TAL TECH

DOCTORAL THESIS

Security-Aware Physical Synthesis of Integrated Circuits

Tiago D. Perez

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY
TALLINN 2023

TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
4/2023

Security-Aware Physical Synthesis of Integrated Circuits

TIAGO D. PEREZ

TAL TECH PRESS

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

The dissertation was accepted for the defence of the degree of Doctor of Philosophy in Information and Communication Technology on 12 December 2022

Supervisor: Professor Dr. Samuel Pagliarini,
Department of Computer Systems, Centre for Hardware Security,
Tallinn University of Technology
Tallinn, Estonia

Opponents: Professor Dr. Ronald D. Blanton,
Carnegie Mellon University,
Pittsburgh, United States

Dr. Marie-Lise Flottes,
Centre National de la Recherche Scientifique,
Montpellier, France

Defence of the thesis: 8 February 2023, 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.

Tiago D. Perez _____
signature

ISSN 2585–6898 (publication)
ISBN 978-9949-83-947-6 (publication)
ISSN 2585–6901 (PDF)
ISBN 978-9949-83-948-3 (PDF)
Printed by Auratrükk

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
4/2023

Integraallülituste turvateadlik füüsiline süntees

TIAGO D. PEREZ

TAL TECH
KIRJASTUS

Contents

List of Publications

The present PhD thesis is based on the following publications.

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

[II] T. Perez, M. Imran, P. Vaz, and S. Pagliarini, "Side-channel trojan insertion - a practical foundry-side attack via eco," in 2021 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5, 2021

[III] T. Perez and S. Pagliarini, "A side-channel hardware trojan in 65nm cmos with 2μW precision and multi-bit leakage capability," in 2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 9–10, 2022

[IV] T. D. Perez, M. M. Gonçalves, L. Gobatto, M. Brandalero, J. R. Azambuja, and S. Pagliarini, "G-gpu: A fully-automated generator of gpu-like asic accelerators," in 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 544–547, 2022

[V] A. Hepp, T. Perez, S. Pagliarini, and G. Sigl, "A pragmatic methodology for blind hardware trojan insertion in finalized layouts," in 2022 International Conference on Computer-Aided Design (ICCAD), 2022

[VI] 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 (TCAD), 2022

Other related publications

[VII] Z. U. Abideen, T. D. Perez, and S. Pagliarini, "From fpgas to obfuscated easics: Design and security trade-offs," in 2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST), pp. 1–4, 2021

Abbreviations

3PIP	Third Party Intellectual Property
AES_LFHD	AES Low-Frequency-High-Density
AES_HFHD	AES High-Frequency-High-Density
AI	Artificial Intelligence
APU	Accelerated Processing Units
ARPANET	Advanced Research Projects Network
ASIC	Application-Specific Integrated Circuit
BEOL	Back End of the Line
BioHT	Blind Insertion of Hardware Trojans
CAGR	Compound Annual Growth Rate
CCR	Correct Connection Rate
CMP	Chemical Mechanical Polarization
CPU	Central Processing Unit
CU	Computing Unit
DFM	Design for Manufacturability
DRC	Design Rule Checking
DSE	Design-Space Exploration
ECO	Engineering Change Order
EDA	Electronic Design Automation
eFPGA	Embedded Field Programmable Gate Array
EM	Electromagnetic
EMSR	Effective Mapped Set Ratio
FEOL	Front End of the Line
FIB	Focused Ion Beam
FPGA	Field Programmable Gate Array
GPU	Graphic Processing Unit
HD	Hamming Distance
HDL	Hardware Description Language
HPC	High-Performance Computing
HT	Hardware Trojan
IC	Integrated Circuits
IIFT	Imprecise Information Flow Tracking
IP	Intellectual Property
LEF	Library Exchange Format
LSI	Large-Scale Integration
LUT	Look-up Table
LVS	Layout Versus Schematic
MCTRL	General Memory Controller
ML	Machine Learning
MUX	Multiplexer
PPA	Performance, Power and Area
PCB	Printed Circuit Board
PDK	Process Design Kit

PE	Processing Elements
PST	Present Crypto Core
PST_LFHD	Present Low-Frequency-High-Density
PST_HFHD	Present High-Frequency-High-Density
RO	Ring Oscillator
RTL	Register-Transfer Level
SADP	Self-Aligned Double Patterning
SCT	Side-Channel Hardware Trojan
SDC	Synopsys Design Constraints
SEM	Scanning Electron Microscope
SIMT	Single-Instruction Multiple Threads
SNR	Signal-to-Noise Ratio
SoC	System-on-Chips
SSF	Signal Selection Function
TCO	Trojan Change Order
TLP	Thread-Level Parallelism
TPU	Tensor Processing Unit
TTE	Time to Evaluate
ULSI	Ultra-Large-Scale Integration
UPF	Unified Power Format
VLSI	Very Large-Scale Integration
WLO	Wirelength Overhead

1 Introduction

The digitalization of society has rapidly changed many aspects of our lives [1]. Today, semiconductors power almost everything in our daily activities [2]. Thus, many critical infrastructures deploy Integrated Circuits (ICs)-based systems. For example, even the financial sector experienced fast and deep digitalization in the past decades [3]. Moreover, in many parts of the world, governments are creating their digital form of currency [4]. Because IC-based systems are increasingly deployed in critical infrastructures, ensuring the trustworthiness of such devices is crucial. A compromised device that handles sensitive data or is essential for the functionality of a critical system can have devastating consequences. Therefore, guaranteeing the trustworthiness of ICs is vital. However, ensuring the IC's security is an open research question the community strives to solve.

All digitalization processes were possible because of the rapid development of ICs. Capable modern system-on-chips (SoCs) require powerful and efficient transistors coupled with optimized system architectures. A modern SoC architecture combines different computing units, often integrating a general-purpose processor (CPU), specific hardware accelerators, memories, and standard interfaces to connect everything. CPUs are very flexible, handling diverse types of workloads with satisfactory performance. Since its conception, CPU architectures have been optimized to increase the number of operations over time, and recently, they are also optimized for power efficiency. However, for some specific applications, CPUs performance is not sufficient. Thus, CPUs are integrated with hardware accelerators to run specific applications or parts of applications more efficiently. Examples of hardware accelerators are crypto cores for efficient encryption/decryption [5] and Graphics Processing Units (GPUs) [6–9] for handling massive parallel computations. Thus, combining CPUs with hardware accelerators achieves superior performance and efficiency, enabling applications previously considered infeasible due to the long execution time.

Designing and manufacturing a single modern IC requires a colossal amount of expertise among different fields of science [10]. In addition, developing and maintaining a high-end semiconductor manufacturing process is a costly endeavor. Reportedly, *Intel* is investing over 17 billion euros into a leading-edge semiconductor manufacturing facility in Germany [11]. Consequently, the conception of a modern hardware device is a collective effort shared between different entities. This characteristic makes the IC supply chain decentralized, complex, and highly globalized. Moreover, for modern hardware devices, the current organization of the IC supply chain is arguably a security threat. A heavily-debated example that illustrates the consequences of a compromised supply chain is the attack from Chinese spies that allegedly targeted almost 30 U.S. companies, including *Amazon* and *Apple*. According to [12], in 2015, an extra component was found in server motherboards that allowed the attackers to create a stealth doorway into any network that included a compromised machine. Those servers had been in use for a couple of years already, with *Apple* reportedly having almost 7000 running.

Ensuring the integrity of the technologies is crucial for protecting digital information and maintaining critical operational systems. The field of Cybersecurity was born in the 1970s with the project Advanced Research Projects Agency Network (ARPANET) [13]. Since then, the security field has advanced significantly. The focus of the security

community has been on the software domain, with hardware security as a secondary thought. However, over the last few years, there has been an exponential growth in hardware vulnerability exposure [14]. The development of patches to fix software vulnerabilities are almost always possible and done very quickly. Different from software, a vulnerability in hardware cannot be updated easily. Thus, attacks, as [12], are potentially more devastating than any other software-based attack. Nevertheless, an electronic system has to be secure from end to end, i.e., secure software running in secure hardware [15].

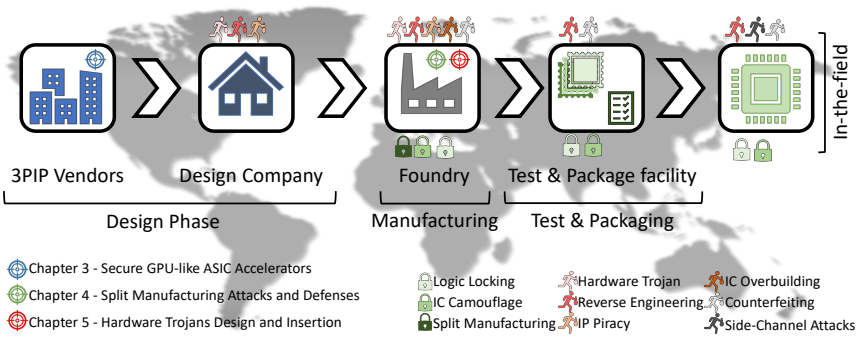

Figure 1: Detailed IC's life cycle phases, possible attacks, and defenses.

An overview of IC's life cycle is illustrated in Figure 1, showing the four phases of an IC's life; design, manufacturing, test & packaging, and field operation. Each phase has an associated set of potential hardware-based threats [16] as illustrated in Figure 1. During the design phase, an adversary can insert hardware trojans [17–20], reverse engineer, and pirate IPs. IC overbuilding, reverse engineering layouts, counterfeiting, and insertion of hardware trojans are associated with the manufacturing phase. A rogue element within the facility for packaging & testing can potentially reverse engineer or pirate the device. Side-channel attacks require access to the physical device; thus, the end-user can perform such attacks.

Hardware security techniques can be implemented in different phases of the IC's life cycle to enhance its security. Examples of these techniques are Split Manufacturing [21, 22], Logic Locking [23–29] and IC Camouflaging [30–32]. As illustrated in Figure 1, Split Manufacturing and IC Camouflaging can combat a series of manufacturing threats. Logic Locking and IC Camouflaging can prevent attacks during the test & packaging and when the chip is in the field by an end-user. Unfortunately, the current state of these techniques makes them unsuitable for large-scale production of ICs, either because of practicality [22] and/or insufficient security guarantees [33]. Without countermeasures, the described hardware-based threats are potential security hazards. Therefore, the emerging research topic of Hardware Security is striving to solve the IC security problem.

Hardware security is becoming an important field of research over the years. As a result, the field has been gaining more attention. Also, more groups of hardware security research have been created, and the topic's popularity in well-regarded conferences has increased. In addition, many specialized conferences dedicated to Hardware security

were created, such as CHES [34], the HOST series [35, 36], COSADE [37], and many others. This community's end goal is to ensure IC-based devices' trustworthiness. To achieve this goal, the community has been studying and exposing potential threats, creating countermeasures for known threats, and developing novel design techniques to enhance the IC's security.

The central theme of this thesis is the study of IC design techniques, either for enhancing IC security or exposing security flaws. First, I will propose a design technique to mitigate the presence of a possible hardware trojan and/or to be used as a fault tolerance technique (similar to triple modular redundancy). Following, I will discuss a countermeasure against threats during the manufacturing, called Split Manufacturing. Finally, I will demonstrate hardware trojan insertion step by step during the manufacturing phase. The threat model for this attack assumes the adversary only holds the victim's IC layout. Therefore, the adversary extracts all the information necessary for performing the attack from the layout.

1.1 Thesis Outline and Contributions

The present thesis comprises the published scientific articles in the List Of Publications section. This manuscript comprises six chapters and presents a study of physical synthesis for securing ICs. The chapters are: Background, three contribution chapters, Conclusion, and Future Work.

A summary of the material of each chapter is listed as follows.

Chapter 2 – Background: In this chapter, I present the essential concepts and theories of the contents of this thesis. The first topic is the semiconductor industry, where I present the evolution of the IC supply chain and the current practices of the semiconductor industry. Following, I introduce how ICs are designed, from the specifications to the finalized layout. After this introduction, I present the state of the art of hardware-based threats and countermeasures. Finally, the last topic covered is hardware accelerators, their architectures, and applications.

Chapter 3 – Secure GPU-like ASIC Accelerators: This chapter comprises the Publication *[IV]*. The contribution of Chapter 3 is a **secure GPU-like ASIC accelerator**. The literature review shows that the lack of an open-source GPU architecture for ASIC is a research gap. The FGPU, a GPU architecture for FPGA platforms, is among the few GPU architectures available. Utilizing the FGPU architecture as the baseline, I translated it to target an ASIC platform. I optimized the architecture to improve its performance, achieving operating frequencies ten times faster than its FPGA counterpart. After tweaking the architecture, I improved the security of the architecture by creating distinct power domains for each computing unit (CU) of the GPU. This feature enables the possibility of choosing which CU the user wants to turn on, and the others can be fully shut down. The result of this study is a fully-automated tool for generating GPU-like accelerators for ASIC. My tool permits the user to modulate the GPU regarding the number of CUs and which one will have its own power domain. The result of this tool is the layout of a GPU ready for being manufactured – this GPU is termed G-GPU.

Chapter 4 – Split Manufacturing Attacks and Defenses: This chapter comprises the Publication *[I]*. The contribution of Chapter 4 is **the first survey** on Split Manufacturing. From a literature review, I identified that the Split Manufacturing technique research was mature and relevant for having a survey. On top of that, I addressed a controversial topic among the recent publications on Split Manufacturing. In this survey, I comprehensively classified every attack against split layouts and every defense technique for enhancing even further split layouts security. In addition, a thorough discussion is presented about the strong and weak points of the current Split Manufacturing state of the art. I argue that this survey is very important for future research on Split Manufacturing, being a focal point to start from for security experts interested in the topic.

Chapter 5 – Hardware Trojans Design and Insertion: This chapter comprises Publications *[II], [III], [V], and, [VI]*. The contribution of Chapter 5 is a full framework for designing and inserting hardware trojans in finalized layouts. This framework is **the first to disclose step by step** how to perform hardware trojan inserting during a fabrication-time attack, where the attacker only holds the victim's layout. To validate this framework, I developed a silicon prototype comprising four crypto cores altered with a side-channel trojan. This work started with developing a technique for modifying a finalized layout. For that, I leveraged a feature called engineering change order (ECO). Using ECO, I modified finalized layouts with additional malicious logic. This is **the first demonstration** of hardware trojan insertion utilizing ECO. Furthermore, I designed a side-channel trojan capable of leaking multiple bits into a single power signature reading to demonstrate the proposed ECO framework's capabilities. The first version of the ECO framework has a deficiency. Critical nodes for connecting the hardware trojans must be located by visually inspecting the layout. Reverse engineering techniques are utilized to address the ECO framework limitation, adding the capability of inserting hardware trojans **totally blindly**. In addition, the framework is also improved by making the insertion iterative and faster.

Chapter 6 – Conclusion and Future Work: In this final chapter, I summarize all the results from the contribution chapters. The final conclusion is drawn, and a list of possible directions for future work is presented.

2 Background

2.1 History and Today's Integrated Circuit

After the invention of the first transistor in 1947, the semiconductor industry experienced rapid growth. In 1961, the first integrated circuit patent was awarded, marking the dawn of the era of IC-based devices [38]. As ICs started to be widely adopted in various electronic appliances, semiconductor supply companies started to invest in developing the IC, engaging in fierce technological and price competition [39]. The advances done by these companies developed the so-called large-scale integration (LSI) era, where a single chip contains hundreds of transistors. From there, the development reached a very large-scale integration (VLSI) phase with chips containing from 100 thousand to 10 million transistors, and finally, the ultra-large-scale integration (ULSI) with more than 10 million transistors per chip. Currently, the chase toward high performance and multiple functions continues. The largest commercial IC available in 2022 is the *M1 Ultra* commercialized by *Apple*. This IC has a transistor count of 114 billion while featuring a dual die in a single package manufactured in a 5nm FinFET technology.

As the IC evolved, the semiconductor supply chain also changed with it. During the 1980s, Japan dominated the semiconductor market since it provided better yield and products at that time [40]. Japanese businesses were fully integrated, vertical conglomerates, managing everything from manufacturing their chips to building their own devices and even global and local distribution of their products. In the 1990s, the emerging economies of Korea and Taiwan started to dominate the semiconductor market. Investing heavily solely in the manufacturing process, with records of frequently spending 100% of their revenue on capital expenditure (i.e., re-investing back in their own company) [41]. Thus, industries with advanced and mature manufacturing processes began to implement the service of only manufacturing semiconductors (i.e., pure-play foundry) [42]. Because these pure-play foundries had, and still have, the best transistors in the market, the IC supply chain experienced a shift to a horizontal system, making this chain decentralized and much more complex. Current semiconductor industry practices are primarily horizontal, where design houses are "fabless" and rely on pure-play foundries to manufacture their designs.

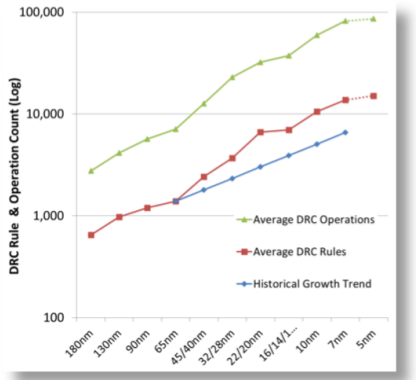

Figure 2: Growth of design rules from CMOS 180nm until finFET 5nm (from [43]).

The pursuit of denser and faster ICs sharply increased the complexity of manufacturing. This increasing complexity drives the need for more powerful electronic design automation (EDA) tools, related IP libraries, and new implementation strategies such as special packaging, stacked die technologies, and other assembly techniques [44]. The number of design rules and the total number of manufacturing steps can represent the increasing complexity of conceiving an IC. Advanced nodes experienced exponential growth of design rules [43], and the exponential jump in the number of rules during the design rule checking (DRC) with the evolution of the nodes is illustrated in Figure 2. The same trend happens for the number of manufacturing steps [45], depicted in Figure 3. To put all these in perspective, a design company needs access to an operational manufacturing process, a capable EDA tool vendor, and a specialized IP provider for manufacturing a modern complex chip. Even companies that control the manufacturing process, such as *Intel*, requires help from other entities to develop their products [46].

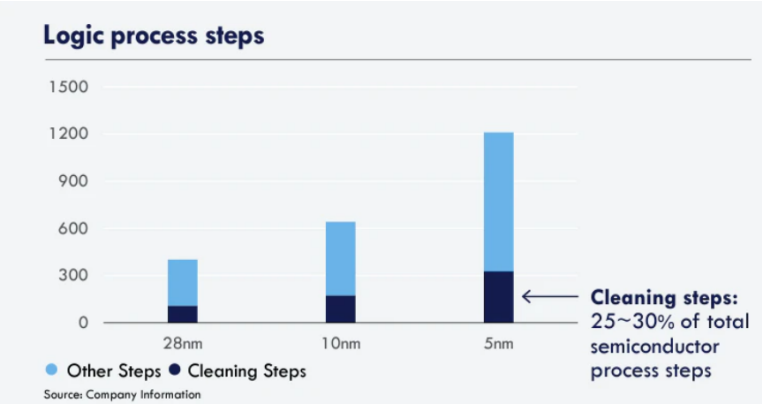

Figure 3: Logic manufacturing process steps comparison between CMOS 28nm, FinFET 10nm, and, FinFET 5nm, technology nodes (from [45]).

Currently, almost all design companies operate as described in Figure 1. First, some blocks are developed in-house during the design, while some parts of the design are IP bought from 3PIP vendors. Then, the finalized layout instantiates IPs provided by a third party, and the design company can develop some parts in-house. Finally, most design companies have to utilize pure-play foundries for manufacturing. Foundries can package and test the chips; however, in many cases, the bare dies are sent to another specialized facility for that process. In addition, to compete in the current market, companies must use EDA tools to produce a modern functional chip. Hence, this brief description of the process of producing an IC shows how the IC supply chain is decentralized, complex, and globalized. As Figure 4 illustrates, the number of high-end foundries has been steadily declining over the past decades. Today, only three companies can manufacture at advanced nodes, and that number is expected to shrink to only two in the future.

In 2020, the semiconductor industry experienced a rapid surge in demand for chips. The leading cause of this increased demand was the epidemic caused by the spread of the COVID-19 virus [48]. Because of the small number of capable facilities to manufacture

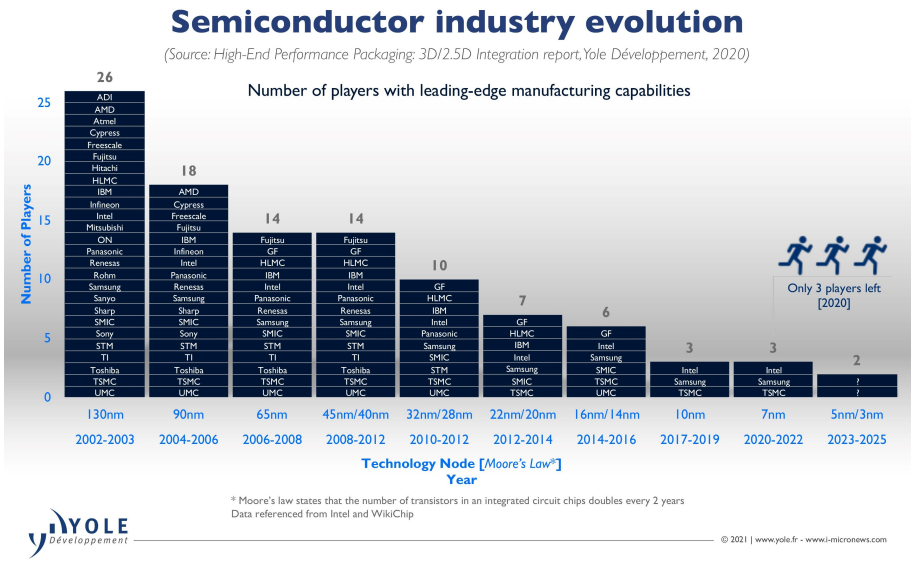

Figure 4: Semiconductor industry evolution (from [47]).

modern ICs (see Figure 4), the market is experiencing a shortage of chips [49, 50]. According to a survey conducted by the European Commission [51], 83.3% of the respondents were directly affected, and 16.7% were indirectly affected. Moreover, most companies interviewed expected the shortage to last until 2024. This shortage portrays how difficult it is to restructure the semiconductor supply chain and how vital chip manufacturing is for the global market [52].

2.2 Integrated Circuit Digital Design Implementation

The complexity of building an advanced IC is very high, requiring hundreds of steps (see Figure 3). Manufacturing processes build the IC from the bottom to the top layer. Those layers can be seen in the cross-section of an IC in Figure 5. At the bottom is the front end of the line (FEOL) layer containing all the transistors. On the top is the back end of the line (BEOL) layer composed of all the metals. The metal layers are referred to as MX, where X is the level of the layer. Metals are interconnected by vias, referred to as VX, following the same naming scheme for metals. Foundries often provide different metal stacks for each technology, differing in the number of metal layers and/or the properties of some of the metals. From a designer's perspective, the number of metal layers available represents the routing resources. For example, a metal stack containing more metals can route a design easier, of course, if compared with another metal stack from the same technology. Nevertheless, cost and technical limitations limit the scalability of the metal stack. In some cases, a small number of metal layers is more than enough for routing the design, reducing the overall cost of the chip.

SoCs can integrate analog circuits, digital logic, and memories in one single chip. The design of analog circuits for ICs is a full-custom design because the designer must

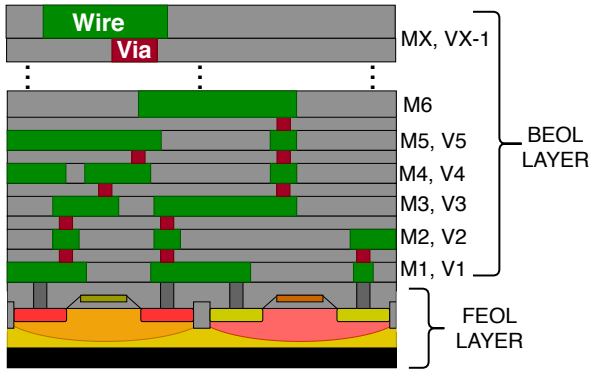

Figure 5: Cross section of an Integrated Circuit (from [22]).

define all layers of the device, i.e., FEOL and BEOL. Hence, the designer can benefit from complete control for optimizing the circuit but trading-off design time. On the other hand, the digital logic implementation utilizes the notion of standard cells. Those cells have standardized sizes regarding their height; thus, they can be placed in rows side-by-side. Nonetheless, placing the cells in rows facilitates the overall placement and the power distribution strategy. Each of those cells, or gates, is either a flip-flop register for storing bits, a buffer, an inverter, or performs a unique logic operation (e.g., AND, OR, XOR). Foundries and specific vendors provide standard-cell IP libraries fully characterized in terms of process variation, voltage, and temperature. Utilizing standard cells, the designer must only define the position of the gates and the metal layers. Thus, the gates already have the FEOL defined, and the designer only must define the BEOL.

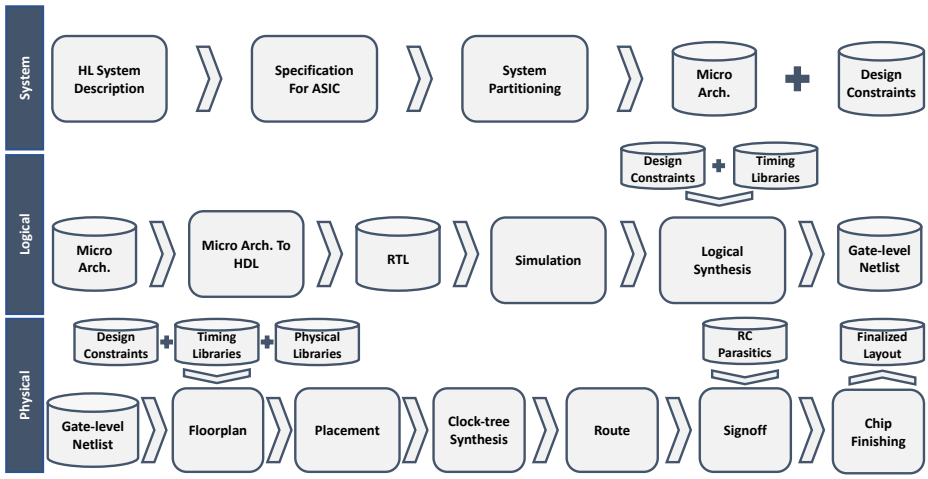

Figure 6: Typical design flow for digital integrated circuits.

Contrary to analog designs with a minimal number of transistors, a single digital sub-block of a modern application-specific integrated circuit (ASIC) can have more than

a million gates[1]. Thus, combining the usage of standard cells with powerful EDA tools for automation becomes a necessity for enabling the implementation of digital designs. Next, a brief introduction of how to perform a typical digital design implementation for ICs is shown.

Implementing a digital design can be separated into three phases: system, logical, and physical. This process results in a layout of all layers (FEOL+BEOL) that the foundries utilize as a blueprint for manufacturing the IC, typically handled in GDSII format. Figure 6 illustrates a diagram flow of this process in detail.

The designer must define a high-level description of the system and a set of constraints that defines the initial specification of the system. Usually, larger systems are partitioned into small microarchitectural blocks, making the implementation more time-efficient. Hence, many engineers can work in parallel, speeding up the implementation process. Generally, companies acquire IP for some microarchitectures of their system or commission its design to other vendors. Later, an SoC integrator connects the blocks back into a single system. This strategy is depicted in Figure 7. A set of design constraints for the specification phase is an estimation of the desired performance, power consumption, and area (PPA). The performance combines the operating frequency, throughput, and delay for generating a valid output.

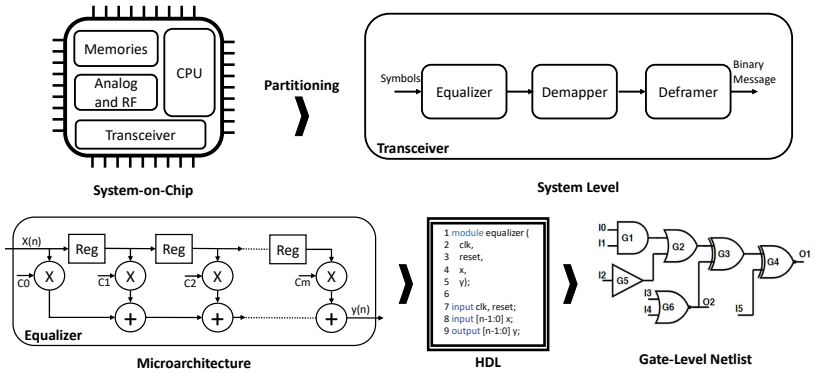

Figure 7: Abstraction levels of a digital system.

Estimating the power consumption of an IC is complex. Total IC power consumption is divided into static and dynamic components. Leakage power is the static component of power and depends mainly on the threshold voltage of the transistors. On the other hand, dynamic power depends on the circuit's activity. Dynamic power is also divided into internal and switching components. Switching power is the driving of output loads, dissipated when internal cell and wires capacitors are charged and discharged. When a cell switch states, an instantaneous short-circuit connection between the core supply voltage and the ground occurs momentarily; during this moment, internal power is dissipated. Therefore, in estimating the power consumption during the specification phase, the designer must reasonably estimate the design's number of gates and operating frequency. The same is true for estimating the total area because it is primarily a

[1]A gate, or standard-cell, contains more than one transistor. Typically, the smallest gate is an inverter with a minimum of two transistors, depending on the IP library and technology.

function of the number of gates and other secondary factors such as density (area populated with gates versus empty space), aspect ratio, and pinout position. Hence, the set of design constraints from the specification phase might not be feasible for implementation. Through the implementation steps, PPA figures are increasingly more accurate to report. Accordingly, often the specifications are adjusted after the logical and physical synthesis.

After the system phase, the design is sequentially represented in three different abstraction levels: register-transfer level (RTL), gate level, and layout. First is the RTL, where the logic behavior of the microarchitecture is described utilizing a hardware description language (HDL), such as VHDL, Verilog, or System-Verilog. RTL is a precise and formal description that allows the automation of digital circuits' simulation. During this phase, the designer's responsibility is to ensure the circuit behaves as expected in terms of functionality and latency (clock cycles to produce a valid output).

After behaviorally checking the RTL, the next abstraction level is the gate level. Generating the gate-level netlist requires a standard-cell IP library. Thus, for this phase, the designer must have decided with which technology the IC will be manufactured. The process of generating the gate-level netlist is called logical synthesis. Inputs required for the synthesis are the RTL, standard-cell timing library, and design constraints. As mentioned, foundries and vendors characterize each gate regarding process variation, voltage, and temperature. These characteristics are usually compiled in a standard Liberty format. Liberty files contain all available gates and their characteristics. Characteristics include logical function, pinout, delay, transition time, input capacitance, dynamic power, leakage power, setup time, hold time, and many more [53]. In addition to the timing library, the designer must set the design constraints, utilizing the Synopsys Design Constraints (SDC) format. The SDC file is where all clocks, input delay, output delay, and many other parameters can be described and constrained.

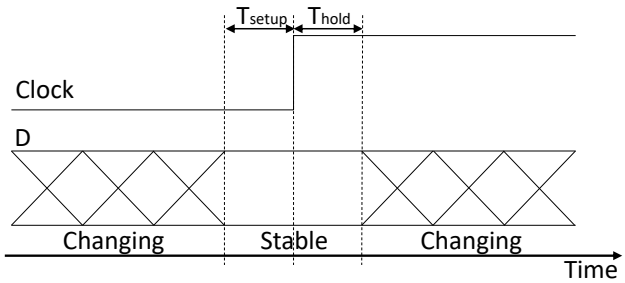

Figure 8: Setup and hold time.

The logical synthesis aims to translate the RTL into logical gates and achieve the performance set in the design constraints. For sequential logic, the circuit works under a set operating frequency where data is stored in registers each clock cycle. Data must travel from register to register in a time under a clock period T_{period}. Thus, logical synthesis tools must analyze setup timing to guarantee that the circuit will operate at the set frequency. Furthermore, during the physical synthesis, hold time is analyzed. Setup and hold time characteristics define when the data must stay stable at the register D pin, as illustrated in Figure 8. For checking for timing violations, the EDA tools

measure the time between each register, called path delay, and calculate the timing slack for setup and hold, as illustrated in Figure 9. Following the example in Figure 9, paths delay are timed as:

1 Data is launched from Reg1/D at the positive t0 clock edge at Reg1/C, requiring $T_{ck->q}$ time units

2 Data travels from Reg1/Q through a combinational logic to Reg2/D, requiring T_{prop} time units

3 Data is captured at Reg2/D at the positive t1 clock edge at Reg2/C. The data must be stable T_{setup} time units before this clock edge and T_{hold} after.

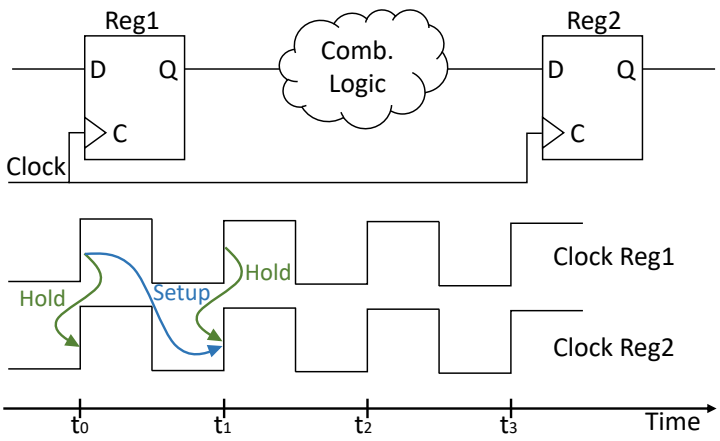

Figure 9: Timing path calculation example.

Then, timing setup analysis at Reg2 is done by checking the stable time before t1 positive clock edge at Reg2/C, i.e., the time slack described by Equation 1. Finally, hold analysis at Reg2 is done by checking the stable time after t1 positive clock edge at Reg2/C, i.e., the time slack described by Equation 2. Note that hold does not depend on the clock period, only setup. EDA tools consider a time setup and hold slack equal to zero as a non-violating timing path (i.e., the circuit can operate without a timing problem). However, typically designers choose a margin of a few picoseconds for both setup and hold slack.

$$\text{Setup slack} = T_{ck->q} + T_{prop} + T_{setup} - T_{period} \tag{1}$$

$$\text{Hold slack} = T_{ck->q} + T_{prop} - T_{hold} \tag{2}$$

Therefore, the designer must analyze the timing after logical synthesis to ensure the slack is within the desired margin or at least positive. If the slack for setup or/and hold is negative, the design has a timing violation and will not function correctly. Fixing timing violations in this phase is done by redefining the design constraints, changing the design architecture for inserting additional pipeline stages, or performing resynthesis/retiming.

In addition, from the gate-level netlist, it is possible to analyze power and area to contrast them with the specifications. Power and area from the gate-level netlist are representative but not accurate enough. The physical synthesis can, in some cases, change these figures drastically.

Finally, the final phase is the physical synthesis to generate the design layout. The layout level of abstraction now requires physical information about the standard cells and metal stack. For digital circuits, the physical synthesis treats each gate as a "black box", i.e., internal details of the transistor level are not required. However, essential information and design rules are required, such as box dimension, pinout position, metal layer, obstruction layer, and orientation. In addition, the EDA tool also must know how to handle the metal layers, e.g., the number of metals, the allowed width of each metal, and the type of vias. Library Exchange Format (LEF) file is the preferred format to describe the physical characteristics of each available gate and the metal stack. Then, inputs for the physical synthesis are the gate-level netlist, timing libraries, design constraints, and LEF files for the gates and the technology LEF file.

The whole process of physical synthesis is very complex, comprising many steps. For the sake of simplicity, the following synthesis explanation is divided into six steps: floorplanning, placement, clock-tree synthesis, routing, signing off, and chip finishing. Also, the following explanation focuses on block implementation. Managing a top-level layout requires many specific steps and decisions that are not covered in this thesis.

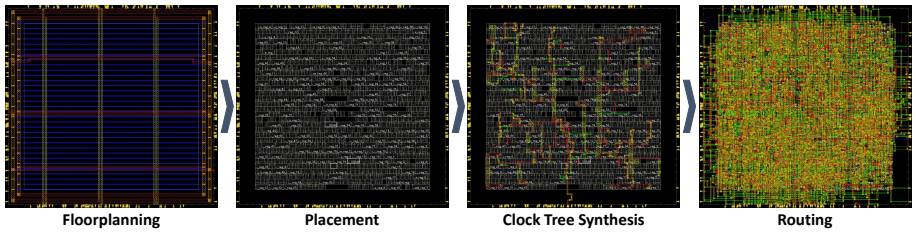

Figure 10: Block design implementation steps; floorplanning, placement, clock-tree synthesis, and, routing.

For block implementation, floorplanning is sizing the block box for a target density, defining the pinout, and power distribution implementation. Setting density is very accurate at this phase because all required gates are in the netlist. Nevertheless, the density difference between floorplanning and the finalized layout may slightly differ. The difference is due to added buffers during the clock-tree synthesis, timing optimization, and the resizing of cells' drive strength. Illustrated in the first panel of Figure 10 is an example of a block floorplan. Figure 10 shows the upper metal stripes for power distribution highlighted in yellow and orange, the bottom metal stripes in blue, and the yellow arrows represent the pinout of the block.

The next step after the floorplanning is the placement. In general, running the placement requires a single command in a commercial EDA tool, such as Innovus from Cadence [54]. Nonetheless, the designer can control many parameters of the placement. The placement algorithm is not only for placing the gates coherently with their interconnections but also is time aware. Therefore, gates are placed in such a

way as to achieve the best setup/hold timing slack. On top of that, modern tools also do a trial route for estimating routing congestion. Consequently, timing can be analyzed more accurately after the placement than in logical synthesis. Furthermore, the trial route provides a good amount of information to check if the design is routable. Illustrated in the second panel of Figure 10 is an example of block placement. Note that the power grid stripes and the trial route are hidden.

Before the clock-tree synthesis, timing analysis does not consider the clock skew, i.e., the clock distribution is ideal and reaches each register simultaneously. However, realistically the clock signal will never reach registers simultaneously; a skew between all clock inputs will always exist. Thus, to balance the clock delay for all clock inputs, the clock-tree synthesis inserts buffers/inverters in the clock routing. Illustrated in the third panel of Figure 10 is an example of a clock tree. After this synthesis, the clock is propagated, considering the expected delay between all clock inputs, making the timing analysis more realistic. With the propagated clock, timing analysis includes the skew between the launch and capture clock when timing the paths. Modern EDA tools can leverage the clock skew to improve performance, a technique called useful skew. For more details on useful skew and other timing optimization techniques, such as borrowing time, I direct the reader to [55, 56].

With all the gates placed and the clock tree routed, the next step is to route all the interconnected gates. Routing a design is to draw wires between all drivers and sinks. Nevertheless, depending on the amount of routing resources, design rules, and congestion, routing can be very challenging and take several hours, or even days, to complete. Moreover, a challenging routing may fail mainly because the tool can only find how to route by violating design rules. In some cases, post-route optimizations can fix the routing if the number of design rule violations is reasonably low. However, if the post-route optimizations cannot fix design rule violations due to the routing, the physical synthesis process must restart from the floorplanning. Then from the floorplanning, the block box can be resized, the pinout repositioned, the power grid adjusted, or all three to make the design routable. Illustrated in the fourth panel of Figure 10 is an example of routing without any design rule violation.

Before the routing, tools calculate the RC parasitics using an estimated wire length. Therefore, all effects considered due to parasitics are estimated. With the design routed, the EDA tools can extract the RC characteristics of all wires with a high degree of accuracy; this is called RC parasitics extraction. Then, a signing-off phase is necessary to consider the RC parasitics information for analyzing the timing. For timing analysis, the more accurate RC information changes the load of the pins for all cells. Pin load affects the speed of the cells; hence, signing-off timing analysis has a more accurate T_{ck-q} and T_{prop} times. Depending on the level of route congestion, the wire's RC parasitic (especially coupled capacitance) could heavily impact the performance. Modern EDA tools can fix timing violations during the signing-off to a certain degree, and even specialized tools for this purpose are available (e.g., Tempus from Cadence). The signing-off phase also includes the analysis of signal integrity and power integrity. For more detail on these analyses, I direct the reader to [57].

Finally, the last phase is chip finishing. For block implementation, chip finishing includes physical verification and layout versus schematic (LVS) checking. Physical

verification is the design rule check (see Figure 2) to make sure that all metal layers (BEOL) defined in the layout are compliant with the design rules. LVS compares the extracted netlist from the layout to the original schematic netlist to check if all devices in the layout match the schematic.

A block layout is considered tapeout-ready if it has no timing violations, no DRC violations, and LVS matches. Nonetheless, EDA vendors also provide additional solutions for logical equivalence checks, structural analysis, timing constraint verification, design for test, and many others. However, these tools do not take into account any security aspect. Either for ensuring security or for checking for potential vulnerabilities. Thus, most companies' implementation flow of digital ICs is oblivious to hardware security.

2.3 Hardware-based Threats and Countermeasures

As electronic systems are increasingly deployed in critical infrastructure, counterfeit and maliciously modified ICs have become a significant concern [58]. Assessing the trustworthiness of the design and manufacturing of ICs has become more challenging over the years [59]. As discussed in Section 2.1, the primary factor for this problem is the decentralization and globalization of the IC supply chain. It is conceivable – if not likely – that a fault in a low-quality counterfeit IC (or even a maliciously modified IC) will effectively disrupt critical infrastructure with dire consequences. Therefore, hardware security has gained more attention in the past decades, emerging as a relevant research topic.

An IC passes through many different entities during its lifecycle (see Figure 1). Thus, establishing trust between all involved parties is very difficult in practice. During the design phase, as shown in Section 2.1, some blocks are in-house developed, some are third-party IPs, and others are commissioned to be developed in a third-party design house. Physical libraries are also a mix of in-house developed and third-party provided for generating the layout. Finally, this layout is sent to the foundry to be manufactured. After manufacturing, the chips are sent to another facility for testing. The testing process searches for any physical defects and verifies the packaged parts to check if the functionality and performance are under the specification. Outsourcing manufacturing and testing to offshore companies are current practices for almost all design companies, with a few exceptions as *Intel*. Thus, sensitive information is almost inevitably exposed to untrusted parties to produce an IC. It is noteworthy that any outsider entity/company is considered untrusted for security.

Today's reality is that ICs are vulnerable to many hardware-based threats, including the insertion of hardware trojans, IP piracy, IC overbuilding, reverse engineering, side-channel attacks, and counterfeiting. Figure 11 presents a systematic classification of these threats, their goal, and the location where they occur [16].

In particular, hardware trojans (HTs) are malicious modifications to an IC, where attackers insert circuitry (or modify the existing logic) for their own malicious purposes [17–20, 60–71]. This attack is (typically) mounted during manufacturing, as the foundry holds the entire layout and can identify critical locations for trojan insertion. Third-party IPs can also contain trojans/backdoors that may contain hidden functionalities and can be used to access restricted parts of the design and/or expose data that

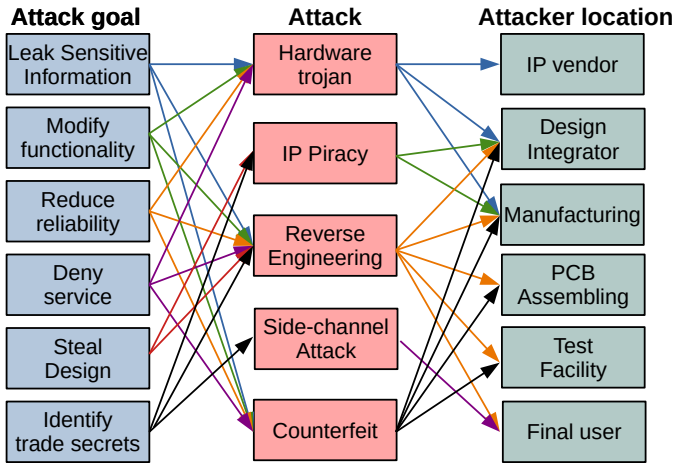

Figure 11: Systematization of hardware security around the attack method (adapted from [16])

would otherwise be unknown to the adversary. HTs are designed to leak confidential information, disrupt a system's specific functionality [72], or even destroy the entire system [73] and have a broad taxonomy [74].

Due to the vast ways an adversary can modify an IC for implementing HTs, they are classified as an additive, parametric, and subtractive. As the name suggests, an additive HT inserts extra malicious logic into the circuit. Contrariwise, subtractive HTs remove part of the existing logic. On the other hand, parametric HTs are very different from the other types. This family of trojans changes the IC layout's parametric characteristics, either the geometry of wires and transistors or the dopant polarity of a few transistors [63]. Thus, parametric HTs add no extra logic resulting in zero overhead of additional transistors and wires. From this point forward, I will focus mainly on additive HT. Additive HT is the most extensive type of HT studied in the literature and is the target of the proposed HT architecture in Chapter 5.

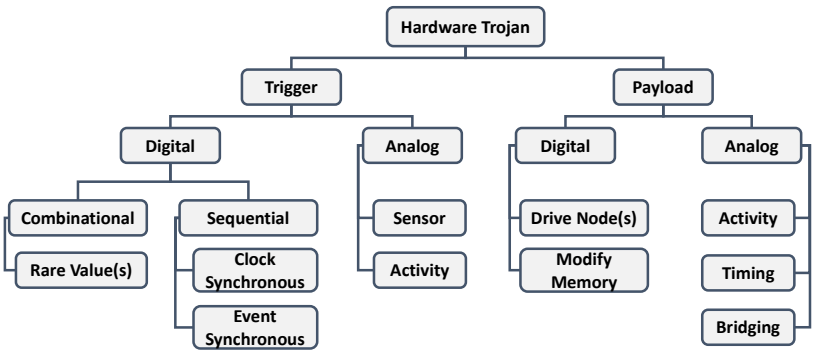

Figure 12: Additive hardware trojan taxonomy based on trigger *and* payload *implementation types (adapted from [61]).*

An HT architecture comprises a payload that implements the malicious behavior and a trigger that activates the HT when a specific condition is met. According to the

authors in [61], the payload and trigger of an additive HT are classified as shown in Figure 12. The payload and trigger components can be either digital [19] or analog [60] and can be realized in diverse manners. An HT trigger is qualified by its stealthiness and contractability. Then, the ideal trigger is activated when dozens of infrequent events occur, increasing the HT's stealthiness. A highly controllable HT can easily deploy the attack, but only by the adversary and not through regular use. As mentioned, an HT's payload can be designed with various effects as described in Figure 12.

As HT modifies the existing circuit, if the modification is apparent, one supposedly could identify the presence of an HT on an IC. However, since ICs are inherently opaque devices, inspecting their internal components is not a trivial task. Therefore, detecting HTs of any type is usually a problematic task [75]. Moreover, by design, HTs are triggered under specific conditions, making them unlikely to be activated and detected when the circuit operates as intended or when random stimuli are applied [73].

Nevertheless, many techniques for detecting the presence of an HT were proposed [76–85]. These detection techniques are either invasive or non-invasive. Invasive methods aim to retrieve information about the IC's internal components. They are usually performed by delaminating the IC to reconstruct the layout layers [84], leading to the destruction of the inspected sample. However, reconstructing the layout layers is time-consuming and requires precise equipment.

On the other hand, non-invasive techniques leverage the IC's physical characteristics and/or IO signal behavior (i.e., timing and state) [73]. For example, a few proposed techniques use the notion of path delay fingerprint to assess if the circuit was modified [76–78]. These techniques will likely detect the HTs that disrupt the circuit's data path. Another class of techniques utilizes power consumption metrics (leakage and total power) for detecting HTs [79, 80]. These techniques will spot the trojan if the HT heavily modifies the chip's power consumption. Chapter 5 presents a more detailed discussion of additive hardware trojans, their insertion, and detection.

IP piracy and IC overbuilding are illegal ownership claims of different degrees. As said before, designing an IC requires third-party and in-house developed IPs to complete the design. Design companies can overuse and copy third-party IPs without the owner's authorization. Similarly, malicious foundries can manufacture a surplus of ICs (overbuilding) without the owner's knowledge and sell these parts on the grey market.

Reversing engineering of ICs has been extensively demonstrated in the specialized literature [84, 86–88]. An attacker can identify the technology node and underlying components (memory, analog, and standard cells), from which he/she can extract a gate-level netlist, and even a high-level abstraction can be inferred [89]. Reverse engineering can be effortlessly executed during manufacturing, as the foundry holds the entire layout and most likely promptly recognizes some of the IP. Moreover, specialized high-level functionality reconstruction tools can recover the purpose of signals. For example, those tools can distinguish control from data paths of a finite-state machine from a target design [88]. In [19], the authors leveraged such tools' output to automate the search of security-critical nodes. In [87], the authors proposed a similar reverse-engineering technique to recover the coefficients of an obfuscated FIR filter.

After manufacturing, – when ICs are already packaged and deployed – reverse engineering is more laborious but can still be executed by a knowledgeable adversary.

Similar to inspecting an IC's internal components, an adversary can delaminate the chip in order to retrieve the layout layers. Reconstructing the layout layers from a physical sample is divided into three steps: depacking, delayering and imaging, and image post-processing. The chip must first be depacked by wet-chemical or mechanical means to access the die. Then, after recovering the bare die, the IC has to be delayered, and each layer has to be optically captured using a scanning electron microscope (SEM) or focused ion beam (FIB). Finally, the digitalized layer images have to be stitched and vectorized to retrieve the layout representation of the chip. Note that this process is yet to be fully automated [84], resulting in a highly time-intensive task prone to errors.

An IC's operating physical characteristics, such as timing, power consumption, electromagnetic radiation, and even sound, can be used as a side channel to indirectly reveal information that should be internal to the IC. Hence, malicious elements can exploit such a side channel to leak secret information from inside an IC. Since side-channel attacks can leak data from privileged parts of a system without permission, the most sought-after targets for side-channel attacks are embedded crypto cores. Many authors have already demonstrated that side-channel attacks can break the most important cryptographic algorithms in use today [90, 91].

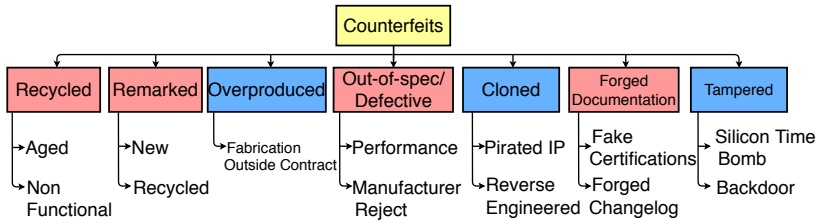

Figure 13: Taxonomy of counterfeit electronics (adapted from [59]).

According to [59], counterfeit components are classified into seven distinct categories, as illustrated in Figure 13. Recycled, remarked, out-of-spec/defective, and forged documentation are inherent after-market problems where products are offered by parties other than the original component manufacturer or authorized vendors. These cases are highlighted in red. On the other hand, overproducing, cloning, and tampering are problems faced during the design and/or fabrication of ICs. These cases are highlighted in blue. It is important to realize that these threats, including hardware trojans, could be avoided if a trusted manufacturing scheme was in place. For example, the old Japanese semiconductor business model from the 1980s discussed in Section 2.1 most likely did not face any of the threats highlighted in blue in Figure 13. However, the escalating cost and complexity of semiconductor manufacturing on advanced technologies made owning an advanced foundry unfeasible for design companies, which now tend to adopt the fabless business model [42].

Governments recognized access to advanced ICs as necessary for their domestic economy and national security. Currently, the US and Europe are making an effort to manufacture advanced semiconductors inside their borders [92, 93]. Access to a domestic manufacturing process arguably could mitigate some hardware-based threats during manufacturing. However, bringing the manufacturing inside their border does

not fix a major security flaw in the IC supply chain; third parties still are responsible for the manufacturing operations. Hence, a shift in the IC supply chain, as experienced in the 1980s, is unlikely to happen in the following decades.

Consequently, security experts are striving to develop creative countermeasures for all known hardware-based threats. Noteworthy defense techniques include Logic Locking [23–28], IC Camouflaging [30–32], and, Split Manufacturing [21, 22].

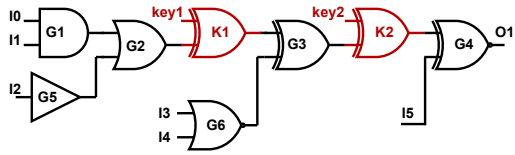

Figure 14: Example of a circuit locked using two XOR key gates, K1 and K2.

Logic locking is a defense technique for locking the design intent behind a key. Additional gates are inserted to prevent the correct propagation of signals unless the correct key is applied to these key gates. An example of Logic Locking is illustrated in Figure 14. For the example circuit to operate as intended, the user has to apply the correct key value to the *key1* and *key2* signals. The key is either programmed at a trusted facility or stored in a tamper-proof memory. According to [26], Logic Locking can protect against adversaries located at the design company, foundry, test facility, and end-user. For example, an IP provider may hide their circuits sold to design companies, protecting against their technology and overuse theft. On the other hand, design companies can utilize Logic Locking against IP theft, overproduction, or hinder the insertion of meaningful hardware trojans.

IC Camouflage is a technique to disguise the functionality of standard cells or parts of a digital circuit. An attacker holding the victim's layout can extract an unnamed gate-level netlist with the original functionality. Techniques such as Logic Locking do not prevent netlist extraction but hide the functionality behind a key. On the other hand, IC Camouflage can hide the functionality of the gates at the layout level. For example, in [31], the authors camouflaged NAND and NOR gates by making their layouts very similar. Thus, making those gates indistinguishable, preventing the extraction of the netlist. Therefore, IC Camouflage can increase resilience against attackers located at the foundry. However, if the camouflage techniques only leverage look-alike cells, the countermeasure might not be enough for an adversary located at the foundry.

Split Manufacturing promotes a hybrid solution between trusted and untrusted fabrication. Because of the nature of the IC structure, it is possible to split the circuit into two parts before manufacturing, the FEOL and BEOL (see Figure 5). Since the FEOL contains all the transistors, a high-end foundry first manufactures this layer. Then, to complete the circuit, a possibly low-end and low-cost foundry manufactures the remaining BEOL on top of the FEOL. Splitting the layout hides the complete design from the high-end untrusted foundry since the FEOL does not contain any wire connection. Thus, only the low-end trusted foundry has complete information about the design. Split manufacturing can combat all threats highlighted in blue in Figure 13. A more detailed discussion of Split Manufacturing is presented in Chapter 4.

2.4 Computing Platforms and Hardware Accelerators

Modern SoCs, over the years, have become more powerful and energy efficient, enabling all sorts of applications that once were deemed unfeasible. Such optimized SoCs are possible not only because of denser and faster ICs. In addition, low-power techniques [94] combined with specialized hardware architecture, i.e., hardware accelerators [95], are also a significant factor in optimized SoC development. Following, I am going to discuss hardware accelerators, their types, advantages, and weaknesses.

General-purpose processing architectures can handle diverse tasks with a solid programming eco-system making it user-friendly. However, these standard processing units cannot efficiently execute some particular tasks. Hence, to improve energy efficiency, SoCs integrate domain- or application-specific hardware accelerators and general-purpose CPUs [96]. Thus, the application running at the CPU offloads specific computing tasks onto the hardware accelerators, enabling greater energy efficiency while maintaining high performance. However, because these accelerators are very complex and designed for a specific system or task, their reusability is greatly diminished. Thus, they are costly, time-consuming, and resource-intense for development.

Such design challenges are overcome by implementing hardware/software co-design techniques or general-purpose hardware accelerators. Instead of offloading an entire application to a hardware accelerator, co-design divides the task into two components; the software component computed by the CPU and the hardware component computed by the accelerator. Thus, reducing the accelerator complexity and speeding up the design process. Furthermore, according to [97], an accelerator can be either loosely or tightly coupled. Loosely coupled accelerators are implemented outside the CPU, which is relatively easier to integrate. Nevertheless, loosely coupled accelerators can suffer performance penalties from the communication interface. In comparison, the tightly alternative is embedded into the CPU architecture as application-specific functional units [98], without interface problems. However, it requires modifications to the instruction set architecture (ISA) [2] of the CPU.

Typically, commercial CPUs do not allow modifications to the ISA. On the other hand, co-design is adopted by several open-source hardware initiatives, such as RISC-V [100], IBM OpenPOWER [101], Sun OpenSPARC [102], and Linux Foundation CHIPS Alliance Project [103]. Thus, a designer can create his own tightly coupled accelerator hardware by utilizing a free and open ISA, such as the RISC-V. In [97], the authors demonstrated an example of the tightly alternative using RISC-V. In addition, they extended the RISC-V ISA to support post-quantum cryptography instructions, speeding up considerably the encryption process.

General-purpose accelerators provide users with a flexible platform, similar to CPUs, but very efficient for domain-specific tasks. Typically, these accelerators can benefit from modern programming languages with practical supporting tools for programming, debugging, and deployment. Furthermore, they can be highly configurable to fit many use cases. Examples of general-purpose accelerators are field programmable gate arrays (FPGA), GPUS, and Tensor Processing Units (TPUs) [104].

FPGA is a programmable fabric that utilizes look-up tables (LUTs) for implementing

[2] The ISA defines the interface between software and hardware [99].

logic functions. In some cases, FPGAs can outperform CPUs; however, it is unlikely to outperform ASICs. The compelling feature of FPGAs is the reconfigurability on demand. In the hardware accelerator context, chip designers can integrate a CPU with an FPGA instead of designing a whole new chip and only having to create the accelerator program. Moreover, the hardware accelerator inside the FPGA can be upgraded at any point in the chip's life span. Thus, due to the flexibility of the FPGAs, they can be implemented outside of a CPU. This configuration is a loosely coupled implementation, with the FPGA as the hardware accelerator. To address the performance loss from the interface, the FPGA can also implement the CPU and the hardware accelerator in a single chip.

Recently a new concept called embedded FPGAs (eFPGA) [105, 106] brings more flexibility to the usage of FPGAs. Embedded FPGAs are IP cores integrated into an ASIC or SoC and have recently gained ground due to their wide range of markets and applications. For example, in the context of hardware accelerators, a designer can use an eFPGA connected to a CPU as a tightly coupled accelerator or even as part of a hardware accelerator. Therefore, eFPGA provides programmability and can accelerate time to market. According to [106], the market share of eFPGA is expected to approach the figure of 10 billion dollars in 2023. Today, users can acquire FPGA-based SoCs [107] or FPGA IPs for integration into their SoCs. Figure 15 illustrates an FPGA-based SoC, where the programmable logic represents the FPGA part.

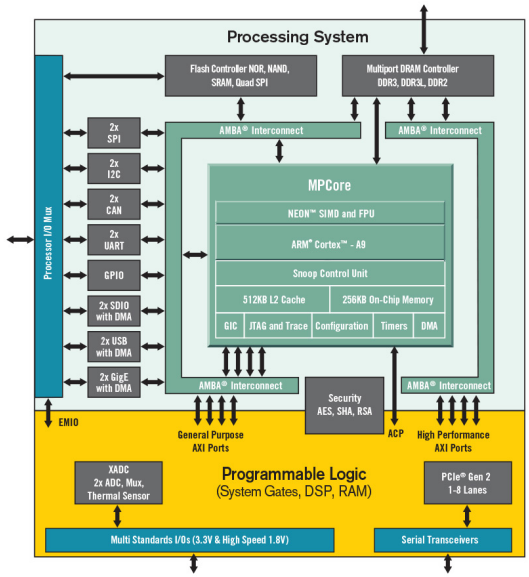

Figure 15: Example of FPGA-based SoC – Zynq-7000s (from [107]).

Traditionally, GPU architectures were developed, as the name suggests, to manipulate computer graphics and image processing. Because of the nature of image processing, GPU architectures focus on specialized massively parallel many-core processors that take advantage of Thread-Level Parallelism (TLP) to handle highly parallelizable applications in a Single-Instruction Multiple Threads (SIMT) paradigm. Thus, GPUs naturally evolved into an efficient general-purpose accelerator for High-Performance Computing

(HPC). Similarly to FPGAs, the user can use rapidly available commercial GPUs for applications other than computer graphics or image processing. Moreover, the GPU vendor *NVIDIA* developed a parallel programming language for GPUs for general purpose processing [8]. Typically, commercial GPUs are sold as discrete cards connected externally to the CPU. However, this configuration does suffer from performance loss from the communicating interface. On the other hand, highly optimized interfaces can achieve outstanding throughput, such as the *PCI Express 4* [108]. Nonetheless, the vendor *AMD* introduced the concept of accelerated processing units (APUs) [98]. An APU incorporates the advantages of a CPU and a GPU into a single package. Therefore, GPUs are a perfect fit for HPC applications such as oil exploration, bioinformatics, and the thriving AI and Machine Learning (ML) domains [109].

HPC, AI, and ML applications are computationally hungry and feasible only with capable hardware. Due to the rise in popularity of these domains, the demand for chips capable of efficiently executing these workloads is sharply increasing. According to the report in [110], the market share of AI chips is forecast to grow at 36.5% CAGR from 2021-2026. Thus, *Google* introduced a new type of computing core called TPU and *NVIDIA* a similar TPU architecture called Tensor Core. TPUs are ASICs that can efficiently solve complex matrix and vector operations at ultra-high speeds and are used specifically for deep learning workloads. On the other hand, *NVIDIA* embeds Tensor Cores in their commercial GPUs, enabling mixed-precision computing to accelerate throughput while preserving accuracy. These cores reportedly achieve very-high speeds in HPC and AI workloads [111].

The hardware architecture itself can be a weakness, where many attacks that take advantage of how the hardware architecture is implemented were recently demonstrated [112,113]. For example, the power of electromagnetic (EM) side-channel attacks against desktop CPUs, mobile CPUs, and FPGAs have been extensively studied [114]. However, not only these architectures but almost all hardware architectures are potentially exploitable. Thus, many other hardware architectures that handle sensitive data and/or are essential to critical systems' functionality are yet to be studied for security vulnerabilities. In [114], the authors described an EM side-channel attack against a GPU AES implementation, and there is no GPU-specific countermeasure for such an attack. Therefore, attacks similar to the one described in [114] are a potential threat to modern SoCs and hardware accelerators. In Chapter 3, the threats against GPUs are discussed in more detail, and a GPU architecture for aiding the research of GPU-specific countermeasures is proposed.

3 Secure GPU-like ASIC Accelerators

This chapter discusses open-source GPUs state-of-the-art and their applications. A literature review revealed the need for open-source GPU architectures for ASIC platforms. The FGPU, a GPU architecture for FPGA, is among the few GPU architectures available. This chapter describes the adaptation of the FGPU for ASIC and how its architecture was optimized. The new architecture is a GPU-like accelerator for ASIC termed G-GPU. Moreover, a fully automated tool for generating G-GPUs termed GPUPlanner was developed. GPUPlanner permits the user to modulate the G-GPU regarding the number of compute units (CUs) with the option of power gating each CU individually. For controlling the power switches, a dynamic power controller is also available. Thus, GPUPlanner enables low-power, design for reliability, and security applications.

This chapter has its content based on the following publication:

[IV] T. D. Perez, M. M. Gonçalves, L. Gobatto, M. Brandalero, J. R. Azambuja, and S. Pagliarini, "G-gpu: A fully-automated generator of gpu-like asic accelerators," in 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 544–547, 2022

3.1 Introduction and Research Gap

New computer applications, especially AI, keep pushing the need for more energy-efficient hardware architectures [96]. For many years, designers have been utilizing application- and domain-specific accelerators as the standard choice for achieving energy efficiency. Those accelerators were designed and tailored according to a specific workload. Examples of hardware accelerators are crypto cores for efficient encryption/decryption [5], and GPUs [6–8] for handling massive parallel computations.

As discussed in Section 2.4, GPU architectures are specialized to handle highly parallelizable applications in a SIMT paradigm used for graphics applications. However, due to the GPU architecture, its application in HPC applications was a natural shift. These applications have a broad range, applicable in bioinformatics, oil exploration, AI, and ML domains [109]. For instance, several supercomputers in the top500 rank utilize GPUs from the vendor *NVDIA* [115].

Nonetheless, research in GPU architectures still needs to be improved because of the need for modern open-source GPU architectures at a sufficiently low level of abstraction. For instance, only FlexGripPlus [116] and FGPU [117] configurable open-source GPUs are available in the literature. Furthermore, the FlexGriPlus architecture is based on the decade-old G80 architecture from the vendor *NVIDIA*, which was never deployed to an FGPA board. On the contrary, the FGPU is explicitly designed for being deployed to FGPA platforms. Consequently, designing, configuring, and implementing modern GPU architectures for ASIC are still a challenge to be tackled by the literature – ASIC platforms represent challenges far from those in FGPA design. However, ASIC platforms can achieve higher performance compared with FPGAs. Hence, all vendors design their GPUs for ASIC.

In order to **bridge this gap**, the GPUPlanner is proposed, an automated and

open-source framework for generating ASIC-specific GPU-like accelerators as IP – this general-purpose accelerator architecture is termed G-GPU. GPUPlanner helps designers generate GPU-like accelerators through user-driven customization and automated physical implementation. For example, G-GPU has a series of user-defined parameters to customize the computation characteristics (e.g., number of processing units), memory access (e.g., cache sizes), and power gating implementation (e.g., insertion of power switches to specific CUs). Therefore, GPUPlanner provides designers with high scalability, facilitating the search for the appropriate G-GPU IP for their systems. On top of that, GPUPlanner explores *smart memory* and on-demand pipeline insertion implementation strategies to optimize even further the G-GPU architecture.

3.2 G-GPU Baseline: the FGPU

As the baseline for G-GPU architecture, the FGPU architecture is utilized. The FPGU is an open-source GPU-like soft processor, highly configurable, to accelerate workloads that fit in the SIMT paradigm, developed for FPGA platforms. However, porting RTL design descriptions that initially targeted FPGAs to ASIC platforms is possible, which requires precise adaptations, especially to the memory hierarchy. In addition to the GPU-like open-source HDL code, this architecture has a supporting LLVM-based compiler. As a result, existing OpenCL kernels can be compiled, providing designers with the ability for fast software development, debugging, and deployment. Moreover, the FGPU architecture can scale up to 64 processing units (and beyond with additional support), being deeply configurable regarding operations, instructions, and memory access.

The FPGU architecture overview is presented in Figure 16. As illustrated, the main component of FGPU is the CU, a SIMD machine of 8 identical Processing Elements (PE0 - PE7) capable of spatially replicating up to eight times. A CU has the computation capacity to run up to 512 work items (i.e., a computational kernel in OpenCL), supporting full thread divergence and allowing each work item to take a different path in the control flow graph. Furthermore, work items are grouped into Wavefronts (WFs) executed concurrently in a CU. Then, WFs are combined into Workgroups (WGs) that share a program counter and are assigned to a particular CU. Therefore, the FGPU architecture is deeply parallelized. Notice that the number of CUs and PEs in each CU are entirely configurable when implementing the FGPU.

In addition, the FGPU architecture has a Runtime Memory (RTM) and a Data cache. The FGPU data cache is a central, multi-port, direct-mapped, and write-back system capable of simultaneously serving multiple read/write requests. Likewise, several data movers are integrated to parallelize the data traffic on up to four AXI Data interfaces [118]. A single AXI Control interface on the hardware side controls the whole FGPU architecture. Then, only standard OpenCL_API procedures are required to control the FGPU on the software side. The width and depth of the AXI Data interface can also be configured.

