, three versions of AES, SEED, TDEA, OpenMSP430_2, and SPARX [33]–[35]).

Open-source PDK: Like similar academic efforts, the chosen PDK/standard cell library in [31] is the Nangate 45nm Open Cell Library [36] since it is freely available. The metal stacks considered are 6M and 10M, depending on (complexity of) the benchmark.

We remind the reader that the scoring formula emphasizes a balance between security and PPA, as evidenced in Eq. 2. On the design side, even if we did employ customized implementation scripts for each and every benchmark, the scripts correspond to traditional parameter exploration in physical synthesis and therefore will not be discussed in detail. In the text that

$$Score = \overbrace{\left(0.1 \times \frac{(des\_p\_total)_{\text{sec}}}{(des\_p\_total)_{\text{bl}}} + 0.3 \times \frac{(des\_perf)_{\text{sec}}}{(des\_perf)_{\text{bl}}} + 0.3 \times \frac{(des\_area)_{\text{sec}}}{(des\_area)_{\text{bl}}} + 0.3 \times \frac{(des\_issues)_{\text{sec}}}{(des\_issues)_{\text{bl}}}\right)}^{DesignQuality} \times \left(\overbrace{\frac{1}{2} \times \left(0.5 \times \frac{(fsp\_fi\_ea\_c)_{\text{sec}}}{(fsp\_fi\_ea\_c)_{\text{bl}}} + 0.5 \times \frac{(fsp\_fi\_ea\_n)_{\text{sec}}}{(fsp\_fi\_ea\_n)_{\text{bl}}}\right)}^{Security\ (fsp/fi)} + \overbrace{\frac{1}{2} \times \left(0.6 \times \frac{(ti\_sts)_{\text{sec}}}{(ti\_sts)_{\text{bl}}} + 0.4 \times \frac{(ti\_fts)_{\text{sec}}}{(ti\_fts)_{\text{bl}}}\right)}^{Security\ (ti)}\right) \tag{2}$$

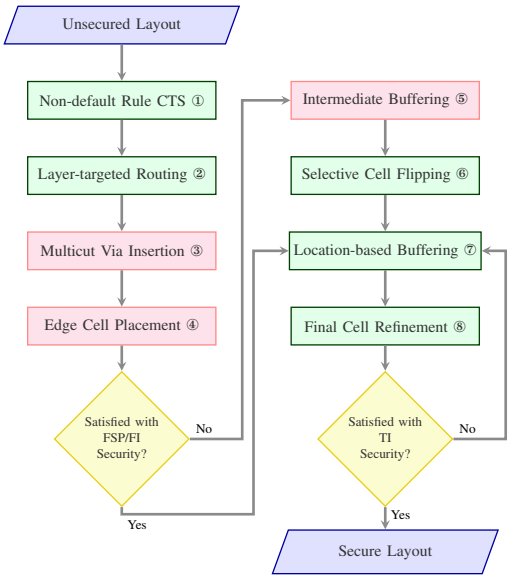


Fig. 3. SALSy framework. Red boxes highlight techniques that are not feasible for the tapeout. Green boxes highlight techniques that can be used in both open-source PDKs and in our tapeout.

Algorithm 1 Layer-targeted Routing Algorithm

1: $net\_assets \leftarrow List\_of\_net\_assets$
2: $other\_nets \leftarrow List\_of\_other\_nets$
3: $prf\_lays\_assets \leftarrow [M2, M3]$
4: $prf\_lays\_others \leftarrow [M4, M5, M6]$
5: $width\_for\_assets \leftarrow width(M2)$ ▷ This value is the minimum width according to the library
6: $width\_for\_others \leftarrow width(M2) \times 2$
7: **foreach** *net* **in** *net_assets* **do**
8: **assign** *prf_lays_assets* **to** *route_layer*
9: **assign** *width_for_assets* **to** *width_rule*
10: **end for**
11: **route** *net_assets* **with** *width_ruler* **in** *route_layer*
12: **if** $(route\_err)$ **then**
13: **route** *net_assets* **with** *default_rules*
14: **end if**
15: **foreach** *net* **in** *other_nets* **do**
16: **assign** *prf_lays_others* **to** *route_layer*
17: **assign** *width_for_others* **to** *width_rule*
18: **end for**
19: **route** *other_nets* **with** *width_ruler* **in** *route_layer*
20: **if** $(route\_err)$ **then**
21: **route** *other_nets* **with** *default_rules*
22: **end if**

follows, we will focus on the security aspects. Furthermore, since the scoring formula considers different metrics for front-side probing/fault injection (*fsp_fi*) and HT insertion (*ti*), we explain the related SALSy techniques separately.

A. Countermeasures against FSP/FI

1) Non-default Rule Clock Tree Synthesis: The insight of this strategy is to change the default rules for Clock Tree Synthesis (CTS)[2] in order to cover more assets by enlarging the clock distribution wires. Note that CTS routing utilizes fewer resources than signal routing. Hence, CTS wires can be widened many times more than signal wires. As shown in Fig. 4(a), the enlarged clock tree can significantly cover more exposed areas under it. Quite often the quality of the CTS is improved by using non-default rules.

2) Layer-targeted Routing: Recall that the exposed area metric, which applies to cells and nets, corresponds to the asset area directly accessible from the front side. In the first step, we try to hide the net assets under the other non-asset nets to protect them against FSP/FI, as shown in Algorithm 1. For this purpose, we dedicate the lowest possible metal layers[3] to the net assets only (line 3). It should be noted that we use the minimum width for routing these asset-related wires to be later able to 'hide' them under the other nets (line 5).

Next, all remaining non-asset nets are defined to be routed with higher metal layers (line 4). Furthermore, we choose a wider width instead of the default one to increase the chance of covering net and cell assets (line 6). If the routing tool fails to route the nets with the modified width or in the preferred metal layer, it will try to route them with the default width and in default metal layers (lines 12-14, 20-22). For the physical synthesis tool utilized, routing constraints are soft constraints. I.e., the tool will try the hardest to follow the constraints; if it fails, the constraints are disregarded.

As an example, in Algorithm 1, we consider M2 and M3 layers for routing only the net assets (line 3), and the higher metal layers M4-M6 are used for non-asset net routing (line 4). In this example, the width of the non-asset nets is 2x wider than the asset nets (line 6), but this number can be increased if more resources are available[4]. After applying this technique, the congestion increases significantly, and therefore, more cell assets and net assets can be protected against FSP/FI, as shown in Fig. 4(b).

3) Multi-cut Via Insertion: A vertical connection named *via* (cut) connects different metal layers in the metal stack of an IC. By default, the physical synthesis tool uses the minimum number of vias and smallest vias available for the connection to optimize routing resource usage and prevent routing congestion. However, our strategy aims to increase the congestion on top of the cell assets to cover them as much as possible. Hence, we use multi-cut vias between M1 and M2 layers. By doing this, a larger piece of metal is routed on the top of the cell assets, improving the coverage, as shown in

[2]CTS is an essential step in the design process of digital ICs, involving the construction of a network of clock branches to distribute clock efficiently signal across the entire circuit. By carefully constructing the network, observing delay balancing, and skew management, timing can be improved and power consumption can be reduced. CTS routing usually takes precedence and priority over signal routing, which we leverage for security purposes.

[3]Several metal layers form a metal stack, where the lowermost layers are usually the thinnest of them.

[4]In a real IC, the number of metal layers depends on the technology/metal stack agreed with the foundry. Current technologies often offer 10+ layers.

