DOCTORAL THESIS

Leveraging FPGA Reconfigurability as an Obfuscation Asset

Zain Ul Abideen

TALLINNA TEHNIKAÜLIKOOL
TALLINN UNIVERSITY OF TECHNOLOGY
TALLINN 2024

TALLINN UNIVERSITY OF TECHNOLOGY
DOCTORAL THESIS
1/2024

Leveraging FPGA Reconfigurability as an Obfuscation Asset

ZAIN UL ABIDEEN

TAL TECH PRESS

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

The dissertation was accepted for the defence of the degree of Doctor of Philosophy in Information and Communication Technologies on 14 December 2023

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

Opponents: Prof. Dr. Giorgio Di Natale,
TIMA, Université Grenoble Alpes CNRS
Grenoble, France

Prof. Dr. Christian Pilato,
Department of Electronics, Information and Bioengineering
Polytechnic University of Milan
Milan, Italy

Defence of the thesis: January 15, 2024, Tallinn

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

Zain Ul Abideen _____
signature

European Union
European Regional
Development Fund

Investing
in your future

ISSN 2585–6898 (publication)
ISBN 978-9916-80-098-0 (publication)
ISSN 2585–6901 (PDF)
ISBN 978-9916-80-099-7 (PDF)
DOI https://doi.org/10.23658/taltech.1/2024
Printed by EVG Print

Abideen, Z. U. (2024). *Leveraging FPGA Reconfigurability as an Obfuscation Asset* [TalTech Press]. https://doi.org/10.23658/taltech.1/2024

TALLINNA TEHNIKAÜLIKOOL
DOKTORITÖÖ
1/2024

FPGA ümberkonfigureeritavuse rakendamine hägustamise vahendina

ZAIN UL ABIDEEN

TAL TECH
KIRJASTUS

Contents

List of Publications

The present Ph.D. thesis is based on the following publications that are referred to in the text by Roman numbers.

I 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)*, Shanghai, China, 2021, pp. 1-4. DOI: https://doi.org/10.1109/AsianHOST53231.2021.9699758

II Z. U. Abideen, T. D. Perez, M. Martins and S. Pagliarini, "A Security-aware and LUT-based CAD Flow for the Physical Synthesis of hASICs," *in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 10, pp. 3157-3170, 2023. DOI: https://doi.org/10.1109/TCAD.2023.3244879

III Z. U. Abideen, R. Wang, T. D. Perez, G. J. Schrijen and S. Pagliarini, "Impact of Orientation on the Bias of SRAM-based PUFs," *in IEEE Design & Test*, vol. X, no. X, pp. XXX-XXX, 2023. DOI: https://doi.org/10.1109/MDAT.2023.3322621

IV Z. U. Abideen, S. Gokulanathan, M. J. Alijafar and S. Pagliarini. "An Overview of FPGA-inspired Obfuscation Techniques," *in arXiv*, under review for ACM Computing Surveys, 2023. DOI: https://doi.org/10.48550/arXiv.2305.15999

Other related publications

V M. Imran, Z. U. Abideen, and S. Pagliarini, "An Experimental Study of Building Blocks of Lattice-based NIST Post-quantum Cryptographic Algorithms," *in Electronics*, vol. 9, no. 11, pp. 1953, 2020. DOI: https://doi.org/10.3390/electronics9111953

VI M. Imran, Z. U. Abideen, and S. Pagliarini, "An Open-source Library of Large Integer Polynomial Multipliers," *in 2021 24th International Symposium on Design and Diagnostics of Electronic Circuits & Systems (DDECS)*, Vienna, Austria, 2021, pp. 145–150. DOI: https://doi.org/10.1109/DDECS52668.2021.9417065

VII M. Grailoo, Z. U. Abideen, M. Leier, and S. Pagliarini, "Preventing Distillation-based Attacks on Neural Network IP," *in arXiv*, 2022. DOI: https://doi.org/10.48550/arXiv.2204.00292

VIII G. Basiashvili, Z. U. Abideen and S. Pagliarini, "Obfuscating the Hierarchy of a Digital IP," *in A. Orailoglu, M. Reichenbach, M. Jung (eds) Embedded Computer Systems: Architectures, Modeling, and Simulation. SAMOS 2022. Lecture Notes in Computer Science*, vol. 13511. Springer, Cham. DOI: https://doi.org/10.1007/978-3-031-15074-619

IX M. Imran, Z. U. Abideen, and S. Pagliarini, "A Versatile and Flexible Multiplier Generator for Large Integer Polynomials," *in Journal of Hardware and Systems Security*, 2023. DOI: https://doi.org/10.1007/s41635-023-00134-2

X M. J. Aljafar, Z. U. Abideen, A. Peetermans, B. Gierlichs and S. Pagliarini. "SCALLER: Standard Cell Assembled and Local Layout Effect-based Ring Oscillators," *in IEEE Transactions on Circuits and Systems II: Express Briefs*, under review, 2023.

Author's Contributions to the Publications

I In Publication I, I was the main author and proposed the design obfuscation concept. I also conducted the initial analysis of the results, prepared the figures, and wrote the manuscript.

II In Publication II, I was the main author, implemented the design obfuscation concept with security-aware CAD flow. I conducted experiments, executed physical synthesis flow, analyzed the results, prepared figures, and wrote the manuscript.

III In Publication III, I was the main author and designed a chip for an SRAM-based PUF. I designed the PCB, wrote the test plan, set up the equipment for the experiment, and conducted measurements from the chip. I also analyzed the results, prepared figures, and wrote the manuscript.

IV In Publication IV, I was the main author and conducted a comprehensive survey of various academic papers. I meticulously analyzed the data, formulated remarks, prepared figures, and wrote the manuscript.

Abbreviations

3PIP	Third-party Intellectual Property
ABEL	Advanced Boolean Expression Language
AI	Artificial Intelligence
AES	Advanced Encryption Standard
ASIC	Application-specific Integrated Circuit
AT	Arrival Time
ATPG	Automatic Test Pattern Generation
BCHD	Between-class Hamming Distance
BEOL	Back-End-Of-the-Line
CAD	Computer-Aided Design
CGRRA	Coarse-grained Runtime Reconfigurable Array
CLB	Configurable Logic Block
CMOS	Complementary Metal-Oxide Semiconductor
CPU	Central Processing Unit
CTS	Clock Tree Synthesis
DIP	Dual-In-Line
DPA	Differential Power Analysis
DP	Double Pattern
DRC	Design Rule Check
DRAM	Dynamic Random Access Memory
DSP	Digital Signal Processing
DUO	Design Under Obfuscation
EDA	Electronic Design Automation
eFPGA	embedded-Field Programmable Gate Array
EUIPO	European Union Intellectual Property Office
FC	Functional Composition
FEOL	Front-End-Of-the-Line
FFT	Fast Fourier Transform
FPGA	Field-Programmable Gate Array
FPU	Floating Point Unit
GDS	Graphic Data System
GE	Gate Equivalent
GPU	Graphics Processing Unit
GSHE	Giant Spin Hall Effect
HDL	Hardware Description Language
HPC	High Performance Computing
HW	Hamming Weight
HVT	High Voltage Threshold
hASIC	hybrid ASIC
IC	Integrated Circuits
IIR	Infinite Impulse Response
IoT	Internet of Things
IP	Intellectual Property
ITU	International Telecommunication Union
LL	Logic Locking
LSI	Large-Scale Integration

LUT	Look-Up Table
LVT	Low Voltage Threshold
MLP	Machine Learning Processing
MRAM	Magnetic-Random Access Memory
MTJ	Magnetic Tunnel Junction
LEF	Liberty Exchange Format
MESO	Magneto-Electric Spin-Orbit
MHW	Masked Hamming Weight
MIPS	Microprocessor without Interlocked Pipelined Stages
NIST	National Institute of Standards and Technology
NVM	Non-Volatile Memory
NoC	Network on Chip
P&R	Place & Route
PDK	Process Design Kit
PID	Proportional Integral Derivative
PLL	Phase-Locked Loop
PPA	Power-Performance-Area
PCB	Printed Circuit Board
PUF	Physical Unclonable Function
PSCA	Power Side-Channel Attacks
QP	Quad Patterning
QoR	Quality of Results
RAM	Random Access Memory
RE	Reverse Engineering
RISC	Reduced Instruction Set Computer
RSA	Rivest–Shamir–Adleman
RTL	Register-Transfer Level
RT	Required Time
SAT	Satisfiability
SBM	Schoolbook Multiplier
SDC	Synopsys Design Constraints
SHA	Secure Hash Algorithm
SMIC	Semiconductor Manufacturing International Corporation
SoC	System on Chip
SOT	Spin-Orbit Torque
SP	Single Pattern
SRAM	Static Random Access Memory
STT	Spin-Transfer Torque
STA	Static Timing Analysis
SVT	Standard Voltage Threshold
TNS	Total Negative Slack
TOTe	Tunable design Obfuscation Technique
TP	Tripple Pattern
TPU	Tensor Processing Unit
TRAP	TRAnsistor-level Programming
TSMC	Taiwan Semiconductor Manufacturing Company
ULSI	Ultra Large-Scale Integration
VLSI	Very Large-Scale Integration
WCHD	With-in Class Hamming Distance
WCSHD	With-in Class Sequential Hamming Distance
WNS	Worst Negative Slack

1 Introduction

The digitalization of critical infrastructure has become increasingly important in modern society [1]. Critical infrastructure refers to the systems and assets that are essential for a country's economy, security, and public health, such as transportation networks, energy grids, water supply systems, and healthcare facilities [2]. The goal of digitalizing critical infrastructure is to make human tasks more efficient, convenient, and faster, which can have a significant impact on various aspects of our daily lives [3, 4]. The process of digitizing critical infrastructure is complex and multifaceted. Nowadays, technological advances such as Artificial Intelligence (AI), Internet of Things (IoT), and multi-cloud computing are also being utilized [5, 6].

Integrated Circuits (IC)-based systems are pivotal components in technological advances, playing a crucial role in the digitalization of critical infrastructure. Achieving high performance in ICs requires their fabrication on advanced technology nodes, enabling rapid information processing, lower power consumption, and leading to higher transistor densities on a chip. IC foundries continuously refine their fabrication processes to meet these evolving demands, ensuring ongoing development and improvement. However, it is important to note that the production of ICs requires access to specialized equipment and advanced fabrication processes that only a few foundries are equipped with. Notable players in the industry, including Intel, Taiwan Semiconductor Manufacturing Company (TSMC), Samsung, and Semiconductor Manufacturing International Corporation (SMIC), possess the capabilities to fabricate ICs using cutting-edge process nodes, such as 7nm technology [7]. Building and maintaining a foundry becomes more complex as the industry evolves, resulting in rising costs. For instance, an estimated USD 33-34B would be required to build a foundry with 2nm technology [8].

To compete globally and produce high-performance ICs, design houses increasingly rely on globalized IC supply chains. For example, Apple will outsource the fabrication of their processor chips on 3nm from TSMC [9]. Adopting a globalized IC supply chain offers design houses the advantage of accessing high-end semiconductor facilities [10]. It has become common practice for various entities, including corporations and governments, to contract IC fabrication to third-party foundries. The globalized IC supply chain involves numerous dependent and interdependent tasks that can be hectic to manage. Figure 1 illustrates the primary stages of the globalized IC supply chain. The complete scenario of the IC design and the globalized IC supply chain could be broken down into four major parts: design, fabrication, testing, and deployment.

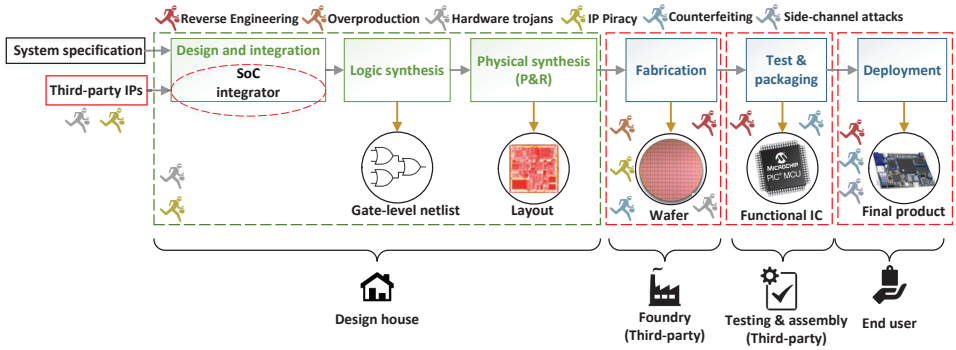


Figure 1: Typical stages involved in the globalized IC supply chain: untrusted stages are highlighted in red.

The design process of an IC involves block-level implementations and then organizing these blocks and interconnects with the help of Electronic Design Automation (EDA) tools. Often, the design houses purchase third-Party Intellectual Property (3PIP) blocks to incorporate into a design. This practice significantly minimizes the design effort and it helps to meet strict time-to-market constraints [11]. After developing certain blocks internally, they are combined with 3PIPs and subjected to logic synthesis. This process converts the design into a gate-level netlist. EDA tools utilize the gate-level netlist to generate a layout. The design layout includes different components and interconnections on the chip, sent to the foundry in a Graphic Data Stream (GDS) file. Once the ICs have been fabricated, they are packaged to be integrated into IC-based systems. Packaging involves enclosing the IC in a protective casing that shields it from external elements and provides electrical connections to the device. Often, the packaging of ICs is outsourced to third-party companies that specialize in this service. ICs also undergo testing to meet the desired performance and quality standards. In the end, ICs are finally deployed in the products.

In short, the design of an IC and all other steps that follow it involve multiple entities in the globalized IC supply chain. The green color in Figure 1 visually indicates the trustworthy steps in the design flow. The color red indicates the untrusted stages in the globalized IC supply chain. The layout of a design is exposed to untrusted entities. This does pose significant security risks, as illustrated in Figure 1. While all entities involved provide assurances, it is essential to acknowledge the potential lack of a 100% guarantee concerning their trustworthiness and integrity. This lack of guarantee primarily stems from zero-trust, which assumes that the foundry and its employees may pose potential adversarial threats. Fabrication holds the greatest importance among the various stages because the foundry can access detailed design information at a low level.

The potential consequences of these security threats can be severe, including service interruptions, compromised public data integrity, and financial losses. Notably, both the European Union (EU) and the United States (US) have issued warnings about the national security risks associated with scammers taking advantage of the globalized IC supply chain crunch [12]. The International Telecommunication Union (ITU) and the European Union Intellectual Property Office (EUIPO) have jointly disclosed that counterfeit electronics were responsible for a 12.9% reduction in legit smartphone sales in 2015. This resulted in a substantial monetary loss of EUR 45.3B for legitimate industries [13]. According to a study on Intellectual property (IP) piracy losses, the US experiences an annual loss of up to USD 600B due to IP piracy [14]. It is evident from the losses mentioned above that *security threats must be addressed* to ensure security in the globalized IC supply chain.

1.1 Security Threats in the Globalized IC Supply Chain

As depicted in Figure 1, various security threats need to be considered, such as Reverse Engineering (RE), overproduction, hardware trojans, IP piracy, counterfeit ICs, and side-channel attacks [15].

There are two types of RE: physical RE and logical RE. Physical RE is accomplished through various imaging tools and methods. This process involves intricate steps, including removing the package of an IC, delayering, alignment, and image analysis to reconstruct the design's netlist [16]. RE is often associated with IP piracy but can also be a tool for identifying vulnerabilities. It also poses risks concerning malicious logic insertion and extracting sensitive information, including cryptographic keys. Despite the difficulties involved in RE, determined adversaries can still perform RE using available

resources. On the other hand, the first step in logical RE is to convert the layout to a gate-level netlist [17]. Logical RE involves using techniques, such as structural and statistical analysis, to understand the functionality of an IC [18]. If any part of the circuit cannot be reverse-engineered as a gate-level netlist, then an adversary could recover the circuit's full functionality by utilizing other methods, such as the data flow of flip flops (FFs) [19]. This also applies to designs that contain Finite State Machines (FSMs), where the adversary aims to track the states and obtain the full functionality of the FSM [20].

Overproduction occurs when the foundry fabricates more ICs than required or specified. Untrusted foundries may be motivated to overproduce ICs and distribute them at lower prices in the grey or black market [21]. Foundries often find overproduction cost-effective as they can use the same set of masks to produce ICs.

Hardware trojans describe malicious modifications by adding complicated and hidden logic to an IC. Such trojans are designed to disrupt the normal operations of the IC or extract sensitive data [22]. It should be noted that trojans or backdoors embedded in 3PIPs may also include hidden functionalities that reveal restricted design aspects or extract confidential information. The malicious logic is often traced to 3PIPs integrated into the design or introduced during fabrication. Detecting and identifying these trojans can be challenging due to their small size within the IC layout and the lack of a reference or "golden" design for cross-validation. With its extensive access to the layout, the foundry can determine suitable locations for trojan insertion [23].

IP piracy occurs when the 3PIPs used in designs can be unlawfully obtained. Untrusted foundries may be interested in the unauthorized use, reproduction, and distribution of IP. In a foundry environment, unauthorized individuals may also engage in illicit activities, such as stealing valuable information through RE or selling IPs without proper authorization from the owner.

Counterfeit ICs are unauthorized replicas that intentionally resemble genuine ICs, exhibiting similar functionality. These ICs are less reliable and have degraded performance. Unauthorized companies or unethical sellers often supply ICs for use in products, which can lead to issues related to ICs after fabrication. They can be classified into seven types such as recycled, remarked, out-of-spec/defective, cloned, forged documentation, and tampering, as shown in Figure 3 of [24]. Conversely, these issues are associated with the design and/or fabrication stages of ICs.

Regarding attacks at the end-user stage, side-channel attacks are a major concern [25]. Side-channel attacks take advantage of leaked information in the form of current, voltage, timing, acoustic, or electromagnetic emissions. The side-channel attacks target the cryptosystem but can reveal valuable information for other design implementations [26]. Differential Power Analysis (DPA) is a side-channel attack that has successfully broken many cryptographic implementations [27]. In a DPA attack, power samples from the IC under attack are collected for a broad range of plaintext inputs. Once the samples are gathered, they are subjected to statistical analysis to extract the key. The attack does not aim to break the cryptographic algorithm; instead, it targets the implementation and looks for vulnerabilities to extract the key [28].

It is important to note that these threats can be effectively mitigated if the design stages are carried out within a trusted environment. However, these vulnerabilities persist due to the need to share design layout with untrusted foundry. The side-channel attacks are an exception because they leverage the leakage that is also dependent on the design. Nevertheless, researchers have developed numerous techniques as countermeasures to combat these threats, and the field continues to evolve as researchers strive to introduce

novel, practical, and resilient approaches [29].

1.2 Countermeasures

Countermeasure techniques aimed at enhancing IC security encompass various approaches, such as watermarking [30, 31, 32], fingerprinting [33, 34], camouflaging [35, 36, 37], split manufacturing [38, 39, 40], metering [41, 42, 43], Logic Locking (LL) [44, 45, 46, 47, 48, 49], and reconfigurable-based obfuscation techniques [50, 51, 52, 53, 54, 55, 56, 57, 58, 59].

Watermarking involves embedding the designer's unique signature, such as a secret design, into the IC to establish ownership or detect unauthorized modifications. On the other hand, fingerprinting incorporates both the designer's and the end-user's signatures to trace instances of piracy. These passive techniques aid in identifying IP piracy but do not actively prevent it. They can be integrated at the logic and physical synthesis stage [33]. The objective of camouflaging is to impede RE attempts by replacing specific gates in the design with camouflaged equivalents. When viewed from the top, these camouflaged gates closely resemble their non-camouflaged counterparts. Camouflaging techniques employ dummy contacts, filler cells, or diffusion-programmable standard cells to achieve their purpose [35]. The process of split manufacturing, which involves separating the front-end-of-the-line (FEOL) and backend-of-the-line (BEOL) metal layers during the design stage of an IC and producing them in different foundries, effectively prevents piracy by untrusted foundries [60]. However, it does not offer protection against end-users [38]. To address this issue, metering techniques are utilized, which assign a unique identifier to each IC. Passive metering techniques identify piracy, while active metering techniques allow the IC owner to track and monitor their behavior during in-field operations [43]. LL is generally implemented at the gate-level after the logic synthesis stage by inserting additional logic to hide the functionality of a design behind key bits. However, selecting the gates to be locked is challenging because not all the keys help to provide the security level. Moreover, the LL may increase Power, Performance, and Area (PPA) depending upon the used technique. Strategies have been developed to select gates for locking based on maximum security per overhead unit [44].

All the techniques mentioned above aim to provide security against threats during IC fabrication. Some techniques also offer protection against post-fabrication attacks. Table 1 compares the countermeasures on different stages of the globalized IC supply chain. Camouflaging involves creating gates with camouflage designs to protect untrusted end-users in the deployment stage. Split manufacturing is a layout-level technique that offers protection during fabrication but partially relies on a trusted foundry. Passive metering offers protection in the testing & packaging and deployment stage. LL can protect against rogue elements at any point in the design flow except for the trusted design house. LL does not require foundry support like camouflaging and does not require trusted BEOL foundry like split manufacturing.

Significant concerns exist regarding attacks on camouflaging, split manufacturing, and LL. For instance, removal and Boolean Satisfiability (SAT) attacks on camouflaging could fully or partially deobfuscate the design [35, 61]. Numerous attacks and defenses have been proposed on LL [44, 45, 46, 47, 48]. Attacks on split manufacturing are also prevalent, as shown in [62, 63]. While LL provides high security at all stages of the IC supply chain, the SAT attack on LL broke its security. The initial SAT attack compromised the security measures implemented by LL [64]. Then, LL has recently seen a rise in the interplay between the countermeasure and attack techniques. Still, it

Table 1: The security of countermeasures at various stages of the globalized IC supply chain.

Countermeasure technique	System on chip (SoC) integrator	Fabrication	Test & Packaging	Deployment
Camouflaging [35, 36, 37]	✗	✗	✗	✔
Split Manufacturing [38, 39]	✗	✔	✗	✗
Metering (passive) [41, 42, 43]	✗	✗	✔	✔
Logic Locking [44, 45, 46, 47, 48]	✔	✔	✔	✔

Security: ✔ (Yes), ✗ (No).

has been observed that LL is vulnerable to numerous attacks [65, 64, 66, 67, 68, 69].

1.3 Motivation and Objectives

Reconfigurable-based obfuscation techniques have emerged as highly promising methods that effectively protect against various security threats. The concept is as simple as a Field Programmable Gate Array (FPGA), where the design is not present until a bitstream is loaded [70]. This approach consists of a reconfigurable part within the circuit, offering robust security measures against untrusted fabrication. A crucial and relatively small part of the circuit remains locked, taking advantage of its reconfigurable nature. The design is currently non-functional and can be made functional by using the appropriate bitstream [71].

Reconfigurable-based obfuscation involves a combination of standard logic and reconfigurable logic elements. These techniques utilize embedded-Field Programmable Gate Array (eFPGA) or reconfigurable elements to achieve a high level of obfuscation. The use of reconfigurable-based obfuscation techniques has shown significant potential in providing quality obfuscation that can withstand various security threats. These techniques employ various types of reconfigurable elements, such as Static Rrandom-access Memory (SRAM)-based Look-Up Tables (LUTs) [50, 51, 70, 71, 72], FF-based LUTs [73, 74] and Non-Volatile Memory (NVM)-based LUTs [52, 53, 54, 75, 76, 77, 78]. Additionally, other reconfigurable-based approaches have been introduced in recent years to enhance the protection of digital designs [55, 56, 57, 58, 59, 79, 80, 81, 82, 83, 84, 85]. Few other approaches involve configuring transistors and switches instead of LUTs, as described in [57, 79]. Most of the reconfigurable-based obfuscation techniques use LUTs as valuable assets in safeguarding the integrity of a design.

LL and reconfigurable-based obfuscation techniques generally aim to protect IP against supply chain attacks. Reconfigurable-based obfuscation has gained increased attention for its high resiliency against state-of-the-art attacks, but debates have arisen regarding the trade-offs between security and PPA. Figure 2 represents the conceptual difference between LL and reconfigurable-based obfuscation. LL involves adding gates with key inputs to the original design and requires the correct configuration of the secret key. In contrast, reconfigurable-based obfuscation relies on loading the correct bitstream and utilizes reconfigurable logic elements. These approaches exhibit some similarities, and as a result, attacks developed for the LL attacks can also be applied to reconfigurable-based obfuscation techniques. As previously mentioned, the SAT attack is incredibly powerful and has the ability to break the security measures of many LL countermeasures. SAT attack can also be applied to reconfigurable-based obfuscation techniques, which rely on a complex arrangement of reconfigurable elements. SAT attack creates a large bitstream or key bits with reconfigurable elements. It is essential to note that a large bitstream creates a vast search space with 2^n key combinations where the value of n represents the number of key inputs. This leads to an exponential

increase in the search space for the correct key, making the SAT attack computationally infeasible.

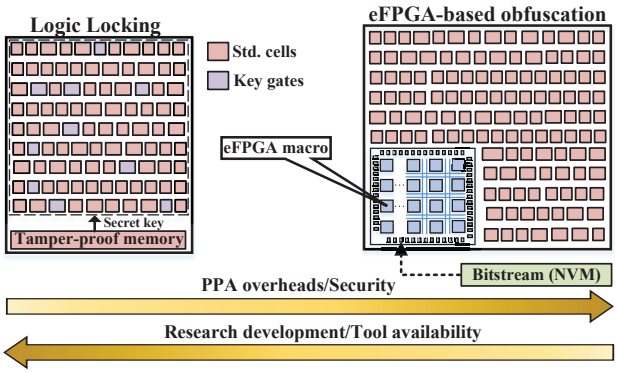


Figure 2: Comparison Diagram: LL vs. eFPGA-based obfuscation.

In addition to the SAT attack, LL is also vulnerable to various other attacks, such as structural attacks, algorithmic attacks, and side-channel attacks [35, 86, 87]. As a result, the design's overall security may be compromised. The adversary gains access to the entire design, comprising the original IP and the key gates. Reconfigurable-based obfuscation also conceals a selected portion of the design, ensuring that only a fraction is susceptible to potential adversaries. This technique presents a promising way to augment the security of ICs against attacks that target the globalized IC supply chain.

In the domain of LL, designers use the conventional Computer-Aided Design (CAD) design flow to leverage various tools for logic synthesis, timing analysis, and design optimization. On the other hand, reconfigurable-based obfuscation lacks a compatible tool with the conventional CAD flow that can support these functionalities [58]. Figure 2 demonstrates an instance of reconfigurable-based obfuscation that exploits eFPGA to lock the design. The Application-Specific Integrated Circuits (ASIC) part of an eFPGA-based obfuscation is considered to be static. Hence, it is referred to as the static part. Currently, no automatic tool is available for logic partitioning between eFPGA part and the static part. Therefore, developing customized tools is necessary to implement such techniques, which requires significant effort. LL benefits from more sophisticated techniques and readily available tools, as indicated by the arrows pointing towards the right in Figure 2. Consequently, reconfigurable-based obfuscation results in higher security and PPA overheads when compared to LL. As a result, the floorplan of eFPGA-based obfuscation appears larger, as illustrated in Figure 2.

For efficient utilization of reconfigurable logic, it becomes necessary to consider the PPA overheads. Reconfigurable-based obfuscation techniques face several challenges during the design, fabrication, testing, and deployment stages. For example, the SRAM-based LUT implementation requires careful consideration of the placement of SRAM. On the other hand, Spin-Transfer Torque (STT), Spin-Orbit Torque (SOT), and Magnetic-Random Access Memory (MRAM) are hybrid and emerging technologies that require specific considerations and capabilities during the fabrication process. These technologies present operational challenges impacting PPA overheads. For example, the TRAnsistor-level Programming (TRAP) fabric, as described in [57, 79], adopts a transistor and switch box-based approach for providing obfuscation in the design.