The FGPU architecture was adapted in the literature to fit different application domains. One example of adaptation proposed by the authors in [119] specialized in the FGPU architecture for persistent deep learning (PDL). The authors added new

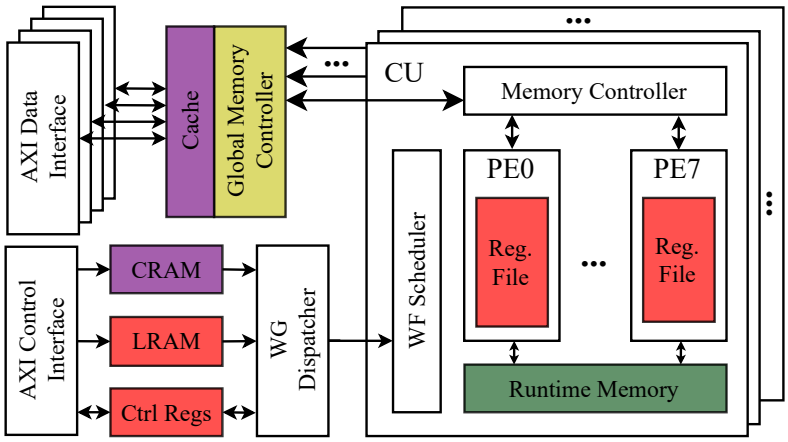

Figure 16: FGPU architecture with memories colored according to the layouts displayed in Figs. 3 and 4 (from [9]).

instructions and enhancements to the microarchitecture and compiler. These adaptations reportedly speed up 56 to 693x in PDL applications. However, the resulting code with the modifications is not publicly available.

Another GPU-like general-purpose accelerator is MIAOW [120], based on the AMD Southern Islands architecture and its ISA. However, the authors described the MIAOW architecture using behavioral C/C++. Thus, it is not fully synthesizable. In [121], the authors proposed the Scratch architecture, a MIAOW extension with the automatic identification of specific requirements of each application kernel. In addition, the authors proposed a tool for generating application-specific and FPGA-implementable trimmed-down GPU-like architectures. MIAOW is another example of GPU-like general purpose for FPGAs, which also has yet to make the source code publicly available.

Therefore, to bridge the literature gap, I proposed a tool termed GPUPlanner for automatically generating tapeout-ready domain-specific accelerators based on GPU-like architectures, making all source codes publicly available. Therefore, this is the first work to propose a similar framework. Furthermore, the proposed framework facilitates a novel and comprehensive design-space exploration (DSE) of GPU-like architecture regarding logic and memory components.

Compared to related works, the proposed architecture targets ASIC flows rather than FPGAs. Because in FPGA designs there is little to no control over how memories are inferred, GPUPlanner DSE allows significantly more complex designs due to the possible different parameters for the memory hierarchy to explore. On top of that, the proposed design and framework are fully synthesizable, tapeout-ready, and available to the community for further investigations, different from MIAOW and Scratch.

3.3 GPUPlaner Tool and Framework

Since the FGPU was originally designed targetting FPGAs, the experiments started by migrating its architecture to ASIC. Thus, the FGPU's architecture requires a few modifications. FPGA's compilers can automatically infer memories from the RTL; thus,

FGPU's code describes all memory blocks as regular FFs. Differently, in ASIC, memory blocks are hand-instantiated IPs instead of inferred ones. Therefore, for migrating the FGPU code, all memory modules must be clearly defined and instantiated accordingly. In this work, the implementations for the experiments utilize a commercial 65nm CMOS technology. The provided foundry's memory compiler has the option for dual-port low-power SRAM IPs, with address sizes ranging from 16 to 65536 words and word sizes from 2 to 144 bits.

A thorough DSE exercise can achieve the best PPA ratio possible for the G-GPU. First, performance was analyzed to verify the maximum operating frequency, i.e., when the setup and hold timing slacks for the critical path are above zero. The maximum operating frequency found during the logical synthesis for the standard version is 500MHz. Here, the standard version is a version without any optimization proposed in this work. Moreover, G-GPU versions with the same number of CUs have similar performance because the CU itself is the bottleneck for performance in G-GPU's architecture. As expected, the starting point of the critical path for the versions without any optimization is a memory block inside the CU.

The delay in accessing the stored data from memories is proportional to their size. Thus, a larger memory, either in the number of words or word size, displays a higher delay for accessing stored data when compared with a smaller memory. Therefore, dividing memory blocks that belong to critical paths is an efficacious strategy to increase the design's performance [122] – called smart memory. For example, memories can be divided by the number of words, the size of the word, or both. However, the impact on performance when halving the number and size of words simultaneously depends on the technology.

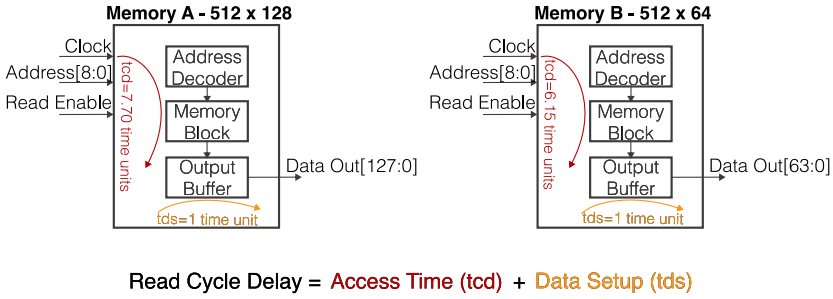

Figure 17: Simplified example of smart memory technique by halving the size of the word.

Applying the smart memory strategy requires a few adaptations in the RTL code. After locating the candidate's memories for the division, the new modules must be adequately instantiated. The input/output data or addresses from the new memory modules require concatenation to maintain the same connections. In our framework, this task is automated to accelerate the optimization process. GPUPlanner has a feature to perform automatic memory division. Figure 17 depicts an example of smart memory division by halving the size of the word. The read cycle delay has two parts, the access time and data setup. When the word size is divided by two, the access time decreases

by almost 25%[3]. Smart memory takes advantage of this characteristic to lower the time necessary to access stored data, increasing the design's performance. In the GPUPlanner framework, the designer only has to point to the candidate's memories and the number of divisions for each memory, then the smart memory division is automatically performed. An extra feature along memory division, GPUPlanner implements pipeline on demand to improve the performance.

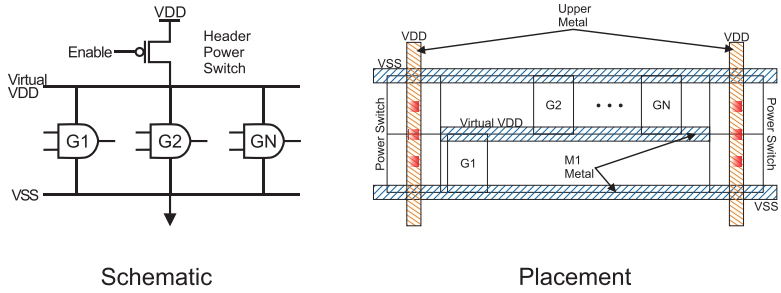

Figure 18: Example of a header power switch schematic (left panel) and placement (right panel).

In addition to the memory division strategy, the GPUPlanner framework also has the option to power gate selected CUs. The power gating allows the complete shutdown of the logic inside the CU. Therefore, GPUPlanner users can use power gating as a low-power strategy, design for reliability, or as a security feature. Our framework uses a coarse-grain header-style power switch provided by the foundry. Figure 18 shows an example of the schematic on the left panel and the placement on the right panel of a header power switch. As illustrated in Figure 18, when the signal *Enable* is de-asserted, the header power switch disconnects the *Virtual VDD* from the *VDD* line, shutting off the gates connected to *Virtual VDD*. Furthermore, coarse-grain power switches utilize lower metal layers to cut the power distribution. In our framework, the utilized power switches break the power distribution at the metal layer M1, highlighted in blue in Figure 18. Moreover, physical synthesis tools can automatically implement power gating intent, using Unified Power Format (UPF) [94] to configure the power switch rules and define the power domains. Thus, GPUPlanner users only have to point to which CU they want to power gate, define the power domain sets, and its implementation is automatically performed.

Another GPUPlanner feature readily available is a dynamic power controller block for power-gated designs. This controller can dynamically turn on and off CUs and pair CUs to work with the same workload. All the controlling is performed on the software side by special instructions added to G-GPU ISA. Figure 19 illustrates the dynamic power controller block. This block is instantiated inside the WG dispatcher to control the number of CUs available, multiplex the workload requests for CUs working in pairs,

[3]The timing presented in Figure 17 are from a commercial memory compiler. However, the actual time figures are not allowed to be disclosed. For that reason, here, these figures are normalized in terms of generic time units.

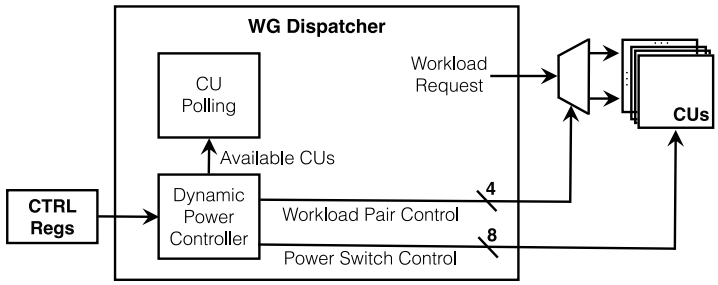

Figure 19: GPUPlanner generic dynamic power controller block diagram.

and enable the power switches. These tasks are controlled by two new instructions added to the ISA: (1) the power status of each CU; (2) adjacent CU working in pairs. The power status is controlled by one bit where logic 0 turns the CU off, and logic 1 turns on the CU. Thus, 8 bits of the instruction are allocated where the bit index is related to a specific CU. Only pairs of adjacent CUs can share the same workload for the mirroring workload feature, i.e., CU #1 sharing the workload with CU #2. Then, 4 bits of the instruction are allocated to enable the mirroring feature.

GPUPlanner is an open-source tool to generate G-GPU IPs from RTL to GDSII automatically; its framework is highlighted in Figure 20. Firstly, a GPUPlanner user has to define the desired specification from the G-GPU. The proposed architecture can configure the number of CUs ranging from 1 to 8 and the option to power gate any CU grouped in different power domains. A G-GPU with more CUs has more computation capacity. Also, the designer has to specify the desired operating frequency of the G-GPU.

After the designer sets the specifications for his/her requirements, one or more versions of G-GPU can be feasible. With a single push of a button, GPUPlanner's framework performs logic and physical synthesis of all G-GPU versions. After each logic and physical synthesis, the resulting PPA has to be checked to guarantee that the design is under the initial specification. If the G-GPU is out of specification, the designer should adjust it and restart the process. Finally, all the generated layouts are ready to be integrated into a system as a tapeout-ready IP.

3.4 Results and Discussion

After a thorough DSE exercise of the GPUPlanner, 12 versions of the G-GPU with a worth PPA trade-off in a general manner were found. These versions have 1, 2, 4, and 8 CUs, and each variant was optimized to run at 500MHz, 590MHz, and 667MHz. Table 1 describes the physical characteristics of each version considered. As expected, the G-GPU sizes grow linearly with the number of CUs. On the other hand, during the optimization phase (see Figure 20), improving G-GPU's performance does not increase area linearly with the frequency increment. For example, increasing the frequency from 500MHz to 590MHz increases the area by an average of 10%. However, when increasing from 590MHz to 667MHz, the area overhead is reduced, increasing only by an average of 2%. In this optimization stage, the divided memories belong to the top level with

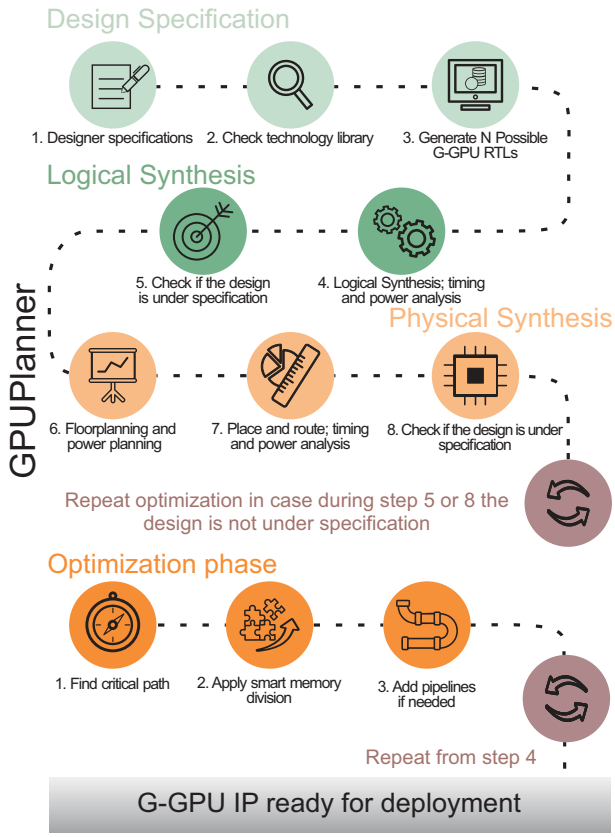

Figure 20: GPUPlanner's G-GPU generation flow (adapted from [9]).

lower implementation density – hence, the small jump in area overhead. Nonetheless, for applications that do not prioritize power consumption, the versions running at 667MHz are a good fit for having a negligible increase in area. Therefore, the G-GPU architecture has potential scalability facilitated by the GPUPlanner framework.

Conventionally, power gating is used for low-power applications. However, it can be used in design for reliability and security applications. For example, shutting off faulty or compromised circuit parts to isolate a problem can be beneficial. Thus, the optional power gating provided by GPUPlanner enables several use cases other than the traditional low-power design. Those use cases can be essential for an optimal GPU, especially for security, where recently a few GPU-specific side-channel attacks have been demonstrated without countermeasures [114, 123, 124].

According to the authors in [114], power or electromagnetic (EM) side-channel attacks against desktop CPUs, mobile CPUs, and FPGAs have been extensively studied. However, modern GPUs are rarely taken into account. In [114], the authors demonstrated the effectiveness of an EM side-channel attack against a GPU AES implementation.

Table 1: Characteristics of 12 different GGPU solutions generated by our tool after logic synthesis in Cadence Genus.

#CU & Freq.	Total Area (mm^2)	Memory Area (mm^2)	#FF	#Comb.	#Memory	Leakage (mW)	Dynamic (W)	Total (W)
1@500MHz	4.19	2.68	119778	127826	51	4.62	1.97	2.055
2@500MHz	7.45	4.64	229171	214243	93	8.54	3.63	3.77
4@500MHz	13.84	8.56	437318	387246	177	16.07	6.88	7.14
8@500MHz	26.51	16.39	852094	714256	345	30.79	13.33	13.86
1@590MHz	4.66	3.15	120035	128894	68	4.73	2.57	2.66
2@590MHz	8.16	5.34	229172	221946	120	8.73	4.63	4.81
4@590MHz	15.03	9.72	436807	397995	224	16.41	8.70	9.02
8@590MHz	28.65	18.49	850559	737232	432	31.25	16.81	17.40
1@667MHz	4.77	3.26	120035	130802	71	4.65	2.62	2.72
2@667MHz	8.27	5.45	229172	222028	123	8.72	4.69	4.87
4@667MHz	15.15	9.83	436807	398124	227	16.43	8.75	9.07
8@667MHz	28.69	18.60	848511	730506	435	30.21	19.10	19.76

They argue that the literature lacks GPU-specific countermeasures for GPU-based cryptographic implementation, which should be investigated in practice. As mentioned, the literature also needs an ASIC open-source GPU architecture, arguably contributing to the lack of such studies. Therefore, the proposed G-GPU architecture can aid the research of GPU-specific countermeasures for practical applications.

Due to the physical characteristics of the CU, inserting power switches does not entail area or performance overhead[4]. The only requirement from the user side is the implementation of a controller for the power switch signals. For example, a power controller can dynamically scale the number of CUs available accordingly to the workload. Thus, saving power when a workload can run with fewer CUs. Even more, with a few adjustments on the WG dispatcher (see Figure 16), the workload can be mirrored between sets of CUs and work similarly to a TMR technique. Thus, a voter plus the power controller can select the desired set of CUs to work similarly to a TMR. For example, suppose a user can implement a G-GPU with 8 CUs with four sets of power domains with 2 CUs each. Then, each set can work in lockstep to detect any fault in the circuit. Moreover, the voter can potentially detect the presence of a hardware trojan inserted in a specific CU.

Power side-channel attacks without the aid of SCTs require acquiring a tremendous amount of long power traces. Thus, a power controller can turn on the G-GPU only when executing a task, potentially reducing the chance of such power side-channel attacks by reducing the timing window of the power/EM traces. In [114], the authors needed to collect thousands of significant EM traces to retrieve the crypto key from a GPU AES application. Alternatively, tasks can be executed each time in a different CU while the other remains shut off. Thus, if an HT compromises a particular CU, this strategy can diminish the attack's success rate. However, these examples of power gating are application and user-specific – GPUPlanner does not implement particular applications of power gating. More importantly, GPUPlanner allows users to perform power gatings controlled by a generic dynamic power controller, which can be tweaked in the G-GPU architecture according to their needs and at runtime.

For the physical synthesis, was chosen six versions of the G-GPU: (1) 1CU@500MHz,

[4]Power gating can introduce area and performance overhead. For high-density designs, the insertion of power switches will increase the area. In addition, power-on latency can degrade the design's performance. Suppose the power gatings are not carefully performed; IR drop due to the power switches can also hinder the performance.

(2) 1CU@500MHz with power switches, (3) 1CU@667MHz, (4) 1CU@677MHz with power switches, (5) 8CU@500MHz, and (6) 8CU@667MHz. The floorplan of the G-GPU is broken into two partitions, one with the CU(s) and one with the rest of the blocks (see Figure 21). For the CUs, the density was set to 70%. Then, the rest of the logic was placed and routed at the top level, and the top level size was set to fit all routing wires for the connections rather than achieving high densities. Because the top level comprises three modules and does not have as many memory blocks as the CU, achieving a high density of utilization was possible. Thus, the top has an average density of 75%. Nonetheless, this floorplan strategy allows designers to scale the G-GPU architecture without any extra effort regarding the number of CUs. Once a CU partition is fully placed and routed, it can be implemented in versions with more than 1 CU by cloning the partition in the final floorplan of the design. Moreover, the user can easily create a collection of different CU layout blocks and scale the floorplan regarding the number of CUs for different application scenarios.

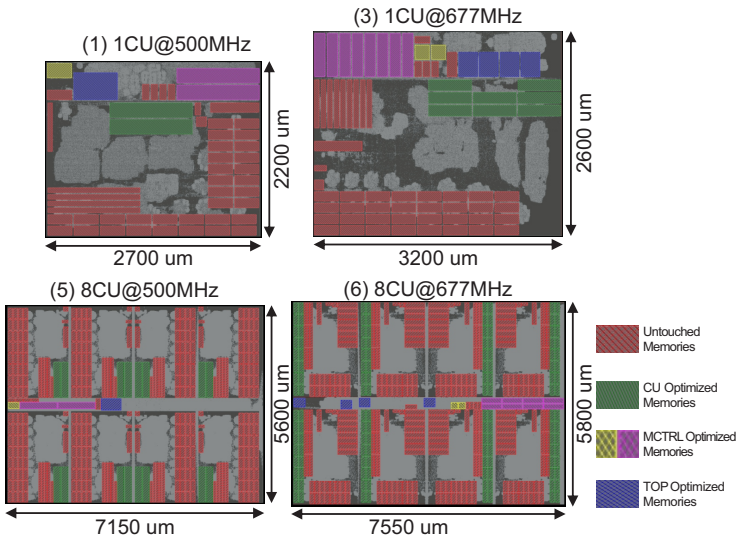

Figure 21: Layout comparison between the minimum and maximum performance of G-GPUs with 1 CU (top) and 8 CUs (bottom).

The layouts for the versions with 1 and 8 CUs without power switches are depicted in Figure 21. Only the layouts with the same number of CUs are in size scale. The block memories divided for augmenting the performance are highlighted in green for the CU partition, yellow and pink for the general memory controller (MCTRL), and blue for the top. Note how different the floorplan is between the version with optimizations running at 667MHz and without optimizations running at 500MHz. Extracting maximum performance requires strategic memory block placement, hence, the difference in the floorplan.

Figure 22 illustrates the G-GPUs with power gating. Note that the layout size is the same between the version with and without power gating, and the memory placement is

Table 2: Comparison of power consumption for 1CU@500MHz and 1CU@677 versions with and without power gating.

# CU & Freq.	Power Gating	# Power Switches	Dynamic Power (W)			Leakage (μW)			
			Total	Always-on	CU	Total	Always-on	CU	Power Switch
1@500MHz	Yes	942	1.768	0.36	1.408	385	89.5	292	0.503
1@500MHz	No	0	1.753	-	-	384	-	-	-
1@667MHz	Yes	1110	2.966	0.737	2.22	522.4	157.8	364	0.6
1@667MHz	No	0	2.957	-	-	520.6	-	-	-

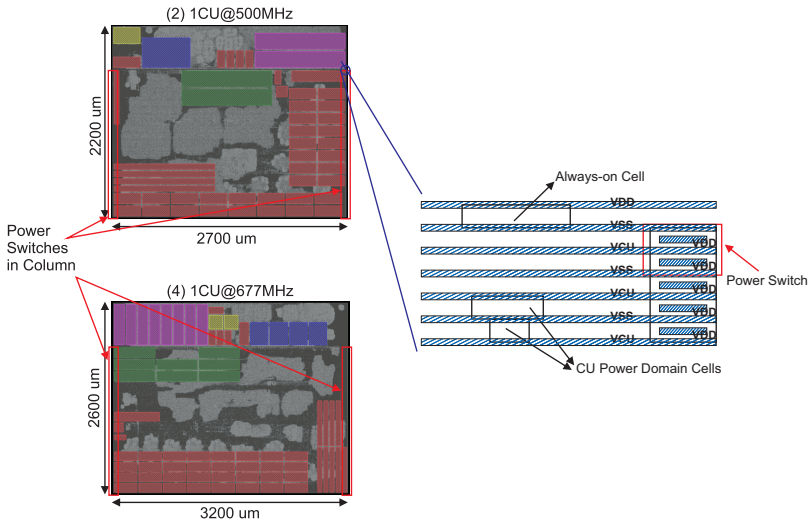

Figure 22: Layout comparison between G-GPU (2) 1CU@500MHz and (4) 1CU@677MHz with power gating.

slightly different. Due to the number of memory blocks and their placement, the edges without pinouts usually are not populated with cells. In the case of the G-GPU, those edges are where was placed the power switches in column fashion, as highlighted in red in Figure 22. As described before, the power switches break the power distribution at M1 metal. In Figure 22, the always-on power net is VDD, and the CU partition power net is VCU, with a shared ground VSS. Therefore, the power switch connects to VDD, VSS, and VCU. The connection between VDD and VCU is controlled by the signal enable (see Figure 18). The overhead difference between with and without power gating is only perceived when all power domains are turned on. However, there is only a negligible difference in leakage. The power results of the G-GPUs with and without power gating are described in Table 2. Even with a large number of power switches (more than a thousand for (3) 1CU@667MHz), the difference in leakage is less than 1 μW. On the other hand, when the G-GPU is idle, the user can turn off the entire CU partition. The power reduction comparison between a version with and without power gating depends on how the design handles the clock. If the clock is always running without any clock gating, the reduction is a part of the dynamic plus the leakage power.

Figure 23 illustrates CU partition dynamic power versus switching activity [5] for (2) 1CU@500MHz and (4) 1CU@677MHz. As described in Section 2.2, dynamic power comprises internal and switching power. Note that the largest portion of the dynamic power is from internal power. Hence, even if the CU is not running at capacity (idle), the consumption is still high. Therefore, the power reduction achievable when power gating is massive, more than 2W for the (4) 1CU@677MHz. On the other hand, if the design has the means to stop the CU clock, the power reduction difference when power gating is only the leakage – approximately 370μW for the (4) 1CU@677MHz.

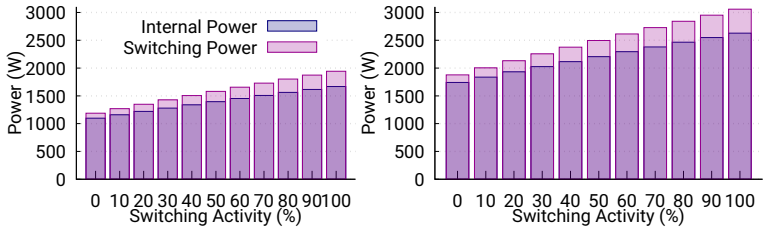

Figure 23: Compute unit partition dynamic power versus switching activity for (2) 1CU@500MHz (left panel) and (4) 1CU@677MHz (right panel).

A performance comparison was made between the popular RISC-V architecture to evaluate the G-GPU as an ASIC accelerator. For the comparison, the *OpenHW group Core-V cv32e40p* RISC-V was utilized [125]. Furthermore, the logic synthesis was done utilizing the same technology for both architectures (commercial 65 nm CMOS). Since the G-GPU's maximum operation frequency is 667MHz, both RISC-V and G-GPU operating frequencies are set to 667MHz for a fair comparison. The RISC-V was implemented with 32kb of memory, and the G-GPU with 1/2/4/8 CUs. Seven micro-benchmarks from the AMD OpenCL SDK were chosen for the experiments. The payload size is set as the largest that the RISC-V compiler can handle without crashing. In the same way, for the G-GPU, the payload sizes are chosen to make its computing units fully utilized. For the evaluation, a pessimistic approach for the G-GPU is considered to compare the performance of the different-input size applications. For example, one could increase RISC-V application input sizes by multiplying its cycle count by the G-GPU/RISC-V input size ratio. These results are shown in Figure 24.

The first experimental evaluation compares raw performance between G-GPU and RISC-V for the exact input sizes. As illustrated in Figure 24, G-GPU with 8 CUs is up to 233.4 times faster than RISC-V. However, a higher speed-up magnitude is only achieved for applications that take advantage of high parallelism. G-GPU can be as low as only 1.2 times faster than RISC-V for applications with low to no parallelism. However, as G-GPU is a domain-specific ASIC accelerator, such results are expected once it will not becomes the best option for general-purpose applications. Therefore, a user interested in implementing a G-GPU as an accelerator can utilize these provided data to ponder if this type of architecture is a good fit for his/her system, considering

[5] Switching activity is set to emulate the circuit operation when test vectors are not available. The switching activity is how much a signal switches in relationship with the clock, e.g., 20% of switching activity means signals switch one time per 5 clock cycles.

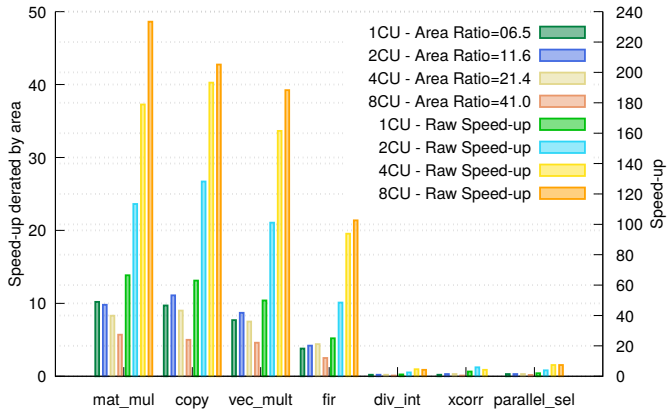

Figure 24: Speed-up over RISC-V.

only the raw speed-up.

The measured area is factored into performance speed-up for the second experimental evaluation. For that, the previously measured speed-up is derated by dividing the area ratio between G-GPU and RISC-V (G-GPU/RISC-V). The G-GPU with 1 CU has an area that is 6.5 times larger than the RISC-V, and it achieves the best increase in performance per area of 10.2 times the RISC-V's. On the other hand, G-GPU with 8 CUs has an area that is 41 times bigger than RISC-V's, thus achieving a performance per area of 5.7 times faster than RISC-V's. This trend happens mainly because data dependency and global memory communication limit parallelism. Thus, the provided increased processing power of a G-GPU configuration with more CUs.

Currently, the GPUPlanner can generate tapeout-ready GPU-like ASIC accelerator IPs with the flexibility to choose the number of CUs, layout size, frequency of operation, and power gating implementation. Moreover, the results show that the G-GPU performs better than a general-purpose accelerator like the RISC-V in specific scenarios. Furthermore, the GPUPlanner can still be improved to include additional features such as the power gating controller and architecture featuring more than 8 CUs. Finally, the GPUPlanner tool is publicly available to users interested in utilizing the already implemented features and users interested in improving the GPUPlanner capabilities even further [126].

4 Split Manufacturing: Attacks and Defenses

This chapter discusses the Split Manufacturing technique state of the art. From a literature review, I identified that the Split Manufacturing technique research was mature and relevant for having a survey. Thus, I present in this chapter a reduced version of the survey published in [22]. It is noteworthy to mention it was the first published survey on the topic. In this survey, I comprehensively classified every attack against split layouts and every defense technique for improving even further split layouts security. On top of that, I addressed a controversial topic (efficacy of the Split Manufacturing technique) among the recent publications. Finally, this survey is very important for future research on Split Manufacturing, being a focal point to start from for security experts interested in the topic.

This chapter has its content based on the following publication:

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

4.1 Introduction

As discussed in Sections 2.1 and 2.3, ensuring the integrity and trustworthiness of ICs has become more challenging [59] over time. Mainly because of the restructuring of the IC supply chain, which is now very complex and highly globalized. Moreover, counterfeiting and IP infringement are growing problems in the electronics sector. For example, in Europe, seizures of counterfeit electronics products increased by almost 30% when comparing 2014-2016 to the 2011-2013 period [58]. Legitimate electronics companies reported about $100 billion in sales losses every year because of counterfeiting [127].

In Section 2.3, I discussed several techniques that have been proposed to combat threats during the IC's life cycle individually. However, very few of these techniques directly address the lack of trust in the manufacturing process. Nevertheless, Split Manufacturing emerged to promote a hybrid solution between trusted and untrusted manufacturing. Around 2006, Carnegie Mellon and Stanford universities prepared a white paper proposing the technique to draw Defense Advanced Research Projects Agency (DARPA) [128] attention. Later, the technique was picked by IARPA, which then launched the Trusted IC program [129].

In Split Manufacturing, as already discussed in Secion 2.3, the key concept is to *split* the circuit into two, the FEOL and BEOL parts. The FEOL contains the transistors and perhaps the first couple of metal layers, and the BEOL contains the remaining ones. Then, these parts can be manufactured in different foundries. The FEOL is assumed to be first manufactured in a high-end modern foundry to access advanced transistors. After, the BEOL is stacked on top of the FEOL by a second, most likely low-end, foundry. The stacking process requires electrical, mechanical, and/or optical alignment techniques to secure the connection between the two.

As mentioned before in Section 2.1, only a few foundries are capable of manufacturing advanced ICs (see Figure 4). Consequently, almost all design companies have to outsource their IC manufacturing to these untrusted foundries. This outsourcing

practice exposes their designs against threats that may occur during manufacturing. Nonetheless, design companies can apply the Split Manufacturing technique to protect their designs, thus, combating threats such as overproduction, cloning, and tampering (these threats are highlighted in blue in Figure 13). By splitting the design into FEOL and BEOL, Split Manufacturing protects the design by hiding sensitive data from the untrusted foundry. In advanced technologies, the FEOL contains the transistors and possibly a few metal layers of ultra-thin metals, which are the most complex part of a CMOS process to manufacture [130]. Thus, it is logical to utilize the few high-end foundries for manufacturing the FEOL layer, despite being untrusted foundries. For manufacturing the remaining metal layers, a low-end foundry may be capable of completing the IC by stacking the BEOL on top of the FEOL. Split Manufacturing was successfully demonstrated in [131–133], where designs were manufactured with ~0% of faults and reportedly have performance overhead of roughly 5%. These demonstrations show that Split Manufacturing, in principle, is feasible. Thus, design companies can use the technique while outsourcing their IC manufacturing to advanced foundries without fully exposing their designs.

Nonetheless, applying Split Manufacturing has to be done with caution. The technique's success depends on the compatibility between the technologies used to build the FEOL and BEOL. A layout, in theory, can be split at any layer if the chosen layer presents a good interface between FEOL and BEOL. However, since advanced technologies utilize the dual-damascene fabrication process, the layout can only be split into metal layers [134], as the FEOL cannot terminate in a via layer. This is because the dual-damascene process of metal deposition fills via-metal pairs simultaneously. Thus, via-metal pairs (e.g., V1 and M2) must mandatorily be built in the same facility.

Since after splitting the layout, the FEOL ends on metal, the first bottom layer on the BEOL is a via layer. Hence, staking the BEOL is only possible if there is a way to land the via on the FEOL uppermost layer without violating any DRC of both technologies. Thus, both technologies are compatible with each other, enabling Split Manufacturing. As discussed in Section 2.2, DRCs guarantee manufacturability and functionality. These rules include geometric characteristics of the metal layers, such as minimum enclosure, width, spacing, and as well, density checks, ERC checks, and others. For advanced technologies, designers have a rich selection of via shapes. Thus, the technologies are compatible if at least one via shape is valid.

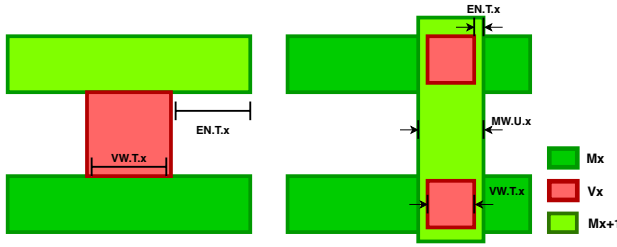

Figure 25: Compatibility rules between FEOL and BEOL (adapted from [131]).

According to [131], compatibility between two technologies can be generalized by

enclosure rules as in Equation 3, where MW.U.x is the minimum width of Mx on an untrusted foundry, VW.T.x is the minimum width of Vx on a trusted foundry and EN.T.x is the minimum enclosure on the trusted foundry. These rules are illustrated in Figure 25, where the left side of the image portrays a cross-section view, and the right side shows the top view. According to Figure 25, the minimum enclosure width, Mx.EX.Vx must be compatible between the two foundries. Nonetheless, Equation 3 is not sufficient for advanced technologies, as it does not consider the complex rule for vias and line endings (enclosure from one side, two sides, three sides, T-shaped/hammerheads, and many others).

$$MW.U.x \geq VW.T.x + (2EN.T.x) \tag{3}$$

The Split Manufacturing design flow is similar to the regular one illustrated in Figure 6. However, it presents some challenges and slight modifications to the design flow. For instance, if two different technologies are utilized for generating the layout, a hybrid process design kit (PDK) is required for the physical synthesis. Since no company offers a Split Manufacturing service, the hybrid PDK must be created in-house. Furthermore, depending on the metal layer where the layout is to be split, it may affect the existing IPs. For example, standard cell IP typically requires two metal layers, while memory IP may utilize 4 to 5 metals. Thus, using such IPs limits the metal layout in which the layout can be split. If that is the case, standard cells and memories must be re-designed to enable the split in lower metal layers. Hence, this presents a significant challenge, making Split Manufacturing much harder to execute.

Hiding part of the BEOL layer from the untrusted foundry is argued to expose enough information to be exploited by a potential attacker. The BEOL connections can be effectively retrieved from an attack against the FEOL by making educated guesses. Nevertheless, the success of the guessing process depends heavily on the amount of information the attacker possesses. Thus, the assumed threat model determines the efficiency of attacks against FEOL. The literature describes two distinct threat models:

- **Threat model I:** an attacker located at the untrusted foundry holds the FEOL layout and wants to retrieve the BEOL connections.

- **Threat model II:** an attacker located at the untrusted foundry has the information of the entire gate-level netlist in their hands. That netlist is assumed to be handed over by a malicious observer. Nonetheless, the attacker inside the foundry only holds the FEOL and wants to retrieve the BEOL connections [135].

As the primary purpose of Split Manufacturing is to expose the minimum of information possible to the untrusted foundry, the second threat model completely nullifies any security introduced when splitting the circuit. For example, reverse engineering a layout while holding the gate-level netlist becomes a trivial task if the attacker only holds the FEOL or the complete layout. Moreover, assuming an attacker inside a third-party company knows such sensitive data (e.g., gate-level netlist) challenges the integrity of the design company itself. Even more severe, if a rogue element inside the design company can steal the gate-level netlist, other design representations could be equally

stolen as well, including the complete layout (i.e., FEOL plus BEOL layers). Thus, the vulnerability described by threat model II is so severe that Split Manufacturing has virtually no reason to be applied. Accordingly, for the remaining manuscript, threat model I is the focus for discussing Split Manufacturing. Nevertheless, all related works that assumed threat model II were covered in the conducted survey.

From the threat model I, an attacker holding the FEOL is interested in recreating the entire design as close as possible, ideally the same as the original. For that, he/she must retrieve the BEOL connections. Typically, it is assumed that the attackers are skilled and work within the untrusted foundry in some capacity. Hence, they have an excellent knowledge of the technology utilized to generate the victim's layout. Therefore, extracting the incomplete gate-level netlist from the FEOL layout is a trivial task [136].

Split Manufacturing presents a promising technique to enhance the security of ICs in this era of fabless design companies. However, the technique still faces some serious challenges:

> *Logistical challenge:* currently, Split Manufacturing is not integrated into the IC supply chain. Herefore, finding foundries with compatible technologies willing to work together is not trivial. Thus, a commercial Split Manufacturing solution is unlikely to be created soon.
>
> *Technological challenge:* even within compliant technologies, non-negligible overheads can be introduced if they are vastly different[6]. In the worst-case scenario, it can make routing impossible. Thus, this fact narrows down the technology choices available and the feasibility of particular layers as candidates for splitting.
>
> *Security challenge:* the attained security of straightforward Split Manufacturing is still under debate. Attacks against the FEOL can be effective, where the hidden connections can be retrieved.

In the following sections, related works in the literature about Split Manufacturing are categorized as attacks and defenses. For *attacks*, authors propose modifications of existing attacks to improve their effectiveness and new types of attacks. In *defenses*, the authors proposed new techniques to be applied along Split Manufacturing to enhance its security level.

4.2 Attacks on Split Manufacturing

Many attacks against the FEOL have been proposed, most of which are termed *proximity attacks* [138–144]. The attacks are compiled in Table 3, where is described the threat model used, the type of attack, the novelty of the attack, benchmarks, and the size of the circuits utilized to assess the attacks. Furthermore, a few results were selected from each work described in Table 4. For better contrast, these results are selected for the smallest and larger circuits available. Following, I present a brief discussion about attacks against the FEOL. Finally, for a thorough and complete discussion, I direct the reader to [22].

[6]For a thorough discussion and silicon results on BEOL-related overheads, please refer to [137].

Table 3: Threat Models, Attacks, and Metrics.

Work	Year	Threat model	Attack type	Novelty	Benchmark suite(s)	Largest circuit size (gates)	Avg. circuit size (gates)
[138]	2013	I	Proximity	Attack Based on Proximity	ISCAS'85	3.51k	1288
[139]	2016	II	Proximity	Placement and routing proximity used in conjunction	ISPD'11	1.29M	951k
[140]	2018	I	Proximity	Network-Flow-Based with Design Based Hints	ISCAS'85 & ITC'99	190.21k	9856
[141]	2018	I	Proximity	Proximity Attack Based on Machine Learning	ISPD'11	1.29M	951k
[142]	2019	I	Proximity	Proximity Attack Based on Deep Neural Network	ISCAS'85 & ITC'99	190.21k	9856
[143]	2019	I	SAT	SAT Attack without Proximity Information	ISCAS'85 & ITC'99	190.21k	9856
[144]	2019	I	SAT	SAT attack dynamically adjusted based on proximity information	ISCAS'85 & ITC'99	190.21K	9856

As previously alluded, when implementing the IC, EDA tools focus mainly on optimizing PPA. Hence, the solution found by the placement algorithm often places connected cells close to one another to reduce area, wire length, and delay. Consequently, the missing BEOL connections could be found by assessing the input and output pins in proximity, hence, the name proximity attack. However, the number of missing connections increases the probability of making a wrong connectivity guess. In turn, a circuit split into a lower metal layer has a high level of security. In [138], the authors reported the first proximity attack against the FEOL. They utilized the distance between output-input pairs as a metric to recover the missing BEOL connections (i.e., a proximity attack). The authors reported an average effectiveness of 96% of Correct Connection Rate (CCR) across all the benchmarks considered.

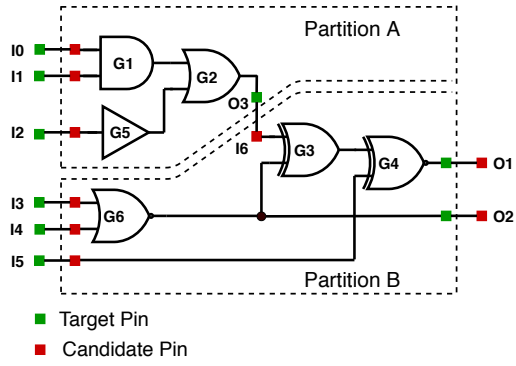

Figure 26: Example of a partitioned circuit (from [22]).

To understand the hints left behind by EDA tools, consider as an example the partitioned circuit illustrated in Figure 26. The circuit contains two partitions, A and B, each with three gates. Not considering connections within the partitions, partition A has three inputs and one output pin, while partition B has three inputs and two output pins. The partitions are connected to each other by one input-output, where the output pin of gate G2 is connected to one of the inputs of gate G3. Let us assume the output pin from partition A $P_{x,A,out}$ is a candidate for its corresponding input pin from partition B $P_{x,B,in}$. From the perspective of EDA tools, those pins will most likely be placed as close as possible. Therefore, using this insight, an attacker may recover the

missing connection in the FEOL layout. The authors in [138] have argued that their proposed attack flow is successful because it can leverage the following "hints" provided by the EDA tools:

> *Hint 1 - Input-Output Relationship*: partition input pins are connected either to another partition output pin or to an input port of the IC (i.e., input-to-input connections are excluded from the search space).
>
> *Hint 2 - Unique Inputs per Partition*: input-output pins between partitions are connected by only one net. If a single partition output pin feeds more than one input pin, the fan-in and fan-out nodes are usually placed within the partitions (i.e., one-to-many connections are ruled out from the search space).
>
> *Hint 3 - Combinational Loops*: only specific structures normally utilizes combination loops (e.g., ring oscillators). These structures are straightforward to identify. Thus, in most cases, random logic does not contain combinational loops – connections that would lead to it can be eliminated from the search space.

Nonetheless, missing connections can be correctly retrieved by identifying the closest pin from a list of possible candidates. Utilizing the hints described above, the attacker can create a list of possible candidates. Candidates' pins are separated into unassigned inputs and outputs pins. Hence, a metric based on the minimum routing distance can be used to connect the unassigned pins.

Based on the work presented in [138], other similar attacks towards the FEOL were developed. A more advanced proximity attack is proposed by [139], where the authors take into account other metrics besides the distance of the unassigned pins. They proposed four different techniques to identify a small search neighborhood area for each candidate. The techniques are called *placement proximity*, *routing proximity*, *crouting proximity*, and *overlap of placement and routing proximity*.

On the other hand, the attack proposed in [140–142] leverages statistical analysis to improve the search for the missing connections in proximity attacks. In [140], the authors proposed a network-flow-based attack framework, where the missing connections are found by solving a min-cost network-flow problem [145]. A Machine Learning (ML) framework was created by the authors in [141] in an attempt to improve the attack proposed in [139]. Finally, the authors in [142] proposed a more sophisticated deep neural network, using placement and routing hints as vector and image-based features to formulate the challenges.

Moreover, the authors in [143, 144] proposed an SAT solver-based attack method derived from CycSat [146]. Contrary to proximity attack, the authors claim their SAT attack does not need (or depend on) any proximity information or hint from EDA tools. Instead, they model a interconnect network as key-controlled multiplexers (MUX) with all the missing connections. Hence, as input to the SAT-solver, the FEOL circuit with the MUX network is utilized, and a packaged IC serves as an oracle. Thus, the threat model considered in [143, 144] is slightly different; the authors assume that a working circuit exists.

Table 4: Benchmark Size and Comparison of Attack Results.

Work	Benchmark	Attack	Split Layer	Size (In Gate Count)	Metric	Result
[138]	c17	Proximity	Not Defined	6	CCR(%)	100
[138]	c7552	Proximity	Not Defined	3513	CCR(%)	94
[139]	Superblue 1	Placement Proximity	M2	847k	% Match in List	12.84
[139]	Superblue 1	Placement Proximity	M2	847k	CCR(%)	5.479
[139]	Superblue 1	Routing Proximity	M2	847k	% Match in List	71.08
[139]	Superblue 1	Routing Proximity	M2	847k	CCR(%)	0.651
[139]	Superblue 1	Overlap (P&R) Proximity	M2	847k	% Match in List	13.05
[139]	Superblue 1	Overlap (P&R) Proximity	M2	847k	CCR(%)	3.977
[139]	Superblue 1	Crouting Proximity	M2	847k	% Match in List	82.08
[139]	Superblue 1	Crouting Proximity	M2	847k	CCR(%)	0.651
[140]	c7552	Network-flow Based Proximity	Not Defined	3513	CCR(%)	93
[140]	c7552	Proximity	Not Defined	3513	CCR(%)	42
[140]	B18	Network-flow Based Proximity	Not Defined	94249	CCR(%)	17
[140]	B18	Proximity	Not Defined	94249	CCR(%)	< 1
[141]	Superblue 1	Proximity	M6	847k	% Match in list	33.40
[141]	Superblue 1	Proximity	M6	847k	CCR(%)	0.76
[141]	Superblue 1	ML	M6	847k	% Match in list	83.12
[141]	Superblue 1	ML	M6	847k	CCR(%)	1.91
[141]	Superblue 1	ML-imp	M6	847k	% Match in list	74.65
[141]	Superblue 1	ML-imp	M6	847k	CCR(%)	2.11
[141]	Superblue 1	ML-imp	M4	847k	% Match in list	75.45
[141]	Superblue 1	ML-imp	M4	847k	CCR(%)	2.58
[142]	B18	DL Network	M1	94249	CCR(%)	4.59
[142]	B18	DL Network	M3	94249	CCR(%)	23.74
[142]	c7552	DL Network	M1	3513	CCR(%)	11.11
[142]	c7552	DL Network	M3	3513	CCR(%)	72.30
[143]	c7552	SAT Attack	Not Defined	3513	Logical Equivalence(%)	100
[143]	B18	SAT Attack	Not Defined	94249	Logical Equivalence(%)	100
[144]	c7552	Improved SAT Attack	Not Defined	3513	Logical Equivalence(%)	100
[144]	B18	Improved SAT Attack	Not Defined	94249	Logical Equivalence(%)	100

4.3 Split Manufacturing Defenses

As discussed in the previous section, attacks against Split Manufacturing can be effective. Attackers can realistic retrieve the missing BEOL connections. Thus, any security introduced by Split Manufacturing is annulled if the connections are successfully recovered. Consequently, several works question straightforward Split Manufacturing. Several authors have proposed techniques to use together with Split Manufacturing to increase security against attacks. A list of defense techniques is compiled in Table 5. This comprehensive list describes the threat model and defense metric utilized in each work, depending on the attack they are combating. Often defense techniques introduce heavy PPA overhead. Table 5 also reports if the studied work addressed overheads and which one was taken into account, such as wirelength overhead (WLO), PPA, and the number of swaps performed. The defense's results are reported in terms of CCR or effective mapped set ratio (EMSR). The EMSR metric attempts to quantify the ratio of the real gate location of a given mapping during a simulated annealing-based attack.

Table 5: Split Manufacturing Defenses.

Work	Year	Threat Model	Category	Defense	Metrics	Defense Overheads Presented
[138]	2013	I	Proximity Perturbation	Pin Swapping	Hamming Distance	-*
[135]	2013	II	Wire Lifting	Wire Lifting	k-Distance	Power, Area, Delay and WireLength
[132]	2014	I	Layout Obfuscation	Layout Obfuscation for SRAMs and Analog IPs	-	Performance, Power and Area
[147]	2014	I	Layout Obfuscation	Obfuscation Techniques	Neighbor Connectedness and Entropy	Performance and Area
[148]	2015	I	Layout Obfuscation	Automatic Obfuscation Cell Layout	Neighbor Connectedness and Entropy	Performance, Power and Area
[149]	2015	I	Layout Obfuscation	Obfuscated Built-in Self-Authentication	Obfuscation Connection	Number of Nets
[139]	2016	I	Wire Lifting	Artificial Blockage Insertion	Number of Pins	-*
[150]	2016	I	Wire Lifting	Net Partition, Cell Hidden and Pin Shaken	-	-*
[151]	2017	I	Proximity Perturbation	Routing Perturbation	Hamming Distance	Performance and WireLength
[152]	2017	I	Wire Lifting	Secure Routing Perturbation for Manufacturability	Hamming Distance	Performance and WireLength
[153]	2017	I	Proximity Perturbation	placement-centric Techniques	CCR	Performance, Power and Area
[154]	2017	II	Proximity Perturbation	Gate Swapping and Wire Lifting	Effective Mapped Set Ratio and Average Mapped Set Pruning Ratio	WireLength
[155]	2018	I	Wire Lifting	Concerted Wire Lifting	Hamming Distance	Performance, Power and Area
[140]	2018	I	Proximity Perturbation	Secure Driven Placement Perturbation	Hamming Distance	Power and WireLength
[156]	2018	I	Proximity Perturbation	placement and routing perturbation	Hamming Distance	Performance, Power and Area
[157]	2019	I	Layout Obfuscation	Isomorphic replacement for Cell Obfuscation	Isomorphic Entropy	-*
[158]	2019	II	Layout Obfuscation	Dummy Cell and Wire Insertion	k-security	Area and WireLength

* Authors do not present any discussion regarding overhead.

Defenses techniques can be divided into three categories; proximity perturbation, wire lifting, and layout obfuscation. Proximity perturbation aims to change the location

of cell pins to mislead proximity attacks. On the other hand, wire lifting moves routing wires to upper layers in order to increase the amount of hidden routing. Finally, layout obfuscation hides the circuit structure from the attacker. Nonetheless, defense techniques do overlap. For example, a technique that primarily promotes proximity perturbation may lead to indirect wire lifting. Hence, the categorization of defense techniques described here is done in the best effort to list state of the art comprehensively. The results for the Proximity Perturbation and Wire Lifting categories are compiled in Tables 6 and 7.

As the name suggests, proximity perturbation defenses focus on reducing the hints introduced by the EDA tools. Thus, this defense category aims to diminish the proximity information between the exposed pins on the FEOL by making targeted changes to the circuit and decreasing the success rate of proximity attacks toward Split Manufacturing.

Among many proximity perturbation defense techniques proposed, authors in [138] utilized pin swapping as a countermeasure against proximity attacks. Partition pins are rearranged to alter their distance, misleading attackers interested in performing a proximity attack. For example, if the pins $P_{G3,B,in}$ and $P_{G6,A,in}$ (Figure 26) are swapped, their connection would be incorrectly guessed during a proximity attack. Therefore, the authors in [138] propose using hamming distance to quantify the difference between the outputs from the original netlist and the modified one. For them, the optimum netlist is arguably achieved for a Hamming distance of 50%, which induces maximum ambiguity for a potential attacker.

In [140, 151, 156], the authors also leverage the Hamming distance for their proximity perturbation techniques. In [151], the authors proposed a routing perturbation-based defense to increase the Hamming distance. The authors use layer elevation, routing detours, and wire decoys to achieve the optimum Hamming distance. In parallel, test principles are used to choose the perturbations. Similarly to the technique proposed in [138], in [140], the authors proposed placement-based defense. However, differently from the pin swapping in [138], they consider the incurred wirelength overhead as a metric. On top of that, they also perform a logic-driven perturbation with a weighted logical difference (WLD) metric, which incurs a sizeable logical difference from its neighbors. Considerably different from the other proximity perturbation techniques, in [156] are proposed modifications on the netlist instead of placement/routing during physical synthesis. These modifications have the purpose of inserting partial randomization, and later the proper functionality is restored in the BEOL with the help of correction cells that resemble switch boxes. Alternatively to Hamming distance, the proximity perturbation technique proposed in [153] utilizes an information-theoretic metric to increase the resilience of a layout against proximity attacks. According to [153], the amount of information revealed by the distance between the exposed pins can be quantified using mutual information (MI). Then, applying a placement-driven technique minimizes the amount of exposed information quantified by MI.

The wire-lifting technique approaches the insecurity problem differently than proximity perturbation. As previously explained, splitting the circuit at lower metal layers increases the Split Manufacturing security level. Following the same idea, wire lifting proposes moving wires from the FEOL layer to the BEOL. Thus, increasing the number of exposed pins and potentially increasing the security level.

Table 6: Results for Defense Techniques based on Proximity Perturbation.

Work	Attack Type	Benchmark	Defense Technique	Defense Metric	Defense Overhead	Split Layer	Result without Defense	Result with Defense
[138]	Proximity	c17	-	Hamming Distance	1 Swap for 50% HD	-*	100% CCR	78% CCR
[138]	Proximity	c7552	-	Hamming Distance	49 Swaps for 50% HD	-*	94% CCR	91% CCR
[154]	Proximity	c432	Modified Greedy Gate Swapping	EMSR	75% of WLO	-*	90% EMSR	25% EMSR
[154]	Proximity	c432	Modified Greedy Gate Swapping	EMSR	300% of WLO	-*	78% EMSR	10% EMSR
[151]	Proximity	c432	-	Hamming Distance	3.1% WLO for 46.1% HD	-*	92.4% CCR	78.8% CCR
[151]	Proximity	c432	-	Hamming Distance	4.1% WLO for 31.7% HD	-*	62.8% CCR	37.9% CCR
[153]	Proximity	c432	Random	Mutual Information	$<$ 10% PPA	M1	17% CCR	$<$ 1% CCR
[153]	Proximity	c432	g-color	Mutual Information	$<$ 10% PPA	M1	17% CCR	2% CCR
[153]	Proximity	c432	g-type1	Mutual Information	$<$ 10% PPA	M1	17% CCR	6% CCR
[153]	Proximity	c432	g-type2	Mutual Information	$<$ 10% PPA	M1	17% CCR	4.5% CCR
[153]	Proximity	c7552	Random	Mutual Information	$<$ 10% PPA	M1	13% CCR	$<$ 1% CCR
[153]	Proximity	c7552	g-color	Mutual Information	$<$ 10% PPA	M1	13% CCR	2% CCR
[153]	Proximity	c7552	g-type1	Mutual Information	$<$ 10% PPA	M1	13% CCR	4% CCR
[153]	Proximity	c7552	g-type2	Mutual Information	$<$ 10% PPA	M1	13% CCR	3% CCR
[140]	SAT	c432	BEOL+Physical	Perturbation	4.5% WLO	-*	58% CCR	56% CCR
[140]	SAT	c432	Logic+Physical	Perturbation	5.57% WLO	-*	58% CCR	58% CCR
[140]	SAT	c432	Logic+Logic	WLD	1.68% WLO	-*	58% CCR	52% CCR
[140]	SAT	b18	BEOL+Physical	Perturbation	8.06% WLO	-*	15% CCR	14% CCR
[140]	SAT	b18	Logic+Physical	Perturbation	1.70% WLO	-*	15% CCR	17% CCR**
[140]	SAT	b18	Logic+Logic	WLD	0.61% WLO	-*	15% CCR	16% CCR**
[156]	Proximity	c432	Netlist Randomization	Hamming Distance	$<$ 10% PPA overall	-*	92.4% CCR	0% CCR
[156]	Proximity	c7552	Netlist Randomization	Hamming Distance	$<$ 10% PPA overall	-*	94.4% CCR	0% CCR

* Split layer not specified by the authors.

** These results are counter-intuitive, the applied defense degrades the metric.

In [135], wire lifting was first presented considering Split Manufacturing as a 3D IC implementation [159]. However, their technique is analogous to Split Manufacturing, even the notion of untrusted FEOL vs. trusted BEOL. Their implementation consists of two or more independently manufactured ICs, where each IC represents a tier that is vertically integrated. For integrating the tiers, vertical metal pillars are used – referred to as through-silicon vias (TSVs). In [135], their 3D implementations comprise two tiers; the bottom tier consists of the transistors and some routing wires (same as the FEOL); the top tier consists of only routing wires. However, both tiers are manufactured in untrusted foundries. Nonetheless, the authors in [135] provide a security notion based on existing multiple mapping between gates in the unlifted and complete netlists. Referred to as *k-security*, this metric qualifies that gates across the design are indistinguishable from at least $k-1$ other gates. Thus, a defender wants to lift wires in a way to guarantee the higher $k-security$ possible. Two procedures are proposed to achieve this goal, one utilizing a greedy heuristic targeted at small circuits (due to scalability issues) and another that utilizes partitioning to solve those issues.

Now utilizing standard Split Manufacturing, in [139] the authors proposed artificial routing blockage[7] to promote wire lifting. Since commercial EDA tools are built to provide the best PPA possible, it routes signals preferably in lower metals. Hence, the insertion of routing blockages can force some signals to be routed above the split layer. The result is an artificial wire lifting done during the routing stage.

The authors in [152] argued that previous wire-lifting works have largely neglected Design for Manufacturability (DFM) concerns (i.e., lithography checks, critical feature analysis, pattern matching, and others). Thus, the authors in [152, 160] proposed two DFM-related wire-lifting techniques; (1) Chemical Mechanical Planarization (CMP); (2) Self-Aligned Double Patterning (SADP) [161]. The first technique, CMP-friendly routing defense, is divided into layer elevation, wire selection, and re-routing. For that, wires located in dense regions are selected to be re-rerouted in sparse areas. The second is SADP-compliant, wire-lifting, and re-routing, disregarding the density of the regions, with the solemn purpose of extending the wire's length [162]. Moreover, according to [152], solving SADP violations by wire extension can also increase security, increasing the distance between vias.

To avoid the PPA overhead introduced by wire-lifting-based defenses, the authors in [155] proposed a cost-security trade-off approach, i.e., PPA margins for a given security budget. The authors claim that their concerted wire-lifting method enables higher degrees of security while being cost-effective. They utilize elevating cells for lifting the wires together with three strategies: lifting high-fanout nets, controlling the distance for open pin pairs, and obfuscating short nets.

Both proximity perturbation and wire-lifting try to hide hints of hidden connection at FEOL from the attackers. However, even without knowing where all connections are, an attacker can identify regular structures just by looking at the FEOL layout, perchance leading to easier attacks. For hiding those regular structures, layout obfuscating is used to make them indistinguishable.

[7] This terminology is used in IC design to mean that a specific area should be avoided by the EDA tool for a specific task. A blockage can be for placement and/or for routing.

Table 7: Results for Defense Techniques based on Wire Lifting.

Work	Attack Type	Benchmark	Defense Technique	Defense Metric	Defense Overhead	Split Layer	Result without Defense	Result with Defense
[135]	SAT	c432	Wire Lifting	*k-security*	477% of WLO	-*	k=1	k=48
[139]	Proximity	Superblue 1	Routing Blockage Insertion	$E[LS]$	Not Presented	M4	1.51	1.77
[139]	Proximity	Superblue 1	Routing Blockage Insertion	FOM	Not Presented	M4	1222.8	1433
[155]	Proximity	c432	Concerted Lifting	Hamming Distance	7.7% of Area	Average**	23.4	45.9
[155]	Proximity	c432	Concerted Lifting	CCR	13.2% of Power	Average**	92.4	0
[155]	Proximity	c7552	Concerted Lifting	Hamming Distance	16.7% of Area	Average**	1.6	25.7
[155]	Proximity	c7552	Concerted Lifting	CCR	9.3% of Power	Average**	97.8	0
[152]	Proximity	c2670	CMP-Friendly	Hamming Distance	3.4% of WLO	-*	14.5%	20.4%
[152]	Proximity	c2670	CMP-Friendly	CCR(%)	3.4% of WLO	-*	48.1%	33.4%
[152]	Proximity	b18	CMP-Friendly	Hamming Distance	0.4% of WLO	-*	21.6%	27.6%
[152]	Proximity	b18	CMP-Friendly	CCR(%)	0.4% of WLO	-*	12.1%	10.7%
[152]	Proximity	c2670	SADP-Compliant	Hamming Distance	7.49% of WLO	-*	14.5%	24.4%
[152]	Proximity	c2670	SADP-Compliant	CCR(%)	7.49% of WLO	-*	48.1%	6.4%
[152]	Proximity	b18	SADP-Compliant	Hamming Distance	4.64% of WLO	-*	21.6%	29.6%
[152]	Proximity	b18	SADP-Compliant	CCR(%)	4.64% of WLO	-*	12.1%	2.7%
[150]	Proximity	s526	Net Partitioning	CCR(%)	Not Presented	-*	40%***	0%***
[150]	Proximity	s526	Net Partitioning & Cell Hiding	CCR(%)	Not Presented	-*	40%***	0%***
[150]	Proximity	s526	Net Partitioning & Cell Hiding & Pin Shaking	CCR(%)	Not Presented	-*	40%***	0%***
[150]	Proximity	s9234.1	Net Partitioning	CCR(%)	Not Presented	-*	30%***	4%***
[150]	Proximity	s9234.1	Net Partitioning & Cell Hiding	CCR(%)	Not Presented	-*	30%***	1.5%***
[150]	Proximity	s9234.1	Net Partitioning & Cell Hiding & Pin Shaking	CCR(%)	Not Presented	-*	30%***	1.5%***

* Split layer not specified by the authors.
** Results are given as an average between M3, M4, and M5.
*** These results cannot be directly compared with previous ones as the transistor technology is vastly different.

As described in Section 2.2, design companies often use 3PIPs in their ICs, both soft and hard IPs. Soft IPs usually come in code form, giving the task of implementing to the customer. However, it also gives the customer flexibility to modify the IP to meet their needs. Therefore, soft IPs are not challenging in a Split Manufacturing design flow paradigm. On the other hand, hard IPs are entirely designed by the vendor and are technology-dependent.

The security of hard IPs in a Split Manufacturing context is analyzed in [132]. To assess security, the authors proposed a recognition attack flow: an attacker holding the FEOL layer starts his attack by isolating a target embedded memory or analog hard IP. From the knowledge of recognizing leaf cells utilizing layout. Since the targeted hard IP has a high probability of being constructed by compilation of leaf cells, layout pattern recognition software [163] can be used for trivial leaf-cell identification. Then, the attack combines this knowledge with proximity hints to improve the proximity attack's effectiveness. As demonstrated in [132], embedded memories, such as SRAM, are susceptible to the proposed recognition attack. Defending against recognition attacks can be achieved by employing layout obfuscation.

Because of the potential success of recognition attacks, many authors proposed layout obfuscating to improve the resilience of Split Manufacturing [132, 147–149, 157, 158]. In [132], the authors proposed a synthesis framework flow for obfuscating SRAM and analog IP. Their synthesis flow has three goals to achieve layout obfuscation: randomizing periphery cells, thus avoiding predictable; minimizing regularized topologies used for peripheral circuits such as pre-decoders, word line decoders, and sense amplifiers; adding non-standard application-specific functions to improve obfuscation and performance. Moreover, in [147] proposed four techniques for layout obfuscation, (1) limited standard-cell library, (2) smart-dummy cell insertion, (3) isomorphic cells, and (4) non-optimal cell placement. Their goal is to increase Time To Evaluate (TTE). The authors in [147] argue that if a TTE is high enough, an adversary would be discouraged from reverse engineering the IC.

The other layout obfuscation techniques are presented in Table 5, following the same principle described above. Finally, for a complete discussion and results presentation, we direct the reader to [22].

4.4 Discussion

Despite our effort to present the results of the many studied papers in the fairest way possible, it is clear that the hardware security community lacks a *unified benchmark suite* and/or a *standard criteria* for assessing results. Instead, researchers often use benchmark suites that are popular in the Test community but have no real applicability in security. For example, most benchmark suites (e.g., ISCAS'85) used for assessing Split Manufacture have no crypto cores, which are fundamental for security research. In [164], the authors proposed a game-theoretic framework to evaluate the existing Split Manufacturing attacks and defenses. The authors concluded that larger circuits are secured by naïve Split Manufacturing. Hence, larger circuits do not require additional defense mechanisms. Consequently, the community would primarily benefit from using circuits that better represent the IC design practices of this decade, where IPs often

have millions of gates, and ICs have billions of transistors.

It is noteworthy to mention the disparity in the attack models proposed so far. As previously pointed out, threat model II is too strong, almost nullifying any secure sense introduced by Split Manufacturing. However, the real problem is how complicated is defining a threat model to establish the attacker's capabilities in the best manner possible. By definition, formalizing the capabilities of an attacker requires understanding his motivations, technical proficiency, and availability of resources. In threat models that underestimate the attacker's capabilities, useless defense strategies can be devised and assumed to be effective. On the other hand, in case the attacker's capabilities are overestimated, convoluted defense strategies might be employed, leading to unnecessary PPA overheads. Thus, defining a precise threat model is a challenge for Split Manufacturing and many other techniques that promote obfuscation.

Another topic that has led to no consensus is whether an attacker can use a partially recovered netlist. For instance, let us assume a design that instantiates the same block multiple times. If one of the blocks is correctly recovered, a cursory inspection of the structure may allow the attacker to recover all other instances of the same block. The same line of thinking can be applied to datapaths and some regular cryptographic structures. An analysis of the functionality of the recovered netlist could be combined with existing attacks for further improvement of correctly guessed connections.

Many of the works studied in this survey have yet to demonstrate their approach in silicon – only 15% have a silicon demonstration. Hence, the hardware security community should strive to validate not only Split Manufacturing techniques but many other security approaches in silicon as often as possible. In the case of Split Manufacturing, however, finding two foundries willing to diverge from their established practices could be next to impossible. For this reason, only a small percentage of the reported works have validated their techniques in silicon.

5 Hardware Trojans Design and Insertion

This Chapter discusses the hardware trojan threat during IC manufacturing. The literature has many hardware trojan demonstrations, a few even in silicon; however, not a single one disclosed how their hardware trojan is inserted. Hence, in this Chapter, I will demonstrate a full framework for designing and inserting hardware trojans in finalized layouts. To validate this framework, I developed a silicon prototype comprising four crypto cores altered with a hardware trojan. For inserting the hardware trojans, I leverage the engineering change order (ECO) feature, which is readily available in commercial EDA tools. Furthermore, I propose a side-channel trojan capable of leaking multiple bits into a single power signature reading to demonstrate the capabilities of the proposed ECO framework. Finally, a reverse engineering technique is discussed to find critical nodes to connect the hardware trojans.

This Chapter has its content based on the following publications:

[II] T. Perez, M. Imran, P. Vaz, and S. Pagliarini, "Side-channel trojan insertion - a practical foundry-side attack via eco," in 2021 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5, 2021

[III] T. Perez and S. Pagliarini, "A side-channel hardware trojan in 65nm cmos with 2μW precision and multi-bit leakage capability," in 2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 9–10, 2022

[V] A. Hepp, T. Perez, S. Pagliarini, and G. Sigl, "A pragmatic methodology for blind hardware trojan insertion in finalized layouts," in 2022 International Conference on Computer-Aided Design (ICCAD), 2022

[VI] 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 (TCAD), 2022

5.1 Introduction

Because of the current IC supply chain organization, as discussed in depth in Chapter 2, the trustworthiness of an IC can be potentially affected – a foundry (or a *rogue element* within the foundry) could manipulate the design for their own malicious purposes [59]. Hence, the IC is exposed to the many fabrication-time attacks studied in recent decades [165]. Furthermore, the many defense techniques to combat these threats discussed in the previous Chapters are not suitable for the large-scale production of ICs. Because of either practically [22] and/or insufficient security guarantees [16].

One of the many fabrication-time attacks an IC can potentially suffer is a malicious modification, i.e., an HT [16, 73]. For example, HTs (see Section 2.3) are designed to leak confidential information, disrupt a system's specific functionality, or even destroy the entire system (referred to as a time bomb). Various HTs have been studied recently [64–67, 69, 70], demonstrating the potential threat of this type of attack.