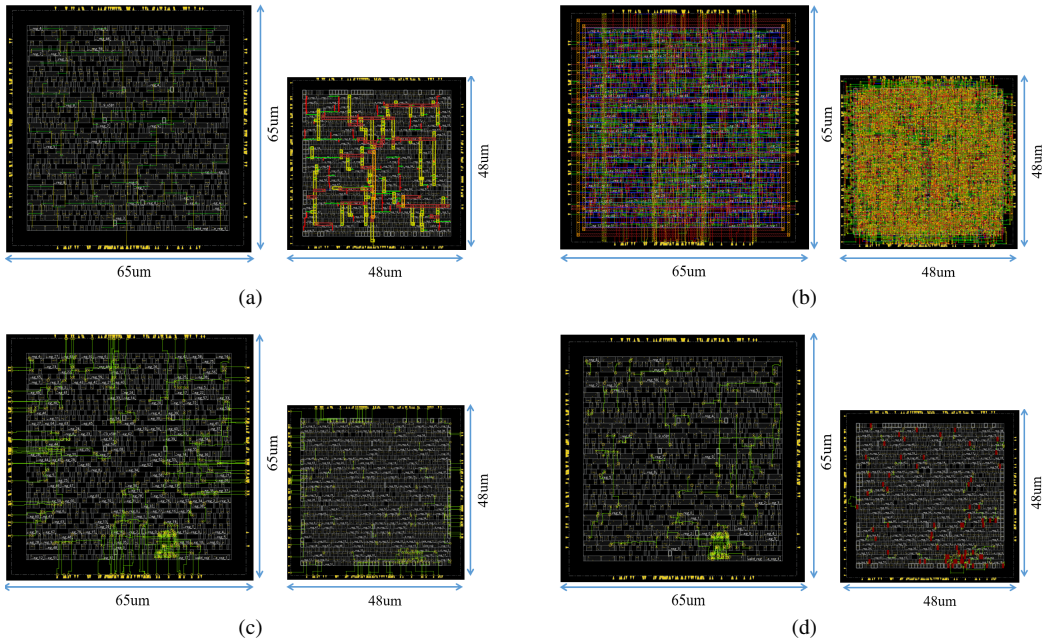


Fig. 4. Different techniques used in SALSy. The design on the left is always the BL variant, and the design on the right is always the SEC variant. a) Non-default Rule CTS, b) Increased congestion by applying Layer-targeted Routing, c) Edge Cell Placement for shortening the long net assets (highlighted in green), and d) Reducing the length of the net assets (highlighted in green) by applying Intermediate Buffering technique (added buffers appear in red).

Fig. 5. The reason that we use multi-cut vias only between M1 and M2 layers is that we do not want to touch the higher metal layer resources, leaving them for signal routing.

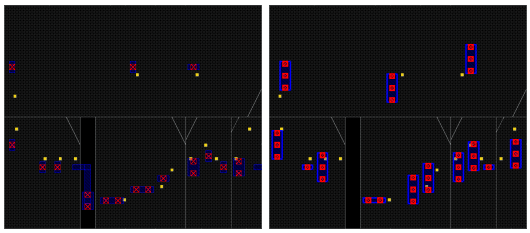


Fig. 5. Using the default rules for via insertion (left) and multi-cut via insertion (right) to increase the coverage of cell assets

4) Edge Cell Placement: In some of the benchmarks, it is observed that net assets include long wires that travel a long distance from IO pins to their sinks (see the wires highlighted in green in Fig. 4(c)). Hence, we use a technique in which the sink cell connected to the IO pins via net assets is moved to the closest possible position to their driver. In this case, the length of the net assets becomes significantly shorter, which makes it easier to be covered by other nets on upper layers since shorter nets tend to have fewer turns and jogs.

5) Intermediate Buffering: The previous technique for shortening net assets only works for those long wires connected to the IO pins. However, net assets are not always connected to IO pins, and it is challenging to protect them due to their long length. In this case, when both driver and sink are inside the core area (the region where all the cells are placed in), we put buffer(s) in between them to reduce the length of the long net assets, as shown in Fig. 4(d). It should be noted that buffer insertion might significantly impact the design's timing and power consumption. Hence, we consider this technique iteratively with multiple checkpoints. If the buffer insertion introduces a timing-violating path, that buffer can be removed, and the circuit goes back to its previous state without the violation.

6) Selective Cell Flipping: In some cases, the exposed area of the net assets can be significantly reduced by changing the orientation of the cell (flip over the Y axis). In this case, the physical synthesis tool automatically re-routes the nets connected to the flipped cell, and the chance of hiding the net asset under other nets increases, as depicted in Fig. 6. It should be noted that this technique is performed in the very last steps of our methodology, and only the net assets with the worst exposed area are selected.

B. Countermeasures against HT Insertion

We now explain the techniques used for preventing HT insertion. Recall the *exploitable region* notion, a set of continuous gaps, filler cells, unconnected cells, or non-functional cells, that an adversary can use to place his/her malicious logic. In principle, available routing resources are also considered when determining exploitable regions since the HT circuitry needs to be somehow connected to the original

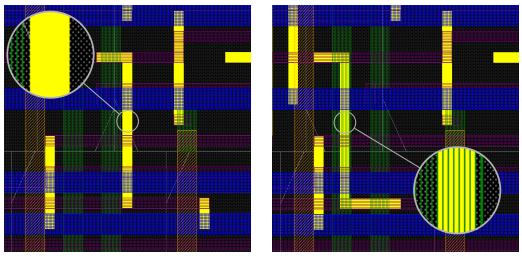


Fig. 6. An example of covering a net asset by flipping the cell: The exposed area (solid yellow regions in the left image) is totally covered by the nets in the upper metal layer(s) after the net is re-routed (right image).

design. However, our efforts are directed toward mitigating the free placement sites. This is mainly because if we eliminate all the gaps, there would be no room for the HT cells to be placed in the design. Therefore, available routing resources become meaningless.

7) Location-based Buffering: Even after shrinking the design area to make the core area as dense and compact as possible, several gaps may still exist and create large exploitable areas. As the threshold for continuous gaps to be considered an exploitable region is 20 placement sites, we developed a script to search for these regions and insert buffers to fill or cut the continuous gaps below 20 sites. It should be noted that whenever buffer insertion is considered, there are overheads in power and, potentially, in timing. However, the trade-off between security and PPA is arguably beneficial for this particular technique.

8) Final Cell Refinement: There might be cases where the buffer insertion fails due to a lack of routing resources in congested areas. It may also succeed but create timing violations. We try to eliminate the remaining exploitable regions in these cases by slightly moving the surrounding cells. This simple technique can be done using algorithmic approaches as introduced in [27] or manually by the physical design engineer (if there are few cases).

Fig. 7 shows how we eliminate all exploitable regions in a design by adopting the mentioned techniques.

C. Scores for Open-source PDK and Comparisons

Here, we provide the scores for each benchmark after applying the mentioned techniques for the open-source PDK we have considered. Moreover, we compare our results with other works that enhance the security of the layouts for the same benchmarks. For this purpose, we collect the results of the participating teams in the security closure contest which was held along with the ISPD'22 conference. The logistics and structure of the contest are described in [31].

The score formula, given in Eq. 1, is meant to normalize results against the baselines while giving equal importance to design quality and security. However, since there is a multiplicative factor between these two score components, achieving a score of 0 in security (which is impossible in practice) would also bring the overall score to 0.