However, incorporating reconfigurable elements into the design introduces a certain level of PPA overhead, which may affect the overall performance of the design. During

the design phase, PPA overheads are important and remain so during obfuscation. As advancements in ASIC designs progress, balancing robust security measures with high performance and low area utilization proves to be a constant challenge for designers. For eFPGA-based obfuscation, the most sensitive part of the design is redacted to the eFPGA, while the remaining portion or static part uses standard cells [55]. In both cases, either LUT-based obfuscation or eFPGA-based obfuscation requires *storing the bitstream* of the design. The aforementioned techniques utilize SRAM, FF, NVM, STT, SOT, and MRAM technology to store the bitstream.

In the deployment stage, the security of the bitstream is crucial as it contains the most sensitive part of the design. It has been shown that many attacks are capable of reverse engineering the bitstream of FPGAs or reconfigurable hardware [88, 89, 90, 91, 92]. These days, many modern FPGA devices are equipped with encryption and authentication techniques. For instance, the Xilinx Vivado Design Suite supports Advanced Encryption Standard (AES) and Rivest–Shamir–Adleman (RSA)-based authentication [89]. AES is a widely recognized standard supported by the National Institute of Standards and Technology (NIST) and the U.S. Department of Commerce [93]. This ensures the bitstream remains unmodified and can only be deciphered using a dedicated on-chip decryption block. Physical Unclonable Functions (PUFs) are promising primitives for cryptography and hardware security which ensure the robustness of the AES and RSA cryptocores for securing the bitstream. PUFs are used to generate a unique secret key of these crytocores [94, 95]. Additionally, their non-reproducible and unclonable properties result in the production of a unique signature. PUFs can too be classified into different groups including ring-oscillator based PUF (RO-PUF) [96], arbiter PUF [97], Dynamic Random Access Memory (DRAM) PUF [98], and SRAM-based PUF [99, 100, 101, 102, 103, 104, 105, 106, 107]. SRAM-based PUFs, in particular, offer a combination of simplicity, low cost, high reliability, scalability, and cryptographic strength, making them a popular choice for commercial PUF solutions [94]. Additionally, they rely on standard SRAM IP, which is readily available to designers and eliminates the need for customization.

SRAM-based PUFs must meet various quality criteria to serve as a root of trust effectively. These criteria encompass reliability, entropy, uniqueness, randomness, and bias pattern [108]. Several factors, including environmental conditions, post-processing, and fabrication processes, influence the characteristics of SRAM-based PUFs. The designer's choices also play a significant role in determining these characteristics. For example, SRAM with different number of addresses, words, and aspect ratios will produce varying responses and different levels of robustness. Additionally, the floorplan, location, rotation, and power delivery strategy decisions during the physical synthesis can impact the PUF's characteristics. When creating SRAM-based PUFs, it is crucial to consider memory and chip-level decisions carefully to ensure the robustness of reconfigurable-based obfuscation. Therefore, the impact of design time decisions should be considered while designing SRAM-based PUFs.

Reconfigurable-based obfuscation necessitates integrating a new custom tool into the traditional CAD flow. Reconfigurable-based obfuscation techniques can pose challenges during the CAD flow implementation due to their complex nature. To address this, a platform or framework is needed to empower designers to make informed decisions and manage trade-offs effectively. This platform would allow for assessing design versus security trade-offs to evaluate better the impact of implementing security measures. The objective of this thesis is to introduce and implement a new method of obfuscation that ensures security throughout the global IC supply chain, resulting in an automated

obfuscation tool. During the deployment stage, the bitstream will be encrypted using a secret key. This secret key is utilized for the encryption algorithm to ensure the security of the bitstream. The secret key will be generated using SRAM-based PUF. Therefore, a robust analysis of the secret key generation is also incorporated.

1.4 Novelty, Contributions & Outline of the Thesis

The *outcome of my thesis* is a custom tool fully compatible with a standard CAD flow for obfuscating the design. Figure 3 presents the overall structure of the thesis. In Chapter 2, the background is explained. Each subsequent chapter in this thesis is a unique contribution; the specifics are outlined below.

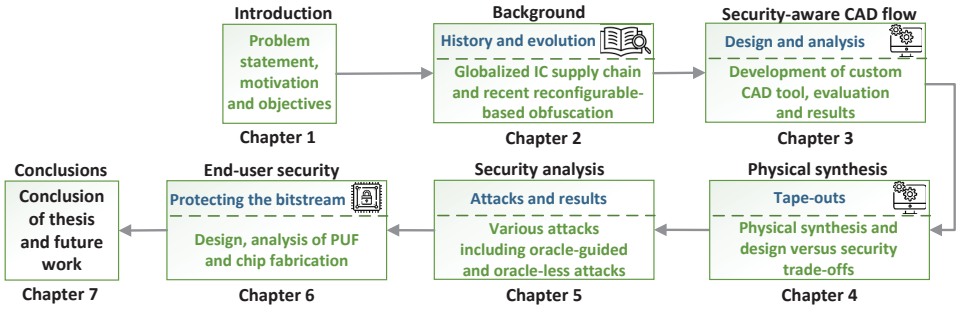


Figure 3: Organization of the thesis.

- **Chapter 2** This chapter provides a comprehensive overview of advanced IC fabrication and evaluation of technology nodes, with a specific focus on reconfigurable-based obfuscation techniques. It begins by introducing the background of these techniques and discussing their origins, motivations, and key principles. To facilitate a systematic understanding of reconfigurable-based obfuscation, it classifies these techniques based on three essential factors: the technology employed, the type of elements used for obfuscation, and the protected IP. This classification enables a structured analysis and comparison of different reconfigurable-based obfuscation approaches. In addition to classification, it also presents a comparative analysis of reconfigurable-based obfuscation techniques in terms of their PPA overheads. Furthermore, it provides a detailed security analysis of reconfigurable-based obfuscation techniques. It explores various threat models and examines recent attack attempts targeted at these techniques.

- **Chapter 3** This chapter introduces my design obfuscation concept and emphasizes the trade-offs between design and security considerations. It begins by discussing the security-aware CAD flow, which involves various stages, such as generating Register-Transfer level (RTL) code of the obfuscated design, logic synthesis, and physical synthesis. It also provides a detailed explanation of different phases within a custom tool called "**T**unable design **O**bfuscation **Te**chnique" (TOTe). Furthermore, it presents initial results from numerous designs, focusing on the PPA overheads. Recognizing that PPA is a critical factor in design optimization, the chapter introduces additional techniques included in the tool to decompose the LUTs and enhance the Quality of Results (QoR). It highlights the analysis and experimental findings using the TOTe tool.

- **Chapter 4** The main focus of this chapter is to showcase the physical implementation of hybrid ASIC (hASIC) using a commercial Complementary Metal-oxide-semiconductor (CMOS) technology. It provides a more realistic assessment by presenting the physical implementation of the selected designs. The analysis includes large circuits, including combinational and sequential circuits, with varying levels of obfuscation applied. Furthermore, it presents the final layouts for baseline and optimized variants. The optimized variants incorporate the techniques presented in the previous chapter, highlighting the improvements achieved through optimization.

- **Chapter 5** This chapter presents a detailed threat model and security analysis of the obfuscated circuits. Throughout the analysis, various attacks, including the oracle-guided and oracle-less attacks, are executed to assess the security of the obfuscated designs. It also provides a comprehensive understanding of design obfuscation, including custom structural attacks, to evaluate the effectiveness and robustness of obfuscated circuits.

- **Chapter 6** This chapter provides a comprehensive analysis of the SRAM-based PUF to enhance the security of the hASIC's bitstream. By integrating PUF technology into hASIC, the encryption/decryption keys for the bitstream can be securely protected using unique secret keys. To evaluate the robustness and the impact of different SRAm-based PUF characteristics chosen by the designer, a chip was designed and implemented using 65nm CMOS technology. It includes the findings, emphasizing the importance of carefully considering the design choices and orientations of SRAM-based PUF.

- **Chapter 7** In conclusion, this chapter marks the end of the thesis and outlines potential future directions for research and development.

2 Background

This chapter provides a concise overview of the history of ICs. It explains the evolution of technology nodes, their dependency on the globalized IC supply chain, and the background of reconfigurable-based obfuscation. It also provides information about various existing approaches for reconfigurable-based obfuscation and their security analysis. In addition, It provides details on the background of SRAM-based PUF.

2.1 History of the IC

The introduction of the transistor by Bell Labs scientists John Bardeen, Walter Brattain, and William Shockley in 1947 was a major milestone in the field of electronics [109]. It replaced the bulky and unreliable vacuum tube commonly employed in electronic devices at the time with a smaller, more reliable, and power-efficient alternative. This breakthrough led to the development of ICs, which were created by integrating multiple transistors on a single chip [110]. In addition to transistors, an IC also includes capacitors and resistors, all integrated into a single, compact package. The first IC developed by Jack Kilby at Texas Instruments in 1958 only contained a few transistors. In 1961, the world's first commercial IC [111], the N51x series, was released, demonstrating great potential to revolutionize electronics as shown in Figure 4. Figure 5 illustrates the timeline of computer chips and transistor counts. In the early 1950s, it was challenging to integrate many transistors on a single chip. By the early 1960s, dozens of transistors could be integrated into a single chip, leading to Medium-Scale Integration (MSI) of circuits. By the end of the 1960s, designers could integrate hundreds of transistors, leading to Large-Scale Integration (LSI). By integrating more and more electronic components onto a single chip, designers can reduce the size of electronic devices while improving their performance and reducing power consumption. By the 1970s, thousands of transistors were being integrated onto a single chip, resulting in Very Large-Scale Integration (VLSI) of circuits.

Figure 4: The SN514 IC released by Texas Instruments [112].

This has led to the development of highly sophisticated and portable personal computers. In the 1980s, the number of transistors on a single chip had increased to thousands, and the technology was dubbed Ultra-Large-Scale Integration (ULSI) ULSI. The era of ULSI began with the integration of millions of transistors into a single chip. In the 2000s, this number increased to hundreds of millions of transistors on a chip, resulting in the development of 64-bit microprocessors for personal computers. The need for more compact, powerful, and efficient electronic devices drives the progression towards LSI, VLSI, and USLI [113]. The evolution of ICs has been driven by advances in semiconductor technology, which have enabled the creation of increasingly complex circuits. In the 2010s, remarkable advances in technology led to the integration of

billions of transistors onto a single chip. This trend is expected to continue with the development of new materials, fabrication techniques, and design methodologies, leading to even more advanced integrated circuits. The principle of Moore's Law states that the number of transistors in an IC doubles approximately every two years. Advancements in transistor technology have enabled an impressive increase in the number of transistors that can be integrated into a single chip [114].

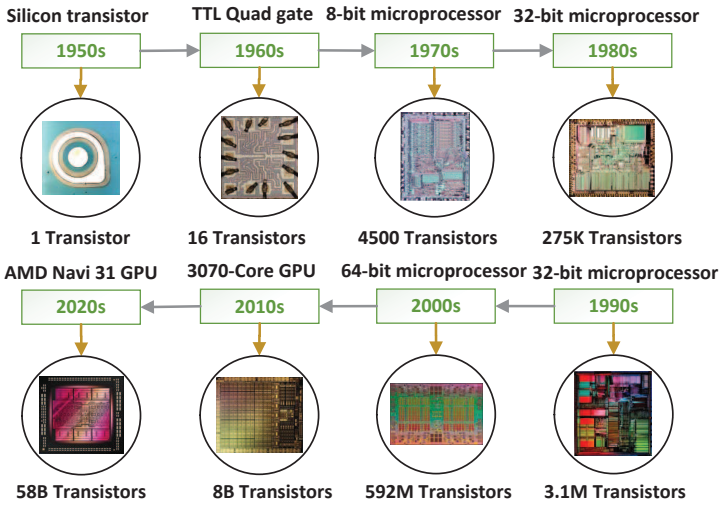


Figure 5: The timeline of the semiconductors in computers [114].

2.2 Evolution of the Technology Node

The semiconductor industry has undergone significant changes over the years, and consequently, the globalized IC supply chain has also evolved. In the 1980s, Japan dominated the semiconductor market due to its superior fabrication processes that provided better yield [115]. In the 1990s, the semiconductor market experienced a significant shift as emerging economies like Korea and Taiwan began to dominate the industry. These countries made substantial investments, primarily focusing on the fabrication process. Notably, they were known for their exceptional commitment to capital expenditure, often reinvesting 100% of their revenue back into their own companies. This strategic approach allowed Korea and Taiwan to rapidly expand their semiconductor fabrication capabilities and gain a competitive edge in the global market [116]. By consistently allocating a significant portion of their resources towards capital expenditure, they could enhance their production capacity, upgrade equipment and technologies, and improve overall efficiency.

After that, a significant shift occurred as companies with advanced and mature fabrication processes began to adopt a specialized approach. This approach involved offering the service of pure-play foundries, focusing solely on the fabrication of semiconductors [117]. By specializing in fabrication, pure-play foundries offered various benefits to semiconductor companies. They provided access to state-of-the-art fabrication facilities, advanced process technologies, and extensive production capacity. Semiconductor companies could outsource their fabrication needs to these foundries, allowing them to focus on their product's research, design, and marketing aspects. This trend enabled semiconductor companies to optimize resources, reduce capital expenditure, and enhance

flexibility in meeting market demands. It also allowed smaller semiconductor companies to access cutting-edge fabrication technologies without significant upfront investments in fabrication facilities. With the advent of globalization and the rise of new players in the market, the semiconductor supply chain has become globalized, more complex, and diversified.

The technology landscape of the semiconductor industry has undergone significant changes over the years [118], as depicted in Figure 6. The number of cutting-edge fabrication facilities globally is decreasing, while the older technologies such as 130nm and 90nm are still operational. This reduction can be attributed to various factors, including the high costs of building and maintaining advanced semiconductor fabrication facilities. The cost of constructing and operating a foundry has reached unprecedented levels. The investment required to develop and upgrade fabrication plants to accommodate the latest technology nodes has become prohibitively expensive for many companies [119]. This has led to consolidation in the industry, with fewer players capable of affording the significant capital expenditures needed to stay at the cutting edge of semiconductor technology. The semiconductor industry is predominantly led by three major players: Intel, Samsung, and TSMC, as shown in Figure 6.

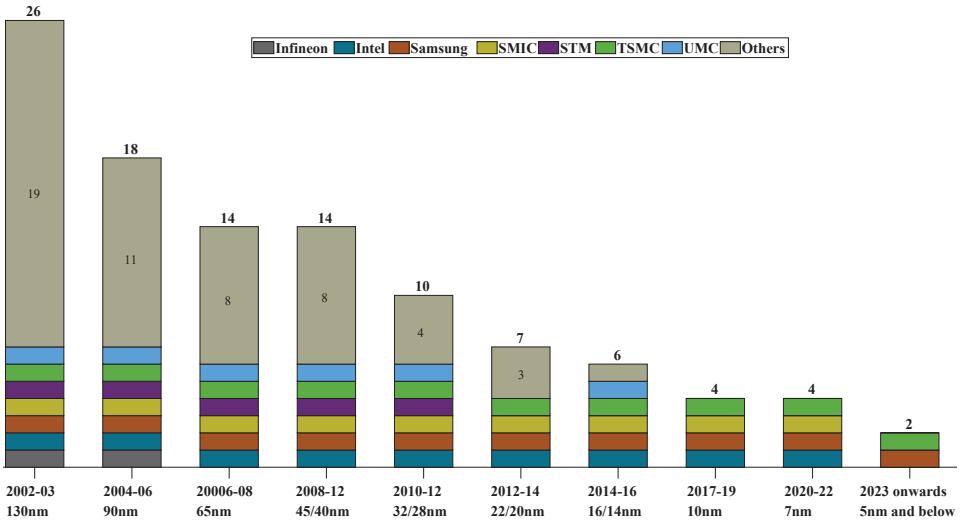


Figure 6: The semiconductor industry evolution up to 10nm [120].

During the period of 2017 to 2019, the fabrication industry witnessed the emergence of the 10nm technology node, which received significant attention from major players in the industry. In early 2020, an additional entrant emerged as SMIC (Semiconductor Manufacturing International Corporation), a pure-play foundry [121]. SMIC announced its fabrication on the 7nm technology node, joining the ranks of Intel, Samsung, and TSMC as key players in this advanced node. This development showed the expanding competition in the semiconductor industry and the growing interest in pushing the boundaries of process technology. These companies have the financial resources and technical expertise to invest in advanced fabrication processes and push the boundaries of chip fabrication. In addition to the high fabrication costs, the semiconductor market also witnesses significant R&D expenditures. Companies allocate substantial resources to research and development activities to drive innovation, improve fabrication processes, and meet the demands of emerging technologies and applications [122]. These R&D

investments are essential for maintaining competitiveness and staying ahead in the highly dynamic semiconductor industry. This scenario highlights the challenges faced by the semiconductor industry in terms of cost-effectiveness, technological advancement, and maintaining a competitive edge.

Pursuing denser and faster ICs has led to a significant increase in the complexity of the fabrication process [123]. The increasing complexity in chip design has led to the adopting of advanced EDA tools, customized IP libraries, and innovative implementation techniques. Advanced packaging techniques, stacked die technologies, and other assembly methods have become essential to meet the demands of advanced IC designs [124]. Design companies require access to Process design kits, collaboration with capable EDA tool vendors, and partnerships with specialized IP providers to successfully fabricate a modern complex chip. Even industry giants like Intel, who control their fabrication processes, often seek assistance from external entities to develop their products. The complexity of device fabrication and testing has experienced significant growth, particularly as the industry transitioned to advanced technology nodes, such as 10nm and 7nm. The challenges in printing intricate designs and conducting thorough device testing have increased exponentially with each new node, as illustrated in Figure 7. The complexity of conceiving an IC is also reflected in the number of design rules and fabrication steps involved in the fabrication process. Advanced technology nodes have witnessed exponential growth in design rules, indicating the increasing intricacy of IC design [125]. The complexity of masks or patterns differs from one technology to another, as shown in Figure 7. For the 65nm to 28nm technologies, Single Patten (SP) is used, while for 20nm to 14nm, Double Pattern (DP) is used. For further technologies, Tripple Pattern (PT) and Quad Pattern (QP) are used for mask formulation. The complexity of fabricating advanced ICs underscores the need for a collaborative ecosystem. These collaborations enable companies to leverage specialized expertise, access state-of-the-art fabrication processes, and effectively tackle the challenges posed by the growing complexity of IC design and fabrication.

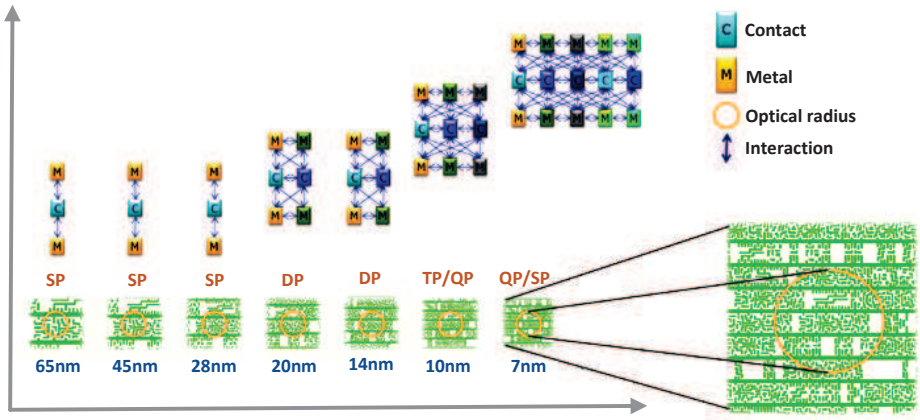


Figure 7: The complexity of device fabrication on the advanced technology nodes [126, 127].

2.3 Today's Semiconductor Industry and Trustworthy Designs

Currently, the semiconductor industry is witnessing the presence of three primary players, namely Samsung and TSMC, who are actively fabricating ICs on the advanced 5nm technology node. Intel also plans to start the fabrication on 5nm in 2023 [128]. These

players have been at the forefront of technological advancements, pushing the boundaries of semiconductor fabrication. However, the industry is already looking ahead, with plans to transition to the even more advanced 3nm technology node. The transition to the 3nm node is anticipated to occur in the coming years, with the production of ICs expected to commence in 2024 [129]. Adopting the 3nm technology node is a significant milestone to drive innovation further and shape the semiconductor industry's future. Nowadays, semiconductor companies are following a globalized IC supply chain. For instance, the design stage is mainly done in the United States, while the fabrication stage is dominated by companies in Asia, particularly Taiwan, China, and South Korea [116]. 3PIPs are developed in various locations and then integrated into a SoC for final design.

The globalized IC supply chain provides design houses with the advantage of accessing high-end semiconductor foundries. However, this supply chain also introduces inherent security threats, particularly when the layout of the design is exposed to untrusted entities. These security threats are explained in Chapter 1, emphasizing the need for robust security measures to protect the integrity and confidentiality of the design throughout the supply chain. Fabless design houses have difficulty ensuring the security of their designs from potential threats that may arise during the fabrication process at an untrusted foundry. To address these concerns, there are numerous countermeasures, but the *reconfigurable-based obfuscation techniques* has emerged as a viable solution, offering comprehensive protection against security threats.

2.4 Pre-obfuscation and Design for Security Eras

In the past four decades, reconfigurable devices like FPGAs and FPGA-based SoCs have gained widespread usage primarily as standalone solutions. However, it is only recently that the concept of reconfiguring a design has been recognized as a means of obfuscation. The evolution and utilization of reconfigurable devices can be divided into two distinct phases: the **pre-obfuscation era** and the **design for security era**, as depicted in Figure 8.

2.4.1 Pre-obfuscation Era

In 1984, Xilinx introduced the pioneering FPGA, known as the XC2064 [130]. This FPGA was featured with 64 logic cells and could be programmed using the Advanced Boolean Expression Language (ABEL) Hardware Description Language (HDL). As the capacity of FPGAs was increased and their cost was decreased in the late 1980s and early 1990s, they gained significant popularity [131]. During this period, Xilinx and Altera emerged as the leading manufacturers of FPGAs, shaping the landscape of reconfigurable devices.

Figure 9 displays the traditional island-style architecture of an FPGA. The FPGA comprises fundamental components, such as interconnect wires, Configurable Logic Blocks (CLBs), a switch matrix, and input/output (I/O) banks. Each CLB contains several logic gates that can be customized to perform specific logic functions. The I/O banks are responsible for interfacing between the FPGA and the outside world. It can be programmed to establish different routing configurations based on the application's specific requirements. A typical CLB consists of three main components: a LUT, a FF, and a multiplexer (MUX). The LUT is responsible for implementing combinational logic functions, while the FF stores the state of a sequential logic circuit. The MUX allows the selection of different inputs for the logic element. Modern FPGAs have

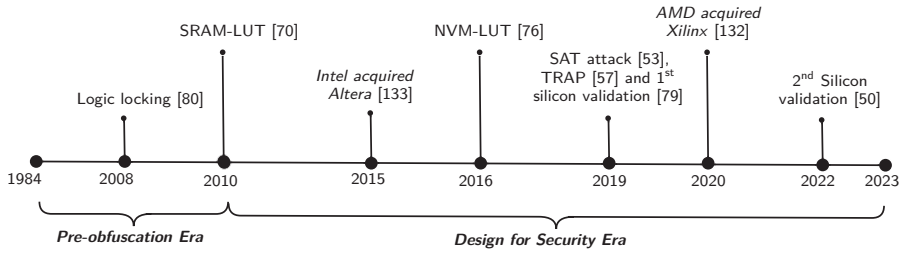


Figure 8: The transition from the pre-Obfuscation era to the security era.

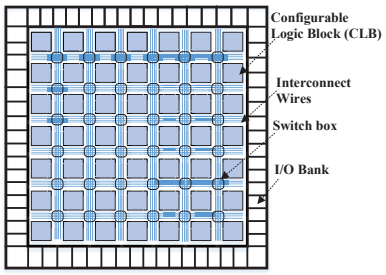


Figure 9: The conventional island-style architecture of FPGAs, adapted from [138].

evolved to incorporate more complex and non-uniform device grids, often placing I/Os in columns instead of around the perimeter [134]. Additionally, the size of LUTs in FPGAs has observed a progression over time. For instance, the Virtex 4 family of FPGAs employed 4-input LUTs, while the Virtex 5 and Virtex 6 families adopted 5-input and 6-input LUTs, respectively [135, 136]. Some companies even offer FPGAs with larger 8-input LUTs [137]. This evolution in LUT sizes enables FPGAs to support increasingly intricate and sophisticated digital logic designs. While there may be slight variations in terminology among different companies, the architecture depicted in Figure 9 provides a representative overview of an FPGA.

In the early 2000s, changes in process technology and the emergence of new application domains brought significant advancements in FPGA architecture. One notable development was the ability to partially program and dynamically reconfigure FPGA architectures. This innovation increased flexibility in digital circuit design and implementation by allowing modification of only a small portion of the fabric while the rest of the design remained operational. This feature enabled designs to be switched between on a single board, thereby enhancing design flexibility [139]. In FPGA-based SoCs, the FPGA fabric and periphery IPs can both be partially and dynamically reconfigured, offering even greater adaptability in FPGA-based SoC designs [140].

Current FPGA architectures have evolved significantly, with various modules offering functionalities, such as memory, Digital Signal Processing (DSP), Phase-locked Loops (PLLs), clocking, and networking, along with others [141, 142]. These modules are expected to continue evolving and expanding their capabilities. Notably, FPGA architectures now include large blocks, such as hardware accelerators [143]. In 2003, Xilinx created a hybrid architecture by integrating their FPGA technology into IBM's ASICs, enabling designers to directly integrate programmable logic into their designs without needing a separate FPGA board [144]. FPGA-based SoCs, which are aimed

at DSP applications, have also improved their computational power through reconfigurable solutions that often use a matrix of computational elements with programmable interconnections [145, 146, 147, 148]. It is important to note that FPGA-based SoCs are complex devices that contain more than embedded processors [149]. Additionally, they may include components like memory, I/O interfaces, accelerators, and more. A recent trend has been seen to equip FPGA-based SoCs with a Network-on-Chip (NoC) subsystem to facilitate interconnections among all the modules [150].

ASICs are designed for a specific application or purpose. They are tailored to perform a specific set of functions, making them more efficient and cost-effective than general-purpose ICs. In the last 20 years, the line between ASIC and FPGA design has become less distinct. eFPGA emerged as a small FPGA module that can be seamlessly integrated into ASIC. eFPGA IP can be licensed for usage like other IP. Designers of eFPGA IP can customize the number of logic units, DSP units, and machine learning processing units for specific applications. This approach reduces costs, increases flexibility, and shrinks eFPGA IP area. If a custom or specialized FPGA architecture is needed, it can be implemented as an eFPGA to enhance reconfigurability. IP providers offer eFPGA blocks with varying granularity and architectures [151].

2.4.2 Design for Security Era

In terms of security, there are numerous techniques available for obfuscation and LL is also a promising technique. LL emerged in 2008 [80], as highlighted in Figure 8. LL is a technique employed during the design phase to protect ICs against threats in the supply chain. It involves introducing additional logic into a circuit and securing it with key bits. Key bits are stored in a tamper-proof memory and incorporated into the locked circuit with the original inputs. The additional logic can encompass combinational elements, such as MUX, AND, OR, and XOR gates [46]. The locked circuit functions correctly and generates the expected output only when the correct value on the key bit is applied. Otherwise, its output differs from that of the original design.