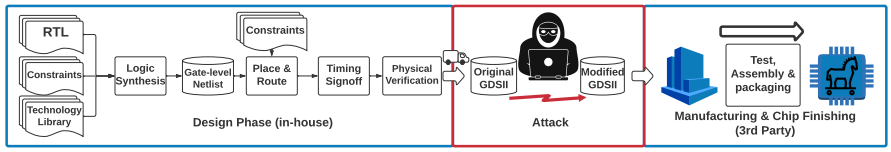

Figure 27: A typical IC design flow. Highlighted in red is the stage where a rogue element may mount an attack (modified from [17]).

Moreover, an HT can be specialized to assist SCAs – often referred to as side-channel HTs (SCTs). The first side-channel HTs for assisting power SCAs was proposed in [64], called "Malicious Off-chip Leakage Enabled by Side-channels" (MOLES). In this Chapter, I propose a new SCT for assisting power SCAs. With the aid of SCTs, power SCAs can immensely reduce their attack time as no further processing is required. However, the disadvantage of SCTs is their invasive nature. Inserting an SCT requires a modification of the circuit at fabrication time. Modifying a finalized layout might seem challenging; however, a capable attacker can perform it, as demonstrated later in this Chapter by the proposed framework to insert the SCT or any additional HT in highly dense designs without hindering the target circuit. On top of that, the framework utilizes EDA tools to automate the insertion of additional malicious logic, hence, DRC-aware and having a relatively fast runtime.

The proposed SCT attack utilizes a similar model as the *threat model I*, described in Chapter 4. However, with a few modifications. The principal adversary is also a rogue element inside the untrusted foundry utilized by a design company to manufacture their designs. In this attack, the adversary aims to o insert malicious logic into the finalized layout handed over by the victim. Here I emphasize that the attack occurs before the fabrication (see Figure 27), and a single rogue element inside the foundry is sufficient to perform the proposed attack. The foundries or a few companies licensed by the foundries provide the standard cell library to the design companies to implement their designs. Therefore, it is assumed that an attacker inside the foundry can access all technology and cell libraries and distinguish individual gates and their functionality.

Additionally, in the SCT attack threat model, the attacker can identify the presence of a crypto core in a layout, which is a reasonable assumption, especially for well-known AES implementations that display regularity (due to the round-based key schedule structure). Finally, notice that to perform such an attack, the adversary does not need to understand the entire victim's design, nor is there a need for it. Instead, it is assumed he/she needs only to recognize the layout/structure of a single crypto core within a larger design, in line with the assumptions made in [65, 67].

Furthermore, the adversary also: 1) is versed in IC design, 2) enjoys access to modern EDA tools, and 3) has no means to make radical modifications to the circuit (e.g., adding new IOs or making changes in the clock domains). Therefore, with the help of the inserted logic in the form of an SCT, the attacker will attempt to leak confidential information via a power signature. The preferred target of this type of attack is crypto cores [67, 68]; hence, this is also the choice of target to demonstrate our SCT insertion framework. As the proposed SCT attack deals with power signature reading, stopping

some part of the clock delivery, or even entirely, would be highly beneficial for the attack. However, the attacker is assumed to not know about the clock domains or clock distribution in general. Therefore, synchronizing and controlling the HT's trigger to stop the clock delivery is not considered feasible for the SCT threat model, nor is the addition of an external trigger controlled by an IO. Thus, the attacker has no direct access to the trigger or payload of the trojan.

A typical IC physical implementation flow is described in the left portion of Figure 27. The attack occurs after the victim's layout in GDSII format is sent for fabrication (see the red portion of Figure 27). Suppose the attacker had access to all of the victims' data required to generate the layout (i.e., RTL, netlists, constraints, and many others). In this case, he/she could replicate the physical implementation flow to achieve a layout similar to the one created by the victim, yet now containing his malicious logic. This effort is theoretically possible but largely unpractical. Although replicating the physical implementation is possible, this scenario is not a threat model considered in the literature. Finally, the SCT attack threat model assumes that the attacker only has access to the finalized layout. Design companies have to hand in their finalized layout to the foundry for fabrication. Usually, the layouts require some pre-processing steps before the start of the fabrication, which a foundry employee handles. Thus, it is during this period that the attack can be mounted.

5.2 Side-Channel Trojan and its Insertion via ECO

The proposed SCT architecture is an additive hardware trojan to aid a side-channel attack with a digital sequential synchronous event trigger and a digital payload that drive nodes (see Figure 12). The SCT architecture is designed to create artificial power consumption, which can leak sensitive information through this extra induced power. In order to retrieve the leaked bits, the SCT has to create the extra power in a controlled manner. Because the most significant portion of an IC's power consumption comes from the switching activity (dynamic power), a great candidate to be a controlled power sink is a structure with a controllable frequency of operation.

An example of a power sink with a controllable frequency of operation is a ring oscillator (RO) with dynamically adjustable stages, as illustrated in Figure 28. The RO delay stages of the proposed architecture are broken into branches controlled by N_{leak} leaking bits. Each branch has two active paths: a direct connection to the next branch or a series of delay cells. Therefore, each set of N_{leak} leaking bits has a specific power consumption increment. This artificial power consumption created by the RO is similar to a pulse-amplitude modulation technique, with an order equal to $2^{N_{leak}}$. The architecture illustrated in Figure 28 is an example of the proposed RO architecture capable of leaking **two** bits per power signature reading, i.e., $N_{leak} = 2$. The active paths' configuration is described in Table 8, where the leaking bits become branch selectors and are referred to as S0 and S1.

An attacker has to design our SCT with a dual-sided constraint in mind: (1) the induced dynamic power consumption has to be large enough to retrieve the leaking bits while (2) minimizing the increase in leakage power. The first constraint is regarding the effectiveness of the attack; the largest the induced power amplitude, the easiest it

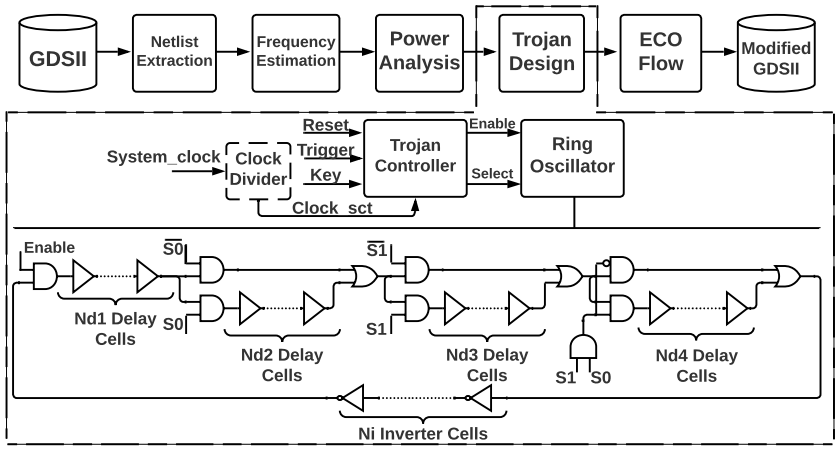

Figure 28: The proposed trojan insertion methodology for an SCT capable of leaking 2 bits per power signature reading (modified from [17]).

Table 8: Ring oscillator active path configuration

S0	S1	Delay Cells	Inverter Cells	Freq.
0	0	N_{D1}	N_i	High
1	0	$N_{D1}+N_{D2}$	N_i	Mid-high
0	1	$N_{D1}+N_{D3}$	N_i	Mid-low
1	1	$N_{D1}+N_{D2}+N_{D3}+N_{D4}$	N_i	Low

is to retrieve the leaking bits. The second is regarding the SCT detection by the chip owner; as the SCT is an additional HT, its presence increases leakage power directly proportional to the SCT size. Dynamic power can be calculated using equation (4), where C_{load} is the capacitance load at the output nets, F_{sa} is the switching activity factor, V_{DD} is the supply voltage, and E is the total energy of a cell. The switching activity factor describes how many switches will occur per second. As for the RO, since the signals are constantly switching, this factor is two times the RO's oscillation frequency, which can be estimated by calculating the total path delay of the ring as in equation (5).

$$P_{dynamic} = \frac{1}{2}{V_{DD}}^2 F_{sa} \sum_{i_{net}} C_{load}(i) + F_{sa} \sum_{cell_j} E(j) \tag{4}$$

$$F_{sa} = 2F_{RO} = \frac{1}{\tau_{chain}} \tag{5}$$

Moreover, in addition to the carefully designed RO-based SCT structure, the SCT trigger must be accordingly planned. For example, in the proposed SCT architecture, the trojan is not allowed to compete with the dynamic power consumption of the crypto core – the SCT triggers right after the crypto core finishes its cryptographic operation. For this reason, our SCT has a trigger signal that is connected to the "done" signal coming from the crypto core.

As the SCT is designed for a specific target layout, the attacker has to perform a few analyses before, as illustrated in Figure 28: (1) netlist extraction, (2) frequency estimation, and (3) power analysis. First, in (1), the attacker has to extract the gate-level netlist from the victim layout [136] – our threat model considers the attacker only holds the layout. Then, with the gate-level netlist on hand, in (2), the attacker has to estimate the operating frequency of the target circuit by performing STA [20]. Finally, in (3), the attacker can perform a typical power analysis with the knowledge of the operating frequency and the gate-level netlist. For relatively large circuits, static power can be estimated very precisely even without input vectors[8].

With the SCT designed accordingly with the target circuit, the next step is its insertion. Then, the attacker can utilize the pre-mask ECO feature provided by commercial EDA tools for inserting the SCT. The primary purpose of ECO is to fix minor bugs in a finalized layout instead of re-implementing the whole design. Hence, saving a tremendous amount of runtime to finalize a given design – essential for design companies where time-to-market is crucial. However, this feature is leveraged in the proposed ECO framework to insert malicious logic rather than fix bugs. I emphasize that no EDA vendor supports this type of usage of the ECO feature. In addition, the pre-mask ECO does not require special cells (e.g., space cells) and is a one-time operation. For more information about ECO and its features, I direct the reader to [20].

Nonetheless, for the SCT insertion via ECO, an attacker can achieve his/her goal without utilizing spare cells. Since we previously established that the attacker could discern any gate in a layout, he can replace filler and spare cells for his malicious logic. Contrarily to spare cells, every digital circuit layout has filler cells. During placement, EDA tools have to spread the standard cells to assure routability, thus mandatorily leaving gaps between cells. For more details about the relationship between placement density and HT insertion, we direct the reader to [166].

After the ECO, the attacker has to perform timing sign-off to guarantee that the performance of the victim's design was not disturbed. The SCT insertion is not likely to perturb the target's performance; it is only connected to a register (crypto key storage) and some control signals, adding a small capacitive load. Besides, the coupling capacitance inserted by the additional routing wires is minimal due to the SCT's lightweight characteristic and the inherent goal of the ECO flow: *not to disturb the existing logic*. However, even if unlikely, the addition of the SCT could hinder the target performance. Since the ECO makes this attack relatively fast, the attacker can try different SCT architectures until he/she finds a suitable trojan for their target circuit.

5.3 Testchip: Results and Discussion

For the experimental investigation, I have utilized AES-128 and Present (PST) [167] crypto cores with $N_{key} = 128$ and $N_{key} = 80$, respectively. The AES crypto core was chosen due to its standardized status and popularity, while PST was chosen due to its lightweight characteristic [168].

[8]For crypto cores, in particular, it is a fair assumption to consider the plaintext to be randomly assigned, the adversary does not need precise vectors to estimate the (order of magnitude) of the power consumption.

In order to demonstrate the potential malicious capabilities of the ECO flow (see Figure 28), I designed a silicon proof of concept comprising four crypto cores altered with the proposed SCT. The SCTs utilized for the chip are carefully crafted to stress test the ECO flow and its limitations: the chosen circuits are synthesized for their maximum frequency and challenging densities, making the SCT insertion even more challenging. The proposed framework includes all steps necessary for assessing the GDSII database, designing a hardware trojan, and inserting it in a finalized layout.

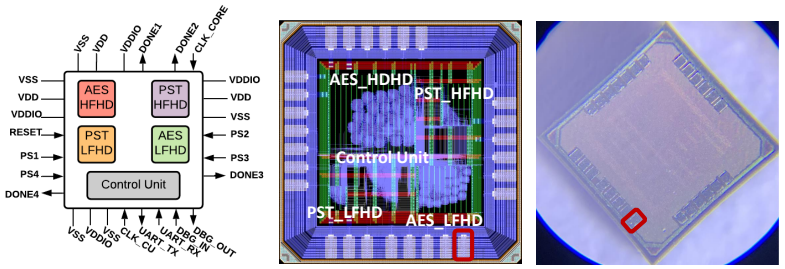

Figure 29: ASIC prototype top-level diagram (left), layout (middle), and its bare die (right). The highlighted pin identifies the lower-right corner in red (adapted from [17]).

Figure 29 illustrates the top level of the chip, containing the four crypto cores and a control unit for handling the data traffic in and out of the chip. The crypto cores are the AES High-Frequency-High-Density (AES_HFHD), AES Low-Frequency-High-Density (AES_LFHD), PST High-Frequency-High-Density (PST_HFHD), PST Low-Frequency-High-Density (PST_LFHD). The signals UART_TX and UART_RX are utilized for communicating with the control unit. In addition, the signals DONE_1, DONE_2, DONE_3, and DONE_4 indicate the end of a cryptographic operation for AES_HFHD, AES_LFHD, PST_HFHD, and PST_LFHD, respectively. These signals are exposed as primary outputs only for debug reasons; their presence is not required for the attack. Internally, these same signals are the triggers for the SCTs. To help the reader better visualize the operation of the SCT, Figure 30 illustrates a SPICE simulation of the SCT using the AES_LFHD target as an example. The set of leaked keys in the image is $\{00\text{-}01\text{-}10\text{-}11\}$. The RO operating frequency and power results are from a SPICE-level simulation with parasitics, and the total power of the AES_LFHD is estimated from physical synthesis.

Similarly to the G-GPU, each crypto core is power gated using coarse-grain header power switches inserted in a column fashion (see Figure 18), with the power switch "enable" controlled by the signals *PSx*. Implementing the crypto cores with the possibility of total shut-down is extremely valuable for evaluating our attack because we only read the power signature from the enabled core.

A different RO is designed for each crypto core according to its physical characteristics described in Table 9. In Table 9, the results are separated into before and after SCT insertion, where the design density, leakage, clock-tree (CT) power, and total power are reported. To design the ROs for the ASIC prototype is utilized before SCT insertion results. In the proposed ROs, the maximum power step generated by a RO is 10% of the leakage plus CT power. Note that this percentage is not a hard constraint nor a

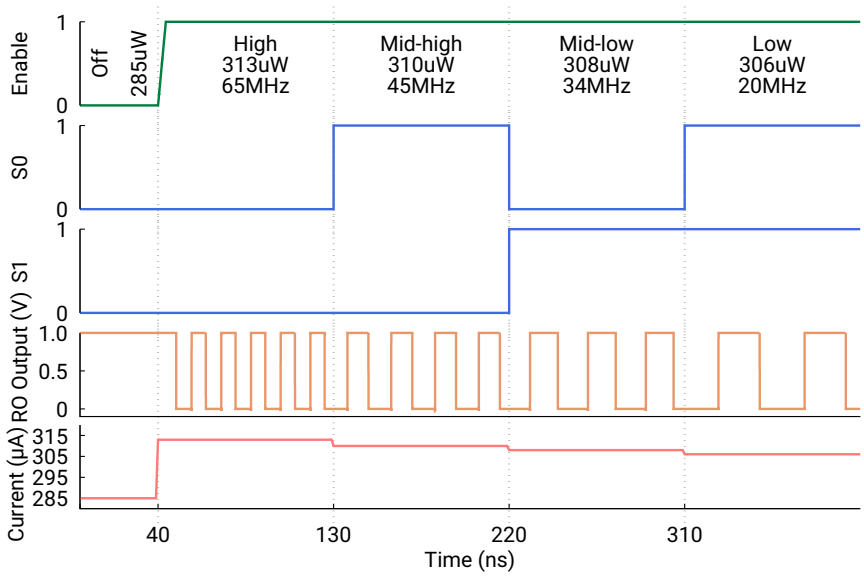

Figure 30: Post-layout simulation of SCT architecture in Cadence Spectre. The target design is AES_LFHD and the Trojan payload is configured as RO_{D6I10} (from [20]).

Table 9: Physical synthesis results for our considered targets, before and after trojan insertion.

Core	Frequency (MHz)	Before SCT insertion Density (%)	Leakage (μW)	CT (μW)	Total Power (μW)	After SCT insertion Density (%)	Leakage (μW)	CT (μW)	Total Power (μW)
AES_LFHD	100	75	75.8	116.7	1660	78.20	79	117.6	1720
AES_HFHD	1000	72	1036	1241	22610	73.02	1040	1252	22830
PST_LFHD	95	70	14.09	31.89	371.2	82.05	17.72	32.85	428.5
PST_HFHD	950	69	34.13	329.10	3785	80.26	36.96	341.5	4015

limitation of the proposed architecture; attackers can choose any reasonable threshold value to design their ROs. However, the 10% margin is arguably a good trade of capability of leaking the bits and stealthiness. The designed ROs for the ASIC prototype are described in Table 10, reporting the oscillation frequency and power consumption of each designed RO, where the RO name"DXIY" suffix represents X amount of delay cells and Y amount of inverter cells. These results are from detailed SPICE-level simulations. Most importantly, Table 10 shows that the induced power step separation is clearly visible in increments of a few microwatts; thus, the leaking bits can indeed be modulated in the power consumption of the chip.

After designing the RO and synthesizing the remainder of the SCT logic, the attacker is ready to perform the insertion via the ECO methodology described in Figure 28. For the ASIC prototype, the ECO flow was completed in a single run, i.e., calling the ECO command a single time. The results for SCT insertion are described on the right side of Table 9 ('After SCT insertion'). For all scenarios considered, the ECO flow could successfully place and route the SCT, even for highly dense layouts. A visual comparison of the density increase for the AES_HFHD and PST_HFHD SCTs is given in the bottom part of Figure 31. Note that the placement of the targets (top part of Figure 31) was kept identical, and only filler cells were removed for the SCT insertion

Table 10: RO operating frequency and power consumption from a SPICE-level simulation for four variants of AES and PST.

Target Core	RO	Power & Frequency (μW & MHz)			
		S=00	S=01	S=10	S=11
AES_LF	RO_{D6I10}	19@65	17@45	15@34	13@20
AES_HF	RO_{D10I10}	198@551	182@483	161@390	140@300
PST_LF	RO_{D6I4}	16@112	11@58	10@39	8@20
PST_HF	RO_{D8I10}	42@79	36@61	31@46	26@31

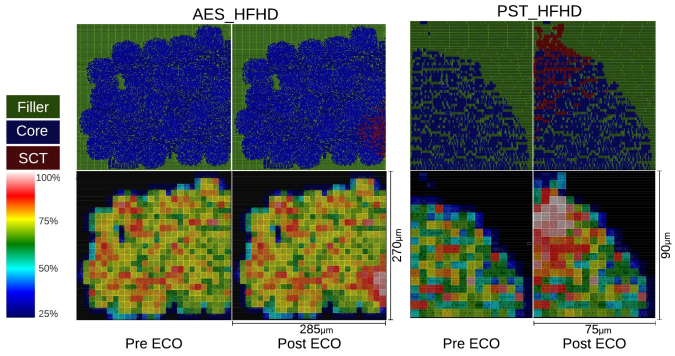

Figure 31: Placement view (top panels) and density map (bottom panels) of the AES_HFHD and PST_HFHD cores, before and after SCT insertion via ECO (modified from [17]).

via ECO. Therefore, this is a key finding of our work and confirms the feasibility of the attack.

Aside from being able to insert the SCT, the ECO flow also has to preserve the performance of the target circuit. As discussed in Chapter 2.2, the coupling capacitance from adjacent routing wires affects the propagation delay. Thus, the added routing wires from the SCT could negatively impact the target circuit's overall performance. The comparison of performance for AES_HFHD and PST_HFHD cores is illustrated in Figure 32, where we contrast the pre- and post-ECO timing slack. These results show that the impact is more significant on the PST_HFHD implementation, which is explained by the high-density increase reported in Table 9. Therefore, the impact of the SCT insertion did not degrade the crypto core's performance. Finally, the chip was manufactured utilizing commercial 65nm technology at a partner foundry in March

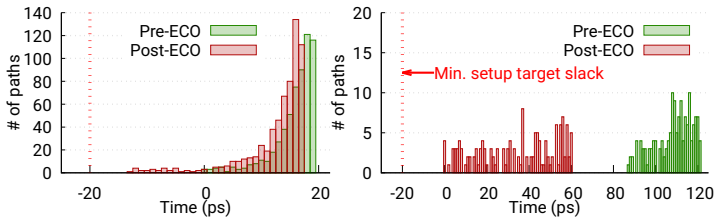

Figure 32: Pre- and post-ECO setup timing slack comparison of AES_HFHD (right) and PST_HFHD (left) (from [17]).

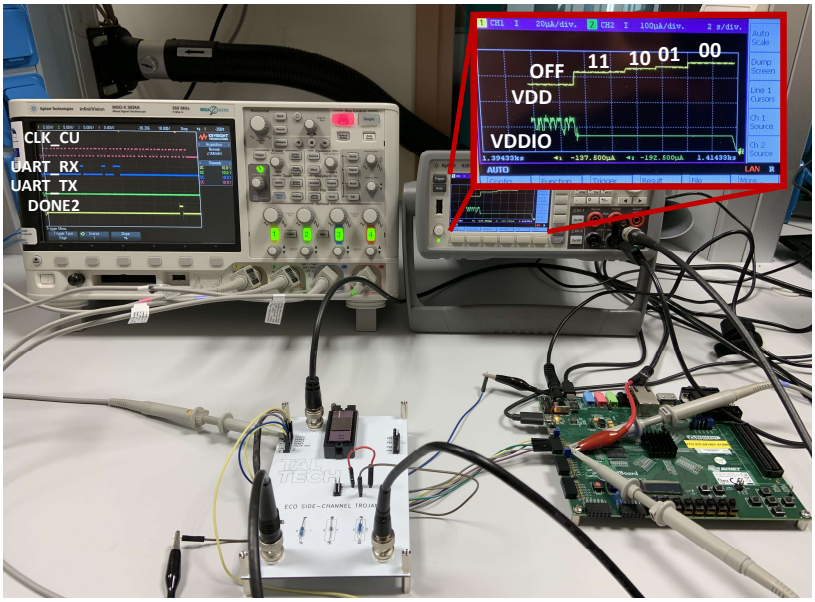

Figure 33: Setup used for bringing up the testchip. On the left side, we show the signals used for controlling the chips. On the right side, the current consumption of the chip when the RO is active (from [20]).

2021. The bench tests of the 25 packaged samples of the chip were conducted in July 2021. All packaged samples were confirmed to be 100% functional.

The testchip bench tests were performed utilizing the setup illustrated in Figure 33. The setup has a custom printed circuit board (PCB), a ZedBoard from Avnet with a Xilinx Zynq-7000 (see Figure 15), a 4-channel digital oscilloscope, and a 2-channel power supply with an ammeter with pico ampere precision. To fully validate the chip, the tests are divided into two phases: the total power and leakage were measured; second, all SCTs were tested to assess the feasibility of the attack. As a result, the total power average and leakage results are given in Table 11, and its distribution across the samples is depicted in Figure 34 for the worst, typical, and best-case scenarios (SS-0.9V-0^{o}C, TT-1V-25^{o}C, FF-1.1V-125^{o}C, respectively). Corners provided by the vendor are for extreme cases, i.e., the best-case scenario is characterized at 125^o with an over voltage of 1.1V; in this work, the test bench measurements were performed at room temperature and at a nominal voltage of 1.0V. In Figure 34, it is clear that the samples are skewed towards the best-case scenario, demonstrating higher average

Table 11: Power domains, clock, average total power, and leakage across the samples tested.

Block	Clock	Switch Signal	Leakage (μW)	Total Power (μW)
Control Unit	CLK_CU @1MHz	Always on	46.69±4.75	-
AES_HFHD	CLK_CORE @1GHz	PS1	743.79±108.07	101160±10781
AES_LFHD	CLK_CORE @100MHz	PS2	131.57±10.35	3139.32±85.38
PST_HFHD	CLK_CORE @950MHz	PS3	80.75±7.82	9661.3±758.52
PST_LFHD	CLK_CORE @95MHz	PS4	74.35±6.84	868.56±57.90

leakage. The slowest sample is near the typical case, while the fastest sample is far from the typical best case.

For testing the SCTs, the following procedure was performed: (1) a cryptokey with the 8 first bits set to "11-10-01-00" was programmed in the Control Unit's register bank; (2) a command for single encryption was issued; (3) right after the encryption is done, the chip asserts one of the "DONE" outputs to mark the time at which the RO starts operating; (4) using only the clock signal CLK_CORE, three bursts of clocks were sent to shift the cryptokey connected to the RO three times; (5) during the whole procedure, the current consumed by the chip is monitored.

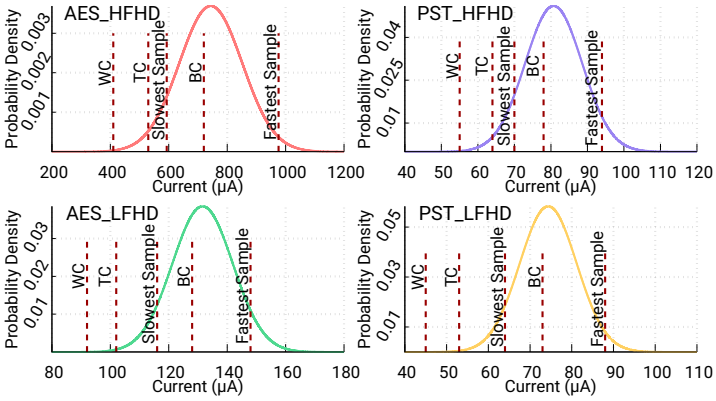

Figure 34: Leakage distribution for each crypto core contrasted with the leakage from the physical synthesis report for three corner cases and the leakage of outlier samples (from [20]).

Figure 33 illustrates an example of the procedure described above for the AES_LFHD core. As clearly depicted in the ammeter, there are discrete steps representing the leaked bits "11-10-01-00" from left to right, respectively, as expected from the key programmed for this experiment. Next, each chip's core was tested following the described procedure three times to confirm the behavior. Repeating the measurements is a common practice to reduce undesired external interference – three repetitions are deemed enough. Finally, the measured current values were approximated to normal distributions, as represented in Figure 35.

Comparing the RO performance from the simulations (see Table 10) with ASIC measurements illustrated in Figure 35, it is clear that the slowest ROs are performing as expected. However, the fastest RO targeting the AES_HFHD core can only operate at a low frequency, generating a power step of about 25% of what was expected. In this case, the ECO insertion had to spread the RO cells farther away because of the lack of empty spaces nearby (see Fig. 31). For this core, the planned power steps were in the order of 200 μA, and the actual power steps after manufacturing were in the order of 60 μA. However, the attack will still enjoy a high chance of success due to the distinct separation of the power steps, even if 95% confidence intervals of the distributions almost overlap. Moreover, the experimental measurement results obtained show that the variability in the manufacturing process does not affect the effectiveness of the RO for the smaller designs (AES_LFHD, PST_LFHD, and PST_HFHD), meaning that the attack can be carried out with the same probability of success, regardless of the silicon

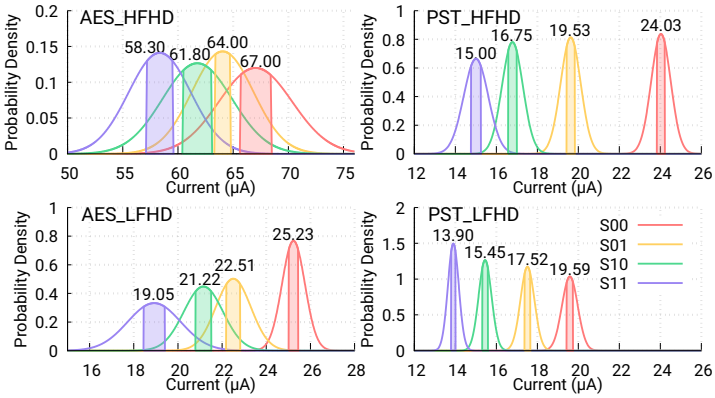

Figure 35: Power consumption "steps" distribution for each crypto core. The shadowed area represents the 95% confidence interval (from [20]).

quality for a given sample.

Nonetheless, one can determine the effectiveness of the proposed SCT insertion framework by verifying three characteristics: (1) the success rate of the attack, (2) the probability of detection (i.e., its stealthiness), and (3) the feasibility of the insertion of the malicious logic during the fabrication-time attack. As the testchip results showed, the SCT was successful in (1) because the cryptokey was leaked as intended, i.e., the attack was fully accomplished. However, since the SCT is an additive HT, it has a probability of being detected by the chip owner. Detecting a trojan of any kind is generally a problematic task [75]. Because SCTs do not alter the device's functionality under attack, any method that relies on observing corrupted bits or any degree of incorrect computation is likely to fail to detect the trojan. Therefore, only techniques that rely on observing the chip's internal structures and/or its power traces have a chance of detecting SCTs. For a complete discussion of all detection methods, I direct the reader to [20].

To verify (3), the attack threat model must first be revisited. In the SCT threat model, the attacker has a limited time window for modifying the victim layout. Thus, manually placing and routing an SCT is unreasonable in such a limited time. Then, the SCT insertion must be automated by utilizing an EDA tool. Inserting an SCT by re-implementing the design has a significant runtime. For example, the testchip illustrated in Figure 29 is a tiny chip compared with today's typical commercial circuits. Still, it requires at least 7 hours and 18 minutes to be implemented (see Figure 36). However, replicating the entire chip is problematic; doing so without the original timing and power constraints is very difficult, with a very high chance of affecting the target performance and thus decreasing the stealthiness of the attack.

Nevertheless, the proposed ECO flow demonstrates that the insertion of malicious logic during a fabrication-time attack can be automated and fast. For example, leveraging the ECO flow, the insertion of the proposed SCT requires only 1 hour and 11 minutes – more than 6 hours faster than re-implementing the whole testchip. On top of that, as previously alluded, the ECO flow can keep the original design untouched, increasing the attack's stealthiness. In addition, the proposed ECO flow does not require the

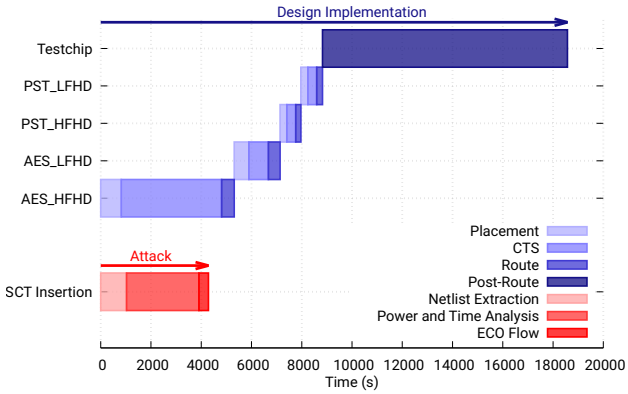

Figure 36: Physical implementation execution time (s) for each step of the flow, and execution time (s) for inserting the SCT in each implemented crypto core (from [20]).

original power and timing constraints; an estimation can be used without significant loss. Moreover, the short runtime associated with the ECO flow makes the fabrication-time attack feasible in a realistic scenario, where the time window that a rogue engineer has for modifying the layout is (very) limited. Therefore, our proposed ECO flow method for inserting SCTs and any malicious logic is compelling and a rogue element could exploit the proposed framework to perform a fabrication-time attack. Furthermore, the proposed ECO framework can be utilized as a platform to assess the layout's vulnerabilities against additive HT insertion.

5.4 Blind Insertion of HTs Framework

The proposed HT insertion framework by ECO (see Figure 28) has a limitation: the attacker has to spot the security-critical nodes by visual inspection. Even though visual inspection is sufficient to locate security-critical nodes for specific targets, such as the AES crypto core, this weakness limits the framework's applicability. To further demonstrate ECO's framework capabilities, an upgraded version is proposed for inserting HTs blindly [19].

An attacker can recover the purpose of signals inside a design by utilizing high-level functionality reconstruction tools. For example, such tools can recover a finite-state machine of a target design, distinguishing control and data path nodes [88]. Therefore, automating the search of security-critical nodes can be done utilizing the output of these tools. Hence, the proposed framework illustrated in Figure 37 leverages high-level functionality reconstruction tools for blindly inserting HTs in finalized layouts – this framework is termed Blind insertion of Hardware Trojans (BioHT).

The BioHT framework assumes an equal threat model as the SCT insertion. It only differs when inspecting the recovered gate-level netlist. Thus, the BioHT is an additional feature to the framework illustrated in Figure 28, performed after the gate-level netlist extraction, comprising five steps: (1) netlist recovery; (2) design analysis; (3) trojan netlist generation; (4) signal selection for connecting the HT; (5) trojan insertion.

BioHT step (1) is performed similarly to the previous ECO framework resulting in a gate-level netlist we refer to as *unamed* since the original hierarchy and name of cells

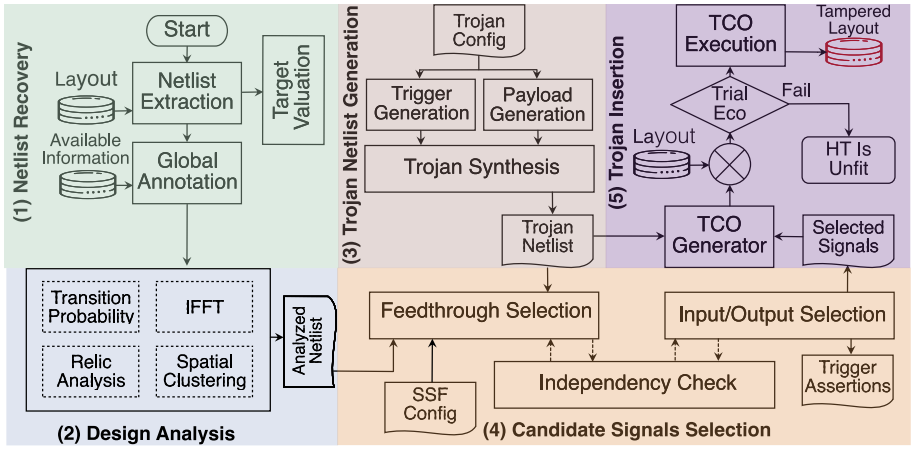

Figure 37: Steps 1)–5) of the BioHT Framework explained in detail. The flow starts at the top left, while the tampered layout (highlighted in red) is the result (adapted from [19]).

and nets are assumed to be absent in the layout. Then, with the gate-level netlist, during step (2), BioHT generates several metrics to aid the search of nodes to use as triggers and payload for the HTs. Those metrics are calculated by applying reverse engineering techniques. However, since these calculations have a considerable runtime proportional to the desired level of understanding of the design, it becomes a tradeoff between runtime and design understanding. Hence, the adversary must carefully choose the metrics to keep the total runtime of the attack short. BioHT generates four different metrics: *transition probability*; *spatial clustering*; *information flow tracking of selected signals*; *RELIC scoring*. *Transition probability* is a metric to find signals with a low probability of transitioning [169, 170] that are suitable for being triggers, increasing the HT stealthiness.

Spatial clustering maps candidate cells to hook the HT while minimizing the wire length of the signals as much as possible to increase the routability of the HT. *Information flow tracking of selected signals* is useful for HTs that intend to leak information. Using imprecise information flow tracking (IIFT) [171], the availability of secret information that each logic gate from the selected signals carries can be measured in an overestimated manner. Thus, this metric has to be complemented with other metrics that explain the functionality of signals. *RELIC score and FSM identification* is valuable high-level information to identify whether a register belongs to a control logic or data path used to design the HT payload to target specific parts of the design functionality. For example, a payload for modifying the control FSM or leaking valuable processed data. The RELIC scoring is performed utilizing the NETA toolset [172].

BioHT step (3) uses a configuration file to generate the HT netlists, where the user can choose any trigger/payload combinations, and parameter values illustrated in Figure 38. The available HTs cover known architectures [65, 173, 174], as well as a few novel payloads (i.e., leakage through FSK/DBPSK, fault sweeping). It is worth mentioning that step (3) is not a limitation of BioHT; a user can skip step (3) and use their own HT netlist or even include new architectures to the BioHT HT generator.

After gathering all metrics during step (2) and generating the desired HT netlists in

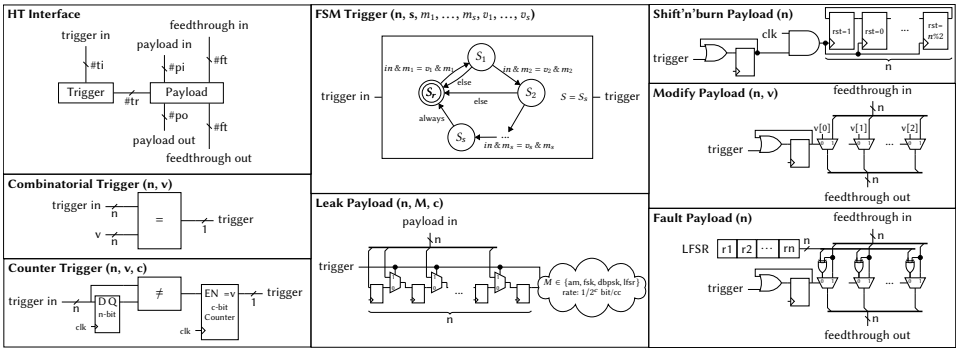

Figure 38: HT Interface and available trojan triggers and payloads. Trigger and payload parameters are given in parentheses (from [19]).

step (3), an adversary can proceed to step (4) to search for appropriate security-critical signals where to connect the HT. The search process starts by associating a signal selection function (SSF) for each interface port of the HT and iteratively selecting candidate signals from the target circuit to connect to each HT port – all based on one or multiple metrics calculated during step (2). In addition, step (4) also performs an independency check on all candidate signals. Avoiding mutually dependent signals is highly beneficial. For example, using a signal as a trigger to activate a dependent payload signal could generate a combinational loop. Moreover, the Modify or Fault payloads should connect to independent signals to maximize the effectiveness of the HT.

Finally, using the HT netlist and the selected signals for each HT, BioHT generates the files for inserting the HT. In addition, BioHT step (5) introduces the Trojan Change Order (TCO) format to make the attack faster. The TCO file follows the same syntax as the ECO file, adding commented lines containing directives for the BioHT tool. Those directives are used to configure the type of HT (e.g., leak, deplete, modify or fault), the number of connections, and the location of the HT gate-level netlist. Instead of providing a modified netlist to perform the ECO, EDA tools also support ECO files. These files describe the modifications to be done by the ECO, with the advantage of performing it interactively. Hence, it is necessary to load the design once for analyzing multiple ECO files. Thus, it is possible to pre-generate TCO files for several types of HT and specialize them according to the target's evaluation. This feature enables the creation of a database of HTs rapidly available for an attack. Finally, the attacker can commit the changes if the TCO trial is successful.

Three crypto cores are utilized for targets, AES, SHA-256, RSA, and the general-purpose PULPino microcontroller to evaluate the BioHT framework. In total, it is explored 96 combinations of triggers, payloads, and targets. The DSE exercise results showed that BioHT could automatically find suitable secure critical nodes for inserting sophisticated HTs into a victim layout. Moreover, the experiments demonstrated that the HT insertion vulnerability of a layout is not correlated to the design's density, i.e., empty space to insert the additional malicious logic. In the two low-density designs, SHA-256 and PULPino, the HT insertion partially failed. In contrast, in the high-density designs, HT insertion succeeded, even for large HTs with hundreds of cells, independent

of the increase in wire length. Thus, the BioHT goes beyond a proof of concept that blindly attacking a layout is possible. The framework can quickly produce a boundary of HT insertion feasibility, provide a risk assessment and guide physical defense strategies for HT insertion. To access all results and a more in-depth discussion of the BioHT framework, I direct the reader to [19]. All the 96 explored combinations results are available in [175].

6 Conclusions and Future Work

Integrated circuits have become a significant part of our daily life, and their integration is constant at a fast pace. Moreover, critical infrastructures are increasingly deploying IC-based systems. Thus, a compromised chip belonging to one of these systems can lead to the leakage of sensitive data and even more dire consequences. For these and many other reasons, the hardware security field has recently increased in popularity. The main goal of this community is to guarantee the trustworthiness of integrated circuits throughout their life span. Many threats and defenses have been recently studied; however, the overall IC's security level is still being determined, while finding many other new threats is still possible. All the presented results in this PhD thesis aim to accelerate the hardware security field research to establish the IC's security, i.e., precisely defining the IC's vulnerabilities and potential countermeasures. Thus, the main contributions of this PhD thesis are a new ASIC-like GPU accelerator with security features, a survey on a defense technique called Split Manufacturing, and an extensive study of hardware trojans in a fabrication-time attack paradigm.

Capable modern SoCs are also essential to the fast development of IC-based systems. In **Chapter 3**, I proposed an open-source GPU architecture to aid the research of domain-specific ASIC accelerators based on GPU-like accelerators – termed G-GPU. A fully automated framework called GPUPlanner is also made publicly available for generating G-GPU IPs from the RTL to a tape-out-ready layout. The G-GPU experimental results demonstrated the feasibility of its architecture as domain-specific ASIC accelerators. Furthermore, the performance comparison between the G-GPU and the RISC-V shows that the G-GPU proposed architecture has excellent benefits for applications with high parallelism. In addition, the GPUPlanner can power gate G-GPU's compute units. This feature enables the creation of low-power, design for reliability, and security solutions. Finally, because the GPUPlanner is an open-source framework, the community can explore the design space of GPU-like accelerators – as the literature lacks GPU architectures targetting ASIC. Moreover, the optional dynamic power control can enable and expand the research of GPU-specific countermeasures against power and EM side-channel attacks. Therefore, the proposed G-GPU architecture and the GPUPlanner framework go beyond analyzing a reasonable GPU-like accelerator in 65nm. The proposed framework can be extended for future work to support other baseline GPU architectures, new solutions to enhance the design's security/reliability, and other technologies.

The current semiconductor supply chain is decentralized, complex, and highly globalized. Design companies must rely on pure-play foundries to manufacture their designs, which is arguably a security threat for ICs. Exposing their layouts to third-party entities can reveal trademark IP secrets, and in the worst scenario, a rogue element inside such foundries could manipulate the layout for malicious reasons. In **Chapter 4**, I surveyed the Split Manufacturing technique, a countermeasure to secure ICs during manufacturing. The surveyed works showed a significant disparity in how the technique is approached. First, there is no consensus on benchmark suites and metrics to use when evaluating the technique, difficulting the comparison between the studies and, in some cases, making it impossible. Despite this difficulty, it was possible to classify

the studies, demonstrating the many interpretations of the technique, its attacks, and defenses. Nonetheless, the results are presented to illustrate the present state of the technique. Therefore, this work can be beneficial for future researchers to contextualize their techniques for augmenting Split Manufacturing.

Predominantly, Spilt Manufacturing's security level is still under debate. Some studies consider the straightforward Split Manufacturing security level enough to protect the layout against fabrication-time attacks, while others argue it is insufficient to secure the layout. However, as previously alluded, the lack of unified benchmark circuits and set of metrics could have diverged the conclusions for many different scenarios. Hence, creating a unified benchmark suite specifically crafted for Split Manufacturing evaluation and a set of metrics to quantify/qualify its performance could facilitate the discussion about Split Manufacturing's security level. In addition, increasing the number of demonstrations in silicon could also help with evaluation and adoption issues related to Split Manufacturing.

One of the many potential threats to an IC during manufacturing is the insertion of a hardware trojan. The literature has many hardware trojan demonstrations, a few even in silicon; however, not a single one disclosed how their hardware trojan is inserted. In **Chapter 5**, I proposed a complete framework based on the ECO feature for inserting HTs in a finalized layout, together with a novel SCT architecture to demonstrate the framework. The SCT insertion was detailed step by step, showing that a rogue element inside a foundry can replicate it effortlessly. Furthermore, the SCT attack was validated by the developed ASIC prototype. The ASIC bench test results demonstrated the attack's success for all samples available, where the cryptokey was extracted via power signature. The measurements have also demonstrated the robustness of the SCT against skews from the manufacturing process. On top of that, the testchip had all 4 SCTs inserted in less than two hours, making the attack viable in an actual fabrication-time attack as it has a limited time window.

One limitation of the proposed HT insertion framework by ECO is that the attacker has to spot the security-critical nodes by visual inspection. Thus, in **Chapter 5**, an upgraded version of the framework for blindly inserting HTs was discussed to further demonstrate ECO's framework capabilities – termed BioHT. Hence, the BioHT framework leverages reverse engineering techniques to introduce sophisticated trojan into circuits, with little knowledge about the target designs. Furthermore, the BioHT experiments demonstrated that the complete approach is fast, allowing the user to execute it multiple times in the time frame between the tape-out and manufacturing. Thus, enabling the selection of the optimum trojan out of several possibilities. Moreover, BioHT also demonstrates how a realistic trojan insertion would be performed and can guide risk assessment, defense, and future research on the topic. Finally, the BioHT framework provides all information and capabilities to advance countermeasures against HT insertion threats.

List of Figures

List of Tables

References

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

[2] Make Use Of, "8 reasons why semiconductors are important to modern living." ttps://www.makeuseof.com/why-semiconductors-important. Accessed: June 15, 2022.

[3] International Monetary Fund, "Digitization of Money and Finance: Challenges and Opportunities." https://www.imf.org/en/News/Articles/2018/05/08/sp050818-digitization-of-money-and-finance-challenges-and-opportunitie. Accessed: June 15, 2022.

[4] European Central Bank, "A digital euro." https://www.ecb.europa.eu/paym/digital_euro/html/index.en.html. Accessed: June 15, 2022.

[5] C. Mucci, L. Vanzolini, A. Lodi, A. Deledda, R. Guerrieri, F. Campi, and M. Toma, "Implementation of aes/rijndael on a dynamically reconfigurable architecture," in *2007 Design, Automation Test in Europe Conference Exhibition*, pp. 1–6, 2007.

[6] J. Nickolls and W. J. Dally, "The gpu computing era," *IEEE Micro*, vol. 30, no. 2, pp. 56–69, 2010.

[7] J. D. Owens, M. Houston, D. Luebke, S. Green, J. E. Stone, and J. C. Phillips, "Gpu computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879–899, 2008.

[8] M. Garland, S. Le Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Y. Zhang, and V. Volkov, "Parallel computing experiences with cuda," *IEEE Micro*, vol. 28, no. 4, pp. 13–27, 2008.

[9] T. D. Perez, M. M. Gonçalves, L. Gobatto, M. Brandalero, J. R. Azambuja, and S. Pagliarini, "G-gpu: A fully-automated generator of gpu-like asic accelerators," in *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 544–547, 2022.

[10] The Wall Street Journal, "There Aren't Enough Chips - Why Are They So Hard to Make?." https://www.wsj.com/story/there-arent-enough-chips-why-are-they-so-hard-to-make-3e29c7e0. Accessed: June 15, 2022.

[11] Intel, "Intel Announces Initial Investment of Over €33 Billion for RD and Manufacturing in EU." https://www.intel.com/content/www/us/en/newsroom/news/eu-news-2022-release.html. Accessed: Aug 21, 2022.

[12] Bloomberg, "The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies." https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies. Accessed: June 15, 2022.

[13] Cybermagazine, "The history of cybersecurity." https://cybermagazine.com/cybersecurity/history-cybersecurity. Accessed: June 15, 2022.

[14] Help Net Security, "Threats to hardware security are growing." https://www.helpnetsecurity.com/2022/05/10/hardware-security-threats-video. Accessed: June 15, 2022.

[15] Semiengineering, "Hardware Security: A Critical Piece Of The Cybersecurity Puzzle." https://semiengineering.com/hardware-security-a-critical-piece-of-the-cybersecurity-puzzle/. Accessed: June 15, 2022.

[16] 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.

[17] T. Perez, M. Imran, P. Vaz, and S. Pagliarini, "Side-channel trojan insertion - a practical foundry-side attack via eco," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2021.

[18] T. Perez and S. Pagliarini, "A side-channel hardware trojan in 65nm cmos with $2\mu\mathrm{W}$ precision and multi-bit leakage capability," in *2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 9–10, 2022.

[19] A. Hepp, T. Perez, S. Pagliarini, and G. Sigl, "A pragmatic methodology for blind hardware trojan insertion in finalized layouts," in *2022 International Conference on Computer-Aided Design (ICCAD)*, 2022.

[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 (TCAD)*, 2022.

[21] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?," in *2013 DATE*, pp. 1259–1264, 2013.

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

[23] M. Yasin, J. J. Rajendran, O. Sinanoglu, and R. Karri, "On improving the security of logic locking," *IEEE TCAD*, vol. 35, no. 9, pp. 1411–1424, 2016.

[24] K. Zamiri Azar, H. Mardani Kamali, H. Homayoun, and A. Sasan, "Threats on logic locking: A decade later," in *GLSVLSI '19*, p. 471–476, 2019.

[25] J. Sweeney, V. Mohammed Zackriya, S. Pagliarini, and L. Pileggi, "Latch-based logic locking," in *2020 IEEE HOST*, pp. 132–141, 2020.

[26] M. Yasin, A. Sengupta, M. T. Nabeel, M. Ashraf, J. J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, (New York, NY, USA), p. 1601–1618, Association for Computing Machinery, 2017.

[27] T. Hoque, R. S. Chakraborty, and S. Bhunia, "Hardware obfuscation and logic locking: A tutorial introduction," *IEEE Design & Test*, vol. 37, no. 3, pp. 59–77, 2020.

[28] Z. U. Abideen, T. D. Perez, and S. Pagliarini, "From fpgas to obfuscated easics: Design and security trade-offs," in *2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pp. 1–4, 2021.

[29] A. Chakraborty, N. G. Jayasankaran, Y. Liu, J. Rajendran, O. Sinanoglu, A. Srivastava, Y. Xie, M. Yasin, and M. Zuzak, "Keynote: A disquisition on logic locking," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 10, pp. 1952–1972, 2020.

[30] 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.

[31] R. P. Cocchi, J. P. Baukus, L. W. Chow, and B. J. Wang, "Circuit camouflage integration for hardware ip protection," in *DAC*, pp. 1–5, 2014.

[32] 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.

[33] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *2015 IEEE HOST*, pp. 137–143, 2015.

[34] CHES, "Conference on Cryptographic Hardware and Embedded Systems." https://ches.iacr.org/. Accessed: Aug 21, 2022.

[35] HOST, "IEEE International Symposium on Hardware Oriented Security and Trust (HOST)." http://www.hostsymposium.org/. Accessed: Aug 21, 2022.

[36] AsianHOST, "Asian Hardware Oriented Security and Trust Symposium (AsianHOST)." http://asianhost.org/. Accessed: Dec 11, 2022.

[37] COSADE, "International Workshop on Constructive Side-Channel Analysis and Secure Design." https://www.cosade.org/. Accessed: Aug 21, 2022.

[38] R. Fair, "History of some early developments in ion-implantation technology leading to silicon transistor manufacturing," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 111–137, 1998.

[39] The Pragmatic Programmers, "The Caculator Wars." https://medium.com/pragmatic-programmers/the-calculator-wars-66bdf4cbab3d. Accessed: June 19, 2022.

[40] Fabricated Knowledge, "Lessons from History: The 1980s Semiconductor Cycle(s)." https://www.fabricatedknowledge.com/p/history-lesson-the-1980s-semiconductor. Accessed: June 19, 2022.

[41] Fabricated Knowledge, "Lessons from History: The 1990s Semiconductor Cycle(s)." https://www.fabricatedknowledge.com/p/lessons-from-history-the-1990s-semiconductor. Accessed: June 19, 2022.

[42] Anysilicon, "What is a Fabless Company." https://anysilicon.com/what-is-a-fabless-company/. Accessed: June 19, 2022.

[43] Semiconductor Engineering, "Design Rule Complexity Rising." https://semiengineering.com/design-rule-complexity-rising/. Accessed: June 19, 2022.

[44] Semiconductor Digest, "Shortage to Surplus Cycle Hits Semi But One Segment Escapes." https://www.semiconductor-digest.com/shortage-to-surplus-cycle-hits-semi-but-one-segment-escapes/. Accessed: June 19, 2022.

[45] Fabricated Knowledge, "The Rising Tide of Semiconductor Cost." https://www.fabricatedknowledge.com/p/the-rising-tide-of-semiconductor. Accessed: June 19, 2022.

[46] EETimes, "Intel Will Rely on TSMC for its Rebound." https://www.eetimes.com/intel-will-rely-on-tsmc-for-its-rebound/. Accessed: Sept. 9, 2022.

[47] i-Micronews, "High-End Performance Packaging 2022 – Focus on 2.5D/3D Integration." https://www.i-micronews.com/products/high-end-performance-packaging-2022-focus-on-2-5d-3d-integration/. Accessed: Aug 17, 2022.

[48] World Health Organization, "Coronavirus disease (COVID-19) pandemic." ttps://www.who.int/europe/emergencies/situations/covid-19. Accessed: Sept. 14, 2022.

[49] Bloomberg, "The Chip Shortage Isn't Over Quite Yet." ttps://www.bloomberg.com/news/newsletters/2022-08-19/the-chip-shortage-isn-t-over-quite-yet. Accessed: Sept. 14, 2022.

[50] IEEE Spectrum, "How and When the Chip Shortage Will End, in 4 Charts." https://spectrum.ieee.org/chip-shortage. Accessed: Sept. 14, 2022.

[51] European Chips, "Survey report." https://digital-strategy.ec.europa.eu/en/library/european-chips-survey. Accessed: Sept. 9, 2022.

[52] Manufacturing Tomorrow, "6 Implications of the Chip Shortage for Auto Manufacturing." https://www.manufacturingtomorrow.com/story/2022/05/6-implications-of-the-chip-shortage-for-auto-manufacturing/18744/. Accessed: Sept. 19, 2022.

[53] Synopsys, "What is Library Characterization?." https://www.synopsys.com/glossary/what-is-library-characterization.html. Accessed: June 21, 2022.

[54] Cadence, "Innovus Implementation System." https://www.cadence.com/content/dam/cadence-www/global/en_US/documents/tools/digital-design-signoff/innovus-implementation-system-ds.pdf. Accessed: Sept. 15, 2022.

[55] J. Kim and T. Kim, "Useful clock skew scheduling using adjustable delay buffers in multi-power mode designs," in *The 20th Asia and South Pacific Design Automation Conference*, pp. 466–471, 2015.

[56] K. Chae, S. Mukhopadhyay, C.-H. Lee, and J. Laskar, "A dynamic timing control technique utilizing time borrowing and clock stretching," in *IEEE Custom Integrated Circuits Conference 2010*, pp. 1–4, 2010.

[57] R. Signh, *Signal integrity effects in custom IC and ASIC designs*. IEEE Press, 2002.

[58] European Union Intellectual Property Office (EUIPO), "2019 Status Report On IPR Infringement," [Online]. Available: https://euipo.europa.eu/ohimportal/en/web/observatory/status-reports-on-ip-infringement.

[59] 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.

[60] 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.

[61] 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.

[62] R. Kumar, P. Jovanovic, W. Burleson, and I. Polian, "Parametric trojans for fault-injection attacks on cryptographic hardware," in *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 18–28, 2014.

[63] G. T. Becker, F. Regazzoni, C. Paar, and W. P. Burleson, "Stealthy dopant-level hardware trojans," in *Cryptographic Hardware and Embedded Systems - CHES 2013* (G. Bertoni and J.-S. Coron, eds.), (Berlin, Heidelberg), pp. 197–214, Springer Berlin Heidelberg, 2013.

[64] L. Lin, W. Burleson, and C. Paar, "Moles: Malicious off-chip leakage enabled by side-channels," in *2009 IEEE/ACM International Conference on Computer-Aided Design*, pp. 117–122, 2009.

[65] L. Lin *et al.*, "Trojan side-channels: Lightweight hardware trojans through side-channel engineering," in *Cryptographic Hardware and Embedded Systems - CHES 2009*, pp. 382–395, 2009.

[66] Y. Jin and Y. Makris, "Hardware trojans in wireless cryptographic ics," *IEEE Design Test of Computers*, vol. 27, no. 1, pp. 26–35, 2010.

[67] Y. Liu, Y. Jin, and Y. Makris, "Hardware trojans in wireless cryptographic ics: Silicon demonstration & detection method evaluation," in *Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 399–404, 2013.

[68] R. Kumar, P. Jovanovic, W. Burleson, and I. Polian, "Parametric trojans for fault-injection attacks on cryptographic hardware," in *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 18–28, 2014.

[69] J.-F. Gallais *et al.*, "Hardware trojans for inducing or amplifying side-channel leakage of cryptographic software," in *Trusted Systems*, pp. 253–270, 2011.

[70] L. Ali and Farshad, "Analog hardware trojan design and detection in OFDM based wireless cryptographic ICs," *Plos One*, vol. 16, no. 7, p. e0254903, 2021.

[71] S. Ghandali, T. Moos, A. Moradi, and C. Paar, "Side-Channel Hardware Trojan for Provably-Secure SCA-Protected Implementations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 6, pp. 1435–1448, 2020.

[72] F. Almeida, M. Imran, J. Raik, and S. Pagliarini, "Ransomware attack as hardware trojan: A feasibility and demonstration study," *IEEE Access*, vol. 10, pp. 44827–44839, 2022.

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

[74] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware trojans," *Computer*, vol. 43, pp. 39–46, Oct 2010.

[75] S. Bhasin and F. Regazzoni, "A survey on hardware trojan detection techniques," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2021–2024, 2015.

[76] M. Li, B. Yu, Y. Lin, X. Xu, W. Li, and D. Z. Pan, "A practical split manufacturing framework for trojan prevention via simultaneous wire lifting and cell insertion," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 265–270, 2018.

[77] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A novel technique for improving hardware trojan detection and reducing trojan activation time," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 1, pp. 112–125, 2012.

[78] 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.

[79] Y. Liu, Y. Jin, A. Nosratinia, and Y. Makris, "Silicon demonstration of hardware trojan design and detection in wireless cryptographic ics," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 4, pp. 1506–1519, 2017.

[80] R. M. Rad, X. Wang, M. Tehranipoor, and J. Plusquellic, "Power supply signal calibration techniques for improving detection resolution to hardware trojans," in *2008 IEEE/ACM International Conference on Computer-Aided Design*, pp. 632–639, 2008.

[81] J. Cruz, Y. Huang, P. Mishra, and S. Bhunia, "An automated configurable trojan insertion framework for dynamic trust benchmarks," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1598–1603, 3 2018.

[82] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor, "Hardware trojans: Lessons learned after one decade of research," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 22, pp. 6:1–6:23, May 2016.

[83] K. Hasegawa, K. Yamashita, S. Hidano, K. Fukushima, K. Hashimoto, and N. Togawa, "Node-wise hardware trojan detection based on graph learning,"

[84] 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 *2022 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 796–801, 2022.

[85] A. Hepp, J. Baehr, and G. Sigl, "Golden model-free hardware trojan detection by classification of netlist module graphs," in *2022 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1317–1322.

[86] R. Torrance and D. James, "The state-of-the-art in semiconductor reverse engineering," *Design Automation Conference*, pp. 333–338, 2011.

[87] L. Aksoy, A. Hepp, J. Baehr, and S. Pagliarini, "Hardware obfuscation of digital fir filters," in *2022 25th International Symposium on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, pp. 68–73, 2022.

[88] T. Meade, S. Zhang, and Y. Jin, "Netlist reverse engineering for high-level functionality reconstruction," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 655–660, 2016.

[89] P. Subramanyan, N. Tsiskaridze, K. Pasricha, D. Reisman, A. Susnea, and S. Malik, "Reverse Engineering Digital Circuits Using Functional Analysis," pp. 1277–1280, March 2013.

[90] P. Rohatgi, *Improved Techniques for Side-Channel Analysis*, pp. 381–406. Boston, MA: Springer US, 2009.

[91] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99* (M. Wiener, ed.), pp. 388–397, 1999.

[92] Intel, "Intel Announces Initial Investment of Over €33 Billion for R&D and Manufacturing in EU." https://www.intel.com/content/www/us/en/newsroom/news/eu-news-2022-release.html. Accessed: June 24, 2022.

[93] TechCrunch, "TSMC to build a $12 billion advanced semiconductor plant in Arizona with US government support." https://techcrunch.com/2020/05/14/tsmc-to-build-a-12-billion-advanced-semiconductor-plant-in-arizona-with-u-s-government-support/. Accessed: June 24, 2022.

[94] IEEE, "Ieee standard for design and verification of low-power, energy-aware electronic systems," *IEEE Std 1801-2018*, pp. 1–548, 2019.

[95] V. Natarajan, A. K. Nagarajan, N. Pandian, and V. G. Savithri, "Low power design methodology," in *Very-Large-Scale Integration* (K. H. Yeap and H. Nisar, eds.), ch. 3, Rijeka: IntechOpen, 2018.

[96] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[97] T. Fritzmann, G. Sigl, and J. Sepúlveda, "Risq-v: Tightly coupled risc-v accelerators for post-quantum cryptography," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, p. 239–280, Aug. 2020.

[98] AMD, "APU 101: All about AMD Fusion Accelerated Processing Units." http://developer.amd.com/wordpress/media/2012/10/apu101.pdf. Accessed: Aug 15, 2022.

[99] ARM, "INSTRUCTION SET ARCHITECTURE (ISA)." https://www.arm.com/glossary/isa. Accessed: Aug 15, 2022.

[100] RISC-V Foundation, "RISC-V Cores and SoC Overview." https://riscv.org. Accessed: Aug 15, 2022.

[101] OpenPower Foundation, "OpenPower." https://openpowerfoundation.org. Accessed: Aug 15, 2022.

[102] Oracle and Sun Microsystems, "OpenSPARC Overview, 2019." https://www.oracle.com/servers/technologies/opensparc-overview.html. Accessed: Aug 15, 2022.

[103] Linux Foundation, "CHIPS: Common Hardware for Interfaces, Processors and Systems." https://chipsalliance.org. Accessed: Aug 15, 2022.

[104] Makeuseof, "What Is a TPU (Tensor Processing Unit) and What Is It Used For?." https://www.makeuseof.com/what-is-tpu-how-is-it-used/. Accessed: June 15, 2022.

[105] Circuit Cellar, "The Future of Embedded FPGAs — eFPGA: The Proof is in the Tape Out." https://circuitcellar.com/insights/tech-the-future/the-future-of-embedded-fpgas-efpga-the-proof-is-in-the-tape-out/. Accessed: Aug 17, 2022.

[106] S. K. Lee, P. N. Whatmough, M. Donato, G. G. Ko, D. Brooks, and G.-Y. Wei, "Smiv: A 16-nm 25-mm^2 soc for iot with arm cortex-a53, efpga, and coherent accelerators," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 2, pp. 639–650, 2022.

[107] Xilinx, "SoCs with Hardware and Software Programmability." https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html. Accessed: Aug 17, 2022.

[108] ONLOGIC, "Your Ultimate Guide to Understanding PCIe Gen 4.0." https://www.onlogic.com/company/io-hub/your-ultimate-guide-to-understanding-pcie-gen-4/. Accessed: Aug 17, 2022.

[109] P. P. Brahma, D. Wu, and Y. She, "Why deep learning works: A manifold disentanglement perspective," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, no. 10, pp. 1997–2008, 2016.

[110] MarketWatch, "Artificial Intelligence (AI) Chips Market Size, Share, Growth and Forecast to 2027 with 36.6% CAGR." https://www.marketwatch.com/press-release/artificial-intelligence-ai-chips-market-size-share-growth-and-forecast-to-2027-with-366-cagr-141-pages-report-2022-09-20. Accessed: Sept. 23, 2022.

[111] NVIDIA, "NVIDIA Tensor Cores: Unprecedented Acceleration for HPC and AI." ttps://www.nvidia.com/en-us/data-center/tensor-cores/. Accessed: Sept. 23, 2022.

[112] TechRepublic, "Massive Intel CPU flaw: Understanding the technical details of Meltdown and Spectre." https://www.techrepublic.com/article/massive-intel-cpu-flaw-understanding-the-technical-details-of-meltdown-and-spectre. Accessed: Feb 19, 2021.

[113] Black Hat, "Exploiting the DRAM rowhammer bug to gain kernel privileges." https://www.blackhat.com/docs/us-15/materials/us-15-Seaborn-Exploiting-The-DRAM-Rowhammer-Bug-To-Gain-Kernel-Privileges.pdf. Accessed: Dec 9, 2022.

[114] Y. Gao and Y. Zhou, "Side-channel attacks with multi-thread mixed leakage," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 770–785, 2021.

[115] NVIDIA, "Record 136 NVIDIA GPU-Accelerated Supercomputers Feature in TOP500 Ranking." https://blogs.nvidia.com/blog/2019/11/19/record-gpu-accelerated-supercomputers-top500/, 2019. Accessed: 2022-11-02.

[116] J. E. R. Condia, B. Du, M. Sonza Reorda, and L. Sterpone, "Flexgripplus: An improved GPGPU model to support reliability analysis," *Microelectronics Reliability*, vol. 109, p. 113660, 2020.

[117] M. Al Kadi, B. Janssen, and M. Huebner, "Fgpu: An simt-architecture for fpgas," in *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '16, (New York, NY, USA), p. 254–263, Association for Computing Machinery, 2016.

[118] ARM, "Learn the architecture - An introduction to AMBA AXI." https://developer.arm.com/documentation/102202/0300/AXI-protocol-overview. Accessed: Sept. 27, 2022.

[119] R. Ma, J.-C. Hsu, T. Tan, E. Nurvitadhi, D. Sheffield, R. Pelt, M. Langhammer, J. Sim, A. Dasu, and D. Chiou, "Specializing fgpu for persistent deep learning," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 14, July 2021.

[120] V. Gangadhar, R. Balasubramanian, M. Drumond, Z. Guo, J. Menon, C. Joseph, R. Prakash, S. Prasad, P. Vallathol, and K. Sankaralingam, "Miaow: An open source gpgpu," in *2015 IEEE Hot Chips 27 Symposium (HCS)*, pp. 1–43, 2015.

[121] P. Duarte, P. Tomas, and G. Falcao, "Scratch: An end-to-end application-aware soft-gpgpu architecture and trimming tool," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, (New York, NY, USA), p. 165–177, Association for Computing Machinery, 2017.

[122] H. E. Sumbul, K. Vaidyanathan, Q. Zhu, F. Franchetti, and L. Pileggi, "A synthesis methodology for application-specific logic-in-memory designs," in *ACM/EDAC/IEEE Design Automation Conference*, pp. 1–6, 2015.

[123] J. Ahn, C. Jin, J. Kim, M. Rhu, Y. Fei, D. Kaeli, and J. Kim, "Trident: A hybrid correlation-collision gpu cache timing attack for aes key recovery," in *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pp. 332–344, 2021.

[124] Y. Gao, W. Cheng, H. Zhang, and Y. Zhou, "Cache-collision attacks on gpu-based aes implementation with electro-magnetic leakages," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, pp. 300–306, 2018.

[125] OpenHW Group, "OpenHW Group CORE-V CV32E40P RISC-V IP." https://github.com/openhwgroup/cv32e40p. Accessed: Aug 15, 2022.

[126] Centre For Hardware Security, Tallinn University of Technology, "GPUPlanner." ttps://github.com/Centre-for-Hardware-Security/gpu-asic, 2022. Accessed: 2022-11-03.

[127] M. Pecht and S. Tiku, "Bogus: Electronic Manufacturing and Consumers Confront a Rising Tide of Counterfeit Electronics," *IEEE Spectrum, vol. 43, no. 5, pp. 37–46*, 2006.

[128] Defense Advanced Research Projects Agency, "Lessons from History: The 1980s Semiconductor Cycle(s)." https://www.darpa.mil/. Accessed: Oct. 5, 2022.