Contestants eventually realized that drawing a single metal plate above the entire design was sufficient to eliminate all security threats. This solution corresponds to using an entire metal layer as a sacrificial layer. This solution does create DRC violations, but the scoring formula, unfortunately, fails to penalize these solutions: since the security score is zero, the design component of Eq. 1 is irrelevant. All in all, the ISPD'22 contest ended with multiple teams tied with "perfect scores" of zero. The final scores are shown in Tab. I. This work is identified as team 'K'.

TABLE I
OVERALL SCORES OF THE PARTICIPATING TEAMS

Benchmarks / Teams	J	N	O	E	L	A	Q	K
AES_1	0.764	0.025	0.000	0.000	0.271	0.000	0.000	**0.000**
AES_2	1.687	0.054	0.000	0.000	0.324	0.000	0.000	**0.000**
AES_3	1.332	0.000	0.000	0.000	0.295	0.000	0.000	**0.000**
Camellia	0.676	0.000	0.000	0.000	0.281	0.000	0.000	**0.000**
CAST	1.687	0.000	0.000	0.000	0.300	0.000	0.000	**0.000**
MISTY	3.178	0.000	0.000	0.000	0.254	0.000	0.000	**0.000**
OpenMSP430_1	0.841	0.000	0.000	0.000	0.344	0.000	0.000	**0.000**
PRESENT	0.629	0.000	0.000	0.000	0.319	0.000	0.000	**0.000**
SEED	2.203	0.000	0.000	0.000	0.207	0.000	0.000	**0.000**
TDEA	0.596	0.003	0.000	0.000	0.246	0.000	0.002	**0.000**
OpenMSP430_2	1.031	0.000	0.000	0.000	0.822	0.000	0.000	**0.000**
SPARX	0.476	0.000	0.000	0.000	0.262	0.000	0.000	**0.000**

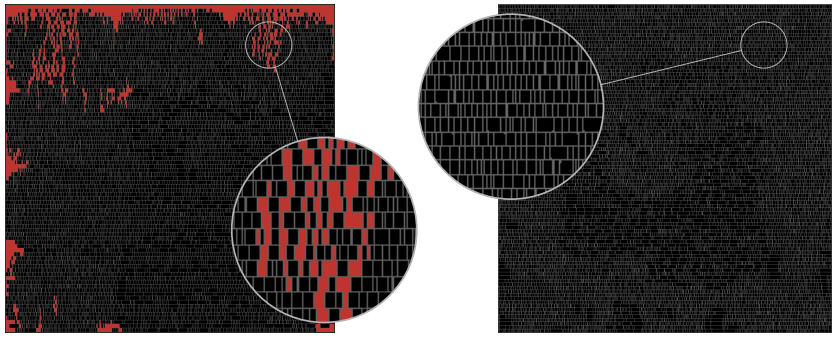
To be very clear, we emphasize that the sacrificial metal layer solution has no practical merit. It does not effectively protect against any of the considered threats. It is only effective in satisfying the contest's scoring. A clearer picture of the overheads imposed by our techniques can be seen in Tab. II, where it can be seen that when considering only the design component of Eq. 1, our scores are rather competitive. These results, along with the considerations of whether our techniques could be ported to a commercial PDK, led us to pursue a tapeout. This effort is described in the next section.

TABLE II
DESIGN QUALITY SCORES OF THE PARTICIPATING TEAMS

Benchmarks / Teams	J	N	O	E	L	A	Q	K
AES_1	0.995	0.713	0.447	0.475	0.527	0.519	1.347	**0.481**
AES_2	3.737	0.702	0.425	0.458	0.539	0.509	0.817	**0.461**
AES_3	2.689	1.059	0.473	0.498	0.566	0.541	1.171	**0.523**
Camellia	0.753	0.746	0.398	0.420	0.470	0.418	0.960	**0.530**
CAST	1.663	0.851	0.412	0.409	0.463	0.439	0.908	**0.495**
MISTY	5.009	0.753	0.418	0.396	0.457	0.417	1.559	**0.458**
OpenMSP430_1	0.756	0.656	0.406	0.440	0.490	0.469	1.025	**0.632**
PRESENT	0.752	0.693	0.359	0.427	0.465	0.446	1.009	**0.306**
SEED	1.917	0.892	0.416	0.442	0.418	0.442	0.924	**0.522**
TDEA	0.750	0.846	0.459	0.526	0.534	0.524	0.808	**0.584**
OpenMSP430_2	0.995	0.777	0.464	0.543	0.524	0.570	0.848	**0.608**
SPARX	0.753	0.663	0.397	0.420	0.422	0.404	1.047	**0.509**

IV. SILICON VALIDATION OF SALSY

In the previous section, we provided details about the different techniques that can be used to improve the security of an IC based on specific evaluation metrics [31]. However, all these techniques were initially developed for open-source PDKs. Commercial PDKs that are used by industry are more detailed than the academic ones and using these commercial PDKs increases the design complexity and introduces certain practical limitations. Hence, we decided to fabricate a chip with the mentioned security features to highlight the gaps and



(a) (b)

Fig. 7. An example of a) design with exploitable regions (highlighted in red), and b) design with zero exploitable regions using our techniques.

limitations with the open-source PDKs and provide solutions to cope with these limitations.

The first step in designing our chip is to localize the scoring system with respect to the commercial library so that we can evaluate the security features of the chip while adopting the same metrics from [31]. Next, we have to decide which designs we are going to tapeout. We have opted for a small chip size (1 mm^2) that is enough to fit four designs arranged as eight blocks: four secured versions (SEC) and four baseline versions (BL). By having a pair of each benchmark on the same chip, we can evaluate and compare the security and design quality of each block fairly. The designs were selected to provide variety in terms of complexity and size – The final candidates include Camellia, CAST, PRESENT, and SEED. The floorplan of the chip is shown in Fig. 8. Different blocks are highlighted in different colors and the rest of the core area is dedicated to the comparison and control unit. To make the comparison fair, we set the same density for all BL versions of the benchmarks as they were set in the open-source experiment. For the SEC variants, density is a function of the SALSy methodology. Furthermore, we implement distinct power domains for each block. This strategic approach enables us to activate only one block at a time while ensuring all other blocks remain in an off-state. Consequently, we can precisely measure the power consumption, one block at a time.

A microscope view of our fabricated chip is shown in Fig. 9. We validate the efficiency of our techniques on all four benchmarks implemented on our chip. However, it should be noted again that our approach is applicable to any design, regardless of function or size. Another advantage of our solution is that it does not need previous knowledge about the design since it is performed at the layout level. We hypothesize that SALSy creates an opportunity to assign security closure to a separate design team since no specific details/characteristics of the design are needed for improving security. The interface between this team and the traditional physical synthesis team would be a simple list of assets.

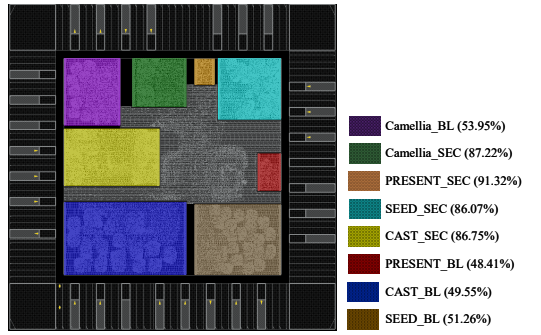


Fig. 8. Floorplan view of the chip including eight blocks and density of each block.

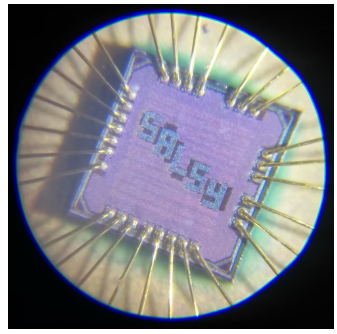


Fig. 9. Microscope view of the fabricated 1mm^2 chip.