For example, consider the circuit shown in Figure 10a exhibits three inputs and one output. The locked version of the circuit is demonstrated in Figure 10a and is characterized by three additional XOR/XNOR key gates. Each key gate has one input connected to a wire from the original design, while the other input, known as the key input, is driven by a key bit stored securely in the tamper-proof memory. When the correct value of the key bit is loaded into memory, all the key gates in the locked circuit, as shown in Figure 10b, function as buffers, producing the correct output for any given input pattern. However, if an incorrect value of the key bit is applied, specific key gates behave as inverters, injecting an error into the circuit. For instance, in the case of the input pattern 000 and key value 010, the key gate KG1 functions as an inverter, resulting in an incorrect output of $Y = 1$.

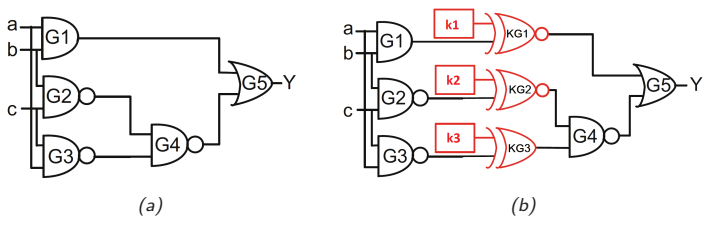


Figure 10: LL using XOR/XNOR gates [46].

During the early phase of the development of reconfigurable devices, known as the **pre-obfuscation era**, reconfigurable-based obfuscation techniques were not yet established. However, **the design for the security era** began in 2010 with the introduction of the first reconfigurable-based obfuscation technique [70]. Since then, numerous methods have been proposed to enhance the security. In recent years, the semiconductor market has seen significant acquisitions that have impacted the industry. One such event was Intel's acquisition of Altera, a leading FPGA technology provider, in 2015 [152]. This acquisition granted Intel access to Altera's state-of-the-art FPGA technology, widely used in data centers, networking, and embedded systems applications. In 2016, Menta introduced the first commercially available eFPGA IP, allowing designers to integrate a reprogrammable logic fabric into their ASIC designs easily [151]. This was a significant advancement in the field. Around the same time, a new obfuscation technique emerged, which used LUTs to hide the design and provide an extra layer of security. From 2018 to 2019, several reconfigurable-based obfuscation techniques were proposed, all utilizing LUTs to conceal circuits and bolster security. In 2019, another reconfigurable-based obfuscation technique that programmed transistors to restore circuit functionality was introduced. The first silicon demonstration for reconfigurable-based obfuscation was presented in 2019, followed by the first SAT attack in the same year [53].

In 2019, a new reconfigurable-based obfuscation called "eFPGA redaction" was introduced, representing another significant advancement. This technique employs eFPGAs as an obfuscation asset in reconfigurable-based obfuscation techniques. The concept of eFPGA redaction is illustrated in Figure 11, where one block of the ASIC is mapped to eFPGA hard IP. However, it should be noted that incorporating this level of obfuscation during the physical synthesis of a design requires expertise in physical implementation techniques that surpass the simplicity of inserting XOR/XNOR gates in a netlist, as shown in Figure 10. It is essential to mention that while the generated layout needs to be shared with an untrusted foundry for fabrication, the bitstream for the eFPGA IP remains confidential and is not shared with the untrusted foundry.

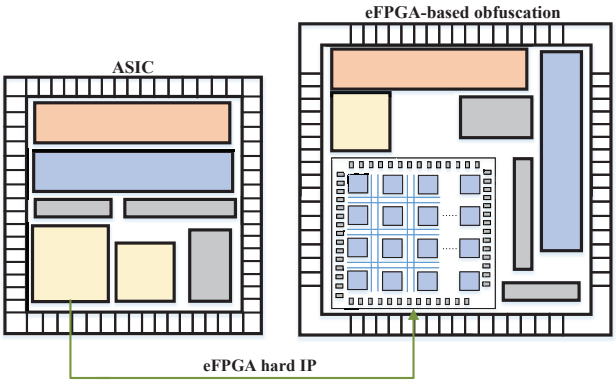


Figure 11: Understanding the eFPGA-based obfuscation technique [83].

In 2020, AMD strategically acquired Xilinx to expand its market presence and bolster its position in the High-Performance Computing (HPC) and data center markets. Xilinx's FPGAs and adaptive SoC solutions complement AMD's existing portfolio of Central Processing Units (CPUs), Graphics Processing Units (GPUs), and other accelerator technologies. Similar to Intel, this acquisition also allows AMD to exercise the FPGA technology in their ASIC chips. From 2010 to 2020, several reconfigurable-based

obfuscation techniques emerged, promising a high level of security in obfuscation. In 2022, the authors of [50] achieved the second silicon validation.

The trend of proposed techniques and attacks until 2022 is illustrated in Figure 12. These techniques aim to protect IP against a variety of hardware security attacks [70, 71, 72, 75, 76, 77, 82]. Researchers have continuously developed defense techniques, and there has been a recent surge of interest in exploring attacks on these obfuscation schemes. Despite the attempts by adversaries to break these schemes, they have had limited success, with most adversaries only able to analyze the behavior of obfuscated circuits.

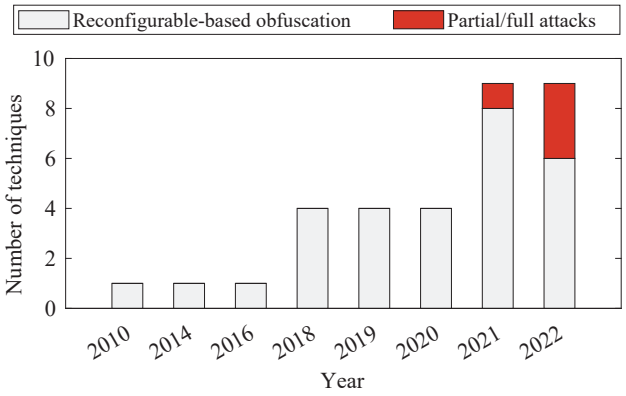


Figure 12: Publications trend for the techniques and attacks on reconfigurable-based obfuscation.

2.5 Reconfigurable-based Obfuscation Techniques

The following section delves into significant defense techniques based on reconfigurable systems. Table 2 comprises two sections: CMOS and Emerging Technologies. The techniques that utilize CMOS-based LUTs fall under the CMOS Technology (CMOS TECH) category. Conversely, Emerging Technologies (Emerging TECH) utilized for LUT implementation, such as STT, Magnetic Tunnel Junction (MTJ), SOT, and MRAM, are classified as Emerging TECH.

Let us take a closer look at how *CMOS technologies* can be used for circuit obfuscation. A recent study in [54] proposed a new approach combining reconfigurable interconnect and logic blocks to counter various attacks. While the authors presented a security analysis, they did not highlight the associated PPA overheads. Another approach in [83] involves RTL-based partitioning and eFPGA redaction to enhance obfuscation. However, a similar partitioning scheme at the behavioral level was proposed in [55, 56, 85], demonstrating an automated partitioning flow for behavioral descriptions during High-Level Synthesis (HLS). It is important to note that this technique is associated with high PPA overheads. Similarly, in [58], portions of the RISC-V control path were obfuscated using a similar approach, with its associated overhead mentioned in Table 2. Another obfuscation algorithm based on LUTs was presented in [51]. The goal is to achieve resilience against SAT attacks while minimizing overhead. A scheme called LUT-Lock was also proposed, focusing on a minimal set of primary output pins to increase obfuscation difficulty [72]. This is achieved by targeting gates with fewer connections to primary outputs and gates with less control over primary inputs, making them preferable for obfuscation.

As research progresses on obfuscation techniques, authors have been considering the balance between security and PPA. A study in [53] shows that circuits with smaller LUT input sizes, such as 2-input LUTs, are easily de-obfuscated. They emphasize that the input size of the LUT is a critical factor in achieving SAT resiliency. While LUT-based obfuscation provides high security, it also incurs significant PPA overhead, as noted in [71]. The authors of [50] have presented a digital IC design obfuscation flow that boasts low overhead and is compatible with existing EDA tools. The proposed approach is highly flexible, as demonstrated by its ability to obfuscate both non-volatile internal (eFuse) and volatile external (SRAM) LUT key configurations. The authors fabricated an IC to validate their concept and measured its design overhead regarding area, performance, and power. The chip was evaluated for different security levels, and the results indicated that the SAT attack could be effectively countered.

Table 2: Comparison of reconfigurable-based obfuscation techniques.

	Technique	Circuits	Area (%)	Power (%)	Delay (%)
CMOS TECH	eRECONF LOGIC [71]	IDU	1595.0	942.8	165.0
		LEON2	34.7	6.7	131.0
	eFPGA REDAC [81]	PicoSoC + 3×3	10.0	30.0	50.0
		PicoSoC + 4×4	30.0	60.0	80.0
		PicoSoC + 5×5	60.0	90.0	200.0
		PicoSoC + 6×6	140.0	130.0	270.0
	FINE-GRAINED eFPGA [58]	RISC-V	89.0	40.0	136.0
		GPS	39.0	46.0	0.0
	SILICON-LUT [50] †	ITC'99	LO: 7.0	LO: 0.0	LO: 0
		OpenCores	MO: 14.0	MO: 3.5	MO: 0
		PicoRV32	HO: 262.0	HO: 17.8	HO: 0
Emerging TECH (Hybrid)	Hybrid STT-LUT [76]	ISCAS	min: 0.1 avg: 6.4 max: 20.6	min: 0.7 avg: 24.9 max: 82.1	min: 0.0 avg: 28.4 max: 82.3
	MTJ-STT-LUT [78]	c2670	91.5	53.3	0.0
		c7552	91.5	20.4	0.0
		B12	60.5	18.5	0.0
		FIR	43.1	17.3	0.0
		IIR	8.4	10.1	0.0
		AES	4.9	2.8	0.0
		DES	3.3	2.5	0.0
	SOT-LUT-16i_G [77]	ISCAS/MCNC	min: 2.5 avg: 12.2 max: 25.4	–	min: 0.0 avg: 20.4 max: 41.8
	SOT-LUT-32i_G [77]	ISCAS/MCNC	min: 6.7 avg: 17.7 max: 27.2	–	min: 0.0 avg: 28.5 max: 47.5
	SOT-LUT-64i_G [77]	ISCAS/MCNC	min: 15.2 avg: 22.2 max: 27.2	–	min: 4.2 avg: 36.1 max: 78.2
	CGRRA [59]	sort	193.0	–	70.0
		cordic	492.0	–	66.0
		interp	432.0	–	170.0
		decim	147.0	–	63.0
		fft	861.0	–	34.0
		cnn	7.0	–	45.0
	TRAP [57]	AMT	4.0	0.2	6.0
		AMT+RSR+BP	9.0	0.2	83.0
		Dispatch	20.0	0.6	164.0

† Low-Obfuscation (LO), Medium-Obfuscation (MO), High-Obfuscation (HO).

It is important to note that *emerging technologies*, such as STT, MTJ, SOT, MRAM, and transistor-level configuration can be used to develop reconfigurable-based obfuscation techniques. Similar hybrid strategies can also be employed for combining switch boxes and LUTs. This starts with STT devices typically using stacked multilayer

sandwich structures.The device structure includes an oxide tunnel barrier, a free magnetic layer, and a pinned magnetic layer, which builds an MTJ [54]. The magnetization direction of the free layer can be switched from a parallel to an antiparallel state using an external magnetic field or a spin-polarized flowing through the junction. These states represent logic '1' or logic '0'. The MTJ itself does not compete with standard cells for the area since it resides between two metal layers.

Let us take an example of MRAM-based LUTs employing STT-MTJ devices and Reconfigurable Logic Interconnects (RLI)-Blocks for obfuscation. Figure 13 depicts the configuration where each cell is accessed through inputs A and B, while the write operation is controlled by the $\overline{WE}$ signal. During write operations, the MTJs in each memory cell change complementary. Based on the input signals A and B, the output O and $\overline{O}$ route to MUX using the RE and $\overline{RE}$ signals. The RIL-Blocks are constructed using commercially available STT-MTJ technology to achieve the desired obfuscation. Incorporating MTJ technology in designs requires minimal die area besides the necessary CMOS circuits and contacts for linking MTJs to CMOS transistors. Nevertheless, there are particular challenges involved in the operation of MTJs. SST structures require a high write current for magnetization switching. The asymmetry between write and read operations results in differences in operation energy and delay. To overcome these challenges, SOT devices has been explored as an alternative write approach, thoroughly discussed in [153]. This makes SOT structures promising for low-power and high-speed data storage and processing applications. Finally, the authors of [77] used hybrid SOT-CMOS circuits to implement reconfigurable logic with lower write currents for LUT programming operations, resulting in reduced hardware overhead, as indicated in Table 2.

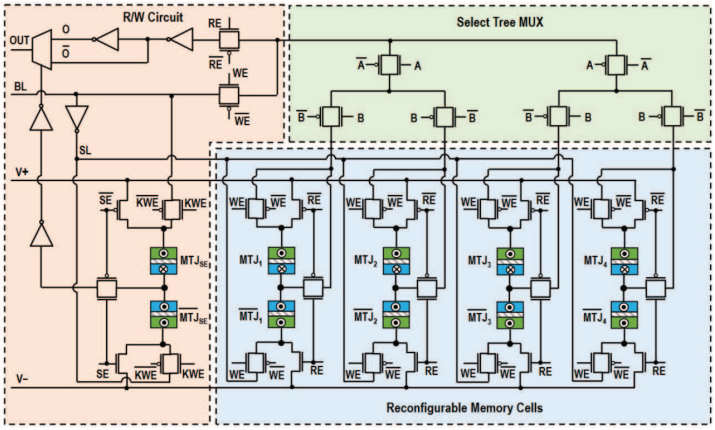


Figure 13: 2-input MRAM-based LUT that utilizes STT and MTJ technology [54].

A recent study [52] analyzed LUT-based obfuscation schemes and proposed a customized approach utilizing STT-LUT with two variants: LUT+MUX-based obfuscation and LUT+LUT-based obfuscation. The goal of this combination was to improve both logic obfuscation security and the creation of SAT resilient solution. Another study [78] investigated the design space for hybrid STT-LUT-based obfuscation, considering four critical design factors: (1) LUT technology, (2) LUT size, (3) number of LUTs, and (4) replacement strategy. The study concluded that the input size of the LUT had the most significant impact on achieving SAT resiliency. It is essential to note that incorporating hybrid technologies into the design flow requires additional parameters and processes to

be considered. This approach differs from conventional design flows since it involves additional steps, such as gate replacement and re-synthesis of the netlist.

The study conducted by [59] proposes a variant of the existing architecture known as a Coarse-Grained Runtime Reconfigurable Array (CGRRA), which selectively maps different sections of the design into a reconfigurable block. This method allows different design portions operating at separate clock cycles to be mapped onto the same CGRRA, thereby avoiding the additional area. Table 2 outlines the overall hardware overhead associated with this scheme. Noteworthy examples of this architecture include the stream transpose processor developed by Renesas Electronics [154] and Samsung's Reconfigurable Processor [155]. Another approach presented in [154] enables reconfiguration at the transistor level. This solution utilizes a "sea-of-transistor" architecture, which facilitates the implementation of custom cell libraries and supports fabric time-sharing. The authors propose a partitioning flow for RTL descriptions, yielding promising results in terms of significantly reduced PPA overhead compared to other solutions, as depicted in Table 2. However, it is important to note some drawbacks associated with this approach. Testing and simulation can become more challenging compared to alternative methods, and the configuration is performed at the transistor level, resulting in extremely large bitstream sizes [57].

A comparison between these techniques will be provided for evaluation purposes. It is crucial to acknowledge that the two categories are entirely different. Consequently, trends and comparisons within the same technology class hold more significance. Comparing different obfuscation techniques involves analyzing their PPA overheads. Table 2 provides valuable insights into how different obfuscation methods impact essential design metrics. The selected techniques in Table 2 highlight the extremes in PPA overheads and exhibit different variations of obfuscation depending on the technology and element types used.

Notably, reconfigurable-based obfuscation techniques utilizing CMOS technology tend to offer analysis for larger designs or benchmarks, showing increased PPA. However, most emerging technology-based techniques have only been evaluated on small designs or ISCAS benchmarks, revealing a significant increase in PPA [58, 71]. Some of these techniques come with substantial overheads. For instance, the authors in [75] reported an area increase of 95.06x for the c2660 circuit from the ISCAS'85 benchmark suite. Another approach in [70] suggests using LUTs for obfuscation, offering various replacement strategies to secure a netlist, although resilience against SAT attacks is not addressed. In contrast, [71] employs an SRAM-based LUT structure as configurable logic for gate replacement, incorporating n inputs, a 2^n-to-1 MUX and 2^n configuration memory cells. This approach permits the dynamic configuration of the replaced gates. However, using SRAM for logic obfuscation often results in a relatively high area overhead, as indicated in Table 2. The authors of [81] explore using an eFPGA to redact the design in PicoSoC, considering the integration of various fabric sizes. The results showed a non-linear percentage increase in PPA concerning the fabric size across all three variants. A similar approach was also implemented, but with significantly higher area and power overheads and a considerable delay overhead [58].

In the work presented in [76], hybrid-STT LUTs are proposed to achieve a relatively small area overhead. However, this technique exhibits power and delay overheads of approximately 82%. In contrast, the method proposed in [77] involves analyzing the internal gates of each class to identify the gate that can be obfuscated with minimal design overhead and path delay. The authors suggest replacing a cluster of 16, 32, or logic gates with 2, 3, and 4-input LUTs. Although the area overhead of this technique is similar to [76], the maximum overhead remains nearly the same for groups of 16, 32,

and 64 gates. Conversely, the approach described in [78] does not incur any performance overhead but results in an approximately 90% increase in area. The technique presented in [59] is also validated on small circuits, but even for a small circuit executing a sorting algorithm, it incurs a large area overhead. The PPA overheads in [57] and [79] are minimal, and these methods exhibit the lowest PPA among reconfigurable-based obfuscation techniques. However, it is essential to note that these approaches have only been validated on small circuits.

2.6 Classification of Reconfigurable-based Obfuscation Techniques

As the interest in utilizing reconfigurable logic for obfuscation techniques grows within the research community, it is essential to establish a standardized classification and terminology for this approach. A comprehensive classification framework based on three critical factors has been developed: technology utilized, element type, and IP type. Figure 14 illustrates this framework.

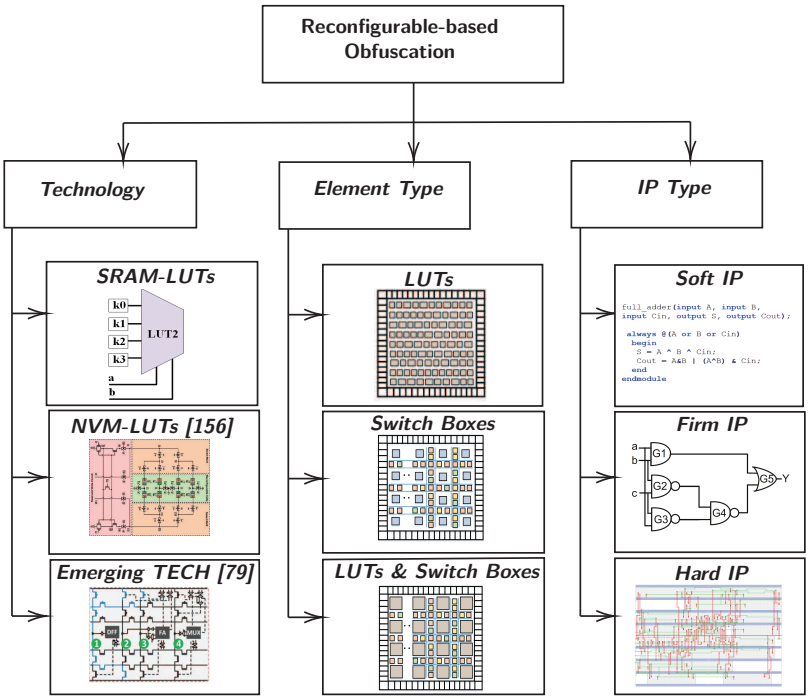


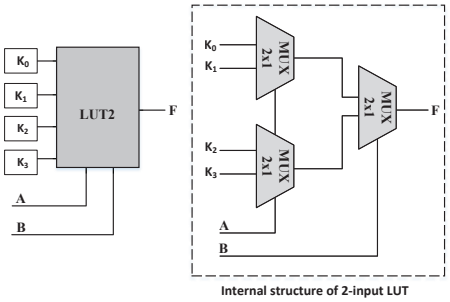
Figure 14: Classification of Reconfigurable-based obfuscation.

2.6.1 Technology

Many reconfigurable-based obfuscation techniques have been proposed, with LUTs playing a crucial role in facilitating logic obfuscation through reconfigurability. During the obfuscation process, specific internal gates from the design are mapped onto LUTs. As shown in Figure 14, various technologies are available to store the essential bits of these LUTs, which can be categorized into three distinct groups. These categories include SRAM-based LUTs [50, 51, 70, 71, 72], NVM-based LUTs [52, 53, 54, 75, 76, 77, 78], and emerging technologies [55, 56, 57, 58, 59, 79, 80, 82, 81, 83, 84, 85]. The SRAM-

based LUTs have been the most widely used technology for programming LUTs. However, NVM-based LUTs have gained popularity due to their ability to provide better security. The category of emerging technology encompasses various technologies for programming the LUTs, including STT-based LUTs that utilize magnetic technology, FF-based LUTs, individual transistor programming, such as in TRAP fabric, and programming of eFuses.

The SRAM-based LUT has been a popular technology for storing key bits due to its desirable characteristics, such as programmability, reconfigurability, fast access time, low power consumption, small area, scalability, and ease of testing. These features make it an ideal choice for implementing logic functions in FPGA designs, but they are also suitable for obfuscation purposes. A 2-input LUT is illustrated in Figure 15, showing the various functions it can potentially implement. With its two inputs, it can realize 16 distinct functions, as listed in the table presented in Figure 15.



K_3	K_2	K_1	K_0	F	Gate
0	0	0	0	0	TIE0
0	0	0	1	A′B′	NOR
0	0	1	0	AB′	INR
0	0	1	1	B′	INV
0	1	0	0	A′B	INR
0	1	0	1	A′	INV
0	1	1	0	AB′+A′B	XOR
0	1	1	1	A′+B′	NAND

K_3	K_2	K_1	K_0	F	Gate
1	0	0	0	AB	AND
1	0	0	1	A′B′+AB	XNOR
1	0	1	0	A	BUF
1	0	1	1	A+B′	IND
1	1	0	0	B	BUF
1	1	0	1	A′+B	IND
1	1	1	0	A+B	OR
1	1	1	1	1	TIE1

Figure 15: The internal architecture of the 2-input LUT and all the possible logic functions based on its configuration bits K_0-K_3.

NVM-based LUTs are implemented using NVM technology. The key benefit of NVM-based LUTs is that they can retain programming even when the device is powered off. However, they have downsides compared to SRAM-based LUTs, such as slower access times and limited, complex programmability. Conversely, NVM-based LUTs offer high-density storage elements, with STT-based LUTs being a promising option for creating highly robust and reverse-engineering resistant LUTs.

In contrast, FF-based LUT implementation makes the framework technology-agnostic, simplifying the floorplanning and placement processes significantly. However, it should be noted that FF-based LUTs do not achieve the same bit density level as SRAM-based solutions. The TRAP fabric is a unique method for obfuscating the design's intent by programming numerous transistors. On the other hand, eFuses are one-time programmable fuses that are permanently programmed with a specific LUT configuration. This contrasts with SRAM-based LUTs, which need reconfiguration every time the device powers up. Although eFuses offer distinct security properties by disallowing reprogramming, they also potentially expose the programmed values to reverse engineering by end-users (although not by the foundry).

2.6.2 Element Type

When it comes to reconfigurable-based obfuscation techniques, one way to categorize them is by the type of components they use. Some approaches solely rely on LUTs for obfuscation purposes, as demonstrated in various studies [50, 52, 53, 70, 71, 72, 75, 76, 77, 78]. These methods mainly focus on obfuscating the logic elements by using LUTs alone. An example of this approach is illustrated in the top center of Figure 14, where the circuit consists mainly of LUTs arranged in a regular structure. On the other hand,

there are obfuscation techniques that exclusively rely on exploiting switch boxes [79, 81]. These methods alter the connections between elements to obfuscate the design, as shown in the center of Figure 14. In this case, the layout combines switch boxes and standard logic, with the switch boxes highlighted in blue. Lastly, some techniques use a combination of both LUTs and switch boxes, known as FPGA redaction techniques [51, 54, 55, 56, 57, 58, 59, 80, 82, 83, 84, 85].

Section 2.4.1 describes an instance of reconfigurable-based obfuscation that employs eFPGA. It seems that obfuscating both LUTs and switch boxes can offer additional benefits in augmenting the design's security. By concealing routing blocks, attackers cannot scrutinize and derive functionality from routing patterns. This provides another opportunity to foil powerful SAT attacks.

2.6.3 IP Type

The implementation of reconfigurable-based obfuscation techniques requires additional steps in the design flow. Designers select critical modules to be redacted or identify suitable gates to replace using LUTs. Figure 14 shows that the obfuscation process can be performed at different IP levels, namely soft IP, firm IP, or hard IP. Soft IP obfuscation involves modifying the RTL code, such as Verilog or VHDL, or high-level codes like C/C++, through user-defined algorithms that identify and obfuscate specific portions of the modules [55, 56, 58, 59, 82, 84, 85]. The most common approach is the firm IP obfuscation, which utilizes the post-synthesis netlist file as input and generates the obfuscated netlist as output [50, 51, 52, 53, 54, 57, 58, 70, 71, 72, 75, 76, 77, 78, 79, 80, 83]. Lastly, hard IP obfuscation involves identifying a critical part of the design and mapping it into a hard IP, utilizing eFPGAs [81, 83] to perform the obfuscation. An example of this approach is illustrated in Figure 11 and explained earlier in Section 2.4.2.

2.7 Security Analysis: Threat Models and Existing Attacks

This section will initially describe LL and the well-known SAT attack and then, it will thoroughly review the recent attacks and their corresponding threat models. Regarding adversarial modeling, there are two types of threat models: oracle-guided and oracle-less. In an oracle-guided attack, an adversary has the reverse-engineered locked netlist and a functional IC, which acts as an oracle. Obfuscation techniques are vulnerable to oracle-guided SAT attacks and their variants, which are quite prevalent [64, 65, 66, 157, 158]. Reconfigurable-based obfuscation can also be vulnerable to these attacks.

2.7.1 LL and SAT Attack

The oracle-guided SAT attack starts by creating two versions of the locked circuit, namely L_{A} and L_{B}, by using different key bits, K_{A} and K_{B} respectively. Then, it generates the miter circuit, which checks if there is a difference between the outputs of the two circuits, as shown in Figure 16. In this figure, the primary inputs (I) are shared between the two locked circuits and the $diff$ output is generated by XORing the corresponding outputs of the two circuits and then ORing them. Algorithm 1 describes the SAT attack, which iteratively finds the Distinguishing Input Pattern (DIP) to eliminate incorrect keys.

The SAT attack begins by identifying the Conjunctive Normal Form (CNF) formulas of L_{A} and L_{B}, before generating the function F_1 on line 2. It then enters a loop on lines 3-7, whenever there is a satisfiable solution on the conjunction of F_{i} with the CNF formula of the miter circuit. This satisfiable solution means that there exists a DIP I_{d},

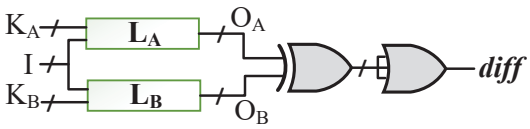


Figure 16: Circuit employed by the SAT attack.

which causes circuits L_{A} and L_{B} to generate incorrect outputs. The SAT assignment on line 4 is used to extract this DIP, which is then applied to the oracle (R) to obtain the output O_{d}. The CNF formula F_{i} is updated with the additional information obtained when O_{d} and I_{d} are applied to circuits L_{A} and L_{B} on line 6. The loop continues until no more DIPs are found. The correct key K_{C} is found as an assignment to K_{A} that satisfies the formula F_{i} on line 8. Note that while the SAT attack and its variants are powerful techniques, they may face issues with circuits that are locked by a large number of key bits, such as reconfigurable-based obfuscation techniques.

