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

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

European Union
European Regional
Development Fund

Investing
in your future

Copyright: Tiago D. Perez, 2023
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

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].

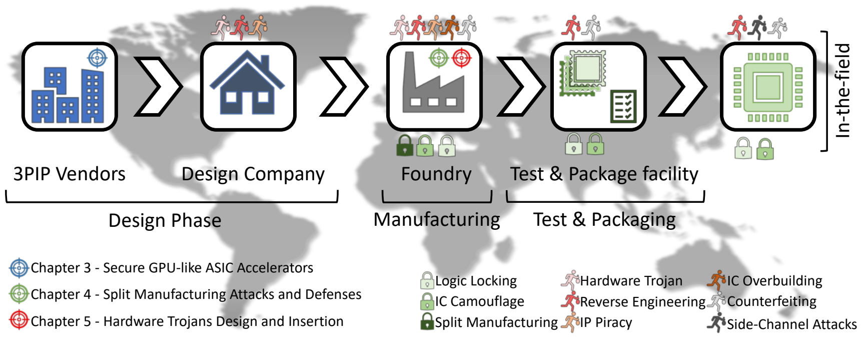


Figure 1: Detailed IC's life cycle phases, possible attacks, and defenses.

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.

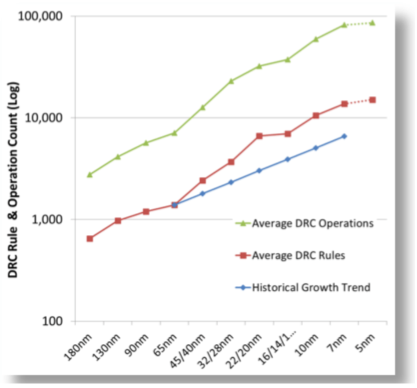


Figure 2: Growth of design rules from CMOS 180nm until finFET 5nm (from [43]).

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].

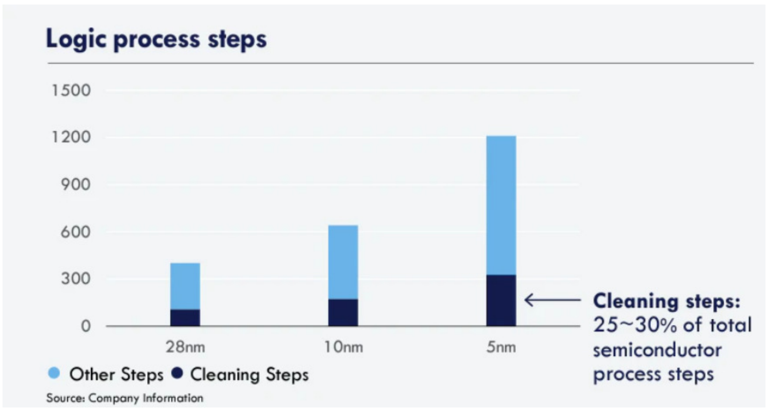


Figure 3: Logic manufacturing process steps comparison between CMOS 28nm, FinFET 10nm, and, FinFET 5nm, technology nodes (from [45]).

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

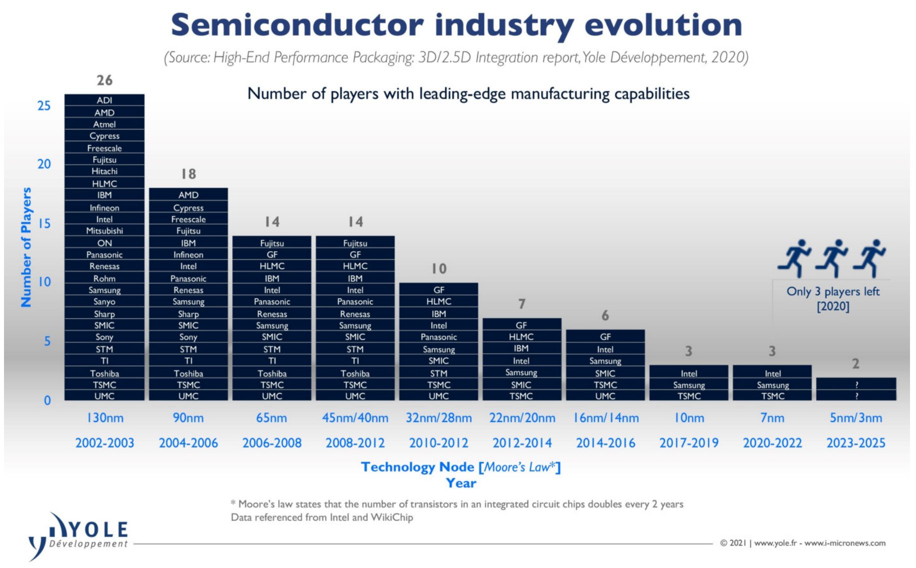


Figure 4: Semiconductor industry evolution (from [47]).