A. Implementation

A key factor to improve HT and FSP/FI security is to increase the design's density. By doing this, first, the chance of HT insertion decreases by reducing the number of gaps, and second, more cell and net assets can be protected against FI/FSP due to the increased wire congestion. Therefore, we shrink all designs as much as possible before applying any specific technique. Note the smaller size of the SEC variants

with respect to the BL variants in Fig. 8. Also note the remarkably high density levels that were achieved for the SEC variants. In the text that follows, we provide more details about the chip implementation. It should be noted that the we have also released several scripts for the implementation flow in a GitHub repository [37]. This too differentiates our work from previous security closure attempts that remain largely impossible to reproduce.

1) Non-default Rule CTS: This technique is applicable to commercial libraries as well but with some restrictions regarding the maximum width of the wires. These restrictions are imposed by the foundry to maintain design compatibility with their equipment. In other words, we cannot enlarge the clock wires by arbitrary factors as we did in the open-source experiment, but still, it is possible to enlarge them from their default size. Although this technique is less effective compared with the open-source experiment, we decided to keep using it since it has negligible overhead on the power and other performance-related metrics.

2) Layer-targeted Routing: Similar to what we did in the open-source experiment, we use the routing strategy from Algorithm 1. The only difference here, when using commercial libraries, is that they are more detailed and characterized to ensure the design rule accuracy and verification. Hence, as the design gets denser, more violations appear due to different reasons to ensure the quality and reliability of the chip during the fabrication process. This means that achieving high density (above 90% for the considered 65nm technology) is very challenging. As a consequence, it is impossible to route all the asset or non-asset nets in their preferred metal layers as we did in the open-source experiment. However, even with the cost of routing some of the asset nets in the top metal layers, this technique still helps to cover a large portion of the exposed area of the assets (see Section V for more details).

3) Multicut Via Insertion: This method is the first one we had to abandon due to the strict constraints in the commercial PDK. Although it is theoretically possible to use multi-cut vias to connect the pins, it creates DRC issues after the wires are connected to the vias. Given the significant challenge of addressing numerous DRC violations, adopting this method in our chip was not practical. Yet, this solution can be revisited for a different commercial library/PDK. Multi-cut vias are commonly employed for power routing, and not so commonly for signal routing as we have attempted.

4) Edge Cell Placement: An important difference between the open-source experiment and an actual tapeout is that each design was treated as a separate chip in the open-source experiment, while we have to put all the designs together on one chip. Hence, we have little freedom in defining the IO pin locations for each design. The decision to put the IO pins on one of the block sides (left, right, bottom, or top) is defined during top-level floorplanning. For instance, if we consider the location of the *PRESENT_SEC* block in Fig. 8, the best place to put the IO pins is on the bottom edge of the block since it has the closest distance to the control and comparison unit which is located in the center of the chip. In this case, the routing would be done with fewer issues and unnecessary resource utilization would be avoided, leading to a more optimized floorplan.

Fig. 10. A design (PRESENT) with most of the IO pins on the bottom side and the net assets (highlighted in white) connected to their relative IO pin.

This restriction in pin placement leads to the limitation of using this technique in our chip. As shown in Fig. 10, there is only a limited area close to the IO pins that are connected to the net assets (highlighted in white), and it is impossible to fit all the connected cells near their driver/sink pins since the congestion in that region increases significantly and the design becomes unroutable. Hence, this technique cannot be used in this specific floorplan. However, it is crucial to highlight that this limitation does not impede the potential application of this technique in other chip implementations, as in the open-source experiment that we could fit the cells near to their relative IO pins due to the square shape of the block and the fact that there was enough space around all four sides of the design (Fig. 4).

5) Selective Cell Flipping: This technique can be used in chip implementation as well as in the open-source experiment setting without facing any specific limitations. However, we minimize the use of this method due to its inherent manual nature. Our overarching objective is to uphold a holistic and automated approach, eschewing the adoption of selective techniques, in order to ensure the comprehensive applicability of our approach.

6) Intermediate Buffering: As mentioned in the previous section, buffer insertion can have undesirable effects on the timing and power of the design. In the open-source experiment, such issues were only considered as a negative factor in the final score to penalize the teams. But in the actual chip, any single issue that violates the timing of the design (i.e. setup time, hold time, etc.) was considered unacceptable. Hence, the timing closure of the design should be perfect and the trade-off between the timing issues and the enhanced security is only possible as long as the timing slack remains positive. Due to this reason, we replaced this technique with a smarter buffer insertion algorithm which is explained in the text that follows.

7) Location-based Buffering: As mentioned in the previous part, we change our buffer insertion strategy such that it is no longer aimed at shortening the long net assets. Instead, it is totally focused on filling the continuous gaps of the design in a completely automated fashion. In other words, the buffer insertion technique turns into a location-based algorithm

Fig. 11. Added buffers (highlighted in red) using our smart algorithm to eliminate exploitable regions.

that seeks exploitable regions, rather than searching for long net assets. The sinks and the drivers of the added buffer are selected from the nearby cells such that it has the least negative impact on the timing of the design, as highlighted in Fig. 11.

8) Final Cell Refinement: Similar to what we did in the open-source experiment, this technique is used in the very late steps of the chip implementation as manual fixes. If any exploitable region is left that can be eliminated with a few cell movements, it is possible to use this method. However, we decided to minimize this technique in our chip for two reasons: i) if an exploitable region is eliminated simply by moving the cells, the adversary can revert the changes to make enough space for his/her malicious logic as well. Although being effective in the open-source experiment, in a realistic scenario it seems to be a useless effort; ii) It conflicts with our aim to create an automated flow.

V. Results

In this section, we provide the experimental results obtained from chip design and measurements. During the physical implementation process, we used Cadence and Siemens toolchains and our target technology is a general-purpose flavor of a 65nm CMOS technology.

We divide the results into two parts, pre-silicon and post-silicon results. The first part represents the results obtained from the final layout sent for fabrication such as the area and density of the blocks. Physical chip measurements, such as power consumption, are provided in the second part.

A. Pre-silicon Results

As mentioned before, we evaluate our techniques using the same scoring system used in [31]. In the following, we provide more details about each metric.

Based on the considered metrics, the final scores of our approach are presented in Tab. III. The table clearly demonstrates that our best scores are related to HT insertion, providing strong evidence that SALSy can effectively function as a prevention technique in a realistic PDK setting. Conversely, as anticipated, our worst scores are associated with power, primarily resulting from the buffer insertion applied to enhance security. The power consumption of our designs consistently exceeded the baselines, as indicated by values greater than 1.0 for all cases. Nevertheless, it is important to acknowledge that a certain level of overhead is inevitable when trading off for enhanced security. The power numbers reported in Tab. III are estimates from physical synthesis; precise power consumption numbers are provided in Section V-B.

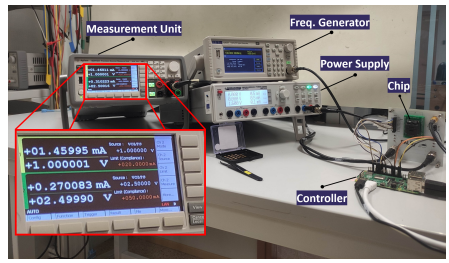


Fig. 12. The testing environment for the fabricated chip.