Algorithm 1: SAT attack [64]

Input : Locked netlist L, Oracle R
Output : Correct key K_C

1 $i \leftarrow 1$;
2 $F_1 \leftarrow L(I, K_A, O_A) \wedge L(I, K_A, O_B)$;
3 **while** $SAT[F_i \wedge (O_A \neq O_B)]$ **do**
4 $\quad I_d \leftarrow \mathsf{SAT\_ASSIGNMENT}_I[F_i \wedge (O_A \neq O_B)]$;
5 $\quad O_d \leftarrow R(I_d)$;
6 $\quad F_{i+1} \leftarrow F_i \wedge L(I_d, K_A, O_d) \wedge L(I_d, K_B, O_d)$;
7 $\quad i \leftarrow i+1$;
8 $K_C \leftarrow \mathsf{SAT\_ASSIGNMENT}_{K_A}(F_i)$;
9 **return** K_C;

In an oracle-less attack, an adversary has only the locked netlist. Structural analysis is the foundation of several oracle-less attacks, as mentioned in [159, 160, 161]. Currently, three attacks on reconfigurable-based obfuscation have been developed and presented in [162, 163, 164], which will be described in the following sections.

2.7.2 Predictive Model Attack

This attack replaces the precise logic implemented on eFPGAs with a synthesizable predictive model. This oracle-guided attack uses machine learning techniques to construct a predictive model that aims to replicate the behavior of the original logic [162].

Let us discuss the threat model of "predictive model attack". In this scenario, the adversary is a skilled IC designer with the necessary knowledge and tools to understand the layout representation. The threat model presented in [162] is outlined below:

- An eFPGA's input and output can be accessed by an adversary through the scan-chain that encircles it. FPGA companies commonly utilizes this technique in their commercial IPs.
- When a scan-chain is absent, an adversary may opt for a probing attack as a viable alternative [165]. Due to the predictable configuration of the eFPGA, a

probing attack can be executed with a relative ease.

- Given that the eFPGA is typically obtained through licensed companies, such as Achronix, Menta, or Quicklogic, it is reasonable to assume that the adversary can access the CAD flow necessary for programming it.

The process of a "predictive model attack" involves three key stages, as shown in Figure 17. The first phase requires using the IC procured from the market to execute applications. During this phase, the inputs and outputs of the eFPGA are recorded to generate a predictive model, and the latency of the obfuscated design is measured to ensure that there are no timing issues when substituting the eFPGA part with a predictive model. The second phase of the attack involves searching for a suitable predictive model that can be mapped onto the eFPGA hardware. To meet specific criteria, such as fitting within the eFPGA fabric, producing outputs within the predefined error threshold, and operating at the same frequency and latency as the original design, this phase encompasses three steps: model fitting, predictive model refinement, and automated multi-layer perceptron exploration. During model fitting, a predictive model, either linear regression or multi-layer perceptron, is trained to approximate the behavior of the original design. The model is adjusted to optimize its accuracy and fit within the constraints of the eFPGA. In the predictive model refinement step, further optimization is performed to obtain the smallest possible model while operating within the specified error threshold. The automated multi-layer perceptron explorer plays a crucial role in the third step. It fine-tunes the multi-layer perceptron configuration to ensure that it fits within the constraints of the eFPGA and meets the required error threshold. The outcome of this phase is a synthesizable C description of the predictive model that fulfills all the given constraints and accurately approximates the behavior of the original design.

The final phase of generating a predictive model for HLS involves obtaining the smallest possible implementation of the predictive model with a latency equivalent to that of the exact version extracted in the initial phase. This is achieved by exploring various combinations of synthesis options for the optimized synthesizable predictive model and generating the most compact implementation with a latency of eFPGA (L_{efpga}). The authors discuss the use of synthesis directives (pragmas) for synthesizing arrays, loops, and functions and how different combinations of these directives result in a unique microarchitecture with specific trade-offs between area and latency. The outcome of this phase includes the pragma combination that yields the smallest predictive model implementation (pragmaopt) and the newly optimized predictive model with the exact latency as the obfuscated circuit (C_{opt}). The final phase involves generating an eFPGA bitstream to configure an eFPGA with the predictive model. This includes HLS, logic synthesis, technology mapping, place and route, and eFPGA bitstream generation. The target synthesis frequency (f_{max}) should match the frequency at which the exact obfuscated circuit operates, enabling the unlocking of every fabricated IC. However, the attack has a significant limitation as it applies only to approximate computing [162].

2.7.3 Break & Unroll Attack

Another attack, known as the "Break & Unroll attack" has been introduced in the literature to recover the bitstream of eFPGA-based redaction schemes efficiently, even when dealing with hard cycles and a large number of keys [163]. This has challenged the common belief that eFPGA-based redaction schemes are secure against oracle-guided attacks, as demonstrated by various studies. In this proposed threat model of

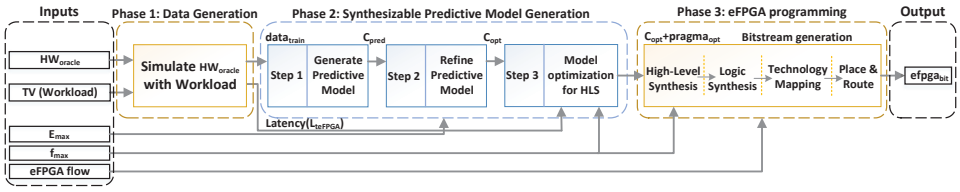


Figure 17: The three primary phases of predictive model attack, adapted from [162].

[166, 58, 81, 167, 74], it is assumed that all ICs are sequential circuits, allowing the attacker to access the scan-chain, which is always present, and the obfuscated netlist. As highlighted in Algorithm 2, the Break & Unroll attack encompasses two primary phases: cycle breaking and unrolling.

During the breaking phase, all cycles are disrupted, and a non-cyclic condition is introduced as a new constraint to the obfuscated circuit. On line 2, all feedback signals of the circuit were searched and stored in a set called W. Subsequently, all cycles are broken one by one, and all the broken feedback signals are accumulated into a CNF on line 4. This CNF is added as a new constraint to the obfuscated circuit on line 5. A new obfuscated circuit is then introduced, which is expected to have no structural cycles on line 6. After adding the constraint, the original SAT solver is run on this new circuit version on line 8. However, if a cycle is missed, the SAT solver may get stuck in an infinite loop. In order to avoid this, the $LoopDetected$ function is introduced on lines 10-12, which recognizes whether running the first part of the Break & Unroll algorithm results in an infinite loop or not. In this function, the new DIP is compared with all members of a set that keeps all DIPs from prior iterations on lines 13-15. If the new DIP does not belong to this set, it is demonstrated that the breaking phase operated correctly, and the correct key will be revealed in the next step on lines 16-17. However, if the newly generated DIP is found in the set, it indicates that the SAT-solver will go into an infinite loop. Therefore, in the second phase, this issue needs to be addressed.

The second phase, unrolling, is employed to mitigate the impact of hard cycles if the first stage fails to unveil the correct key. The unrolling phase addresses the limitations of the breaking phase by sequentially unrolling one cycle at a time. This involves selecting a single feedback, duplicating every gate in the circuit, and creating a new version of the obfuscated circuit. The set W, which represents the current state of the attack algorithm, must be updated after each cycle unrolling. The study aims to demonstrate the vulnerabilities of eFPGA-based redaction schemes to underline the importance of proactive measures to strengthen their security. In general, this attack exploits scan-chain, oracle, and SAT attack, and can recover the bitstream of less than 2000 key bits. The results of this attack show UNSAT for a small circuit with 1134 key bits. The SAT attacks are applicable but fail to handle the circuit with large key bits, such as 100K, as highlighted in the motivation of the thesis.

2.7.4 FuncTeller Attack

A recent attack named "FuncTeller" on eFPGA-based obfuscation has successfully retrieved the hardware IP with only black-box access to a programmed eFPGA. The attack leveraged the effect of modern EDA tools on practical hardware circuits and used this observation to guide the attack [164]. The threat model described later is consistent with previous works on eFPGA-based obfuscation, attacks on eFPGA [162, 163], and the Cybersecurity Awareness Worldwide Competition (CSAW) [168].

Algorithm 2: Break & Unroll attack algorithm [163]

Input : Obfuscated circuit $g(x,k)$ and original function $f(x)$
Output : Key vector k^* such that $g(x,k^*) \equiv f(x)$

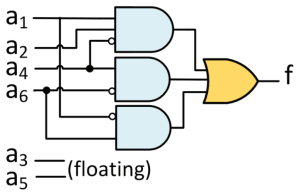
while *(True)* **do**
  $W \leftarrow$ SearchFeedbackSignals$(g(x,k))$
  // $W \leftarrow \{w_0, w_1, \ldots, w_m\}$
  for $w_i \in W$ **do**
    $F(w_i, w_i') \leftarrow$ BreakFeedback(w_i)
    $NCCNF(k) \leftarrow \bigwedge_{i=0}^{m} F(w_i, w_i')$
    // $NCCNF(k) \leftarrow$ `BreakFeedback`$(w_0) \wedge \cdots \wedge$ `BreakFeedback`(w_m)
    $g(x,k) \leftarrow g(x,k) \wedge NCCNF(k)$
  $DIPset \leftarrow \emptyset$
  while $\hat{x} \leftarrow$ *SAT*$(g(x,k_1) \neq g(x,k_2))$ **do**
    if *LoopDetected*$(\hat{x}, DIPset)$ **then**
      $w \leftarrow$ SelectFeedbackSignal(W)
      $g(x,k) \leftarrow$ NewCircuit$(w, g(x,k))$
      break
    $g(x,k_1) \leftarrow g(x,k_1) \wedge (g(\hat{x},k_1) = f(\hat{x}))$
    $g(x,k_2) \leftarrow g(x,k_2) \wedge (g(\hat{x},k_2) = f(\hat{x}))$
    Add$(\hat{x}, DIPset)$
  if $\neg$*SAT*$(g(x,k_1) \neq g(x,k_2))$ **then**
    return $k^* \leftarrow$ SAT$(g(x,k_1))$

The attackers in this model are a collusion of an untrusted foundry/testing facility and an untrusted end-user. The attacker in the foundry/testing facility can access a netlist with the unprogrammed eFPGA and can isolate the eFPGA by analyzing the IC netlist or identifying the scan-chain connected to the eFPGA using reverse engineering [169]. Note that the purchased functional chip is ASIC integrated with the configured (loaded with a certain bitstream) eFPGA. This threat model is presented in detail in [164], which is given as follows:

- One way for an attacker to isolate the eFPGA from the rest of the design is to access the dedicated scan-chains of eFPGA, which is a feature commonly supported by eFPGA vendors [162]. Another possible method is to perform a probing attack, where the attacker locates and accesses the ASIC's internal signals to control or observe the eFPGA's inputs and outputs [162, 165].
- The extraction of hardware IP through side-channel attacks or bitstream extraction is forbidden. Various schemes have been suggested over time to reduce the risks associated with these types of attacks [170, 25, 171, 172, 26].
- Once the eFPGA is isolated, an attacker can bypass scan-chain protections to enable scan-chain access [173, 174, 175, 176]. This enables the attacker to query the hardware IP design through I/O pins accessed via scan-chains. More information on the methods used to unlock or enable scan-chain access, including attacks on scan-chain protections, is presented in [173, 174, 175, 176].

In [164], a security evaluation of eFPGA-based obfuscation techniques is presented. The attack aims to recover an IP that is implemented on an eFPGA with only I/O access.

In practical circuits, logic synthesis optimizes PPA by reducing the number of Prime Implicants (PIs) and literals in the circuit's Prime Implicants Table (PIT). As a result, ON-set minterms are clustered into multiple PIs. This behavior is consistent in hardware designs and has two implications. Firstly, a single PI covers multiple ON-set minterms that share the same literals in the PI representation. The authors utilize this property to predict each PI by expanding from a discovered ON-set minterm (seed). Each value is replaced by a don't care and heuristically verified in this process. Secondly, the hamming distance between any two PIs in a PIT is usually much smaller than the input size. This property reduces the search space when updating the predicted PIT with the new PI. Generating a new PI requires the discovery of the next ON-set minterm. Thus, the search space for the next ON-set minterm is limited to being close to the current PIs. Utilizing the implications of circuit cones, FuncTeller employs a mechanism demonstrated in Figure 18 to implement boolean functions. The circuit cone has n inputs $(a_1, a_2, \cdots, a_n)$ and one output, and the example circuit in Figure 18a implements the Boolean function f with n $= 6$. The boolean function, $f(a_1, a_2, \cdots, a_6) = a_1 a_2 \overline{a_4} + a_4 \overline{a_6} + \overline{a_1} a_6$, is non-canonical in its PIT representation, displayed in Figure 18b.



(a)

a_1	a_2	a_3	a_4	a_5	a_6	f
1	1	-	0	-	-	1
-	-	-	1	-	0	1
0	-	-	-	-	1	1

(b)

Figure 18: An example of a circuit and the PIT [164].

Algorithm 3 describes the recovery of the entire circuit. The algorithm aims to predict the entire functionality of a circuit based on an input oracle and various parameters. It takes the oracle O, which represents the circuit to be predicted, and several parameters: distance parameter d_0, linear parameter p, convergence parameter p_{conv}, and a time limit T. The algorithm's goal is to construct the predicted circuit C_{pred} that represents the entire functionality of the given circuit O. To achieve this, it utilizes several helper functions. On line 1, the main function, *predict_circuit*, begins by determining the number of outputs in the circuit using the function *count_number_of_outputs*, and stores this value in the variable $output\_size$. It initializes an empty set $Cones_{pred}$ to hold the predicted cones. On line 2, the algorithm iterates over each circuit output, represented by variable w, from 1 up to $output\_size$. On line 5, the function *predict_cone* is called to predict the cone corresponding to each output. A cone represents a part of the circuit associated with a particular output. Within *predict_cone*, the algorithm uses the distance parameter d_0, the linear parameter p, the convergence parameter p_{conv}, and the time limit T to perform the prediction. The result, denoted by PIT_{pred}, is a Predicted Information Table representing the cone's functionality. The PIT is then converted to a netlist representation using the function *convert_PIT_to_netlist* on line 6. The netlist describes the cone's structure and behavior, which is stored in the variable $Cones^{w}{}_{pred}$. The algorithm combines the predicted cones in the set $Cones_{pred}$ on line 7 to update the overall predicted circuit. To construct the entire predicted circuit C_{pred}, the function *merge_cones_to_circuit* takes this set as input

on line 8. Finally, on line 9, the algorithm outputs the predicted circuit C_{pred}.

Algorithm 3: Predicting entire circuit's functionality [164]

Input : Oracle O, distance parameter d_0, linear parameter p, convergence parameter p_{conv}, time limit T
Output : Entire predicted circuit C_{pred}

```
1 Function predict_circuit(O, d_0, p, p_conv, T):
2     output_size := count_number_of_outputs(O);
3     Cones_pred := ∅;
4     for w ← 1 to output_size do
5         PIT_pred := predict_cone(O, w, d_0, p, p_conv, T);
6         Cones^w_pred := convert_PIT_to_netlist(PIT_pred);
7         Cones_pred := Cones_pred ∪ Cones^w_pred;
8     C_pred := merge_cones_to_circuit(Cones_pred);
9     return C_pred;
```

This attack leverages the aforementioned threat model to predict the bitstream of eFPGA. However, for some circuits like c1355 and c1908, the attack could not find the key bits. Additionally, the attack assumes that the eFPGA IP can be easily separated from the obfuscated design. While these attacks aim to break eFPGA-based obfuscation, none can completely recover the bitstream. Instead, they predict the bitstream, and there is a probability that the predicted bitstream may be incorrect to some degree.

2.8 Secure Bitstream of Reconfigurable-based Obfuscation

The security of bitstreams has become crucial to protecting ICs deployed in various products. The bitstream, which contains the configuration information of an obfuscated IC, can be vulnerable to attacks by adversaries seeking to exploit the design's IP or gain unauthorized access to sensitive information. To counter these threats, cryptographic techniques are employed to safeguard the bitstream's security. One of the primary concerns regarding bitstream security is the potential for adversaries to leverage an oracle and reverse engineer the bitstream. In this context, encryption involves the generation of a robust key for encryption. To establish strong encryption, a robust key generation mechanism is essential.

PUFs are hardware-based security components that exploit inherent physical variations within integrated circuits. Since PUF responses are unique to each IC, they prevent the cloning or tampering of secret key. These variations are unique to each IC, providing a reliable and unclonable identity. PUF leverages these unique characteristics to generate cryptographic keys, serving as a foundation for secure key generation. PUFs exploit process variation (e.g., gate oxide thickness, size, and threshold voltage) that occurs naturally during the fabrication of ICs. Although the circuits are fabricated with identical layouts, every transistor presents slightly random electric properties that generate a unique identity [177]. By utilizing PUFs as a root of trust for key generation, a strong foundation is established for the encryption of bitstreams [178]. This enhances the overall security of the encryption process and ensures that only authorized entities with access to the correct PUF response can decrypt and access the bitstream.

PUFs generate a set of Challenge-Response Pairs (CRPs) that can be classified into two categories: extensive PUFs and confined PUFs [179]. Extensive PUFs provide exponential CRPs, whereas confined PUFs offer only one or a few CRPs. Different types of PUFs can be classified into various groups, including Ring Oscillator-based

PUF (RO-PUF) [96], arbiter PUF [97], DRAM PUF [98], and SRAM-based PUF [99, 100, 101, 102, 103, 104, 105, 106, 107]. Examples of confined PUFs include SRAM-based PUFs, such as those mentioned in [99, 100, 101, 102, 103, 104, 105, 106, 107]. These types of PUFs are commonly used for storing unique identifiers or long-term secret keys [180]. On the other hand, an extensive PUF can accept multiple challenges and generate a 1-bit response to an n-bit challenge, inducing a random n-variable Boolean function from a computational perspective. Examples of extensive PUFs include RO-PUFs [96] and arbiter PUFs [97]. Extensive PUFs are typically utilized for challenge-response authentication [178].

The digital signature of an SRAM-based PUF is the raw entropy of the SRAM array that is converted into digital bits. SRAM-based PUFs offer a combination of simplicity, low cost, high reliability, and scalability [94]. Additionally, SRAM-based PUFs rely on standard SRAM IP that is commonly available to designers, and the same memory macro utilized for storage can also serve as a PUF. There is no need to customize the memory macro, and the internal architecture of 6T-SRAM cell is illustrated in Figure 19.

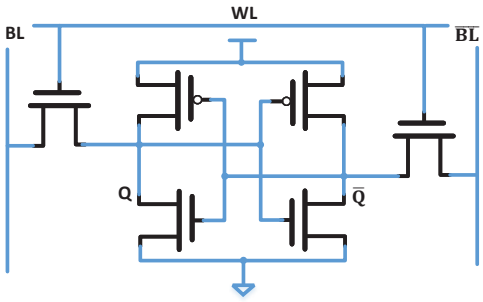


Figure 19: The internal architecture of 6T-SRAM bitcell, adapted from [181].

In a standard 6T SRAM, each bitcell comprises six transistors, including two cross-coupled CMOS inverters and two access transistors, as given in Figure 19. The control signal to access the SRAM bitcell is line WL. During read and write operations, bit lines BL and $\overline{BL}$ carry data. Two signals, namely Q and $\overline{Q}$ are internal signals and one of them becomes output when driving the bit lines BL and $\overline{BL}$. The four transistors placed symmetrically in Figure 19 form bistable inverters. These inverters are symmetrically designed to match size, but there may be mismatches due to random variations during fabrication. These mismatches can be used by SRAM-based PUFs, which take advantage of biases in each SRAM bitcell towards a logic '0' or logic '1' when powered up. Since CMOS devices have different physical parameters during fabrication, such as doping levels and transistor oxide thickness, these variations can affect the power-up state of associated bitcells in an SRAM. Some bitcells may strongly prefer a logic '0' or logic '1' state upon power-up, while others are neutral and power up randomly due to system noise. The cells that strongly prefer a logic '0' or logic '1' state are more useful for PUF response. It has been shown in [108] that not all platforms can function as PUF. By examining the power-up state of an SRAM, a unique identifier can be created since the process variations and preferences are truly random and dependent on physical anomalies. The process of generating a key starts by extracting the PUF response, which is a distinct and unpredictable value derived from the physical characteristics of the IC, as demonstrated in Figure 20. This response is utilized to generate secret keys that are specific to the cryptographic algorithms. The

uniqueness and randomness of PUF responses contribute to the strength and security of the generated keys.

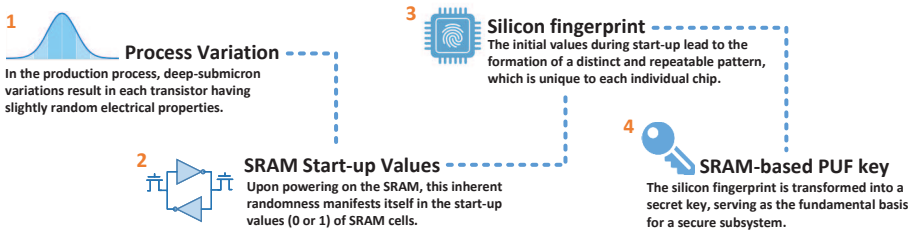


Figure 20: The procedure to harvest unique signature from SRAM-PUF [182].

2.8.1 Robustness of SRAM-based PUF

The output of SRAM-based PUF must stay consistent even when voltage and temperature conditions change [183]. SRAM bitcells can be split into two groups based on their power-up value, which depends entirely on the strength between the two cross-coupled inverters within the SRAM bitcell [184]. Below are the two types of bitcells for SRAM-based PUF:

- Neutral bitcell: The strength difference between two cross-coupled inverters in an SRAM bitcell is minimal. The power-up value of a bitcell may rely on measurement noise and random logic '0' and logic '1' states, as illustrated in Figure 21.
- Skewed bitcell: While powering up an SRAM bitcell, the strength difference between two cross-coupled inverters is crucial in determining whether a logic '0' or logic '1' is produced. However, some cells may have little process variation, resulting in a weak logic '0' or logic '1'. The threshold voltage (V_{th}) of transistors is the most significant factor in determining the start-up value of an SRAM bitcell, as reported in [185]. As the CMOS technology node shrinks, intra-die variability increases, leading to process variability. This variability in fabrication processes results in an increased difference in strength between two cross-coupled inverters, which improves the quality of SRAM-based PUF. These partially skewed cells are suitable for True Random Number Generator (TRNG) and PUF applications in identification where errors can be tolerated. However, power-up values of these bitcells can be impacted by measurement noise, temperature and voltage fluctuations, and aging. On the other hand, a strong mismatch between cross-coupled inverters in an SRAM bitcell may produce a strong logic '0' or logic ''1', as depicted in Figure 21.

2.8.2 Evaluation Metrics for SRAM-based PUF

The quality of an SRAM-based PUF can be defined by its reliability, entropy, uniqueness, and randomness. The *reliability* metric is a key characteristic that defines the ability of a PUF to consistently reproduce its output response, independent of temperature variations and fluctuations in operating voltage. The SRAM-based PUF must generate the same response at all operating conditions in every power-up cycle during its entire lifetime. The reliability of its PUF can be assessed by evaluating its With-in Class Hamming Distance (WCHD), which is the fractional hamming distance between measurements

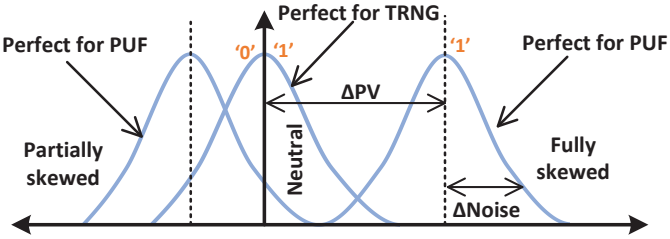


Figure 21: Characteristic of SRAM bitcells on power-up state, adapted from [183].

taken at various conditions during reconstruction and a reference measurement taken at enrollment. Another reliability-related metric is Within Class Sequential Hamming Distance (WCSHD), the Hamming distance between the consecutive responses of the same SRAM-based PUF, defined in [186]. The WCHD can be calculated by Equation 1. Where R_j denotes the reference n-bit response extracted at the normal operating condition (i.e., room temperature and the normal supply voltage) for a chip j. $\overline{R}_{j,s}$ indicates the response at normal operating conditions s, and x is the total number of responses for the PUF.

$$\mathrm{WCHD} = \frac{1}{x}\sum_{S=1}^{x}\frac{HD\left(R_j, \overline{R}_{j,s}\right)}{n} \times 100\% \tag{1}$$

For *entropy*, one important parameter is a bias pattern linked with the PUF's characteristics. Understanding the origin of *bias pattern* is crucial for improving the reliability and security of SRAM-based PUFs. SRAM-based PUFs can be attributed to fabrication variations, temperature changes, or other factors. This requires a comprehensive analysis of their response data. In particular, the response data shows that a set of bitcells within the output response exhibit a consistent bias towards a logic '0' or logic '1', which cannot directly be assessed by fractional hamming weight. This is efined by the percentage of ones in the raw output response. This recurrent phenomenon is usually referred to as the bias pattern. SRAM-based PUFs leverage *entropy* from process variations to build various kinds of unique fingerprints for each identically fabricated chip. Entropy in SRAM-based PUF is derived by inserting the Masked Hamming Weight (MHW) into min-entropy formula and MHW is calculated through the percentage of ones in the output response after an XOR operation with the bias pattern [187]. The mathematical form for the MHW and min-entropy is presented in Equations 2 and 3.

$$MHW = \frac{1}{n}\sum_{i=1}^{n} R_i \oplus m_i \tag{2}$$

$$H_{\min} = -\log_2(\max(MHW, 1-MHW)) \tag{3}$$

The value of the SRAM-based PUF's response (R) on bitcell i is denoted as R_i. This value is derived with the bias pattern (m) value on bit position i. Both the SRAM-based PUF response and the bias pattern have a length of n bits. The metric of *uniqueness* is a determination of the capability of an SRAM-based PUF to produce unique responses across multiple chips. Different SRAM-based PUFs must generate different responses to a given challenge to separate one from another. Different chips may produce nearly

identical PUF responses due to systematic variations, and this is measured by bias. The concept of uniqueness can be quantitatively expressed as a Between-Class Hamming Distance (BCHD) in Equation 4. Where R_i and R_j represent the responses produced by two different chips (i, j) upon the application of the same challenge. n represents the length of the response, and N signifies the total number of chips. In an ideal scenario, the uniqueness, quantified by the BCHD, should be equal to 50%.

$$\text{BCHD} = \frac{2}{N(N+1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{HD(R_i, R_j)}{n} \times 100\% \tag{4}$$

The metric of *randomness* defines the random response generated by a PUF regarding the probability distribution of the logic '0' and logic '1' states. Ideally, the randomness should be balanced at 50%, meaning the probability of obtaining a logic '0' response must be equal to the probability of obtaining a logic '1' response. Randomness also helps to determine whether a PUF is biased or not. For an unbiased SRAM-based PUF, changing one bit in a challenge or address of SRAM should alter approximately half of the response bits.