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

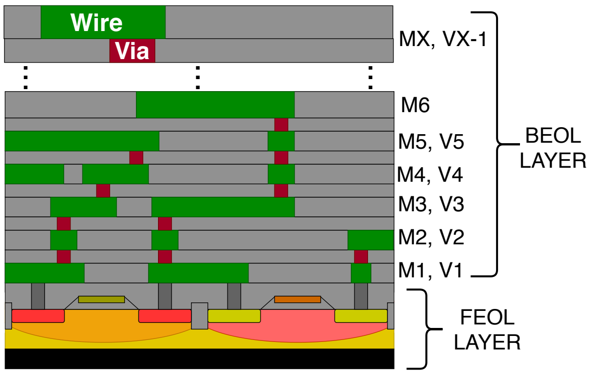


Figure 5: Cross section of an Integrated Circuit (from [22]).

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.

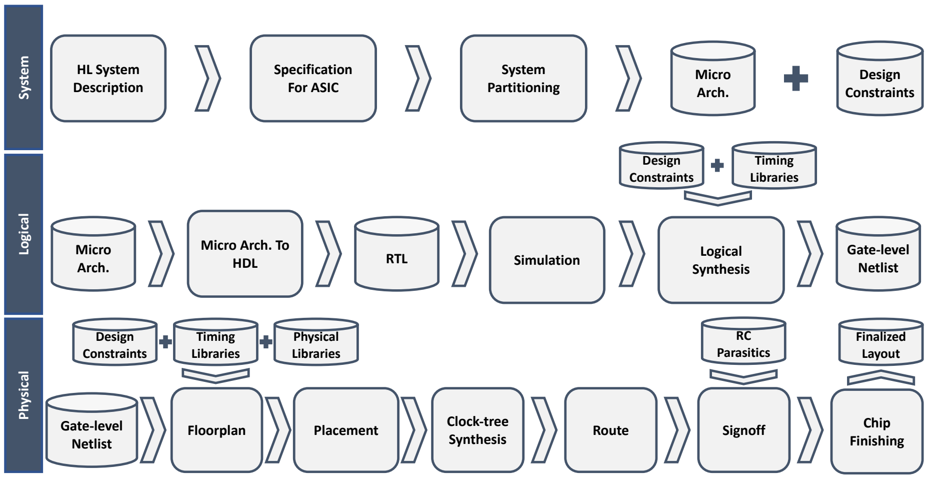
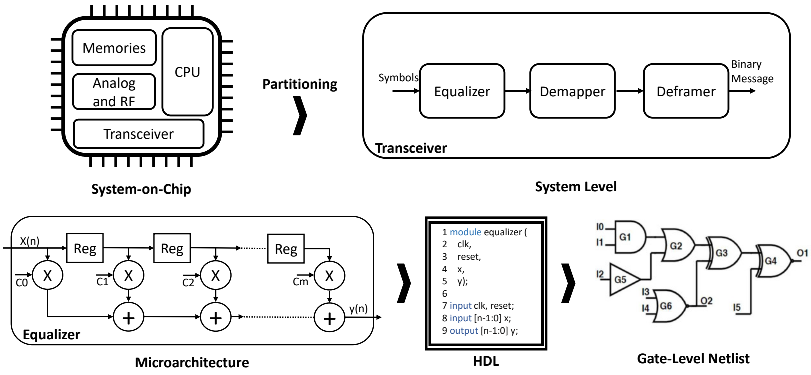


Figure 6: Typical design flow for digital integrated circuits.

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.



```
module equalizer (
   clk,
   reset,
   x,
   y);

input clk, reset;
input [n-1:0] x;
output [n-1:0] y;
```

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.

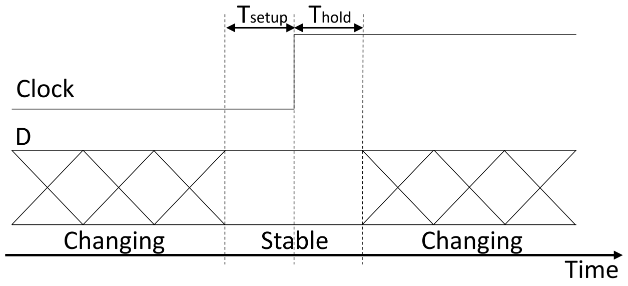


Figure 8: Setup and hold time.

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.

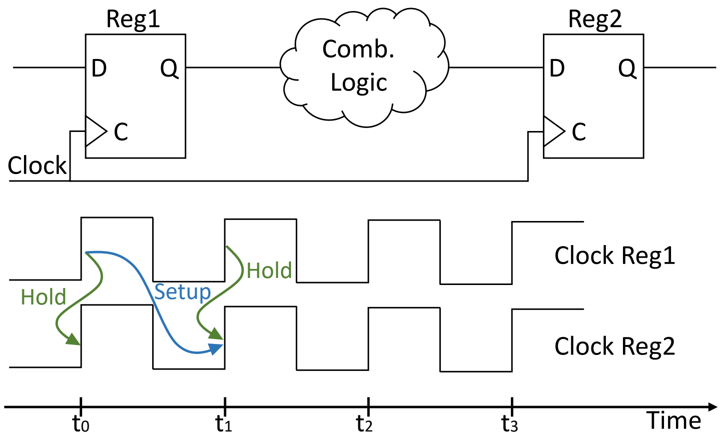


Figure 9: Timing path calculation example.

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.