The presented table illustrates a substantial reduction in the count of *exploitable regions* within our secured version across all benchmarks. This reduction is notably profound, with a 100% decrease observed in the case of Camellia and PRESENT benchmarks. Furthermore, for the CAST and SEED benchmarks, the reduction percentages stand at 95.3% and 90.3% respectively. Regarding FSP/FI assessment, benchmarks exhibit varying outcomes. PRESENT benchmark excels with an impressive 43% *exposed area* reduction from the baseline, while the CAST benchmark demonstrates a more moderate 18.5% enhancement.

To provide a comprehensive understanding of the relationship between each step of SALSy and the resulting scores, individual scores for the PRESENT benchmark are presented in Tab. IV after subsequently applying each technique. The table reveals that the Layer-targeted Routing step has the most significant effect on the *fsp_fi* and overall scores, given its substantial impact on increasing congestion. On the other hand, the Location-based Buffer Insertion technique has the most substantial impact on enhancing the *ti* score, as it drastically reduces the number of gaps in the layout. Remarkably, the overall trend of score improvement, as displayed in Table IV, remains consistent for all other three benchmarks. We omit these results for the sake of space.

B. Post-fabrication Results

In this section, we present the measurement results obtained from the actual chip. The testing environment, illustrated in Fig. 12, comprises several components: a controller responsible for serial communication, input feeding, output reading, and data analysis; a power supply; a frequency generator providing a fast clock; and a precise measuring unit for assessing the chip's power consumption under various scenarios. We conducted the experiments on 20 packaged chips, chosen from a total of 100 fabricated chips.

1) Verifying the Chip Functionality: Before proceeding with power measurements, it is crucial to ensure that our chips, particularly their blocks, are functioning as intended. To

TABLE III
FINAL SCORES OF OUR APPROACH FOR FOUR DIFFERENT BENCHMARKS

Metrics / Benchmarks			Camellia	CAST	PRESENT	SEED
Design Quality	DRC	des_issues	0.000	0.000	0.000	0.000
	PPA	des_perf	0.000	0.000	0.000	0.000
		des_p_total	1.184	1.072	1.161	1.041
		des_area	0.686	0.606	0.597	0.627
	Overall	des	0.467	0.419	0.439	0.417
Security	Trojan Insertion	ti_sts	0.000	0.015	0.000	0.026
		ti_fts	0.000	0.079	0.000	0.169
	Overall	ti	0.000	0.047	0.000	0.097
	FSP/FI	fsp_fi_ea_c	0.842	0.797	0.293	0.762
		fsp_fi_ea_n	0.624	0.833	0.568	0.835
	Overall	fsp_fi	0.733	0.815	0.430	0.799
Final score		OVERALL	0.171	0.181	0.094	0.187

TABLE IV
THE CHANGES IN THE SCORES OF PRESENT BENCHMARK AFTER APPLYING OUR TECHNIQUES IN EACH STEP

Metrics / Steps			Non-default Rule CTS	Layer-targeted Routing	Location-based Buffer Insertion	Final Cell Refinement
Design Quality	DRC	des_issues	0.000	0.000	0.000	0.000
	PPA	des_perf	0.000	0.000	0.000	0.000
		des_p_total	1.018	1.138	1.159	1.161
		des_area	0.597	0.597	0.597	0.597
	Overall	des	0.404	0.434	0.436	0.439
Security	Trojan Insertion	ti_sts	0.010	0.011	0.005	0.000
		ti_fts	0.116	0.117	0.071	0.000
	Overall	ti	0.063	0.064	0.038	0.000
	FSP/FI	fsp_fi_ea_c	0.913	0.318	0.315	0.298
		fsp_fi_ea_n	0.985	0.586	0.583	0.568
	Overall	fsp_fi	0.949	0.452	0.449	0.430
Final score		OVERALL	0.204	0.112	0.106	0.094

accomplish this, we developed a Python script that systematically activates the blocks one by one at the target frequency, while simultaneously verifying the validity of the output data for each chosen block. All chips are deemed functional and we proceed with power measurements. It is worth mentioning that our target frequency for all blocks is 100MHz, whereas the clock frequency for the comparison and control unit is set to 1MHz. A fast 100MHz reference clock is generated by an external frequency generator, as depicted in Fig. 12. We remind the reader that total power is the sum of dynamic and static (leakage) power, which we will report separately. Our dynamic power results are reported at 100MHz.

2) Leakage Power Measurement: Once the functionality of the chip has been verified, we proceed to measure the power consumption. We begin by assessing the Always On (AO) leakage power of the chip. As the name suggests, AO indicates that this type of power consumption is present consistently, regardless of whether the IC is actively performing computations or tasks (functional mode), and measuring AO leakage power does not depend on the switching activity of the transistors of the chip. In this measurement, no inputs other than the power supply signals are fed into the chip. It allows us to capture the baseline power consumption when the chip is in its idle state, and no specific operations are being performed.

Following the AO leakage power measurement, we proceed with measuring the leakage power of each individual block. To achieve this, we activate one block at a time by asserting the appropriate configuration of the input signals specifically designed for the voltage island of that block. Each block is encompassed by power switches, granting us the capability to activate or deactivate them as needed. This meticulous power domain segregation significantly enhances measurement accuracy by eliminating power-sharing with any other block. Similar to the AO leakage measurement, no clock or any other signals are fed to the chip during this procedure. This allows us to precisely assess the power consumption of each individual block in isolation, shedding light on their specific power characteristics.

The leakage power results are depicted in Fig. 13. As illustrated, different chips exhibit distinct power signatures, which can be attributed to process variation. These variations are inherent in the semiconductor fabrication process and can lead to differences in power consumption among individual chips. The observed differences in leakage power highlight the significance of process variability in chip manufacturing and underscore the need for thorough testing and analysis of power characteristics in real-world chip deployments. The static power incurs an average overhead of 1.72%, 1.66%, 15.89%, and 7.24% across the PRESENT, SEED, Camellia, and CAST benchmarks, respectively.

3) Dynamic Power Measurement: The dynamic power measurement test is conducted to assess the power consumption of each design block in functional mode. To achieve this, we activate the blocks one by one and provide them

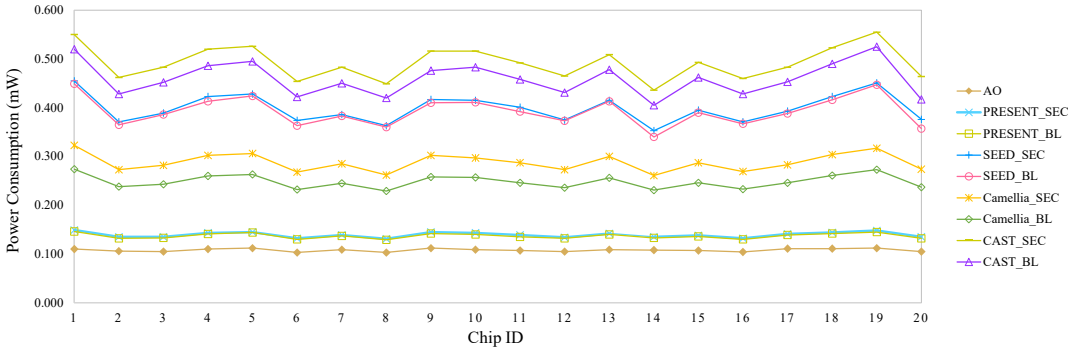


Fig. 13. The measured leakage power for 20 fabricated chips (in mW).