[129] Intelligence Advanced Research Projects Activity (IARPA), "Trusted Integrated Circuits Program," [Online]. Available: https://www.iarpa.gov/index.php/research-programs/tic.

[130] T. Kikkawa and R. Joshi, "Design Technology Co-Optimization for 10 nm and Beyond," in *Proceedings of the IEEE 2014 Custom Integrated Circuits Conference*, pp. 1–1, Sep. 2014.

[131] K. Vaidyanathan, B. P. Das, E. Sumbul, R. Liu, and L. Pileggi, "Building Trusted ICs Using Split Fabrication," *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 1–6, 2014.

[132] K. Vaidyanathan, R. Liu, E. Sumbul, Q. Zhu, F. Franchetti, and L. Pileggi, "Efficient and Secure Intellectual Property (IP) Design with Split Fabrication," in *2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 13–18, 2014.

[133] B. Hill, R. Karmazin, C. T. O. Otero, J. Tse, and R. Manohar, "A Split-Foundry Asynchronous FPGA," in *Proceedings of the IEEE 2013 Custom Integrated Circuits Conference*, pp. 1–4, Sep. 2013.

[134] T. Usui, K. Tsumura, H. Nasu, Y. Hayashi, G. Minamihaba, H. Toyoda, H. Sawada, S. Ito, H. Miyajima, K. Watanabe, M. Shimada, A. Kojima, Y. Uozumi, and H. Shibata, "High Performance Ultra Low-k (k=2.0/keff=2.4)/Cu Dual-Damascene Interconnect Technology with Self-Formed MnSixOy Barrier Layer for 32 nm-node," in *2006 International Interconnect Technology Conference*, pp. 216–218, 2006.

[135] F. Imeson, A. Emtenan, S. Garg, and M. Tripunitara, "Securing computer hardware using 3d integrated circuit (IC) technology and split manufacturing for obfuscation," in *22nd USENIX Security Symposium (USENIX Security 13)*, pp. 495–510, USENIX Association, Aug. 2013.

[136] 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.

[137] S. N. Pagliarini, M. M. Isgenc, M. G. A. Martins, and L. Pileggi, "Application and Product-Volume-Specific Customization of BEOL Metal Pitch," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 9, pp. 1627–1636, 2018.

[138] J. Rajendran, O. Sinanoglu, and R. Karri, "Is Split Manufacturing Secure?," in *Design, Automation and Test in Europe (DATE)*, no. Ic, pp. 1259–1264, 2013.

[139] J. Magaña, D. Shi, and A. Davoodi, "Are Proximity Attacks a Threat to the Security of Split Manufacturing of Integrated Circuits?," *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, vol. 07-10-Nove, no. c, pp. 1–7, 2016.

[140] Y. Wang, P. Chen, J. Hu, G. Li, and J. Rajendran, "The Cat and Mouse in Split Manufacturing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 5, pp. 805–817, 2018.

[141] W. Zeng, B. Zhang, and A. Davoodi, "Analysis of Security of Split Manufacturing Using Machine Learning," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 12, pp. 2767–2780, 2019.

[142] H. Li, S. Patnaik, A. Sengupta, H. Yang, J. Knechtel, B. Yu, E. F. Y. Young, and O. Sinanoglu, "Attacking split manufacturing from a deep learning perspective," in *2019 56th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2019.

[143] S. Chen and R. Vemuri, "On the Effectiveness of the Satisfiability Attack on Split Manufactured Circuits," in *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, pp. 83–88, 2018.

[144] S. Chen and R. Vemuri, "Exploiting Proximity Information in a Satisfiability Based Attack Against Split Manufactured Circuits," *Proceedings of the 2019 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2019*, pp. 171–180, 2019.

[145] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, "Network Flows: Theory, Algorithms, and Applications.," *Upper Saddle River, NJ, USA: Prentice-Hall*, 1993.

[146] H. Zhou, R. Jiang, and S. Kong, "CycSAT: SAT-Based Attack on Cyclic Logic Encryptions," in *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 49–56, 2017.

[147] M. Jagasivamani, P. Gadfort, M. Sika, M. Bajura, and M. Fritze, "Split-Fabrication Obfuscation: Metrics and techniques," *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 7–12, 2014.

[148] O. Otero, J. Tse, R. Karmazin, B. Hill, and R. Manohar, "Automatic Obfuscated Cell Layout for Trusted Split-Foundry Design," *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 56–61, 2015.

[149] K. Xiao, D. Forte, and M. M. Tehranipoor, "Efficient and secure split manufacturing via obfuscated built-in self-authentication," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 14–19, 2015.

[150] P. Yang and M. Marek-Sadowska, "Making split-fabrication more secure," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2016.

[151] Y. Wang, P. Chen, J. Hu, and J. Rajendran, "Routing Perturbation for Enhanced Security in Split Manufacturing," *Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 605–610, 2017.

[152] L. Feng, Y. Wang, J. Hu, W. K. Mak, and J. Rajendran, "Making Split Fabrication Synergistically Secure and Manufacturable," *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, vol. 2017-Novem, pp. 313–320, 2017.

[153] A. Sengupta, S. Patnaik, J. Knechtel, M. Ashraf, S. Garg, and O. Sinanoglu, "Rethinking Split Manufacturing: An Information-Theoretic Approach with Secure Layout Techniques," *IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, vol. 2017-Novem, pp. 329–336, 2017.

[154] Z. Chen, P. Zhou, T. Y. Ho, and Y. Jin, "How Secure is Split Manufacturing in Preventing Hardware Trojan?," *IEEE Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pp. 1–6, 2017.

[155] S. Patnaik, J. Knechtel, M. Ashraf, and O. Sinanoglu, "Concerted Wire Lifting: Enabling Secure and Cost-Effective Split Manufacturing," *Asia and South Pacific Design Automation Conference (ASP-DAC)*, vol. 2018-Janua, pp. 251–258, 2018.

[156] S. Patnaik, M. Ashraf, J. Knechtel, and O. Sinanoglu, "Raise Your Game for Split Manufacturing: Restoring the True Functionality Through BEOL," *Design Automation Conference (DAC)*, pp. 1–6, 2018.

[157] M. A. Masoud, Y. Alkabani, and M. W. El-Kharashi, "Obfuscation of Digital Systems using Isomorphic Cells and Split Fabrication," *International Conference on Computer Engineering and Systems (ICCES)*, pp. 488–493, 2019.

[158] M. Li, B. Yu, Y. Lin, X. Xu, W. Li, and D. Z. Pan, "A Practical Split Manufacturing Framework for Trojan Prevention via Simultaneous Wire Lifting and Cell Insertion," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 9, pp. 1585–1598, 2019.

[159] Tezzarron Semiconductors, "Lessons from History: The 1980s Semiconductor Cycle(s)." http://www.tezzaron.com/media/3D-ICs_and_Integrated_Circuit_Security.pdf. Accessed: March 19, 2020.

[160] N. Jasika, N. Alispahic, A. Elma, K. Ilvana, L. Elma, and N. Nosovic, "Dijkstra's Shortest Path Algorithm Serial and Parallel Execution Performance Analysis," in *2012 Proceedings of the 35th International Convention MIPRO*, pp. 1811–1815, 2012.

[161] D. Z. Pan, B. Yu, and J. Gao, "Design for Manufacturing With Emerging Nanolithography," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 10, pp. 1453–1472, 2013.

[162] Y. Ding, C. Chu, and Wai-Kei Mak, "Throughput Optimization for SADP and E-beam Based Manufacturing of 1D Layout," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2014.

[163] M. Schobert et al., "Degate." http://www.degate.org/, 2011. Accessed: 2022-11-05.

[164] V. Gohil, M. Tressler, K. Sipple, S. Patnaik, and J. Rajendran, "Games, dollars, splits: A game-theoretic analysis of split manufacturing," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 5077–5092, 2021.

[165] S. M. Ben, "Security challenges and requirements for industrial control systems in the semiconductor manufacturing sector," 2012.

[166] T. Trippel *et al.*, "ICAS: An Extensible Framework for Estimating the Susceptibility of IC Layouts to Additive Trojans," *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1078–1095, 2020.

[167] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007* (P. Paillier and I. Verbauwhede, eds.), (Berlin, Heidelberg), pp. 450–466, Springer Berlin Heidelberg, 2007.

[168] S. Ghandali *et al.*, "Side-channel hardware trojan for provably-secure sca-protected implementations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 6, pp. 1435–1448, 2020.

[169] H. Salmani, M. Tehranipoor, and J. Plusquellic, "A novel technique for improving hardware trojan detection and reducing trojan activation time," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, pp. 112–125, Jan 2012.

[170] S. Yu, W. Liu, and M. O'Neill, "An improved automatic hardware trojan generation platform," in *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 302–307, 7 2019.

[171] W. Hu, A. Ardeshiricham, and R. Kastner, "Hardware information flow tracking," vol. 54, no. 4, pp. 83:1–83:39.

[172] T. Meade, "Netlist Analysis Toolset (NETA)." https://github.com/jinyier/NetA, 2018. Accessed: 2022-11-05.

[173] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, "Benchmarking of hardware trojans and maliciously affected circuits," *Journal of Hardware and Systems Security*, vol. 1, pp. 85–102, Mar 2017.

[174] A. Baumgarten, M. Steffen, M. Clausman, and J. Zambreno, "A case study in hardware trojan design and implementation," vol. 10, no. 1, pp. 1–14.

[175] A. Hepp, T. Perez, S. Pagliarini, and G. Sigl, "BioHT (Blind Insertion of Hardware Trojans) Tool." https://github.com/Centre-for-Hardware-Security/bio_hardware_trojan, 2016. Accessed: 2021-11-25.

Acknowledgements

First and foremost, I must thank my supervisor and head of the Centre for Hardware Security, Prof. Dr. Samuel Pagliarini. He was the one that introduced me to the field of Hardware Security. His deep knowledge inspired and gave me the confidence that this topic was worth researching. Completing this PhD would not have been possible without his assistance and tremendous dedication. Thank you very much for your support and understanding over these past three years of hard work. Furthermore, I would like to thank all Centre Hardware for Security members and the Dept. of Computer Systems for their support and collaboration.

I also want to express sincere gratitude to my closest friends, Chaves, Dr. Henrique, Dr. Fernando, Felipe, my sister Isabela, and my brother Rodrigo. When I faced uncertain moments, I could always count on my friends to steer me in the right direction. When my papers were accepted, sharing the news and celebrating with my friends made the accomplishments much more joyful. I deeply value their support, fruitful conversations, and almost daily friendly chats. Not only they helped me with work-related tasks, but they brought work-life balance during my studies. I am fortunate to have such friends.

Thanks to the Tallinn University of Technology for providing the structure, resources, and financial means to complete this PhD thesis. I am deeply grateful to the projects and organizations that have supported me during my PhD studies; "Novel and Competent Solutions Towards Synthesizing Trusted Hardware," Estonian Research Council; SAFEST, European Commission; SMART4ALL, European Commission; IT Academy, European Social Fund and Estonian Education and Youth Board. Special thanks to the Technical University of Munich for having me for almost a month.

Most importantly, I am grateful for my family's unconditional, unequivocal, and loving support.

Abstract
Security-Aware Physical Synthesis of Integrated Circuits

The conception of a modern hardware device is a collective effort shared between different entities. This characteristic makes the integrated circuit supply chain decentralized, complex, and highly globalized. Moreover, for modern hardware devices, the current organization of the IC supply chain is arguably a security threat. Since critical infrastructures are increasingly deploying integrated circuits-based systems, a compromise chip belonging to one of these systems can lead to the leakage of sensitive data and even more dire consequences. Therefore, ensuring the integrity of the technologies is crucial for protecting digital information and maintaining critical operational systems – this is precisely the focus of the hardware security field. The IC is susceptible to many recently demonstrated threats during its lifespan, such as the insertion of hardware trojans, IP piracy, overbuilding, reverse engineering, and side-channel attacks. However, only a handful of countermeasures for specific threats have been proposed – for example, Split Manufacturing, Logic Locking, and IC Camouflage. Unfortunately, the current state of these techniques makes them unsuitable for large-scale production of ICs, either because of practicality and/or insufficient security guarantees. Therefore, this thesis presents a comprehensive study of different hardware security topics. All the presented results in this PhD thesis aim to accelerate the hardware security field to determine the IC's security level. The main contributions are summarized as follows:

Open-source GPU architecture to aid the research of domain-specific ASIC accelerators based on GPU-like accelerators – termed G-GPU. Capable modern system-on-chips are essential to the fast development of IC-based systems. However, the literature lacks an open-source GPU architecture for application-specific integrated circuits. Hence, I propose an open-source GPU architecture targetting ASIC to close this research gap in this thesis. Among the few GPU architectures available in the literature is the FGPU, a GPU for FPGA instead of ASIC. Utilizing the FGPU as a baseline, I translated it to targeting ASIC and optimized the architecture utilizing smart memory division – the resulting architecture is referred to as G-GPU. The G-GPU performance results compared with the RISC-V show that the G-GPU has excellent benefits for applications with high parallelism. In addition, a full framework is also made publicly available for generating G-GPU IPs from the RTL to a tape-out-ready layout – called GPUPlanner. The GPUPlanner also has the possibility of power gating, enabling the creation of low-power, design for reliability, and even security solutions. On top of that, the results include 6 G-GPU versions of tapeout-ready layouts implemented in a 65nm CMOS technology. Those versions vary in the number of computing units, operating frequency, and power gating implementation.

A Survey on Split Manufacturing attacks and defenses. Due to the current IC supply chain organization, most design companies, must outsource their design manufacturing to pure-play foundries. This practice is arguably a security threat for ICs. Exposing their layouts to third-party entities can reveal trademark IP secrets. In the worst scenario, a rogue element inside such foundries could manipulate the

layout for malicious reasons. Split manufacturing is a promising defense technique to overcome concerns associated with outsourcing IC manufacturing. In Split Manufacturing, the Front End of Line (FEOL) layers (transistors and lower metal layers) are manufactured at an untrusted high-end foundry. The Back End of Line (BEOL) layers (higher metal layers) are manufactured at a trusted low-end foundry. The presented survey in this thesis is a detailed overview of the technique, the many attacks towards Split Manufacturing, and the possible defense techniques described in the literature. Different threat models and assumptions for the attacks are concisely presented. The defense techniques studies are classified into proximity perturbation, wire lifting, and layout obfuscation. The primary outcome of our survey is to highlight the discrepancy between many studies – some claim netlists can be reconstructed with near-perfect precision. In contrast, others claim marginal success in retrieving BEOL connections. Finally, future trends and challenges inherent in Split Manufacturing are discussed, including the difficulty of evaluating the efficiency of the technique.

A methodology for inserting hardware trojans into finalized layouts. One of the many potential threats to an IC during manufacturing is the insertion of a hardware trojan. The literature has many hardware trojan demonstrations, a few even in silicon; however, not a single one disclosed how their hardware trojan is inserted. In this thesis, I proposed a complete framework based on the ECO feature for inserting HTs in a finalized layout and a novel side-channel hardware trojan (SCT) architecture to demonstrate the framework. The SCT is designed to aid power side-channel attacks by inducing controlled power consumption. The SCT insertion was detailed step by step, showing that a rogue element inside a foundry can replicate it effortlessly. Furthermore, the SCT attack was validated by the developed ASIC prototype in a 65nm CMOS technology. The ASIC prototype comprises four crypto cores, two versions of the AES, and two of the Present, each altered with an SCT. The chip testbench results demonstrated the attack's success for all samples available, where the cryptokey was extracted via power signature. The measurements have also demonstrated the robustness of the SCT against skews from the manufacturing process. On top of that, the testchip had all 4 SCTs inserted in less than two hours, making the attack viable in an actual fabrication-time attack as it has a limited time window.

Moreover, an upgraded version of the framework for blindly inserting HTs is presented to further demonstrate ECO's framework capabilities – termed BioHT. Hence, the BioHT framework leverages reverse engineering techniques to introduce sophisticated trojan into circuits, with little knowledge about the target designs. Furthermore, the BioHT experiments demonstrated that the complete approach is fast, allowing the user to execute it multiple times in the time frame between the tape-out and manufacturing. Thus, enabling the selection of the optimum trojan out of several possibilities. Moreover, BioHT also demonstrates how a realistic trojan insertion would be performed and can guide risk assessment, defense, and future research on the topic.

Kokkuvõte
Integraallülituste turvateadlik füüsiline süntees

Kaasaegse riistvaraseadme kontseptsioon on ühine jõupingutus, mida jagatakse erinevate üksuste vahel. See omadus muudab integraallülituse tarneahela detsentraliseerituks, keerukaks ja väga globaliseerituks. Veelgi enam, tänapäevaste riistvaraseadmete jaoks on integraallülituse tarneahela praegune korraldus vaieldamatult julgeolekuoht. Kuna kriitilistes infrastruktuurides kasutatakse üha enam integraallülitustel põhinevaid süsteeme, siis võib üks neisse süsteemidesse kuuluv tahtlikult kahjustatud kiip põhjustada tundlike andmete lekkimist või veelgi kohutavamaid tagajärgi. Seetõttu on tehnoloogiate terviklikkuse tagamine ülioluline digitaalse teabe kaitsmisel ja kriitiliste operatsioonisüsteemide ülalpidamisel – just see on riistvaraturbe valdkonna fookus. Integraallülitus on oma eluea jooksul vastuvõtlik paljudele hiljuti demonstreeritud ohtudele, nagu riistvara troojade lisamine, intellektuaalomandi piraatlus, ületootmine, pöördprojekteerimine ja külgkanalite rünnakud. Siiski on välja pakutud vaid käputäis konkreetsete ohtude vastumeetmeid – näiteks jaotatud tootimne, loogika lukustus ja integraal lülituse maskeerimine. Kahjuks muudab antud meetotite praegune seisukord nende praktilisuse ja/või ebapiisavate turvagarantiide tõttu ebasobivaks integraallülituste suuremahulisel tootmisel. Käesolev lõputöö esitab põhjaliku uuringu erievate riistvaraturbe teemade kohta. Esitatud tulemused on suunatud riistvaraturbe valdkonna kiirendamisele, et määrata kindlaks integraallülituste turbetase. Peamised panused on kokku võetud järgmiselt:

Avatud lähtekoodiga graafika töötlemisüksuse (GPU) arhitektuur, mis aitab uurida domeenile iseloomulikke rakendusspetsiifilisi integraallülituskiirendeid (ASIC), mis põhinevad GPU-laadsetel kiirenditel – mida nimetatakse G-GPU-ks. Võimsad kaasaegsed süsteemkiibid on integraallülituspõhiste süsteemide kiireks arendamiseks olulised.Siiski puudub kirjanduses avatud lähtekoodiga GPU arhitektuur ASIC-u jaoks. Seetõttu pakun välja avatud lähtekoodiga GPU-arhitektuuri, mis on suunatud ASIC-ule, et täita see uurimislünk selle lõputööga. Kirjanduses vähe saadaolevate GPU-arhitektuuride hulgas on väljatoodud FGPU, mis on programmeeritaval ventiilmaatriksil (FPGA) baseeruv GPU disain. Baseerudes FGPU lähetkoodile muutsin antud disaini ASIC spetsiifiliseks ja optimeerisin arhitektuuri mis kasutaks nutikat mälujaotust – saadud arhitektuuri nimetatakse G-GPU-ks. G-GPU jõudlustulemused võrreldes RISC-V-ga näitavad, et G-GPU-l on suure paralleelsusega rakenduste jaoks suurepärased eelised. Lisaks on avalikustatud ka täielik raamistik G-GPU intellektuaalomandi genereerimiseks registersiirde tasemelt kuni tootmisvalmis kiibi pinnalaotuse disainini – nimega GPUPlanner. Antud raamistikku on ka lisatud toitevärava lisamise võimekus, mis võimaldab luua väikesema voolutarbega sedmeid, töökindlamaid disaine ja isegi lisada turvalahendusi. Lisaks on tulemustes väljatoodud 6 G-GPU tootmisvalmis pinnalaotuse disaini, mis on realiseeritud 65 nm CMOS-tehnoloogias. Need versioonid erinevad arvutusüksuste arvu, töösageduse ja toitevärava rakendamise poolest.

Ülevaade jaotatud-tootmise rünnakute ja kaitsemehhanismide kohta. Praeguse integraallülituste tarneahela korralduse tõttu peavad enamik disainiettevõtteid

oma disainitootmise allhankima kitsalt tegutsevatele kiibitootmis tööstustele. See praktika on vaieldamatult turvaoht integraallülituste tootmisel. Nende kiibi pinnalaotuse disainide avaldamine kolmandatele osapooltele võib paljastada kaubamärgi intellektuaalomandi saladusi. Halvima stsenaariumi korral võib sellises tehases olev petturlik element pahatahtlikel põhjustel kiibi pinnalaotust manipuleerida. Jaotatud tootmine on paljulubav kaitsetehnika, mis aitab ületada integraallülituste tootmise allhangetega seotud muresid. Jaotatud tootmisel toodetakse FEOL-i (Front End of Line) kihte (transistorid ja alumised metallikihid) ebausaldusväärses kõrgekvaliteedilises kiibitootmis tehases. BEOL (Back End of Line) kihid (kõrgemad metallikihid) on toodetud usaldusväärses madala kvaliteediga tehases. Käesolevas lõputöös esitatud uuring on üksikasjalik ülevaade antud tehnikast, paljudest rünnakutest jaotatud tootmise vastu ja kirjanduses kirjeldatud võimalikest kaitsetehnikatest. Lühidalt on välja toodud erinevad ohumudelid ja eeldused rünnakute kohta. Kaitsetehnikate uuringud on klassifitseeritud järgnevalt: lähedus põhine häirimine, traadi tõstmine ja pinnalaotuse hägustamine. Ülevaate peamine tulemus on tuua esile lahknevus paljude uuringute vahel – väidetavalt on võimalik riistvara kirjeldust rekonstrueerida peaaegu täiusliku täpsusega. Seevastu on esitatud vaid marginaalt edu BEOL-ühenduste taastamisel. Lõpuks arutatakse jagatud tootmisega seotud tulevikusuundumusi ja väljakutseid, sealhulgas tehnika tõhususe hindamise raskusi.

Riistvaratroojalaste lisamise metootika tootmisvalmis pinnalaotus disainile. Üks paljudest potentsiaalsetest ohtudest integraallülitustele tootmise ajal on riistvaralise troojalase lisamine. Kirjanduses on korduvalt demonstratreeritud riistvaralisi troojalasi, mõned isegi ränis; aga ükski neist ei avaldanud, kuidas nende riistvara troojalane on sisestatud. Selles lõputöös pakkusin välja tervikliku raamistiku, mis põhineb ECO-funktsioonil riistvara troojade (HT) sisestamiseks tootmisvalmis pinnalaouts disainile ja uudset külgkanali riistvara trooja (SCT) arhitektuuri, et demonstreerida antud raamistikku. SCT on loodud toetama külgkanalite rünnakuid, mida kutsutakse esile kontrollitud energiatarbimisega. SCT sisestamine on üksikasjalikult kirjeldatud, näidates, et tehases olev petturlik element suudab seda vaevata korrata. Kirjeldatud SCT rünnak valideeriti välja töötatud ASIC prototüübis, mis oli toodetud 65 nm CMOS-tehnoloogias. ASIC-prototüüp koosneb neljast krüptotuumast, kahest AES-i versioonist ja kahest PRESENT-i versioonist, millest igaüks on muudetud SCT-ga. Kiibi testimise tulemused näitasid rünnaku edukust kõigi katsete puhul, kus krüptovõti ekstraheeriti voolutarbe karrakteristiku kaudu. Mõõtmised näitasid ka SCT vastupidavust tootmisprotsessist tulenevate variatsioonide vastu. Peale selle sisestati testkiibile kõik 4 SCT-d vähem kui kahe tunniga, muutes antud rünnaku reaalseks ohuks tegeliku tootmise korral, kuna sellel on piiratud ajavahemik.

Lisaks esitletakse HT-de pimesi lisamist raamistiku täiendatud versioonis, et veelgi demonstreerida ECO raamistiku võimeid – nimega BioHT. Seega kasutab BioHT raamistik pöördprojekteerimise tehnikaid, et viia integraallülitustesse keerukaid troojalasi, omades sealhulgas vähe teadmisi antud disaini kohta. Lisaks näitasid BioHT katsed, et täielik lähenemine on kiire, võimaldades kasutajal troojade

lisamist tootmisvalmis disaini ja reaalse tootmise vahelisel ajal mitu korda läbi viia. Seega tekib võimalus mitme võimaliku toojalase hulgast valida välja kõige optimaalsem. Lisaks näitab BioHT ka seda, kuidas realistlik trooja sisestamine toimuks, ning võib suunata riskianalüüsi, kaitset ja ka sellel teemal tulevasi uuringuid.

Appendix 1

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

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Digital Object Identifier 10.1109/ACCESS.2020.3029339

A Survey on Split Manufacturing: Attacks, Defenses, and Challenges

TIAGO D. PEREZ, SAMUEL PAGLIARINI
Tallinn University of Technology (TalTech)
Department of Computer Systems
Centre for Hardware Security
Tallinn, Estonia
(e-mails: {tiago.perez,samuel.pagliarini}@taltech.ee)

Corresponding author: T. D. Perez (e-mail: tiago.perez@taltech.ee).

This work was supported by the European Union through the European Social Fund in the context of the project "ICT programme".

ABSTRACT

In today's integrated circuit (IC) ecosystem, owning a foundry is not economically viable, and therefore most IC design houses are now working under a fabless business model. In order to overcome security concerns associated with the outsorcing of IC fabrication, the Split Manufacturing technique was proposed. In Split Manufacturing, the Front End of Line (FEOL) layers (transistors and lower metal layers) are fabricated at an untrusted high-end foundry, while the Back End of Line (BEOL) layers (higher metal layers) are manufactured at a trusted low-end foundry. This approach hides the BEOL connections from the untrusted foundry, thus preventing overproduction and piracy threats. However, many works demonstrate that BEOL connections can be derived by exploiting layout characteristics that are introduced by heuristics employed in typical floorplanning, placement, and routing algorithms. Since straightforward Split Manufacturing may not afford a desirable security level, many authors propose defense techniques to be used along with Split Manufacturing. In our survey, we present a detailed overview of the technique, the many types of attacks towards Split Manufacturing, as well as possible defense techniques described in the literature. For the attacks, we present a concise discussion on the different threat models and assumptions, while for the defenses we classify the studies into three categories: proximity perturbation, wire lifting, and layout obfuscation. The main outcome of our survey is to highlight the discrepancy between many studies – some claim netlists can be reconstructed with near perfect precision, while others claim marginal success in retrieving BEOL connections. Finally, we also discuss future trends and challenges inherent to Split Manufacturing, including the fundamental difficulty of evaluating the efficiency of the technique.

INDEX TERMS Hardware Security, Hardware Trojans, Integrated Circuits, IP Theft, Reverse Engineering, Split Manufacturing

I. INTRODUCTION

Counterfeiting and intellectual property (IP) infringement are growing problems in several industries, including the electronics sector. In Europe, for instance, seizures of counterfeit electronics products increased by almost 30% when comparing the 2014-2016 to the 2011-2013 period [1]. Legitimate electronics companies reported about $100 billion in sales losses every year because of counterfeiting [2].

As electronic systems are being increasingly deployed in critical infrastructure, counterfeit and maliciously modified integrated circuits (ICs) have become a major concern. The globalized nature of the IC supply chain contributes to the problem as we lack the means to assess the trustworthiness of the design and fabrication of ICs. It is conceivable – if not likely – that a fault in a low-quality counterfeit IC (or even a maliciously modified IC) will effectively disrupt critical infrastructure with grave consequences. Therefore, hardware security has gained more attention in the past decades, emerging as a relevant research topic.

As the IC supply chain has become more globalized, ensuring the integrity and trustworthiness of ICs becomes more challenging [3]. When a modern IC is conceived, the probability that all involved parties are trusted is, in practice, close to zero. The process of conceiving an IC can be broken down into three major steps: design, manufacturing, and validation. *Designing* an IC involves arranging blocks and

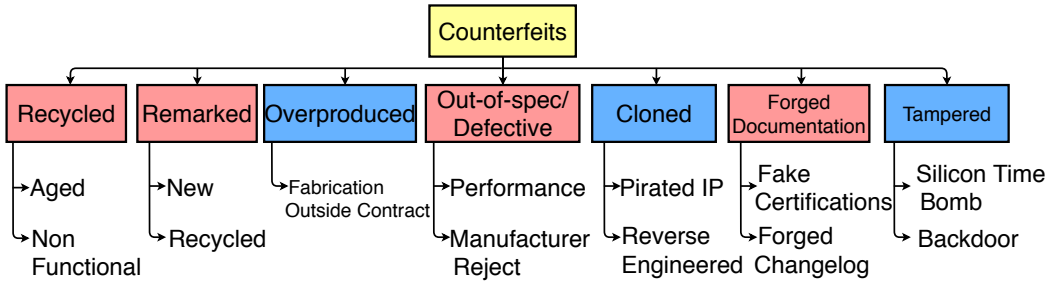

FIGURE 1. Taxonomy of counterfeit electronics (adapted from [3]).

their interconnections. Some blocks are in-house developed, while some are third-party IPs. Finally, a layout is generated by instantiating libraries that might also be in-house developed or provided by third parties. The resulting layout is then sent to a foundry for *manufacturing*. The process of *validation* requires test for physical defects as well as verification of packaged parts for correct functionality. Both test and packaging facilities may be untrusted, as these efforts are often offshored. Thus, in order to produce an IC, sensitive information almost inevitably is exposed to untrusted parties. Today's reality is that ICs are vulnerable to many hardware-based threats, including insertion of hardware trojans, IP piracy, IC overbuilding, reverse engineering, side-channel attacks, and counterfeiting. These threats are discussed in details in [4].

Hardware trojans, in particular, are malicious modifications to an IC, where attackers insert circuitry (or modify the existing logic) for their own malicious purposes. This type of attack is (typically) mounted during manufacturing, as the foundry holds the entire layout and can easily identify critical locations for trojan insertion. Third-party IPs can also contain trojans/backdoors that may contain hidden functionalities, and which can be used to access restricted parts of the design and/or expose data that would otherwise be unknown to the adversary.

IP piracy and IC overbuilding are, essentially, illegal ownership claims of different degrees. As said before, during designing an IC, third-party and in-house developed IPs are utilized. The untrusted foundry (or a rogue employee of it) can copy one of those IPs without the owner's authorization. Similarly, malicious foundries can manufacture a surplus of ICs (overbuilding) without the owner's knowledge, and sell these parts in the grey market.

Reversing engineering of ICs has been extensively demonstrated in the specialized literature [5]. An attacker can identify the technology node and underlying components (memory, analog, and standard cells), from which a gate-level netlist can be extracted and even a high-level abstraction can be inferred [6]. Reverse engineering can be effortlessly executed during manufacturing, as the foundry holds the entire layout and most likely promptly recognizes some of the IP as well. After fabrication, – when ICs are already packaged and deployed – reverse engineering is more laborious but can still be executed by a knowledgeable adversary.

According to [3], counterfeit components are classified into seven distinct categories, as illustrated in Figure 1. Recycled, remarked, out-of-spec/defective, and forged documentation are intrinsic after-market problems, where products are offered by parties other than the original component manufacturer or authorized vendors. These cases are highlighted in red. On the other hand, overproducing, cloning, and tampering are problems faced during the designing and/or fabrication of ICs. These cases are highlighted in blue. For this reason, in this paper, we will focus on these threats. It is important to realize that these threats could be avoided if a trusted fabrication scheme was in place. However, the escalating cost and complexity of semiconductor manufacturing on advanced technologies made owning an advanced foundry unfeasible for design companies, which now have the tendency to adopt the fabless business model [7]. Consequentially, outsourcing of the manufacturing exposes their entire layout to untrusted foundries, leaving their designs vulnerable to malicious attacks.

While many *ad hoc* techniques have been proposed to individually combat these threats, very few solutions directly address the lack of trust in the fabrication process. Split Manufacturing stands out from other techniques as it promotes a hybrid solution between trusted and untrusted fabrication. The technique was first pitched to DARPA circa 2006 in a white paper authored by Carnegie Mellon and Stanford universities. Later, it was picked by IARPA which then launched the Trusted IC program [8] that successfully stewarded much of the research in the area and led to this survey.

In Split Manufacturing, the key concept is to *split* the circuit in two distinct parts before manufacturing, one containing the transistors and some routing wires, and the other containing the remaining routing wires. These parts are then fabricated in different foundries. The anatomy of an IC is illustrated in Figure 2, containing two set of layers, the bottom layer where the transistors are built, called Front end of the Line (FEOL), and the top layer where the metal layers are built for routing purposes, called Back end of Line (BEOL). The metal layers are referred as MX, where X is the level of

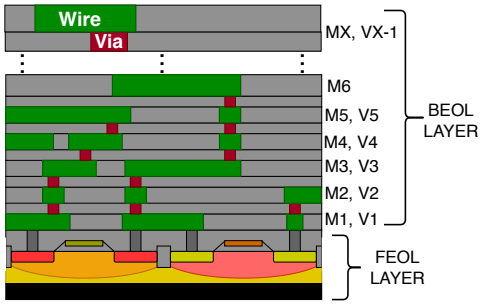

FIGURE 2. Anatomy of an integrated circuit (adapted from [10]).

the layer. M1 is the lowest layer at the bottom of the stack. Connections between metals are made by vias referred as VX, following the same naming scheme for metals. In Split Manufacturing, the FEOL is first manufactured in a high-end foundry, and later the BEOL is stacked on top of it by a second (and possibly low-end) foundry. This process requires electrical, mechanical, and/or optical alignment techniques to ensure the connections between them. Additionally, FEOL and BEOL technologies have to be compliant with each other [9] regarding the rules for metal/via dimensions where the split is to be performed. The split can be performed at the lowest metal layer (M1) or at higher layers, for which trade-offs are established between attained security and overheads. If the BEOL and FEOL technologies are vastly different from one another, Split Manufacturing may incur heavy overheads.

In this work, the focus is on the Split Manufacturing technique. As described above, Split Manufacturing can tackle threats that occur during the fabrication. It avoids overproduction, reverse engineering (to some extent) and unwanted modifications, limiting the capability of attackers. In Section II, we provide a background and more in-depth explanation of the technique. We address security threats in Section II, demonstrating the potential vulnerabilities found in split circuits and describing the state-of-the-art attacks proposed until the present day. In Section IV, the security of split circuits is discussed, showing how it can be improved using enhancement techniques. Future trends and lessons learnt are discussed in Section V. Finally, our conclusions are presented in Section VI.

II. SPLIT MANUFACTURING: BACKGROUND

As mentioned before, in order to have access to advanced technologies, many design companies have to outsource their IC manufacturing to untrusted high-end foundries. Protecting their designs against threats that may occur during manufacturing is a concern. Designs can be protected by applying the Split Manufacturing technique, thus combating all threats highlighted in blue in Figure 1.

Split Manufacturing protects a design by hiding sensitive data from the untrusted foundry. This is achieved by splitting the IC into two parts before manufacturing, a horizontal cut that breaks the circuit into one part containing the transistors and some (local) routing wires, and another containing only routing wires. These parts are termed FEOL and BEOL.

As the FEOL and the BEOL of an IC are built sequentially, first FEOL and then BEOL, this characteristic enables the Split Manufacturing technique. Since the FEOL contains the transistors and possibly a few of the lowest ultra-thin metal layers – the most complex parts of an CMOS process [11] –, it is logical to seek to use a high-end foundry for its manufacturing, even if said foundry is not trusted. Completing the IC can then be done in a trusted low-end foundry, where the BEOL is stacked on top of the FEOL. Split Manufacturing was successfully demonstrated in [9], [12], [13], where designs were manufactured with ~0% of faults, and are reported to present a performance overhead of about 5%. Therefore, the technique is, in principle, feasible and available for design companies to make use of, such that they can utilize advanced foundries without fully exposing their layouts during manufacturing.

However, there are many caveats to Split Manufacturing. The technique can be successfully applied only if the technologies used to build the FEOL and BEOL are "compatible". In theory, a layout can be split at any layer if the chosen layer presents a good interface between FEOL and BEOL. However, since advanced technologies utilize the dual-damascene fabrication process, the layout can only be split on metal layers [14]. Thus, the FEOL cannot terminate in a via layer. The dual-damascene process is characterized by patterning the vias and trenches in such a way that the metal deposition fills both at the same time, i.e., via-metal pairs (e.g., V1 and M2) must mandatorily be built by the same foundry.

Two technologies are said to be compatible with each other if there is a way for a BEOL via to land on the FEOL uppermost layer while respecting all design rule checks (DRCs) of both technologies. DRCs are used to guarantee the manufacturability and functionality of an IC, and are defined with respected to the characteristics of the materials utilized and to tolerance ranges of the manufacturing processes (e.g., polishing, patterning, and deposition). These rules encompass minimum enclosure, width, spacing, and density checks. Modern technologies have several options for via shapes – as long as one via shape is valid, the technologies are compatible for Split Manufacturing purposes. However, in practice, to keep the overhead of the technique under control, an array of via shapes must be feasible, thus providing the physical synthesis with a rich selection for both power and signal routing.

According to [9], compatibility between two technologies can be generalized by enclosure rules as in Eq. 1, where MW.U.x is the minimum width of Mx on untrusted foundry, VW.T.x is the minimum width of Vx on trusted foundry and EN.T.x is the minimum enclosure on trusted foundry. These rules are illustrated in Figure 3, where the left side of the image portrays a cross section view and the right side shows the top view. According to Figure 3, the minimum

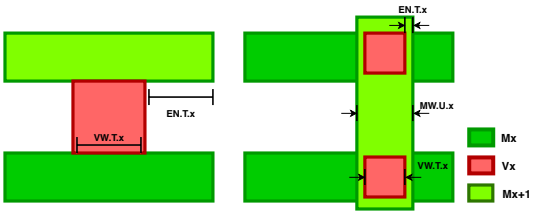

FIGURE 3. Compatibility rules between FEOL and BEOL (adapted from [9]).

enclosure width, Mx.EX.Vx, must be compatible between the two foundries. In modern technologies, Equation 1 is no longer sufficient as it does not capture the intricate rules for vias and line endings (enclosure from 1 side, 2 sides, 3 sides, T-shaped/hammerheads, etc.).

$$MW.U.x \geq VW.T.x + (2EN.T.x) \tag{1}$$

Split Manufacturing also presents challenges on the design flow front, which is illustrated in Figure 4. An in-house team designs the circuit, from RTL to layout. Most likely, the layout contains IPs obtained from third parties. Depending on the metal layer where the layout is to be split, it may affect existing IP. Logic and memory IP may use higher metal layers – memories typically require 4 to 5 metal layers, while standard cells typically require 2 metal layers –, limiting where the split can be done. Standard cells and memories have to be re-designed if they use metal layers that will be split, a grave challenge that may render Split Manufacturing significantly harder to execute.

Still referring to Figure 4, the FEOL and BEOL are generated using a hybrid process design kit (PDK), and then later split to be manufactured. After splitting the layout correctly, the FEOL is first manufactured in a high-end foundry, and later the BEOL is stacked on top of it by a second (and possibly low-end) foundry.

Even by splitting the layout, it is often argued that the FEOL exposes enough information to be exploited. Attacks towards the FEOL can effectively retrieve the BEOL connections by making educated guesses. The efficiency of the guessing process is inherently linked to the threat model assumed, which determines the information the attacker possesses to begin with. The literature describes two distinct threat models:

- **Threat model I:** an attacker located at the untrusted foundry holds the FEOL layout and wants to retrieve the BEOL connections.
- **Threat model II:** an attacker located at the untrusted foundry holds the entire gate-level netlist that is assumed to be provided by a malicious observer. The attacker here still holds the FEOL layout and wants to retrieve the BEOL connections. [15].

It is important to emphasize that the second threat model completely nullifies the security introduced by Split Manufacturing. Possessing the gate-level netlist makes reverse engineering the layout a trivial task, as if the attacker held the entire layout, not only the FEOL. Assuming the attacker has knowledge about the netlist challenges the integrity of the design company itself. If a rogue element exists inside the design company, other representations of the design could be equally stolen, such as the register-transfer level (RTL) code of the design, or even the entire layout (including the BEOL). It could be argued that this vulnerability is so severe that Split Manufacturing virtually stops making sense. For this reason, threat model I is the focus in this work. However, as our goal is to present a comprehensive survey, related works that use threat model II will be covered as well.

Assuming threat model I, an attacker already knowing all the layers that make up the FEOL, is now interested in retrieving the BEOL connections to recreate the full design (or as close as possible). The commonly used assumption is that attackers are powerful and work within the untrusted foundry in some capacity. Thus, the attackers have deep understanding about the technology. Extracting the (still incomplete) gate-level netlist from a layout is, therefore, a trivial task for the attacker.

Many approaches to retrieve the BEOL connectivity have been proposed, several of which are termed *proximity attacks* [10], [16], [17]. Since EDA tools focus on optimizing power, performance, and area (PPA), the solution found by a placement algorithm (that uses heuristics internally) tends to place connected cells close to one another as this will, in turn, reduce area, wirelength, and delay. Therefore, finding the correct missing connections between FEOL and BEOL can be done by assessing input and output pins that are in proximity (thus, the name proximity attack). The more input and output pins to connect, the higher is the probability to make a wrong connectivity guess. Thus, a higher level of security is achieved by splitting the circuit at the lowest metal layer possible.

As a promising technique to enhance the security of ICs in this era of fabless chip design, Split Manufacturing still faces some enormous challenges:

> *Logistical challenge:* Split Manufacturing is not presently incorporated into the IC supply chain. Finding foundries with compliant technologies that are willing to work with each other is not trivial.
> *Technological challenge:* even within compliant technologies, non-negligible overheads can be introduced if they are vastly different[1]. In the worst-case scenario, it can make routing impossible. Thus, this fact narrows down the technology choices available and feasibility of certain layers as candidates for splitting.
> *Security challenge:* the attained security of straightforward Split Manufacturing is still under debate. Attacks towards the FEOL can be effective, where the hidden connections can be retrieved.

[1]For a thorough discussion and silicon results on BEOL-related overheads, please refer to [18].

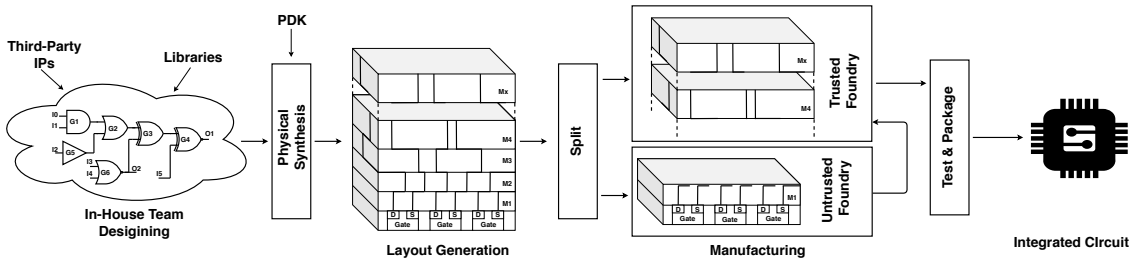

FIGURE 4. Split Manufacturing Design Flow.

For the purpose of this survey, we categorized related works in the literature as attacks and defenses. In *attacks*, authors proposed new attack models or modifications of existing attacks in order to improve their effectiveness. In *defenses*, authors proposed new techniques to use together with Split Manufacturing in order to improve its security level.

III. ATTACKS ON SPLIT MANUFACTURING

The Split Manufacturing technique was developed to protect ICs against threats related to manufacturing in potentially untrusted foundries. In practical terms, to split the layout means to hide some connections from the untrusted foundry. The security provided by Split Manufacturing is based on the fact that the attacker in the FEOL foundry cannot infer the missing BEOL connections. This assumption, however, was challenged by several works where authors proposed attack approaches that can potentially retrieve the missing connections with varying degrees of success. In the text that follows, we present works that proposed Split Manufacturing attacks. These attacks are discussed in chronological order as compiled in Table 1. For each studied attack, we reported the related threat model, attack type, novelty, and benchmark circuits used in experiments. Additionally, we reported the largest and average size of the circuits utilized in each work (measured in number of gates). The circuit size is an important metric when analyzing the Split Manufacturing technique because its effectiveness is often proportional to circuit size. In Table 2, we compiled results for when the smallest and largest studied circuits are under attack. Also, we reported the circuit size and, if defined, the split layer that the author selected to split the circuit at.

The first reported attack is by Jeyavijayan *et al.* and is described in [10]. In this work, the authors assumed that naive Split Manufacturing (i.e., splitting a layout without care for the connections) is inherently insecure. They introduced the concept of proximity attacks that exploits "vulnerabilities" introduced by EDA design tools. Since EDA tools focus on optimizing for PPA, the solution found by a placement algorithm tends to place logically connected cells close to one another so they become physically connected during routing. Therefore, the distance between output-input pairs can be used as a metric to recover the missing BEOL connections.

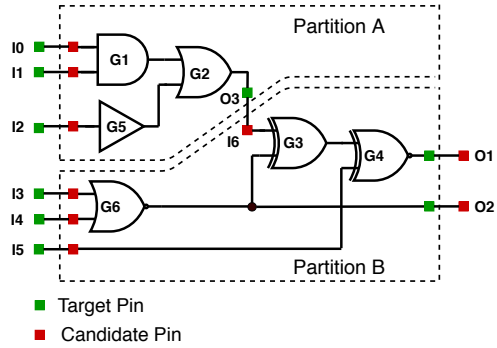

FIGURE 5. Example of a partitioned circuit.

Designs are commonly partitioned during physical implementation, i.e., separated into small logical blocks with few connections between them. That way, the designer has total control of the floorplaning regarding the placement of blocks. This approach also allows for blocks to be implemented separately and later integrated, creating a sense of parallelism in the design flow which can reduce the overall time required for executing physical synthesis. Consider as an example the circuit illustrated in Figure 5 which contains two partitions, A and B, each with 3 gates. Partition A has 3 input pins and one output pin, while partition B has three input pins and two output pins. The partitions have only one connection between them, connecting the output pin of gate G2 with one of the inputs of gate G3. Consider a target output pin from partition A $P_{x,A,out}$ and its corresponding candidate input pin from partition B $P_{x,B,in}$. During placement, the EDA tool will attempt to place the pin $P_{x,A,out}$ as close as possible to $P_{x,B,in}$, possibly closer than any other pin in partition B. Using this insight, an attacker may recover the missing connections in the FEOL layout, performing then a proximity attack. The authors argued that their proposed attack flow is successful due to being able to leverage the following "hints" provided by the EDA tools:

Hint 1 - Input-Output Relationship: partition input pins are connected either to another partition output pin or to

an input port of the IC (i.e., input to input connections are excluded from the search space).

Hint 2 - Unique Inputs per Partition: input-output pins between partitions are connected by only one net. If a single partition output pin feeds more than one input pin, the fan-in and fan-out nodes are usually placed within the partitions (i.e., one-to-many connections are ruled out from the search space).

Hint 3 - Combinational Loops: in general, only very specific structures are allowed to utilize combinational loops (e.g., ring oscillators). These structures are very easy to identify. In most cases, random logic does not contain combinational loops (i.e., connections that would lead to combinational loops can be excluded from the search space).

An attacker can correctly connect a target pin to a candidate pin by identifying the closest pin from a list of possible candidates. The list of possible candidates is created by observing the hints mentioned above. A possible candidate pin is an unassigned output pin of another partition and an unassigned input port of the design. Then, a minimum distance metric is used to connect the pins based on the previously discussed heuristic behavior of EDA tools.

In Algorithm 1, we described the proximity attack detailed in [10]. The input to the algorithm is the FEOL layout, from which the information about unassigned input-output ports can be derived. The algorithm does not describe the specifics of how to derive a netlist from a layout. However, the complexity of this task is rather straightforward. It is assumed that the attacker possesses information about both the PDK and the standard cell library. In many cases, the untrusted foundry is the actual provider of both[2]. From there, a layout in GDSII or OASIS format can be easily reverted to a netlist by any custom design EDA tool.

From the gate-level netlist, the algorithm chooses an arbitrary *TargetPin* from the unassigned partition input pins and output ports, creates a list of possible *CandidatePins*, and then connects the *TargetPin* to the closest pin in this list. After each connection, the netlist is updated. This procedure is repeated until all unassigned ports are connected. When the procedure is over, the attacker obtains the possible missing BEOL connections. If all guesses were correct, the original design has been recovered and Split Manufacturing has been defeated.

Algorithm 1 was originally applied to the ISCAS'85 [24] suite of benchmark circuits. These circuits were originally selected and published to help in comparing automatic test pattern generation (ATPG) tools. Due to the small size of these circuits, they may not be the best option to assess the effectiveness of Split Manufacturing. The difficulty to retrieve the BEOL connections is directly proportional to the size of the circuit. The authors reported an average

[2]For the PDK, it is very natural that it is created by the untrusted foundry itself. For standard cell libraries, the cells might be designed by the foundry or by a third-party licensed by the foundry. In either case, the effort to revert a layout to a netlist remains trivial.

Algorithm 1: Proximity attack

```
Input: FEOL layers
Output: Netlist with BEOL connections
Reverse engineer FEOL layers and obtain partitions;
while Unassigned partition pins or ports exist do
    Select arbitrary unassigned pin/port as a targetPin;
    ListOfCandPins = BuildCandPinsList(targetPin);
    Select candPin from ListOfCandPins that is closest to
      targetPin;
    Connect targetPin and candPin;
    Update netlist;
Return: netlist
BuildCandPinsList(targetPin)
Input: targetPin P_{X,i,in}
Output: CandPins for targetPin
CandPins = Unassigned output pins of other partitions +
  unassigned input ports of the design;
for each Pin_J ∈ CandPins do
    if CombinationalLoop(targetPin, Pin_J) then
        CandPins -= Pin_J;
Return: CandPins
```

effectiveness of 96% of Correct Connection Rate (CCR) across all the benchmarks considered. For the c17 circuit (smallest circuit in the ISCAS'85 suite with only 6 gates), all the connections were retrieved correctly, thus demonstrating that the algorithm is capable of retrieving the missing BEOL connections. In Table 2, we highlight the best and worst results in terms of CCR.

Jeyavijayan *et al.* [10] were the first to question the security of straightforward Split Manufacturing. Their proximity attack showed promising results, even if the considered benchmark circuits were rather small in size. This was the starting point for other studies proposing different attacks to Split Manufacturing in an attempt to retrieve the missing BEOL connections. Improvements over the original proximity attack, as well as other attacks, are compiled in Table 1.

The effectiveness of the proximity attack utilizing distance of unassigned pins alone as metric to find missing BEOL connections was questioned by Magaña *et al.* [19]. The authors proposed to utilize both placement and routing information in augmented proximity attacks. For their results, large-sized circuits from the ISPD-2011 routability-driven placement contest [25] were used. These benchmarks are better representatives of modern circuits as they contain 9 metal layers and up to two million nets in a design. Thus, in an attempt to increase the success rate of the attack for large-sized circuits, they proposed routing-based proximity in conjunction with placement-centric proximity attacks.

A key difference present in [19] is that this work utilizes a different threat model (model II), claiming that the untrusted foundry possesses information about the entire place & routed netlist (as well as the FEOL layout). This assumption is hard to reason if the attacker's intent was to overproduce the IC or pirate the IP. For these goals, clearly, this assumption is unnecessary. The attacker himself can, if he indeed

TABLE 1. Threat Models, Attacks, and Metrics.

Work	Year	Threat model	Attack type	Novelty	Benchmark suite(s)	Largest circuit size (gates)	Avg. circuit size (gates)
[10]	2013	I	Proximity	Attack Based on Proximity	ISCAS'85	3.51k	1288
[19]	2016	II	Proximity	Placement and routing proximity used in conjunction	ISPD'11	1.29M	951k
[20]	2018	I	Proximity	Network-Flow-Based with Design Based Hints	ISCAS'85 & ITC'99	190.21k	9856
[21]	2018	I	Proximity	Proximity Attack Based on Machine Learning	ISPD'11	1.29M	951k
[22]	2019	I	SAT	SAT Attack without Proximity Information	ISCAS'85 & ITC'99	190.21k	9856
[23]	2019	I	SAT	SAT attack dynamically adjusted based on proximity information	ISCAS'85 & ITC'99	190.21K	9856

TABLE 2. Benchmark Size and Comparison of Attack Results.

Work	Benchmark	Attack	Split Layer	Size (In Gate Count)	Metric	Result
[10]	c17	Proximity	Not Defined	6	CCR(%)	100
[10]	c7552	Proximity	Not Defined	3513	CCR(%)	94
[19]	Superblue 1	Placement Proximity	M2	847k	% Match in List	12.84
[19]	Superblue 1	Placement Proximity	M2	847k	CCR(%)	5.479
[19]	Superblue 1	Routing Proximity	M2	847k	% Match in List	71.08
[19]	Superblue 1	Routing Proximity	M2	847k	CCR(%)	0.651
[19]	Superblue 1	Overlap (P&R) Proximity	M2	847k	% Match in List	13.05
[19]	Superblue 1	Overlap (P&R) Proximity	M2	847k	CCR(%)	3.977
[19]	Superblue 1	Crouting Proximity	M2	847k	% Match in List	82.08
[19]	Superblue 1	Crouting Proximity	M2	847k	CCR(%)	0.651
[20]	c7552	Network-flow Based Proximity	Not Defined	3513	CCR(%)	93
[20]	c7552	Proximity	Not Defined	3513	CCR(%)	42
[20]	B18	Network-flow Based Proximity	Not Defined	94249	CCR(%)	17
[20]	B18	Proximity	Not Defined	94249	CCR(%)	< 1
[21]	Superblue 1	Proximity	M6	847k	% Match in list	33.40
[21]	Superblue 1	Proximity	M6	847k	CCR(%)	0.76
[21]	Superblue 1	ML	M6	847k	% Match in list	83.12
[21]	Superblue 1	ML	M6	847k	CCR(%)	1.91
[21]	Superblue 1	ML-imp	M6	847k	% Match in list	74.65
[21]	Superblue 1	ML-imp	M6	847k	CCR(%)	2.11
[21]	Superblue 1	ML-imp	M4	847k	% Match in list	75.45
[21]	Superblue 1	ML-imp	M4	847k	CCR(%)	2.58
[22]	c7552	SAT Attack	Not Defined	3513	Logical Equivalence(%)	100
[22]	B18	SAT Attack	Not Defined	94249	Logical Equivalence(%)	100
[23]	c7552	Improved SAT Attack	Not Defined	3513	Logical Equivalence(%)	100
[23]	B18	Improved SAT Attack	Not Defined	94249	Logical Equivalence(%)	100

possesses the netlist, perform his own physical synthesis and generate his own layout. The interest in reverse engineering the BEOL connections of the original design diminishes. Nevertheless, we report on the strategies employed by the authors of [19] since they build on the approach proposed by [10].

Regarding the attacks, the authors of [19] proposed four different techniques to identify a small search neighborhood for each pin. The goal is to create a neighborhood that is small enough to make further pruning feasible, and therefore increase the likelihood of including the matching pins. The techniques are called *placement proximity*, *routing proximity*, *crouting proximity* and *overlap of placement and routing proximity*, and are described in the text that follows. The circuit illustrated in Figure 6 is the example (before the split) that will guide the discussion on these four techniques.

Placement proximity exploits the placement information of cells. Each split wire is taken from the pin location of the corresponding standard cell that is connected to it. A search neighborhood is defined as a square region centered around the corresponding pin with an area equal to the average areas of the bounding boxes (BB) in a typical design. The authors argued that it can also be measured based on BBs of the non-split wires in the design under attack, under the assumption that the number of wires that remain in the FEOL is also very large in practice. Let us consider the circuit illustrated in Figure 6 as an example. If the split is done at M2, the search area defined using the placement proximity would contain three gates as illustrated in Figure 7 (a). Thus, using the placement proximity search area, the most likely connections are illustrated by the green squares (candidate pins) and by the red squares (target pins). Note that the layer at which the layout is split does not affect the search area defined by placement proximity.

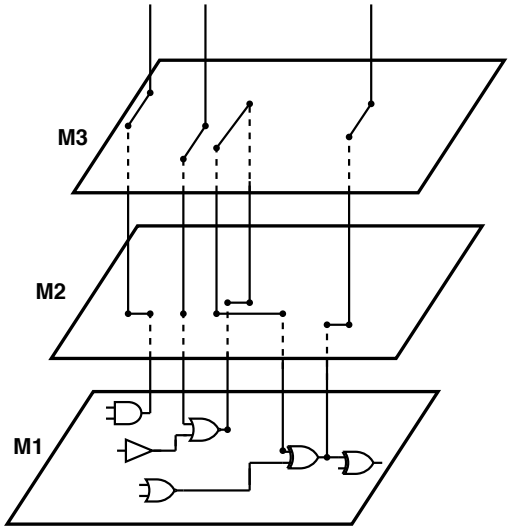

FIGURE 6. Representation of the routing for the first 3 metal layers of a simple circuit (adapted from [19]).

Routing proximity exploits the routing information. First, for each split wire, pins are identified as the point where the wire is actually cut at the split layer, i.e., the via location. Next, a square area centered around those pins is defined. The size of the square area is defined based on the average BBs of the pins on that layer in the design. This procedure for identification results in different neighborhood sizes according to the split layer location, i.e., the search radius adapts to the routing resources of each layer. A search area defined using routing proximity is illustrated in Figure 7 (b), highlighted in gray and containing a set of routing wires and its respective pins.

Crouting proximity takes into account routing congestion by exploiting the union of placement and routing proximity. The search area for each pin is defined in such way that the ratio of number of pins to the search area is equal across all the pins in the split layer. Thus, if a pin is located at a high routing congestion area, the search area will be expanded until the pin density in the new search area reaches a target value or the search area grows to four times its starting value. The starting value is set according to the split layer, set as the average of numbers of pins which fall within a BB. A search area defined using crouting proximity is illustrated in Figure 7 (c).

The last strategy proposed by [19] also combines placement and routing information. It is referred to as *Overlap of placement and routing proximities*. The concept here is to include a subset of pins identified by the placement proximity list which have their corresponding pins included in the routing proximity list. According to the author, intuitively, the overlap then identifies a subset of pins which may be more likely to point towards the direction of the matching pin. A search area defined using the overlap of placement and routing proximities is illustrated in Figure 7 (d).

Magaña et al. [19] assessed each strategy using the benchmark circuit *superblue1*. Different split layers were also considered. In Table 2, we compiled the results for split layer M2. By comparing the results, it becomes clear that no strategy was able to recover 100% of the missing BEOL connections. The best result was only 5.479% of CCR. This is in heavy contrast with the findings of [10]. However, as we previously noted, the circuit sizes differ by orders of magnitude.

According to the authors of [19], proximity alone is in no way sufficient to reverse engineer the FEOL. However, proximity attacks have merit as they can be used to narrow down the list of candidates to a significantly smaller size. Using crouting proximity, in 82.08% of the cases, the search area defined contained the matched pin in the list of candidates. The authors also present results for a circuit split at M8. We opt not to show these results in Table 2. Using the circuit *superblue1* as an example, the number of unassigned pins when the circuit is split at M8 is only 1.2% of the pins when split at M2. Therefore, the small number of unassigned pins to be connected overshadows the large circuit used for their experiments. It must also be emphasized that splitting a circuit in such higher layers is rather impractical since M8 tends to be a very thick metal reserved for power distribution in typical 10-metal stacks. There is very little value in hiding a power distribution network from an adversary that wants to pirate an IP. Once again, we opt not to show this result in our comparisons.

A network-flow based attack model towards flattened designs was proposed by Wang et. al [20]. The authors argued that the proximity attack originally proposed by [10] utilizes hints that can be used only by hierarchical designs, and that modern designs are often flattened [3]. Based on the original proximity attack, they proposed a proximity attack utilizing five hints: physical proximity, acyclic combinational logic circuit, load capacitance constraint, directionality of dangling wires, and timing constraint. Note that the first two hints are already described by [10] and [19]. The three novel hints are described below:

> *Load Capacitance Constraint*: gates can drive a limited load to honor slew constraints. Typically, maximum load capacitance is constrained and has a maximum value defined by the PDK and/or the standard cell characterization boundaries. Hence, an attacker will consider only connections that will not violate load capacitance constraints.
> *Directionality of Dangling Wires*: routing engines tend to route wires from a source to a sink node along the direction of the sink node. Therefore, the directionality of remaining dangling wires at lower metal layers may

[3]We highlight that best practices in circuit design have changed over the years. Hierarchical design was heavily utilized for many years, but it lost favor due to the difficulty in performing reasonable timing budgeting between the many blocks of a system. Thus, flattened designs are often used to facilitate timing closure.

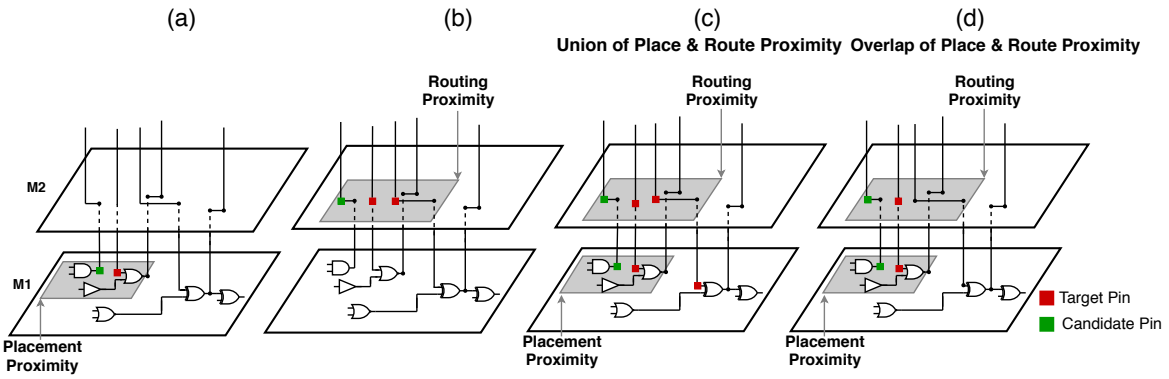

FIGURE 7. Multiple strategies for pin/connectivity search areas according to [19].

indicate the direction of their destination cell with a high degree of certainty[4]. An attacker can disregard connections in the other directions.

Timing Constraint: connections that create timing paths that violate timing constraints can be excluded. An attacker, through an educated guess of the clock period, can determine a conservative timing constraint and exclude any connections that would lead to slower paths.

The network-flow based attacked framework proposed by Yang *et al.* considers two hints proposed by [10] plus the aforementioned hints to create a directed graph $G = (V, E)$, where V is a set of vertices and E is a set of edges. The set (V) is composed by the set of vertices corresponding to the output pins V_o, and a set corresponding to the input pins (V_i), the source vertex (S), and the target vertex (T). The set E consists of E_{So}, edges from S to every output pin vertex, E_{oi}, edges from output pin vertices to input pin vertices, and E_{iT}, which includes edges from every input vertex to the target vertex. An example of this kind of representation is shown in Figure 8, where (a) is the circuit with missing connections and (b) is the network-flow representation. The detailed problem formulation is omitted from this work. To find the connections, a min-cost network-flow problem is solved, where the decision variables are the flow $x_{i,j}$ going through each edge $(i, j) \in E$. The authors utilized the Edmons-karp algorithm [26] to solve this problem. Complexity of the algorithm alone is given by $O(VE^2)$, however, in the worst case it is required to run the algorithm V times; thus, the run-time of the complete network attack is given by $O(V^2E^2)$ in the worst case.

The network-flow approach was applied to ISCAS'85 and ITC'99 [27] benchmark circuits. The ITC'99 circuits were proposed as an evolution of the ISCAS'85 set, since the Test community acknowledged that newer and larger circuits were already in demand. For comparison, the authors applied both

[4]Metals usually have preferred directions that alternate along the stack (i.e., if M1 is vertical, then M2 is horizontal). Therefore, this hint becomes more effective if the attacker can observe more than one routing layer of the FEOL

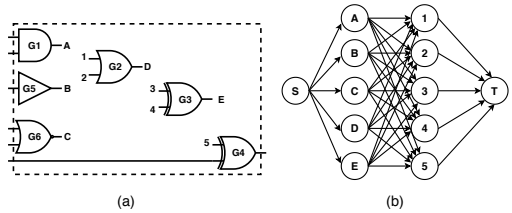

FIGURE 8. (a) Circuit with missing connections. (b) Network-flow model for inferring the missing connections. (Adapted from [20]).

the original proximity attack and the network-flow attack to flattened designs. As shown in Table 2, their network-flow proximity attack outperformed the original attack in terms of CCR. However, despite the evident improvement, the attack could only retrieve 17% of the missing BEOL connections for a medium sized circuit (b18 from the ITC'99 suite).

A Machine Learning (ML) framework was used by Zhang *et al.* [21] in an attempt to improve the attack proposed in [19]. The same setup as previously discussed was utilized. However, more layout features were incorporated in their ML formulation, including placement, routing, cell sizes, and cell pin types.

A high-level overview of their modeling framework is shown in Figure 9 (a). First, they create a challenge instance from the entire layout and only FEOL view. Next, for each virtual pin (point where a net is broken on the split layer), layout information is collected, including placement, routing, cell areas, and cell pin as illustrated in Figure 9 (b). Using this information, samples are generated which are fed into the ML training process. Each sample carries information of a pair of virtual pins which may or may not be matched. Classifiers then are built by the ML framework using training samples. After training and building the regression model, cross validation is used for evaluation which ensures validation of the model is done on data samples which were not used for training. Their framework faces scaling issues when applied to lower split metal layers. An improved ML

framework is then proposed as well, denoted by ML-imp, to solve the scaling issues.

For their experiments, Zhang *et al.* [21] utilize the ISPD'11 benchmark suite. They compare results from their previous work [19] with their ML and ML-imp frameworks. However, they do not show results for lower split metal layers (e.g., M2). Instead, results are provided for M8, M6, and M4 splits. As pointed out before, utilizing higher layers for the split effectively shrinks the otherwise large circuits used in their experiments. A drastic reduction of unassigned pins is expected for such higher layers, as higher metal layers are used often for power routing, not for signal routing. Results for the *superblue1* circuit are shown in Table 2. Regarding recovering missing BEOL connections, ML and ML-imp could only retrieve around 2%, therefore not showing a huge improvement over their previous work. However, search list area accuracy showed significantly better results when compared to their prior work. A caveat worth mentioning is that the proposed machine learning framework needs the entire layout during its modeling phase. This characteristic may, in an extreme case, nullify the applicability for an attacker that only holds the FEOL layout and cannot produce training samples from other sources.

Attacks using proximity information as a metric are not the only solution to recover missing BEOL connections. An effective methodology to apply a Boolean satisfiability based strategy is proposed by Chen *et al.* [22]. The authors claim that their attack methodology does not need (or depend on) any proximity information, or even any other insights into the nature of EDA tools utilized in the design process. The key insight in their work is to model the interconnect network as key-controlled multiplexers (MUX). Initially, all combinations of signal connections between the FEOL partitions are allowed, as illustrated in Figure 10. First, a MUX network is created in order to connect all missing paths in the circuit. This MUX network leads to potential cyclic paths, thus, there is a possibility to generate many combinational loops during the attack process corresponding to incorrect key guesses. Therefore, constraints on the key values are generated in order to avoid activating the cyclic paths. The attack can be summarized in 4 steps: *identification of all cyclic paths*, *generation of cycle constraints*, *cycle constraints optimizations*, and finally, *SAT attack*. The authors utilize a SAT solver-based attack method derived from CycSat [28]. The SAT attack algorithm has as input the FEOL circuit with MUX network and a packaged IC that serves as an oracle. The algorithm outputs keys to be used in the MUXes such that correct BEOL connections are made.