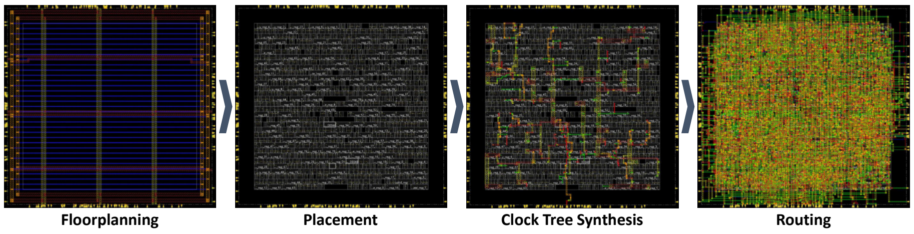


Figure 10: Block design implementation steps; floorplanning, placement, clock-tree synthesis, and, routing.

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

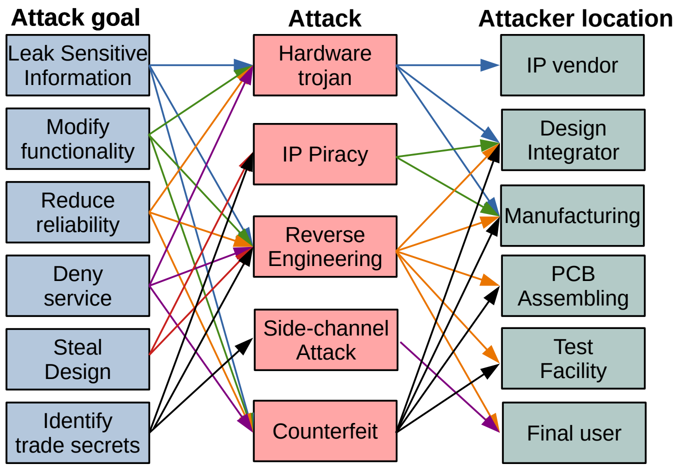


Figure 11: Systematization of hardware security around the attack method (adapted from [16])

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.

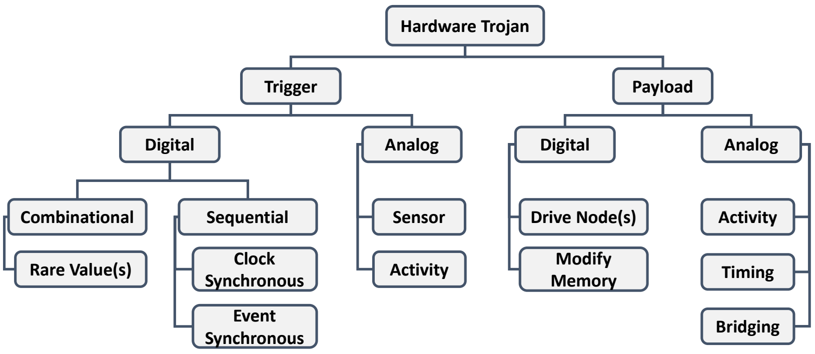


Figure 12: Additive hardware trojan taxonomy based on trigger *and* payload *implementation types (adapted from [61]).*

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].

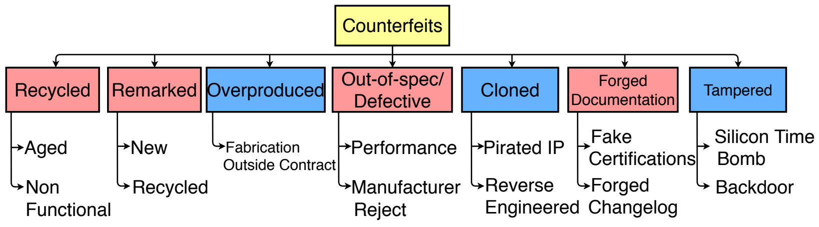


Figure 13: Taxonomy of counterfeit electronics (adapted from [59]).

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].

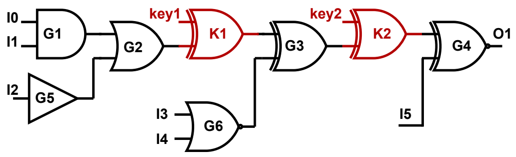


Figure 14: Example of a circuit locked using two XOR key gates, K1 and K2.

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.

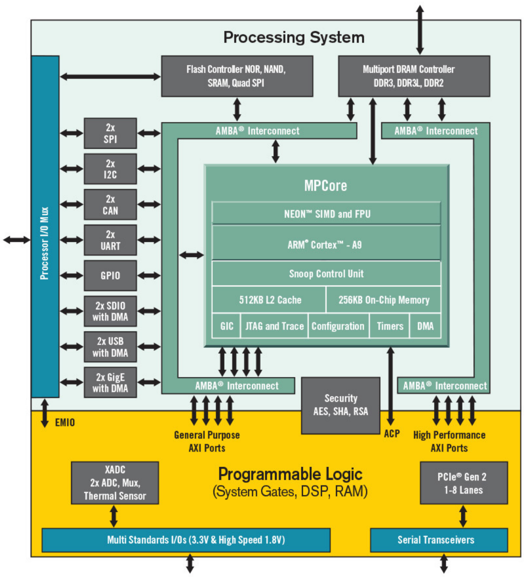


Figure 15: Example of FPGA-based SoC – Zynq-7000s (from [107]).