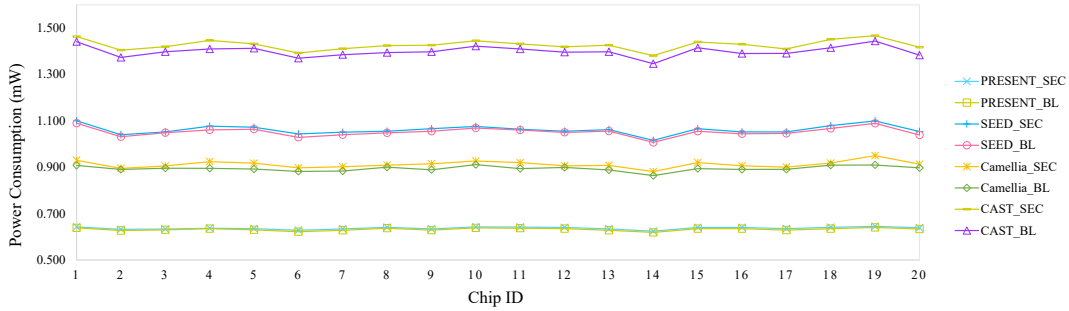


Fig. 14. The measured dynamic power for 20 fabricated chips (in mW).

with appropriate inputs (plain text) while operating at a clock frequency of 100MHz. The plain text inputs can be sourced either from an internal register bank within the chip or from the host controller through the UART protocol. The results of the dynamic power measurement are presented in Fig. 14. Across all benchmarks, the average overhead for dynamic power consumption remains below 3%. Specifically, the PRESENT, SEED, Camellia, and CAST benchmarks exhibit overheads of 0.79%, 0.86%, 2.02%, and 1.96% respectively.

VI. Discussion

SALSy aims to prevent post-design attacks and is evaluated with the metrics from [31]. However, some introduced metrics could arguably be redefined to make the evaluation more realistic - thus, the results could be readily leveraged by industry. For instance, considering the threshold of 20 continuous gaps for the exploitable area might be too optimistic since small HTs only occupy a few placement sites [38]. Furthermore, for the FSP/FI threats, the aim is to protect the design. For instance, in the ideal case, if an attacker tries to compromise the sensitive data by drilling a hole (milling), the chip should fail to operate due to damage to the protective nets above the sensitive nets. Hence, trivial defense schemes such as covering the whole core area with a large metal plate should not be considered a valid solution since the existence of this layer will not be an obstacle for the attacker.

Moreover, the scripted nature of SALSy is a strategic choice that aligns with the scalability needs of the industry, even for the more advanced technology nodes. All proposed techniques can be easily adopted to sub 65nm technology as well, although with more restrictions for more advanced nodes. For example, Non-default Rule CTS and Layer-targeted Routing rely on choosing wider wires, for which there might limited widths available to chose from. In older technologies, the foundry recipes were forgiving and would allow virtually any width between 1X-20X. A modern FinFET technology might allow widths such as 1X, 1.4X, 1.6X, and above. In other word, what used to be a continuum of valid widths is now a discrete set.

While designing a security-aware place and route engine might be an attractive academic pursuit, such an endeavor could lack the scalability required for real-world, large-scale chip designs. By scripting SALSy, we emphasize its flexibility, making it adaptable to various design sizes and complexities, including those that contain memory or analog macros. Notably, handling macros introduces challenges as security engineers have less control over regions around macros due to high wire congestion. Hence, using the techniques like Location-based Buffering in those areas is challenging. However, this

TABLE V
COMPARISON OF THIS WORK (SALSY) WITH THE PREVIOUS TECHNIQUES

Ref.	Technique	Implications				Validated?
[14]	Internal Shielding	Power ☹	Timing ☹	Area ☹	Density ☺	✗
[23]	TroMUX	Power ☹	Timing ☹	Area 😐	Density ☺	✗
[24]	BISA	Power ☹	N/A	Area 😐	Density ☺	✗
[25]	Layout Filling	Power ☹	Timing ☹	Area 😐	Density ☺	✗
[26]	DEFense Framework	Power 😐	Timing ☹	Area 😐	Density ☺	✗
[27]	ASSURER	Power 😐	N/A	Area ☺	Density ☺	✗
[29]	T-TER	Power 😐	Timing 😐	Area 😐	Density 😐	✗
[28]	GDSII-Guard	Power ☹	Timing ☹	Area 😐	Density 😐	✗
	This Work (SALSy)	Power ☹	Timing ☺	Area ☺	Density ☺	✓

issue affects potential attackers as well given the inherent difficulty in inserting malicious logic (i.e.., Trojans) in such congested layout regions.

Additionally, standard deviation values for the leakage power consumption of both the baseline and secured versions of each block were obtained. The minimum values are 5.41×10^{-3} and 5.49×10^{-3} for the baseline and secured versions of the PRESENT benchmark, respectively, while the maximum values are 3.46×10^{-2} for the baseline and 3.39×10^{-2} for the secured versions of the CAST benchmark. These results highlight that the SALSy approach exhibits sensitivity to process variation comparable to that of the conventional security-unaware flow.

We have also compared SALSy against the most related prior arts to give the reader a better understanding of the key differences in power, timing, area, and density promoted by SALSy. As shown in Tab. V, our work is the only one that validated the presented technique in silicon. All other works only aim for security closure, and some of them suffer from the various issues we have highlighted related to the use of limited PDKs/libraries. The ☺ sign indicates improvement, the ☹ sign indicates deterioration of the introduced metrics, while the 😐 sign indicates that there are no considerable changes after applying the individual technique. N/A indicates the metric is not reported by the authors. It should be noted that we consider the increase in density as an improvement since it enhances the security of the design against the considered threats.

VII. CONCLUSION

In this paper, we have introduced SALSy, a design-time methodology to bolster the security of ICs against fabrication-time and post-fabrication attacks. Through a silicon demonstration, we successfully validated our solution, showcasing its fitness for use with a commercial PDK and cell library. In our pursuit of heightened security, our methodology strikes a prudent balance by incurring only a modest increase in power consumption. Although effective against Trojan insertion, there is still room for enhancing security against FSP/FI. Our future research will focus on automating the introduced techniques, including selective methods, and incorporating new approaches to bolster FSP/FI defense and overall security. Additionally, we contemplate introducing new evaluation metrics alongside the existing ones to provide a more realistic and comprehensive assessment.

REFERENCES

[1] C. Ting-Fang, "Tsmc to triple u.s. chip investment to $40bn to serve apple, others," 2022. [Online]. Available: https://asia.nikkei.com/Business/Tech/Semiconductors/TSMC-to-triple-U.S.-chip-investment-to-40bn-to-serve-Apple-others

[2] W. Hu, C.-H. Chang, A. Sengupta, S. Bhunia, R. Kastner, and H. Li, "An overview of hardware security and trust: Threats, countermeasures, and design tools," *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, vol. 40, no. 6, pp. 1010–1038, 2021.

[3] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proc. IEEE*, vol. 102, no. 8, pp. 1283–1295, 2014.

[4] S. Bhunia, M. Abramovici, D. Agrawal, P. Bradley, M. S. Hsiao, J. Plusquellic, and M. Tehranipoor, "Protection against hardware trojan attacks: Towards a comprehensive solution," *Des. Test*, pp. 6–17, 2013.

[5] V. Gohil, S. Patnaik, H. Guo, D. Kalathil, and J. J. Rajendran, "Deterrent: Detecting trojans using reinforcement learning," in *Proc. Des. Autom. Conf.*, 2022, pp. 697–702.

[6] C. Helfmeier, D. Nedospasov, C. Tarnovsky, J. S. Krissler, C. Boit, and J.-P. Seifert, "Breaking and entering through the silicon," in *Proc. Comp. Comm. Sec.*, 2013, p. 733–744.