3 A Security-aware CAD Flow for the Obfuscation Method

This chapter presents a security-aware CAD flow for the proposed obfuscation method. It highlights the architectural aspects and the underlying principles of the algorithm used in the custom CAD tool. Furthermore, the chapter discusses the trade-offs between design and security, highlighting the need for the balance required in obfuscated designs. The results and findings of various obfuscated designs provide insights into their effectiveness and performance.

3.1 Design Obfuscation Concept

The section provides the design obfuscation concept utilizing reconfigureable-based obfuscation. Figure 22a shows the ASIC, a one-time placement that offers best-in-class performance. Let us focus on FPGA as highlighted in Figure 22d. The fabric of an FPGA device typically includes multiple reconfigurable blocks. FPGA is a flexible device that is fully obfuscated hardware, but it incurs performance penalties. The CLB consists of LUT, FF, MUXes, and Carry blocks, which enable arithmetic operations. If the fabric from Figure 22d is taken and embedded into the fabric of Figure 22a, a new device is formed, as shown in Figure 22c. This device is an ASIC, which represents eFPGA-based obfuscation. However, reconfiguring a device leads to PPA overheads compared to ASIC.

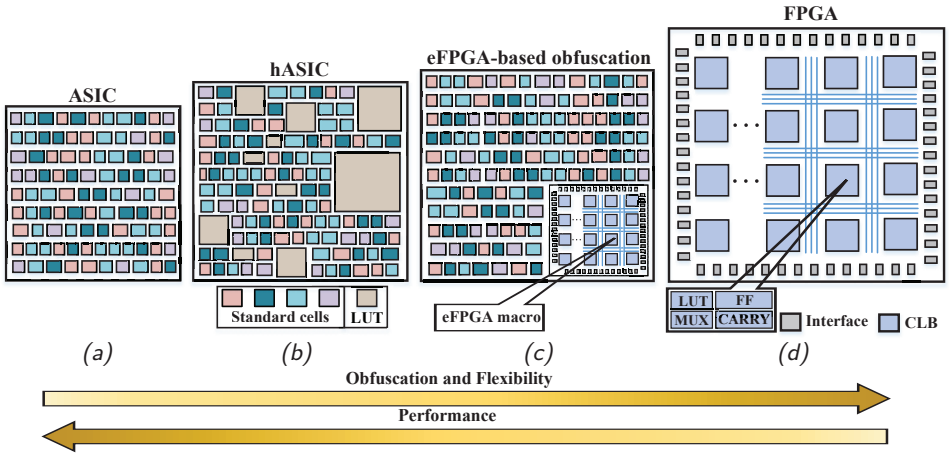


Figure 22: The design obfuscation landscape.

Most techniques aim to minimize the reconfigurable part to avoid significant performance and area overheads. Unfortunately, this compromises the security of the obfuscated design. To adjust the level of obfuscation, the reconfigurable tile in 22c could be increased. However, distributing the logic all over the fabric or fine-grain reconfigurable logic, as shown in Figure 22b, where LUTs and standard cells are entirely mixed, can improve PPA and offer more flexibility. This thesis takes advantage of this possibility along with the reconfigurable elements. Moving from right to left, performance increases, while obfuscation and flexibility increase from left to right. However, neither extreme is an ideal design point for circuits with strict security and performance requirements. A middle solution that can balance performance and security is addressed in this thesis by introducing a "hybrid ASIC" (**hASIC**).

Figure 22b illustrates a hybrid design incorporating fine-grain reconfigurable and static parts to achieve the obfuscation's purpose. The reconfigurable part obscures the circuit, while the static logic provides performance benefits. The generated block architecture consists of a combination of reconfigurable and static cells to explore the design space at the block level. Programmable LUTs are utilized to implement the reconfigurable part, rendering the circuit non-functional until programmed. However, this thesis also emphasizes the importance of a *high degree of obfuscation* in effectively concealing the circuit's intent, generally not investigated in state-of-the-art techniques.

3.2 Security-aware CAD Flow for hASIC

The security-aware CAD flow is depicted in Figure 23, which exploits the custom tool to generate an hASIC design with static and reconfigurable logic. Programmable LUTs, similar to those found in FPGAs, are used to implement reconfigurable logic. The entire obfuscation process is automated. Therefore, the design time experiences a slight increase when compared to the traditional ASIC flow. During the initial phase of the process, a commercial synthesis tool for FPGA creates a Verilog netlist of the targeted design for obfuscation. Then, the synthesis tool generates a timing report and a netlist that includes standard FPGA primitives like LUTs, MUXes, and FFs. To achieve its primary goal of replacing FPGA cells with ASIC cells, TOTe utilizes the ASIC standard cell library of choice, the outputs generated by FPGA synthesis and user-defined obfuscation target obf_c.

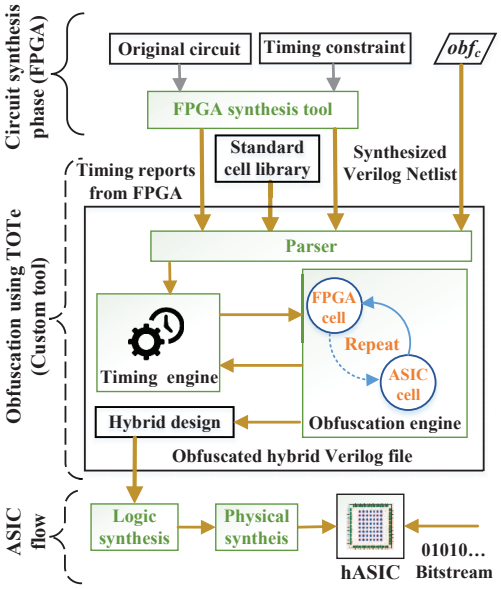


Figure 23: A security-aware CAD flow for hASIC.

In the second phase, The parser reads elements from the netlist and paths from the timing report. These are then sent to a timing engine for processing. Then, it selects LUTs in the critical path and replaces them with standard cells that implement the same logic, removing the programmability aspect. In other words, TOTe recognizes critical paths and replaces 'slow' reconfigurable elements with 'fast' static ones. This process is repeated until no more LUTs can be converted in order to respect the user-defined

obfuscation target obf_c. The obfuscation target controls the ratio of LUTs that must remain programmable. By replacing LUTs with static logic, TOTe reduces the area, power, and delay, thereby improving the frequency of the design. The output of the tool is an obfuscated *hybrid* Verilog file containing both reconfigurable LUTs and static part. Finally, to complete the hASIC design, logic and physical synthesis are performed to generate the layout. The resulting layout is then sent to the foundry for fabrication. The following subsection provides a comprehensive overview of the flow and internal architecture.

3.2.1 Detailed Flow and Internal Architecture of ToTe

The process of obfuscating a design and producing an hASIC through logical and physical synthesis involves a comprehensive 7-step approach, as depicted in Figure 24. Circled numbers represent these steps in the following text.

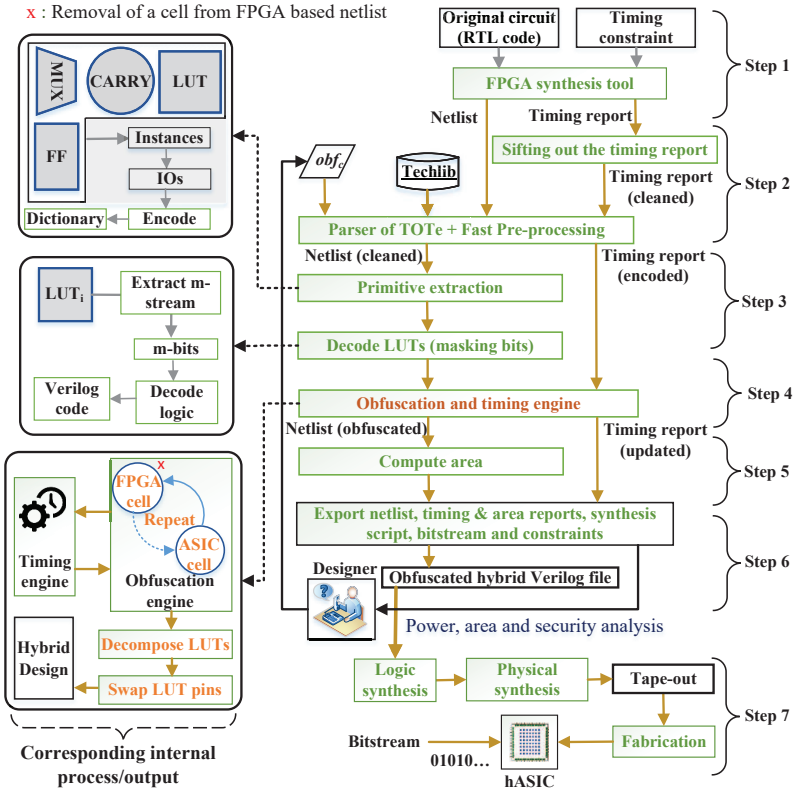


Figure 24: Overview of the obfuscation flow and its inner steps.

During Step (1) of the process, the original circuit is synthesized using a commercial FPGA synthesis tool. Notably, the original circuit requires no special annotations, synthesis pragmas, or any other changes in its representation. The output of this step consists of a synthesized netlist and a timing report, with the netlist comprising all the typical FPGA primitives such as MUXes, LUTs, and FFs. It is important to note that at this point, the logic of the design is 100% obfuscated since the design entirely consists of LUTs.

Moving on to Step (2), the pre-processing stage begins with filtering and interpreting

the timing report and Verilog netlist. Parsing the timing report is relatively straightforward, but the report often contains redundant information, such as empty lines and headers. A bash script has been developed to discard irrelevant data to address this. Once the timing report has been filtered, each analyzed path may contain four FPGA primitives, namely FF, $CARRY$, LUT_{i}, and MUX. TOTe encodes (hashes) instance names to ensure a more efficient representation, reducing the need for lengthy string representations. The pre-processing stage concludes when it generates a list of timing paths, where each path consists of a collection of hashed instances and corresponding delay values. It is important to note that an instance may appear in several paths and under different timing arcs. Finally, the list of timing paths is sorted in ascending order. The path with the highest delay, the Critical Path (CP), is used as a reference. The sum of all CPs is referred to as $sumCP$[1].

In Step (3), TOTe performs primitive extraction and LUT decoding to preserve the circuit structure after optimization. To achieve this, the tool creates a graph representation of the netlist to keep track of port connections. Every instance is annotated with its primitive type, including masking patterns for LUTs. TOTe can interpret the LUT encoding scheme used in the FPGA netlist. For example, when dealing with a LUT_6, the tool extracts a 64-bit masking pattern from the netlist, which is then converted into a truth table with six inputs and one output. Figure 15 shows the truth table for LUT_2. The masking pattern determines which input combinations generate 1s and 0s at the output. This process is repeated for smaller LUTs. Using the populated truth tables, the tool builds combinational logic equivalent to the LUT's logic. Finally, the truth tables are exported as synthesizable Verilog code. For other primitives, such as FF and MUX, no decoding is required, and they are directly translated into their ASIC equivalent logic cells.

The security and performance objectives of the tool are driven by obfuscation and timing engines. These engines are responsible for various important tasks, such as critical path identification, timing analysis, and replacement of reconfigurable cells for static cells, and are utilized in Step (4). Algorithm 4 outlines the various operations performed within the obfuscation algorithm of the tool. In this algorithm, the list of LUTs is denoted as L, the list of timing paths as P, and the obfuscation level as obf_c. Additionally, L_{ST} and L_{RE} are internal variables that represent lists of static and reconfigurable LUTs, respectively. At the start of the obfuscation algorithm, all LUTs are stored in L_{RE} on line 1. It calculates the value of the K variable in terms of the number of LUTs to be realized as a static part on line 2, where the SIZE_OF function returns the number of elements in the list. In the loop on lines 3-9, the critical path is identified on line 4 using the FIND_CRITICAL function, and the slowest LUT on that path is identified using the FIND_SLOWEST function on line 5. If the identified LUT is in L_{RE} on line 6, the lists of LUTs are updated on lines 7-8. Where the INSERT and REMOVE functions insert and remove the LUT, respectively. The timing engine recalculates the affected paths on line 9, where the UPDATE_INSTANCENAME_TIMING function updates the instance name of the corresponding as a static logic and the critical path and delay of the corresponding LUT in the timing report. This loop on lines 3-9 continues until K LUTs are selected for the static part.

After the obfuscation level is met, additional steps on lines 10-17 are required to

[1]It is worth noting that CP and $sumCP$ are analogous to Worst Negative Slack (WNS) and Total negative Slack (TNS) in traditional Static Timing Analysis (STA), except that all paths in this analysis are assumed to pass timing checks, which means that no negative values are considered for the sake of simplicity.

Algorithm 4: Obfuscation procedure

```
   Input: L, P, obf_c
   Output: hASIC
1  L_ST ← φ, L_RE ← L
2  K ← (1 − obf_c) × SIZE_OF(L_RE)
3  while SIZE_OF (L_ST) ≤ K do
4      path ← FIND_CRITICAL(P)
5      lut ← FIND_SLOWEST(path)
6      if lut ∈ L_RE then
7          INSERT(lut, L_ST)
8          REMOVE(lut, L_RE)
9          UPDATE_INSTANCENAME_TIMING(lut, P)
10 for each lut ∈ L_ST do
11     Design_ST ← DECODE(lut)
12 for each lut ∈ L_RE do
13     GEN_CASE_0_1(lut)
14     DECOMPOSE_OPT(lut)
15     SWAP_PINS(lut)
16 Design_RE ← GEN_RE(L_RE)
17 return hASIC ← Design_ST ∪ Design_RE
```

implement hASIC. The DECODE function on line 11 operates on each LUT that was mapped as a static part. The description of these LUTs in Verilog as truth tables is already processed during Step (3). Subsequently, the ASIC synthesis of the truth tables is executed to obtain netlists composed of standard cells. Timing and power analysis during physical synthesis is generated by the function GEN_CASE_0_1 for 'force logic' on line 13. If not generated, each LUT would be timed for its worst timing arc instead of the implemented timing arc when the LUT is programmed. The larger LUTs are decomposed into smaller LUTs by DECOMPOSE_OPT on line 14, which will be described in Section 3.3. On line 15, SWAP_PINS performs a final timing optimization that attempts to swap the LUT pins to improve the delay, which is also discussed later in Section 3.3.6. The function GEN_RE on line 16 generates the reconfigurable part. Ultimately, the algorithm merges the design generated for the static part, $Design_{\mathsf{ST}}$, and the reconfigurable part, $Design_{\mathsf{RE}}$, to build hASIC.

During Step (5) of the obfuscation process, the tool estimates the area of the hASIC design using the formula $A = A_{RE} + A_{ST}$. To determine the area of the reconfigurable part, denoted as A_{RE}, it sums up the area of the reconfigurable LUTs. Similarly, it computes the area of the static part, denoted as A_{ST}, by summing up the area of the standard cells of the static LUTs. It uses an industry-grade physical synthesis tool that properly considers congestion to ensure a highly accurate estimate. In the hASIC design process, Step (6) involves creating files that describe hASIC. This step generates an obfuscated hybrid Verilog file, timing, and area reports. Designers can repeat this process until they achieve their desired level of obfuscation and performance. In Step (7), the obfuscated netlist is synthesized using a commercial synthesis tool. Then, hASIC is implemented using a commercial physical synthesis tool, which executes traditional Place & Route (P&R), CTS (Clock Tree Synthesis), Design Rule Check (DRC), and other necessary steps. The resulting tapeout database is then sent to the foundry for fabrication. Once the fabricated parts are received, programming is required

for the hASIC design to function correctly, which involves using a bitstream, just like in an FPGA design.

3.3 LUT-specific Approaches

This section discusses optimizations related to LUTs and the measures taken to ensure that an hASIC design exhibits an ASIC's high-performance attributes while maintaining an FPGA fabric's obfuscation capabilities.

3.3.1 Custom Standard Cell Based LUTs

Custom LUTs (LUT_1, LUT_2,..., LUT_6) have been designed from *regular standard cells*, following Versatile Place and Route's (VPR) template [188]. Table 3 shows the area, density, number of FFs, combinational cells, and average delays of the implemented LUTs. The area for these macros approximately doubles from LUT_i to LUT_{i+1}. The number of flip-flops grows with the LUT_i size (2^i). The average delay highlights the average of all timing arcs. It should be emphasized once more that the LUTs were generated as macros composed of standard cells, thereby rendering them compatible with standard cell based design flows. The LUTs are highly compact, with the main design goal being area/density. The layouts for LUT_4, LUT_5, and LUT_6 macros are shown in Figure 25.

Table 3: Block implementation results for LUTs.

Macro	Area (μm^2)	Density (%)	# FFs	Comb. cells	Avg. delay (ns)
LUT_1	36.00	76.00	2	1	0.049
LUT_2	64.80	76.26	4	1	0.052
LUT_3	117.00	89.23	8	8	0.119
LUT_4	259.20	85.23	16	15	0.192
LUT_5	491.40	91.50	32	33	0.257
LUT_6	957.60	91.09	64	36	0.295

Commercial FPGAs typically have limited flexibility in terms of implementing a LUT size. However, hASIC can implement designs with different LUT sizes due to its design-specific nature. This means that the reconfigurability aspect of FPGAs is no longer necessary. Additionally, LUT macros are highly compact, allowing for high-density designs. Each LUT includes FFs for storing configuration bits that serve as a lock for the obfuscated design and three extra pins for configuring the registers ($serial\_in$, $serial\_out$, and $enable$). The LUTs are connected in a serial chain, similar to a scan chain. Using FFs, the technology-agnostic framework makes floorplanning and placement effortless. Furthermore, the LUTs are treated as regular standard cells during physical synthesis, allowing TOTe to take full advantage of commercial EDA tool placement algorithms and eliminating the need for custom scripts for placing the LUT macros.

3.3.2 LUT Decomposition

The area and delay of a LUT are directly correlated with its number of inputs. The size is primarily determined by the number of sequential elements needed to store the LUT's truth table, while the speed is proportional to the LUT's internal MUX tree. However, not all 6-input functions need a LUT_6 for implementation. For example, an AND6 can be broken into 5 AND2s, as shown in Figure 26. According to Table 3, it is evident that the area almost doubles for each additional input. The delay increases significantly, with a LUT_6 having almost six times the average delay of a LUT_2. The example demonstrates

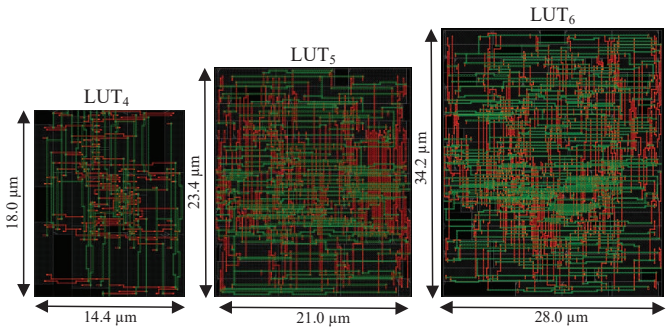


Figure 25: The layout of macros for LUT_4, LUT_5, and LUT_6 [73].

that decomposing a LUT_6 can reduce the area to less than one-third of its original size. Furthermore, the delay is reduced to approximately half. This example offers a promising approach to *enhancing timing and area* of the circuit. It will also reduce the power observed during the physical synthesis. To decompose LUTs, TOTe utilizes Functional Composition (FC) [189]. This approach enables bottom-up association of Boolean functions and offers control over the costs involved in the composition process. This capability sets it apart from traditional top-down functional decomposition, which does not provide a final cost until the complete decomposition process.

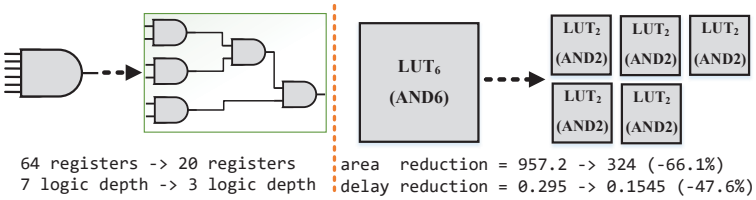


Figure 26: Logic conversion and decomposition of LUT_6 [73].

3.3.3 Functional Composition for LUTs

This section will provide an overview of the FC paradigm and how it can be applied to LUTs. Readers can refer to the sources listed in [189, 190] for more in-depth information. The FC paradigm is a bottom-up approach guided by five core principles. First, it uses bonded pairs (BPs) that consist of a functional part (a canonical implementation of a Boolean function, such as a binary decision diagram or truth table) and an implementation part (the structure being optimized, such as a fanout-free LUT circuit). Second, each BP association performs independent functional/implementation operations, which allows for more complex implementations with simpler functional operations. Third, using partial ordering and dynamic programming, all BPs with the exact cost are stored together in a set (bucket), enabling intermediate solutions as sub-problems and associations' performance in a cost-increasing fashion. Fourth, initial BPs, such as constants and single input variables, are required to initiate any FC algorithm. Finally, the FC paradigm allows the heuristic selection of a subset of permitted functions to reduce the composition search space.

3.3.4 Exhaustive LUT FC method

FC can be exhaustively utilized to create fanout-free implementations that have optimal cost. Algorithm 5 can generate all minimal LUT fanout-free implementations for functions up to 4 variables. To generate functions with I inputs, the algorithm creates all implementations using LUT_{I-1} at a lower cost than $\mathsf{LUT}_{\mathsf{I}}$. Functions with up to 2 inputs cannot be decomposed, and for functions containing I inputs, a set of Boolean operators is required. The Boolean operators refer to NPN class functions from 2 up to $I-1$ inputs and their negated and permuted variants. The cost functions in the optimization process utilize area values from the LUT macros' bounding box, while the delay values are determined by averaging the delay of all timing arcs. More details on the functional decomposition can be found in [189]. However, performing a more complex timing analysis can result in different delays for various permutations, leading to diverse outcomes. To address this issue, the SWAP_PINS capability can be utilized at the end of the flow, as detailed in Section 3.3.6.

The FC-OPT-LUT algorithm is designed to take two variables: the number of LUT inputs N and the cost function C, which considers the area and delay. The final output of this algorithm is ALL_IMP, which is a map of functions and LUT implementations. To begin with, line 1 initializes three variables, namely A_IMP, the output, B, the bucket list containing all the already implemented functions, and i, as a counter. On line 2, MAX_COST provides the single $\mathsf{LUT}_{\mathsf{N}}$ cost. It is worth noting that the bucket list B is initialized with constants and single variables through the method CREATE_INITIAL_FUNCTIONS on line 3. On line 4, the algorithm computes the association of tuples and arrival time (AT). This association comprises tuples containing the indices of the buckets used to combine the functions, from index 0 to $i-1$, where the cost needs to be higher than $B[i-1]$. However, simultaneously, it should be the smaller one of all possibilities. For instance, if the candidate ATs have a cost of 14, 10, 10, 12, and $B[i-1]$ has a cost of 9, the candidate ATs with a cost of 10 will be selected.

Algorithm 5: FC-OPT-LUT Algorithm [189]

Input: N $(number\ of\ LUT\ inputs)$, C $(cost\ function)$
Output: ALL_IMP

```
1  ALL_IMP ← φ, B ← φ, i ← 1
2  MAX_COST ←LUT_COST(C, N)
3  B ←CREATE_INITIAL_FUNCTIONS (I)
4  AT ←NEXT_BUCKET(B, i, C)
5  while COST(AT, C) < MAX_COST do
6  |   B ← (ASSOCIATE (AT, C, ALL_IMP))
7  |   i ← i+1
8  |   AT ←NEXT_BUCKET(B, i, C)
9  if SIZE_OF(IMP_LUT) < 2^(2^I) then
10 |   CREATE_NAIVE_IMPS(ALL_IMP, I)
11 return ALL_IMP
```

The code in lines 5-8 features a while loop that checks if the cost of AT is not greater than MAX_COST. Using a simple solution is advisable when the cost is higher. If the cost is lower, the $ASSOCIATE$ method processes the AT list, pairing or grouping them in sets of two or three (depending on tuple size), and breaks ties using the cost function. The resulting output is added to the bucket list. A new list of AT is

generated to either continue or break the loop. Finally, in lines 9-10, any remaining functions are considered not decomposable (i.e., the cost to decompose them is higher than the naive solution). The CREATE_NAIVE_IMPS method searches for Boolean functions without implementation on ALL_IMP and adds a naive one, ensuring that all Boolean functions with up to I inputs are present on the ALL_IMP map, which is returned on line 11.

3.3.5 Heuristic LUT FC method

It should be noted that the technique outlined in the previous section is limited to generating optimal LUTs with a maximum of 4 inputs. For more complex LUTs, a heuristic is required. LUT decomposition can be considered a factorization problem to simplify the process, where a factored form is converted directly to a LUT tree structure with no fanout. This technique involves modifying the Boolean factoring method presented in [190]. The modifications are crucial in deriving LUT decompositions that can provide a better cost than the naive solution.

Algorithm 6 presents FC-HEUR-LUT, which takes the target function F and the cost function C as inputs and produces the LUT implementation IMP as output. The LUT circuit includes the decomposed naive solution. The algorithm initializes the variables ALL_IMP and B on line 1, where ALL_IMP stores all known implementations for the functions already decomposed, and B contains the buckets. The method CREATE_INITIAL_FUNCTIONS remains the same as in FC_OPT_LUT. The algorithm checks if it is a trivial case on lines 4-5 and returns if so. Line 6 executes the method EXTRACT_ALL_COFACTORS, which computes all the cofactors and cubecofactors (excluding constants) from F and stores them in the ALL_COF set.

A recursive call to the algorithm is made on lines 7-9, providing a LUT implementation to all cofactors and cubecofactors. Then, the combination of cofactors takes place on line 11, using the same strategies presented to expand the "allowed functions" set, as explained in [190]. This expansion guarantees at least two factored subfunctions that will provide at least one functionally equivalent solution when associated with the next step. In line 12, the ASSOCIATE_FUNCTION will perform AND/OR/XOR operations using the rules mentioned and NAND/NOR/XNOR associations using the "not comparable" functions. These associations are discarded if they are not the target function F. Suppose the association is functionally equivalent to F. In that case, the cost function C will compare the current solution (which initially is the naive one) with the current one, replacing it in the case of a better cost. Finally, lines 13-14 will collect the resulting implementation IMP and return.

To speed-up FC-HEUR-LUT, two techniques are applied. Firstly, FC-OPT-LUT results are used to assist in FC-HEUR-LUT. At the start of the algorithm, the ALL_IMP map is utilized to return the optimal implementation quickly if the function F supports four or has fewer inputs. This improves the decomposition QoR and speeds up the process. Secondly, there is a limit on the number of associations of "not comparable" Boolean functions. This limit is necessary as the number of such functions can be substantial, sometimes exceeding 100K. Restricting them avoids significant runtime trying to decompose more complex functions, which generally have worse costs when decomposed.

3.3.6 Pin Swap Approach

The LUTs mentioned in Section 3.3.1 are essentially a MUX tree powered by registers storing a truth table that can be customized. The MUX tree is the primary factor that affects the LUT delay, especially for LUTs with multiple inputs. Therefore, the order in

Algorithm 6: FC-HEUR-LUT Algorithm [190]

Input: F $(target\ function)$, C $(cost\ function)$
Output: IMP

```
ALL_IMP ← φ, B ← φ
B ← CREATE_INITIAL_FUNCTIONS (F, ALL_IMP)
IMP ← ALL_IMP(F)
if IMP ≠ φ then
    return IMP
ALL_COF ← EXTRACT_ALL_COFACTORS (F)
foreach cofactor COF ∈ ALL_COF do
    6 COF_IMP ← FC_HEUR_LUT(COF, C)
    ALL_IMP ← [COF, COF_IMP]
ALL_IMP ← [F, GET_NAIVE_SOLUTION(F)]
COMBINE_COFACTORS(ALL_COF, ALL_IMP)
ASSOCIATE_FUNCTIONS (ALL_COF, ALL_IMP, C)
IMP ← ALL_IMP(F)
return IMP
```

which the pins are arranged significantly impacts the LUT delay. Inputs connected to a MUX closer to the output will have lower logic depth and faster performance.