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

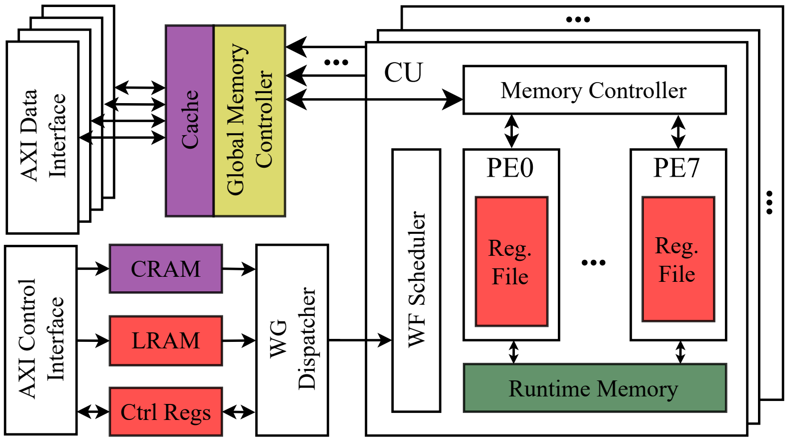


Figure 16: FGPU architecture with memories colored according to the layouts displayed in Figs. 3 and 4 (from [9]).

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.

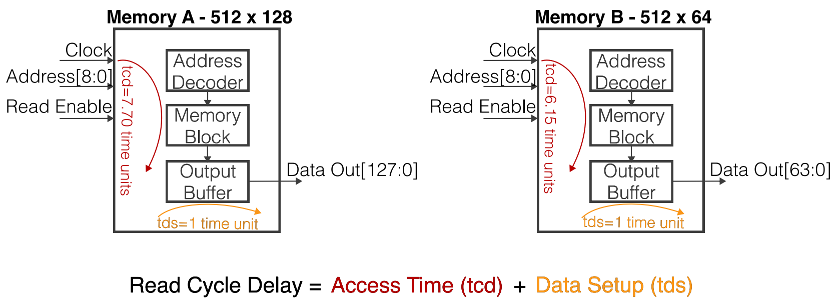


Figure 17: Simplified example of smart memory technique by halving the size of the word.

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.

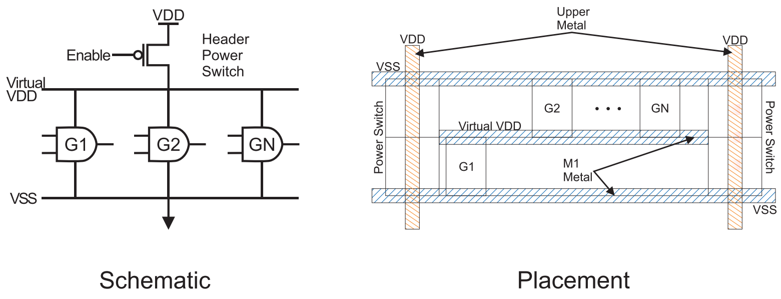


Figure 18: Example of a header power switch schematic (left panel) and placement (right panel).

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.

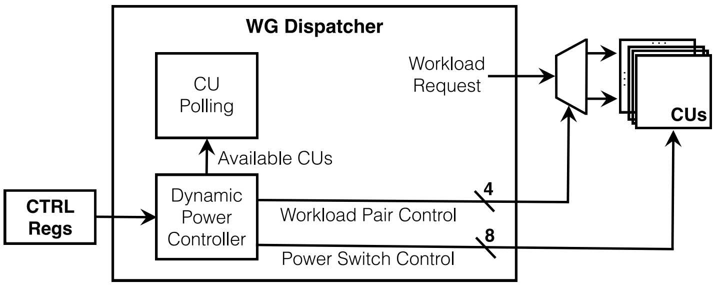


Figure 19: GPUPlanner generic dynamic power controller block diagram.

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

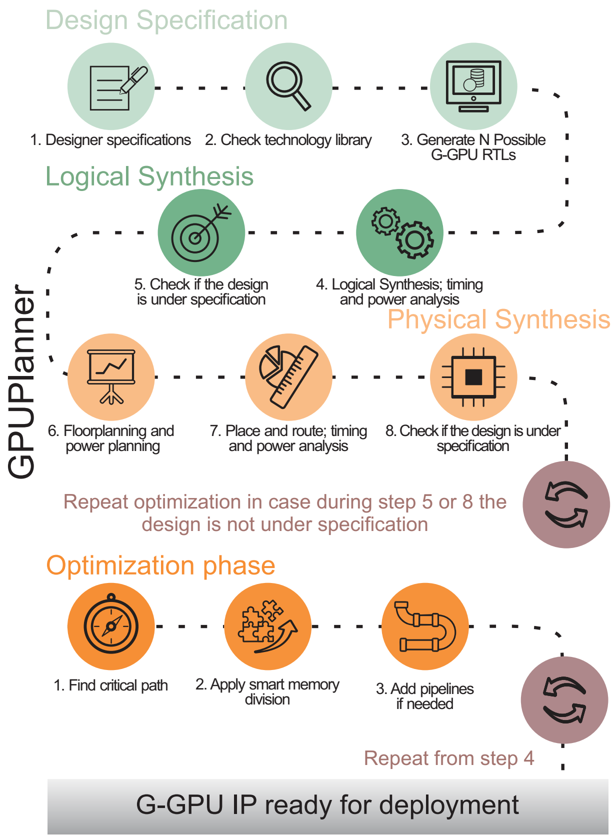

GPUPlanner
Design Specification
1. Designer specifications
2. Check technology library
3. Generate N Possible G-GPU RTLs
Logical Synthesis
4. Logical Synthesis; timing and power analysis
5. Check if the design is under specification
Physical Synthesis
6. Floorplanning and power planning
7. Place and route; timing and power analysis
8. Check if the design is under specification
Repeat optimization in case during step 5 or 8 the design is not under specification
Optimization phase
1. Find critical path
2. Apply smart memory division
3. Add pipelines if needed
Repeat from step 4
G-GPU IP ready for deployment

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.