In reality, [22] presents a different interpretation of Threat model I since the attacker is assumed to possess a functional IC. This IC would then have to be available in the open market for the attacker to be able to purchase it. This characteristic severely narrows down the applicability of this SAT attack. For instance, ICs designed for space or military use will not be freely available, thus an oracle may not be known to the attacker.

Experimental results presented by [22] utilize ISCAS'85 and ITC'99 benchmark circuits. It has been shown that their attack could recover a logically correct netlist for all the studied circuits. However, there is a small clarification to be made that relates to what is a logically correct circuit. In Table 2, two of those results are shown. For seven of the studied benchmarks (c1908, c2670, c5315, c7552, b14, b15, b17), the connections recovered are identical to the BEOL connections. For the remaining benchmarks, the recovered connections are not identical but logically equivalent to the original circuit. In practice, the logically equivalent circuit may present performance deviations from the original design. Matching the performance of the original design can be done by re-executing place and route using the logically equivalent gate-level netlist. Depending on the attack goal, it is possible that the attacker had already planned to re-execute the physical synthesis flow again (say, to resell the IP in a different form or shape). An attack that guarantees 100% of logic equivalence of the recovered netlist is powerful enough, allowing attackers to copy and modify split layouts.

In order to increase the efficiency and capacity of the SAT attack proposed in [22], the authors proposed two improvements in [23]. First, the size of the key-controlled interconnect network that models the possible BEOL connections is reduced. Second, after the MUX network is inserted into the FEOL circuit, the number of combinational cycles it induces in the design for incorrect key guesses should also be reduced. Proximity information is then exploited to achieve the proposed improvements. The improved SAT attack method which exploits proximity information showed significant reduction in the attack time and increase in the capacity. Same as in [22], the circuits tested were 100% recovered, as shown in Table 2.

IV. SPLIT MANUFACTURING DEFENSES

Attacks toward Split Manufacturing showed promising results, as described in the previous section. A malicious attacker has the real potential to recover the missing BEOL connections. If the missing connections are successfully recovered, the security introduced by applying the technique is nullified. Therefore, straightforward Split Manufacturing is questioned by several works. Several authors proposed defense techniques that augment the technique, i.e., techniques that, when used together with Split Manufacturing, do increase the achieved security against attacks. In Table 3, we compile a comprehensive list of defense techniques found in the literature. Each defense technique utilizes a different metric and Threat model, depending upon the type of attack they are trying to overcome. Since many of the studied defense techniques often introduce heavy PPA overheads, Table 3 also shows if the studied work assessed overheads and which ones were addressed.

In the text that follows, the many defense techniques are divided into categories, namely Proximity Perturbation (i.e., change the location of cells or pins), Wire Lifting (i.e., move routing wires to upper layers), and Layout Obfuscation (i.e.,

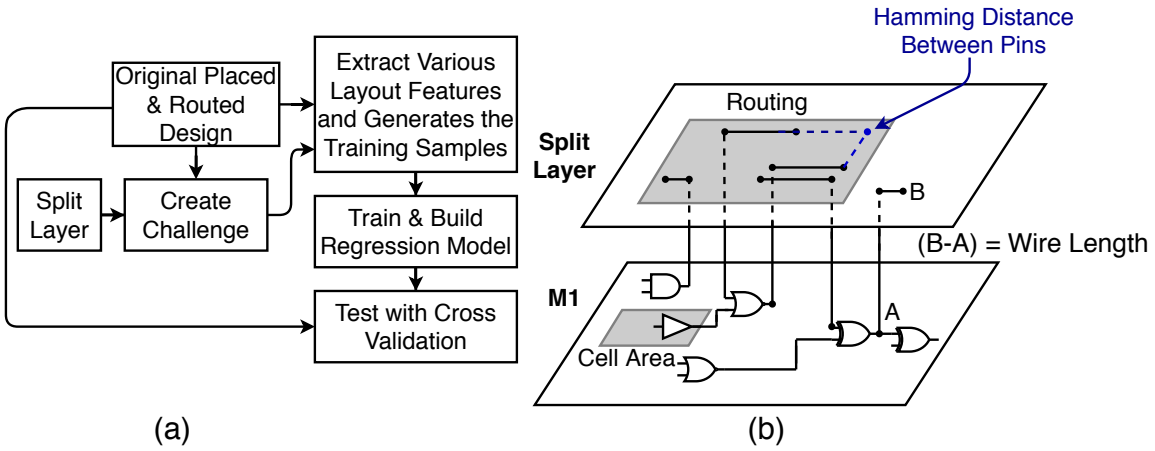

FIGURE 9. (a) Machine Learning Modeling by Zhang et al. [21]. (b) Few Exmples of Layout Features.

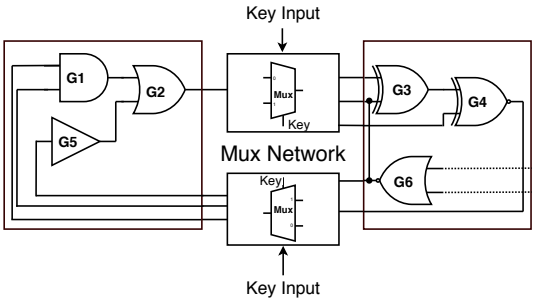

FIGURE 10. MUX network for a bipartitioned FEOL circuit (Adapted from [22]).

hide the circuit structure). We present the categories in this exact order. For some techniques, it is worth mentioning that overlaps do exist and that techniques could be categorized differently. Thus, this categorization is our interpretation of the state of the art and may not be definitive. Furthermore, the boundaries between categories are not strict. For example, a technique may perform a layout modification that promotes proximity perturbation and leads to (indirect) wire lifting. In Tables 4 and 5, we compile the results for the Proximity Perturbation and Wire Lifting categories, respectively. To demonstrate the effectiveness of each defense technique, we compile the results for when the attack is done with and without the defense. The results showed in Tables 4 and 5 are for the smallest and largest circuits addressed in each studied work. Additionally, we show the PPA overhead introduced and, if specified, the split layer.

A. PROXIMITY PERTURBATION

Attacks toward split circuits are generally based on leveraging proximity information. The first category of defenses, Proximity Perturbation, addresses this hint left by the EDA tools. The goal of the techniques within this category is to promote changes in the circuit such that the proximity information between the FEOL pins is less evident. Therefore, the success rate of the proximity attacks is decreased.

In [10], the authors proposed pin swapping to overcome proximity attacks. Rearranging the partition pins can alter their distance in such a way that the attacker is deceived. As an example, if the pins $P_{G3,B,in}$ and $P_{G6,A,in}$ (Figure 5) are swapped, the proximity attack will incorrectly guess the connection between $P_{G2,A,out}$ and $P_{G3,B,in}$. Thus, a sufficient number of pins have to be swapped in order to create a netlist that is significantly different from the original netlist (based on some sort of metric for similarity). In [10], Hamming distance is proposed as a way to quantify the difference between the outputs of the original netlist and the modified netlist. Assuming the outputs of a circuit are arranged as a string of bits, Hamming distance is defined as the number of bits that change when two instances of this string are compared to one another. The authors argued that the optimum netlist is created when the Hamming distance is 50%. Therefore, inducing the maximal ambiguity for a potential attacker. Since the best rearrangement for N pins of partitions might take $N!$ computations (rather computationally expensive), pair-wise swapping of pins is considered in [10]. Pair-wise swapping of pins results in $O(N^2)$ computations.

The modified netlist is created based on a series of rules. Similarly, to the proximity attack, a list of candidates pins to be swapped is created before the actual swap is applied. Since not every pin can be swapped, a candidate pin to be swapped should:

- be an output pin of the partition where the target pin resides
- not be connected to the partition where the candidate pin resides
- not form a combinational loop

Using the above constraints, a candidate pin is selected.

TABLE 3. Split Manufacturing Defenses.

Work	Year	Threat Model	Category	Defense	Metrics	Defense Overheads Presented
[10]	2013	I	Proximity Perturbation	Pin Swapping	Hamming Distance	-*
[15]	2013	II	Wire Lifting	Wire Lifting	k-Distance	Power, Area, Delay and Wire-Length
[12]	2014	I	Layout Obfuscation	Layout Obfuscation for SRAMs and Analog IPs	-	Performance, Power and Area
[29]	2014	I	Layout Obfuscation	Obfuscation Techniques	Neighbor Connectedness and Entropy	Performance and Area
[30]	2015	I	Layout Obfuscation	Automatic Obfuscation Cell Layout	Neighbor Connectedness and Entropy	Performance, Power and Area
[31]	2015	I	Layout Obfuscation	Obfuscated Built-in Self-Authentication	Obfuscation Connection	Number of Nets
[19]	2016	I	Wire Lifting	Artificial Blockage Insertion	Number of Pins	-*
[32]	2016	I	Wire Lifting	Net Partition, Cell Hidden and Pin Shaken	-	-*
[16]	2017	I	Proximity Perturbation	Routing Perturbation	Hamming Distance	Performance and Wire-Length
[33]	2017	I	Wire Lifting	Secure Routing Perturbation for Manufacturability	Hamming Distance	Performance and Wire-Length
[34]	2017	I	Proximity Perturbation	placement-centric Techniques	CCR	Performance, Power and Area
[35]	2017	II	Proximity Perturbation	Gate Swapping and Wire Lifting	Effective Mapped Set Ratio and Average Mapped Set Pruning Ratio	Wire-Length
[36]	2018	I	Wire Lifting	Concerted Wire Lifting	Hamming Distance	Performance, Power and Area
[20]	2018	I	Proximity Perturbation	Secure Driven Placement Perturbation	Hamming Distance	Power and Wire-Length
[37]	2018	I	Proximity Perturbation	placement and routing perturbation	Hamming Distance	Performance, Power and Area
[38]	2019	I	Layout Obfuscation	Isomorphic replacement for Cell Obfuscation	Isomorphic Entropy	-*
[39]	2019	II	Layout Obfuscation	Dummy Cell and Wire Insertion	k-security	Area and Wire-Length

* Authors do not present any discussion regarding overhead.

The target pin also needs to be chosen carefully. In [10], IC testing principles [40] and hints from the original proximity attack are used to choose the target pin. The swapping procedure is described in Algorithm 2, where $TestMetric$ is a metric based on IC testing principles, such as stuck-at fault models which are still utilized in Test today. More details can be obtained from [10]. The proposed defense technique is validated using ISCAS'85 circuits and the original proximity attack. For the smallest circuit, c17, it takes only one swap to achieve a Hamming Distance of 50%. For the largest studied circuit, c7552, it takes 49 swaps. These results are summarized in Table 4.

As demonstrated in [10], rearranging the partition pins can thwart proximity attacks. However, according to Chen *et al.* [35], pin swapping at partition level has limited efficacy. They demonstrated that an attacker holding the FEOL layout as well as the nestlist can insert hardware trojans even when the defense approach of [10] is applied. It must be highlighted that [35] assumes threat model II, which we have previously argued that has the potential to nullify the vast majority of defenses towards split circuits. Thus, they proposed a defense to counter the threat from hardware trojans. Their defense incorporates the global wire-length information, with the goal to hide the gates from their candidate locations, and as result decreasing the effective mapped set ratio (EMSR). The EMSR metric is an attempt to quantify the ratio of real gates location of a given mapping during a simulated annealing-based attack. This defense consists of two steps, first a greedy gate swapping defense [20], and second, a measurement of the security elevation in terms of EMSR. The

technique is evaluated using ISCAS'85 benchmarks circuits and the EMSR metric to quantify the defense effectiveness. The results are shown in Table 4.

Following the same principle of increasing the Hamming Distance, Wang *et al.* [16] proposed a routing perturbation based defense. The optimum Hamming distance is sought to be achieved by layer elevation, routing detours, and wire decoys, while test principles are used to drive the perturbation choices. Layer elevation is essentially a wire lifting technique: without changing the choice of split metal layer, wires are forced to route using higher metal layers, thus being lifted from the FEOL to the BEOL. Intentional routing detours are a way to increase the distance between disconnected pins of the FEOL. If done properly, disconnected pins will not be the closest to each other, deceiving the proximity attack. In some cases, routing detours will increase the distance between disconnected pins, however, they still remain the closest to each other. In this scenario, wire decoys can be drawn near disconnected pins, in such a way that decoys are now the closest and will instead be picked as the ideal candidate pin.

The perturbations proposed in [16] can incur heavy overheads, and, for this reason, wires to be perturbed are chosen by utilizing IC test principles. In [16], fault observability, as defined in SCOAP [41], is used as a surrogate metric for this task. The technique is evaluated using ISCAS'85 and ITC'99 benchmark circuits. For all studied circuits, the Hamming distance increased by an average of 27% at a cost of only 2.9% wire length overhead (WLO), on average. The results for the largest and smallest studied circuits are shown in Table 4.

Sengupta *et al.* [34] take a different direction from other works. They utilized an information-theoretic metric to increase the resilience of a layout against proximity attacks. As demonstrated in [34], mutual information (MI) can be used to quantify the amount of information revealed by the connectivity distance between cells. Mutual information is calculated by taking into account the cells connectivity D, if they are connected or not, and their Manhattan distance X, described by Eq. 2, where $H[\cdot]$ is the entropy. The Manhattan distance of two cells is defined as the sum of horizontal and vertical distances between them. Entropy is a measure of disorder of a system. Therefore, in this work, entropy is utilized as a measure of disorder in the FEOL layer. The distribution of the variable X for a given layout is determined pair-wise for all gates, allowing a straightforward computation of $I(X;D)$. Thus, layouts with the lowest mutual information, i.e., the correlation between cell connectivity and their distance is low, are more resilient against proximity attacks.

$$MI = I(X;D) = H[X] - H[X/D] \tag{2}$$

In order to minimize the information "leaked" from mutual information, [34] applies cell placement randomization and three other techniques: g-color, g-type1, and g-type2. Randomizing the cell placement can achieve the desired low mutual information; however, the PPA overhead incurred is excessive. Minimizing mutual information without excessive PPA overhead can be achieved by the other techniques. From a graph representation of the circuit, graph coloring can be used to hide connectivity information, where gates of the same color must not be connected. Thus, the resulting colored netlist is then partitioned by clustering all cells of same color together. During cell placement, the cells with the same color will be confined within their respective clusters. According to [34], these constraints naturally mitigate the information leakage to a great extent. The g-color technique utilizes only the graph coloring as described above. The other two, g-type1 and g-type2, consider the type of the gate when creating clusters. The g-type1 approach clusters gates only by their functionality, while g-type2 utilizes functionality and the number of inputs for clustering. The authors assessed their techniques utilizing ISCAS'85 and MNCN benchmark suites. Results for the smallest and largest circuits are shown in Table 4.

Similar to the pin swapping technique proposed in [10], Wang *et al.* [20] proposed a placement-based defense with the same objective of deceiving a proximity attack by perturbing proximity information. Differently from pin swapping, their placement-based defense considers the incurred wire-length overhead as a metric. This technique is based

Algorithm 2: Fault analysis-based swapping of pins to thwart proximity attack (adapted from [10]).

Input: Partitions
Output: List of target and swapping pins
$ListofTargetPins = \emptyset$;
$ListofSwappingPins = \emptyset$;
$ListofUntouchedPins$ = All partition pins and I/O ports;
while *Untouched output partitions pins or input ports exist* **do**
  for $UntouchedPin$ **do**
    $SwappingPins$ = BuildSwappingPinsList($UntouchedPin$); **for** $SwappingPin \in SwappingPins$ **do**
      Compute $TestMetric(UntouchedPin, SwappingPin)$;
  Find the $TargetPin$ and $SwappingPin$ with the Highest $TestMetric$ from its SwappingPins;
  $ListofTargetPins+ = TargetPins$;
  $ListofSwappingPins+ = SwappingPins$;
  $ListofUntouchedPins- = TargetPins$;
  $LisofUntouchedPins- = SwappingPin$;
  Swap TargetPin and SwappingPin;
  Update netlist;
Return: ListofTargetPins and ListofswappingPins;
BuildSwappingPinList($TargetPin$);
Input: $TargetPinP_{x,i,out}$
Output: $SwappingPins$ for $TagetPin$
for $Pin_J \in SwappingPins$ **do**
  if $CombinationalLoop(TargetPin, Pin_J)$ **then**
    $SwappingPins- = Pin_J$;
Return: $SwappingPins$;

TABLE 4. Results for Defense Techniques based on Proximity Perturbation.

Work	Attack Type	Benchmark	Defense Technique	Defense Metric	Defense Overhead	Split Layer	Result without Defense	Result with Defense
[10]	Proximity	c17	-	Hamming Distance	1 Swap for 50% HD	-*	100% CCR	78% CCR
[10]	Proximity	c7552	-	Hamming Distance	49 Swaps for 50% HD	-*	94% CCR	91% CCR
[35]	Proximity	c432	Modifed Greedy Gate Swapping	EMSR	75% of WLO	-*	90% EMSR	25% EMSR
[35]	Proximity	c432	Modifed Greedy Gate Swapping	EMSR	300% of WLO	-*	78% EMSR	10% EMSR
[16]	Proximity	c432	-	Hamming Distance	3.1% WLO for 46.1% HD	-*	92.4% CCR	78.8% CCR
[16]	Proximity	c432	-	Hamming Distance	4.1% WLO for 31.7% HD	-*	62.8% CCR	37.9% CCR
[34]	Proximity	c432	Random	Mutual Information	< 10% PPA	M1	17% CCR	< 1% CCR
[34]	Proximity	c432	g-color	Mutual Information	< 10% PPA	M1	17% CCR	2% CCR
[34]	Proximity	c432	g-type1	Mutual Information	< 10% PPA	M1	17% CCR	6% CCR
[34]	Proximity	c432	g-type2	Mutual Information	< 10% PPA	M1	17% CCR	4.5% CCR
[34]	Proximity	c7552	Random	Mutual Information	< 10% PPA	M1	13% CCR	< 1% CCR
[34]	Proximity	c7552	g-color	Mutual Information	< 10% PPA	M1	13% CCR	2% CCR
[34]	Proximity	c7552	g-type1	Mutual Information	< 10% PPA	M1	13% CCR	4% CCR
[34]	Proximity	c7552	g-type2	Mutual Information	< 10% PPA	M1	13% CCR	3% CCR
[20]	SAT	c432	BEOL+Physical	Perturbation	4.5% WLO	-*	58% CCR	56% CCR
[20]	SAT	c432	Logic+Physical	Perturbation	5.57% WLO	-*	58% CCR	58% CCR
[20]	SAT	c432	Logic+Logic	WLD	1.68% WLO	-*	58% CCR	52% CCR
[20]	SAT	b18	BEOL+Physical	Perturbation	8.06% WLO	-*	15% CCR	14% CCR
[20]	SAT	b18	Logic+Physical	Perturbation	1.70% WLO	-*	15% CCR	17% CCR**
[20]	SAT	b18	Logic+Logic	WLD	0.61% WLO	-*	15% CCR	16% CCR**
[37]	Proximity	c432	Netlist Randomization	Hamming Distance	< 10% PPA overall	-*	92.4% CCR	0% CCR
[37]	Proximity	c7552	Netlist Randomization	Hamming Distance	< 10% PPA overall	-*	94.4% CCR	0% CCR

* Split layer not specified by the authors.
** These results are counter-intuitive, the applied defense degrades the metric.

on changing gate locations such that the proximity hint is no longer effective. Their algorithm consists of two phases, one to select which gates to be perturbed and a second phase where the selected gates are (re)placed. Gate selection is done by extracting a set of trees using two techniques, BEOL-driven and logic-ware extraction. The first approach selects all gate trees that contain any metal wires in the FEOL, i.e., connections that are not hidden from the attacker. The second approach considers the wire-length impact and the gate tree impact on the overall security. After extracting the set of trees, the placement perturbation is done in one of two ways: physical-driven or logic-driven. For each extracted tree, the physical-driven perturbation changes the location of gates using a Pareto optimization approach. Also, each solution is evaluated by its wire-length overhead and a perturbation metric that discerns the placement difference from the original layout. According to [20], geometric-based difference alone may be insufficient to enhance the split circuit security. Thus, a logic-driven perturbation is performed with a weighted logical difference (WLD) metric, which encourages perturbation solutions with large logical difference from its neighbors. The authors assessed their techniques combining the gate selection and perturbation as BEOL+Physical, Logic+Physical and Logic+Logic, using ISCAS'85 and ITC'99 circuit benchmarks. Results for the smallest and largest circuits considered are shown in Table 4.

A considerably different approach is proposed by Patnaik *et al.* [37], whereas netlist modifications are promoted (instead of placement/routing modifications during physical synthesis). The goal is to modify the netlist of a design in order to insert (partial) randomization. According to [37], this approach helps to retain the misleading modifications throughout any regular design flow, thereby obtaining more resilient FEOL layouts where the netlist changes are later "corrected" in the BEOL. This methodology is implemented as an extension to commercial EDA tools with custom in-house scripts. The process goes as follows: first, the netlist is randomized. Second, the modified netlist is place and routed. Lastly, the true functionality is restored by re-routing in the BEOL. For the netlist randomization, pairs of drivers and their sinks are randomly selected and swapped. This is done in such way to avoid combinational loops that may be introduced by swapping. The modified netlist then is place and routed, utilizing a 'do not touch'[5] setting for the swapped drivers/sinks to avoid logic restructuring/removal of the related nets. Finally, the true connectivity is restored in the BEOL with the help of correction cells [37] that resemble switch boxes. The technique is evaluated using ISCAS'85

[5]This terminology is used in IC design to mean that a specific cell or family of cells should not be optimized, i.e., not to be touched.

circuits, and the results for the largest and smallest circuit are shown in Table 4.

B. WIRE LIFTING

Hiding routing information from untrusted foundries is the main objective of the Split Manufacturing technique. Since attacks mainly rely on hints left by EDA tools to recover the missing BEOL connections, the amount of hidden information is related to the circuit performance – splitting the circuits at low metal layers increases the security level. Following the same idea, wire lifting proposes 'lifting' wires from the FEOL layer to the BEOL. That is, changing the routing to split metal layers has the potential to increase the security level.

Wire lifting was first presented by Imerson *et al.* [15] where Split Manufacturing is considered as a 3D IC implementation [42]. For the sake of argument, we will continue to refer to this technique as Split Manufacturing, even if the notion of untrusted FEOL vs. trusted BEOL is shifted. This type of 3D implementation consists of two or more independently manufactured ICs, where each IC represents a tier that is vertically integrated on top of each other. Connections between the tiers are done using vertical metal pillars, referred to as through-silicon vias (TSVs). In [15], a 3D implementation consisting of two tiers is used for their experiments. The bottom tier containing the transistors and some routing wires (akin to the FEOL), and the top tier, containing only routing wires (akin to the BEOL). Regarding the manufacturing of these 3D ICs, the bottom tier is built in a high-end untrusted foundry, and the top tier is built in an also untrusted foundry (not necessarily high-end, however).

In [15], threat model II is used, i.e., the adversary is assumed to possess the entire netlist. The problem is formulated as the attacker being the FEOL foundry, which in turn also possesses the so called 'unlifted netlist' extracted from the FEOL layout. By utilizing a graph to represent the circuits as previously described, the attacker seeks a bijective mapping of gates of the unlifted netlist to gates in the complete netlist. According to [15], if the attacker can distinguish any gate between the two netlists, the split circuit does not provide any security. A security notion is provided by the authors, based on existing multiple mapping between gates in the unlifted and complete netlists. Referred to as *k-security*, this metric qualifies that gates across the design are indistinguishable from at least $k-1$ other gates. Thus, a defender wants to lift wires in a way to guarantee the higher $k-security$ possible. Two procedures are proposed to achieve this goal, one utilizing a greedy heuristic targeted at small circuits (due to scalability issues), and another procedure that utilizes partitioning to solve those issues. For their experimental study, they have utilized the ISCAS'85 benchmark suite and a DES crypto circuit with approximated 35000 gates. The results are shown in Table 5, where $k=1$ is the original circuit and $k=48$ is achieved when all the wires are lifted. It is worth to mention that; besides the notion of the security metric, their defense technique was not validated using an actual proximity attack towards the modified netlist.

An artificial routing blockage[6] insertion that promotes wire lifting is proposed by Magaña *et al.* [19]. The goal of this technique is to deceive proximity attacks by wire lifting. As discussed before, the objective of commercial EDA tools is to guarantee the best PPA possible. During the routing stage, lower metals are preferred for signal routing, promoting better PPA. Thus, routing blockages can be inserted at the split layer, forcing signals to be routed above the split layer. The result is an artificial wire lifting done during the routing stage.

Applying this type of procedure must be done considering the design routability and overhead introduced, as well as top level floorplan decisions for the power grid, clock distribution, and resources for busses. Larger designs are generally difficult to be routed – simply reducing the number of routing layers can make the design unroutable. In [19], a procedure is proposed to insert routing blockages ensuring the design routability is kept. After a first routing stage, the design is divided into small rectangular non-overlapping windows. The routing congestion is then analyzed in each window at the split layer for the blockage insertion. If the area has capacity for more routing, a routing blockage is inserted, otherwise the original routing is kept. Utilizing ISPD'11 circuits, the technique is evaluated using the proximity attack proposed by [19], and its effectiveness is measured using two metrics, $E[LS]$ and FOM. The $E[LS]$ metric reports the candidate list size, being an average over different search areas. The FOM metric is a figure of merit obtained from the ratio of candidates list size divided by the search area, when averaged over all the search areas at the split layer. According to [19], a higher value of FOM means it is more challenging for an attack to be mounted because of the density of candidates (over the same search area). The results for the *Superblue 1* circuit are shown in Table 5.

Design for Manufacturability (DFM) has become an extremely important aspect of IC design for many years now. Manufacturing an IC is a sensitive process that involves many critical steps. Hence, a layout is required to be compliant to several rules to ensure its manufacturability. A layout is said to be manufacturable if there are no DRC violations. However, for a design to also achieve high yield, the layout must also pass strict DFM checks. The most common checks are related to wire and via density over predetermined region sizes. Until now, defense techniques discussed were mainly concerned about security and PPA overheads. Feng *et al.* [33] argued that previous works have largely neglected manufacturability concerns. Therefore, they proposed two wire-lifting techniques that address two important DFM-related techniques: Chemical Mechanical Planarization (CMP) and Self-Aligned Double Patterning (SADP) [43]. The first technique, CMP-friendly routing defense is divided into layer elevation,

[6]This terminology is used in IC design to mean that a specific area should be avoided by the EDA tool for a specific task. A blockage can be for placement and/or for routing.

TABLE 5. Results for Defense Techniques based on Wire Lifting.

Work	Attack Type	Benchmark	Defense Technique	Defense Metric	Defense Overhead	Split Layer	Result without Defense	Result with Defense
[15]	SAT	c432	Wire Lifting	*k-security*	477% of WLO	-*	k=1	k=48
[19]	Proximity	Superblue 1	Routing Blockage Insertion	$E[LS]$	Not Presented	M4	1.51	1.77
[19]	Proximity	Superblue 1	Routing Blockage Insertion	FOM	Not Presented	M4	1222.8	1433
[36]	Proximity	c432	Concerted Lifting	Hamming Distance	7.7% of Area	Average**	23.4	45.9
[36]	Proximity	c432	Concerted Lifting	CCR	13.2% of Power	Average**	92.4	0
[36]	Proximity	c7552	Concerted Lifting	Hamming Distance	16.7% of Area	Average**	1.6	25.7
[36]	Proximity	c7552	Concerted Lifting	CCR	9.3% of Power	Average**	97.8	0
[33]	Proximity	c2670	CMP-Friendly	Hamming Distance	3.4% of WLO	-*	14.5%	20.4%
[33]	Proximity	c2670	CMP-Friendly	CCR(%)	3.4% of WLO	-*	48.1%	33.4%
[33]	Proximity	b18	CMP-Friendly	Hamming Distance	0.4% of WLO	-*	21.6%	27.6%
[33]	Proximity	b18	CMP-Friendly	CCR(%)	0.4% of WLO	-*	12.1%	10.7%
[33]	Proximity	c2670	SADP-Compliant	Hamming Distance	7.49% of WLO	-*	14.5%	24.4%
[33]	Proximity	c2670	SADP-Compliant	CCR(%)	7.49% of WLO	-*	48.1%	6.4%
[33]	Proximity	b18	SADP-Compliant	Hamming Distance	4.64% of WLO	-*	21.6%	29.6%
[33]	Proximity	b18	SADP-Compliant	CCR(%)	4.64% of WLO	-*	12.1%	2.7%
[32]	Proximity	s526	Net Partitioning	CCR(%)	Not Presented	-*	40%***	0%***
[32]	Proximity	s526	Net Partitioning & Cell Hiding	CCR(%)	Not Presented	-*	40%***	0%***
[32]	Proximity	s526	Net Partitioning & Cell Hiding & Pin Shaking	CCR(%)	Not Presented	-*	40%***	0%***
[32]	Proximity	s9234.1	Net Partitioning	CCR(%)	Not Presented	-*	30%***	4%***
[32]	Proximity	s9234.1	Net Partitioning & Cell Hiding	CCR(%)	Not Presented	-*	30%***	1.5%***
[32]	Proximity	s9234.1	Net Partitioning & Cell Hiding & Pin Shaking	CCR(%)	Not Presented	-*	30%***	1.5%***

* Split layer not specified by the authors.
** Results are given as an average between M3, M4, and M5.
*** These results cannot be directly compared with previous ones as the transistor technology is vastly different.

wire selection, and re-routing. Layer elevation selects wires for lifting according to following principles [33]:

- The wire has a significant logic difference from its neighboring wires. As such, an incorrect connection in attacking this wire may lead to more signal differences.
- The wire has large observability such that an erroneous guess by the adversary can easily affect the circuit primary output signals.
- The wire segment is originally at a wire-dense region. The wire density of this region would be reduced by the layer elevation and makes the corresponding FEOL layer have more uniform wire density.
- The BEOL region where the wire is elevated to has low wire density so that the density of the corresponding BEOL layer is more uniform.

Principles 1 and 2 have the goal to increase security in the same way as described in [16]. After the wire lifting step, a set of wires is selected for re-routing. The selection has two purposes, CMP-friendliness and security improvement. For CMP-friendliness, wires located in dense regions are selected to be re-rerouted in sparse areas. For the security improvement, decoys are inserted if the routing detour passes through a sparse area. A suspicious attacker may realize that the detour is a defense measure. After selecting the set of wires to be re-routed, wires are re-routed one at a time. According to [33], their routing approach considers wire density, while the routing perturbation proposed by [16] can be solely focused on security, and may not be CMP-friendly. Utilizing a graph representation, their re-routing method is based on the Dijkstra's shortest path algorithm [44] where the density of wires is used as a metric.

With a few exceptions, the SADP-compliant routing defense follows the same approach as described above. During wire lifting, the density is not considered. Wire re-routing is actually wire extension of FEOL wires as in [45]. This wire extension of FEOL wires inevitably leads to re-routing of connected BEOL wires. According to [33], solving SADP violations by wire extension can also increase security, as its increase the distance between vias. The wire extension for simultaneous SADP-compliance and security is realized using Integer Linear Programming. In their experiments, ISCAS'85 and ISPD'11 are used to evaluate their techniques. Each technique, CMP-friendly and SADP-compliant routing, is evaluated separately. The results for the smallest and largest circuits are shown in Table 5.

Wire lifting approaches, in general, are not cost-free. As shown in the discussed results, wire-lifting based defenses introduce a considerable PPA overhead. An approach to establish a cost-security trade-off is proposed by Paitinak *et al.* [36], i.e., PPA margins for a given security budget. In [36],

a concerted wire-lifting method is proposed. The authors claim to enable higher degrees of security while being cost-effective. For their method, custom elevating cells are used for executing the wire-lifting. Elevating cells connect gates or short wires directly to the first layer above the split layer. Their wire-lifting method utilizes three strategies: lifting high-fanout nets, controlling the distance for open pin pairs, and obfuscation of short nets. High-fanout nets are chosen to be lifted for two reasons: (a) a wrong connection made by the attacker propagates the error to multiple locations, and, (b) introduces multiple open pin pairs. As the attack to overcome is the proximity one, controlling the distance between open pin pairs is necessary, which is achieved at will simply by controlling the placement of the elevating cells. According to [36], short nets may be easy for an attacker to identify and localize (from assessing driving strengths). Short wires are obfuscated by inserting an elevating cell with two pins close to each other, one being the true connection and the other a dummy connection. Finally, wires are lifted according to those strategies until a given PPA budget is reached. For their experimental study, ISCAS'85 and ISPD'11 circuits are utilized. However, results for attacks are presented only for ISCAS'85 circuits. For ISPD'11, only the PPA impact result introduced by their technique is presented. Once again, we present the results for the smallest and largest of the studied circuits in Table 5.

While the majority of studies reported in our survey make use of conventional transistors (bulk CMOS technologies with either planar or FinFET transistors), Yang *et al.* [32] proposed a design methodology to secure Split Manufacturing for Vertical Slit Field Effect Transistor (VeSFET)-based integrated circuits. VeSFET is a twin-gate device with a horizontal channel and four metal pillars implementing vertical terminals [46]. While a detailed explanation on VeSFETs is beyond the scope of this work, we do highlight the differences between VesFETs and conventional transistors. In contrast with conventional transistors, a VeSFET can be accessed by both top and bottom pillars, allowing two-side routing and offering a friendly monolithic 3D integration [46]–[48]. While we have so far considered ICs that have two distinct layers, the FEOL and BEOL, a VeSFET-based IC has tiers of the layer containing the transistors. Connections between tiers can be made directly, same as TSV by the pillars, or by a layer containing connections between tiers. A 2D VeSFET design contains only one tier and both top and bottom connections, whereas a 3D design contains two or more tiers. In summary, the notion of tier is pushed down to the transistor level in this device topology, thus making it an interesting platform for Split Manufacturing.

The method proposed by [32] assumes that both foundries are untrusted and have the same capability (i.e., same technology). For 2D designs, the first foundry manufactures the tier with the top connections, comprising most of the connections. Then, the rest of the bottom connections, comprising of the critical minority connections, are completed by the second foundry. For 3D IC designs, they proposed special types of standard cells, referred as cell A and B. Cell A has two tiers that are visible and manufactured by the first foundry, as well as inter-tier connections. Cell B has only the top tier visible and manufactured by the first foundry, the low tier is completed by the second foundry, without inter-tier connections. Thus, transistors can be hidden from the first foundry as a security feature. Vulnerabilities claimed by [32] for both 2D and 3D methods are described in Table 6. Practices of reverse engineering and IC overbuilding are claimed to be impossible because the first foundry controls the number of wafers available to the second foundry.

Increasing the security of both 2D and 3D VesFET designs is achieved by net partitioning, and exclusively for 3D designs, by transistor hiding and pin shacking. Net partitioning is performed similarly to the wire lifting techniques described above, where nets are chosen to be routed in the bottom connection layer, thus, hiding those from the first foundry. Their selection method is done by selecting nets from sequential logic. First, all the high-fanout nets are selected to be partitioned. Next, the remaining nets are selected by a search area, where two approaches are used, distance-first search and high-fanout first search. In distance-first method, a pin in a predefined search window connecting to an un-partitioned net is selected when it has the minimum distance to the currently processed pin pair. The FO-first search method selects the pin connecting to a net having the highest FO in the searching window. Transistor hiding in 3D designs is done by utilizing cells similar to the cell B. Cells connected only by partitioned nets are candidates for hiding. After selecting the candidates, availability of unused transistors that are accessible to the second foundry in the lower tiers of the nearby cells is checked. If the available transistor count is sufficient, then the cell is hidden. The empty space created could provide clues for the first foundry about the security technique. Pin shaking is then applied to obfuscate the empty spaces. Some nearby cells are moved to this area to obfuscate the layout for any distance-based proximity attackers. In [32], 10 MCNC LGSynth'91 benchmark circuits are used to evaluate the effectiveness of their methodology. The best and worst results are shown in Table 5. It is worth to mention that, even though the VesFET implementation mimics the layered structure of Split Manufacturing, the results cannot be compared side by side in a fair manner.

TABLE 6. Vulnerabilities of Split Manufactured VesFET-based Designs Described by [32].

Threats	1st Foundry	2nd Foundry
Design Reconstruction	2D IC: Very Difficult 3D IC: Impossible	Impossible due to a very limited information
Trojan Insertion	Possible, but will be detected	No control of devices
Reverse Engineering	Meaningless	Impossible
IC Overbuilding	Meaningless	Impossible

C. LAYOUT OBFUSCATION

The main goal of Split Manufacturing – to hide sensitive information from untrusted foundries – is compromised once we start to consider more regular structures such as memory. Even without knowing where all the routing goes to, an attacker can easily identify regular structures just by looking at the FEOL layout, possibly leading to easier attacks. Mitigating attacks towards regular structures could be done by obfuscating those structures in such a way that they become indistinguishable. In this section, we discuss works that propose layout obfuscation techniques to be used in a Split Manufacturing context.

During the development of a modern IC, third-party IPs are sought to close a technological gap or to minimize time-to-market. IPs are typically categorized as soft and hard IPs: soft IPs typically come in code form, giving the customer flexibility to modify the IP such that it meets a given specification during synthesis. Therefore, soft IPs do not present a direct challenge for a Split Manufacturing design flow. Perhaps, and on a very specific scenario, a given IP can facilitate a proximity attack because it promotes certain library cells over others (i.e., it leads to a biased composition).

On the other hand, hard IPs are completely designed by the vendor and are technology dependent. In some instances, the vendor only provides an abstract of the IP; the customer then has to rely on the foundry to replace the abstract by the actual layout. Thus, splitting a hard IP is not trivial. Additional information is needed to be provided by the vendor, which is not guaranteed to be provided, making the IP completely incompatible with Split Manufacturing. Even when the customer holds the entirety of the IP layout, differences between the FEOL foundry and the BEOL foundry could make the IP no longer compliant and therefore virtually useless. Furthermore, defense techniques cannot be applied due to the lack of information or lack of feasibility. Hard IPs, such as embedded memories and specialized analog circuits, have been heavily optimized for maximum compactness, performance and yield. In today's IP market, there is still little concern with security in general, so it is not conceivable that any vendors will start to offer split IP any time soon.

The security of hard IPs in a Split Manufacturing context was first analyzed by Vaidyanathan *et al.* [12]. A recognition attack flow was proposed for this purpose. An attacker holding the FEOL layer starts his attack by isolating a target embedded memory or analog hard IP. Since the targeted hard IP has a high probability of being constructed by compilation of leaf-cells, layout pattern recognition software [49] can be used for trivial leaf-cell identification. After recognizing all the leaf-cells, the attacker attempts to infer the missing BEOL connections. Using proximity hints together with the knowledge about the regularized structure, the connections have a high likelihood to be guessed correctly. Demonstrated in [12], embedded memories, such as SRAM, are susceptible to the proposed recognition attack. Defending against recognition attacks can be achieved by means of obfuscation. According to [12], SRAM IPs can be obfuscated by the following methods:

- Randomization of periphery cells, thus avoiding predictable connections.
- Minimization of regularized topologies used for peripheral circuits such as pre-decoders, word line decoders, sense amplifiers, etc.
- Adding non-standard application-specific functions to improve obfuscation and performance.

A synthesis framework is proposed by [12] to obfuscate SRAM IPs. Referred as application-specific SRAM, the methodology synthesizes SRAMs using augmented bitcell arrays and standard cell logic IP (instead of using leaf-cells). Such synthesis, when compared with conventional SRAM compilation, accomplishes all the three obfuscation goals described above while still providing similar performance.

Analog hard IPs are also vulnerable to recognition attacks. In contrast with embedded memories (that are often compiled), analog hard IPs are mostly hand designed to cater for a challenging specification or interface. Even when such degree of customization is employed, the majority of the design is done utilizing leaf-cells (e.g., current mirrors, matched arrays, etc.). Thus, disclosing important information that could be used as leverage for recognition attacks. In [12], two methods are proposed to defend analog hard IPs against such attacks:

- Obfuscation of analog leaf-cells.
- Use of diverse topologies and architectures that enable obfuscation and efficiency.

Next, let us discuss the techniques utilized in order to achieve the goals listed above. First, adding camouflaging dummy transistors in empty spaces can turn leaf-cells indistinguishable. Second, regularizing transistor widths, which allows transistor with different channel lengths to abut each other, thereby obscuring boundaries across different sized transistors. Third, utilizing the same idea behind wire-lifting, routing blockages can be inserted between transistors below the split layer. Such routing scheme would make it difficult to infer the missing BEOL connections, virtually in the same way as it does for a standard-cell based design.

To demonstrate the feasibility and efficacy of their proposed approaches, the authors of [12] designed and fabricated test chips in 130nm technology. For comparison, the same designs were Split Manufactured and conventionally manufactured. Split Manufacturing used Global Foundries Singapore as the untrusted foundry and IBM Burlington as the trusted foundry. Conventional manufacturing was entirely done in Global Foundries Singapore. The first reported design is a smart SRAM that targets an imaging application. Two implementations of a parallel 2x2 access 1Kb SRAM were demonstrated. For conventional manufacturing, the SRAMs were traditionally implemented, and for Split Manufacturing, the SRAMs were implemented using their smart synthesis approach. For their measurements, 10 chips were used to demonstrate the feasibility regarding PPA. Area reported for the split manufactured samples was 75% of the

conventional approach, and, while the power consumption was 88%. Performance was the same between conventional and split manufactured, i.e., both could work with the same clock frequency. The PPA advantage of Split Manufacturing reported in [12] is not from the manufacturing itself. This advantage is from their smart memory synthesis approach, that was not applied on the conventional manufacturing samples.

The second demonstrated design is a DAC with statistical element selection. The test chip contains a high resolution 15-bit current steering DAC. Only a description of the results is presented, where the authors claim there are tiny measurements differences between the performance of the conventional and the split manufactured, emphasizing that the differences are within measurement noise.

An attacker trying to reverse engineer a split IC will try to recover the maximum number of connections as possible, while minimizing the Time To Evaluate (TTE), i.e., the amount of time needed to reverse engineering the IC. For Jagasivamani *et al.* [29], the goal of a designer seeking to secure his design is to create an IC with a high TTE while being cost-effective regarding design effort and PPA overheads. If the TTE is high enough, an adversary would be discouraged from reverse engineering the IC. To achieve this goal, [29] proposed obfuscation methods that do not require any modifications to standard cells nor the implementation of any specialized cell.

Four techniques are proposed by [29] for layout obfuscation, (1) limited standard-cell library, (2) smart-dummy cell insertion, (3) isomorphic cells and (4) non-optimal cell placement. Along with the techniques, a set of metrics is presented to help assess the obfuscation level of a design. *Neighbor connectedness*, a measure of how interconnected cells are to their respective neighbors, i.e., how much proximity information is exposed to the attacker. For a specific cell, this metric is computed as how many connections that cell has for a given radius around it. *Standard-cell composition bias*, a metric that addresses the effort required for composition analysis of a design. The bias signature could leave information of the function of the cell. Thus, this metric measure how skewed a design is according to a specific bias cell. In [29], they utilized three types of bias cells for this analysis: XOR-type, flip-flop type, and adder type of cells. *Cell-level obfuscation*, a metric that measures the percentage of standard-cells that have been obfuscated. *Entropy*, which is similar to the concept of mutual information previously discussed.

Technique (1) aims to achieve obfuscation by reducing the use of complex cells and instead favor only simple cells to compose the design (i.e., to prefer single stage cells over complex multi-stage cells). Removing specialized complex cells could obfuscate functional signatures due to the larger granularity that is employed to construct the cell. However, since the functionality of complex gates will have to be reconstructed through basic cells, a heavy PPA overhead is likely to occur when applying this technique. Technique (2) aims to obfuscate composition analysis by adding dummy cells in such a way that a neutral bias composition is achieved. Dummy cells are inserted as spare cells [7], focusing solely on obfuscating the composition analysis. Technique (3) obfuscates the layout by regularizing the shapes of the cells in a library. All layouts of logic standard cells are made FEOL-identical such that the overall circuit layout appears to be a sea of gates. The functionality of the cells is defined later by the BEOL connections. Thus, the true functionally of the cell is hidden at the BEOL, making cell-level identification harder. Technique (4) employs the same strategy from placement perturbation discussed before.

For their experimental study, the authors of [29] made use of a multiplier block with a high number of adder cells and a crypto-like circuit. Experiments were separated into limited library and smart dummy insertion. Results are shown as a percentage relative to the baseline circuit, i.e., without any protection approach applied. Neighbor connectedness (%) for a radius $\leq 25nm$ decreased substantially for both test cases and circuits (for more information see [29]). Overhead results are shown in Table 7, where the figures presented are normalized with respect to the baseline circuit.

TABLE 7. Impact on Performance from the Defense Approaches of [29].

Benchmark	Metric	Limited Library	Smart Dummy
mult24	Area	94.9%	72.6%
	Timing Slack	-64.8%	3.4%
a5/1	Area	69.8%	69.4%
	Timing Slack	-27.2%	-1.2%

Utilizing exactly the same concepts and metrics described in [29], Otero *et al.* [30] proposed a "trusted split-foundry toolflow" based on cellTK [50]. The concept of the cellTK-based flow is to have on-demand cell generation from a transistor-level netlist. This is heavily in contrast with a traditional ASIC flow that relies on a predefined (and thoroughly characterized) cell library. Leveraging cellTK, [30] proposed an extension referred as split-cellTK. This extension can generate multiple physical layouts for each unique cell without modifying the circuit topology, which is then used to implement obfuscation strategies. Two strategies are proposed, referred as Uniform and Random. The Uniform strategy tries to standardize the size and spacing between cells by inserting dummy transistors to equalize the number of nMOS and pMOS devices and, after the cell placement, dummy cells are inserted in empty gaps. A Random strategy is also proposed in order to reduce the overheads introduced by the Uniform strategy. Instead of deliberately standardizing the size and spacing between cells, a specific number of empty spaces is chosen for these tasks. A more in-depth explanation about their strategies is beyond the scope of this work because they are closely related to cellTK itself.

[7] Spare cells are extra logic usually inserted during physical synthesis. These cells are used when an engineering change order (ECO) is required, such that small tweaks to the circuit logic can be performed with minimal changes to placement and routing.

However, their goals and evaluations are the same as in [29]. For their experimental study, they utilized the island-style asynchronous FPGA developed in [51]. A test chip was Split Manufactured in 65nm and the design was synthesized with a cellTK-based approach, i.e., without any defense strategy. Their defense strategies were evaluated only by simulation. The performance results for the baseline, Uniform, and Random strategies, when applied to obfuscate an adder, are shown in Table 8. Trustworthiness results are given in terms of neighbor connectedness and, for all implementations discussed, neighbor connectedness results were significantly smaller than the results reported in [29].

TABLE 8. Performance Results Reported by [30] to Obfuscate an Adder Circuit.

Technique	Area (μm^2)	Power (mW)	Energy (pJ)	Perf. (MHz)
Baseline	462	0.146	0.257	568
Uniform	717	0.149	0.307	486
Random	760	0.164	0.303	542

In the context of obfuscation, but also generally for Split Manufacturing, a higher level of security is achieved when the chosen layer to perform the split is the lowest possible. Xiao *et al.* [31] pointed out that splitting at lower metal layers could increase the cost to manufacture the IC; it is argued that the FEOL-BEOL integration process must be more 'precise' for correct alignment. Thus, a closer technology match between the trusted and untrusted foundries is required. As we previously argued, if the goal is to make use of the best silicon available from an untrusted foundry, the implication is that the trusted foundry cannot provide a legacy technology, but perhaps a mature yet still relevant technology is sufficient. In [31], a methodology for obfuscating the layout is proposed for split at M3 or higher, meanwhile keeping the cost as low as possible and at the same time providing a high level of security. Their strategy is similar to the insertion of dummy cells; however, functional cells are inserted instead. Referred as obfuscated built-in self-authentication (OBISA) cells, the inserted functional cells are connected together to form a circuit. As the circuit is connected to the original circuit it is trying to protect, they claim this fact makes it extremely difficult for an attacker to separate the OBISA design from the original design. The idea behind OBISA is to obfuscate the layout by hindering neighbor connectedness analysis and standard-cell composition bias analysis while also perturbing the proximity between gates. As illustrated in Figure 11, two additional functional cells, O1 and O2, were placed between gates G2 and G3, and, G3 and G4, respectively. The insertion of these additional functional cells could deceive proximity attacks, assuming that the EDA tool would place the gates between OBISA cells farther apart than in the original circuit.

The proposed OBISA circuitry has two operating modes: functional and authentication. During functional mode, the OBISA circuitry stays idle and incoming signals and clock are gated/blocked. Thus, the original circuit is not affected

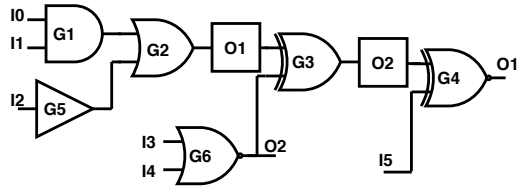

FIGURE 11. Circuit representation with OBISA cells (square cells) inserted (adapted from [31]).

by OBISA operating as it should. As the name suggests, when in authentication mode, OBISA is used to verify the trustworthiness of the manufactured IC (in the field). The specifics of the authentication are beyond the scope of this work and will not be discussed. The insertion of OBISA cells follows a similar strategy of dummy cell insertion as discussed in [29]. The connections of the inserted cells are done in a way to promote the testability of the OBISA circuit and increase the obfuscation strength. Their approaches were evaluated using benchmark circuits from OpenCores. Results for the smallest and largest circuits are shown in Table 9.

TABLE 9. Implementation Results from [31].

Benchmark	Gate count	OBISA cell count	Total nets	Nets $\geq$ M4
DES3	1559	158	1799	127
DES_perf	49517	2090	49951	1343

Another study using look-alike cells is reported by Masoud *et al.* [38] where the goal remains to make the attacker unable to distinguish cells and their inputs/outputs, thus mitigating attacks to some degree. In this study, two types of search algorithms are proposed to replace cells for isomorphic cells. In contrast with [29] where all cells are replaced, in [38], only cells with high impact on the security are replaced. Thus, the overhead introduced by cell replacement can be controlled (i.e., a trade-off is established). The proposed algorithms are based on 'gate normalization', whereby truth tables of cells are analyzed in order to balance the occurrence of 0s and 1s (e.g., XOR and XNOR gates are normalized by definition). An analysis is made by replacing existing gates by XORs and comparing the deviation from the original circuit. If the deviation is larger than a given deviation threshold, the gates are effectively replaced.

A novel layout obfuscation framework is proposed by Li *et al.* [39] which builds on the wire lifting concept of [15]. According to the authors, wire lifting alone is not enough to secure a design. If an attacker can tell the functionality of a specific gate that had its wires lifted, the security is already compromised. To address this problem, a framework that considers dummy cells and wire insertion simultaneously with wire lifting is proposed. As in [15], threat model II was used. The proposed framework makes use of mixed-integer linear programming (MILP) formulation for the FEOL layer

generation and a Lagragian relaxation (LR) algorithm to improve scalability. The generation of the new FEOL layout considers three operations: wire-lifting, dummy cell insertion, and dummy wire insertion. Dummy wire insertion is done only on dummy cells; thus, the original functionality of the circuit is guaranteed to remain and floating pins are avoided. Utilizing a graph representation, they re-formulate the security metric to accommodate dummy cell and dummy wire insertion. Since the original graph isomorphic relationship is lost when new nodes are inserted, a new approach has to be used to formalize the relationship between the original and the new FEOL; this concept is denoted as k-isomorphism [52] and the associated security analysis is denoted as *k-security*. In their experimental study, TrustHub [53] trojan insertion methods are used to select the nodes for protection. They used ISCAS'85 benchmark circuits together with functional units (shifter, alu, and div) from the OpenSPARC T1 processor [54]. Comparison between MILP and LR algorithms are done for several *k-security* levels, and the results are given in terms of area overhead (AO) and wirelength overhead (WLO). The results for a few of the security levels are shown in Table 10.

TABLE 10. Comparison Between MILP and LR Algorithms for the c4232 circuit [39].

Security Level	Algorithm	AO(%)	WLO(%)
15	MILP	18	180
20	MILP	41	220
25	MILP	58	295
15	LR	18	200
20	LR	40	230
25	LR	60	305

V. FUTURE TRENDS AND CHALLENGES

Despite our effort to present the results of the many studied papers in the most fair way possible, it is clear that the hardware security community lacks a *unified benchmark suite* and/or a *common criteria* for assessing results. Often, researchers make use of benchmark suites that are popular in the Test community but have no real applicability in security. For instance, the ISCAS'85 suite has no crypto cores in it, which are the bread and butter of the research in the area. Furthermore, we believe the community would largely benefit from using circuits that better represent IC design practices of this decade where IPs often have millions of gates and ICs have billions of transistors.

While the lack of a common criteria is an issue for the academic community, the lack of an industry-supported path for Split Manufacturing is even more troubling. Today, more than ever, foundries compete for the title of 'best silicon' and rarely engage in cross-foundry cooperation. Efforts of the past, such as the now defunct Common Platform of IBM, Samsung and GF, could have been a catalyzer for the adoption of Split Manufacturing. Without such collaboration, it is hard to foresee a future where the technique will gain traction again. Furthermore, the study of DFM-related implications of the technique is really cumbered by the fact that we cannot measure yield from massively produced Split Manufactured chips.

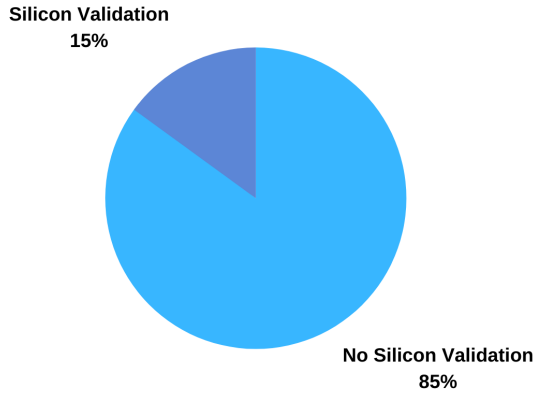

FIGURE 12. Techniques validated in silicon among presented works.

We have discussed in details how many attacks leverage heuristics and hints left behind by the EDA tools. Many of these hints are very logical and can be appreciated, even graphically, as we illustrated in Figure 7. It is entirely possible that machine learning approaches can detect subtle biases in the tools that are not easy to appreciate graphically. There is no consolidated knowledge of what these biases are and to which extent machine learning is effective in detecting them. This avenue of research is certainly interesting and we believe it will be the target of many papers in the near future.

It is also worth discussing the attack models that have been proposed so far. We have previously highlighted how strong Threat model II is, but the fact of the matter is that defining the threat model is a complicated exercise in which we seek to establish what are the capabilities of the attacker. By definition, formalizing the capabilities of an attacker requires understanding his motivations, technical proficiency, and availability of resources. If the attacker is underestimated, useless defense strategies can be devised and assumed to be effective. If the attacker is overestimated, convoluted defense strategies might be employed, leading to unnecessary PPA overheads. This is a challenge for Split Manufacturing and many other techniques that promote obfuscation.

Another topic that has led to no consensus is whether an attacker can make use of a partially recovered netlist. For instance, let us assume a design that instantiates the same block multiple times. If one of the blocks is correctly recovered, perhaps a cursory inspection of the structure will allow the attacker to recover all other instances of the same block. The same line of thinking can be applied to datapaths and some cryptographic structures that are regular in nature. In a sense, an analysis of the functionality of the recovered netlist could be combined with existing attacks for further

improvement of correctly guessed connections.

We note that many of the works studied in this survey have not actually demonstrated their approach in silicon. This fact is summarized in Figure 12. As a community effort, we should strive to validate our approaches in silicon as often as possible. However, as discussed before, finding two foundries willing to diverge from their established practices could be next to impossible. This is likely the main reason that such small percentage of the works herein reported have validated their techniques in silicon.

VI. CONCLUSION

Our findings showed a big disparity on how the Split Manufacturing technique is approached among the surveyed studies. A variety of benchmark suites and metrics were used for evaluation, making direct comparisons between studies very difficult – and, in some cases, impossible. In spite of that, we were able to classify the studies, clearly demonstrating the many interpretations of the technique, its attacks, and defenses. Our belief is that this survey assesses the most significant studies about Split Manufacturing as we focused on papers that appear on highly-regarded venues. Results gathered from the surveyed studies were compiled such that main features, metrics, and performance results are available. Regarding the results themselves, these are presented in such manner to illustrate the present state of the technique. Therefore, this work can be very helpful for future researchers to contextualize their own techniques for augmenting Split Manufacturing.

Overall, the security of Split Manufacturing is still under debate. Some studies conclude that the technique is indeed secure, and others that it is not. However, these conclusions are reached for different scenarios, i.e., using different benchmark circuits and set of metrics. Creating a unified benchmark suite suitable for Split Manufacturing evaluation, along with a unified set of metrics to quantify/qualify its performance, could facilitate the discussion about its security. In addition, increasing the number of demonstrations in silicon could also help with evaluation and adoption issues related to Split Manufacturing.

REFERENCES

[1] European Union Intellectual Property Office (EUIPO), "2019 Status Report On IPR Infringement," [Online]. Available: https://euipo.europa.eu/ohimportal/en/web/observatory/status-reports-on-ip-infringement.

[2] M. Pecht and S. Tiku, "Bogus: Electronic Manufacturing and Consumers Confront a Rising Tide of Counterfeit Electronics," IEEE Spectrum, vol. 43, no. 5, pp. 37–46, 2006.

[3] 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.

[4] 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.

[5] R. Torrance and D. James, "The State-of-the-Art in Semiconductor Reverse Engineering," 2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 333–338, 2011.

[6] P. Subramanyan, N. Tsiskaridze, K. Pasricha, D. Reisman, A. Susnea, and S. Malik, "Reverse Engineering Digital Circuits Using Functional Analysis," pp. 1277–1280, March 2013.

[7] AnySillicon, "Fabless Company Sales By Region 2018," [Online]. Available: https://anysilicon.com/fabless-company-sales-by-region-2018.

[8] Intelligence Advanced Research Projects Activity (IARPA), "Trusted Integrated Circuits Program," [Online]. Available: https://www.iarpa.gov/index.php/research-programs/tic.

[9] K. Vaidyanathan, B. P. Das, E. Sumbul, R. Liu, and L. Pileggi, "Building Trusted ICs Using Split Fabrication," IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 1–6, 2014.

[10] J. Rajendran, O. Sinanoglu, and R. Karri, "Is Split Manufacturing Secure?," in Design, Automation and Test in Europe (DATE), no. Ic, pp. 1259–1264, 2013.

[11] T. Kikkawa and R. Joshi, "Design Technology Co-Optimization for 10 nm and Beyond," in Proceedings of the IEEE 2014 Custom Integrated Circuits Conference, pp. 1–1, Sep. 2014.

[12] K. Vaidyanathan, R. Liu, E. Sumbul, Q. Zhu, F. Franchetti, and L. Pileggi, "Efficient and Secure Intellectual Property (IP) Design with Split Fabrication," in 2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 13–18, 2014.

[13] B. Hill, R. Karmazin, C. T. O. Otero, J. Tse, and R. Manohar, "A Split-Foundry Asynchronous FPGA," in Proceedings of the IEEE 2013 Custom Integrated Circuits Conference, pp. 1–4, Sep. 2013.

[14] T. Usui, K. Tsumura, H. Nasu, Y. Hayashi, G. Minamihaba, H. Toyoda, H. Sawada, S. Ito, H. Miyajima, K. Watanabe, M. Shimada, A. Kojima, Y. Uozumi, and H. Shibata, "High Performance Ultra Low-k (k=2.0/keff=2.4)/Cu Dual-Damascene Interconnect Technology with Self-Formed MnSixOy Barrier Layer for 32 nm-node," in 2006 International Interconnect Technology Conference, pp. 216–218, 2006.

[15] F. Imeson, A. Emtenan, S. Garg, and M. Tripunitara, "Securing computer hardware using 3d integrated circuit (IC) technology and split manufacturing for obfuscation," in 22nd USENIX Security Symposium (USENIX Security 13), pp. 495–510, USENIX Association, Aug. 2013.

[16] Y. Wang, P. Chen, J. Hu, and J. Rajendran, "Routing Perturbation for Enhanced Security in Split Manufacturing," Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 605–610, 2017.

[17] K. Yang, U. Botero, H. Shen, D. Forte, and M. Tehranipoor, "A Split Manufacturing Approach for Unclonable Chipless RFIDs for Pharmaceutical Supply Chain Security," Asian Hardware Oriented Security and Trust Symposium (AsianHOST), vol. 2018-May, pp. 61–66, 2018.

[18] S. N. Pagliarini, M. M. Isgenc, M. G. A. Martins, and L. Pileggi, "Application and Product-Volume-Specific Customization of BEOL Metal Pitch," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 9, pp. 1627–1636, 2018.

[19] J. Magaña, D. Shi, and A. Davoodi, "Are Proximity Attacks a Threat to the Security of Split Manufacturing of Integrated Circuits?," IEEE/ACM International Conference on Computer-Aided Design (ICCAD), vol. 07-10-Nove, no. c, pp. 1–7, 2016.

[20] Y. Wang, P. Chen, J. Hu, G. Li, and J. Rajendran, "The Cat and Mouse in Split Manufacturing," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 26, no. 5, pp. 805–817, 2018.

[21] W. Zeng, B. Zhang, and A. Davoodi, "Analysis of Security of Split Manufacturing Using Machine Learning," IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 27, no. 12, pp. 2767–2780, 2019.

[22] S. Chen and R. Vemuri, "On the Effectiveness of the Satisfiability Attack on Split Manufactured Circuits," in 2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC), pp. 83–88, 2018.

[23] S. Chen and R. Vemuri, "Exploiting Proximity Information in a Satisfiability Based Attack Against Split Manufactured Circuits," Proceedings of the 2019 IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2019, pp. 171–180, 2019.

[24] F. Brglez, D. Bryan, and K. Kozminski, "Combinational Profiles of Sequential Benchmark Circuits," in IEEE International Symposium on Circuits and Systems,, pp. 1929–1934 vol.3, 1989.

[25] N. Viswanathan, C. J. Alpert, C. Sze, Z. Li, G.-J. Nam, and J. A. Roy, "The ISPD-2011 Routability-Driven Placement Contest and Benchmark Suite," in Proceedings of the 2011 International Symposium on Physical Design, ISPD '11, p. 141–146, Association for Computing Machinery, 2011.

[26] T. L. M. R. K. Ahuja and J. B. Orlin, "Network Flows: Theory, Algorithms, and Applications.," Upper Saddle River, NJ, USA: Prentice-Hall, 1993.

[27] F. Corno, M. S. Reorda, and G. Squillero, "Rt-level itc'99 benchmarks and first atpg results," IEEE Design Test of Computers, vol. 17, no. 3, pp. 44–53, 2000.
[28] H. Zhou, R. Jiang, and S. Kong, "CycSAT: SAT-Based Attack on Cyclic Logic Encryptions," in 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 49–56, 2017.
[29] M. Jagasivamani, P. Gadfort, M. Sika, M. Bajura, and M. Fritze, "Split-Fabrication Obfuscation: Metrics and techniques," IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 7–12, 2014.
[30] C. T. O. Otero, J. Tse, R. Karmazin, B. Hill, and R. Manohar, "Automatic Obfuscated Cell Layout for Trusted Split-Foundry Design," IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), pp. 56–61, 2015.
[31] K. Xiao, D. Forte, and M. M. Tehranipoor, "Efficient and secure split manufacturing via obfuscated built-in self-authentication," in 2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), pp. 14–19, 2015.
[32] P. Yang and M. Marek-Sadowska, "Making split-fabrication more secure," in 2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), pp. 1–8, 2016.
[33] L. Feng, Y. Wang, J. Hu, W. K. Mak, and J. Rajendran, "Making Split Fabrication Synergistically Secure and Manufacturable," IEEE/ACM International Conference on Computer-Aided Design (ICCAD), vol. 2017-Novem, pp. 313–320, 2017.
[34] A. Sengupta, S. Patnaik, J. Knechtel, M. Ashraf, S. Garg, and O. Sinanoglu, "Rethinking Split Manufacturing: An Information-Theoretic Approach with Secure Layout Techniques," IEEE/ACM International Conference on Computer-Aided Design (ICCAD), vol. 2017-Novem, pp. 329–336, 2017.
[35] Z. Chen, P. Zhou, T. Y. Ho, and Y. Jin, "How Secure is Split Manufacturing in Preventing Hardware Trojan?," IEEE Asian Hardware Oriented Security and Trust Symposium (AsianHOST), pp. 1–6, 2017.
[36] S. Patnaik, J. Knechtel, M. Ashraf, and O. Sinanoglu, "Concerted Wire Lifting: Enabling Secure and Cost-Effective Split Manufacturing," Asia and South Pacific Design Automation Conference (ASP-DAC), vol. 2018-Janua, pp. 251–258, 2018.
[37] S. Patnaik, M. Ashraf, J. Knechtel, and O. Sinanoglu, "Raise Your Game for Split Manufacturing: Restoring the True Functionality Through BEOL," Design Automation Conference (DAC), pp. 1–6, 2018.
[38] M. A. Masoud, Y. Alkabani, and M. W. El-Kharashi, "Obfuscation of Digital Systems using Isomorphic Cells and Split Fabrication," International Conference on Computer Engineering and Systems (ICCES), pp. 488–493, 2019.
[39] M. Li, B. Yu, Y. Lin, X. Xu, W. Li, and D. Z. Pan, "A Practical Split Manufacturing Framework for Trojan Prevention via Simultaneous Wire Lifting and Cell Insertion," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 38, no. 9, pp. 1585–1598, 2019.
[40] M. Bushnell and V. Agrawal, Essentials of Electronic Testing for Digital, Memory and Mixed-Signal VLSI Circuits. Springer Publishing Company, Incorporated, 2013.
[41] L. H. Goldstein and E. L. Thigpen, "SCOAP: Sandia Controllability/Observability Analysis Program," in 17th Design Automation Conference, pp. 190–196, 1980.
[42] T. Semiconductors, "3D-ICs and Integrated Circuit Security," [Online]. Available: http://www.tezzaron.com/media/3D-ICs_and_Integrated_Circuit_Security.pdf.
[43] D. Z. Pan, B. Yu, and J. Gao, "Design for Manufacturing With Emerging Nanolithography," IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 32, no. 10, pp. 1453–1472, 2013.
[44] N. Jasika, N. Alispahic, A. Elma, K. Ilvana, L. Elma, and N. Nosovic, "Dijkstra's Shortest Path Algorithm Serial and Parallel Execution Performance Analysis," in 2012 Proceedings of the 35th International Convention MIPRO, pp. 1811–1815, 2012.
[45] Y. Ding, C. Chu, and Wai-Kei Mak, "Throughput Optimization for SADP and E-beam Based Manufacturing of 1D Layout," in 2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6, 2014.
[46] W. Maly, N. Singh, Z. Chen, N. Shen, X. Li, A. Pfitzner, D. Kasprowicz, W. Kuzmicz, Y. Lin, and M. Marek-Sadowska, "Twin Gate, Vertical Slit FET (VeSFET) for Highly Periodic Layout and 3D Integration," in Proceedings of the 18th International Conference Mixed Design of Integrated Circuits and Systems (MIXDES), pp. 145–150, 2011.
[47] M. Weis, A. Pfitzner, D. Kasprowicz, R. Emling, T. Fischer, S. Henzler, W. Maly, and D. Schmitt-Landsiedel, "Stacked 3-dimensional 6T SRAM Cell with Independent Double Gate Transistors," in IEEE International Conference on IC Design and Technology, pp. 169–172, 2009.
[48] X. Qiu and M. Marek-Sadowska, "Can Pin Access Limit the Footprint Scaling?," in Design Automation Conference (DAC), pp. 1100–1106, 2012.
[49] M. Schobert et al., "Degate.," [Online]. Available: http://www.degate.org/documentation.
[50] R. Karmazin, C. T. O. Otero, and R. Manohar, "CellTK: Automated Layout for Asynchronous Circuits with Nonstandard Cells," in IEEE 19th International Symposium on Asynchronous Circuits and Systems, pp. 58–66, 2013.
[51] B. Hill, R. Karmazin, C. T. O. Otero, J. Tse, and R. Manohar, "A Split-Foundry Asynchronous FPGA," in Proceedings of the IEEE Custom Integrated Circuits Conference, pp. 1–4, 2013.
[52] J. Cheng, A. Fu, and J. Liu, "K-Isomorphism: Privacy Preserving Network Publication Against Structural Attacks," in Proceedings of the ACM SIGMOD International Conference on Management of Data, pp. 459–470, 2010.
[53] H. Salmani, M. Tehranipoor, and R. Karri, "On Design Vulnerability Analysis and Trust Benchmarks Development," in 2013 IEEE 31st International Conference on Computer Design (ICCD), pp. 471–474, 2013.
[54] P. Nguyen, T. Tran, P. Diep, and D. Le, "A Low-Power ASIC Implementation of Multi-Core OpenSPARC T1 Processor on 90nm CMOS Process," in 2018 IEEE 12th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSoC), pp. 95–100, 2018.