The SWAP_PINS method used in Algorithm 4 utilizes the flexibility of LUT functions to allow for arbitrary input pin swaps by permuting the function's truth table. This approach uses a LUT function and timing information and outputs the permuted truth table and a new order of input pins/nets. The example in Figure 27 demonstrates a successful pin swap that improves design slack. The pin swap algorithm considers the LUT function, Arrival Time (AT) for each input net (referred to as [$n0$, $n1$, $n2$]), the cell arc delay (DLY) associated with each input, and the required time (RT) at the output. In the presented example, the critical arc is $n2$, with a total delay of 1.23, and RT=1.1. The algorithm explores all input permutations to minimize WNS, and if negative slack is detected in two or more arcs, it also attempts to reduce TNS. If a new order improves WNS and/or TNS, the truth table is permuted to maintain the same functionality. The algorithm outputs the truth table $0x10$ and the new net order [$n2$, $n0$, $n1$].

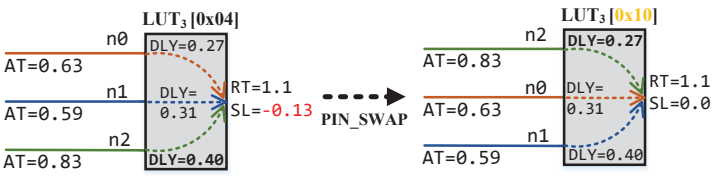


Figure 27: Example of a beneficial pin swap [73].

3.4 Experimental Results

This section presents a comparison between performance and security trade-offs for various designs at different degrees of obfuscation. The objective is to present a range of representative designs, including established benchmarks and circuits for different

applications such as crypto cores, filters, CPU, GPU, etc. The designs were selected for their diverse architecture, applications, and number of LUTs in the critical path. All experimental results were obtained by executing FPGA synthesis in Vivado, with a target device of Kintex-7 XC7K325T-2FFG900C containing 6-input LUTs. Subsequently, Cadence Genus was employed for logic synthesis, using three flavors of a commercial 65nm standard cell library (LVT/SVT/HVT). It is important to note that TOTe is completely *agnostic with respect to PDKs, libraries, and tools*.

For the initial experiment, it was desired to cover all possible FPGA primitives with a compact design. A schoolbook multiplier (SBM) design was utilized [191]. To analyze the effects of obfuscation on performance and area trends, 8-bit SBM has been obfuscated by varying obf_c from 55% to 100%. The synthesis targeted a challenging frequency of 540MHz, and the timing engine calculations showed that CP and $sumCP$ values became 0.490ns and 16088.69ns, respectively. The values are obtained for a design at 100% obfuscation level, representing a design analogous to FPGA. It is important to note that they represent a simple timing analysis. During the obfuscation process, CP and $sumCP$ remained consistent. This consistency is sufficient to determine critical paths generally, and realistic timing values can only be obtained during the final timing analysis using a commercial physical synthesis tool.

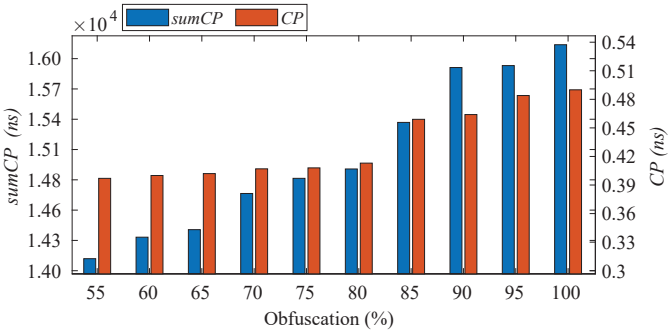


Figure 28: Obfuscation versus performance trade-off for SBM [74].

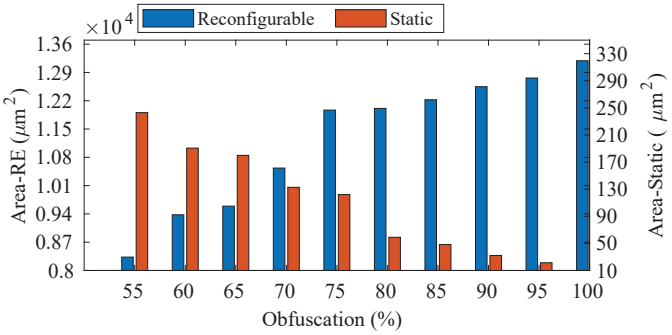


Figure 29: Obfuscation versus area trade-off for SBM [74].

The performance of the 8-bit SBM was analyzed after performing obfuscation at different levels. The timing characteristics are illustrated in Figure 28. The results showed that increasing the level of obfuscation led to a decrease in performance and vice versa. The trend depicted in Figure 28 was that CP improved inversely with the obfuscation, but it saturated when the obfuscation was below 80%. However, this was

not the case for $sumCP$, as the decrease in obfuscation caused continuous improvement. The obfuscation versus area profile of the 8-bit SBM is also illustrated in Figure 29.

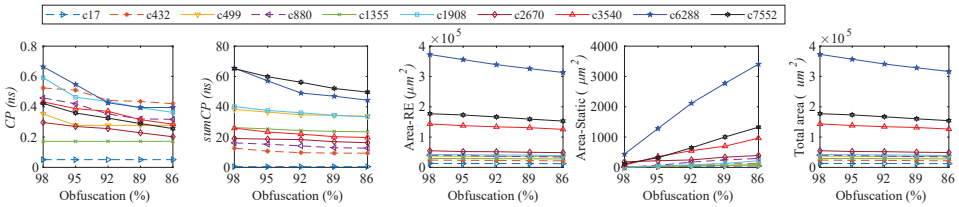


Figure 30: Obfuscation results for ISCAS'85 benchmarks [73].

An investigation was carried out to determine if similar saturation would be exhibited by other designs. To accomplish this, the ISCAS'85 benchmarks were opted for and the results are presented in Figure 30. These combinational benchmarks were chosen as they have only one stage of logic, making it easier to trace the correlation between CP and $sumCP$ (i.e., the critical path remains unchanged irrespective of different reg2reg paths). Surprisingly, even in these simple designs, saturation occurs remarkably fast. Obfuscation has also been applied to more representative designs to cover more comprehensive results. The results for IIR, PID, Median Filter, SHA-256, and other cryptocores and large designs are presented in detail in Table 4. The designs listed in Table 4 are sorted based on the number of LUTs used. Graphical representations of the results for AES, RISC-V, and SHAKE-256 designs have also been included in Figure 31, which provide visualization of the trends. Regarding optimization, the results for the decomposed LUTs will be presented later, where physical synthesis will be executed for a fair comparison between the baseline and optimized designs.

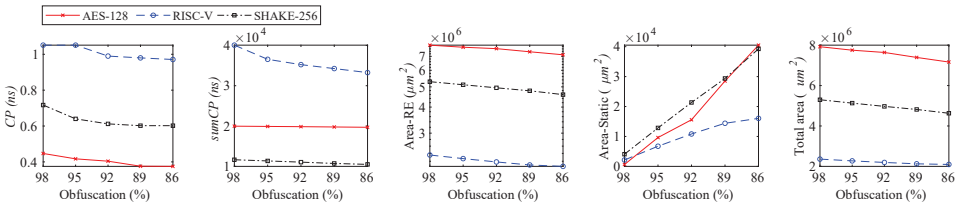


Figure 31: Obfuscation results for AES-128, RISC-V and SHAKE-256 [73].

To summarize, the findings presented in this chapter validate the trade-offs between design and security for numerous designs. It is evident that using a LUT-based circuit representation, similar to an FPGA, affects delay and area differently. Regarding area, the trend is straightforward - the smaller the obfuscation target, the more compact the circuit. However, when it comes to delay, it seems that hASIC incurs performance penalties that reducing the targeted obfuscation level alone cannot overcome. Therefore, the functional decomposition of LUTs is employed to achieve better performance. The next chapter will present a detailed physical synthesis analysis, including applying optimization methods described in Section 3.3 to enhance performance.

Table 4: Detailed results for selected designs.

Design	**Obf. (%)**	***sumCP*** **(ns)**	***CP*** **(ns)**	**Area-RE (μm^2)**	**Area-ST (μm^2)**	**LUT (RE)**	**LUT (ST)**
IIR [192]	98	1574.48	0.591	55031.04	257.4	584	11
	95	1553.32	0.526	54104.40	720.72	566	29
	92	1534.39	0.526	53177.76	1184.04	548	47
	89	1501.29	0.526	52251.36	1647.36	530	65
	86	1489.93	0.526	51324.48	2110.68	512	83
PID [193]	98	2547.58	0.756	445590.00	2816.82	896	18
	95	2466.25	0.642	432340.92	9441.36	869	45
	92	2391.96	0.592	421365.95	14928.84	841	73
	89	2348.61	0.568	407273.76	21974.94	814	100
	86	2322.46	0.543	392345.64	29439.00	787	127
Median Filter [194]	98	637.584	0.963	499601.16	2860.2	979	19
	95	563.584	0.747	483561.00	10880.28	949	49
	92	504.805	0.597	469323.72	17998.92	919	79
	89	480.018	0.543	448850.52	28235.52	889	109
	86	466.454	0.543	427346.27	38987.64	859	139
SHA-256 [195]	98	7425.73	0.962	1313150.76	10291.86	2195	44
	95	7354.59	0.871	1275984.00	28875.24	2128	111
	92	7322.15	0.871	1233448.56	50142.96	2060	179
	89	7301.94	0.871	1179674.64	77029.92	1992	246
	86	7164.02	0.871	1125799.56	103967.46	1925	313
FPU [196]	98	2909.06	0.707	1031676.84	1250.028	2487	50
	95	2734.00	0.650	1003225.68	2672.586	2412	126
	92	2572.95	0.650	966715.20	4498.11	2336	202
	89	2478.73	0.650	935060.04	6080.868	2259	279
	86	2410.21	0.650	893005.56	8183.592	2183	355
GPU (OR1200-HP) [197]	98	237699.30	0.933	21317740.2	124971.48	40739	831
	95	215696.68	0.871	21009821.4	278931.10	40102	2078
	92	185520.65	0.750	20015822.2	495521.11	39492	3352
	89	154560.56	0.650	19552256.6	781521.30	38243	4521
	86	135802.32	0.625	18552023.3	1011230.2	36125	5806

4 Physical Implementation

This chapter discusses the validation of the design obfuscation technique in physical implementation. The focus is on the physical synthesis of well-known designs, including cryptographic cores at varying levels of obfuscation. The aim is to analyze the impact of various obfuscation levels on physical synthesis results, including area, power, timing, and security trade-offs.

In cases, where the design has been obfuscated, TOTe generates a structural Verilog description of hASIC. The process of implementing an hASIC can be divided into two phases: logic synthesis and physical synthesis. Figure 32 provides a detailed diagram flow of this process. The logic synthesis converts the Verilog code into gate-level netlist while meeting the performance requirements specified in the design constraints. Generating the gate-level netlist requires a standard-cell IP library and design constraints. Therefore, the designer must have decided on the technology to fabricate the IC during this phase. The inputs required for the logic synthesis are the Verilog code of hASIC, standard-cell timing library, and design constraints.

Generally, foundries characterize each gate regarding process variation, voltage, and temperature in timing libraries. These characteristics are typically compiled in a standard Liberty format. In addition to the timing library, the designer must set the design constraints using 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. Although the gate-level netlist is useful for estimating PPA values, physical synthesis provides more accurate results due to its consideration of routing, placement, and clock propagation.

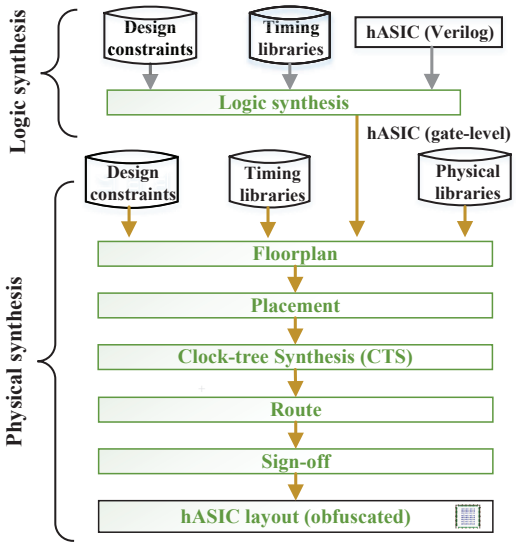


Figure 32: The steps involved in the physical synthesis of hASIC.

4.1 Physical Synthesis for hASIC

This section contains the physical implementation results for hASIC. As shown in Figure 32, the output of physical synthesis is a layout of all layers used by the foundries as a blueprint for the fabrication of an IC, which is usually handled in GDSII format. At this

level, physical information about standard cells and the metal stack is required. The EDA tool is able to manage metal layers, including the number of metals, the allowable width of each metal, and the type of vias. The Library Exchange Format (LEF) file is the preferred format for describing the physical characteristics of each gate and the metal stack. Inputs for physical synthesis include the gate-level netlist, timing libraries, design constraints, and LEF files for the gates and technology.

The physical synthesis consists of several steps. Floorplanning involves sizing the block box for a target density, defining the pinout, and implementing power distribution. At this stage, the density setting is precise since all the required gates except for the buffers in the netlist are present. The gates will still be modified while the optimization phase is ongoing. After floorplanning comes placement, which not only places gates coherently with their interconnections, but also considers timing and congestion. The main goal of CTS is to balance clock delay for all sequential elements in the design.

After completing the CTS, the clock delay is balanced to all clock inputs by inserting buffers/inverters along the clock routes, which makes timing analysis more realistic. With all gates placed and the clock tree routed, the next step is to route all interconnected gates. Routing involves drawing wires between all drivers and sinks. Depending on the amount of routing resources, design rules, and congestion, routing can be very challenging, taking several hours or even days to complete. After routing, sign-off completes the design process for hASIC to carry out physical verification. Finally, the layout of hASIC is exported in GDSII format.

The physical synthesis is a complex and time-consuming process. The final layouts for various obfuscation levels are presented using two designs. These designs are well-known cryptocores, AES-128 [198] and SHA-256 [195]. These designs are medium designs and represent practical examples. During physical implementation, Cadence Innovus is used with a commercial 65nm PDK for physical synthesis.

4.2 Physical Implementation of AES-128

The Verilog code for AES-128 was obtained from [198]. Three obfuscation levels, 60%, 70%, and 80%, were selected to analyze the design versus security trade-offs. LUTs defined in Section 3.3.1 are exploited for the design with aforementioned obfuscation levels. Table 5 presents the results after the physical synthesis of AES-128. The analysis started with the initial FPGA implementation, which achieved a frequency of only 103 MHz, with the target device being a Kintex-7. Table 5 provides the results for obfuscation levels of 80%, 70%, and 60%. The results indicate that the level of obfuscation does not affect the utilization density of the design, which is determined by the ratio of placement sites that are occupied vs. total area. The designs achieved a high utilization density of around 80% for all designs, even with the inclusion of many LUTs. This indicates that the macros do not compromise global routing resources.

The implementation for the 60% obfuscation level shows the lowest area and a performance of 260 MHz for TOTe. The results also demonstrate that decreasing the obfuscation level improves the frequency, and vice versa. Timing results were obtained after physical synthesis and are for the worst process corner (SS), $VDD = 0.9 * VDD_{nominal}$, and a temperature of $125°$C. It is worth noting that the area of TOTe-generated designs increases as the obfuscation level increases, and the number of LUTs also increases with the obfuscation level. This behavior aligns with the original goal of TOTe, which was to establish a trade-off between performance (ASIC) and security (FPGA). As previously mentioned, LUTs are used for security purposes but exhibit an area penalty, as confirmed by physical synthesis. Furthermore, leakage and

Table 5: Implementation results of AES-128 under different obfuscation levels.

Design	Obf.	Dens.	Area (μm^2)	Freq. (MHz)	Leakage Power (mW)	Dynamic Power (mW)	# LUT	# Buffer	# Comb.	# Inv.	# Seq.	Total Wire-length (mm)
FPGA	100%	–	–	103	6.2	587	10688	–	–	–	6000	–
TOTe	80%	78%	14062200	240	97.51	2246.49	9332	31376	7527	3634	15332	17432.17
TOTe	70%	80%	12118975	249	86.52	1989.45	8165	27972	14165	4440	14165	15758.92
TOTe	60%	81%	10386950	260	75.43	1744.57	6999	23398	28395	5491	12999	13846.95
ASIC	NONE	73%	410688	833	4.32	124.08	–	1810	99394	9769	6000	2664.06

Obf. is obfuscation, Dens. is density, comb. is combinational, inv. is inverters, and seq. is sequential.

dynamic power values are proportional to security since reconfigurable logic is less efficient in terms of frequency than static, primarily due to using FFs to store the LUT truth tables.

Lastly, the last five columns of Table 5 display the resource requirements for hASIC, including the number of buffers, combinational cells, inverters, sequential cells, and the total wirelength. The total wirelength of a design is the combined length of all wires present. In Innovus, the primary aim of the placer is to minimize this total wire length, which helps reduce the chip's size and cost. Additionally, minimizing the length of wires also reduces power consumption and delay, which are directly proportional to the wire length. By examining the last column of Table 5, it is evident that the total wire length increases as the obfuscation level increases. The ASIC results show higher performance and lower PPA values. The targeted frequency is the maximum frequency. Therefore, the results for TOTe lie between FPGA and ASIC.

In Figure 33, different views of layouts of AES-128 under various obfuscation levels are presented. The metal stack considered here has seven metals assigned to signal routing. Figure 33a-c depict the layouts for 60%, 70%, and 80% obfuscation levels. The dimensions of layouts are included on the bottom and left sides of each panel. All six variants of LUTs are highlighted with different colors. The static part of hASIC is highlighted in red, and as expected, the design remains predominantly a sea of LUTs. The design comprises LUT_4 and LUT_6, but LUT_6 constitutes the majority. Therefore, the layouts seem to be dominated by orange boxes.

Figure 33d-f shows the layouts for a 70% obfuscation level. Figure 33d illustrates the layout after routing. The design features mostly vertical orange lines corresponding to M6. Figure 33e provides a closer look at the placement pattern in an hASIC design, which consists of a combination of LUT macros and standard cells. The macros are positioned in alignment with the standard cell rows, leading to a uniform power rail and power stripe configuration throughout the design. The space between the macros is filled with standard cells. Figure 33f highlights the same design but with some routing layers filtered out (only M2, M3, and M4 are shown). As seen in Figure 25, the implemented LUTs utilize the mentioned metal layers, resulting in a visually regular hASIC structure in panel Figure 33f.

4.3 Physical Implementation of SHA-256

In the following subsection, the physical synthesis of SHA-256 has been performed with the same obfuscation level as mentioned earlier. Additionally, the physical synthesis of optimized designs is also presented for comparison. The Verilog code of the SHA256 core was obtained from the repository mentioned in [195]. Similar to the analysisresults

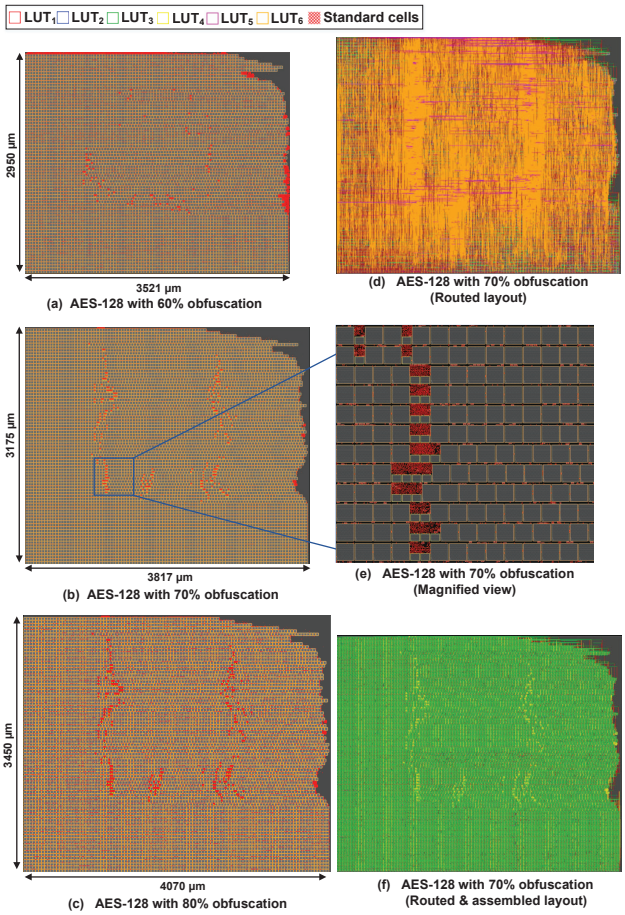


Figure 33: Implementation results for AES-128 with different obfuscation levels.

of AES-128 on FPGA, the device achieved a frequency of 77 MHz for SHA-256. Table 6 shows the utilization of LUTs and FFs for SHA-256. Another point for analysis is being considered here, starting with the 100% obfuscation level as a baseline. This level is fully reconfigurable and comparable to an FPGA design. When compared to FPGA, hASIC exhibits higher performance. However, it also shows increased leakage, dynamic power, and the number of FFs. This FF count includes the ones required for configuring the bitstream.

Let us consider the analysis with a 5% increase in obfuscation level. Table 6 shows the implementation results for obfuscation levels of 90%, 85%, 80%. The implementation of SHA-256 showed a similar trend to that observed in the initial analysis of TOTe. Similar to AES-128, the utilization density of the design remained around 80% for all designs despite a large number of macros. Similar to AES-128, the trend is evident from Table 6. Increasing the security of the design incurs PPA penalties, and vice versa. The baseline hASIC design runs at 223 MHz, as shown in the Freq. column of Table 6. The leakage and dynamic power figures are proportional to security, as reconfigurable logic is less efficient than the static part in hASIC. Similarly, the number of LUTs also decreases as the obfuscation level decreases and vice versa. The last five columns of Table 6 show the resource requirements for hASIC (number of buffers, combinational

cells, inverters, sequential cells, and the total wirelength). All these observations were also analyzed for AES-128.

Table 6: Implementation results for baseline and optimized variants of SHA-256 under different obfuscation levels

Design	Opt.	Obf.	Dens.	Area (μm^2)	Freq. (MHz)	Leakage Power (mW)	Dynamic Power (mW)	# LUT	# Buffer	# Comb.	# Inv.	# Seq.	Total Wire-length (mm)
FPGA	–	100%	–	–	77	2.4	191	2238	–	–	–	1830*	–
TOTe	No	100%	81%	1751500	223	14.85	505.05	2238	5846	93470	6175	105128	9247.65
TOTe	No	90%	77%	1638500	234	12.23	438.47	2015	4626	84107	5017	94876	7505.59
TOTe	No	85%	80%	1507000	241	12.10	430.98	1904	4846	80304	5585	90420	7207.02
TOTe	No	80%	80%	1409700	248	11.05	386.89	1792	4406	75083	4564	83790	6724.43
TOTe	Yes	100%	61%	1155000	307	8.00	301.49	10182	3583	29352	15261	53868	3391.74
TOTe	Yes	90%	65%	979200	312	7.55	273.54	9127	1797	27115	13538	49016	3242.97
TOTe	Yes	85%	67%	940800	322	7.03	256.36	8676	1882	26011	13136	46796	2982.62
TOTe	Yes	80%	64%	883600	357	6.44	278.37	8124	1726	24614	12340	43830	2889.25
TOTe (Swap)	Yes	80%	64%	883600	368	5.93	283.35	8124	1726	24614	12340	43830	2889.76
ASIC	–	NONE	91%	40804	550	0.299	23.86	–	675	7981	1456	1806	181.44

Obf. is obfuscation, Dens. is density, comb. is combinational, inv. is inverters, and seq. is sequential.

Figure 34a-c illustrate the layouts for 80%, 85% and 90% obfuscation levels. The dimensions of the layouts are indicated on the bottom and left sides of each panel. As expected, the design remains primarily a sea of LUTs. From visual inspection, the difference between AES-128 and SHA-256 is clear. AES-128 used LUT_4 and LUT_6 only, but the SHA-256 uses all different LUT variants. This is because the commercial synthesis tool prioritizes using large LUTs, such as LUT_6, to maximize their utilization. In contrast, the FPGA synthesis tool is targeted with LUT_4, which differs from the standard approach taken by most commercial tools during synthesis. During placement, this technique aids in creating a more streamlined and uniform structure for the hASIC, resulting in a more compact design. All six variants of LUTs are highlighted with different colors and the static part of hASIC is highlighted in red. Figure 34d demonstrates the final post route layout of hASIC under 85% obfuscation level. Figure 34e shows the magnified view of the placement in an hASIC desig under 85% obfuscation level. Figure 34f illustrates the assembled view of design under 85% obfuscation level with certain routing layers filtered out. Only M2, M3, and M4 are shown.

The same levels of obfuscation were taken into consideration when working on the optimized designs. The analysis in Table 6 shows a similar trend for the optimized designs discussed in the previous paragraph. With optimization, the baseline frequency increased significantly from 223 to 307MHz. However, placing and routing become more challenging due to a large number of small LUTs, mainly LUT_2s. As a result, the maximum utilization density is approximately 65% across optimized designs. The optimized design resulted in an area reduction of 36% compared to baseline designs. Regarding frequency improvement, designs with obfuscation levels of 100%, 90%, and 85% resulted in a 35% improvement on average. However, the design at 80% obfuscation showed a frequency improvement of 43%.

To enhance the performance, the next step is to apply the pin-swapping technique. In this technique, the same logic function can be generated using different input orders and masking bits (truth table). This technique can swap their pins and effectively reduce the overall delay by identifying LUTs that appear on the critical paths. To demonstrate

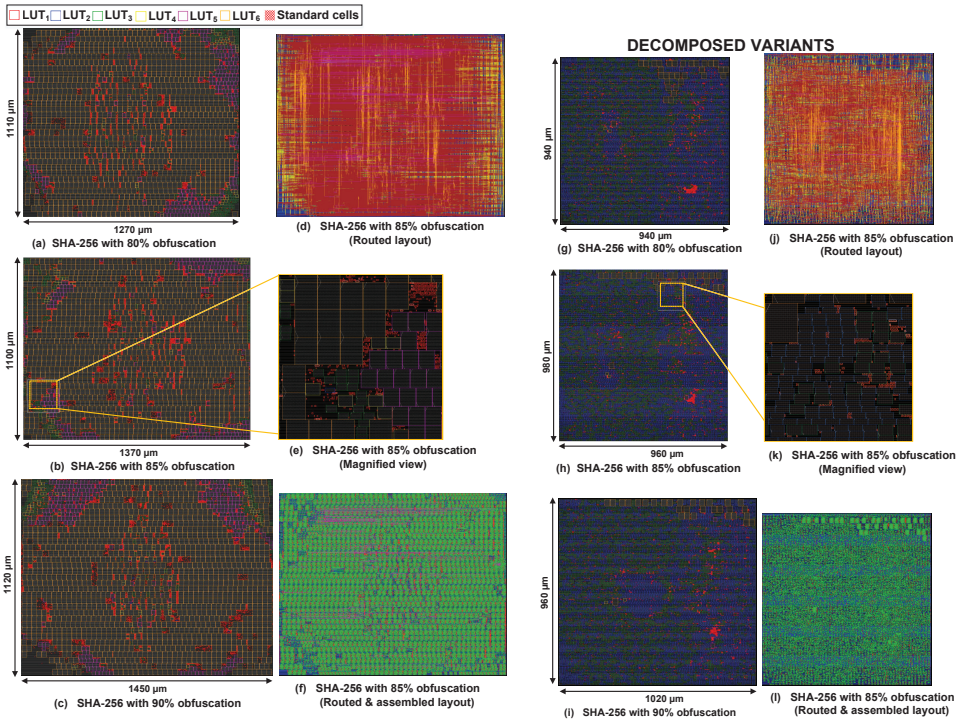


Figure 34: Implementation results for SHA-256 with different obfuscation levels [73].

the effectiveness of pin swapping, a hypothetical situation is considered, where the target frequency of the design is increased, resulting in several paths violating setup timing. The number of violating paths determined the number of LUTs considered for pin swapping, establishing a trade-off between runtime and quality of results. More aggressive frequency targets meant more LUTs were considered for pin swapping. All LUTs from the violating paths were selected as candidates and saved in a list. Starting with the worst violating path, the pins of the LUTs were iteratively swapped until the critical path was improved, as measured by the WNS. The number of swaps versus TNS and WNS is shown in Figure 35. The initial swaps improved the WNS without any effect on TNS. However, continuing to swap improved the TNS without any change in WNS. Improving TNS indicates a potential for a better WNS, so swapping continued until the next jump in WNS. After 200 swaps, WNS improved by approximately 70ps, and TNS improved by 2ns, increasing the frequency of design by 11MHz. With an obfuscation level of 80%, the same design exhibited a 48% performance improvement compared to the baseline design. Nonetheless, decomposition is highly beneficial, offering significant PPA gains compared to non-optimized versions.