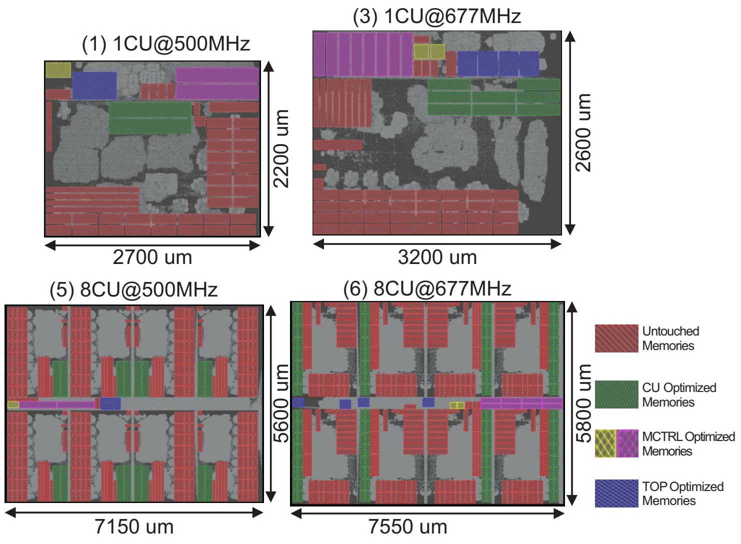


Figure 21: Layout comparison between the minimum and maximum performance of G-GPUs with 1 CU (top) and 8 CUs (bottom).

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.

			Dynamic Power (W)			Leakage (μW)			
# CU & Freq.	Power Gating	# Power Switches	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	-	-	-

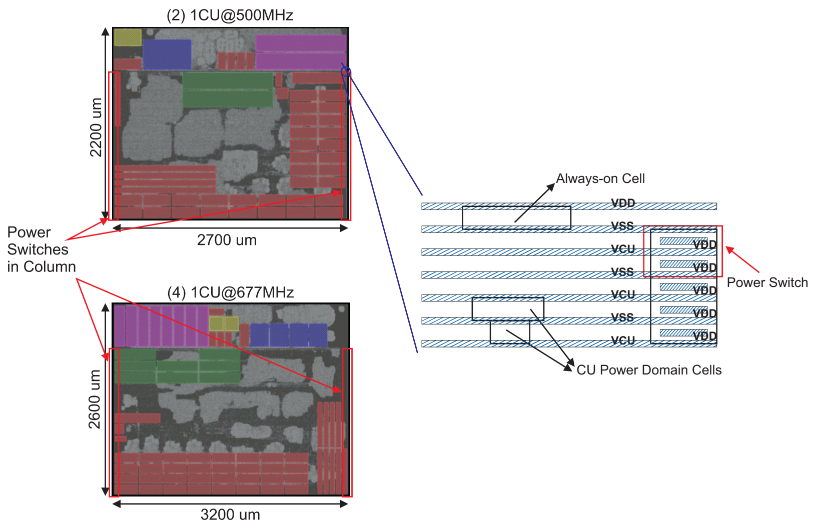


Figure 22: Layout comparison between G-GPU (2) 1CU@500MHz and (4) 1CU@677MHz with power gating.

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.

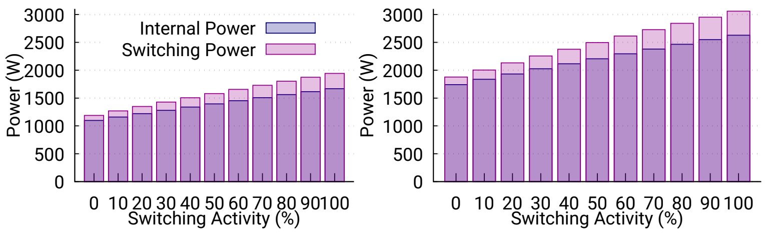


Figure 23: Compute unit partition dynamic power versus switching activity for (2) 1CU@500MHz (left panel) and (4) 1CU@677MHz (right panel).

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.

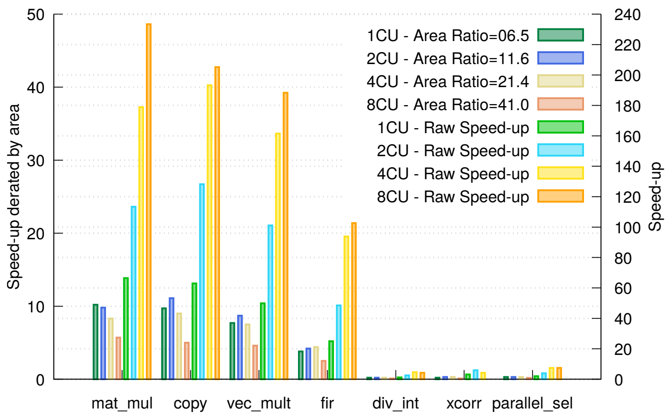


Figure 24: Speed-up over RISC-V.