[7] S. E. Quadir, J. Chen, D. Forte, N. Asadizanjani, S. Shahbazmohamadi, L. Wang, J. Chandy, and M. Tehranipoor, "A survey on chip to system reverse engineering," *J. Emerg. Technol. Comput. Syst.*, pp. 1–34, 2016.

[8] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware trojan attacks: Threat analysis and countermeasures," *Proc. IEEE*, pp. 1229–1247, 2014.

[9] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *Des. Test*, p. 10–25, 2010.

[10] N. G. Tsoutsos, C. Konstantinou, and M. Maniatakos, "Advanced techniques for designing stealthy hardware trojans," in *Proc. Des. Autom. Conf.*, 2014, pp. 1–4.

[11] A. Barenghi, L. Breveglieri, I. Koren, and D. Naccache, "Fault injection attacks on cryptographic devices: Theory, practice, and countermeasures," *Proc. IEEE*, pp. 3056–3076, 2012.

[12] M. Nagata, "Exploring fault injection attack resilience of secure ic chips : Invited paper," in *Proc. Int. Rel. Physics Sym.*, 2022, pp. 1–6.

[13] H. Wang, D. Forte, M. M. Tehranipoor, and Q. Shi, "Probing attacks on integrated circuits: Challenges and research opportunities," *Des. Test*, pp. 63–71, 2017.

[14] H. Wang, Q. Shi, A. Nahiyan, D. Forte, and M. M. Tehranipoor, "A physical design flow against front-side probing attacks by internal shielding," *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, pp. 2152–2165, 2020.

[15] M. Eslami, J. Knechtel, O. Sinanoglu, R. Karri, and S. Pagliarini, "Benchmarking advanced security closure of physical layouts: Ispd 2023 contest," in *Proc. Int. Symp. Phys. Des.*, 2023, p. 256–264.

[16] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, pp. 39–46, 2010.

[17] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware trojans: Lessons learned after one decade of research," *Trans. Des. Autom. Elec. Sys.*, pp. 1–23, 2016.

[18] A. Hepp, T. Perez, S. Pagliarini, and G. Sigl, "A pragmatic methodology for blind hardware trojan insertion in finalized layouts," in *Proc. Int. Conf. Comp.-Aided Des.*, 2022, pp. 1–9.

[19] T. D. Perez and S. Pagliarini, "Hardware trojan insertion in finalized layouts: From methodology to a silicon demonstration," *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, vol. 42, no. 7, pp. 2094–2107, 2023.

[20] B. Selmke, J. Heyszl, and G. Sigl, "Attack on a DFA protected AES by simultaneous laser fault injections," in *Proc. Worksh. Fault Diag. Tol. Cryptogr.*, 2016, pp. 36–46.

[21] R. A. C. Viera, P. Maurine, J.-M. Dutertre, and R. Possamai Bastos, "Simulation and experimental demonstration of the importance of ir-drops during laser fault injection," *Trans. Comp.-Aided Des. Integ. Circ. Sys.*, pp. 1231–1244, 2020.

[22] C. Boit, S. Tajik, P. Scholz, E. Amini, A. Beyreuther, H. Lohrke, and J. P. Seifert, "From IC debug to hardware security risk: The power of backside access and optical interaction," in *Proc. Int. Symp. Physical Failure Analys. IC*, 2016, pp. 365–369.

[23] F. Wang, Q. Wang, B. Fu, S. Jiang, X. Zhang, L. Alrahis, O. Sinanoglu, J. Knechtel, T.-Y. Ho, and E. F. Young, "Security closure of ic layouts against hardware trojans," in *Proc. Int. Symp. Phys. Des.*, 2023, p. 229–237.

[24] K. Xiao and M. Tehranipoor, "Bisa: Built-in self-authentication for preventing hardware trojan insertion," in *Proc. Int. Symp. Hardw.-Orient. Sec. Trust*, 2013, pp. 45–50.

[25] P.-S. Ba, S. Dupuis, M. Palanichamy, M.-L. Flottes, G. Di Natale, and B. Rouzeyre, "Hardware trust through layout filling: A hardware trojan prevention technique," in *Proc. Comp. Soc. Symp. VLSI*, 2016, pp. 254–259.

[26] J. Knechtel, J. Gopinath, J. Bhandari, M. Ashraf, H. Amrouch, S. Borkar, S.-K. Lim, O. Sinanoglu, and R. Karri, "Security closure of physical layouts iccad special session paper," in *Proc. Int. Conf. Comp.-Aided Des.*, 2021, pp. 1–9.

[27] G. Guo, H. You, Z. Tang, B. Li, C. Li, and X. Zhang, "Assurer: A ppa-friendly security closure framework for physical design," in *Proc. Asia South Pac. Des. Autom. Conf.*, 2023, p. 504–509.

[28] X. Wei, J. Zhang, and G. Luo, "Gdsii-guard: Eco anti-trojan optimization with exploratory timing-security trade-offs," in *2023 60th ACM/IEEE Design Automation Conference (DAC)*, 2023, pp. 1–6.

[29] T. Trippel, K. G. Shin, K. B. Bush, and M. Hicks, "T-ter: Defeating a2 trojans with targeted tamper-evident routing," in *Proc. of the 2023 ACM Asia Conf. on Comp. and Comm. Sec.*, 2023, p. 746–759.

[30] ——, "Icas: an extensible framework for estimating the susceptibility of ic layouts to additive trojans," in *Proc. Symp. Sec. Priv.*, 2020, pp. 1742–1759.

[31] J. Knechtel, J. Gopinath, M. Ashraf, J. Bhandari, O. Sinanoglu, and R. Karri, "Benchmarking security closure of physical layouts: ISPD 2022 contest," in *Proc. Int. Symp. Phys. Des.*, 2022, p. 221–228.

[32] H. Salmani and M. M. Tehranipoor, "Vulnerability analysis of a circuit layout to hardware trojan insertion," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1214–1225, 2016.

[33] T. Sugawara, N. Homma, T. Aoki, and A. Satoh, "Asic performance comparison for the iso standard block ciphers," in *JWIS*, 2007. [Online]. Available: http://www.aoki.ecei.tohoku.ac.jp/crypto/web/cores.html

[34] O. Girard et al., "openmsp430 at opencores.org," 2021. [Online]. Available: https://opencores.org/projects/openmsp430

[35] M. Hicks et al., "Mit-ll common evaluation platform (cep) at github.com," 2021. [Online]. Available: https://github.com/mit-ll/CEP

[36] "Nangate freepdk45 open cell library." [Online]. Available: http://www.nangate.com/?page_id=2325

[37] "Salsy repository." [Online]. Available: https://github.com/Centre-for-Hardware-Security/SALSy

[38] K. Yang, M. Hicks, Q. Dong, T. Austin, and D. Sylvester, "A2: Analog malicious hardware," in *Proc. Symp. Sec. Priv.*, 2016, pp. 18–37.

Mohammad Eslami received his M.S. degree in computer engineering from the Shahid Bahonar University of Kerman, Kerman, Iran, in 2018. Currently, he is pursuing his doctoral studies at the Centre for Hardware Security, Tallinn University of Technology (TalTech), Tallinn, Estonia.

His research interests primarily revolve around hardware security, with a particular focus on physical design automation and secure ASIC design.

Tiago D. Perez received the M.S. degree in electric engineering from the University of Campinas, São Paulo, Brazil, in 2019. He has also received the Ph.D. degree from Tallinn University of Technology (TalTech), Tallinn, Estonia.

From 2014 to 2019, he was a Digital Designer Engineer with Eldorado Research Institute, São Paulo, Brazil. His research interests include digital signal processing, telecommunication systems, IC design, and hardware security.