TIAGO D. PEREZ received the M.S. degree in electric engineering from the University of Campinas, São Paulo, Brazil, in 2019. He is currently pursuing a Ph.D. degree at 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 fields of work include digital signal processing, telecommunication systems and IC implementation. His current research interests include the study of hardware security from the point of view of digital circuit design and IC implementation.

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.

Appendix 2

[II]
T. Perez, M. Imran, P. Vaz, and S. Pagliarini, "Side-channel trojan insertion - a practical foundry-side attack via eco," in 2021 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5, 2021

Side-Channel Trojan Insertion – a Practical Foundry-Side Attack via ECO

Tiago Perez, Malik Imran, Pablo Vaz and Samuel Pagliarini
Department of Computer Systems - Tallinn University of Technology, Tallinn, Estonia
Emails: {tiago.perez,malik.imran,pablo.vaz,samuel.pagliarini} @taltech.ee

***Abstract*—Design companies often outsource their integrated circuit (IC) fabrication to third parties where ICs are susceptible to malicious acts such as the insertion of a side-channel hardware trojan horse (SCT). In this paper, we present a framework for designing and inserting an SCT based on an engineering change order (ECO) flow, which makes it the first to disclose how effortlessly a trojan can be inserted into an IC. The trojan is designed with the goal of leaking multiple bits per power signature reading. Our findings and results show that a rogue element within a foundry has, today, all means necessary for performing a foundry-side attack via ECO.**

***Index Terms*—hardware security, manufacturing-time attack, hardware trojan horse, side-channel attack, VLSI, ASIC.**

I. Introduction

The ever-increasing cost to build high-end semiconductor manufacturing facilities – estimated to cost $15-20B [1] – has made most design companies migrate to a fabless model. In practice, design houses can market integrated circuit (IC) solutions, but fabrication is outsourced to a third party. The practice of outsourcing can potentially affect the trustworthiness of an IC as a foundry (or a rogue element within the foundry) can manipulate the design for malicious purposes [2], [3].

Manufacturing-time attacks can tamper an otherwise trustworthy IC by inserting malicious logic or modifying specific aspects of the manufacturing process [4], [5]. These kinds of modifications are often referred to as hardware trojans (HTs). HTs are designed to leak confidential information, to disrupt a system's specific functionality, or even to destroy the entire system. Various types of HTs have been recently studied [6]–[15], demonstrating the potential threat of this type of attack. Moreover, a class of HTs has emerged for assisting side-channel attacks (SCA). Lin *et al.* [6] were the first to propose an architecture for assisting a power SCA. This specific type of trojan is the centerpiece of our work and in the remainder of this text is referred to as a side-channel trojan (SCT).

An IC's operating characteristics (e.g., timing, power consumption, electromagnetic radiation, etc.) can be used as a side-channel to indirectly reveal information that should be internal to the IC. For this reason, keys of crypto cores [16] are often targeted. However, to mount a successful SCA, it is necessary to acquire a large amount of data to perform correlation on. Using SCTs, on the other hand, the attack time and complexity is drastically reduced. The disadvantage of SCTs is that they require a circuit modification at fabrication time, which we later show is an **effortless exercise** for the attacker.

In [7], two lightweight SCT architectures are proposed, both with the intent to induce power consumption in order to leak a crypto key. The first architecture makes use of an adapted code-division multiple access (CDMA) scheme to distribute the leakage of bits over time. The modulated bits are forwarded to a special "leakage circuit" that creates a CDMA channel over the power side-channel. The second architecture, in addition to the CDMA scheme, also implements intermediate states within the AES key schedule to facilitate a differential power analysis (DPA) attack. Both architectures are implemented in a field programmable gate-array (FGPA) platform.

A silicon validated HT is presented by the authors of [8], [9]. Their demonstration is a cryptographic IC composed of an AES core and an Ultra-WideBand transmitter that leaks the key together with the transmission of the 128 bits of ciphertext. To broaden the scope of SCTs from dedicated crypto hardware to general-purpose processors (GPPs), an interesting architecture is described in [12], where software models of crypto standards (AES and RSA) are executed on GPPs. A number of simple micro-architectural modifications has been described to induce information leakage via faulty computations or variations in the latency and power consumption of certain instructions.

Despite the encouraging results reported from the SCT studies mentioned so far, no study discusses how SCTs could be inserted **from the perspective of the attacker**. In this work, we present not only an SCT design methodology, but also a novel framework for SCT insertion. We assume that a rogue agent inside the foundry is the adversary and that he/she makes use of **readily available engineering change order (ECO)** capabilities of physical design tools.

II. Threat Model and Attacker Capabilities

An attacker inside the foundry has the objective of inserting malicious logic in a finalized layout. Thus, he/she enjoys access to all technology and cell libraries utilized by the victim. This is particularly true for advanced nodes where only a handful of cell libraries per node exist. We assume the attacker is capable of identifying a crypto core in a layout, which is a reasonable assumption for well-known AES implementations that are often regular. We do not assume the adversary understands the entire victim's design. Instead, we assume the adversary can recognize the layout/structure of a crypto core within a larger design. Our assumptions are in line with [7], [9]. Furthermore, we also assume the adversary: 1) is versed in IC design, 2) enjoys access to modern EDA tools. With the help of the inserted logic in the form of an SCT, the attacker will then attempt to leak confidential information via a power signature. For this reason, crypto cores are often the target in this type of attack [9], [10] – this is also the scenario in our work.

A typical physical implementation flow is described in the upper portion of Fig. 1. The attack takes place after the victim's layout is sent for fabrication (see red portion of Fig. 1). Our threat model assumes that the attacker only has access to the layout (which is the norm when outsourcing IC fabrication)

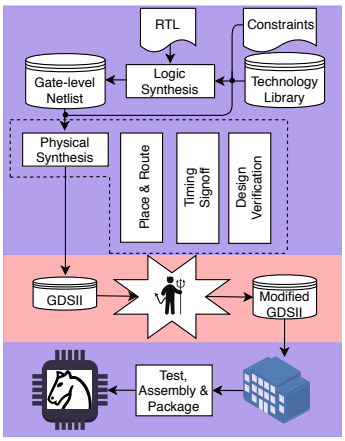

Fig. 1: A typical IC design flow. Highlighted in red is the untrusted fabrication stage where the attack takes place.

– he/she would not be able to insert the malicious logic by replicating the physical implementation flow since he/she does not have access to the RTL code, netlists, constraints, etc.

Nevertheless, EDA tools already have the capability to deal with finalized designs. This functionality is a feature referred to as ECO. Thus, an attacker holding only the layout could use an ECO to modify or insert additional logic in a finalized layout. An ECO flow requires four inputs: technology library, cell library, gate-level netlist, and a timing constraint. The adversary already possesses the first two, but must generate/estimate the others. A gate-level netlist can be effortlessly obtained from the victim's layout through extraction [17]–[19], while the timing constraint can be estimated to a certain degree [20]–[22]. Our novel trojan insertion framework is shown in Fig. 2, where these two steps are considered.

III. Side-Channel Trojan Design and Insertion

A. Side-Channel Trojan Design

Our proposed SCT is designed for creating an "artificial" yet controllable power consumption through which information is leaked. Since the majority of the power consumption in a circuit comes from the switching activity (dynamic power), a great candidate to be a power sink is a structure with a controllable frequency such as a dynamic ring-oscillator (RO). Our RO architecture implements delay stages broken into branches that are controlled by N_{leak} bits. Each RO branch has two active path options: a direct connection to the next branch or a series of delay cells. The power consumption created by paths is similar to a pulse-amplitude modulation with an order equal to $2^{N_{leak}}$. An example of SCT architecture for $N_{leak} = 2$ is illustrated in Fig. 2. The branch configuration is described in Table I, where the leaked bits are selectors S0 and S1.

TABLE I: Ring oscillator active path configuration

S0	S1	Delay Cells	Inverter Cells	Freq.
0	0	N_{D1}	N_i	High
1	0	$N_{D1} + N_{D2}$	N_i	Mid-high
0	1	$N_{D1} + N_{D3}$	N_i	Mid-low
1	1	$N_{D1} + N_{D2} + N_{D3} + N_{D4}$	N_i	Low

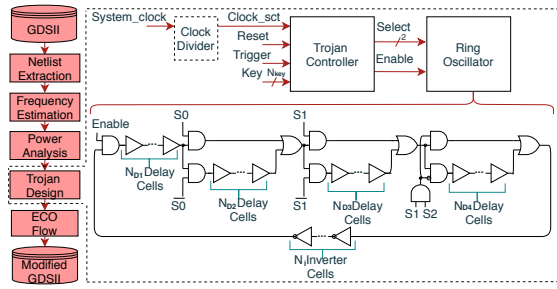

Fig. 2: Our SCT insertion methodology detailed.

A dual-sided constraint guides the attacker: he/she has to induce as much dynamic power as possible (i.e., to increase the effectiveness of the attack) while increasing as little leakage power as possible (i.e., to avoid detection). In this sense, not only the SCT has to be carefully planned, as well as when exactly will the trojan be triggered. Our approach is to not allow the trojan to compete with the dynamic power consumption of the crypto core. Therefore, when the core is actively working, the trojan is silent and the RO is not switching. When the crypto core is idle, the trojan kicks in. For this reason, our proposed SCT trojan has a *Trigger* signal that is connected to the *Done* signal coming from the crypto core, which marks the end of a cryptographic operation.

When triggered, the SCT connects a set of the leaking bits per clock cycle in the RO until all the N_{key} bits from the crypto key are leaked. Thus, our SCT requires a connection to the system clock and reset, a trigger signal, and the crypto key. Its architecture is illustrated in Fig. 2, consisting of three blocks: clock divider (DV), the trojan controller (TC), and the RO. The DV is required when the system clock is high and is responsible for dividing the frequency as the name suggests. Thus, the *Clock_sct* signal is either connect directly to the *System_clock* or to the DV. The TC is responsible for enabling the RO and for connecting the leaking bits in the RO. The RO starts running when the enable signal is asserted. The frequency is controlled by the select signals *S0* and *S1*.

To reduce the detection probability and increase the attack's feasibility, SCTs are tailored for each target circuit. Therefore, the SCT is designed with size and power constraints, i.e., we set thresholds for the SCT based on the target's size and static power. The attacker has to acquire such information from the layout. According to Fig. 2, the layout is inspected as follows:

Netlist extraction: since the attacker only holds the layout, a gate-level netlist has to be extracted by a CAD tool [17]–[19].
Frequency estimation: the attacker needs to estimate the target circuit operating frequency by performing static timing analysis on the extracted gate-level netlist. The attacker can try different clock frequencies and, by observing the critical path(s), can increase/decrease the frequency as needed until the timing slack is positive but near zero. The caveat is that multi cycle and false paths are expected to violate STA, and for this reason we say the frequency of operation is *estimated*.
Power analysis: with the extracted gate-level netlist and the estimated operating frequency, the attacker can perform a

typical power analysis. For relatively large circuits, a near-accurate static power estimation can be achieved even without input vectors.

Therefore, after inspection, the attacker has estimated frequency and power consumption and is now ready to draw his SCT. The RO's dynamic power is tweaked by choosing an adequate number of delay cells in each individual branch as well as the number of inverter cells in the feedback path. The achieved amplitude steps have to be sufficiently different from one another for the attack to be successful.

B. Side-Channel Trojan Insertion

After designing the SCT, the next step is its insertion. The attacker can utilize the ECO feature provided by commercial EDA tools for inserting the SCT. Typically, ECO is used to perform slight modifications in a finalized layout after its manufacturing (i.e., post-mask ECO). A special type of spare cell is utilized to enable ECOs. These cells do not add any functionality to the original design but, when needed, are instantiated by the ECO flow. By doing so, a new design can be generated with minimal changes in the fabrication mask set.

For the SCT insertion via ECO, since we previously established that the attacker can discern any gate in a layout, the attacker can replace both filler and spare cells by his malicious logic [23]. Contrarily to spare cells, every layout has filler cells. During placement, EDA tools have to spread the standard cells to assure routabilility, thus mandatorily leaving gaps between cells. For more details about the relationship between placement density and HT insertion, we direct the reader to [23].

According to Fig. 2, the ECO flow is the last step for the SCT insertion. In order to identify the filler/spare cells and remove them to create the gaps needed for the SCT, a single Cadence Innovus command is required. After the ECO, the attacker has to perform a timing sign-off to guarantee that the performance of the victim's design was kept. The SCT insertion is not likely to perturb the target's performance; it is only connected to a register (key storage) and some control signals, adding a small capacitance load. Besides, the coupling capacitance inserted by the additional routing wires is minimal due to the SCT's lightweight characteristic and the inherent goal of the ECO flow: not to disturb the existing logic. However, if the target circuit performance is perturbed, even if unlikely, it means that the size constraint used for designing the SCT was inappropriate - the adversary then proceeds to pick a different value and leak less bits per clock cycle. The attacker also has to check whether the SCT itself has timing violations. If so, the optional clock divider must be included. Every division by two requires one additional D-type flip-flop.

IV. Experiments and Results

For our experimental investigation, we have utilized AES and Present (PST) crypto cores with N_{key}=128 and N_{key}=80, respectively. AES was chosen due to its standardized status while PST was chosen due to its lightweight characteristic [24]. To allow the analysis of changes in *frequency* and *density*, the combination of these variables is explored as low-frequency low-density (LFLD), low-frequency high-density (LFHD), high-frequency low-density (HFLD), and high-frequency high-density (HFHD). Results from physical synthesis of the considered targets are presented in Table II. A 65nm CMOS technology was utilized to exercise very challenging placement densities (e.g., 75% for AES_LFHD) and frequencies (e.g., 0.95GHz for PST_HFLD). The values reported are for a typical corner.

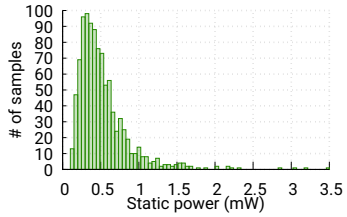

Fig. 3: PST_HFLD static power histogram, 10K MC samples.

Based on the pre-ECO results reported in Table II, different SCTs were designed for each target. We assume the attacker has no means to stop clock delivery to the whole circuit, so we included the clock tree power as it has to be accounted for in our SCT power constraint. Notice how the clock tree power is significant w.r.t. the leakage power of the targets, even for the LF variants. In the results that follow, we therefore set a power budget for our SCTs of 10% of the sum of leakage and clock tree power. Importantly, this is not a limitation of the methodology, an attacker can pick any other threshold.

Aiming to obtain a better representation of the static power of the cores, we performed a Monte Carlo (MC) simulation using Cadence Spectre. This simulation was performed for 1000 samples, varying only the process with temperature fixed to 25°C. The simulation results match the values reported in Table II for the typical corner. Fig. 3 depicts the static power distribution of PST_HFLD. As the SCT is implemented in the very same region of the IC as the target, we can also expect the same variation in its power.

Once the power constraint has been established, the attacker can proceed to estimate the multiple operating frequencies of the RO (and the associated power values that effectively leak the key). Moreover, as previously alluded, we have to take into account the placed and routed version of the SCTs. For this goal, we have taken each of our SCTs and performed a custom simulation using Cadence Spectre. The oscillation frequency and power consumption of the ROs are reported in Table III, where each RO has been termed with a "DXIY" suffix. X and Y represent the number of delay and inverter cells, respectively. Notice how we do not differentiate density in the results reported in Table III: either the trojan fits or it does not. The SCT design is nearly agnostic to placement density.

A visual representation of how the SCT performs is given in Fig 4. The set of leaked keys in the image is {00-10-01-11} and the target circuit is AES_LFHD. We also highlight an extreme case in the RO_{D6I4} which targets the PST_LF core. Here, the SCT alone represents about 10% of the size of the PST core. Since area and leakage have a linear dependency, the SCT's leakage already is about 10% of the target's leakage. Hence, the power constraint is violated. This extreme example assumes the entire IC consists of a single PST core. For a large system-on-chip containing multiple cores, the power budget for

TABLE II: Physical synthesis results for our considered targets, before and after trojan insertion.

Core	Frequency (MHz)	Before SCT insertion				After SCT insertion			
		Density (%)	Leakage (μW)	Clock Tree Power (μW)	Total Power (μW)	Density (%)	Leakage (μW)	Clock Tree Power (μW)	Total Power (μW)
AES_LFLD	100	61	77.4	115.2	1670	63.45	80	115.8	1720
AES_LFHD	100	75	75.8	116.7	1660	78.20	79	117.6	1720
AES_HFLD	1000	58	1048	1228	22800	59.37	1052	1238	23015
AES_HFHD	1000	72	1036	1241	22610	73.02	1040	1252	22830
PST_LFLD	95	53	14.13	32.05	371.3	67.33	20.71	34.75	483.4
PST_LFHD	95	70	14.09	31.89	371.2	82.05	17.72	32.85	428.5
PST_HFLD	950	52	34.02	325.30	3744	60.89	36.85	338.1	4022
PST_HFHD	950	69	34.13	329.10	3785	80.26	36.96	341.5	4015

TABLE III: RO operating frequency and power consumption

Target core	RO	Power & Frequency (μW & MHz)			
		S=00	S=01	S=10	S=11
AES_LF	RO_{D6I10}	19.52@65	16.89@45	14.94@34	12.96@20
AES_HF	RO_{D10I10}	198.4@551	182.5@483	160.7@390	139.8@300
PST_LF	RO_{D6I4}	15.95@112	11.55@58	10.22@39	8.7@20
PST_HF	RO_{D8I10}	42.02@79	35.56@61	30.88@46	25.66@31

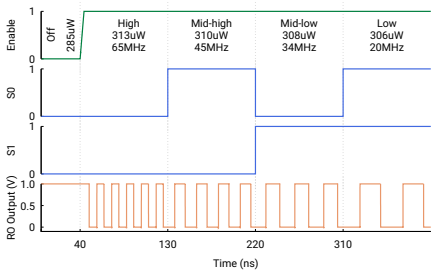

Fig. 4: Side channel trojan functionality example for the AES_LFHD, where the SCT utilizes the RO_{D6I10}.

designing the SCT would be much more forgiving.

Alongside the custom-simulated ROs, the SCTs are synthesized for each N_{key} and at the same clock frequency of the target. Exclusively for the HF targets, we added the CD block to ensure the SCT does not violate timing. For AES_HF, the system clock was divided by 8 while for PST_HF it was divided by 16. Area and cell count for the SCTs are given in Fig. 5.

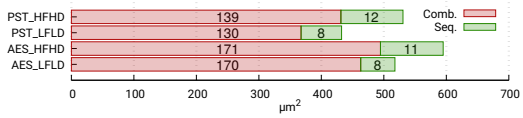

Fig. 5: Comparison of area and number of cells between SCTs.

After designing the RO and synthesizing the remainder of the SCT logic, the attacker is ready to perform the insertion via ECO. Insertion results are described on Table II ('After SCT insertion'). For all considered scenarios, the ECO flow was capable of placing and routing the SCT successfully, even for dense layouts. Considering that high density implies less routing resources, we verified that the ECO flow purposefully utilizes the least congested metal layers. We also provide a visual comparison of the density increase for the PST_HFHD SCT in the left side of Fig. 6. Note that the placement of the target was kept identical and only filler cells were removed

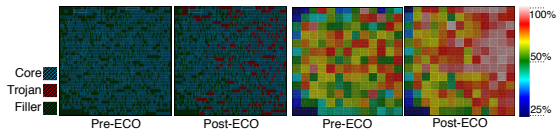

Fig. 6: Placement view (left) and density map (right) of the PST_HFHD core, before and after SCT insertion via ECO.

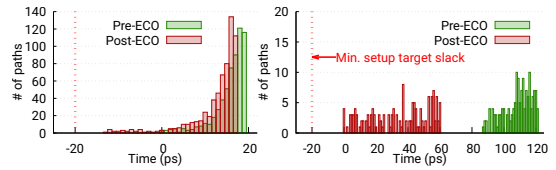

Fig. 7: Pre- and post-ECO setup timing slack comparison of AES_HFHD (right) and PST_HFHD (left).

during ECO. This is the key finding of this paper: **an adversary can effortlessly insert an SCT into a finalized layout**.

Besides enabling the SCT insertion, the ECO flow also has to preserve the performance of the target circuit. The impact on the performance of AES_HFHD and PST_HFHD cores is illustrated in Fig. 7. The difference in pre- and post-ECO timing slack is attributed to additional load and coupling capacitances. One can appreciate how the red bars in Fig. 7 are shifted to the left (w.r.t. the green bars). However, this shift was not sufficient to degrade the performance of any core. The PST_HFHD implementation is affected slightly (which is explained by the increase in density reported in Table II) but does not violate our safety margin of 20ps applied to all paths. Furthermore, we argue that our proposed methodology is not only capable of inserting an SCT in a high density layout, but also of keeping the target's performance regardless of its (challenging) frequency. Finally, there are very few techniques that would assuredly counter the ECO-enabled trojan insertion [3], [25].

V. Conclusions

In this work, we proposed an SCT design methodology as well as a novel framework for SCT insertion via ECO. The SCT insertion was detailed step by step, showing that a rogue element inside a foundry can replicate it effortlessly. Furthermore, our results show how efficient an otherwise benign ECO flow can be when used for malicious reasons. Our future work includes a silicon demonstration of the inserted HT. A tapeout was completed during the writing of this paper and the fabricated ICs are expected to arrive by Jan/2021.

ACKNOWLEDGMENT

This work has been partially conducted in the project "ICT programme" which was supported by the European Union through the European Social Fund.

REFERENCES

[1] M. Lapedus, "Big trouble at 3nm," [Online]. Available at: https://semiengineering.com/big-trouble-at-3nm/.

[2] U. Guin *et al.*, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1207–1228, 2014.

[3] S. Pagliarini, J. Sweeney, K. Mai, S. Blanton, S. Mitra, and L. Pileggi, "Split-chip design to prevent ip reverse engineering," *IEEE Design & Test*, 2020.

[4] 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.

[5] 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.

[6] L. Lin, W. Burleson, and C. Paar, "Moles: Malicious off-chip leakage enabled by side-channels," in *2009 IEEE/ACM International Conference on Computer-Aided Design*, pp. 117–122, 2009.

[7] L. Lin *et al.*, "Trojan side-channels: Lightweight hardware trojans through side-channel engineering," in *Cryptographic Hardware and Embedded Systems - CHES 2009*, pp. 382–395, 2009.

[8] Y. Jin and Y. Makris, "Hardware trojans in wireless cryptographic ics," *IEEE Design Test of Computers*, vol. 27, no. 1, pp. 26–35, 2010.

[9] Y. Liu, Y. Jin, and Y. Makris, "Hardware trojans in wireless cryptographic ics: Silicon demonstration & detection method evaluation," in *Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 399–404, 2013.

[10] R. Kumar, P. Jovanovic, W. Burleson, and I. Polian, "Parametric trojans for fault-injection attacks on cryptographic hardware," in *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 18–28, 2014.

[11] 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.

[12] J.-F. Gallais *et al.*, "Hardware trojans for inducing or amplifying side-channel leakage of cryptographic software," in *Trusted Systems*, pp. 253–270, 2011.

[13] K. Hasegawa, M. Yanagisawa, and N. Togawa, "Trojan-feature extraction at gate-level netlists and its application to hardware-trojan detection using random forest classifier," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, 2017.

[14] S. Bhasin and F. Regazzoni, "A survey on hardware trojan detection techniques," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2021–2024, 2015.

[15] S. Yu, C. Gu, W. Liu, and M. O'Neill, "A novel feature extraction strategy for hardware trojan detection," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2020.

[16] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99* (M. Wiener, ed.), pp. 388–397, 1999.

[17] Cadence Design Systems, "Virtuoso Layout Suite," [Online]. Available at: https://www.cadence.com/en_US/home/tools/custom-ic-analog-rf-design/layout-design/virtuoso-layout-suite.html.

[18] Mentor, "Calibre ®," [Online]. Available at: https://www.mentor.com/products/ic_nanometer_design/.

[19] Synopsys, "Synopsys Custom Design Platform," [Online]. Available at: https://www.synopsys.com/implementation-and-signoff/custom-design-platform.html.

[20] R. Torrance and D. James, "The state-of-the-art in semiconductor reverse engineering," *Design Automation Conference*, pp. 333–338, 2011.

[21] N. Albartus, M. Hoffmann, S. Temme, L. Azriel, and C. Paar, "Dana - universal dataflow analysis for gate-level netlist reverse engineering," 2020. https://eprint.iacr.org/2020/751.

[22] G. L. Zhang, B. Li, B. Yu, D. Z. Pan, and U. Schlichtmann, "Timingcamouflage: Improving circuit security against counterfeiting by unconventional timing," in *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 91–96, 2018.

[23] T. Trippel *et al.*, "ICAS: An Extensible Framework for Estimating the Susceptibility of IC Layouts to Additive Trojans," *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1078–1095, 2020.

[24] S. Ghandali *et al.*, "Side-channel hardware trojan for provably-secure sca-protected implementations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 6, pp. 1435–1448, 2020.

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

Appendix 3

[III]
T. Perez and S. Pagliarini, "A side-channel hardware trojan in 65nm cmos with 2μW precision and multi-bit leakage capability," in 2022 27th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 9–10, 2022

A Side-Channel Hardware Trojan in 65nm CMOS with 2µW precision and Multi-bit Leakage Capability

Tiago Perez, Samuel Pagliarini
Department of Computer Systems - Tallinn University of Technology
Tallinn, Estonia
{tiago.perez,samuel.pagliarini} @taltech.ee

Abstract — In this work, a novel architecture for a side-channel trojan (SCT) capable of leaking multiple bits per power signature reading is proposed. This trojan is inserted utilizing a novel framework featuring an Engineering Change Order (ECO) flow. For assessing our methodology, a testchip comprising of two versions of the AES and two of the Present (PST) crypto cores is manufactured in 65nm commercial technology. Our results from the hardware validation demonstrated that keys are successfully leaked by creating microwatt-sized shifts in the power consumption.

Our SCT architecture leaks information through an artificially and controllable induced power consumption [1, 2, 3]. This is done by implementing a ring-oscillator (RO) for dynamically creating the extra power consumption. For generating the power consumption multi-steps required for leaking multiple bits, our RO utilizes delay steps that are controlled by N_{leak} bits. In Fig. 1, an example of our SCT architecture for $N_{leak} = 2$ is depicted. Our SCT is triggered when the target is idle. An example of how our SCT performs is illustrated in Fig. 2.

For assessing our methodology, a testchip was manufactured utilizing a 65nm commercial technology, comprising of 4 crypto cores and a control block. We have implemented AES and PST with Nkey=128 and Nkey=80, respectively. Two versions of each crypto core were designed for different operating frequencies (AES@1GHz, AES@100MHz, PST@950MHz, and PST@95MHz) and a challenging density ($\approx 70\%$). Utilizing the flow depicted in Figure 1, for each crypto core we designed a specific SCT based on its power consumption. The novelty of our work is in the manner that the trojan logic is inserted in the design, i.e., via an ECO flow. Our SCT competes with the target leakage power, thus, it must induce as much dynamic power as possible (i.e., to increase the effectiveness of the attack) while increasing as little leakage power as possible (i.e., to avoid detection). The SCT insertion was possible even with the chosen challenging density, which represents fewer empty spaces for introducing the SCT. An example for the denser design, the PST@950MHz, is depicted in Fig. 3. A key feature of the ECO is not disrupting the performance of the target, as demonstrated at the setup timing slack comparison between pre- and post-ECO in Fig. 4.

This work has been partially conducted in the project "ICT programme" which was supported by the European Union through the European Social Fund

Our chip was designed in November of 2020, fabricated in March of 2021, and bench tests were conducted during July of 2021. Our bare die and its layout are contrasted in Fig. 5. For the hardware validation, we have access to 25 packaged samples of the chip. A printed circuit board (PCB) was designed specifically for the tests. For controlling the chip, we utilized a ZedBoard that contains a Xilinx Zynq-7000 All Programmable SoC. This setup is shown in Figure 6. The measurements were done as following: 1) a cryptokey with the 8 first bits being "11-10-01-00" was programmed; 2) a command for a single encryption was issued; 3) right after the encryption is done, all clock sources were turned-off; 4) three bursts of clocks were sent in order to shift the cryptokey connected to the RO. Our design can enable one crypto core at the time on the same chip. Thus, the values measured in our tests include only the leakage from the control block, the leakage from the selected crypto core and the dynamic power from the RO itself. The average across the samples of total power and leakage for each core is shown in Tab. 1. An example for AES@100MHz is shown in Figure 6, as seen in the ammeter, there are discrete steps representing the leaked bits "11-10-01-00" from the left to the right, respectively. The RO steps were approximated to a normal distribution, depicted in Figure 7. As seen for the PST@95MHz, our SCT was successfully capable of creating distinct steps with a precision of only 2μA.

The experimental measurements results obtained show that the variability in the manufacturing process does not affect the effectiveness of the RO for the smaller designs (AES@100MHz, PST@95MHz and PST@950MHz), meaning that the attack can be carried on with the same probability of success regardless of the sample. For the biggest design, AES@1GHz, the ECO had to spread farther away the RO cells because of the lack of empty spaces nearby. For this crypto core, the planned power steps were in the order of 200 μA, and the actual power steps after manufacturing were in the order of 60 μA. However, if the adversary has total control of the clock sources of the target, the attack will have a high chance of success due the distinctly separation of the power steps.

Our findings and results from the hardware phase validation demonstrated that our SCT was successfully inserted and was capable of leaking the cryptokey. Even more, it demonstrates that the ECO flow is a valid option for enabling a foundry-side attack. Therefore, proving that a foundry-side attack via ECO can be done by a

rogue element within a foundry, hence he/she has all the means necessary for performing it.

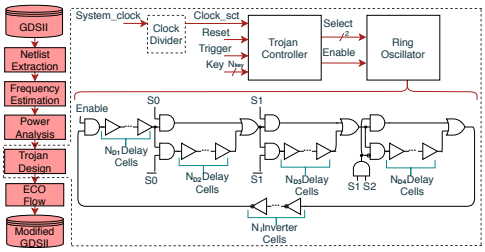

Fig. 1: Our SCT insertion methodology detailed.

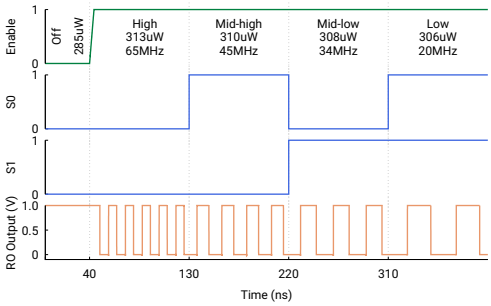

Fig. 2: Side channel trojan functionality example for the AES@100MHz.

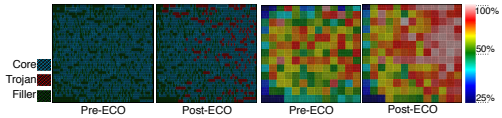

Fig. 3: Placement view (left) and density map (right) of the PST@950MHz core, before and after SCT insertion via ECO.

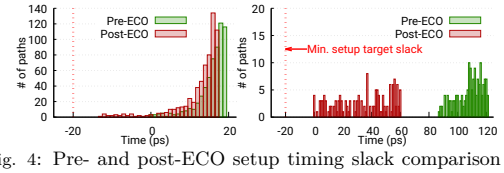

Fig. 4: Pre- and post-ECO setup timing slack comparison of AES@1GHz (right) and PST@950MHz (left).

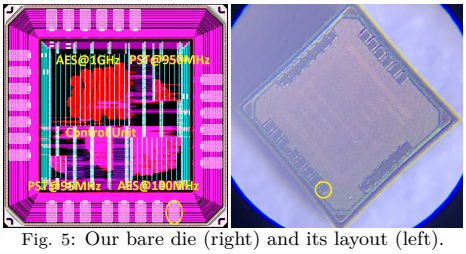

Fig. 5: Our bare die (right) and its layout (left).

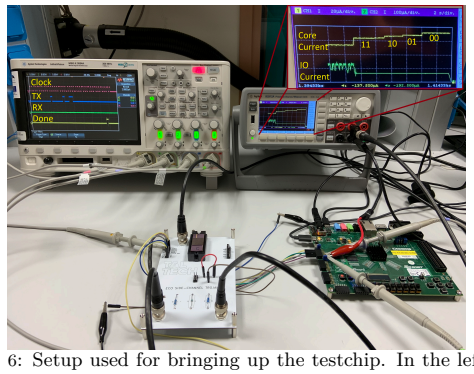

Fig. 6: Setup used for bringing up the testchip. In the left is shown the signals used for controlling, and, in the right the current consumption of the chip when the RO is active.

Table 1: Average of the total power and leakage across the tested chips for each crypto core.

Core	Total Power (μW)	Leakage (μW)
AES@1GHz	101160±10781	743.79±108.07
AES@100MHz	3139.32±85.38	131.57±10.35
PST@950MHz	9661.3±758.52	80.75±7.82
PST@95MHz	868.56±57.90	74.35±6.84

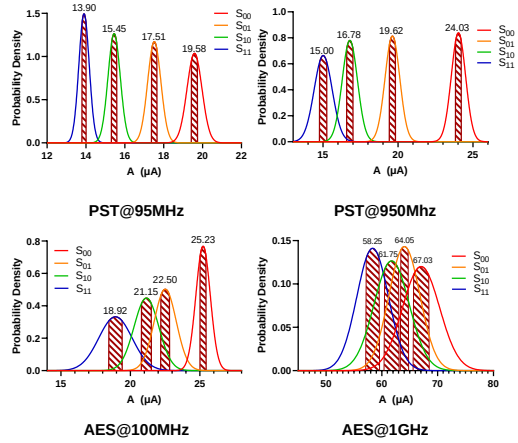

Fig. 7: Power consumption "steps" distribution for each crypto core. The red shadowed area represents the 95% confidence interval.

References

[1] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE D&T*, 2010.

[2] J.F. Gallais *et al.*, "Hardware trojans for inducing or amplifying side-channel leakage of cryptographic software," *Trusted Systems*, 2011.

[3] T. Perez, M. Imran, P. Vaz, and S. Pagliarini, "Side-channel trojan insertion - a practical foundry-side attack via ECO," *ISCAS'21*.

Appendix 4

[IV]
T. D. Perez, M. M. Gonçalves, L. Gobatto, M. Brandalero, J. R. Azambuja, and S. Pagliarini, "G-gpu: A fully-automated generator of gpu-like asic accelerators," in 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), pp. 544–547, 2022

G-GPU: A Fully-Automated Generator of GPU-like ASIC Accelerators

Tiago D. Perez*, Márcio M. Gonçalves†, Leonardo Gobatto†, Marcelo Brandalero‡, José Rodrigo Azambuja†, Samuel Pagliarini*

* Department of Computer Systems, Tallinn University of Technology (TalTech), Estonia
† Institute of Informatics, Federal University of Rio Grande do Sul (UFRGS), Brazil
‡ Brandenburg University of Technology (B-TU), Germany

Emails:{tiago.perez,samuel.pagliarini}@taltech.ee,{marcio.goncalves,leonardo.gobato,jose.azambuja}@inf.ufrgs.br,marcelo.brandalero@b-tu.de

***Abstract*—Modern Systems on Chip (SoC), almost as a rule, require accelerators for achieving energy efficiency and high performance for specific tasks that are not necessarily well suited for execution in standard processing units. Considering the broad range of applications and necessity for specialization, the design of SoCs has thus become expressively more challenging. In this paper, we put forward the concept of G-GPU, a general-purpose GPU-like accelerator that is not application-specific but still gives benefits in energy efficiency and throughput. Furthermore, we have identified an existing gap for these accelerators in ASIC, for which no known automated generation platform/tool exists. Our solution, called GPUPlanner, is an open-source generator of accelerators, from RTL to GDSII, that addresses this gap. Our analysis results show that our automatically generated G-GPU designs are remarkably efficient when compared against the popular CPU architecture RISC-V, presenting speed-ups of up to 223 times in raw performance and up to 11 times when the metric is performance derated by area. These results are achieved by executing a design space exploration of the GPU-like accelerators, where the memory hierarchy is broken in a smart fashion and the logic is pipelined on demand. Finally, tapeout-ready layouts of the G-GPU in 65nm CMOS are presented.**

***Index Terms*—ASIC generator, domain-specific accelerators, general-purpose gpu architectures, integrated circuits**

I. Introduction

New computer applications, especially in the field of Artificial Intelligence (AI), keep pushing the need for more energy-efficient hardware architectures [1]. For many years, application- and domain-specific accelerators, designed by specializing to the task at hand, have been the standard choice for achieving high energy efficiency. Canonical examples are crypto cores [2] and Graphics Processing Unit (GPUs) for which even specialized programming languages and paradigms have been proposed [3]. GPU architectures focus on specialized massively parallel many-core processors that take advantage of Thread-Level Parallelism (TLP) to handle highly parallelizable applications in a Single-Instruction Multiple Threads (SIMT) paradigm. GPUs have been traditionally designed for graphics applications but have recently evolved into efficient general-purpose accelerators for High-Performance Computing (HPC). HPC applications have a wide range, including oil exploration, bioinformatics, and the thriving AI and Machine Learning (ML) domains [4]. NVIDIA GPUs, for instance, are used as accelerators in several top500 supercomputers.

However, despite its widespread use as accelerators, research in GPU architectures is limited due to the lack of open-source models at a sufficiently low level of abstraction and that are representative of modern architectures. To the best of our knowledge, the only configurable open-source GPU architectures available in the literature are FlexGripPlus [5] and FGPU [6]. The first is based on the NVIDIA G80 decade-old architecture and has never been deployed to an FPGA board. The second was designed specifically for FGPA platforms. Therefore, the literature has not yet tackled the challenges in designing, configuring, and implementing modern GPU architectures for ASICs – a platform that presents challenges that are far from those in FPGA design. Still, all commercial GPUs are designed as ASICs.

This work proposes to **bridge this gap** with GPUPlanner, an automated and open-source framework for generating ASIC-specific GPU-like accelerators as IP. We term these general-purpose accelerators G-GPUs. GPUPlanner helps designers in generating GPU-like accelerators through user-driven customization and automated physical implementation. Customization is performed according to a given GPU architecture through a series of parameters that define computation characteristics (e.g., number of processing units) and memory access (e.g., cache sizes), thus providing designers a high degree of scalability to better fit the generated IP into their systems. Implementation strategies explore the use of *smart memories* and on-demand pipeline insertion.

We evaluate our proposed framework by implementing four flavors of G-GPU architectures in terms of performance, power, and area (PPA). Additionally, we provide a reasonable comparison with the popular CPU architecture RISC-V [7], [8] in terms of raw performance speed-up and performance speed-up derated by area. Our main contributions is an open-source framework for automated generation of GPU-like accelerators, from RTL to GDSII – the GPUPlanner.

II. Hardware Accelerators and our Baseline GPU

In a nutshell, domain- or application-specific accelerators cost too much. Recent developments in High-Level Synthesis (HLS) [9] are encouraging and have helped in accelerate the development of domain-specific hardware accelerators. Yet, for ASIC designs, the performance for flexibility trade-off is not interesting, or the performance is insufficient [10]. This scenario presents itself as an opportunity where general-purpose accelerators have gained ground. Our proposed GPUPlanner framework combines the efficiency from domain-specific accelerators and the ease of use from general-purpose architectures into G-GPU. The result is an automatically

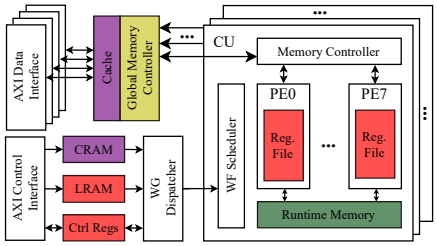

Fig. 1: FGPU architecture colored according to Fig 3.

generated domain-specific ASIC accelerator based on GPU architectures that can be easily programmed with modern programming languages.

FGPU is a configurable open-source GPU-like soft processor designed to accelerate workloads that fit in the SIMT paradigm [6]. Fig. 1 presents an overview of FGPU's architecture. Its main component is the Compute Unit (CU), a SIMD machine of 8 identical Processing Elements (PE0 - PE7) that can be spatially replicated up to eight times. A single CU can run up to 512 work-items (a computational kernel in OpenCL) and supports full thread-divergence, i.e., each work-item is allowed to take a different path in the control flow graph. Work-items are grouped into Wavefronts (WFs) that execute concurrently in a CU, and WFs are combined into Workgroups (WGs), which share a program counter and are assigned to a CU. FGPU is also deeply pipelined. On the software side, only standard OpenCL-API procedures are needed. Most importantly, FGPU can be artlessly scaled up to 64 processing units and is deeply configurable in terms of operations, instructions, and memory access.

Several past works have modified the FGPU to adapt it to different application domains. In [11], the authors have included new instructions along with micro-architecture and compiler enhancements to specialize FPGU for persistent deep learning, achieving 56–693x speed-up in PDL applications. MIAOW [12] is GPU-like implementation based on the AMD Southern Islands architecture and supporting its ISA. Scratch [13] extended MIAOW with automatic identification of the specific requirements of each application kernel and a tool that allows for the generation of application-specific and FPGA-implementable trimmed-down GPU-inspired architectures. Our work is the first to propose a tool that automatically generates tapeout-ready domain-specific accelerators based on GPU-like architectures and to make it publicly available.

III. GPUPlanner Framework

Our experimental investigation started from migrating the FGPU, originally designed for FGPA, to ASIC. To this end, a few changes in the architecture were necessary. As compilers for FGPA have a feature to infer memory from RTL automatically, all the memory blocks in the FGPU code were described as regular FFs. In ASIC, memory IPs are hand-instantiated instead of inferred. Thus, the first task was to clearly define intended behavior from the code and instantiate memory modules, utilizing a 65nm commercial technology.

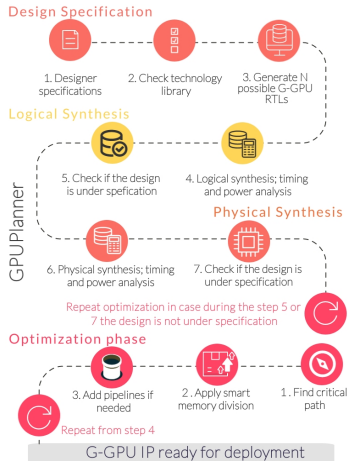

Fig. 2: GPUPlanner's G-GPU generation flow.

One of our main goals is to achieve the best PPA ratio possible from the G-GPU, exercising the maximum possible design space. The first aspect analyzed was the performance. This is done by finding the maximum operating frequency, which does not violate timing. For the logical synthesis, the value found for the standard version (without any of the optimizations done in this work) is 500MHz. The G-GPU has a similar performance across versions with different numbers of CUs because the CU itself is the bottleneck for performance in this architecture. As expected, the critical path for the version without any optimization has its starting point at a memory block. Also, the critical path was found inside the CU partition.

Larger memories display a higher delay for accessing the stored data when compared with smaller memories. This observation guides our design space exploration: dividing the memory blocks in the critical path is a valid strategy for increasing the performance of a design [14]. Memory division can be applied by diving the number of words, the size of the word, or both. This strategy requires a few alterations in the RTL code. First, the new modules have to be instantiated properly, substituting the target memories for the optimization. Second, the address or the input/output data have to be concatenated accordingly. To attain faster results, this task was fully automated in our framework.

The area of the memory blocks is not linear w.r.t. their size. In fact, two blocks of size $M \times N$ are larger and more power-hungry than a single block of size $2M \times N$ or $M \times 2N$. From the memory division alone, we are increasing the area and power. Also, a small extra logic is necessary to accommodate the addressing control. When exercising the memory division to enhance the design performance, we found cases where the critical path was not in memory blocks. For solving such timing issues, pipelines were introduced in those paths. As a result, we created an open-source tool to automatically generate G-GPU IPs, from RTL to GDSII. The flow of GPUPlanner is highlighted in Fig. 2. For starters, the designer has to define the specifications required from the G-GPU. Our architecture can be configured for CUs ranging from 1 to 8. Also, the designer

has to specify the operating frequency of the G-GPU.

After surveying the possible versions of the G-GPU for desired application scenarios, the designer can generate a specification for each scenario. Then, these specifications are contrasted with the characteristics of the technology intended to be used to create a first-order estimation of the G-GPU PPA. In this phase, there is a possibility to find several versions suitable for the given specification. Still, it also might happen that a configuration that suits the designer's requirements does not exist. However, our framework is not a static input generator. Instead, we provide a map on how to achieve a realistic PPA that might be close enough to the designer's requirements. This map is a dynamic spreadsheet, where the user input the delay of the memory blocks required for the non-optimized version of the G-GPU. Our map gives the maximum performance and which memory has to be divided or where to introduce pipelines to enhance the performance. This is an iterative process and can be repeated until the designer finds the desired performance. Thus, using our map, the designer can rapidly adapt his specification or create new versions of G-GPU. The only hard constraint in our framework is that many of the G-GPU memories have to be dual-port. Further development for single-port memories is scheduled as future work.

From a single push of a button, our framework can perform logic and physical synthesis of the list of designs. After the logic and physical synthesis, the resulting PPA is checked to guarantee it is under the initial specification. If the resulting G-GPU is out of the specifications, the designer should modify it and restart the process. In any case, the resulting layouts are ready to be integrated in a system as a tapeout-ready IP.

IV. Results and Discussion

From the exercise of the GPUPlanner, we found 12 versions worth the PPA trade-off in a general manner. These versions have 1, 2, 4, and 8 CUs. Their variants run at 500MHz, 590MHz, and 667MHz. The characteristics of each version are shown in Table I. In terms of area, the G-GPU size grows linearly with the number of CUs. The optimizations done for augmenting the performance increased the area by an average of 10%, from 500MHz to 590MHz, and 2%, from 590MHz to 667MHz. Thus, if the power consumption is not a priority, the 667MHz is a good fit for having a negligible increase in area in trade-off a better performance. These results demonstrate the potential scalability of the G-GPU architecture.

We chose four versions to perform the physical synthesis. Those are the 1CU@500MHz, 1CU@667MHz, 8CU@500MHz, and 8CU@677MHz. During this phase, the G-GPU is broken into three partitions during implementation: the CU, the general memory controller (MCTRL), and the top. The density of the CU and the MCTRL was set to 70%. Because of our floorplan strategy of breaking the design into partitions, the top has a low density of 30%. Nevertheless, breaking the design in partitions allows the designer to scale G-GPU without any extra effort. Once a CU partition is fully placed and routed, it can be implemented in versions with more than 1 CU by cloning the partition in the final floorplan of the design. Moreover, the user can create a collection of

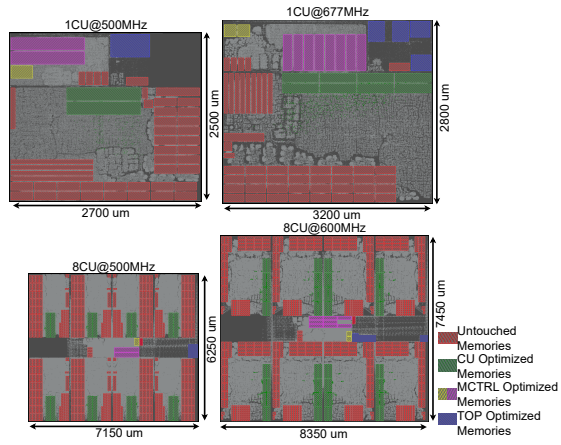

Fig. 3: Layout comparison between minimum and maximum performance of G-GPUs with 1 CU (top) and 8 CUs (bottom).

different CU layout blocks and scale the floorplan regarding the number of CUs for different application scenarios easily.

The layouts for the versions with 1 and 8 CUs are depicted in Fig. 3. Size scale in the figure only applies to the ones with the same number of CUs. The block memories divided for augmenting the performance are highlighted in green for the CU partition, yellow and pink for the MCTRL, and blue for the top. Note how different the floorplan is between the version with optimizations running at 600MHz and without optimizations running at 500MHz. Block memories have to be strategically placed in order to extract the maximum performance, hence, the differences in the floorplan. The layout of the versions 1CU@500MHz, 1CU@667MHz, 8CU@500MHz have the same performance expected from the logical synthesis (i.e., they can run at the specified clock frequency without any timing violation). However, the layout of version 8CU@667MHz can only run at 600MHz. This is explained by analyzing the floorplan of its layout (see Fig. 3). The connecting routing wires introduce a significant capacitance because of the long distance between the peripheral CUs and the general memory controller.

To fully evaluate the G-GPU as an ASIC accelerator, we compared its performance with the popular RISC-V architecture. We synthesized both architectures using the same technology used before with an operating frequency of 667MHz, the RISC-V having 32Kb memory and the G-GPU with 1/2/4/8 CUs. We chose seven micro-benchmarks from the AMD OpenCL SDK and increased their inputs up until crashing the RISC-V compiler. We further increased the input size of the G-GPU applications to make its computing units fully utilized. To compare the performance of the different-input size applications, we took a pessimistic approach for G-GPU and considered that one could increase RISC-V application input sizes by multiplying its cycle count by the G-GPU/RISC-V input size ratio. These results are shown in Fig. 4.

Our first evaluation compares raw performance between G-GPU and RISC-V for the same input sizes. For applications

TABLE I: Characteristics of 12 different GGPU solutions generated by our tool after logic synthesis in Cadence Genus.

#CU & Freq.	Total Area (mm^2)	Memory Area (mm^2)	#FF	#Comb.	#Memory	Leakage (mW)	Dynamic (W)	Total (W)
1@500MHz	4.19	2.68	119778	127826	51	4.62	1.97	2.055
2@500MHz	7.45	4.64	229171	214243	93	8.54	3.63	3.77
4@500MHz	13.84	8.56	437318	387246	177	16.07	6.88	7.14
8@500MHz	26.51	16.39	852094	714256	345	30.79	13.33	13.86
1@590MHz	4.66	3.15	120035	128894	68	4.73	2.57	2.66
2@590MHz	8.16	5.34	229172	221946	120	8.73	4.63	4.81
4@590MHz	15.03	9.72	436807	397995	224	16.41	8.70	9.02
8@590MHz	28.65	18.49	850559	737232	432	31.25	16.81	17.40
1@667MHz	4.77	3.26	120035	130802	71	4.65	2.62	2.72
2@667MHz	8.27	5.45	229172	222028	123	8.72	4.69	4.87
4@667MHz	15.15	9.83	436807	398124	227	16.43	8.75	9.07
8@667MHz	28.69	18.60	848511	730506	435	30.21	19.10	19.76

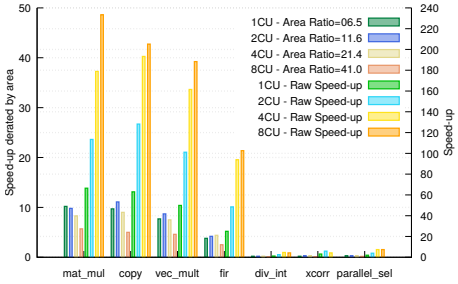

Fig. 4: Speed-up over RISC-V.

with low to no parallelism, G-GPU can be as low as only 1.2 times faster than RISC-V. As G-GPU is a domain-specific ASIC accelerator, such results are expected, once it will not be the best option for general-purpose applications. Therefore, a user interested in implementing a G-GPU as an accelerator can utilize these provided data to ponder if this type of architecture is a good fit for his system, considering only the raw speed-up.

Our second evaluation factors previously measured area into performance speed-up. We derated the previously measured speed-up by dividing the area ratio (G-GPU/RISC-V). A G-GPU with 1 CU has an area that is 6.5 times larger than the RISC-V, and it achieves the best increase in performance per area of 10.2 times the RISC-V's. On the other hand, G-GPU with 8 CUs has an area that is 41 times bigger than RISC-V's, thus achieving the best increase in performance per area of 5.7 times faster than RISC-V's. This trend happens mainly because data dependency and global memory communication limit parallelism. Thus, the provided increased processing power of a G-GPU configuration with more CUs.

We are planning to update the GPUPlanner to be able to implement the 8-CU G-GPU without performance loss. The performance problem of the layouts with 8 CUs has the possibility to be solved by replicating the general memory controller, shortening the distance between the peripheral CUs, and reducing the delay introduced by the routing wires. Also, we intend to include support of memory hierarchy and incorporate single-port memories into GPUPlanner.

V. Conclusion

Our results showed that G-GPUs are feasible domain-specific ASIC accelerator. Furthermore, when the G-GPU performance is contrasted with that of a RISC-V, it shows that our architecture has tremendous benefits for applications with high parallelism. Moreover, as GPUPlanner is an open-source framework, it gives the community the opportunity to explore the design space of GPU-like accelerators. Our work goes beyond the analysis of what constitutes a reasonable G-GPU accelerator in 65nm, as our tool can be easily extended to support other baseline GPU architectures and technologies.

Acknowledgments

This work has been partially conducted in the project "ICT programme", supported by the European Union through the European Social Fund. This work was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001, CNPq, and FAPERGS.

References

[1] V. Sze, Y.-H. Chen, T.-J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proc. of the IEEE*, vol. 105, no. 12, pp. 2295–2329, 2017.

[2] C. Mucci, L. Vanzolini, A. Lodi, A. Deledda, R. Guerrieri, F. Campi, and M. Toma, "Implementation of aes/rijndael on a dynamically reconfigurable architecture," in *2007 Design, Automation Test in Europe Conference Exhibition*, pp. 1–6, 2007.

[3] T. D. Han and T. S. Abdelrahman, "hicuda: High-level gpgpu programming," *IEEE Trans. on Parallel and Distributed Systems*, vol. 22, no. 1, pp. 78–90, 2011.

[4] P. P. Brahma, D. Wu, and Y. She, "Why deep learning works: A manifold disentanglement perspective," *IEEE Trans. on Neural Networks and Learning Systems*, vol. 27, no. 10, pp. 1997–2008, 2016.

[5] J. E. R. Condia, B. Du, M. Sonza Reorda, and L. Sterpone, "Flexgripplus: An improved GPGPU model to support reliability analysis," *Microelectronics Reliability*, vol. 109, p. 113660, 2020.

[6] M. Al Kadi, B. Janssen, and M. Huebner, "Fgpu: An simt-architecture for fpgas," in *ACM/SIGDA Int. Symp. on Field-Programmable Gate Arrays*, p. 254–263, ACM, 2016.

[7] M. Gautschi *et al.*, "Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices," *IEEE Trans. on VLSI Systems*, vol. 25, no. 10, pp. 2700–2713, 2017.

[8] OpenHW Group, "Cv32e40p risc-v ip," 2016. https://github.com/openhwgroup/cv32e40p.

[9] A. Canis *et al.*, "Legup: High-level synthesis for fpga-based processor/accelerator systems," FPGA'11, p. 33–36, ACM, 2011.

[10] J. Weng, S. Liu, V. Dadu, Z. Wang, P. Shah, and T. Nowatzki, "Dsagen: Synthesizing programmable spatial accelerators," in *ACM/IEEE Int. Symp. on Computer Architecture*, pp. 268–281, 2020.

[11] R. Ma *et al.*, "Specializing fgpu for persistent deep learning," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 14, July 2021.

[12] V. Gangadhar *et al.*, "Miaow: An open source gpgpu," in *IEEE Hot Chips Symp.*, pp. 1–43, 2015.

[13] P. Duarte, P. Tomas, and G. Falcao, "Scratch: An end-to-end application-aware soft-gpgpu architecture and trimming tool," in *IEEE/ACM Int. Symp. on Microarchitecture*, p. 165–177, ACM, 2017.

[14] H. E. Sumbul, K. Vaidyanathan, Q. Zhu, F. Franchetti, and L. Pileggi, "A synthesis methodology for application-specific logic-in-memory designs," in *ACM/EDAC/IEEE Design Automation Conference*, pp. 1–6, 2015.

Appendix 5

[V]
A. Hepp, T. Perez, S. Pagliarini, and G. Sigl, "A pragmatic methodology for blind hardware trojan insertion in finalized layouts," in 2022 International Conference on Computer-Aided Design (ICCAD), pp. 9–10, 2022

A Pragmatic Methodology for Blind Hardware Trojan Insertion in Finalized Layouts

Alexander Hepp
alex.hepp@tum.de
Technical University of Munich
Department of Electrical and Computer Engineering
Munich, Germany

Tiago Perez
Samuel Pagliarini
tiago.perez@taltech.ee
samuel.pagliarini@taltech.ee
Tallinn University of Technology
Department of Computer Systems
Tallinn, Estonia

Georg Sigl
sigl@tum.de
Technical University of Munich
Department of Electrical and Computer Engineering
Munich, Germany
Fraunhofer AISEC
Munich, Germany

ABSTRACT

A potential vulnerability for integrated circuits (ICs) is the insertion of hardware trojans (HTs) during manufacturing. Understanding the practicability of such an attack can lead to appropriate measures for mitigating it. In this paper, we demonstrate a pragmatic framework for analyzing HT susceptibility of finalized layouts. Our framework is representative of a fabrication-time attack, where the adversary is assumed to have access only to a layout representation of the circuit. The framework inserts trojans into tapeout-ready layouts utilizing an Engineering Change Order (ECO) flow. The attacked security nodes are blindly searched utilizing reverse-engineering techniques. For our experimental investigation, we utilized three crypto-cores (AES-128, SHA-256, and RSA) and a microcontroller (RISC-V) as targets. We explored 96 combinations of triggers, payloads and targets for our framework. Our findings demonstrate that even in high-density designs, the covert insertion of sophisticated trojans is possible. All this while maintaining the original target logic, with minimal impact on power and performance. Furthermore, from our exploration, we conclude that it is too naive to only utilize placement resources as a metric for HT vulnerability. This work highlights that the HT insertion success is a complex function of the placement, routing resources, the position of the attacked nodes, and further design-specific characteristics. As a result, our framework goes beyond just an attack, we present the most advanced analysis tool to assess the vulnerability of HT insertion into finalized layouts.

CCS CONCEPTS

• **Security and privacy → Malicious design modifications**; *Hardware reverse engineering*.

KEYWORDS

hardware security, reverse engineering, manufacturing-time attack, hardware trojan horse, VLSI, ASIC

ICCAD '22, October 30-November 3, 2022, San Diego, CA, USA

ACM ISBN 978-1-4503-9217-4/22/10…$15.00
https://doi.org/10.1145/3508352.3549452

ACM Reference Format:
Alexander Hepp, Tiago Perez, Samuel Pagliarini, and Georg Sigl. 2022. A Pragmatic Methodology for Blind Hardware Trojan Insertion in Finalized Layouts. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD '22), October 30-November 3, 2022, San Diego, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3508352.3549452

1 INTRODUCTION

Securing the development and manufacturing of integrated circuits (ICs) is a problem that the Hardware Security community is trying to solve [7]. As owning a foundry is not financially viable for most design houses, they have become fabless entities that have to rely on third-party foundries for manufacturing their designs. In this business model, the circuit layout is developed in-house and its manufacturing is outsourced. Exposing the layout to a third party is a potential threat to the IC's trustworthiness. A malicious individual could take ownership of this layout and manipulate it for his own purposes. Many potential threats have been discussed [30, 42], including insertion of hardware trojans (HTs), IP piracy, IC overbuilding, reverse engineering, and counterfeiting.

Even though only a few validated examples have been observed [13], the risk of a security breach due to hardware tampering has been in focus for many years [42]. Thus, in the past decade many potential vulnerabilities and possible countermeasures have been demonstrated. In this work, we focus on the feasibility of HT insertion during a fabrication-time attack.

HTs are designed to leak confidential information, to disrupt a system's specific functionality, or even to destroy the entire system [35] and have a broad taxonomy [16]. They comprise a *payload* implementing the malicious behavior and a *trigger* that ensures that the HT remains dormant until a specific condition is met. The *target* of an HT is the circuit into which the HT is inserted, e.g., a crypto-core or any other IP in a SoC design.

Several digital HT architectures have been proposed recently [42], with a few even demonstrated in silicon [13]. However, not many disclosed how their HT is inserted during the attack. Fabrication-time HT insertion in previous works relies on extensive knowledge of the victim's circuit [2, 27], for example of the security-critical nodes. In a fabrication-time attack, only the layout is available to the attacker, limiting the applicability of the approaches. The main contribution of our work is *to shed light on how successful an attack can be under the assumption of very limited information* about the target circuit.

High-level functionality reconstruction tools can be used to reconstruct the purpose of signals inside the design. For example, the

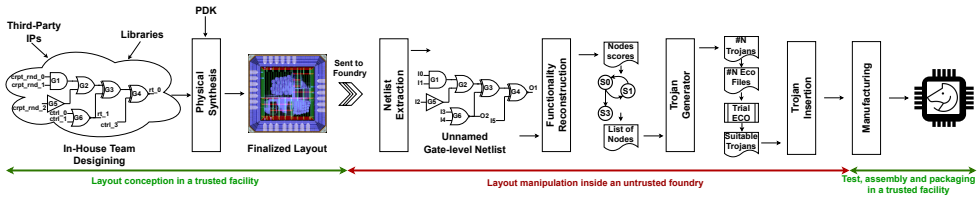

Figure 1: A typical IC design flow. Highlighted in red is the stage where a rogue element may mount an attack.

finite-state machine of a target design can be recovered and control and data paths nodes can be distinguished [22]. The output of such tools can be utilized to automate the search of security-critical nodes on a given layout – this is the case in our work. Our main contribution is a full framework for inserting a HT utilizing only the target layout as information. For assessing our methodology, we utilized as targets the cryptocores AES-128, SHA-256, and RSA, and the general purpose PULPino microcontroller. For each target, multiple different HTs are selected for insertion. We report the success of insertion based on stealthiness, impact on power and performance and required time and discuss HT defense technique capabilities in context of our attack.

2 BACKGROUND

2.1 Related Works and Motivation

Artificial HT creation is necessary, as no public HT examples are known. The majority of published artificial HTs have been created manually (e.g., Trust-Hub benchmarks [33]). Other HTs are designed to test or overcome specific detection methods [9]. A recent survey [42] lists 27 research articles presenting HTs, but only 4 of them perform insertion of a sophisticated, non-parametric HT on the layout level. The constraints of manual HT insertion into layouts have been studied in [2]. The authors conclude that a core utilization rate of >80% will prohibit trojan insertion, but only use one HT sample with varied parameters. In [39], the authors claim that removing unused ("dead") space is sufficient to thwart HT insertion. Trippel et al. [37] analyze the susceptibility of layouts to HTs by three metrics, the unused space available for HT gates, the amount of space available for wires and the distance between HT cell placement and attacked signals. However, the authors experiment with only four HT samples and manually insert them into three fully-known IC designs. This limits the explanatory power, as we show in Section 4.

In [27], the authors propose a full framework for HT insertion during a fabrication time-attack. The authors utilized engineering change order (ECO) technique for inserting HTs, which allows the attacker to perform the attack holding only the layout database. Nevertheless, [27] makes a **strong assumption** on how the attacker searches for the security-critical nodes: the attacker can spot these nodes by visual inspection. This is not true for targets with little available information or targets with irregular placement. This limits their attack framework to a few targets, e.g., the AES core used in their demonstration.

With the rise in popularity of machine learning, various automated HT insertion frameworks have been proposed and used to provide enough HT samples to train on. Cruz et al. [4] present a HT generator for gate-level netlists. Their tool generates diverse triggers, but relies on manual insertion for payloads and does not cover the insertion at layout. In addition, trigger location is determined by a simulation-based probability metric, requiring simulation testbenches to be available. In [3] the authors improve upon the previous tool. The authors emphasize that their tool generates similar HTs to those in the training set (hence the name MIMIC). This approach might not generate a trojan set that allows generalization in machine learning, as this requires a very diverse set of trojan variants, locations, and target designs [11].

A similar tool is presented in [43], also only diversifying the trigger insertion into netlists, but improving the trigger signal selection method to be executed without a simulation testbench. Still, payload location selection is simplistic and requires manual effort.

In [32], reinforcement learning is used for HT insertion, but their targets are small ISCAS-85 benchmarks. The reinforcement learning reward driving the insertion is chosen to be dependent on the ratio of circuit inputs involved in HT activation, on the observability of the payload changes, and on the usage of at least one net with low controllability. This approach is more diverse, but tuning the reward function is a non-deterministic task requiring a high level of understanding of both the circuit and the machine learning – which is infeasible for a blind attack.

As the implementations of previous works have not been published, multiple authors of detection techniques resorted to implementing template HT generators, varying the internal structure of the HTs, but not the HT locations [11, 10].

In conclusion, existing methods lack crucial features; even automated methods require extensive knowledge about the attacked design and thus are unfit for the blind insertion attacker model. When utilizing prior knowledge, most automated tools cannot diversify payload insertion, either because payload locations must be chosen with manual effort, or because only local variation around known payload locations are possible. In this paper, we overcome the limitations by the targeted usage of reverse engineering techniques and integrate it into an end-to-end framework for blind HT insertion.

2.2 Threat model and Attacker Capabilities

In our framework, we assume that an attacker within the foundry has the objective of inserting malicious logic in a finalized layout. Thus, since the attacker is familiar with the manufacturing process of the foundry, he/she enjoys access to all technology and cell libraries utilized by the victim when creating the layout. We assume the attacker has no detailed knowledge about the victim's design, such as timing/power constraints, clock domains, exact functionality of the input/output pins, or high-level functionality.

For performing the attack, we assume the adversary is skilled in IC design and enjoys access to modern EDA tools including their scripting languages. From the victim's side, the attacker only has access to the layout database – typically handled in the GDSII format.

For manipulating the victim's layout, we also assume the attacker knows how to apply reverse engineering techniques. Specifically, for the attack proposed in this work, the adversary has access to tools for extracting the gate-level netlist [29] and partially reconstructing the high-level functionality of the target design [22]. Furthermore, we also assume the attacker has no means to make radical modifications to the layout, e.g., manipulating the clock domains and/or changing the I/O configurations.

Designing an IC is typically executed as illustrated in Fig. 1. The layout-level work is generally considered trusted, done in-house. This is done using third-party IPs and a process design kit (PDK) together with standard cells provided by the foundry for a given technology. Here, we assume this process is trusted in which no malicious alteration is made for generating the finalized layout. The attack takes place when the victim's layout is handed over to the foundry. Precisely, the adversary is a rogue element inside the foundry that can manipulate the victim's layout before the start of the manufacturing (see the red portion of Fig. 1). This resembles attack model B as given in [41]. According to Karri et al. [16], the attacker inserts digital, gate-level HTs during fabrication time. In addition, we restrict the time when the rogue element has access to the victim's database to a full day (24 hours) [24]. This reduces attacker's capabilities to analyze and reverse engineer the victim's design, and to perform the HT insertion.