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.

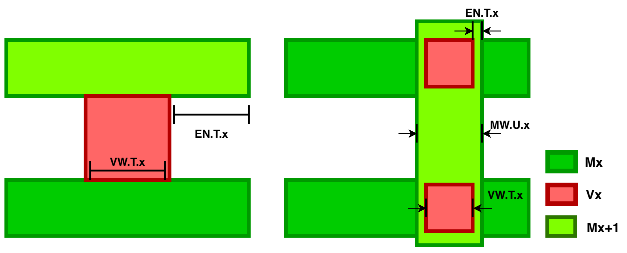


Figure 25: Compatibility rules between FEOL and BEOL (adapted from [131]).

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.

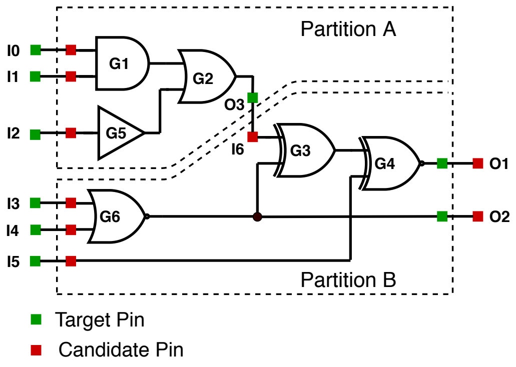


Figure 26: Example of a partitioned circuit (from [22]).

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.

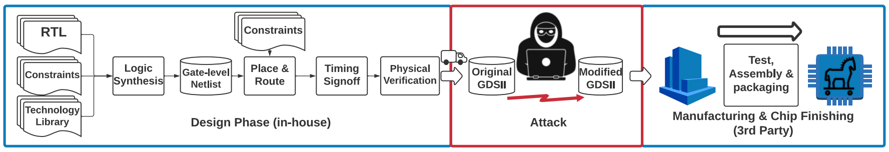


Figure 27: A typical IC design flow. Highlighted in red is the stage where a rogue element may mount an attack (modified from [17]).

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

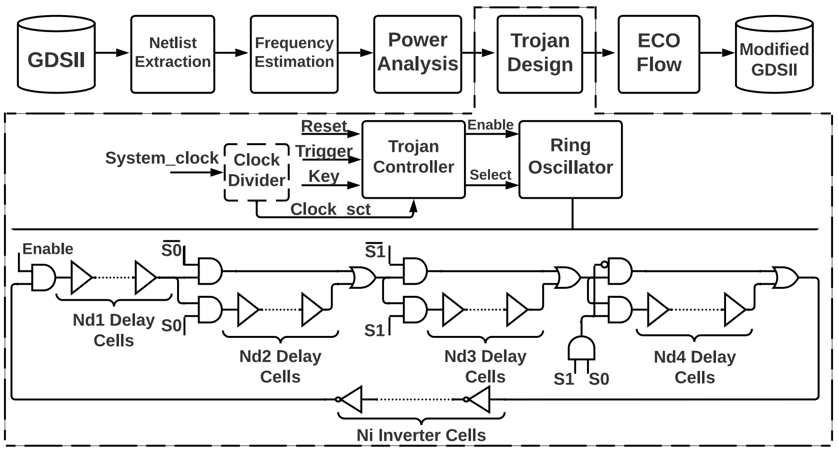


Figure 28: The proposed trojan insertion methodology for an SCT capable of leaking 2 bits per power signature reading (modified from [17]).

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.

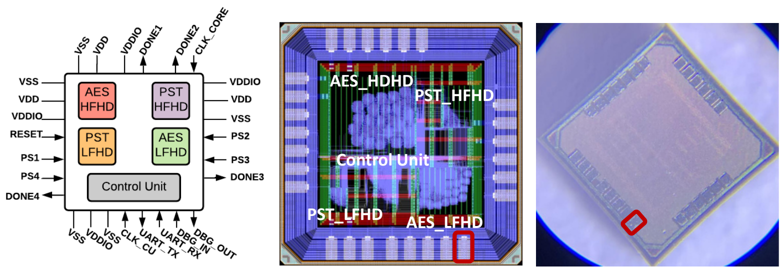


Figure 29: ASIC prototype top-level diagram (left), layout (middle), and its bare die (right). The highlighted pin identifies the lower-right corner in red (adapted from [17]).