Samuel Pagliarini (M'14) received the PhD degree from Telecom ParisTech, Paris, France, in 2013.

He has held research positions with the University of Bristol, Bristol, UK, and with Carnegie Mellon University, Pittsburgh, PA, USA. He is currently a Professor of Hardware Security with Tallinn University of Technology (TalTech) in Tallinn, Estonia where he leads the Centre for Hardware Security. His current research interests include many facets of digital circuit design, with a focus on circuit reliability, dependability, and hardware trustworthiness.

Curriculum Vitae

1. Personal data

Name	Mohammad Eslami
Date and place of birth	02.02.1992, Shiraz, Iran
Nationality	Iranian

2. Contact information

Address	Tallinn University of Technology (TalTech), School of Information Technologies, Department of Computer Systems, Akadeemia tee 15a, 12618, Tallinn, Estonia
E-mail	mohammad.eslami.89@gmail.com, mohammad.eslami@taltech.ee

3. Education

2020–2024	Tallinn University of Technology, School of Information Technologies, Information and Communication Technology, Ph.D.
2015–2018	Shahid Bahonar University of Kerman, Iran Computer Engineering, MSc
2010–2014	University of Birjand, Iran Computer Engineering, BSc

4. Language competence

Persian	Native
English	Fluent
Arabic	Moderate
Estonian	Beginner

5. Professional employment

November 2020–September, 2024	Center for Hardware Security, TalTech, Estonia Early Stage Researcher
September, 2015–September, 2018	RESD Lab, SBUK, Iran Research Assistant

6. Computer skills

- Operating systems: Windows and Linux
- Document preparation: LaTeX, Microsoft Word
- Programming languages and scripting: C, C++, C#, TCL, Python, MATLAB
- Hardware Description Languages: Verilog, VHDL, SystemVerilog
- EDA Tools: Cadence Genus, Cadence Innovus, Cadence JasperGold suite, Siemes EDA Calibre, Cadence Xcelium Logic Simulator, Siemes EDA Questa Sim, Xilinx Vivado IDE, KiCAD EDA

7. Honours and awards

- HARNO: Awarded the ICT sector's highest recognition in the field of doctoral studies and a 5,000-euro scholarship (Link to the news: https://taltech.ee/en/news/taltech-phd-students-research-awarded-itl-ustus-agur-scholarship-focuses-hardware-security)
- HeLLoCTF'22: Won the 1st place in HeLLoCTF 2022 Contest and a 15,000-USD prize (Link to the web page: https://helloctf.org/22/winners)
- ISPD'22: Won the 3rd place in ISPD 2022 Contest: Security Closure of Physical Layouts and a 500-USD prize (Link to the webpage: https://wp.nyu.edu/ispd_22_contest/results)
- SBUK: Ranked 1st among the same entrance M.Sc. students of the Computer Engineering Department in a three-year period of 2015 to 2018
- SBUK: Introduced as a talented student by the Talented Students Guidance Office of Shahid Bahonar University of Kerman

8. Defended theses

- 2018, An Efficient FPGA-based Emulation Approach for Fault Simulation of Digital Circuits, M.Sc., Prof. Dr. Behnam Ghavami, Shahid Bahonar University of Kerman, Kerman, Iran
- 2014, Designing a High-frequency Phase Frequency Detector (PFD) using XOR Gates and H-Spice Simulation, B.Sc., Prof. Dr. Abolfazl Bijari, University of Birjand, Birjand, Iran

9. Field of research

- Hardware Security
- Application-Specific Integrated Circuit (ASIC) design
- Digital Circuits
- Electronic Design Automation (EDA)

Elulookirjeldus

1. Isikuandmed

Nimi	Mohammad Eslami
Sünniaeg ja -koht	02.02.1992, Shiraz, Iraan
Kodakondsus	Iraanlane

2. Kontaktandmed

Aadress	Tallinna Tehnikaülikool, Infotehnoloogia Teaduskond, Arvutisüsteemide Instituut, Akadeemia tee 15a, 12618, Tallinna, Eesti
E-post	mohammad.eslami.89@gmail.com, mohammad.eslami@taltech.ee

3. Haridus

2020–2024	Tallinna Tehnikaülikool, Infotehnoloogia Teaduskond, Informatsiooni-ja kommunikatsioonitehnoloogia, doktoriõpe
2015–2018	Shahid Bahonari ülikool Kermanis, Inseneriteaduskond, Arvutitehnika, magistrikraad
2010–2014	Birjandi Ülikool, Elektrotehnika ja Arvutitehnika Teaduskond, Arvutitehnika, bakalaureusekraad

4. Keelteoskus

Pärsia keel	emakeel
Inglise keel	kõrgtase
Araabia keel	mõõdukas
Eesti keel	algaja

5. Teenistuskäik

November 2020–september 2024	Riistvara turvalisuse keskus, TalTech, Eesti Doktorant-nooremteadur
September 2015–september 2018	RESD labor, SBUK, Iraan Teadusassistent

6. Arvuti oskused

- Operatsioonisüsteemid: Windows ja Linux
- Dokumendi ettevalmistamine: LaTeX, Microsoft Word
- Programmeerimiskeeled ja skriptimine: C, C++, C#, TCL, Python, MATLAB
- Riistvara kirjelduskeeled: Verilog, VHDL, SystemVerilog
- EDA Tööriistad: Cadence Genus, Cadence Innovus, Cadence JasperGold suite, Siemes EDA Calibre, Cadence Xcelium Logic Simulator, Siemes EDA Questa Sim, Xilinx Vivado IDE, KiCAD EDA

7. Autasud ja auhinnad

- HARNO: Auhinnati IKT sektori kõrgeima tunnustusega doktorantuuri valdkonnas ja 5000 euro suuruse stipendiumiga (Link: https://taltech.ee/uudised/itl-i-ustus-aguri-stipendiumi-palvinud-taltechi-doktorandi-uurimustoo-keskendub-riistvara)
- HeLLoCTF'22: Võitis esikoha HeLLoCTF 2022 võistlusel ja 15,000 USD suuruse auhinna (Link: https://helloctf.org/22/winners)
- ISPD'22: Võitis 3. koha ISPD 2022 võistlusel: Security Closure of Physical Layouts ja 500 USD suuruse auhinna (Link: https://wp.nyu.edu/ispd_22_contest/ results)
- SBUK: Hinnatud esimeseks arvutitehnika osakonna samade sisseastujate magistriõppe üliõpilaste seas perioodil 2015–2018
- SBUK: Shahid Bahonari ülikool Kermanis Andekate Üliõpilaste Juhendamisbüroo poolt andekaks tudengiks tituleeritud

8. Kaitstud lõputööd

- 2018, Tõhus FPGA-põhine emulatsioonimeetod digitaalsete vooluringide rikkesimulatsiooniks, M.Sc., Prof. Dr. Behnam Ghavami, Shahid Bahonari ülikool Kermanis, Kerman, Iraan
- 2014, Kõrgsagedusliku faasi-sageduse detektori (PFD) projekteerimine, kasutades XOR väravaid ja H-Spice simulatsiooni, B.Sc., Prof. Dr. Abolfazl Bijari, Birjandi Ülikool, Birjand, Iraan

9. Teadustöö põhisuunad

- Riistvara turvalisus
- Rakenduspõhise integraallülituse (ASIC) disain
- Digitaalsed vooluringid
- Elektroonilise disaini automatiseerimine (EDA)

ISSN 2585-6901 (PDF)
ISBN 978-9916-80-197-0 (PDF)