Typically, an ECO flow is used to fix small bugs in finalized layouts. ECOs are designed for post-mask modifications utilizing pre-populated gate-array cells, and pre-mask fixes, avoiding the time-consuming re-implementation of a design. However, Perez et al. [27] demonstrated that the ECO flow is a powerful tool for inserting HTs. The layout changes are local routing perturbations, as the original placement is kept intact. For inserting the HT, the ECO re-purposes empty spaces (filled with filler and spare cells[1]) with malicious logic. Since the ECO flow is executed by an industry-grade EDA tool, potential errors from manual modifications and design rule check (DRC) violations are avoided. The penalty for HT insertion is a slightly negative impact on the overall target performance due to the extra capacitance from HT wiring (see Fig. 5 for a performance comparison before and after the HT insertion). For a complete explanation of how to utilize the ECO for inserting an HT, we direct the readers to [27].

3 BIOHT TOOL FRAMEWORK

Our main contribution is a framework for blind insertion of hardware trojans in finalized layouts, termed BioHT. All the steps of this framework are automated; inserting HTs requires just a push of a button. Fig. 1 illustrates the BioHT framework, which comprises five distinct steps: 1) Netlist Recovery (Section 3.1), 2) Design Analysis (Section 3.2), 3) HT Netlist Generation (Section 3.3), 4) Hooking Signal Selection (Section 3.4), 5) HT insertion and Trigger Validation (Section 3.5). Fig. 2 shows the flow in detail.

3.1 Netlist Recovery from Layout

The attack begins once the rogue element within the foundry receives the victim's layout. First, a gate-level netlist has to be extracted from the layout. We refer to this gate-level netlist as *unnamed*, since the original hierarchy and names of cells and nets are assumed to be absent in the layout. Only the individual functionality of cells and their connectivity are recovered from the netlist extraction. Thus, in order to perform the attack, i.e., a HT-insertion with meaningful functionality, further reverse engineering is necessary.

An HT targets a specific part of the circuit, thus, **full understanding of the complete design is not necessary**. Instead of performing a full functional recovery, the attacker has to only identify security-critical signals and registers to hook the HT to. Using available information such as layout markings, datasheets, marketing material or patents, I/O-port descriptions can be inferred [28]. Global signals such as clock and reset can be identified from their connections to flip-flop pins. The resulting netlist is converted into a verilog gate-level netlist and input to design analysis. The process of netlist recovery can be seen in the top-left section of Fig. 2.

3.2 Design Analysis

During design analysis, *metrics* are generated to aid an adversary in searching for signals to be used for trigger and payload. This search requires a certain degree of understanding of the victim's design. For calculating those metrics, reverse engineering techniques are applied. As reverse engineering takes some time, it is a tradeoff to choose the desired level of design understanding. The metrics must be chosen carefully to keep the total runtime for the attack low.

Suitable metrics are transition probability, imprecise information flow tracking of selected signals, and the RELIC score [20]. Design analysis can be paralellized, as shown in the bottom-left portion of Fig. 2. In the remainder of this section, the calculation of these metrics is outlined.

Transition Probability: The HT trigger must seldomly activate in order to avoid detection during functional tests [41]. Thus, a metric is necessary to identify signals with low probability to activate the HT. In previous literature, the transition probability was introduced [31, 43]. If the transition probability is low, the probability that the required transitions for HT activation occur is also low. The transition probability is a function of the signal probability. The signal probability ($P_s(x)$) of x is the fraction of clock cycles during which x is 1. A transition occurs if a 1 follows a `0` or vice versa, i.e. the transition probability can be estimated as $P_t(x) = 2 \cdot P_s(x) \cdot \big(1 - P_s(x)\big)$, assuming temporal independence [26].

To initialize the calculation, set $P_s = 0.5$ for the primary inputs and all flip-flop outputs. This assumption is reasonable for crypto-cores and approximately correct for other large circuits. For each combinatorial gate in the fan-out of the initialization, P_s of the outputs is calculated. The output $P_s(o)$ of a 2-input or-gate is $P_s(a) + P_s(b) - P_s(a) \cdot P_s(b)$, the output $P_s(o)$ of a 2-input and-gate is $P_s(a) \cdot P_s(b)$ and the output $P_s(o)$ of a not-gate is $1 - P_s(a)$. The necessary formulas for other n-input 1-output gates can be produced by decomposing them into 2-input gates. In order to improve the probability results for flip-flop outputs, further iterations of the algorithm can be performed. In each iteration, the flip-flop input probabilities override the initial $P_s = 0.5$. Finally, the $P_t(x)$ is calculated for all signals and saved as the metric value.

Spatial Clustering: Co-location of HT hooking signals is desired for short wire lengths, but repeated distance calculation has a high overhead. Thus, BioHT precalculates using complete linkage clustering [25] with a L_1 distance metric: The target's cells are agglomerated into larger clusters. The agglomeration stops once a distance limit is reached, i.e., in the final clusters the cells remain closer than the limit. The mapping of cells to these clusters is saved as a metric.

[1] Typically, the placement density utilization (i.e, area with active cells versus unused area) is in the range of 50 to 60% for modern SoCs in the FinFET era. Thus, nearly half of the area of a commercial IC today can be populated by spare/filler cells.

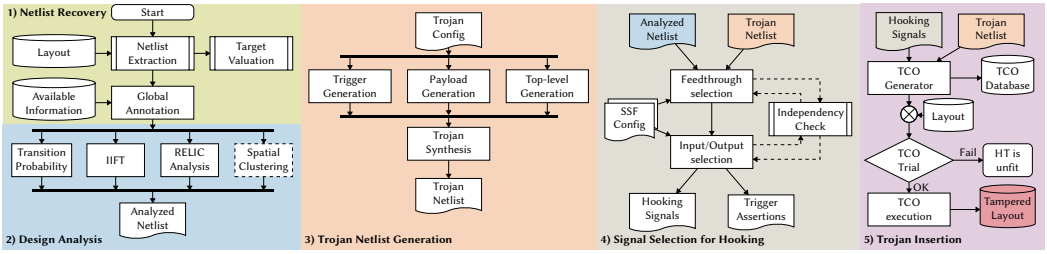

Figure 2: Steps 1)–5) of the BioHT Tool Framework explained in detail. Colored nodes represent an end-product of a previous step in the flow. The flow starts at the top left, while the tampered layout (highlighted in red) is the end-result.

Imprecise Information Flow Tracking: A HT that leaks information requires a metric that explains where valuable (tainted) input data can be found inside the circuit. Imprecise Information Flow Tracking (IIFT) [14] estimates an upper bound of information flow, as it assumes that each logic gate carries tainted information from its inputs to the output. The IIFT metric therefore marks as tainted any signal that exists in the output cone of the initially tainted signals. The mark, i.e., tainted or untainted, is saved as the metric value. This metric overestimates the availability of secret information at a specific signal. It must be complemented with another metric that explains the functionality of signals, such as the RELIC score.

RELIC Scoring and FSM identification: A valuable high-level information is the identification of whether a register (flip-flop or latch) belongs to the control logic or the data path. Using this information, HT payloads can be targeted to specific parts of the design functionality, for example to modify the control FSM or leak valuable processed data.

Finding the registers belonging to either the control or datapath logic can be performed with the NETA toolset [21]. RELIC 2 assigns every register a z-score that explains the level of dis-similarity to other registers. The underlying assumption is that data registers in a data-word show similar fan-in structure, while control registers are more unique. REDPEN returns pairs of dependent registers, i.e., a register dependency graph, in which every edge represents a path from the first register to the second register. TJSCC analyzes the register-dependency graph for its strongly connected components (SCC). As a result, there is a SCC for each register in the design. Combining RELIC and TJSCC, the SCC containing the register with the highest RELIC z-score implements the (most important) FSM in the design. Two metrics are saved for each register, its z-score and the SCC number it belongs to.

3.3 Hardware Trojan Netlist Generation

The BioHT HT Netlist Generator receives a configuration file, in which the user can choose any trigger/payload combinations and parameter values from our templates (see Fig. 3). Our selection of trigger and payloads covers known architectures [33, 18, 1], as well as novel payloads (i.e., leakage through FSK/DBPSK, fault sweeping). A configuration file generator is provided to ease the process of generating multiple configuration files. Nevertheless, the HT generation can be skipped, our framework is flexible enough for the user to implement their own trigger and/or payload architectures since the HT insertion only cares about the interface (i.e., where the HT is hooked to the target circuit).

For our HT architectures, we use three kinds of triggers; combinatorial, counter, and FSM; and four kinds of payloads; leak, shift'n'burn, modify, and fault. The explanation of the triggers and payloads is presented in Fig. 3. The HT triggers either wait for a *Combinatorial* condition v from n bits, waiting for v changes of n bits using a *Counter*, or traverse a s-state FSM on n-bit conditions v_i, masking some of the n bits with m_i. The payloads *Leak* n bits through a side channel code M at a rate of $1/2^c$ bits per clock cycle, *Shift'n'burn* energy with n transitions per clock cycle, *Modify* n-bits to a value v, or try to *Fault* n bits by flipping them in any combination (inspired by [2]). In order to differentiate the interfacing connections, we distinguish HT ports as: input-only, output-only, and feedthrough. HT input-only ports are used for triggering or for payloads that do not drive any node. HT output-only ports drive nodes, however, do not have any corresponding input hooked to the target circuit. HT feedthrough ports are a special case, they are pairs of input-output ports that disrupt original connections between two nodes of the original circuit. Thus, the HT has control of the bit value of the disrupted connection. This type of port is utilized by the Modify and Fault payloads (see Fig. 3). The entire process of the HT Netlist Generator is shown in step 3) of Fig. 2.

3.4 Signal Selection for Hooking

The fourth step of the BioHT framework is to select appropriate security-critical signals to hook the HT generated in the previous step. For searching those signals, the tool requires the metrics calculated during the design analysis, and the HT interface characteristics (i.e., number of input-only, output-only, and feedthrough ports). The process starts by associating a signal selection function (SFF) for each interface port of the HT. Then, the SSF iteratively yields candidate signals from the target circuit to be hooked to each HT port, all this based on one or multiple provided metrics. The *T* SSF yields all signals in the circuit in increasing order of transition probability, while ensuring that duplicate nets and buffer trees are avoided. This SSF is used if the total trigger probability is a sum of individual probabilities (e.g. for the *Counter* trigger). In contrast, the *TR* SSF yields signals with low transition probability randomly using a negative-exponential weighting function. This overcomes detection tools searching for rare signals only, while providing better diversity than a threshold-based approach [4]. These SFFs might require the additional use of the spatial clustering metric (TC and TCR) if signals selected by T and TR are spatially distant. Further SSFs are tailored to selecting payload signals. An explanation of SSFs implemented, and their behavior is given in Tab. 1.

Selection of signals to hook the HT to has to be meticulously done, and it is not only a function of the calculated metrics. It is

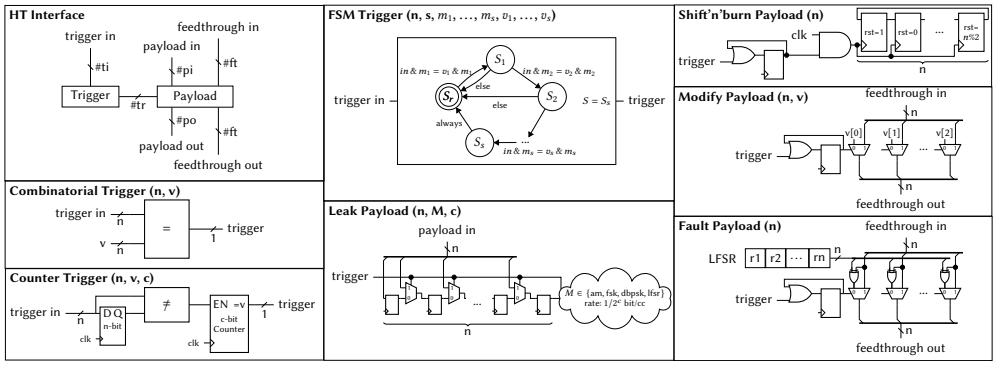

Figure 3: HT Interface and available trojan triggers and payloads. Trigger and payload parameters are given in parentheses.

Table 1: Available Signal Selection functions

Function	metrics used	Behavior
T	Trans. Prob.	sample low probability signals
TC	Trans. Prob., Spat. Clust.	sample low probability signals from adjacent clusters.
TR	Trans. Prob.	randomly sample low probability signals.
TCR	Trans. Prob., Spat. Clust.	randomly sample low probability signals from adjacent clusters.
RLR	RELIC z-score	randomly sample low z-score (i.e. data) signals
RLT	RELIC z-score, IIFT	sample tainted signals with z-score below threshold
RHS	RELIC z-score, SCC	sample signals from the highest-z-score SCC (i.e. FSM)
RHST	RELIC z-score, SCC, IIFT	sample signals from the tainted FSM (e.g. tainting instr. memory)
D	—	Connect no signal to this I/O. (e.g. shift'n'burn feedback signal)

highly desirable to avoid mutually dependent signals: If a signal used for triggering is dependent of an active payload signal, a combinational loop could be generated. Also, hooked signals used as Modify or Fault payloads should be independent, for maximizing the effectiveness of the HT.

For avoiding the situations described above, our signal search engine repeatedly checks the dependencies of each candidate signal. Only independent signals are considered for hooking the HT. To check the independence of two signals, a directed acyclic graph representation of the target design is created by replacing all registers by virtual input and output ports. Two signals are deemed independent if they do not have a common ancestor in the directed acyclic graph representation. This approach increases the amount of available trigger input signals significantly compared to the topological order approach used in other works [4, 43].

3.5 Trojan Insertion

For inserting the HTs into the victim's layout, we utilize an ECO flow similar to [27], as described in Section 2.2. In our framework (see Fig. 1), we approach the ECO differently than the one demonstrated in [27]. Instead of directly modifying the victim's netlist, we utilized the ECO file format. This format is supported by EDA tools and procedurally describes the modifications to be performed by the ECO. The advantage of utilizing this format is doing the ECO interactively. It is necessary to load the design only once for analyzing multiple ECO files before committing the modifications. By doing this, the adversary significantly saves execution time, since loading large designs can take several hours. Instead, loading an ECO file takes a few minutes. Before committing, the adversary can check if the HT fits in terms of placement resources and timing with a *trial insertion*.

To make the attack even faster, we introduce the concept of Trojan Change Order (TCO) format file. The TCO file is generated from the signals to hook the HT in combination with the HT netlist. We utilize the same syntax as the ECO file, however, with commented lines containing directives for the BioHT tool. Those directives are used to configure the type of HT (e.g., leak, deplete, modify or fault), number of connections and location of the HT gate-level netlist. Thus, it is possible to pre-generate TCO files for several types of HT, and specialize it according to the target's evaluation. Thus, it is feasible to create a database of HTs rapidly available for an attack, as shown in Fig. 2. Finally, if the HT fits, the attacker can commit the changes with a final TCO execution.

Trigger validation is performed after HT insertion by adding the respective SystemVerilog assertions (Trigger Assertions in Fig. 2) to the final netlist module and using any formal verification tool capable of handling cover property assertions. Besides the netlist, modern formal verification tools require only little additional information to prove HT triggering. Clock and reset inputs and polarities are found automatically, as well as most initialization sequences. If available from the design analysis step, the user can specify additional formal verification constraints and constants, e.g., for scan enable pins. If the validation is successful, the formal verification tool provides the attacker with a testbench to trigger the HT.

4 BIOHT FRAMEWORK EVALUATION AND RESULTS

In this section, we demonstrate our methodology and the settings utilized for evaluating the BioHT framework. All experiments performed in this work use industry-grade EDA tools. The layouts generated during the experiments are tapeout-ready, meaning they have proper power planning, timing closure, and no design rule violations.

Targets, Evaluation Settings and Procedure: For evaluating the BioHT capabilities, we have utilized four designs as targets – three crypto cores and one microcontroller. We chose the crypto cores AES-128, SHA-256, and RSA. The microcontroller utilized is the open-source SoC PULPino featuring one 32-bit RISC-V core [36]. The PULPino uses multiple memories and shows that BioHT can insert into large and complex targets.

For the implementation, we have utilized a commercial 65nm CMOS technology with a nominal voltage of 1.2V. For timing the designs, we utilized the slow process corner (SS), temperature of 125°C, and under voltage of 1.12. This is the worst case setup corner recommended by the vendor. For reporting the power consumption, we utilized the typical process corner (TT), temperature of 25°C, and the nominal voltage of 1.2V. These practices are in line with the standard flow adopted by the IC industry.

The first step is to generate a tapeout-ready layout for each target as the victim would do. For PULPino, RSA, and AES-128, we balanced density and performance, with the goal to achieve high-density designs operating at a considerably fast clock. For SHA-256, we set the density to a less ambitious value (i.e., around 50%) and then aimed for maximum performance. The results are presented in Tab. 2. It was possible to achieve both high-density and high-speed for AES-128 and RSA, above 80% of utilization and a clock frequency of 750MHz. In contrast, PULPino's density and clock frequency are low due to the presence of 8 memories (see Fig. 4). However, the achieved frequency of 285MHz is competitive with other PULPino implementations in a similar 65nm CMOS technology [6].

We extracted the gate-level netlist from the layout, estimated the operating frequency of the target and the power consumption (see Tab. 2), similarly as done in [27]. Steps 2)–4) were implemented in Python using several open-source libraries [8, 34, 23, 5]. Note that for all targets except PULPino, we assumed that signal tainting was only possible for the I/O. In PULPino, for the HT variant with Combinatorial trigger and Modify payload, we assumed that the attacker has acquired additional information, for example from an insider. This shows that our framework is capable to adapt to any additional available data about the design under attack. We assume that the data allows to taint the core control FSM registers, so that the payload can insert random pipeline flushes for performance degradation. After generating the TCO files, the actual HT-insertion is performed. To demonstrate the boundaries of HT insertion, we also executed unfit HTs that lead to trivially tampered layouts. Evaluation of the performance of the tampered layouts was done **using signoff settings** and provides accurate performance impact.

Experimental Results: To show the diversity of achievable HTs using BioHT, we performed the insertion of 96 variations of HTs. In the remaining results presentation, we are going to show only a handful of the possible HTs, the full results for all HT variants can be found at [12]. The results in Tab. 2 show the injected type of trigger and payload, the number of sequential/combinational cells, and the number of connections that the HT hooks to the target circuit. As mentioned before, BioHT can generate a vast number of HTs when employing all configuration options and trigger-payload combinations. Depending on the available target information, the attacker can narrow the range of fitting configurations and trigger-payload combinations. For example, a bit-flip fault as produced by the fault payload fits a crypto-core better than PULPino. Parametrization was guided by Trial TCO, as well as the independency check. For example, the independency check highlighted for both AES and RSA that there is no FSM, as only one independent FSM state register was found. This is why the Modify payload was not used for FSM tampering with these targets. Trial TCO guided towards low register count (that is lower n and c parameters), as register count was directly related to unfittability. Note that, among the selected HTs presented in this work are smaller and larger possible HTs. The SSFs were selected so that their behavior (see Tab. 1) fits the HT. We can conclude that BioHT is capable to produce diverse HTs for virtually any given target, while adapting to the amount of available information.

One goal of HT insertion is to guarantee that layout performance is not affected. The impact on the target layout is shown in Tab. 2 in terms of density, total power, timing (critical path slack), and design rules violations. An actual attacker would drop the HT variants that show unfit timing (large negative slack) or design rules violations (e.g. the last RSA variant). For all other variants, the impact of their insertion is not enough for breaking the design, making the resulting tampered layout manufacturable.

We selected the targets PULPino and RSA for demonstrating the HT insertion results in more detail. The HT variant with a Counter trigger and a Modify payload is challenging for its large number of cells (highlighted in bold in Tab. 2). The contrast between the layouts before and after the HT insertion is depicted in Fig. 4, where the malicious cells are highlighted in red, and the hooked cells are highlighted in light blue. For this image, we omitted the filler cells to improve the contrast between the original cells and HT cells.

The impact on the timing performance is demonstrated in Fig. 5. The green bars show the distribution of the timing slack of paths before and the red bars after inserting the HT. As portrayed in this figure, the overall impact is greater on the denser design, i.e., RSA. This is expected since the increase in the routing congestion is lesser in low-density designs. This claim is supported by the wire length statistics reported in Tab. 3[2]. From these results, it is clear that routing the additional HT logic is easier in PULPino (low-density design), because the additional wiring can be evenly distributed in all metal layers. The opposite is true for RSA (high-density design), where it was required to use an additional layer that was empty before, and in M5 more than the double amount of routing metal was used. Therefore, the timing impact on RSA from the additional coupling capacitance is stronger.

For this work, we executed further fit and unfit HT TCOs for demonstrating the dependency between the overall design implementation, HT architecture, and, number/location of chosen signals to hook the HT. Tab. 2 shows that the density of the design is not the only factor that makes an HT unfit. In the case of the AES-128, even with a density of 84.54%, all HTs were inserted without breaking the target design. This contradicts previous works that stated it would be impossible to insert HTs in designs with 80% of density or higher [2], or being a very difficult task if the placement resource available is small [37]. The size of the HT and the number of hooked signals are also not a sufficient metric to decide if an HT is unfit. Still using AES-128 as an example, the total timing impact does not depend on those HT characteristics, where the large timing

[2]In the 9-metal stack used in our experiments, M1 cannot be used for signal routing. For this reason, M1 is not shown. Similarly, M8/M9 are reserved for power distribution and are not shown.

impact is not from the largest HT. Therefore, it is not possible to rely on these metrics to assess with precision the vulnerability of a layout against HT insertion. Another experiment on PULPino was performed using the cluster-based SSFs (TC, TCR). By enforcing co-location of hooks, all PULPino variants become feasible. We conclude that the feasibility to co-locate HT signals (for a given attack objective) has a large impact on layout vulnerability.

When developing BioHT, an important aspect was to ensure a runtime within the 24h time window available to perform the attack. In Fig. 6, we show the runtime required for our attack in contrast with the design implementation runtime for PULPino and RSA. For PULPino, we show the HT-insertion with and without the use of the Spatial Clustering (optimized) Metric for searching the hooking signals. Trial TCO can easily determine if the overhead for spatial clustering is required. The runtime for RSA, PULPino with optimization, and PULPino without, are 50 minutes, 44 minutes, and, 24 minutes, respectively. As shown in the Fig. 6, the most time intensive tasks are the netlist extraction and power/time analysis. Those tasks are mandatory if the intention of the attacker is to insert meaningful and stealthy HTs. Our tool BioHT represents less than 20% of the attack total runtime. We conclude that HT generation with BioHT achieves competitive runtime overhead and would allow repeated attack execution for optimum performance.

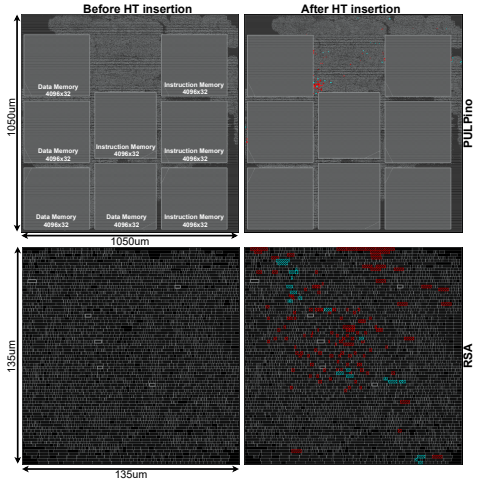

Figure 4: Layout contrast before and after HT insertion for PULPino and RSA. The HT inserted has a trigger counter and a modify payload. Its cells are highlighted in red.

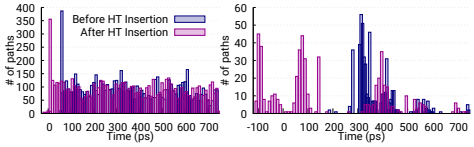

Figure 5: Timing impact of the hardware trojan Insertion, for PULPino (left) and RSA (right).

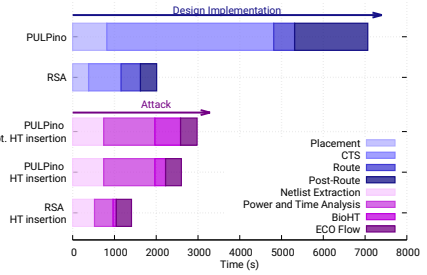

Figure 6: HT insertion using BioHT execution time (s) in contrast with the physical implementation of a target.

5 DISCUSSION

During the experiments, the ease of the BioHT flow proved valuable, as few configuration options have to be adapted to run HT insertion on a new design. Configuration was also guided by the tool itself. For example, the independency check proved to have a significant advantage over mere topological ordering. It aids the exploration of possible HT configurations, as it immediately highlights that specific configuration parameters are infeasible. It must be noted, however, that the experiments showed that, in rare cases, the independency check cannot guarantee that a HT can be triggered. In this case, the user is warned and recommended to use a different random seed for the trigger SSFs configuration, or select a different SSF. In our experiments, this occured once and was solved with a new random seed immediately.

An important and unforeseen result is that the influence of circuit cell density on the feasibility of HT insertion is much less than expected. In the two low density designs, SHA and PULPino, the HT insertion partially failed, while in the high density designs HT insertion succeeded, even for large HTs with hundreds of cells, and independent of the increase in wire length. We conclude that the vulnerability of a layout for HT insertion cannot be assessed by a few metrics. Instead, we propose to use BioHT as an empirical solution to assess the vulnerability by random HT insertion. BioHT goes beyond a proof-of-concept that blindly attacking a layout is possible. The framework can quickly produce a boundary of HT insertion feasibility, provide a risk assessment and guide physical defense strategies for HT insertion. Our claim is that no other work in the literature can provide this information.

In order to defend against sophisticated foundry-level attacks as in this work, two possibilities are pre-silicon design-for-trust or post-silicon detection techniques. Pre-silicon design-for-trust implements measures to render HT insertion more difficult or to provide trust anchors in the design to identify tampered locations. For example, logic locking renders understanding an unknown layout more difficult [17]. As previously explained, little available information is enough to insert meaningful HTs with BioHT. In how far this limits the effect of locking remains as a future work. Other design-for-trust techniques such as on-chip sensors must be circumvented, but automated solutions exist for many techniques (e.g. [40]), which are easily integrated into BioHT.

Post-silicon detection generally is either an exhaustive, error-prone and costly reverse engineering effort [19] to fully inspect the design, or can by principle only achieve partial coverage (e.g., with side-channel analysis, logic testing, etc.) [17]. BioHT can be

Table 2: Physical synthesis results before and after HT insertion for the target designs and several HT architectures.

Target	Hardware Trojan Characteristics							Physical synthesis results						
	Trigger	Payload	SSF		Seq.	Comb.	Conn.	Freq. (MHz)	Density (%)		Total power (µW)		Slack (ps)		Viol. #
			Trig.	Payl.	#	#	#	before	before	after	before	after	before	after	after
PULPino	FSM	Modify	TR	RHS	7	14	20			54.00		72.20		-4	0
	FSM	Shift'n'burn	TR	D	40	75	6			54.23		72.59		-8	0
	Comb.	Modify	TR	RHST	4	8	19			53.99		72.48		-9	0
	Comb.	Modify	TCR	RHST	4	15	31			54.03		72.77		2	0
	Comb.	Shift'n'burn	TR	D	37	71	8	285	53.97	54.26	71.88	72.61	15	-10	0
	Counter	**Modify**	**T**	**RHS**	**24**	**79**	**18**			**54.33**		**72.52**		**-18**	**0**
	Counter	Modify	TC	RHS	24	82	15			54.43		73.00		0	0
	Counter	Shift'n'burn	T	D	57	140	3			54.51		72.99		-46	0
	Counter	Shift'n'burn	TC	D	57	140	3			54.66		73.04		3	0
SHA-256	FSM	Modify	TR	RHS	24	85	28			54.38		47.27		20	0
	FSM	Shift'n'burn	TR	D	40	75	6			54.54		47.28		31	0
	Comb.	Modify	TR	RHS	4	18	38			53.94		47.28		-7	0
	Comb.	Shift'n'burn	TR	D	37	71	8	500	53.84	54.58	46.48	51.74	49	34	0
	Counter	Modify	T	RHS	24	85	25			54.38		47.27		20	0
	Counter	Shift'n'burn	T	D	57	140	3			55.03		51.94		45	0
RSA	FSM	Fault	TR	RLR	13	31	30			87.40		15.58		-84	0
	FSM	Shift'n'burn	TR	D	40	75	6			89.62		15.73		139	0
	Comb.	Fault	TR	RLR	10	29	33			87.21		15.87		-46	0
	Comb.	Shift'n'burn	TR	D	37	71	8	750	86.19	89.39	15.58	15.62	196	194	0
	Counter	**Fault**	**T**	**RLR**	**30**	**96**	**24**			**89.34**		**15.90**		**-108**	**0**
	Counter	Shift'n'burn	T	D	44	109	3			91.57		15.97		N/A	71
AES-128	FSM	Fault	TR	RLR	13	31	28			84.85		21.96		76	0
	FSM	Shift'n'burn	TR	D	40	75	6			85.85		22.11		30	0
	Comb.	Fault	TR	RLR	10	29	38			84.79		21.95		52	0
	Comb.	Shift'n'burn	TR	D	37	71	8	750	84.46	85.55	21.82	22.12	201	116	0
	Counter	Fault	T	RLR	30	96	24			85.99		22.15		119	0
	Counter	Shift'n'burn	T	D	10	29	2			84.78		21.94		163	0

Table 3: Routing length in µm per metal layer for PULPino and RSA, before and after the HT-insertion.

	RSA		PULPino	
Metal layer	before	after	before	after
M2	18.7k	18.9k	323.8k	326.5k
M3	23.2k	25.1k	439.2k	441.7k
M4	11.5k	15.9k	378.2k	382.4k
M5	2.5k	5.1k	357.1k	360.6k
M6	-	1.0k	221.3k	223.8k
M7	-	-	161.6k	163.2k

adapted to evade non-exhaustive post-silicon detection by modifying HT configuration and SSFs. An interesting approach is to include self-test structures in pre-silicon to improve post-silicon imaging detection [38]. In the end, post-silicon detection tries to prove that the design is identical to the pre-silicon layout, but a solution with guaranteed coverage is yet to be found.

As BioHT resembles a real-world blind foundry-level attack, the framework can complement post-silicon inspection by guiding inspection to vulnerable locations. This reduces the effort required for inspection, as only selected regions of the design must be inspected, namely those where BioHT potentially attacked the circuit.

Orthogonally, the use of machine-learning in any pre- and post-silicon HT detection is on the rise [17, 15]. Providing a vast amount of diverse HT samples is essential to train any machine-learning based detection technique. BioHT can be used blindly on any design to generate diverse training samples. Unlike with previous methods, it is not necessary to manually implement a sample set of trojans or to perform manual adaptation of the HT insertion technique. We expect the diversity of HTs and targets to be necessary and beneficial to future work on machine-learning based HT detection.

6 CONCLUSION

In this work, we presented BioHT, a framework to insert hardware trojans into unknown ASIC layouts. The tool is using an end-to-end approach, starting at the victim's layout delivered to the foundry and ending with the tampered layout ready for fabrication. Through the use of reverse engineering techniques, little knowledge about the design is necessary to introduce sophisticated trojans into the circuit. The use of state-of-the-art physical synthesis tools and an ECO flow allows performing the actual insertion as trouble-free as a layout bug fix. The end result is a DRC-clean layout that is ready for manufacturing, or an error message to the attacker that the parameters of insertion must be changed. Our experiments show that the complete approach can be executed multiple times in the time-frame between tape-out and manufacturing, so that the optimum trojan could be selected out of several possibilities. However, BioHT is also a tool to show how realistic trojan insertion would be performed, and can guide risk assessment, defense, and future research. To conclude, HT insertion in finalized layouts is a real threat to today's globalized IC manufacturing and must not be taken lightly. Our framework provides the necessary capabilities and information to advance countermeasures against this threat.

ACKNOWLEDGMENTS

This work has been partially conducted in the project "ICT programme" which was supported by the European Union through the European Social Fund. It was also partially supported by European Union's Horizon 2020 research and innovation programme under grant agreement No 952252 (SAFEST).

REFERENCES

[1] Alex Baumgarten, Michael Steffen, Matthew Clausman, and Joseph Zambreno. 2011. A case study in hardware Trojan design and implementation. *Int. J. Inf. Secur.*, 10, 1, (Feb. 1, 2011), 1–14. DOI: 10.1007/s10207-010-0115-0.

[2] S. Bhasin, J. Danger, S. Guilley, X. T. Ngo, and L. Sauvage. 2013. Hardware Trojan Horses in Cryptographic IP Cores. In *2013 Workshop on Fault Diagnosis and Tolerance in Cryptography*. (Aug. 2013), 15–29. DOI: 10.1109/FDTC.2013.15.

[3] Jonathan Cruz, Pravin Gaikwad, Abhishek Nair, Prabuddha Chakraborty, and Swarup Bhunia. 2022. Automatic Hardware Trojan Insertion using Machine Learning. (2022). https://arxiv.org/abs/2204.08580 arXiv: 2204.08580.

[4] Jonathan Cruz, Y. Huang, P. Mishra, and S. Bhunia. 2018. An automated configurable Trojan insertion framework for dynamic trust benchmarks. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*. (Mar. 2018), 1598–1603. DOI: 10.23919/DATE.2018.8342270.

[5] Chris Drake. 2015. PyEDA: Data Structures and Algorithms for Electronic Design Automation. In *Proceedings of the 14th Python in Science Conference*. Kathryn Huff and James Bergstra, (Eds.), 25–30. DOI: 10.25080/Majora-7b98e3ed-004.

[6] Michael Gautschi, Pasquale Davide Schiavone, Andreas Traber, Igor Loi, Antonio Pullini, Davide Rossi, Eric Flamand, Frank K. Gürkaynak, and Luca Benini. 2017. Near-Threshold RISC-V Core With DSP Extensions for Scalable IoT Endpoint Devices. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25, 10, (Oct. 2017), 2700–2713. DOI: 10.1109/TVLSI.2017.2654506.

[7] Ujjwal Guin et al. 2014. Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain. *Proceedings of the IEEE*, 102, 8, 1207–1228. DOI: 10.1109/JPROC.2014.2332291.

[8] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proc. SciPy '08*, 11–15.

[9] S. K. Haider, C. Jin, M. Ahmad, D. Shila, O. Khan, and M. van Dijk. 2017. Advancing the State-of-the-Art in Hardware Trojans Detection. In *IEEE Transactions on Dependable and Secure Computing*. Vol. PP, 1–1. DOI: 10.1109/TDSC.2017.2654352.

[10] Kento Hasegawa, Kazuki Yamashita, Seira Hidano, Kazuhide Fukushima, Kazuo Hashimoto, and Nozomu Togawa. 2022. Node-wise Hardware Trojan Detection Based on Graph Learning, (Mar. 15, 2022). Retrieved Apr. 4, 2022 from arXiv: 2112.02213.

[11] Alexander Hepp, Johanna Baehr, and Georg Sigl. 2022. Golden Model-Free Hardware Trojan Detection by Classification of Netlist Module Graphs. In *2022 Design, Automation Test in Europe Conference Exhibition (DATE)*, 1317–1322. DOI: 10.23919/DATE54114.2022.9774760.

[12] Alexander Hepp, Tiago Perez, Samuel Pagliarini, and Georg Sigl. 2022. BioHT (Blind Insertion of Hardware Trojans) Tool. https://github.com/Centre-for-Hardware-Security/bio_hardware_trojan.

[13] Alexander Hepp and Georg Sigl. 2021. Tapeout of a RISC-V crypto chip with hardware trojans: a case-study on trojan design and pre-silicon detectability. In *Proceedings of the 18th ACM International Conference on Computing Frontiers*. CF '21: Computing Frontiers Conference. (May 11, 2021), 213–220. DOI: 10.1145/3457388.3458869.

[14] Wei Hu, Armaiti Ardeshiricham, and Ryan Kastner. 2021. Hardware Information Flow Tracking. *ACM Comput. Surv.*, 54, 4, (May 3, 2021), 83:1–83:39. DOI: 10.1145/3447867.

[15] Z. Huang, Q. Wang, Y. Chen, and X. Jiang. 2020. A Survey on Machine Learning Against Hardware Trojan Attacks: Recent Advances and Challenges. *IEEE Access*, 8, 10796–10826. DOI: 10.1109/ACCESS.2020.2965016.

[16] R. Karri, J. Rajendran, K. Rosenfeld, and M. Tehranipoor. 2010. Trustworthy Hardware: Identifying and Classifying Hardware Trojans. *Computer*, 43, 10, (Oct. 2010), 39–46. DOI: 10.1109/MC.2010.299.

[17] Konstantinos G Liakos, Georgios K Georgakilas, Serafeim Moustakidis, Nicolas Sklavos, and Fotis C Plessas. 2020. Conventional and machine learning approaches as countermeasures against hardware trojan attacks. *Microprocessors and Microsystems*, 79, 103295. DOI: https://doi.org/10.1016/j.micpro.2020.103295.

[18] Lang Lin, Markus Kasper, Tim Güneysu, Christof Paar, and Wayne Burleson. 2009. Trojan Side-Channels: Lightweight Hardware Trojans through Side-Channel Engineering. In *Cryptographic Hardware and Embedded Systems - CHES 2009*. Christophe Clavier and Kris Gaj, (Eds.), 382–395.

[19] Matthias Ludwig, Ann-Christin Bette, and Bernhard Lippmann. 2021. ViTaL: Verifying Trojan-Free Physical Layouts through Hardware Reverse Engineering. In *2021 IEEE Physical Assurance and Inspection of Electronics (PAINE)*. 2021 IEEE Physical Assurance and Inspection of Electronics (PAINE). (Nov. 2021), 1–8. DOI: 10.1109/PAINE54418.2021.9707702.

[20] T. Meade, Y. Jin, M. Tehranipoor, and S. Zhang. 2016. Gate-level netlist reverse engineering for hardware security: Control logic register identification. In *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2016 IEEE International Symposium on Circuits and Systems (ISCAS), 1334–1337. DOI: 10.1109/ISCAS.2016.7527495.

[21] [SW] Travis Meade, Netlist Analysis Toolset (NETA), Mar. 16, 2018. URL: https://github.com/jinyier/NetA.

[22] Travis Meade, Shaojie Zhang, and Yier Jin. 2016. Netlist reverse engineering for high-level functionality reconstruction. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, 655–660. DOI: 10.1109/ASPDAC.2016.7428086.

[23] Aaron Meurer et al. 2017. SymPy: symbolic computing in Python. *PeerJ Computer Science*, 3, (Jan. 2017), e103. DOI: 10.7717/peerj-cs.103.

[24] Michael Muehlberghuber, Frank K. Gürkaynak, Thomas Korak, Philipp Dunst, and Michael Hutter. 2013. Red Team vs. Blue Team Hardware Trojan Analysis: Detection of a Hardware Trojan on an Actual ASIC. In *Proceedings of the 2nd International Workshop on Hardware and Architectural Support for Security and Privacy* (HASP '13) Article 1, 8 pages. DOI: 10.1145/2487726.2487727.

[25] Daniel Müllner. 2011. Modern hierarchical, agglomerative clustering algorithms. (2011). https://arxiv.org/abs/1109.2378 arXiv: 1109.2378.

[26] F.N. Najm. 1994. A survey of power estimation techniques in VLSI circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2, 4, (Dec. 1994), 446–455. DOI: 10.1109/92.335013.

[27] Tiago Perez, Malik Imran, Pablo Vaz, and Samuel Pagliarini. 2021. Side-Channel Trojan Insertion - a Practical Foundry-Side Attack via ECO. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–5. DOI: 10.1109/ISCAS51556.2021.9401481.

[28] Shahed E. Quadir, Junlin Chen, Domenic Forte, Navid Asadizanjani, Sina Shahbazmohamadi, Lei Wang, John Chandy, and Mark Tehranipoor. 2016. A Survey on Chip to System Reverse Engineering. *J. Emerg. Technol. Comput. Syst.*, 13, 1, Article 6, (Apr. 2016), 6:1–6:34. DOI: 10.1145/2755563.

[29] Rachel Selina Rajarathnam, Yibo Lin, Yier Jin, and David Z. Pan. 2020. ReGDS: A Reverse Engineering Framework from GDSII to Gate-level Netlist. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 154–163. DOI: 10.1109/HOST45689.2020.9300272.

[30] Masoud Rostami, Farinaz Koushanfar, and Ramesh Karri. 2014. A primer on hardware security: Models, methods, and metrics. *Proceedings of the IEEE*, 102, 8, 1283–1295. DOI: 10.1109/JPROC.2014.2335155.

[31] H. Salmani, M. Tehranipoor, and J. Plusquellic. 2012. A Novel Technique for Improving Hardware Trojan Detection and Reducing Trojan Activation Time. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20, 1, (Jan. 2012), 112–125. DOI: 10.1109/TVLSI.2010.2093547.

[32] Amin Sarihi, Ahmad Patooghy, Peter Jamieson, and Abdel-Hameed A. Badawy. 2022. Hardware Trojan Insertion Using Reinforcement Learning. (2022). arXiv: 2204.04350 [cs.LG].

[33] Bicky Shakya, Tony He, Hassan Salmani, Domenic Forte, Swarup Bhunia, and Mark Tehranipoor. 2017. Benchmarking of Hardware Trojans and Maliciously Affected Circuits. *Journal of Hardware and Systems Security*, 1, 1, (Mar. 2017), 85–102. DOI: 10.1007/s41635-017-0001-6.

[34] Shinya Takamaeda-Yamazaki. 2015. Pyverilog: A Python-Based Hardware Design Processing Toolkit for Verilog HDL. In *Applied Reconfigurable Computing* (Lecture Notes in Computer Science). Vol. 9040. (Apr. 2015), 451–460. DOI: 10.1007/978-3-319-16214-0_42.

[35] Mohammad Tehranipoor and Farinaz Koushanfar. 2010. A survey of hardware trojan taxonomy and detection. *IEEE Design and Test of Computers*, 27, 1, 10–25. DOI: 10.1109/MDT.2010.7.

[36] Andreas Traber, Florian Zaruba, Sven Stucki, Antonio Pullini, Germain Haugou, Eric Flamand, Frank K. Gürkaynak, and Luca Benini. 2015. PULPino: A small single-core RISC-V SoC. https://github.com/pulp-platform/pulpino.

[37] T Trippel et al. 2020. ICAS: An Extensible Framework for Estimating the Susceptibility of IC Layouts to Additive Trojans. *2020 IEEE Symposium on Security and Privacy (SP)*, 1078–1095. DOI: 10.1109/SP40000.2020.00083.

[38] Nidish Vashistha, Hangwei Lu, Qihang Shi, Damon L. Woodard, Navid Asadizanjani, and Mark Tehranipoor. 2021. Detecting Hardware Trojans using Combined Self Testing and Imaging. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 1–1. DOI: 10.1109/TCAD.2021.3098740.

[39] Xiaoxiao Wang, M. Tehranipoor, and J. Plusquellic. 2008. Detecting malicious inclusions in secure hardware: Challenges and solutions. In *Hardware-Oriented Security and Trust, IEEE International Workshop on*. (June 2008), 15–19. DOI: 10.1109/HST.2008.4559039.

[40] Xinmu Wang, Seetharam Narasimhan, Aswin Krishna, Tatini Mal-Sarkar, and Swarup Bhunia. 2011. Sequential hardware Trojan: Side-channel aware design and placement. In *2011 IEEE 29th International Conference on Computer Design (ICCD)*. 2011 IEEE 29th International Conference on Computer Design (ICCD). (Oct. 2011), 297–300. DOI: 10.1109/ICCD.2011.6081413.

[41] K. Xiao, D. Forte, Y. Jin, R. Karri, S. Bhunia, and M. Tehranipoor. 2016. Hardware Trojans: Lessons Learned After One Decade of Research. *ACM Trans. Des. Autom. Electron. Syst.*, 22, 1, Article 6, (May 2016), 6:1–6:23. DOI: 10.1145/2906147.

[42] Mingfu Xue, Chongyan Gu, Weiqiang Liu, Shichao Yu, and Máire O'Neill. 2020. Ten years of hardware Trojans: a survey from the attacker's perspective. English. *IET Computers & Digital Techniques*, 14, 6, (Nov. 2020), 231–246, 6, (Nov. 2020). eprint: https://ietresearch.onlinelibrary.wiley.com/doi/pdf/10.1049/iet-cdt.2020.0041. DOI: https://doi.org/10.1049/iet-cdt.2020.0041.

[43] S. Yu, W. Liu, and M. O'Neill. 2019. An Improved Automatic Hardware Trojan Generation Platform. In *2019 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*. (July 2019), 302–307. DOI: 10.1109/ISVLSI.2019.00062.

Appendix 6

[VI]
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 (TCAD), (to appear), 2022

Hardware Trojan Insertion in Finalized Layouts: From Methodology to a Silicon Demonstration

Tiago Perez, Samuel Pagliarini
Department of Computer Systems - Tallinn University of Technology, Tallinn, Estonia
Emails: {tiago.perez, samuel.pagliarini} @taltech.ee

***Abstract*—Owning a high-end semiconductor foundry is a luxury very few companies can afford. Thus, fabless design companies outsource integrated circuit fabrication to third parties. Within foundries, rogue elements may gain access to the customer's layout and perform malicious acts, including the insertion of a hardware trojan (HT). Many works focus on the structure/effects of a HT, while very few have demonstrated the viability of their HTs in silicon. Even fewer disclose how HTs are inserted or the time required for this activity. Our work details, for the first time, how effortlessly a HT can be inserted into a finalized layout by presenting an insertion framework based on the engineering change order flow. For validation, we have built an ASIC prototype in 65nm CMOS technology comprising of four trojaned cryptocores. A side-channel HT is inserted in each core with the intent of leaking the cryptokey over a power channel. Moreover, we have determined that the entire attack can be mounted in a little over one hour. We also show that the attack was successful for all tested samples. Finally, our measurements demonstrate the robustness of our SCT against skews in the manufacturing process.**

***Index Terms*—hardware security, manufacturing-time attack, hardware trojan horse, side-channel trojan, VLSI, ASIC.**

I. Introduction

The ever-increasing cost to build high-end semiconductor manufacturing facilities – building a 3nm production line is estimated to cost $15-20B [1] – has made most design companies migrate to a fabless business model. In practice, fabless design houses can design and market integrated circuit (IC) solutions, but the actual IC fabrication is outsourced to a third party. This practice can potentially affect the trustworthiness of an IC as a foundry (or a *rogue element* within the foundry) can manipulate the design for malicious purposes [2]. Many fabrication-time threats have been studied recently [3]. For combating these threats, numerous techniques have been proposed for increasing the trustworthiness of an IC. Examples of these techniques are Split Manufacturing [4], Logic Locking [5], [6] and IC Camouflaging [7]. Unfortunately, the current state of these techniques makes them unsuitable for large-scale production of ICs, either because of practicality [4] and/or insufficient security guarantees [8]. Thus, the current practices of a globalized supply chain leave the fabrication of ICs vulnerable to attacks.

Tampering an otherwise trustworthy IC can be done by inserting malicious logic or modifying specific aspects of the manufacturing process [10], [11]. These kinds of modifications are often referred to as hardware trojans (HTs). HTs are designed to leak confidential information, to disrupt a system's specific functionality, or even to destroy the entire system (referred to as time-bomb). Various types of HTs have been studied recently [12]–[20], demonstrating the potential threat of this type of attack.

An IC's operating physical characteristics, such as timing, power consumption, electromagnetic radiation, and even sound, can be used as a side-channel to indirectly reveal information that should be internal to the IC. For this reason, side-channel attacks (SCAs) often target keys of embedded crypto cores [21]. However, to mount a successful SCA, acquiring a large amount of data is usually required, followed by correlation/statistical analysis. Moreover, a very specific type of HT has been proposed for assisting SCAs. Lin *et al.* [12] were the first to propose an HT architecture for assisting a power SCA, referred to as "Malicious Off-chip Leakage Enabled by Side-channels" (MOLES). This specific type of trojan is the centerpiece of our work; in the remainder of this text it is referred to as a side-channel trojan (SCT). By using SCTs, the attack time can be drastically reduced as no further processing is required. The disadvantage of SCTs is their invasive nature. Inserting an SCT requires a modification of the circuit at fabrication time. While this might seem a difficult task at first sight, we later show how it can be executed by a capable attacker.

Despite the encouraging results reported from the SCT studies, no study discusses how SCTs could be inserted **from the perspective of the attacker**. Even more concerning, this is true even for silicon validated trojans [14], [15], [20]. This is also generally true for other types of HTs. In this work, we present not only an SCT design methodology, but also a novel framework for SCT insertion. We assume that a rogue element inside the foundry is the adversary and that he/she makes use of **readily available engineering change order (ECO)** capabilities of physical design tools. Consequently, the main contributions of this work are a full framework for inserting SCTs, an ASIC prototype for validating our methodology, and also a rich discussion regarding its effectiveness. We fabricated a testchip comprising of 4 cryptocores in a 65nm commercial technology, making our technique silicon proven. Our in-depth analysis of the side-channel versus the variation in the manufacturing process demonstrated that our SCT is robust against this type of skew. On top of that, we include an analysis of the attack time required for inserting our SCT in a finalized layout when also utilizing our ECO flow methodology. These characteristics sharply contrast our work with prior art.

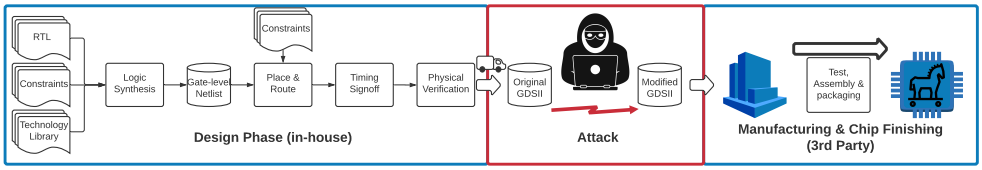

Fig. 1: A typical IC design flow. Highlighted in red is the stage where a rogue element may mount an attack (mod. from [9]).

II. Threat Model and Attacker Capabilities

In this work, the principal attacker we are concerned with is a **rogue element** inside the foundry. He/she has the objective of inserting malicious logic in a finalized layout. We emphasize that the attack occurs before the fabrication. Thus, since the attacker is located inside the foundry, he/she enjoys access to all technology and cell libraries[1] utilized by the victim when creating the layout. We assume the attacker is capable of identifying the presence of a crypto core in a layout, which is a reasonable assumption specially for well-known AES implementations that display regularity (due to the round-based key schedule structure). To be very clear, we do not assume that the adversary understands the entire victim's design (nor there is a need for such knowledge). Instead, we assume that the adversary can recognize the layout/structure of a single crypto core within a larger design, in line with the assumptions made in [13], [15].

Furthermore, we also assume the adversary: 1) is versed in IC design, 2) enjoys access to modern EDA tools, 3) has no means to make radical modifications to the circuit (e.g., adding new IOs or making changes in the clock domains). With the help of the inserted logic in the form of an SCT, the attacker will then attempt to leak confidential information via a power signature. Crypto cores are often the target in this type of attack [15], [16] – this is also the case in our work. As our attack deals with power signature reading, stopping some part of the clock delivery, or even entirely, would be highly beneficial for the attack. However, the attacker is assumed to have no knowledge about the clock domains or clock distribution in general. Synchronizing and controlling the HT's trigger to totally stop the clock delivery is not an option we have considered feasible, nor is the addition of an external trigger controlled by an IO.

A typical IC physical implementation flow is described in the left portion of Fig. 1. The attack takes place after the victim's layout in GDSII format is sent for fabrication (see red portion of Fig. 1). Suppose the attacker had access to all of the victims' data required to generate the layout (i.e., RTL, netlists, constraints, etc.). In this case, he/she could replicate the physical implementation flow to achieve a layout similar to the one created by the victim, yet now containing his malicious logic. This effort is theoretically possible but largely unpractical. Our threat model, therefore, assumes that the attacker only has access to the finalized layout. Design companies have to hand over their finalized layout to the foundry for fabrication. Normally, the layouts require some pre-processing steps before the start of the fabrication, which are handled by a foundry employee. Thus, it is during this time period that the attack can be mounted.

Nevertheless, EDA tools already have the capability to deal with finalized designs. This functionality is a feature referred to as ECO. Thus, an attacker holding only the layout could use an ECO to modify or insert additional logic in a finalized layout. An ECO flow requires four inputs: a technology library, a cell library, the gate-level netlist, and a timing constraint. The adversary already possesses the first two, but must generate the third and estimate the fourth input. A gate-level netlist can be extracted from the victim's layout [22], while the timing constraint can be estimated to a certain degree. Our proposed trojan insertion framework is shown in Fig. 2, where these two steps are considered. The details of the framework are described in the next section.

[1]This is particularly true for advanced nodes where only a handful of cell libraries per node exist. Typically, the foundry or a company licensed by the foundry provides a standard cell library. In either case, we assume the attacker has no difficulty identifying individual gates and their functionality.

III. Side-Channel Trojan Design and Insertion

A. Side-Channel Trojan Design

Our SCT is an additive hardware trojan with the intention of aiding a side-channel attack. For more details regarding hardware trojans taxonomy and concepts we direct the reader to [23]. Thus, it is designed for creating an artificial power consumption through which information is leaked. This has to be performed in a controlled manner, naturally. Knowing that the majority of the power consumption in a circuit comes from the switching activity (dynamic power), a great candidate to be a controlled power sink is a structure with a controllable frequency of operation.

A ring-oscillator (RO) is an example of such power sink if the number of stages in the RO can be adjusted dynamically as shown in Fig. 2. Our architecture implements variable delay stages broken into branches that are controlled by N_{leak} leaking bits. Each branch of our RO has two active path options: a direct connection to the next branch or a series of delay cells. Thus, each set of the N_{leak} is associated with a distinct change in the power consumption amplitude. This artificial power consumption created by the RO is similar to a pulse-amplitude modulation technique, with an order equal to $2^{N_{leak}}$. An example of this RO architecture for $N_{leak} = 2$ is illustrated in Fig. 2. The active paths' configuration is described in Table I, where the leaking bits become branch selectors and are referred to as S0 and S1.

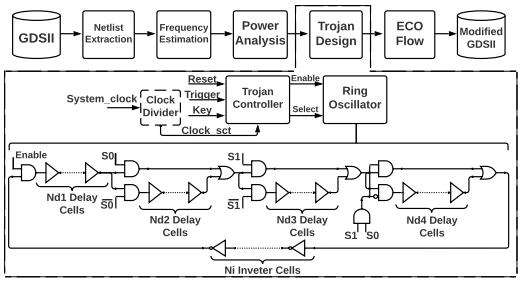

Fig. 2: Our trojan insertion methodology for a SCT capable of leaking 2 bits per power signature reading (modified from [9]).

TABLE I: Ring oscillator active path configuration

S0	S1	Delay Cells	Inverter Cells	Freq.
0	0	N_{D1}	N_i	High
1	0	$N_{D1} + N_{D2}$	N_i	Mid-high
0	1	$N_{D1} + N_{D3}$	N_i	Mid-low
1	1	$N_{D1} + N_{D2} + N_{D3} + N_{D4}$	N_i	Low

A dual-sided constraint guides the attacker's effort: he/she has to induce a discernible amount of dynamic power (i.e., to increase the effectiveness of the attack) while increasing leakage power as little as possible (i.e., to avoid detection). In this sense, not only the RO-based SCT structure has to be carefully planned, but a decision has to be made as to when exactly will the trojan be triggered. Our approach is to not allow the trojan to compete with the dynamic power consumption of the crypto core. Therefore, when the core is actively working, the trojan is silent and the RO is not switching. When the crypto core is idle, the trojan is triggered. For this reason, our proposed SCT trojan has a Trigger signal that is connected to the "done" signal coming from the crypto core, which marks the end of a cryptographic operation.

When triggered, the SCT connects a set of the leaking bits per clock cycle in the RO until all the N_{key} bits from the crypto key are leaked. Thus, our SCT requires a connection to the system clock and reset, a trigger signal, and the crypto key. Its architecture is illustrated in Fig. 2, consisting of three blocks: clock divider (DV), the trojan controller (TC), and the RO. Notice that our SCT does not required any additional external connections (i.e., I/Os or pads). All of its signals are connected to existing wires of the target circuit. The invasive portion of our attack is the insertion of new cells and routing wires.

The DV is responsible for dividing the frequency, as the name suggests. This feature is interesting for two reasons: there are scenarios when the attacker wants to slow down the speed at which bits are leaked, thus giving him/her control over the amount of time the attack will take. In other scenarios, the crypto core operates at a frequency that the trojan controller cannot match, so to avoid timing violations in the HT itself, clock dividing proves useful. Thus, the clock_sct signal is either connected directly to the system_clock or to the DV. The TC is responsible for enabling the RO and for connecting the leaking bits to the RO. The RO starts running when the enable signal is asserted; its operating frequency is controlled by the select signals S0 and S1 (see Fig. 4).

To reduce the detection probability and increase the attack's feasibility, the SCT has to be customized for each targeted circuit. Therefore, the SCT is designed with size and power constraints. The size constraint is set as a percentage of the target circuit size, and the power, on its turn, is a percentage of the target circuit's idle power. As the size and power constraints are set by analyzing the circuit's physical characteristics, the attacker has to acquire such information from the layout. According to the flow detailed in Fig. 2, the layout is inspected as follows:

Netlist extraction: since the attacker only holds the layout, a gate-level netlist has to be extracted. Such effort is considered a trivial task for an expert IC designer, demonstrated in [22].

Frequency estimation: the attacker has to estimate the operating frequency of the target circuit by performing static timing analysis on the extracted gate-level netlist. The attacker can observe the critical path(s) and then increase/decrease the frequency as needed to make the timing slack positive but near zero[2].

Power analysis: with the extracted gate-level netlist and the estimated frequency, the attacker can perform a typical power analysis. For relatively large circuits, static power can be estimated very precisely even without input vectors[3].

Therefore, after the layout is inspected, the attacker has acquired the estimated frequency, estimated power consumption, and exact size of the circuit (number of gates). From these, the attacker is now ready to draw the constraints necessary to design the SCT. First, the RO's dynamic power can be tweaked according to the power constraint. We remind the reader that the total power consumption can be divided into static and dynamic components as in (1). Leakage power is the static component of power and depends mainly on the threshold voltage of the transistors. On the other hand, dynamic power depends on the circuit's activity. Consequently, leakage power is proportional to the number of cells in the circuit while the dynamic power is proportional to the operating frequency. Thus, to model the amplitude steps required for the RO, we need to carefully model its dynamic power consumption.

$$P_{total} = P_{static} + P_{dynamic} \tag{1}$$

$$P_{dynamic} = \frac{1}{2} {V_{DD}}^2 F_{sa} \sum_{i_{net}} C_{load}(i) + F_{sa} \sum_{cell_j} E(j) \tag{2}$$

$$F_{sa} = 2F_{RO} = \frac{1}{\tau_{chain}} \tag{3}$$

Dynamic power can be calculated using (2), where C_{load} is the capacitance load at the output nets, F_{sa} is the switching

[2]Currently, our method does not take into account multi-cycle and false paths, which can reduce the accuracy of the frequency estimation but does not prevent the attack.

[3]For crypto cores in particular, it is a fair assumption to consider the plaintext to be randomly assigned, the adversary does not need precise vectors to estimate the (order of magnitude) of the power consumption.

activity factor, V_{DD} is the supply voltage, and E is the total energy of a cell. The switching activity factor describes how many switches will occur per second. As for the RO, since the signals are always switching, this factor is two times the RO's oscillation frequency, which can be estimated by calculating the total path delay of the ring as in (3).

The total path delay of the RO is estimated using (4), where τ_{delay} is proportional to the number of delay cells in the active path times the delay of each cell (τ_{dcell}), see (5). The $\tau_{inverter}$ delay is from the inverter cells in the feedback path, described by (6). The $\tau_{control}$ delay is from the logic cells that controls the active paths, described by (7). Since $\tau_{inverter}$ and $\tau_{control}$ are fixed for a given implementation, τ_{delay} is the knob utilized to change the frequency of oscillation dynamically. Therefore, the RO can be designed by choosing the adequate number of delay cells in each individual branch as well as the (static) number of inverter cells in the feedback path.

$$\tau_{chain} = \tau_{delay} + \tau_{inverter} + \tau_{control} \quad (4)$$

$$\tau_{delay} = (N_{D1} + N_{D2} + N_{D3} + N_{D4})\tau_{dcell} \quad (5)$$

$$\tau_{inverter} = N_i\tau_{invcell} + \tau_{nand} \quad (6)$$

$$\tau_{control} = 7\tau_{and} + 3\tau_{or} \quad (7)$$

Finally, the equations above give a first-order estimation of the power profile of the RO. However, since the RO will still undergo place and route, cell position and length of wires have to be properly accounted for. This is achieved by utilizing a SPICE-level simulator and is addressed in Section IV.

B. Side Channel Trojan Insertion

After designing the SCT, the next step is its insertion. The attacker can utilize the ECO feature provided by commercial EDA tools for inserting the SCT. Typically, this feature is used to perform slight modifications in a finalized layout after its manufacturing, referred to as post-mask ECO. A special type of logic cell, called spare cell, is utilized to enable the ECO methodology. Spare cells are typically inserted in commercial ICs and, when needed, are instantiated by the ECO flow. By doing so, a new design can be generated with minimal changes in the fabrication mask set.

For the SCT insertion via ECO, an attacker can achieve his/her goal without utilizing spare cells. Since we previously established that the attacker can discern any gate in a layout, the attacker can replace both filler and spare cells for his malicious logic. Contrarily to spare cells, every layout of a digital circuit has filler cells. During placement, EDA tools have to spread the standard cells to assure routability, thus mandatorily leaving gaps between cells. For more details about the relationship between placement density and HT insertion, we direct the reader to [24].

According to Fig. 2, the ECO flow is the last step for the SCT insertion. Before the ECO, the attacker has to identify the filler/spare cells and remove them to create the gaps needed for his own logic. This is achieved by *literally one command* in physical synthesis tools, as deleting fillers is a typical operation to be performed. After the ECO, the attacker has to perform timing sign-off to guarantee that the performance of the victim's design was not disturbed. The SCT insertion is not likely to perturb the target's performance; it is only connected to a register (crypto key storage) and some control signals, adding a small capacitance load to them. Besides, the coupling capacitance inserted by the additional routing wires is minimal due to the SCT's lightweight characteristic and the inherent goal of the ECO flow: *not to disturb the existing logic*. However, even if unlikely, the addition of the SCT could hinder the target performance. In that case, it means that the size constraint used for designing the SCT was inappropriate.

The attacker also has to check whether the SCT itself has timing violations. If so, the optional clock divider must be included to slow the SCT clock (w.r.t. the system clock). Every division by two requires one additional D-type flip-flop.

IV. Implementation and Simulation Results

In this section, we demonstrate our methodology and justify our chosen target designs, i.e., the crypto cores that we are going to insert our trojans on. The first part of the experiments are from simulations done using industry-grade EDA tools. After validating the methodology through simulation, the next step is the demonstration of our ASIC prototype, discussed in Section V.

A. Targets and Conditions

For our experimental investigation, we have utilized AES-128 and Present (PST) [25] crypto cores with $N_{key} = 128$ and $N_{key} = 80$, respectively. The AES crypto core was chosen due to its standardized status and popularity, while PST was chosen due to its lightweight characteristic [26].

To allow the analysis of SCT insertion for both AES and PST cores regarding changes in *frequency* and *placement density*, the combination of these variables is explored in Table III. In the column titled 'Acronym', we define the terminology used for referring to the many variants of the cores. Results from physical synthesis of the considered targets are presented in Table II. A 65nm commercial CMOS technology was utilized to exercise very challenging placement densities (e.g., 75% for AES_LFHD) and frequencies (e.g., 0.95GHz for PST_HFLD). The values reported are for typical process corner (TT) and a nominal temperature of 25°C.

B. SCT Design Results

Initially, all studied cores were physically synthesized for the placement and frequency conditions set above. These results are obtained from Cadence Innovus and are reported as pre-ECO results in Table II ("Before SCT insertion"). As we assume the attacker has no means to stop the clock delivery to the entire circuit, we have included the clock tree (CT) power in our results as it has to be accounted for in the SCT power constraint. Notice how the CT power is significant when compared to the leakage power of the targets, even for the LF variants. Different SCTs were designed for each target by setting a power budget for the SCTs to be 10% of the sum of leakage and CT power. Importantly, this is **not** a limitation of the methodology, an attacker can pick any other threshold and still design the SCT accordingly.

TABLE II: Physical synthesis results for our considered targets, before and after trojan insertion.

Core	Frequency (MHz)	Before SCT insertion Density (%)	Leakage (μW)	CT (μW)	Total Power (μW)	After SCT insertion Density (%)	Leakage (μW)	CT (μW)	Total Power (μW)	Δ Density (%)
AES_LFLD	100	61	77.4	115.2	1670	63.45	80	115.8	1720	+2.45
AES_LFHD	100	75	75.8	116.7	1660	78.20	79	117.6	1720	+3.20
AES_HFLD	1000	58	1048	1228	22800	59.37	1052	1238	23015	+1.37
AES_HFHD	1000	72	1036	1241	22610	73.02	1040	1252	22830	+1.02
PST_LFLD	95	53	14.13	32.05	371.3	67.33	20.71	34.75	483.4	+14.33
PST_LFHD	95	70	14.09	31.89	371.2	82.05	17.72	32.85	428.5	+12.05
PST_HFLD	950	52	34.02	325.30	3744	60.89	36.85	338.1	4022	+8.89
PST_HFHD	950	69	34.13	329.10	3785	80.26	36.96	341.5	4015	+11.26

TABLE III: Naming convention for the crypto cores regarding their frequency and placement density

Core	Frequency	Placement density	Acronym
AES	Low	Low	*AES_LFLD*
		High	*AES_LFHD*
	High	Low	*AES_HFLD*
		High	*AES_HFHD*
PST	Low	Low	*PST_LFLD*
		High	*PST_LFHD*
	High	Low	*PST_HFLD*
		High	*PST_HFHD*

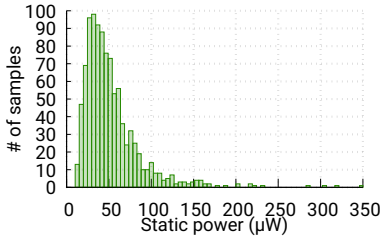

Fig. 3: PST_HFLD static power histogram, 10K MC samples (from [9]).

With the goal of obtaining a better understanding of the static power consumption of the cores, we performed a Monte Carlo (MC) simulation using Cadence Spectre. The MC simulation was performed for 10000 samples, varying only the process characteristics, with the temperature fixed to 25°C. Fig. 3 depicts the static power distribution of the PST_HFLD core. As expected, there are obvious outliers. Nevertheless, the distribution average matches the value reported in Table II for the typical corner. We will later show that the SCT is implemented in the very same region of the IC as the target, therefore we can also expect the same shifts in power due to process variation. This is an important observation: if a given die falls closer to the FF corner than to TT, it will be faster and more power hungry – but so will be the trojan, nearly at the same rate. Therefore, we argue that we can safely utilize the nominal power values reported in Table II for RO design, regardless of the quality of the fabricated silicon.

Once the power constraint has been established, the attacker can proceed to estimate the multiple operating frequencies of the RO (and the associated power values that effectively leak the key). For this goal, we have taken each of our SCTs and performed custom simulations using Cadence Spectre. The oscillation frequency and power consumption of the ROs are

TABLE IV: RO operating frequency and power consumption for four variants of AES and PST.