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

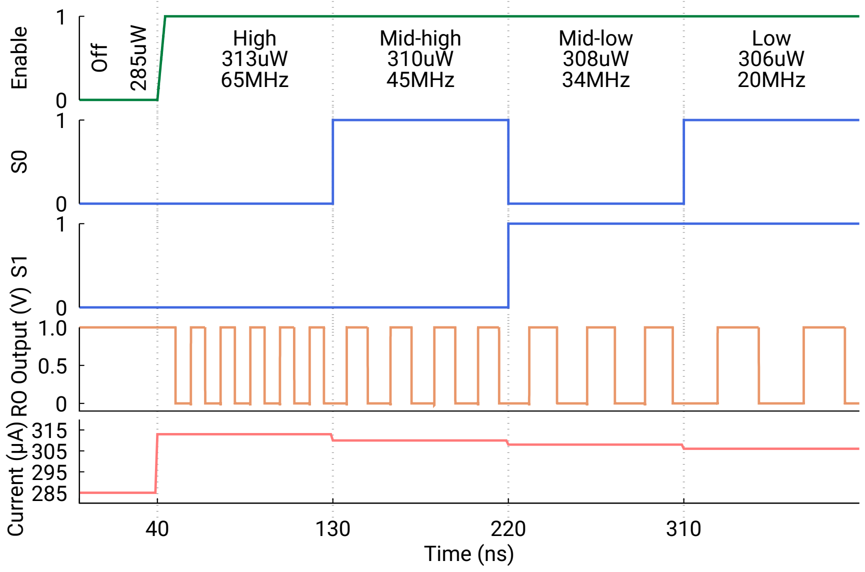


Figure 30: Post-layout simulation of SCT architecture in Cadence Spectre. The target design is AES_LFHD and the Trojan payload is configured as RO_{D6I10} (from [20]).

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

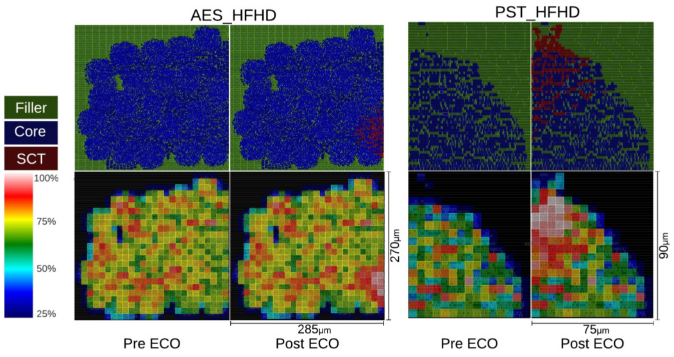


Figure 31: Placement view (top panels) and density map (bottom panels) of the AES_HFHD and PST_HFHD cores, before and after SCT insertion via ECO (modified from [17]).

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

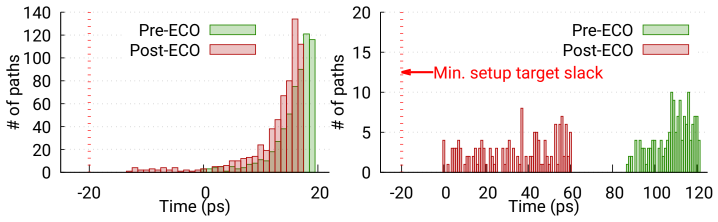


Figure 32: Pre- and post-ECO setup timing slack comparison of AES_HFHD (right) and PST_HFHD (left) (from [17]).

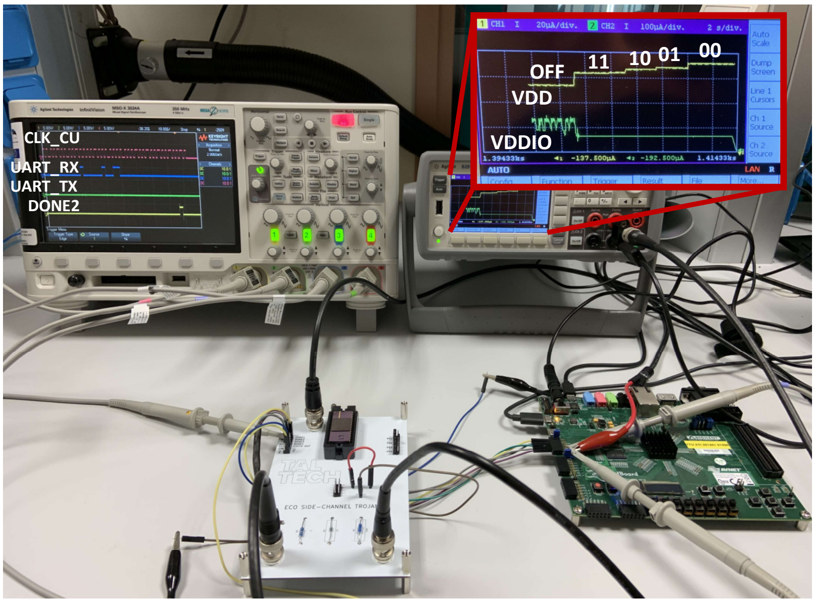


Figure 33: Setup used for bringing up the testchip. On the left side, we show the signals used for controlling the chips. On the right side, the current consumption of the chip when the RO is active (from [20]).

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.

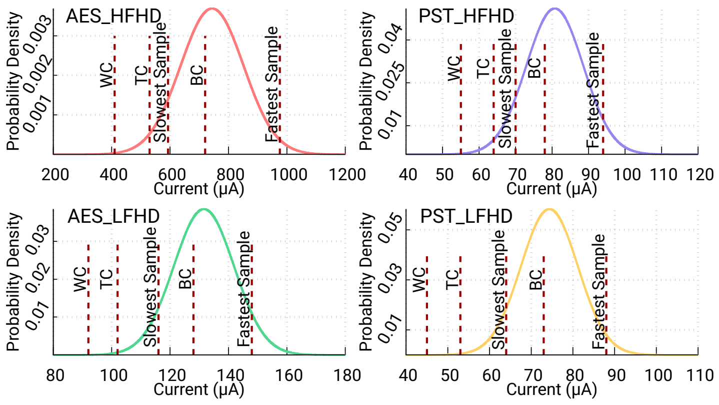


Figure 34: Leakage distribution for each crypto core contrasted with the leakage from the physical synthesis report for three corner cases and the leakage of outlier samples (from [20]).