The runtime of the physical synthesis flow is not significantly impacted by decomposition, making it worthwhile for design optimization. For instance, the runtime to apply the decompositions in the SHA-256 circuit with 100% obfuscation level containing 2238 LUTs was 11 minutes on an Intel Core i7-6700K. When obfuscation levels were applied at 100%, 90%, and 85%, the designs resulted in an average of 40% improvement in dynamic power. But, when the 80% obfuscation level is considered, only a 27% improvement is shown.

Figure 34g-l presents the layouts of optimized designs while maintaining the same

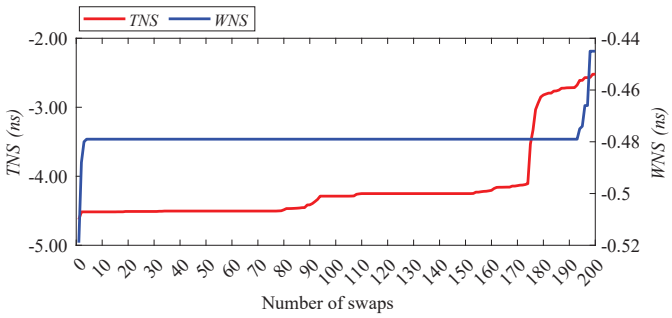


Figure 35: Change in the TNS/WNS concerning the swap of LUT pins [73].

design and conditions. The scaled layouts highlight the area reduction achieved by decomposition. Figure 34l still demonstrates regularity after decomposition, which is expected. Regarding the ASIC implementation, the best possible frequency is 550MHz. Area, leakage, dynamic power and other values are listed in Table 6. The number of FFs in the FPGA implementation differs from the ASIC implementation. Sometimes, it seems that Vivado may replicate registers more aggressively. From Table 6, it is evident that Vivado uses FF cloning to address high fanout buffering for SHA-256. Thus, there is an increase in the number of registers w.r.t. ASIC.

It is worth noting that hASIC has a highly regular structure upon visual inspection. This characteristic can be adjusted to enhance its effectiveness against reverse engineering adversaries. One way to achieve this is by mapping LUTs of various sizes to LUT_6, creating a more uniform layout. Another option is to arrange LUTs in a perfect grid pattern. While both design choices are relatively straightforward to implement during physical synthesis, they also come with additional overhead costs that may not be beneficial. In a recent obfuscation research, there has been a growing trend in using eFPGA technology [55, 56]. This approach offers several advantages but is typically employed selectively to protect only specific design parts, thereby minimizing the performance penalty. However, the challenge lies in determining which circuit modules require protection and which do not. The methodology of hASIC is designed to bypass this issue by only revealing (parts of) critical paths as they are selected for static logic. This approach offers a distinct advantage over other methods. The authors in [199] present a top-down methodology for implementing ASICs with eFPGAs. Their designs share many similarities with hASIC solution while incorporating more regularity through logic tiles, similar to those found in commercial FPGAs. hASIC, which does not utilize tiles, prioritizes performance, as shown in the layouts of Figure 34 and the corresponding results in Table 6.

5 Security Analysis

This chapter provides the results of various designs and conducts a thorough security analysis, encompassing both oracle-guided and oracle-less attacks. The security analysis evaluates the resilience of the obfuscated designs against different attack scenarios, identifies potential vulnerabilities, and guides to enhance the overall security of the design.

5.1 Threat Model for hASIC

When considering the threat model for hASIC, the main concern is the untrusted foundry, regardless of whether the adversary is an institutional entity or a rogue employee. The security of the design depends on both the static part, which is fully exposed, and the reconfigurable part, which is protected by a bitstream serving as key. The static part is vulnerable to attacks as the adversary can extract relevant information. An adversary can exploit the regular structure of AES-128 to extract valuable information, thereby making it relatively easy for the adversary to guess the bitstream. The adversary can easily guess the bitstream of reconfigurable part if the static part is too large. On the other hand, if the reconfigurable part is too large, it provides high security, but leads to overheads in terms of PPA. Therefore, it is important to determine the right level of obfuscation to ensure that the design is secure against well-known attacks. Based on these factors, the following assumptions are considered for the security of the hASIC design:

- The adversary aims to reverse engineer the design to copy its IPs, produce excessive amounts of the IC, or implant complex hardware trojans. To accomplish this, the adversary is required to discover the bitstream.
- The adversary may have the objective of determining the circuit or known circuit, even if obfuscation techniques have been implemented. It is worth noting that in this scenario, the adversary does not necessarily need to discover the bitstream.
- Due to their proficiency in IC design, the adversary possesses the necessary expertise and resources to comprehend the layout. They have access to the GDSII file of the hASIC design submitted for fabrication.
- The adversary can identify the standard cells. Consequently, the gate-level netlist of the obfuscated circuit can be retrieved without much difficulty [17].
- Through analyzing the reconfiguration pins, the adversary can easily identify all LUTs and their programming order with no difficulty [200, 201].
- Assuming a perfect reconstruction of LUTs, the adversary can group the standard cells found within the static logic and convert them to a LUT representation.

The threat model summary is depicted in Figure 36. An assessment was conducted to evaluate the security resistance of hASIC against conventional oracle-guided and oracle-less attacks, which are commonly used in LL attacks. All the experiments were performed on a server with 32 processors (Intel(R) Xeon(R) Platinum 8356H CPU @ 3.90GHz) and 1.48TB of RAM. In order to assess the security of hASIC oracle-guided attacks (such as the SAT attack) and oracle-less attacks, security evaluation techniques like Synthesis-based COnstant Propagation Attack for Security Evaluation (SCOPE), as well as custom structural and compositional analysis attacks are utilized.

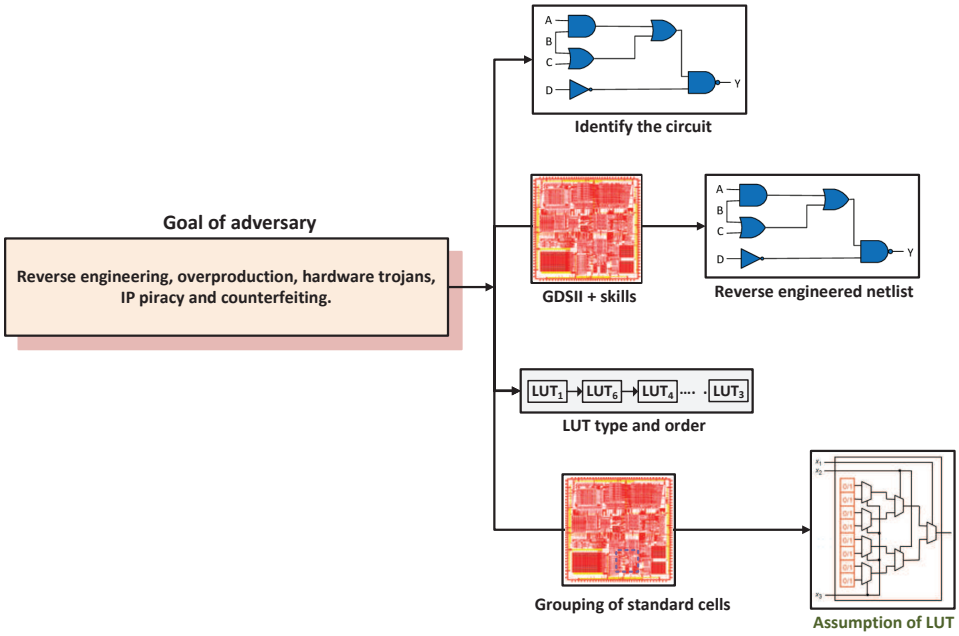


Figure 36: The summary of the threat model for hASIC.

5.2 Oracle-guided Attacks

The goal of the oracle-guided attack is to retrieve a key or a key guess. In hASIC, LUTs act as key gates in contrast to the traditional LL [80]. A single LUT_6 provides 64 bits of key for obfuscation in hASIC. The SBM circuit, presented in Section 3.4, has a total of 25 LUTs, including 11 LUT_6, at an obfuscation rate of 86%. The LUT_6 alone contributes to a key search space of $2^{11\times64}$, which is extremely discouraging for an adversary attempting SAT attacks on hASIC. However, enumerating the key search space may seem simple, but it is a naive approach to evaluate security. Actual attacks, particularly well-known SAT attacks, are necessary. Three different SAT attacks are employed to evaluate the security hardness of hASIC. These attacks are conventional SAT [64], AppSAT [157], and ATPG-based SAT [202]. These attacks operate on bench files and accept only combinational circuits as input. hASIC uses FFs to store a serial input bitstream. A script is written to convert them to combinational logic with parallel key bits.

The ISCAS'85 large combinational circuits, c6288 and c7552, were selected to evaluate hASIC's security against SAT attacks. The results for the selected designs are presented for two different variants of hASIC, baseline and optimized. Figure 37a and Figure 37b show the execution time for different SAT attacks at varying obfuscation rates for c7552 and c6288, respectively. As expected, the execution time increases with the increase in obfuscation level. The region to the left of the green line displays successful SAT attacks, while the region on the right corresponds to unsuccessful attacks where the solver took more than 48 hours to return an answer. Different designs can experience timeouts at varying obfuscation rates when using the SAT solver. c7552 encountered timeouts at a 40% obfuscation rate, while c6288 only experienced them at a 15% obfuscation rate. It is important to note that these designs are extremely small by modern standards.

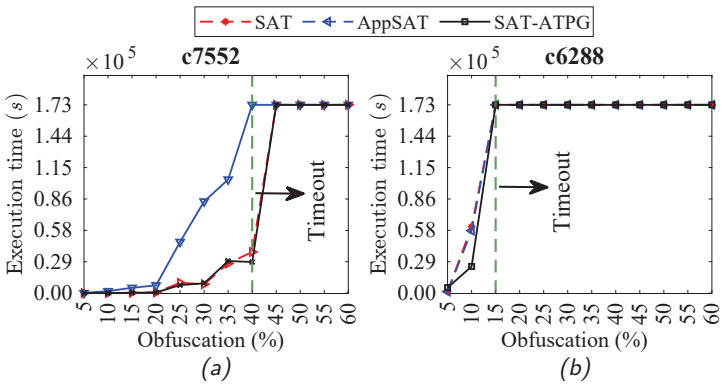


Figure 37: The execution time of SAT attacks.

In this analysis, the SAT problem is considered as a *circuitSAT* problem in order to understand better the behavior of the SAT solver for the selected designs. The analysis is customized specifically for the selected designs. Any other benchmarks may require a different analysis. Figure 38 illustrates the behavior of the SAT solver when dealing with obfuscated circuits. The SAT solver is responsible for determining the satisfiability of a boolean formula. One way to measure the attack convergence probability is by calculating the ratio of variables to clauses of the SAT solver. Figures 38a and 38b show the progression of the variables to clauses ratio for c7552 and c6288, respectively, as the obfuscation level increases. The lower the value of the ratio, the more complex the *circuitSAT* problem becomes. The complexity trend varies with the obfuscation level, but the problem becomes hard after 45% obfuscation level for 7552. On the other hand, the problem becomes difficult after 20% obfuscation level for c6288.

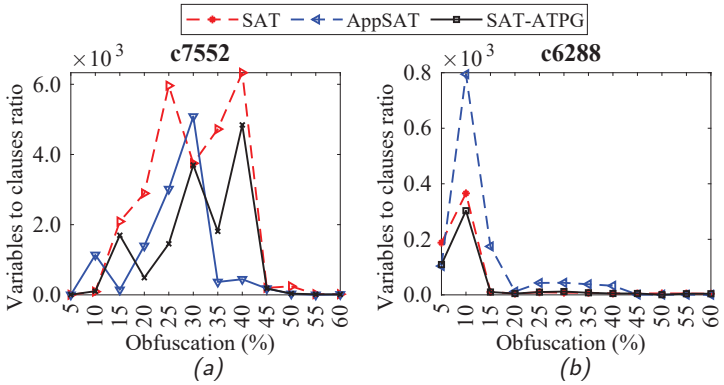


Figure 38: The variables to clauses ratio of SAT attacks for two different designs.

To enhance power, area, and performance, TOTe decomposes LUTs into smaller ones, resulting in improved designs. However, this process also reduces the size of the bitstream, potentially making it vulnerable to attacks. To ensure the security of the optimized version of the c7552 design, it is necessary to verify that the reduced bitstream size does not expose it to existing attacks. Figure 39a shows the execution time for the optimized variant of c7552, while Figure 39b displays the variable to clauses ratio. The security analysis indicates that successful attacks take less time to complete,

but none succeed beyond 40% obfuscation. Additional information on the c7552 design, including bitstream size for different designs and two obfuscation rates (55% and 60%), can be found in Table 7. Interestingly, the decomposed designs exhibit a better variables to clauses ratio, suggesting that the decomposition keeps keys less correlated with each other, making each individual key bit relatively more effective. However, this does not imply that the baseline designs were less secure. The analysis showed that the circuitSAT problem is hard for selected designs, considering different obfuscation levels and lower ratios. For more information on the SAT attack, please refer to [203], and for a discussion on key interference, see [202].

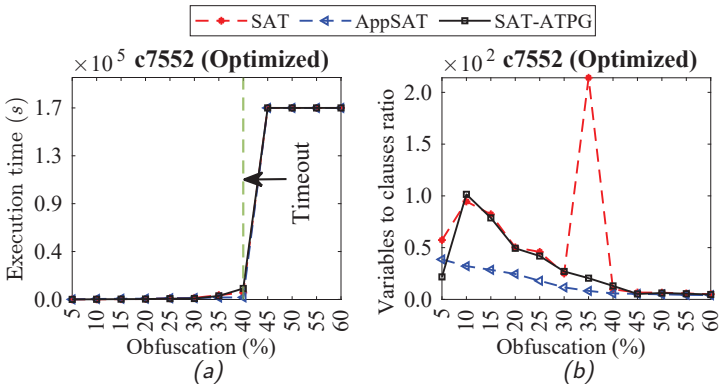


Figure 39: The optimized results for c7552 regarding the execution time and the ratio of variables to clauses in SAT attacks.

Table 7: Analysis of variables to clauses ratio for $obf_c = 55\%$ and 60%

Attack	Obf. (%)	*Bitstream length* (bits)	*Variables*	Clauses	Iterations	Ratio
SAT [64]	55	7494	32318070	1597559	820	17.2
	60	8582	33315150	1898618	540	17.5
	55∗	4014	3434578	598599	139	5.7
	60∗	4406	2829008	564533	106	5.0
AppSAT [157]	55	7994	15843074	1127281	8	14.0
	60	8582	15412584	1181330	7	13.0
	55∗	4014	1320652	302801	2	4.36
	60∗	4406	1419176	346847	2	4.09
ATPG-SAT [202]	55	7494	27243806	1800999	630	15.1
	60	8582	31166810	2247522	636	13.8
	55∗	4014	3642116	664817	148	5.4
	60∗	4406	2274084	480548	85	4.7

∗ Results for the optimized designs

5.3 Oracle-less Attacks

This section discusses oracle-less attacks including the SCOPE attack and custom attacks developed for security analysis. To evaluate the security strength of hASIC, two different attacks have been proposed, one based on the design's structure and the other based on the composition of various circuits. It is believed that knowledge can be gained and information can be extracted by exploiting the static portion of the design, which includes the frequency of specific masking patterns. Such capability would

enable the attacker to reduce the search space for the key that unlocks the design. The frequency of masking patterns can be effectively used as a comparative template for assessing different designs. This means that the composition of LUTs in a design can be vulnerable to structural attacks [159].

5.3.1 SCOPE Attack

The SCOPE attack aims to retrieve a key or a key guess. Oracle-less attacks refer to attacks that do not rely on a functional IC (oracle). Instead, they target the netlist of obfuscated circuits directly. This attack requires no prior knowledge of obfuscation techniques. SCOPE analyzes a single key bit through synthesis and extracts crucial design features, such as area, power, and delay, that may enable the derivation of the correct key bits. Figure 40a compares the execution time for both the baseline and optimized designs of c7552. As shown in Figure 40a, the execution time increases with the level of obfuscation. This trend is observed for both the baseline and optimized designs, with different rates of increase. It is important to note that the COPE metric provides a rough estimate of the level of vulnerability (%) to SCOPE attacks. Figure 40b shows the COPE metric, which decreases with increasing levels of obfuscation.

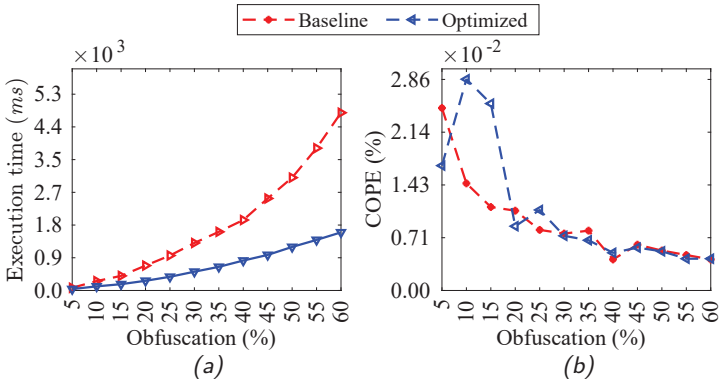


Figure 40: The comparison between the baseline and optimized design for the oracle-less SCOPE attack.

After running SCOPE, a guess is generated for the key with each bit assigned either a '1', a '0', or an 'X' to indicate that it is undetermined. After comparing the guess generated by SCOPE to the known key bits, it was discovered that 50% of the key bits were correctly guessed, which is a random guess regardless of the level of obfuscation. This percentage remains constant for both baseline and optimized designs. Therefore, for hASIC, SCOPE cannot perform better than a random guess.

To identify design intent, a composition analysis attack must have access to a high-quality database of known designs. Therefore, while using TOTe, it is recommended to employ very high obfuscation rates to prevent such attacks. Additionally, reconfigurable-based obfuscation schemes are generally less susceptible to attacks than LL counterparts, making it important to maintain high obfuscation rates.

5.3.2 Structural Analysis Attack

This attack aims to utilize statistical analysis methods to reduce the search space and facilitate the bitstream recovery process. The obfuscation engine used by TOTe consists of six variants of LUTs, with LUT_6 being the most prevalent due to the packing algorithm

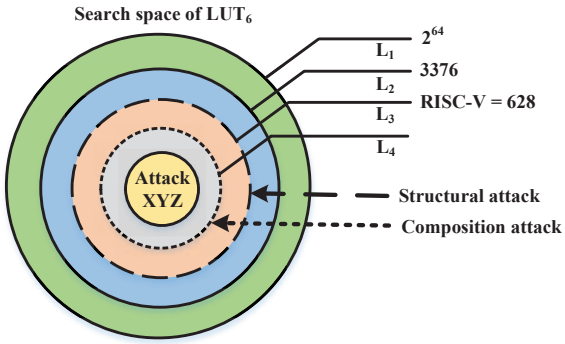


Figure 41: The search space of LUT_6 as it shrinks with different attacks [74]

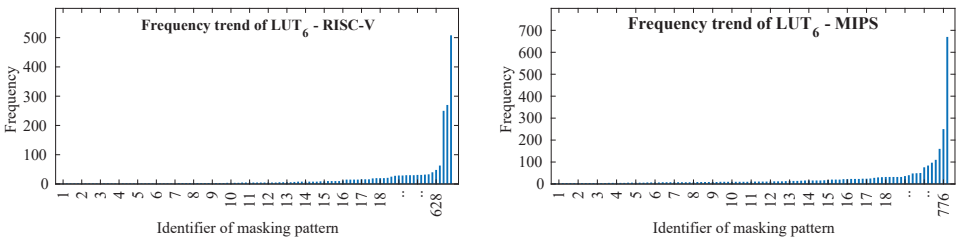
of commercial FPGA synthesis, such as Xilinx [204]. The decomposition method only applies to the reconfigurable part of the design, whereas the initial knowledge obtained by the adversary is from the static part, which remains unchanged regardless of the use of decomposition. The majority of the static part is composed of LUT_6, prompting the adversary to focus their analysis on this type of LUT. Based on the situation analysis, the findings will now be presented. The number of possible keys for a LUT_6 is 2^{64}, but this is only feasible if the FPGA synthesis tool can realistically explore the entire key search space. However, it appears that this is not the case. All unique LUT_6 masking patterns were extracted from the netlists of 31 representative designs of varying size, complexity, and functionality through synthesis. These masking patterns are denoted as ump_i. The results of the analysis for various designs are presented in Table 8. From the third to the eighth column, the table shows the total number of corresponding LUTs and the unique making patterns for LUT_4, LUT_5, and LUT_6. The last three columns of this table illustrate the maximum frequency of the unique masking pattern in the listed design. After analyzing all the unique masking patterns, it was found that the combined number of unique masking patterns for LUT_6 in the fourth column of Table 8 formed a set of $M = \bigcap_{i=1}^{31} |\{ump_i\}| = 3376$ elements, which appears to have settled. As shown in Figure 41, this empirical result reduces the global search space from 2^{64} to $3376 = 2^{11.72}$.

Based on the available information, it appears that an attacker could exploit the frequency of LUTs within a netlist to launch a structural analysis attack. In order to do so, the attacker would need to determine the values of ump_i for a given circuit C_i, despite only having partial knowledge of the design. The question then becomes whether it is possible to estimate ump_i through statistical analysis of a part of C_i. To investigate, two processor designs were analyzed: MIPS and RISC-V. For each circuit, $\langle pattern, frequency \rangle$ tuples were used to track the repetition of masking patterns, with the masking pattern represented by 64-bit hexadecimal numbers and ordered by frequency. Figure 42a and Figure 42b show the bar charts of RISC-V and MIPS, respectively. The MIPS netlist contains 776 unique LUT_6s, with only a few masking patterns that occur more than 50 times. Similarly, in RISC-V, there are 628 unique LUT_6s, with only three occurring more than 100 times.

Figure 43a and Figure 43b investigate the masking pattern frequency for RISC-V and MIPS, respectively. Netlists generated by TOTe at different obfuscation levels were used for this purpose. The experiment assumes that the attacker has visibility of the static part with only a small percentage of LUTs, ranging from 2% to 14%, depending on the obfuscation level. The attacker then tries to predict the distribution of actual

Table 8: Global search space analysis.

Design	Obf. (%)	# LUT_6		# LUT_5		# LUT_4		Max. Frequency		
		Total	Unique	Total	Unique	Total	Unique	# LUT_6	# LUT_5	# LUT_4
c17 [205]	100	0	0	0	0	2	2	0	0	2
c432 [205]	100	20	19	49	36	92	38	3	7	19
c499 [205]	100	40	6	80	12	173	52	17	17	33
c880 [205]	100	34	27	87	57	143	52	5	9	25
c1355 [205]	100	18	3	62	10	88	6	11	23	49
c1908 [205]	100	33	28	89	67	148	59	4	6	22
c2670 [205]	100	51	25	119	49	226	66	17	18	17
c3540 [205]	100	133	86	323	181	570	206	14	27	80
c5315 [205]	100	143	90	333	181	611	211	10	15	39
c6288 [205]	100	419	45	847	94	1760	108	77	78	153
c7552 [205]	100	161	142	378	276	703	281	4	13	57
DES [206]	100	768	92	1593	151	3082	184	114	142	310
RSA [207]	100	586	115	1374	208	2477	177	172	172	472
GFX430 [208]	100	1212	519	3275	951	5149	676	85	193	671
MIPS [209]	100	3162	776	7124	662	13228	488	670	734	1372
JPEG DEC [210]	100	2413	347	5971	619	10585	505	401	510	1090
USB HOST [211]	100	502	123	1138	194	2067	175	264	215	528
CORDIC [212]	100	516	209	1141	330	2175	244	46	185	613
FM [213]	100	188	149	579	311	788	405	9	76	38
SIGMA DELTA [214]	100	32	3	66	7	218	12	29	30	61
openMSP430 [215]	100	760	371	2048	703	3624	509	26	142	439
SBM [191]	100	11	5	26	14	52	20	8	7	15
AES-128[198]	100	9280	45	0	0	1408	178	1153	0	209
SHAKE-256 [216]	100	4438	35	3083	64	5496	53	1395	1215	2584
PID [193]	100	364	175	989	336	1561	317	28	84	50
Median Filter [194]	100	410	27	1077	60	1815	60	97	129	407
SHA-256 [195]	100	1349	69	706	133	96	19	513	50	2584
GPU (OR1200-HP) [197]	100	22611	260	7563	458	758	374	11809	1236	618
RISC-V [217]	100	2240	628	5016	507	9831	404	508	300	1018
FPU [196]	100	823	233	2122	423	3764	373	48	70	372
IIR [192]	100	1	1	4	4	148	3	2	2	2
Total	–	52718	4653	47262	7098	72838	6257	–	–	–



(a) *(b)*

Figure 42: Frequency of masking patterns for RISC-V and MIPS [73].

masking patterns in the design based on their observation of the exposed LUTs in the static portion of hASIC. Polynomial trendlines are used to aid the adversary's guessing attempt in Figure 43. For MIPS and RISC-V, the attacker can estimate to some degree

which masking patterns are unique. Extrapolation is not trivial to determine the actual number of unique maksing patterns, ump_i, since many patterns appear only once or a few times, as shown in Figure 42a and Figure 42b. Some circuits, such as PID, IIR, GPU, SHA-256, etc., have a similar profile where only a few high-frequency LUTs appear. The attack exploits only the static part, but when the decomposition is applied, the adversary needs in-depth knowledge of the decomposition algorithm to estimate the frequency of the unique masking pattern for the reconfigurable part.

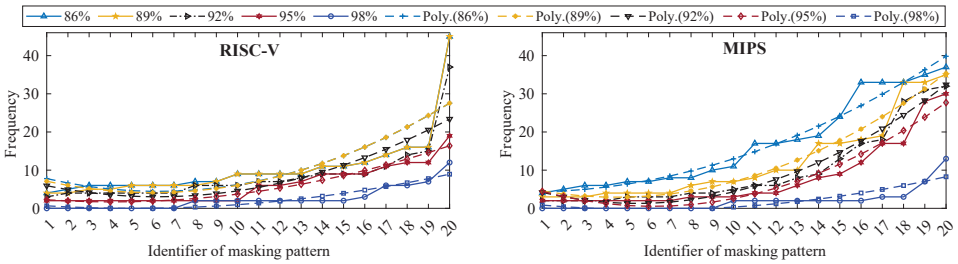


Figure 43: The structural analysis of RISC-V and MIPS [73].