Target core	RO	Power & Frequency (μW & MHz) S=00	S=01	S=10	S=11
AES_LF	RO_{D6I10}	19@65	17@45	15@34	13@20
AES_HF	RO_{D10I10}	198@551	182@483	161@390	140@300
PST_LF	RO_{D6I4}	16@112	11@58	10@39	8@20
PST_HF	RO_{D8I10}	42@79	36@61	31@46	26@31

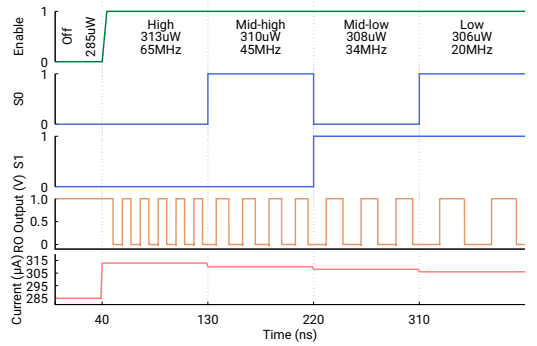

Fig. 4: Post-layout simulation of our SCT architecture in Cadence Spectre. The target design is AES_LFHD and the Trojan payload is configured as RO_{D6I10} (modified from [9]).

reported in Table IV, where each RO has been termed with a "DXIY" suffix, where X and Y represent the amount of delay and inverter cells, respectively. Notice how we do not differentiate density in the results reported in Table IV: either the trojan fits or it does not. The SCT design is nearly agnostic to placement density; this is the reason why the table contains 4 entries instead of 8.

To help the reader better visualize the operation of the SCT, the illustration in Fig. 4 displays a Spectre simulation of the SCT using the AES_LFHD target as an example. The set of leaked keys in the image is {00-01-10-11}. The results for the RO are from a SPICE-level simulation, and the total power of the AES_LFHD is estimated from physical synthesis.

We also highlight an extreme case in the RO_{D6I4} which targets the PST_LF core. For instance, both PST high frequency versions, PST_HFLD and PST_HFHD, are 2.55 and 2.6 times larger than their low frequency counterparts, respectively. Here, the SCT alone represents about 10% of the size of the PST_LF core. Since area and leakage have a linear dependency, the SCT's leakage already is about 10% of

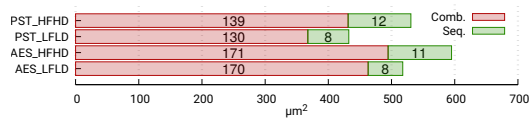

Fig. 5: Comparison of area and number of cells between SCTs.

the target's leakage. Hence, the power constraint is violated. However, this extreme example assumes the entire IC consists of a single PST core. For a large system-on-chip containing multiple cores, the power budget for designing the SCT would be much more forgiving.

Alongside the custom-simulated ROs, the SCTs are synthesized for each N_{key} and at the same clock frequency as of the target. Exclusively for the HF targets, we added the DV block to ensure the synchronous portion of the SCT does not violate timing. For AES_HF, the system clock was divided by eight while for PST_HF it was divided by sixteen. The characteristic of the SCTs are illustrated in Fig. 5, where we show the area and number of cells for each SCT.

C. SCT Insertion Results

After designing the RO and synthesizing the remainder of the SCT logic, the attacker is ready to perform the insertion via ECO. The results for insertion are described on the right side of Table II ('After SCT insertion'). For all considered scenarios, the ECO flow was capable of placing and routing the SCT successfully, even for extremely dense layouts. Considering that high cell density implies less routing resources, we verified that the ECO flow purposefully utilizes the least congested metal layers. This trend is noticeable in Table V[4], where the routing length per metal layer is reported for the PST_HFHD target. Notice the significant increase in metals M5, M6, and M7. Also note that the lower metal layers are more closely associated with the circuit performance [27], so overheads in M5 and above are unlikely to affect critical paths.

TABLE V: Routing length per metal for the PST_HFHD implementation, pre- and post-ECO.

	Wirelength (μm)	
Metal layer	pre-ECO	post-ECO
M2	5568	5759
M3	7036	8332
M4	4580	6223
M5	3182	6417
M6	2528	5283
M7	-	706

We also provide a visual comparison of the density increase for the AES_HFHD and PST_HFHD SCTs in the bottom part of Fig. 6. Note that the placement of the targets (top part of Fig. 6) was kept identical and only filler cells were removed for the SCT insertion via ECO. This is a key finding of our work and confirms the feasibility of the attack.

[4]In our considered metal stack, M1 cannot be used for signal routing. For this reason, M1 is not shown. Similarly, M8/M9 are reserved for power distribution.

Besides being able to insert the SCT, the ECO flow also has to preserve the performance of the target circuit. The additional malicious logic increases the load on the paths to which the SCT is connected, and, in general, the SCT routing could increase the coupling capacitance to adjacent paths. Thus, the impact on the target performance due to the SCT insertion is related to the number of connections between them and the increase in density. For the AES implementations, the addition of the SCT increased their total density by a small margin. On the other hand, for the PST cores, the SCT represents a large portion of the total circuit area. This is illustrated in the bottom part of Fig. 6, where the density map of the PST_HFHD and AES_HFHD layouts are shown.

The impact on the performance of the AES_HFHD and PST_HFHD cores is illustrated in Fig. 7, where we contrast the pre- and post-ECO timing slack. These results show that the impact is greater on the PST_HFHD implementation, which is explained by the high density increase reported in Table II. One can appreciate how the red bars in Fig. 7 are shifted to the left (w.r.t. the green bars). However, this shift was not sufficient to degrade the performance of the cores. In particular for the PST core, the ECO engine completed successfully by using some of the safety margin (20ps) we applied to all our paths. This safety margin is small and compatible with industry practices, where often margins in the range of tens of picoseconds are utilized. Thus, in terms of performance, our attack appears to be adequate for realistic commercial designs.

Furthermore, considering that the reported timing slack results are for implementations with a challenging operating frequency, we argue that our proposed methodology is not only capable of inserting an SCT in a high density target, but also of keeping the target performance, regardless of its frequency. Thus, for an attacker, inserting malicious logic by repurposing filler/spare cells with the help of an ECO feature is more than adequate, it is almost ideal. First, this methodology has an area overhead of 0% because we utilize space that is otherwise unused. Therefore, a slight increase in density is the only measurable "overhead". Second, the performance of the target is very likely to remain the same. Third, the runtime is only a fraction of other methods for inserting malicious logic, such as re-implementation. In the following section, we discuss these ECO characteristics in detail.

V. Testchip Design and Validation

In this section, we present our fabricated ASIC prototype and its many details. Initially, we present the chip architecture and its functionality.

A. ASIC Prototype Architecture

Our main goal while designing a silicon proof of concept for our methodology is to demonstrate the malicious potential of the ECO flow. For this purpose, we developed a full framework for performing a fabrication-type attack (see Fig. 2). Our proposed SCTs are carefully crafted in order to stress test the ECO flow and its limitations: the chosen circuits are synthesized for their maximum frequency and utilizing challenges densities, making the SCT insertion even more challenging.

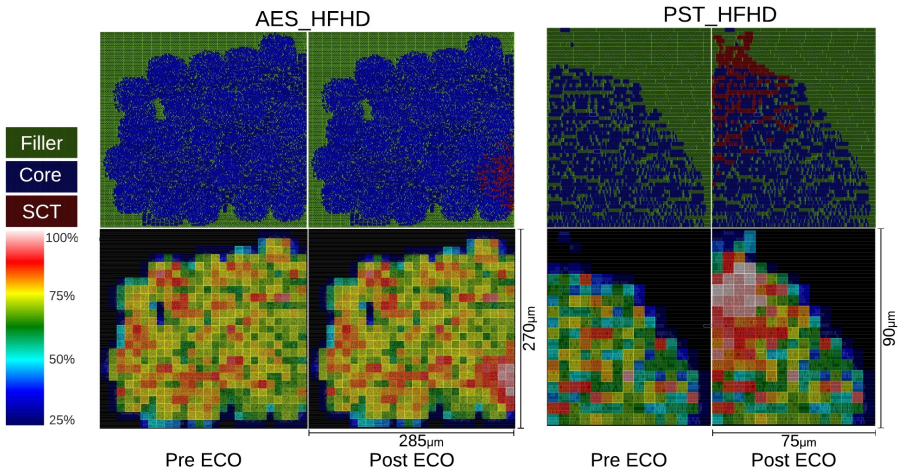

Fig. 6: Placement view (top panels) and density map (bottom panels) of the AES_HFHD and PST_HFHD cores, before and after SCT insertion via ECO (modified from [9]).

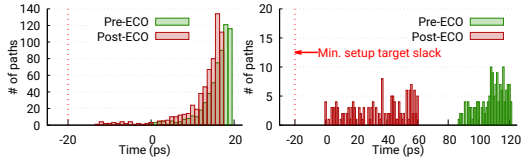

Fig. 7: Pre- and post-ECO setup timing slack comparison of AES_HFHD (right) and PST_HFHD (left) (from [9]).

Our framework includes all steps necessary for assessing the GDSII database, designing a hardware trojan, and inserting it in a finalized layout.

As discussed in the previous section, our targets are AES and PST crypto cores. For our ASIC prototype, we chose 4 of the 8 versions of the crypto cores described in Section IV. These versions are the highest density ones (AES_LFHD, AES_HFHD, PST_LFHD, and PST_HDHD), purposefully selected for the difficulty in manipulating a dense layout. The top level architecture and the floorplan of our ASIC design are depicted in Fig. 8.

Our chip contains the four chosen crypto cores and a control unit for handling the data traffic in and out of the chip. This control unit has a UART-like communication protocol and a register bank to store the plaintext, the cryptokey, and the ciphertext. We note that the plaintext and the cryptokey can be programmed externally via UART. For communicating with the control unit, the signals UART_TX and UART_RX are utilized. The signals DONE_1, DONE_2, DONE_3, and, DONE_4 indicate the end of a cryptographic operation for AES_HFHD, AES_LFHD, PST_HFHD, and PST_LFHD, respectively. These signals are exposed as primary outputs only for debug reasons, their presence is not required for the attack. Internally, these same signals are the triggers for the SCTs.

Our architecture has 5 clocks domains: the “always-on”

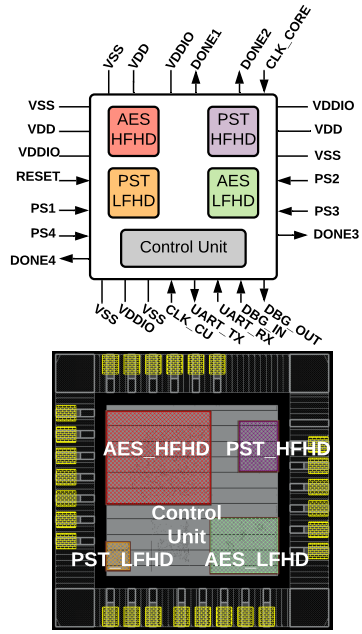

Fig. 8: Top-level diagram (top panel) and floorplan of our testchip (bottom panel).

clock is delivered by the signal CLK_CU, and the other 4 domains are connected to the signal CLK_CORE (which assumes 4 different frequencies). For switching the cores on/off selectively, the signals PS1, PS2, PS3 and, PS4 are utilized as described in Tab. VI. Both signals DBG_IN and DBG_OUT are used for debugging purposes only. The power-

ground scheme has 2 power sources (VDD and VDDIO) and a common ground (VSS): VDD supplies the core cells with a nominal voltage of 1.0V and VDDIO supplies the IO cells with a nominal voltage of 3.0V.

For manufacturing the chip, we have utilized a commercial 65nm technology (the exact same technology utilized in the previous section). We also made use of three standard cell flavors (LVT, SVT, and HVT) and power switch IP for isolating power domains. Our idea in utilizing multiple voltage threshold cells is to bring our implementations in line with industry practices, thus adding another degree of realism to our attack. The power switches were utilized to create a power domain for each crypto core, making it possible to enable one core at a time on the same chip. Implementing the crypto cores with the possibility of total shut-down is extremely valuable for evaluating our attack, because we are only reading the power that come from the enabled core. However, in our chip, the control unit is on an "always-on" domain, thus, this portion of the circuit is always enabled. Nonetheless, this characteristic later did not affect our tests or measurements in a negative manner. The power domain information and the related switch signals are described in Tab. VI.

The ideal ROs designed in the previous section (see Tab. IV) only consider the leakage from the crypto core alone. In our ASIC prototype, we have extra components that compete with the leakage of the currently enabled core – even if here we assume that only one core is on at a time. Therefore, the ROs require small adjustments to accommodate the extra competing leakage, which is a trivial exercise: an attacker can create a database of SCT architectures for known targets and apply small shifts to them by modulating the number of inverters or delay cells in the RO. The newly adjusted ROs for the ASIC prototype are described in Tab.VII. These results are from detailed SPICE-level simulations.

Our chip was designed in November 2020, fabricated at a partner foundry in March 2021, and bench tests were started in July 2021. Our bare die and its layout are contrasted in Fig. 12. For the validation of the design, we have 25 packaged samples of the chip. All packaged samples were confirmed to be 100% functional.

B. SCT Insertion

In our framework, for fully inserting the SCT into a layout, the attacker has to inspect the layout, extract the netlist, estimate the operating frequency, estimate the power consumption, modify the netlist, and finally, insert the SCT utilizing the ECO flow. The time necessary to inspect the layout in order to find the security-critical nodes depends on the expertise of the attacker, which makes it very difficult to estimate. On the other hand, the other steps can have their runtime measured. These times are depicted in red in Fig. 9 and are contrasted with the total time required for the physical implementation of the original design, shown in blue. In our case, the required time for implementing the original cores and building the top-level layout is about 5 hours and 9 minutes. The most time-intensive tasks are the clock-tree synthesis and the post-route optimizations. For our testchip, the post-route step also includes the assembling of the cores, the top-level clock-tree synthesis, the top-level routing, and a chip-level design rule check. The netlist extraction was executed in a server with an *Intel Xeon* 5122 CPU @ 3.6GHz and 96GB of RAM, while all other tasks were executed in a server with an *Intel Xeon* X5690 @ 3.47Ghz and 768GB of RAM. When multithreading was supported for a given task, the number of concurrent CPUs in use was set to 8. All execution times were measured 5 times and the presented values are the average execution time.

For the attack, the first task is extracting the netlist from the layout, which is done in 17 minutes. Then, the estimation of the operating frequency and power consumption is done in 48 minutes, which includes full parasitic extraction, static timing analysis, and a power analysis – all utilizing the highest effort available. The static timing analysis was done utilizing two corners and the power analysis only one. Differently of a chip sign-off task executed by a designer, where all corners are considered, the attacker can make use of a couple of corners only for a representative evaluation of the IC operation.

TABLE VI: Power domains, clock, average total power, and, leakage across the samples tested.

Block	Clock	Switch Signal	Leakage (μW)	Total Power (μW)
Control Unit	CLK_CU @1MHz	Always on	46.69±4.75	-
AES_HFHD	CLK_CORE @1GHz	PS1	743.79±108.07	101160±10781
AES_LFHD	CLK_CORE @100MHz	PS2	131.57±10.35	3139.32±85.38
PST_HFHD	CLK_CORE @950MHz	PS3	80.75±7.82	9661.3±758.52
PST_LFHD	CLK_CORE @95MHz	PS4	74.35±6.84	868.56±57.90

TABLE VII: RO operating frequency and power consumption for each crypto core of the ASIC prototype.

Target core	RO	Power & RO Frequency (μW & MHz) S=00	S=01	S=10	S=11
AES_LFHD	RO_{D8I14}	32@90	27@61	23@46	20@31
AES_HFHD	RO_{D12I14}	249@551	227@483	198@390	169@300
PST_LFHD	RO_{D8I6}	22@169	19@90	16@46	13@21
PST_HFHD	RO_{D10I10}	30@90	24@60	20@37	17@19

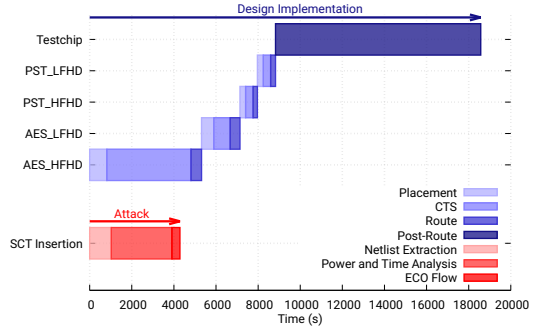

Fig. 9: Physical implementation execution time (s) for each step of the flow, and, execution time (s) for inserting the SCT in each implemented crypto core.

Inserting the SCT is done individually for each core, and the total time for inserting all four SCTs is only 6 minutes. Thus, the entire attack for this design requires 1 hours and 11 minutes. Had an attacker decided to modify the layout by re-implementing the extracted netlist, the total time of the attacks jumps to 6 hours and 21 minutes. Therefore, by using an ECO flow, not only the original design remains untouched – i.e., the cells' placement and routing remain the same after the attack – the attacker can drastically reduce the attack time. This is true for both SCTs and functional trojans.

For large ICs, how can an adversary insert a HT in a limited time frame? Or, in other words, how does the attack time scale with the complexity of the IC? To answer this question, we ought to look at the individual steps of the attack. First, for the netlist extraction, there are no alternative routes for avoiding/bypassing this specific task in the EDA tools. However, the execution time scales fairly with the size of the circuit.

On the other hand, the timing and power analysis can become overwhelming depending on the circuit size. Fortunately for the attacker, the runtime for these tasks can be reduced by utilizing different effort levels and/or by reducing the number of corners utilized. Another option is utilizing a wire-load model instead of a more precise RC model for the parasitic extraction. All of those alternatives would change the quality of the analysis, e.g., the static timing analysis in low effort with only the best-case corner will report a very optimistic timing requirement for setup (i.e., a higher operating frequency). From the attacker's point of view, an optimistic timing analysis would lead to a higher power budget, making the SCT less stealthy but more likely to succeed. An overall picture of the trade-offs between the analysis quality, the SCT insertion runtime, SCT stealthiness, and the attack success rate is provided in Tab. VIII. Here we clarify that the number of corners considered for the timing analysis is always two (BC/WC analysis). For the power analysis, only one corner is taken into account. The RC model utilized for the low effort is the wire-load model; other efforts utilize more precise models.

It must also be highlighted that the ECO insertion time does not scale with chip size: the ecoPlace and ecoRoute engines are available within a physical synthesis tool specifically for the purpose of performing local changes.

Thus, even for large ICs, the attacker can significantly reduce the time required for inserting the SCT if he/she is willing to compromise the trojan stealthiness or its success rate. But, most importantly, the time required for SCT insertion when utilizing our proposed framework is always a fraction of the time required for implementing the original design (comparatively shown in Fig. 9). This relationship does not depend entirely on the size of the circuit.

As shown on the last row of Tab. VIII, an adversary could potentially make a 'cut' in the GDSII database and treat only a part of it. To some extent, this would be the equivalent of doing a block analysis instead of chip analysis. The possibility of cutting a layout, although supported by the tools, is not studied in this work for the reason that an adversary would have to execute said cut very precisely and, after the SCT insertion, merge the modified and original layout, a task that is not trivial and could potentially damage the original layout if mishandled.

C. Tests and Measurements

For the purpose of bench testing our ASIC prototype, we have designed a custom printed circuit board (PCB). The PCB itself only contains passive components utilized for helping with the measurements and filtering noise from the power supplies. An attacker does not need our PCB and/or test setup to mount the attack. The chip is controlled by a ZedBoard from Avnet with a Xilinx Zynq-7000 All Programmable SoC. Our complete bench test setup is shown in Fig. 10, where we also make use of a 4-channel digital oscilloscope and a 2-channel power supply with an ammeter with pico ampere precision.

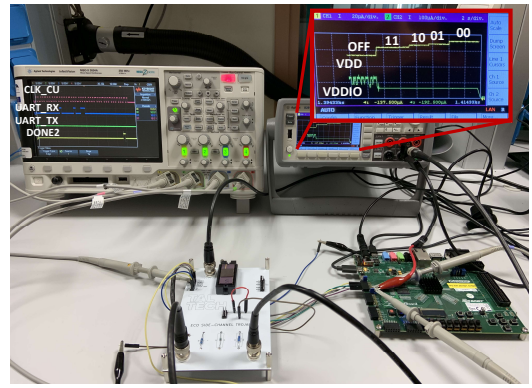

Fig. 10: Setup used for bringing up the testchip. On the left side, we show the signals used for controlling the chips. On the right side, the current consumption of the chip when the RO is active.

The first phase of the validation was to measure total power and leakage power from each block across all 25 samples. For this, all the primary inputs were set to "0", and each core was enabled one at a time by asserting its respective switch signal (see Tab. VI). For the total power, we delivered the clock signal for each block utilizing the specified operating frequency. The results of the total power average and leakage are given in Tab. VI, and its distribution across the samples is depicted in Fig 11. These results are in line with the expected from the power reports done during the physical implementation, as these results are contrasted in Fig. 11 for the worst, typical, and, best-case scenarios (SS-0.9V-0^{o}C, TT-1V-25^{o}C, FF-1.1V-125^{o}C, respectively). The corners provided by the vendor are for extreme cases, i.e., the best-case scenario is characterized at 125^o with an over voltage of 1.1V, in our case the measurements were done at room temperature and at a nominal voltage of 1.0V. Our samples are skewed towards the best case scenario, demonstrating higher average leakage. The slowest sample is near the typical case while the fastest sample is very far from the expected best case. Thus, our samples have a high variance between them (i.e., their leakage values are very spread from the mean), with variance values

TABLE VIII: Trade-off comparison between the SCT insertion runtime, SCT stealthiness, and the attack success rate, for different effort configurations of parasitic extraction, static timing analysis, and power analysis.

Cut Layout	Parasitic Extraction	Static Timing Analysis	Power Analysis	Runtime	SCT Stealthiness	Attack Success Rate
No	Low	Low	Low	Short	Weak	Medium
No	Medium	Medium	Medium	Average	Strong	Medium
No	High	High	High	Very Long	Very Strong	Very High
Yes	High	High	High	Short	Very Strong	Very High

of 1212, 102, 59, and 44 for the AES_HFHD, AES_LFHD, PST_HFHD, and PST_LFHD cores, respectively.

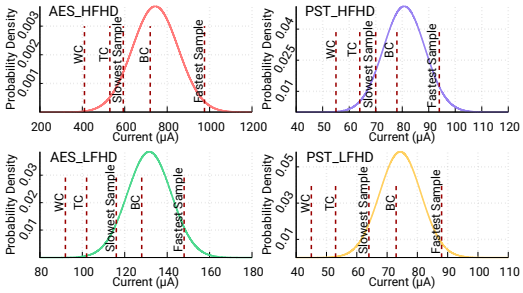

Fig. 11: Leakage distribution for each crypto core contrasted with the leakage from physical synthesis report for 3 corner cases, and, the leakage of outlier samples.

In the second phase of the experiments, we assessed the SCTs and the feasibility of the attack. This was done by the following procedure:

- A cryptokey with the 8 first bits set to "11-10-01-00" was programmed in the Control Unit's register bank
- A command for a single encryption was issued
- Right after the encryption is done, the chip asserts one of the "DONE" outputs to mark the time at which the RO starts operating
- Using only the clock signal CLK_CORE, three bursts of clocks were sent in order to shift the cryptokey connected to the RO three times
- During the whole procedure, the current consumed by the chip is monitored

An example of this procedure for the AES_LFHD core is shown in Fig. 10; the "UART_TX" signal carries the single encrypt command. As a visual aid, as soon as the "DONE" signal is asserted, the clock sources are turned off in this example (in Fig. 10 only CLK_CU is shown). As clearly depicted in the ammeter, there are discrete steps representing the leaked bits "11-10-01-00" from left to the right, respectively, as expected from the key programmed for this experiment. This was repeated 3 times for each core of each chip to confirm the behavior. The measured current values were approximated to normal distributions, as represented in Fig. 13.

By comparing the RO performance from the simulations (see Tab.VII) with ASIC measurements, it is clear that the slowest ROs are performing as expected. However, the fastest RO targeting the AES_HFHD core can only operate at a low frequency, generating a power step of about 25% of what was expected. In this case, the ECO insertion had to spread the RO cells farther away because of the lack of empty spaces nearby (see Fig. 6). For this core, the planned power steps were in the order of 200 μA, and the actual power steps after manufacturing were in the order of 60 μA. However, the attack will still enjoy a high chance of success due to the distinct separation of the power steps, even if 95% confidence intervals of the distributions almost overlap. We have also confirmed that the interconnect delay wire load was higher than we expected from SPICE-level simulations. For extreme cases such as the AES_HFHD core, the attacker has to make extra considerations for implementing a RO with high power consumption, in case he/she desires high fidelity from the RO power steps. The best-case scenario is achieved for PST_LFHD (see Fig. 13, bottom right): there is absolutely no overlapping between adjacent steps, with very low variance, which highly increases the success rate of the attack.

The experimental measurement results obtained show that the variability in the manufacturing process does not affect the effectiveness of the RO for the smaller designs (AES_LFHD, PST_LFHD, and PST_HFHD), meaning that the attack can be carried out with the same probability of success, regardless of the silicon quality for a given sample. The quality of the silicon impact on the SCT performance is directly connected with its size and how the cells were placed. If the cells are heavily spread, as occurred in the AES_HFHD, the variance of the steps is very high when compared with an SCT with a similar size (see Fig. 5) placed with low spread, as in the PST_HFHD. We hypothesize that the higher the physical spread between the cells that compose the SCT, the more susceptible to local variation they become. Visually, this can be seen by the width of the shadowed areas in Fig. 5.

However, we learned from our experimental results that even in a high spread scenario, the attack is successful for all implemented cores. For a single given die, there is no difficulty in differentiating the 4 possible leakage states associated with the two bits of the key. Moreover, these results make it evident that our SCT is successfully capable of creating distinct steps with a 2μA precision. Furthermore, the induced power consumption for the smaller crypto cores (PST_LFHD) was in the 20μW range. This makes our SCT a perfect fit for targeting designs that consume very low power while maintaining a reasonable level of stealthiness.

VI. Discussion

For a SCT insertion framework like ours, its effectiveness can be determined by three characteristics: (1) the success rate of the attack, (2) probability of detection (i.e., its stealthiness), and, (3) feasibility of the insertion of the malicious logic during the fabrication-time attack. As we have already discussed, our SCT was successful in (1) by making the attack of leaking

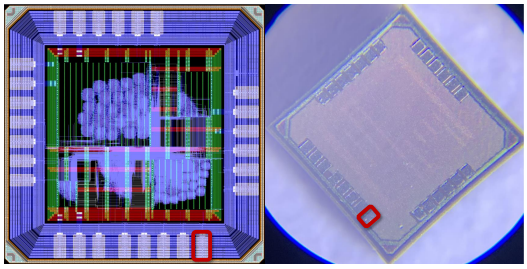

Fig. 12: Our bare die (right) and its layout (left). The lower-right corner is identified by the highlighted pin.

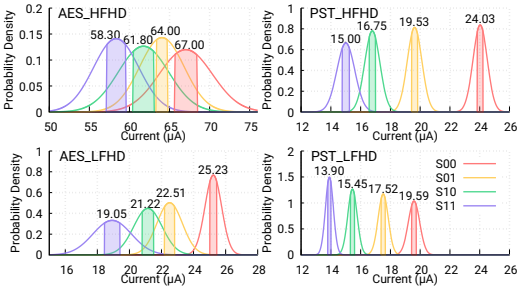

Fig. 13: Power consumption "steps" distribution for each crypto core. The shadowed area represents the 95% confidence interval.

the cryptokey viable, i.e., the attack was fully accomplished. Nevertheless, we have not yet discussed the probability of our SCT being detected.

Detecting a trojan of any kind is generally a difficult task [28]. For SCTs, any method that relies on observing corrupted bits or any degree of incorrect computation is bound to fail – SCTs do not alter the functionality of the device under attack. Because of the inherent opaqueness of ICs, inspecting their internal components is not trivial. Methods for inspecting ICs are separated into two classes, invasive and non-invasive. Invasive methods are generally done by delaminating the IC to reconstruct the layout layers, which leads to the destruction of the inspected sample. Our SCT is likely to be detected by an invasive method due to its size and amount of connected wires. However, these techniques are incredibly time consuming and also require costly and precise equipment. We emphasize that it is not a standard practice of the IC industry to perform this type of analysis.

Differently, non-invasive methods include analyzing physical characteristics of the IC, and/or, the behavior of the IO signals (i.e., timing and state) [10]. Our SCT does not disrupt any data path and our insertion methodology also does not interfere with the overall performance of the target. Thus, the probability of it being detected by analyzing the IO signals is effectively zero. Detection techniques that consider the path delay, e.g., path delay fingerprint [29], would have a low probability to detect our SCT. Nonetheless, our SCT changes the overall power consumption of the target. First, the extra leakage could raise a red flag if the IC is thoroughly inspected. The chance of being detected in this type of inspection is related with the percentage of the extra leakage from the SCT. This is also true for any HT that inserts additional logic. However, if the percentage is insignificant compared with the target, the extra leakage has a high chance to be interpreted as a skew from the manufacturing process and/or imprecision of the measurements. Second, the artificial extra consumption when the SCT is triggered can also be a red flag. In this scenario, the engineer conducting the inspection would need to know the exact moment when the SCT is triggered to suspect any alteration. Specialized detection methodologies have been proposed that utilize leakage and total power as input [30], [31]. By utilizing these advanced methods of detection, our SCT could be detected due to the trigger scheme. Since our SCT is triggered after each cryptographic operation, a periodic power fluctuation would be visible. However, the trigger scheme utilized in our silicon validation can be further modulated by an attacker by creating rarer trigger conditions, making the SCT stealthier.

In our threat model, we assumed the attacker only has access to the layout and utilizes the extracted netlist for inserting the SCT. This netlist does not contain any node names, making it impossible to distinguish nodes of interest by name. Thus, the attacker has to identify the circuit functionality by inspecting the layout for 'clues'. In the case of AES, this target circuit can be easily identified in a layout because of its implementation regularity, and, from there, the same holds true for the nets/registers that carry the cryptokey. On the other hand, visually locating a Present core (or any other core) would be a more challenging task. Nevertheless, visual inspection is not the only technique that can be utilized for searching for security-relevant nodes. In a future work, we are planning to automate the search for these nodes by submitting the netlist to a high-level functionality reconstruction tool [32]. Such tools can be used to reconstruct the finite-state machine of the target design and/or to give a score to each node in the circuit, thus distinguishing between control and data paths nodes. If a database of known cryptocores can be established in this manner, their localization would become a much simpler task.

A clear advantage of our SCT architecture is its robustness to manufacturing process skew. As demonstrated in the previous section, even with a large difference in performance between the fastest and slowest sample (see Fig. 11), the SCA was successful (see Fig. 13). When compared with a similar approach presented in [15], our SCT architecture does not need any workaround to be implemented because of the performance of the technology. Even more, our flow anticipates difficulties during the SCT insertion by including the optional clock divider. However, our architecture has the disadvantage of being large in comparison with other similar SCTs. The size of our SCT is proportional to the number of bits leaked at a time and to the total number of bits intended to be leaked (i.e., the SCT is proportional to the key length). This characteristic increases the probability of detection.

Our attack is arguably prevented by a few techniques. Split Manufacturing [4], as mentioned before, is a powerful prevention technique for HT insertion overall, where the attacker has

access only to the layer that contains the devices - the connections between them are hidden from the untrusted foundry. Hence, the attacker would only be able to find the nodes by visual inspection without any connection information, making the SCT insertion a 'blind' effort. Another relevant technique is the insertion of dummy cells and routing wires [33] with the intent to reduce the empty spaces where – potentially – a HT could be inserted. As demonstrated in this work, our insertion methodology overcomes incredibly high densities. Hence, these techniques would only be effective if the entire chip is populated with dummy cells and routing wires, increasing the design density above 95%, which for new technologies is very challenging and can potentially hinder the IC performance. On top of that, dummy cells have transistors inside them, which increases the leakage of the chip proportionally to the number of additional dummy cells. Thus, the leakage overhead could be in the range of 40-50% – assuming that a typical design has an approximate density in the range of 60-50%. For industrial designs, this type of technique might not be practical in terms of the potential performance loss, making the trade-off between security and power consumption not interesting for many vendors and applications. Therefore, the adoption of this technique as a countermeasure against malicious logic is very unlikely.

Another metric to qualify an SCT insertion attack is the total time required to perform the attack. Our threat model assumes the attack occurs in the untrusted foundry and only the layout is accessible to a rogue element. Foundries are typically working at full capacity year-round, hence, the timing window that the layout is processed to begin the manufacturing is limited. This time window is precisely the period of time in which a rogue element has to mount a fabrication-time attack. In recent SCT works that contemplate silicon validation [14], [15], [20], the techniques for inserting the malicious logic are not disclosed – making it difficult to address if the attack can be replicated in a realistic scenario.

Placing and routing an SCT manually is a time-intensive task and prone to errors, even if the HT design has only a dozen of cells. Thus, the insertion of an SCT has to be automated by utilizing an EDA tool. Inserting an SCT by re-implementing the design has a significant runtime, in the case of our testchip (see Fig. 9) it is required at least 7 hours and 18 minutes. Replicating an entire chip without the original timing and power constraints could be very difficult, which can potentially affect the target performance, thus decreasing the stealthiness of the attack. In the case of the ECO flow, the runtime for inserting the SCT in our testchip is only 1 hour and 11 minutes. Nonetheless, as previously alluded, the ECO flow has the advantage of keeping the original design untouched which increases the stealthiness of the attack. Even more, our proposed ECO flow does not require the original power and timing constraints, an estimation can be used (see Secion III-A). Consequently, our proposed ECO flow method for inserting not only SCTs, but any type of malicious logic, is arguably a **superior option**. Furthermore, the short runtime associated with the ECO flow makes the fabrication-time attack feasible in a realistic scenario, where the time window that a rogue engineer has for modifying the layout is (very) limited.

VII. Conclusions

In order to steal secret information from crypto-capable ICs, a rogue element within the foundry may insert a side-channel trojan. The SCT architecture described in this work has the advantage of not violating any design specification of the target circuit, nor is any datapath obstructed. This is all possible because of the use of an ECO flow for inserting our SCT. Since this feature is readily available, adversaries may maliciously utilize it.

Our findings and results from the validation of our ASIC prototype demonstrated the feasibility of the framework – from the layout inspection to the actual attack. The attack was successful for all 25 samples available, succesfully extracting the cryptokey via power signatures. The measurements have also demonstrated the robustness of the SCT against skews from the manufacturing process.

For our testchip, all the 4 SCTs were inserted in less than two hours. Consequently, the attack would be viable in a real fabrication-time attack. As a venue for future work, we intend to improve the search of security-relevant nodes for inserting hardware trojans in order to understand if a capable adversary is able to execute a "blind" yet successful insertion, i.e., one that does not require prior information and/or knowledge about the targeted core.

Acknowledgment

This work has been partially conducted in the project "ICT programme" which was supported by the European Union through the European Social Fund. It was also partially supported by the Estonian Research Council grant MOBERC35.

References

[1] M. Lapedus, "Big trouble at 3nm," [Online]. Available at: https://semiengineering.com/big-trouble-at-3nm/.

[2] U. Guin *et al.*, "Counterfeit integrated circuits: A rising threat in the global semiconductor supply chain," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1207–1228, 2014.

[3] S. M. Ben, "Security challenges and requirements for industrial control systems in the semiconductor manufacturing sector," 2012.

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

[5] K. Zamiri Azar, H. Mardani Kamali, H. Homayoun, and A. Sasan, "Threats on logic locking: A decade later," in *GLSVLSI '19*, p. 471–476, 2019.

[6] M. Yasin, J. Rajendran, and O. Sinanoglu, *Trustworthy Hardware Design: Combinational Logic Locking Techniques*. Springer, Cham, 2019.

[7] 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.

[8] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *2015 IEEE HOST*, pp. 137–143, 2015.

[9] T. Perez, M. Imran, P. Vaz, and S. Pagliarini, "Side-channel trojan insertion - a practical foundry-side attack via eco," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2021.

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

[11] 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.

[12] L. Lin, W. Burleson, and C. Paar, "Moles: Malicious off-chip leakage enabled by side-channels," in *2009 IEEE/ACM International Conference on Computer-Aided Design*, pp. 117–122, 2009.

[13] L. Lin *et al.*, "Trojan side-channels: Lightweight hardware trojans through side-channel engineering," in *Cryptographic Hardware and Embedded Systems - CHES 2009*, pp. 382–395, 2009.

[14] Y. Jin and Y. Makris, "Hardware trojans in wireless cryptographic ics," *IEEE Design Test of Computers*, vol. 27, no. 1, pp. 26–35, 2010.

[15] Y. Liu, Y. Jin, and Y. Makris, "Hardware trojans in wireless cryptographic ics: Silicon demonstration & detection method evaluation," in *Int. Conf. on Computer-Aided Design (ICCAD)*, pp. 399–404, 2013.

[16] R. Kumar, P. Jovanovic, W. Burleson, and I. Polian, "Parametric trojans for fault-injection attacks on cryptographic hardware," in *2014 Workshop on Fault Diagnosis and Tolerance in Cryptography*, pp. 18–28, 2014.

[17] 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.

[18] J.-F. Gallais *et al.*, "Hardware trojans for inducing or amplifying side-channel leakage of cryptographic software," in *Trusted Systems*, pp. 253–270, 2011.

[19] L. Ali and Farshad, "Analog hardware trojan design and detection in OFDM based wireless cryptographic ICs," *Plos One*, vol. 16, no. 7, p. e0254903, 2021.

[20] S. Ghandali, T. Moos, A. Moradi, and C. Paar, "Side-Channel Hardware Trojan for Provably-Secure SCA-Protected Implementations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 6, pp. 1435–1448, 2020.

[21] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Advances in Cryptology — CRYPTO' 99* (M. Wiener, ed.), pp. 388–397, 1999.

[22] 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.

[23] A. Jain, Z. Zhou, and U. Guin, "Survey of recent developments for hardware trojan detection," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2021.

[24] T. Trippel *et al.*, "ICAS: An Extensible Framework for Estimating the Susceptibility of IC Layouts to Additive Trojans," *2020 IEEE Symposium on Security and Privacy (SP)*, pp. 1078–1095, 2020.

[25] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *Cryptographic Hardware and Embedded Systems - CHES 2007* (P. Paillier and I. Verbauwhede, eds.), (Berlin, Heidelberg), pp. 450–466, Springer Berlin Heidelberg, 2007.

[26] S. Ghandali *et al.*, "Side-channel hardware trojan for provably-secure sca-protected implementations," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 28, no. 6, pp. 1435–1448, 2020.

[27] S. N. Pagliarini, M. M. Isgenc, M. G. A. Martins, and L. Pileggi, "Application and product-volume-specific customization of beol metal pitch," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 26, no. 9, pp. 1627–1636, 2018.

[28] S. Bhasin and F. Regazzoni, "A survey on hardware trojan detection techniques," in *2015 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2021–2024, 2015.

[29] M. Li, B. Yu, Y. Lin, X. Xu, W. Li, and D. Z. Pan, "A practical split manufacturing framework for trojan prevention via simultaneous wire lifting and cell insertion," in *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 265–270, 2018.

[30] Y. Liu, Y. Jin, A. Nosratinia, and Y. Makris, "Silicon demonstration of hardware trojan design and detection in wireless cryptographic ics," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 4, pp. 1506–1519, 2017.

[31] R. M. Rad, X. Wang, M. Tehranipoor, and J. Plusquellic, "Power supply signal calibration techniques for improving detection resolution to hardware trojans," in *2008 IEEE/ACM International Conference on Computer-Aided Design*, pp. 632–639, 2008.

[32] T. Meade, S. Zhang, and Y. Jin, "Netlist reverse engineering for high-level functionality reconstruction," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, pp. 655–660, 2016.

[33] 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.

Tiago D. Perez received the M.S. degree in electric engineering from the University of Campinas, São Paulo, Brazil, in 2019. He is currently pursuing a Ph.D. degree at 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 fields of work include digital signal processing, telecommunication systems and IC implementation. His current research interests include the study of hardware security from the point of view of digital circuit design and IC implementation.

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 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 hardware trustworthiness. His research on hardware security has been funded by DARPA, IARPA, the European Commission, the AFRL, and the Estonian Research Council.

Appendix 7

[VII]
Z. U. Abideen, T. D. Perez, and S. Pagliarini, "From fpgas to obfuscated easics: Design and security trade-offs," in 2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST), pp. 1–4, 2021

From FPGAs to Obfuscated eASICs: Design and Security Trade-offs

Zain Ul Abideen, Tiago Diadami Perez, Samuel Pagliarini
Centre for Hardware Security, Tallinn University of Technology (TalTech), Estonia
{zain.abideen, tiago.perez, samuel.pagliarini}@taltech.ee

***Abstract*—Threats associated with the untrusted fabrication of integrated circuits (ICs) are numerous: piracy, overproduction, reverse engineering, hardware trojans, etc. The use of reconfigurable elements (i.e., look-up tables as in FPGAs) is a known obfuscation technique. In the extreme case, when the circuit is entirely implemented as an FPGA, no information is revealed to the adversary but at a high cost in area, power, and performance. In the opposite extreme, when the same circuit is implemented as an ASIC, best-in-class performance is obtained but security is compromised. This paper investigates an intermediate solution between these two. Our results are supported by a custom CAD tool that explores this FPGA-ASIC design space and enables a standard-cell based physical synthesis flow that is flexible and compatible with current design practices. The results after physical implementation are generated for the obfuscated circuits in a 65nm commercial technology, demonstrating the attained obfuscation quantitatively. Furthermore, our security analysis revealed that for truly hiding the circuit's intent (not only portions of its structure), the obfuscated design also has to chiefly resemble an FPGA: only some small amount of logic can be made static for an adversary to remain unaware of what the circuit does.**

***Index Terms*—Hardware Obfuscation, Secure ASIC Design, CAD, Reconfigurable obfuscation, Reverse engineering**

I. Introduction

Shipment of semiconductor devices is forecast to surpass one trillion units in the year 2021, the third time this mark is surpassed in a calendar year since 2018 [1]. The majority of those devices are being manufactured by foundries that subscribe to the fab-for-hire model. Many potential threats regarding third-party foundries have been studied in recent years, include tampering, counterfeiting, reverse engineering, and overproduction. On the other hand, many techniques have been devised to mitigate threats from untrusted fabrication. Countermeasure techniques to increase the IC security against not only third-party foundries but also from the end-user have been recently demonstrated. Notable examples include IC Camouflaging [2]–[4], Logic Locking [5]–[7], and Split Manufacturing [8], [9].

Generally speaking, all of the aforementioned countermeasures attempt to "hide" the design from adversaries and can be classified as obfuscation techniques. Unfortunately, none of these techniques is currently adopted in large-scale production of ICs, for reasons that include (lack of) practicality [8] and insufficient security guarantees [10]. Another approach towards obfuscation is the use of an FPGA (or FPGA-like) design, where the configuration *bitstream serves as a key* to unlock the functionality of the circuit [11]. Our paper too explores this possibility. The fabric in an FPGA contains reconfigurable elements, but this flexibility incurs a limited performance. On the other hand, ASIC requires one-time placement, it is static (non-reconfigurable), but it provides best-in-class performance. As shown in Fig. 1, performance increases if we move from right to left. Contrarily, area, obfuscation, and flexibility increase if we move from left to right.

This work has been partially conducted in the project "ICT programme" which was supported by the European Union through the European Social Fund. It was also partially supported by the Estonian Research Council grant MOBERC35.

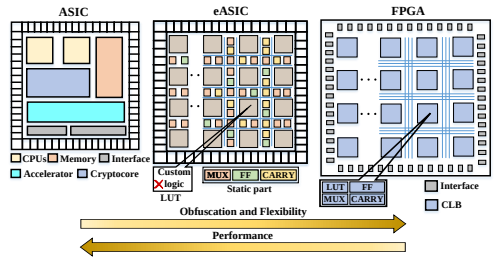

Fig. 1: The design obfuscation landscape, from ASICs to FPGAs. The relative sizes are notional.

Design obfuscation concept: In this work, we propose to obfuscate a design by exploiting the best of both worlds. The generated device is a hybrid which includes *reconfigurable elements* (analogous to the FPGA) and also includes ASIC cells as *static elements*, i.e., gates with fixed functionality after fabrication. Previous research on obfuscation by reconfigurable elements has focused on keeping the reconfigurable portion as small as possible [12], [13], which is logical if the goal is to keep overheads under control. However, we later show that true hiding of the circuit's intent requires a *high degree of obfuscation* that is usually not explored in the state of the art. We term our in-between solution an "embedded ASIC" (**eASIC**). Thus, our eASIC device is largely non-functional until it is programmed. Our main contribution is a tool for automatically obfuscating a design in the form of eASIC, where the obfuscation range can be from 0 to 100%. Furthermore, its physical synthesis flow is standard-cell based that is compatible with current design and fabrication practices.

II. A CAD flow for eASIC

Our CAD flow is centered around a tool named **T**uneable Design **O**bfuscation **T**echnique using **e**ASIC, or **TOTe** for short. This section explains the CAD flow of eASIC and TOTe's main features. TOTe generates a hybrid design with static and reconfigurable elements, which we refer to as eASIC. For the reconfigurable elements, we implement the logic utilizing the notion of programmable LUTs (Look Up Tables) - same as in FPGAs. The complete TOTe design flow for generating an eASIC is shown in Fig. 2 and it consists of three phases. In the **first phase** of our flow, the design under obfuscation, described in register-transfer level (RTL) form, is synthesized using a commercial FPGA synthesis tool. As a result, the netlist contains all typical FPGA primitives, i.e., FFs, MUXs, and LUTs. The input design requires no special annotations, synthesis pragmas, or any other change in its representation.

Next, in the **second phase**, TOTe requires the ASIC standard cell library of choice. As highlighted in the center of Fig. 2, the core idea of TOTe is to **replace reconfigurable logic for static logic**. For TOTe, only the LUTs are treated as reconfigurable logic, and, any other primitives from the FPGA synthesis are automatically transformed into static logic. TOTe utilizes its own **obfuscation and**

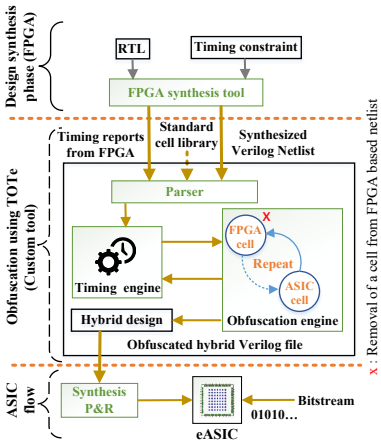

Fig. 2: The CAD flow for eASIC, combining FPGA/ASIC synthesis.

timing engines. These engines drive the security vs. performance trade-offs of the tool. For this phase, the designer provides an obfuscation target in terms of percentage, which determines the portion of logic that will remain reconfigurable as LUTs. TOTe builds a tree representation of the circuit where primitive types are annotated for every instance. For LUTs, in particular, the tool also annotates their masking patterns (i.e., the portion of the bitstream associated with an individual LUT). By using truth tables populated by the masking patterns, TOTe builds combinational logic that is equivalent to the LUT's intended usage (static logic). Finally, TOTe exports an obfuscated hybrid Verilog file (eASIC), timing report, and area report. A designer can repeat this procedure until he achieves his obfuscation and performance targets.

In the **third and final phase**, the obfuscated netlist from TOTe is synthesized using any commercial ASIC CAD tool and implemented using an also commercial tool where traditional P&R, CTS, DRC, LVS, etc. steps are executed. Finally, the tapeout database is sent to the foundry for fabrication.

III. Experimental Results using TOTe

This section reports the analysis of security versus performance, security versus area for selected designs and reports the results for numerous designs after obfuscation. For all experimental results that follow, FPGA synthesis was executed in Vivado and the targeted device is Kintex-7 XC7K325T-2FFG900C, which contains only 6-input LUTs. For the ASIC flow, the implementations are done using a commercial 65nm PDK with three standard cell flavors (LVT/SVT/HVT) and tools from Cadence (i.e., Genus and Innovus).

Custom standard-cell based LUTs: The premise of eASIC is to have reconfigurable and static elements that can be integrated transparently. For this reason, we have designed our own custom LUTs (LUT_1, LUT_2,..., LUT_6) by following VPR's template [14]. Different from FPGAs that generally implement only one LUT size, for eASIC we have the flexibility to implement more than one size because our design intent will not change. By doing this, we preserve area and potentially increase the performance of eASIC. These blocks are highly compact since the main design goal for them was area/density. Each LUT has its own registers for storing the configuration, a functionality that is enabled by including three extra configuration pins ($serial\_in$, $serial\_out$, and $enable$). The LUTs are connected to one another in a daisy chain that is analogous to a scan chain.

TABLE I: Detailed results for selected designs using TOTe

Design	Obf. (%)	*sumCP* (ns)	*CP* (ns)	Area-RE (μm^2)	Area-ST (μm^2)	LUT (RE)	LUT (ST)
SBM	98	16088.690	0.490	13190.04	0	29	0
	95	15895.826	0.484	12762.00	21.40	28	1
	92	15877.962	0.464	12547.80	32.11	27	2
	89	15458.506	0.461	12438.72	37.56	26	3
	86	15370.682	0.459	12224.52	48.27	25	4
SHA-256	98	7425.731	0.962	1313150.76	10291.86	2195	44
	95	7354.593	0.871	1275984.00	28875.24	2128	111
	92	7322.155	0.871	1233448.56	50142.96	2060	179
	89	7301.945	0.871	1179674.64	77029.92	1992	246
	86	7164.025	0.871	1125799.56	103967.46	1925	313
FPU	98	2909.063	0.707	1031676.84	1250.028	2487	50
	95	2734.008	0.650	1003225.68	2672.586	2412	126
	92	2572.952	0.650	966715.20	4498.11	2336	202
	89	2478.732	0.650	935060.04	6080.868	2259	279
	86	2410.211	0.650	893005.56	8183.592	2183	355

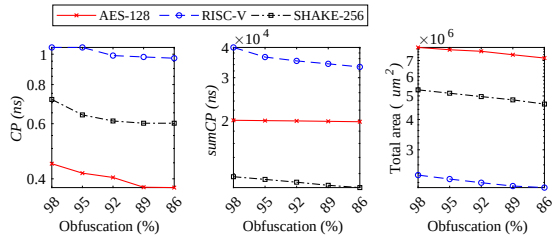

Fig. 3: Obfuscation results for AES-128, RISC-V and SHAKE-256.

Design Space Exploration in TOTe: For our first experiment, we selected a small but representative design which covers all possible FPGA primitives: a schoolbook multiplier (SBM), which is a bit-serial polynomial multiplication circuit. For a SBM design that is synthesized targeting a very high frequency, the CP and $sumCP$ become, as calculated by TOTe, 0.490 ns and 16088.69 ns, respectively. These values correspond to a design obfuscated at 100%, i.e., the design has only reconfigurable logic. The absolute accuracy of these values is not relevant since final timing analysis is performed using a commercial physical synthesis engine later.

While the SBM design is an interesting motivational example, it showed that CP tends to saturate while the $sumCP$ continues to improve as the obfuscation is reduced. Next, we wanted to determine if the same saturation trend appears for other designs and the results are reported in Table I. From these experiments, it is possible to conclude that the performance of numerous designs saturates incredibly fast as we decrease the obfuscation level, even when the obfuscation range is limited to 86-100%. Moreover, the results for AES-128, RISC-V, and SHAKE-256 have been depicted in Fig. 3. Several other designs, including ISCAS'85 benchmarks and known opencores, have been evaluated. The complete set of results can be found in our git repository [15].

IV. Security Analysis

As compared to conventional logic locking, the LUTs introduced in eASIC are the key-gate equivalents. In principle, a single LUT_n ought to be equivalent to 2^n XOR/XNOR key-gates. In practice, the LUT logic has similarities to a run of key-gates (see [6]) due to the n-to-1 multiplexing nature of it, which reduces the search space and may possibly make eASIC vulnerable to well-known oracle-based attacks

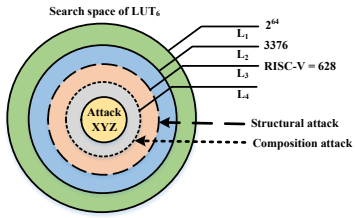

Fig. 4: The search space of LUT_6 as it shrinks with different attacks.

(e.g., SAT). However, notice that we are considering designs with target obfuscation rates higher than 86%, which results in bitstreams with thousands of bits. Even for a small and combinational design as the ISCAS'85 c7552, 50% of obfuscation requires a bitstream with approximately 11k bits. The SAT attack is not able to find the correct key, even running for more than 60 hours. We make the following assumptions to build a threat model:

- The adversary goal is to identify the circuit intent, even in the presence of obfuscation. For this goal, the adversary **does not need** to recreate the bistream.
- The adversary has access to the GDSII file of the eASIC design. He or she is skilled in IC design and has no difficulty in understanding this layout representation.
- The attacker can recognize the standard cells, thus the gate-level netlist of the obfuscated circuit can be easily recovered [16].
- We assume that the attacker can differentiate between design inputs and reconfiguration pins [10], [17].
- We assume the adversary can group the standard cells present in the static logic and convert them back into reconfigurable logic (i.e., LUT representation)[1].

In order to evaluate the security hardness of eASIC, we propose two different attacks: one based on the *structure* of design and another based on the *composition* of known different circuits.

Structural Analysis Attack: The goal of this attack is to decrease the key search space and attempt to recover the bitstream. As we mentioned before, the key search space is 2^{64} for a single LUT_6. But this assumption only holds if the FPGA synthesis is actually capable of exercising the entire key search space, which our results reveal that is far from possible. We have synthesized a large number of representative designs (>30) and counted how many unique LUT_6 masking patterns appear in the corresponding netlists. Designs of varied complexity, size, and functionality where added until the combined number of unique masking patterns appears to settle, forming a set of $m = 3376$ elements. This result alone, albeit being empirical, reduces the global search space from L_1 to L_2 as illustrated in Fig. 4.

We utilize tuples of $\langle pattern, frequency \rangle$ for tracking how often masking patterns repeat. The tuples are referenced by integer identifiers and ordered by frequency. Our analysis reveals that the RISC-V netlist has 628 unique LUTs and only 3 occur more than 100 times.In practice, if the attacker could know for a fact that the obfuscated circuit is indeed RISC-V, the search space would shrink further. The shrunk search spaces are labeled L_3 in Fig. 4. The question then becomes whether the static portion of the circuit is large enough for the adversary to be confident that the circuit under attack can be labelled as circuit C_1, C_2, or C_n. We investigate this possibility by

[1]This is a very generous concession since the static logic is repeatedly optimized during logic and physical synthesis.

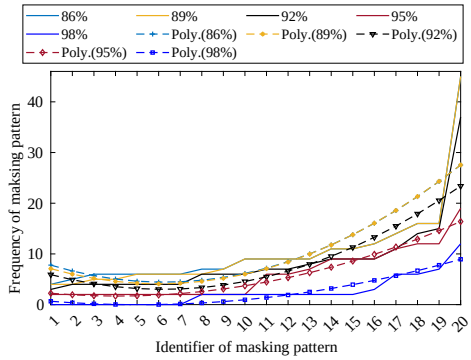

Fig. 5: The structural analysis of MIPS and RISC-V.

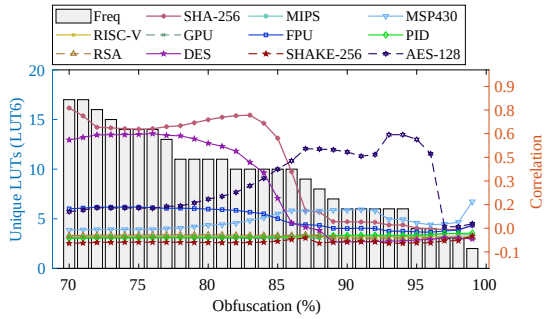

Fig. 6: The correlation of SHA-256 versus numerous other designs.

further analysing the behaviour of the frequency of masking patterns, as depicted in Fig. 5. For this, we utilized polynomial trendlines for a portion of identifier of masking pattern, considering netlists generated by TOTe at 98%, 95%, 92%, 89%, and 86% obfuscation levels. It is noteworthy that the trendlines become better frequency predictors as the obfuscation level is decreased. For RISC-V, in particular, the adversary can guess a small number outliers and the best guess (when obfuscation is 86%) is far from the original frequencies (>100).

Composition Analysis Attack: The goal of this attack is to identify the circuit by correlation to known circuits. This attack also exploits the frequency of the LUT_6, but here we correlate entire designs (instead of pattern-frequency tuples) based on their composition. We consider that the attack is successful if the adversary is able to identify the circuit (see threat model, 1st bullet). In this experiment, we performed correlation analysis for the well-known SHA-256 crypto core as shown in Fig. 6. The objective of this experiment is to analyze the leaked information from the static part of an obfuscated design against a database[2] of circuits. We have obfuscated SHA-256 in the 70-100% range and then correlated the static portion of the design with the database of known circuits. In Fig. 6, we show the results where the x-axis shows the obfuscation percentage and the y-axis shows correlation (right) and number of unique LUTs (left). For this circuit, three regions of interest can be defined: 97-100% (no correlation), 86-96% (strong correlation to another circuit), and 70-85% (correlation to itself). This attack reveals that if the adversary goal is solely to identify the circuit's intent,

[2]We assume the adversary can obtain samples of open source cores from repositories and execute FPGA synthesis on them with his tool of choice.

TABLE II: Results for the implementation of SHA-256 for different obfuscation levels

CAD Flow	Obf. (%)	Density	Area (μm^2)	Freq. (MHz)	T. Power (mW)	# LUT	Comb.	Seq.
FPGA	100	–	–	77	191	2238	–	1830
TOTe	100	46%	1412227	166.7	274.24	2238	82756	105128
TOTe	90	45%	1274690	178.6	262.21	2015	83452	94876
TOTe	85	46%	1215328	200	277.82	1904	79626	90420
TOTe	80	54%	1135752	200	262.77	1792	74000	83790
TOTe	0	71%	43097	200	6.93	0	3165	1806
ASIC	0	71%	60563	769	33.55	0	3165	1806

eASIC can be as vulnerable as an ASIC design. To mitigate this undesirable effect, obfuscation levels should remain relatively high. Otherwise, if the obfuscation lies between 70 and 84%, the search space would shift from L_3 to L_4 as shown in Fig. 4.

V. Physical Synthesis for eASIC

This section contains the physical implementation results for an obfuscated SHA-256 core. We have selected SHA-256 as it is popular and widely used in cryptography. The variants of the design with different obfuscation levels are implemented with the aid of the LUTs defined in Section III. The results obtained after implementation are focused on performance vs. area trade-offs for the 80-100% obfuscation range as determined by the security analysis of Fig. 6. Initially, we synthesized and implemented the SHA-256 core on FPGA. This implementation achieves a frequency of only 77 MHz (for reference, the Kintex-7 family is produced on a 28nm CMOS technology). To start the analysis, we select 100% obfuscation as a baseline design because it is fully reconfigurable and somewhat analogous to an FPGA design.

The implementation results for 0%, 80%, 85%, 90%, and 100% obfuscation are listed in Table II, obtained after physical synthesis and are for the worst process corner (SS) and a nominal temperature of 25°C. It is noteworthy that the performance of the design is increasing as we decrease the level of security. This behaviour is clearly depicted in the fourth column of Table II and matches the goal we set from the beginning: to trade performance for security. Here we also show that performance saturates rather quickly, as predicted by TOTe in Section III. The area of the design is proportional to the obfuscation level which means that increasing the security of design will cause area overhead. The results obtained from the physical synthesis justify trade-offs and Table II show the resource requirements.

VI. Comparison and discussion

From the many results, we conclude that obfuscation levels should be relatively high to achieve a considerable security, thus the majority of the eASIC logic should be reconfigurable logic (i.e., LUTs). Having a large portion of reconfigurable logic provides an opportunity to correct the issues/bugs that could be easily fixed during the reconfiguration phase. Naturally, there are limitations since a portion of the system consists of static logic and cannot be modified. This limitation could be eased if the eASIC layout were to include spare LUTs. To some degree, those spare LUTs could also be used to make side-channel attacks less successful.

A recent trend in obfuscation research is the use of embedded FPGA (eFPGA) [18], [19]. A very similar approach is also found in [20], where authors perform obfuscation with transistor-level granularity. While there are advantages to this practice, it has been used selectively to only protect key portions of a design and therefore keep the performance penalty as low as possible. The challenge is in determining which portions of the circuit merit protection and which ones do not. Our eASIC approach bypasses this question almost completely by only revealing (portions of) critical paths when they are selected to become static logic, which we consider an advantage if the ASIC-equivalent performance can be sacrificed.

VII. Conclusions

In this paper, we have developed a custom tool (TOTe) that obfuscates a design and transforms it into an eASIC device. Our eASIC solution contrasts with the current practice of eFPGA for obfuscation and this is not by coincidence: our experimental results show that obfuscation rates have to be high to protect not only the bitstream but also the design's intent. This is a key finding of our research which we hope can help to steer current obfuscation practices in the literature. Our findings are also validated in a commercial physical synthesis tool with industry-strength timing and power analysis, from which we confirm that TOTe's trade-off analysis is sufficiently accurate.

References

[1] IC Insights, "Semiconductor units forecast to exceed 1 trillion devices in 2021," [Online]. Available at: https://www.icinsights.com/news/bulletins/Semiconductor-Units-Forecast-To-Exceed-1-Trillion-Devices-Again-In-2021/.

[2] M. Yasin *et al.*, "Removal attacks on logic locking and camouflaging techniques," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 517–532, 2020.

[3] R. P. Cocchi *et al.*, "Circuit camouflage integration for hardware ip protection," in *DAC*, 2014, pp. 1–5.

[4] M. Li *et al.*, "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.

[5] K. Zamiri Azar *et al.*, "Threats on logic locking: A decade later," in *GLSVLSI '19*, 2019, p. 471–476.

[6] M. Yasin *et al.*, "On improving the security of logic locking," *IEEE TCAD*, vol. 35, no. 9, pp. 1411–1424, 2016.

[7] J. Sweeney *et al.*, "Latch-based logic locking," in *2020 IEEE HOST*, 2020, pp. 132–141.

[8] T. D. Perez *et al.*, "A survey on split manufacturing: Attacks, defenses, and challenges," *IEEE Access*, vol. 8, pp. 184 013–184 035, 2020.

[9] J. Rajendran *et al.*, "Is split manufacturing secure?" in *2013 DATE*, 2013, pp. 1259–1264.

[10] P. Subramanyan *et al.*, "Evaluating the security of logic encryption algorithms," in *2015 IEEE HOST*, 2015, pp. 137–143.

[11] B. Liu *et al.*, "Embedded reconfigurable logic for asic design obfuscation against supply chain attacks," in *DATE*, 2014, pp. 1–6.

[12] H. Mardani Kamali *et al.*, "Lut-lock: A novel lut-based logic obfuscation for fpga-bitstream and asic-hardware protection," in *2018 IEEE ISVLSI*, 2018, pp. 405–410.

[13] S. D. Chowdhury *et al.*, "Enhancing sat-attack resiliency and cost-effectiveness of reconfigurable-logic-based circuit obfuscation," in *2021 IEEE ISCAS*, 2021, pp. 1–5.

[14] K. E. Murray *et al.*, "Vtr 8: High-performance cad and customizable fpga architecture modelling," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 13, no. 2, 2020.

[15] Z. U. Abideen *et al.*, "TOTe (Tuneable Design Obfuscation Technique using eASIC)," 2021. [Online]. Available: https://github.com/Centre-for-Hardware-Security/eASIC

[16] R. Torrance *et al.*, "The state-of-the-art in ic reverse engineering," in *CHES 2009*, C. Clavier *et al.*, Eds., 2009, pp. 363–381.

[17] M. Yasin *et al.*, "Provably-secure logic locking: From theory to practice," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, p. 1601–1618.

[18] B. Hu *et al.*, "Functional obfuscation of hardware accelerators through selective partial design extraction onto an embedded fpga," in *GLSVLSI '19*, 2019, p. 171–176.

[19] J. Chen *et al.*, "DECOY: DEflection-Driven HLS-Based Computation Partitioning for Obfuscating Intellectual PropertY," in *2020 IEEE DAC*, ser. DAC '20. IEEE Press, 2020.

[20] M. M. Shihab *et al.*, "Design obfuscation through selective post-fabrication transistor-level programming," in *DATE*, 2019, pp. 528–533.

Curriculum Vitae

1. Personal data

Name	Tiago D. Perez
Date and place of birth	24 September 1990 São Paulo, Brazil
Nationality	Brazillian

2. Contact information

Address	Tallinn University of Technology, School of of Information Technologies, Department of Computer Systems, Ehitajate tee 5, 19086 Tallinn, Estonia
Phone	+372 5393 1682
E-mail	tiago.perez@taltech.ee

3. Education

2020–present	Tallinn University of Technology, School of Information, Information and Communication Technology, PhD studies
2017–2019	University of Campinas, Faculty of Electrical and Electronic Engineering, Electrical Engineering, MSc
2010–2014	University of São Paulo, Faculty of Electrical and Computer Engineering, Electrical Engineering, BSc

4. Language competence

Portuguese	native
English	fluent

5. Professional employment

2020–present	Tallinn University of Technology, Early Researcher
2019–2020	NXP, Digital Design Engineer
2014–2019	Institute Eldorado, Digital Design Engineer

6. Computer skills

- Operating systems: GNU/Linux and Windows
- Document preparation: Emacs and LaTeX
- Programming languages: C/C++, C# and Python
- Hardware description languages: VHDL, Verilog and System Verilog

7. Honours and awards

- 2022, 3rd Place Award, Security Closure of Physical Layouts Design Contest, International Symposium on Physical Design (ISPD).

8. Defended theses

- 2019, Study on performance improvements of digital satellite receivers in the presence of multipath channel using decision feedback equalization, MSc, Prof. Dr. Luís Geraldo P. Meloni University of Campinas, Institute of Electical Engineering and Telecommunications
- 2014, Development of a control system for an autonomous surface unmanned vehicle, Prof. Dr. José Roberto B. A. Monteiro. Universty of São Paulo, Institute of Electrical and Electronic Engineering

9. Field of research

- Hardware security
- Hardware trojans
- Integrated circuits design

Elulookirjeldus

1. Isikuandmed

Nimi	Tiago Diadami Perez
Sünniaeg ja -koht	24.09.1990, Guarulhos, Brasiilia
Kodakondsus	Brasiilia

2. Kontaktandmed

Aadress	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Arvutisüsteemide instituut, Ehitajate tee 5, 19086 Tallinn, Estonia
Telefon	+372 5393 1682
E-post	tiago.perez@taltech.ee

3. Haridus

2020–present	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Arvutisüsteemid, doktoriõpe
2017–2019	University of Campinas, Faculty of Electrical and Electronic Engineering, Electrical Engineering, magistratuur
2010–2014	University of São Paulo, Faculty of Electrical and Computer Engineering, Electrical Engineering, bakalaureus

4. Keelteoskus

portugali keel	emakeel
inglise keel	kõrgtase

5. Teenistuskäik

2020–present	Tallinn University of Technology, Early Researcher
2019–2020	NXP, Digital Design Engineer
2014–2019	Institute Eldorado, Digital Design Engineer

6. Arvutialased oskused

- Operating systems: GNU/Linux and Windows
- Document preparation: Emacs and LaTeX
- Programming languages: C/C++, C# and Python
- Hardware description languages: VHDL, Verilog and System Verilog

7. Autasud

- 2022, 3rd Place Award, Security Closure of Physical Layouts Design Contest, International Symposium on Physical Design (ISPD).

8. Kaitstud lõputööd

- 2019, Study on performance improvements of digital satellite receivers in the presence of multipath channel using decision feedback equalization, MSc, Prof. Dr. Luís Geraldo P. Meloni University of Campinas, Institute of Electical Engineering and Telecommunications
- 2014, Development of a control system for an autonomous surface unmanned vehicle, Prof. Dr. José Roberto B. A. Monteiro. Universty of São Paulo, Institute of Electrical and Electronic Engineering

9. Teadustöö põhisuunad

- Hardware security
- Hardware trojans
- Integrated circuits design

ISSN 2585-6901 (PDF)
ISBN 978-9949-83-948-3 (PDF)