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

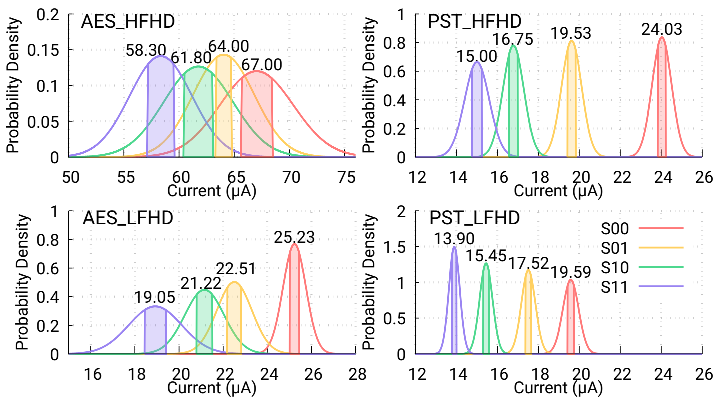


Figure 35: Power consumption "steps" distribution for each crypto core. The shadowed area represents the 95% confidence interval (from [20]).

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

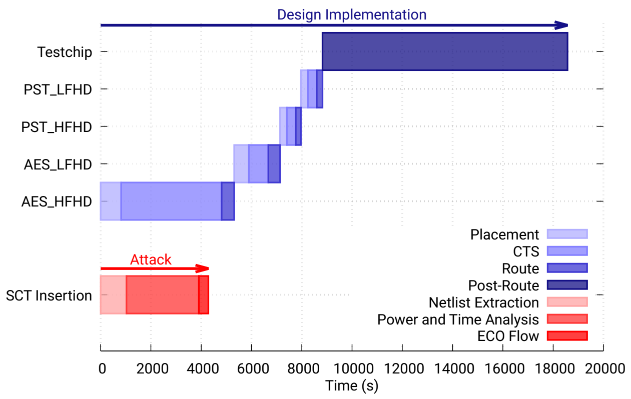


Figure 36: Physical implementation execution time (s) for each step of the flow, and execution time (s) for inserting the SCT in each implemented crypto core (from [20]).

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

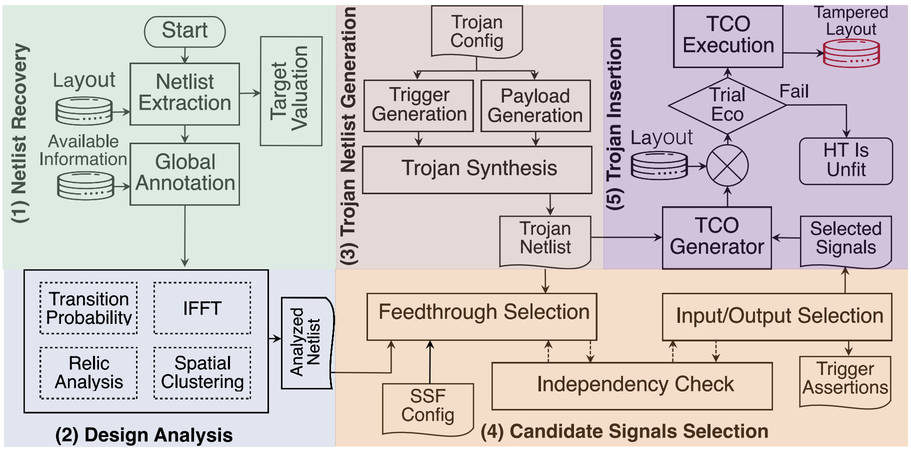


Figure 37: Steps 1)–5) of the BioHT Framework explained in detail. The flow starts at the top left, while the tampered layout (highlighted in red) is the result (adapted from [19]).

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

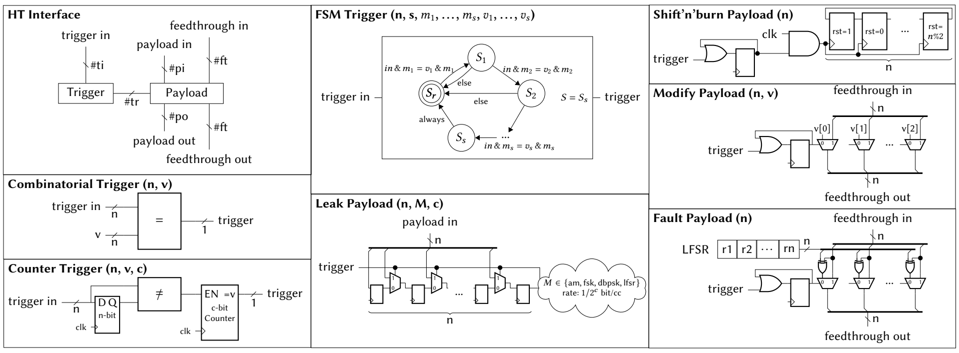


Figure 38: HT Interface and available trojan triggers and payloads. Trigger and payload parameters are given in parentheses (from [19]).

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/cyber-security/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