5.3.3 Composition Analysis Attack

This attack aims to correlate the unknown circuit with the known circuits for identification. The adversary's sole objective is deciphering a circuit (specifically, "What is this circuit?"). In this attack, the frequency of the LUTs is also exploited. However, it involves correlating multiple designs with one another based on their composition. It is worth noting that the attack can be deemed successful if the adversary can identify the circuit, rendering the need to break the key unnecessary.

Figure 44a and Figure 44b show correlation analysis for two crypto cores: SHA-256 and AES-128, respectively. The goal of this analysis is to examine the leaked information from the static part against a database[2] of circuits that are known to the attacker. The obfuscation of SHA-256 and AES-128 is performed in the range of 70-100%, followed by the correlation of their static parts with the designs available in the database.

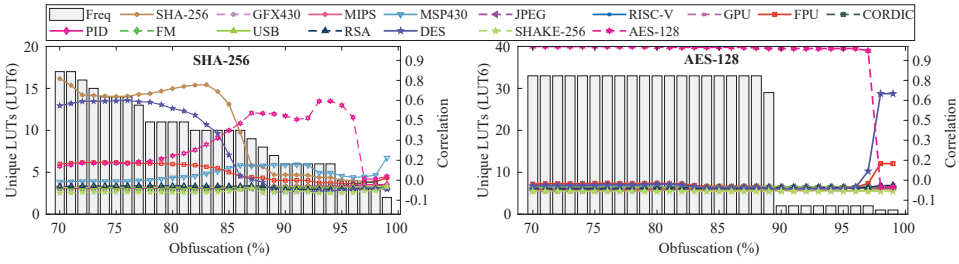


Figure 44: The correlation of SHA-256 and AES-128 versus numerous other designs [73].

Based on the correlation results, interesting trends have been revealed. For SHA-256 in Figure 44a, three regions of interest have been identified depending on the degree of

[2]It is assumed that the adversary can obtain circuits from open-source repositories and execute FPGA synthesis to create a database.

obfuscation: 97-100% (no correlation), 86-96% (strong correlation to another circuit), and 70-85% (correlation to itself). Figure 44b shows a similar analysis for AES-128. The correlation between obfuscated AES-128 and itself is almost one for obfuscation levels below 97%, while the correlation for obfuscated AES-128 versus other designs is almost zero for obfuscation levels in the same range. In the scenario where the adversary can identify the vulnerability of hASIC, it could potentially be equivalent to that of an ASIC design. However, this is not the case for the circuit SHA-256, as opposed to circuit AES-128, where different ranges of obfuscation levels can confuse the adversary. In the instance of circuit SHA-256, this range is found to be between 86-96%. In this case, the search space will be shifted to L_4 for the corresponding design, as shown in Figure 41.

To further reduce the key search space, an adversary interested in obtaining the bitstream could use the correlation analysis described in this study. If the attacker knows that the obfuscated circuits are AES-128, SHA-256, or any other, his/her key guessing will rely on the circuit with the highest correlation. It is important to note that this attack depends on the adversary's ability to reconstruct the LUTs from the static part, and the availability of enough datapoints in the database of known circuits. For example, in the previous subsection, the search space would shrink from 3376 to 776 for MIPS and from 3376 to 628 for RISC-V. As mentioned in Section 5.3.2, this attack only exploits the static part of the design, and a decomposed design does not make it easier or harder to correlate. However, to obtain the actual key, an adversary would need to use other attacks which are not specific to hASIC. These attacks could be either an oracle-guided attack or any 'XYZ' attack, as illustrated in Figure 41.

It is worth noting that hASIC has a highly regular structure upon visual inspection. This characteristic can be adjusted to enhance its effectiveness against RE. One way to achieve this is by mapping LUTs of various sizes to LUT_6, creating a more uniform layout. Another option is to arrange LUTs in a perfect grid pattern. While both design choices are relatively straightforward to implement during physical synthesis, they also come with additional overhead costs that may not be beneficial.

6 Securing the Bitstream of hASIC

The security of the bitstream is of utmost importance when it comes to design obfuscation. This involves encrypting the bitstream, which requires a key to decrypt it. The encryption process ensures security and confidentiality. For the ASIC, SRAM-based PUF is employed as the primary cryptographic key generator to secure the hASIC's bitstream. This chapter explores the design principles of SRAM-based PUFs and their potential as secure key generators. The performance, reliability, and security aspects of the PUF-based encryption scheme for securing the bitstream of hASIC are evaluated.

6.1 Encrypting the Bitstream of hASIC

The security of the bitstream is a critical aspect of obfuscation, as it faces threats from the end-user, who may attempt to tamper with or expose the protected design. To enhance design security, utilizing a cryptocore to encrypt the bitstream is a viable option. hASIC encryption scheme employs the AES algorithm, ensuring that the bitstream is protected from unauthorized access. The level of security provided by the encrypted bitstream is highly reliable, as it cannot be copied or reverse engineered. The encryption scheme uses AES-256. NIST states that approximately are approximately 1.1×10^{77} possible combinations for a 256-bit key [218]. Symmetric encryption algorithms, like AES, use the same key for encryption and decryption. The safety of the data is directly related to the confidentiality of the key. The security of bitstream encryption relies on the confidentiality of the key.

The process of generating a secret key, encrypting, and decrypting a bitstream is illustrated in Figure 45. The encryption chip contains an SRAM-based PUF, error correction logic, control logic, and an AES encryption block for encrypting the bitstream. The SRAM-based PUF generates a unique secret key on the power-up state of bitcells. However, each time the SRAM starts up, a slightly different pattern may emerge, creating a noise component dependent on temperature, voltage ramp, and operating conditions. Despite this noise, it is possible to reconstruct a reliable key every time the SRAM is powered, thanks to error correction, such as "helper data algorithms" [219]. The hASIC bitstream is encrypted with a secret key. The output of the encryption chips is an encrypted bitstream and the secret key. This entire process takes place in a trusted environment.

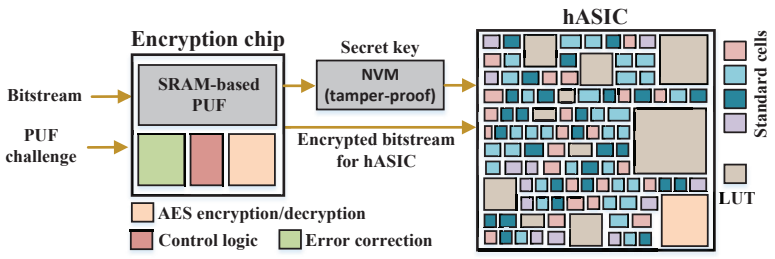


Figure 45: Encryption/decryption scheme of hASIC with SRAM-based PUF.

At a trusted facility, the hASIC is loaded with the secret key from a tamper-proof memory before being fed the bitstream. During configuration, hASIC performs the opposite process by decrypting the incoming bitstream, as illustrated in Figure 45. The encryption logic employed by hASIC uses a 256-bit encryption key. It is essential to note that the AES decryption logic in hASIC is solely dedicated to bitstream decryption

and cannot be utilized for any other purpose. In Chapter 3, it was explained that the hASIC comprises static logic that takes the form of standard cells and reconfigurable logic in the form of LUTs. This way, the AES decryption logic is also integrated into the hASIC as a block, as depicted in Figure 45. If the origin of the encryption key can be trusted and the keys are extracted securely in hardware, they form the so-called "root of trust" of the device. As the security of a bitstream depends solely on the secret key, the robustness of SRAM-based PUFs is crucial in this analysis. The SRAM-based PUF should generally satisfy certain characteristics, which will be discussed in the upcoming sections.

6.2 Internal Architecture of SRAM

This section explains the internal architecture of SRAM before presenting the design of SRAM-based PUF and its evaluation. SRAM relies on the bitcell, consisting of two CMOS inverters connected in a positive feedback loop, to form a bistable storage element. The initial state of each bitcell is determined by the process variation that occurs during the IC's manufacturing process. The stability of each bit is dependent on the degree of threshold voltage mismatch between the local devices. The typical 6T-SRAM cell has a preferred state due to stochastic variations in the threshold voltages of its transistors. The randomness in the initial values of 6T-SRAM results in an unpredictable yet repeatable pattern of zeros and ones that is unique to each device.

Figure 46 clearly illustrates the high-level information of memory architecture. During the placement phase of an ASIC, the designer can rotate memories. Figure 46 illustrates some possible memory rotations. Two types of memories from a major foundry were considered in this study: high-speed and low-density, and low-speed and high-density. The high-speed memory uses standard threshold voltage for both the periphery and bitcells, while the low-speed memory employs mixed threshold voltage for the periphery and high threshold voltage for bitcells. The SRAMs are arranged in an array of memory locations, where each memory access involves reading or writing all the bits in a single location. SRAM macros can be organized in various ways depending on the user's specification for the desired number of addresses and datawidth. The memory compiler automatically makes these decisions.

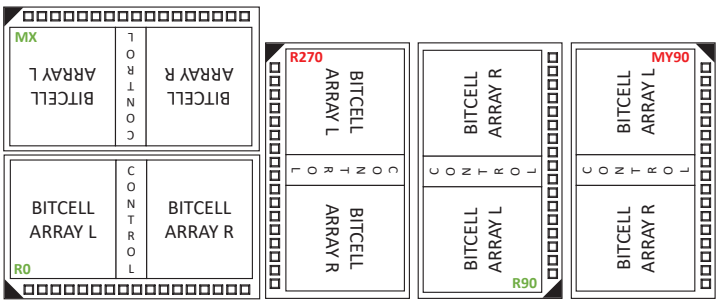


Figure 46: The simplified architecture and orientations of memory [220].

The detailed architecture of a single port low-speed memory is shown in Figure 47. Often, commercial SRAM compilers generate memories with half of the bits on the right and the other half on the left. The control circuitry is located in the center. This arrangement is identical for high-speed and high-density variants. To create large bitcell arrays, a memory matrix (M) of size $j \times k$ is replicated multiple times to form a larger

matrix of size $M \times C$. The memory compiler determines the aspect ratio of the memory and the number of bitcells that need to be MUXed together by selecting values for j, k, and M. For instance, consider a memory with a datawidth of 64 bits and a depth of 128 locations, resulting in a memory of 8Kbits. The address A has a length of 7 bits, where {A0, A1} index the columns, and {A2, A3, A4, A5, A6} index the rows. Here, the memory matrix M has dimensions of 2×4, and C consists of 16 copies of M.

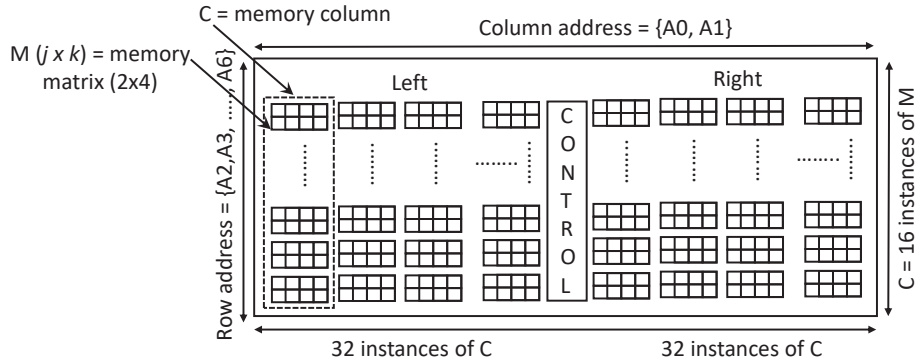


Figure 47: Architecture of low-speed memory with 64-bit datawidth and 128 location depth.

The column mux ratio, denoted by m, is a critical parameter in memory arrays as it determines the number of memory cells connected to a shared bitline. Selecting an appropriate value for m involves a trade-off between memory density, access time, and power consumption. A higher column mux ratio affects the aspect ratio and memory matrix M. However, this also leads to larger capacitance and longer bitlines, causing slower access times and potentially higher power consumption. In contrast, a lower column mux ratio enhances access time. A series of tests were conducted using the foundry compiler to generate multiple memory IPs with varying speeds and densities before designing the SRAM-based PUF. Memories with sizes of 1kbytes and 4kbytes were selected from the results. The memories with a bitcell size of approximately $\sim 0.65 \mu m^2$[3] are labeled as low-density and high-speed, while those with a bitcell size of approximately $\sim 0.50 \mu m^2$ are labeled as high-density but low-speed. This process helped to choose the most appropriate memories for the design.

6.3 Design and Evaluation of SRAM-based PUFs

As depicted in Figure 45, hASIC requires a single SRAM-based PUF to generate the secret key. After selecting suitable representative SRAMs for SRAM-based PUFs, the next step is to find an appropriate and robust SRAM-based PUF for hASICs. In this regard, the investigation has focused on studying the impact of design-time decisions on the effectiveness and quality of SRAM-based PUFs. A chip was designed using a 65nm commercial PDK to assess the impact of various memory- and chip-level parameters. The chip featured eleven SRAM macros, comprehensively evaluating its performance. The SRAM compiler considered several parameters at the memory level, such as the number of addresses, words, aspect ratio, and bitcell design. Furthermore, during the floorplan phase, chip-level decisions were made concerning the placement, rotation, and power delivery strategy of each SRAM macro within the testchip. All of these

[3]The SRAM IPs are generated by a major foundry's compiler and are considered foundry IPs. Further details are omitted.

factors were taken into account for the evaluation. The study analyzed 50 fabricated chips through physical measurements to assess the reliability, bias pattern, entropy, uniqueness, and randomness of different SRAM configurations.

6.3.1 Silicon Demonstration

The primary objective in creating silicon is to demonstrate the assessment of SRAM-based PUFs. A total of 11 SRAMs with different possible orientations have been utilized in the test chip, as depicted in Figure 46. The initial orientation is $R0$, which represents a rotation angle of zero degrees. The abbreviation MX indicates a mirroring process along the x-axis. The symbol R270 indicates a rotation of 270 degrees, while $R90$ denotes a 90-degree rotation. $MY90$ signifies a mirroring process along the y-axis, followed by a 90-degree rotation (all rotations are in the anti-clockwise direction). Figure 48 illustrates the placement of SRAMs and their respective orientations inside the chip. The chip includes six separate SRAM-based PUFs, some replicas identified by underscored letters (a, b, c). This results in eleven SRAM-based PUFs within the chip[4].

The chip has a simple serial interface and eleven different SRAM-based PUFs. All SRAM memories are single-port and use six transistors per bitcell. Additionally, two distinct types of memories were utilized. A streamlined architecture allows for seamless data acquisition across several SRAM-PUFs. A data vector is transmitted via the serial interface using the $shift\_in$ input, while $shift\_in$, $shift\_enable$, and $data\_out\_enable$ serve as control bits. The serial input reads one bit of the data vector during each clock cycle. After the data vector read operation is complete, the shift register accumulates the entire data vector. The serial interface utilizes 15 bits for addressing and selecting eleven SRAM-based PUFs, with 10 bits for accessing the address of the SRAM-PUF[5]. To select the PUF, 4 bits are needed, along with an additional bit to enable the read operation. After loading the shift register, the data is dispersed to memory selection and address block. At the output of the serial interface, 70 bits are retrieved, including 64 bits of data, as well as three start bits and three stop bits. For smaller data widths, such as 1024×32, the length of the data vector is still 64 bits to maintain consistency, but the last 32 bits will be zeros. This will make the read operation more convenient and ensure consistent data-read from the architecture.

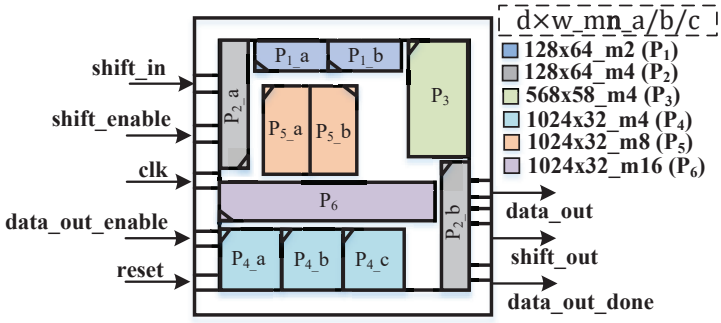
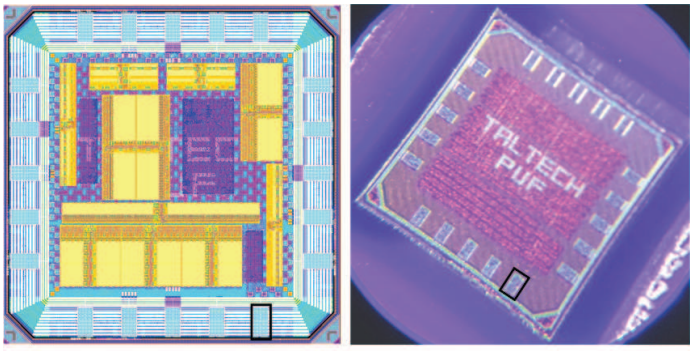


Figure 48: The simplified architecture of the SRAM-based PUFs.

[4]d denotes the depth or number of addresses, w denotes the datawidth, n denotes the number for ratio, and m denotes the column mux ratio.

[5]The maximum number of addresses that can be accommodated is 1024, equivalent to 2^{10}. When dealing with addresses less than 1024, the residual address bits should be set to zero.

The chip comprises fast and slow memories, with smaller ones (128×64) categorized as fast. To maximize the area utilization, all memories were manually placed multiple times. To start, the design was synthesized using the commercial tool Cadence Genus. The chip does not require any special constraint for the timing; a target frequency of 8MHz has been maintained to enable sequential data reading without encountering data corruption. Three flavors of the standard cell library (LVT/SVT/HVT) have been utilized to align with the industrial standard. The chip layout was generated using Cadence Innovus for P&R. The design underwent physical compliance verification, including DRC and Layout Versus Schematic (LVS) checks. The control logic inside the chip is minimal, with most of the area occupied by memories. As a result, the number of buffers, combinational cells, inverters, and sequential cells in the circuit is less than 5%, with 233 buffers, 640 combinational cells, 881 inverters, and 54 sequential cells. After sign-off, the layout is generated as a GDSII file, which is then sent to the foundry for fabrication. The chip was fabricated successfully, and bench tests were conducted to gather data. The layout, shown in Figure 49, reveals the placement of I/O cells and memories, with a black rectangle included to aid in identifying the lower right corner of the chip. The chip's I/O pins and power stripes are visible, running both horizontally and vertically. The placement of the SRAM-based PUFs is clearly visible, with a yellow color in the left panel of the same figure. Signal routing in the design utilized all metals between M2 and M7. To create a power ring around the core, M5 and M6 were employed. In addition, power distribution across the core was achieved through the use of horizontal and vertical stripes in M8 and M9. The layout also includes the seal ring, die and metal fills to meet the foundry requirements. The chip size is $1mm^2$. To validate the design, 50 chip samples were packaged in a DIP-28 form, all confirmed to be fully functional.

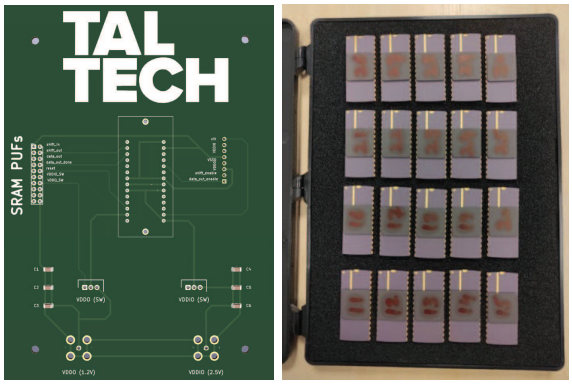


(a) *(b)*

Figure 49: Layout (left) and die micrograph (right) of the fabricated chip. The highlighted pin marks the lower-right corner [220].

6.3.2 Testing and Measurement of SRAM-based PUFs

To evaluate the ASIC prototype, a custom Printed Circuit Board (PCB) was designed and manufactured for this specific task. The PCB contains necessary components, such as a DIP-28 socket, relays, and passive components, to facilitate measurements and filter out power supply noise. Figure 50a illustrates a 2D representation of the PCB layout and component placement. Additionally, Figure 50b shows the images of the received packaged chips, visually representing the final product.



(a) (b)

Figure 50: The designed PCB and fabricated chips.

Figure 51 depicts the testing setup for the chip, where the chip is mounted on the PCB. Raspberry Pi 3 Model B was used to control the chip during testing, enabling smooth communication and efficient management of the testing process. The relays on the PCB played a critical role in controlling the power supply, selectively turning on and off the VDD (1.2V) and VDDIO (3.3V) power sources, allowing for flexibility and control over the chip's power supply configuration.

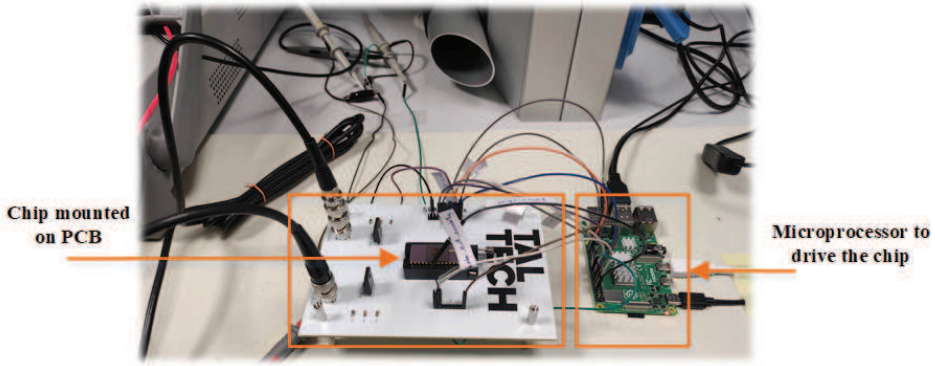


Figure 51: Testing setup for the chip.

In the initial validation phase, the leakage power of all 50 samples was measured. This was done using a picoamp precision ammeter while the chip was idle. The results and distribution of these samples are depicted in Figure 52. The distribution's mean value and standard deviation were 6.70 and 2.17, respectively. The typical case was found to be near the mean value. These results were consistent with the expected power reports obtained during the physical implementation for the typical corner at 25°C with an operating voltage VDD of 1.2V. The best and worst chips are highlighted in terms of performance. These chips consume the highest and lowest leakage currents, as illustrated in Figure 52. The analysis reveals that chip samples do not exhibit a bias towards a specific process corner, but exhibit notable skews between them. This means that process variation significantly impacts the samples, which can ultimately influence the behavior of PUFs. During the second phase of the experiments, the responses of

the PUF were recorded from the chips by power cycling each chip ten times. It was recognized that relays were important for switching the chip on and off, allowing for a passive power cycle completion. To collect the corresponding PUF response, the Raspberry Pi transmitted serial bits to the chip and stored them in a text file. This process was repeated for subsequent experiments, with the chip being turned on and off. The experiment was repeated ten times to assess the stability of the PUF's response. The total number of bits stored was calculated by multiplying the datawidth of each memory by its respective depth. Four instances of 128×64, six instances of 1024×32, and one instance of 568×58 contain a total of 32.768K, 196.608K, and 32.944K bits, respectively. The total number of bits collected is 262.320K per chip for a single power cycle . When considering 10 power cycles, this number increases to 2623.2K bits per chip.

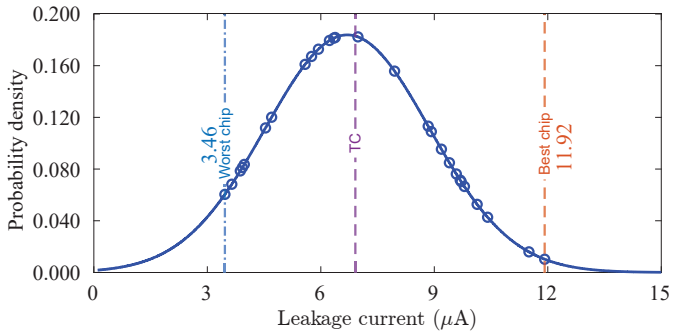


Figure 52: The distribution of leakage current across 50 chips and the typical corner (TC) from physical synthesis.

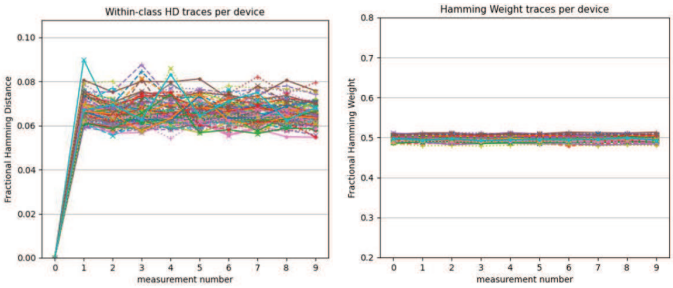
6.4 Results and Observations

This section evaluates the robustness of SRAM-based PUFs for various sizes, including reliability, bias pattern, entropy, uniqueness, and randomness. The metric corresponding to each SRAM-based PUF measurement was evaluated using the data from 50 chips. This analysis was performed for all SRAM-based PUFs within a single chip.

6.4.1 Robustness Evaluation

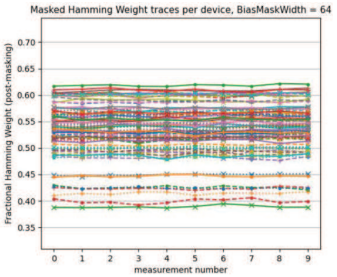
The panels in Figures 53, 54, 55, 56, 57, and 58 illustrate the trends of WCHD, HW, MHW, and BCHD for six distinct SRAM-based PUFs. The x-axis represents the measurement number (power cycles 0-9) and the y-axis shows the corresponding value. The WCHD of all the SRAM-based PUFs is less than 10%, indicating good reliability. Most of the SRAM-based PUFs exhibit a WCHD of approximately 7%, which suggests favorable PUF characteristics for environmental effects. Two chips show a different profile, while the others lie between 5-7%. In order to assess the stability of a specific bitcell's response over time, measurements of WCSHD for P_4 and P_5 were taken. The results indicate that the behavior of the two responses appears to be quite close and unaffected by environmental or measurement noise. The trend of WCSHD indicates that the responses exhibit a WCSHD of approximately 6-7% over measurements.

MHW displays the evaluation of entropy for SRAM-based PUFs. MHW is calculated by XORing the startup pattern with the bias pattern and finding the percentage of ones. The results show that MHW for all SRAM-based PUFs is within the range of 0.5 $\pm$

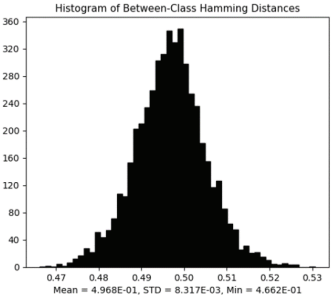


(a) WCHD

(b) HW

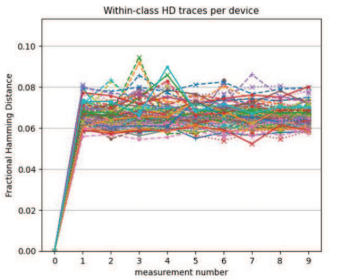


(c) MHW

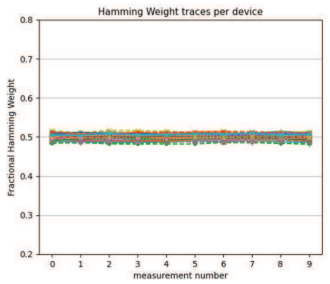


(d) BCHD

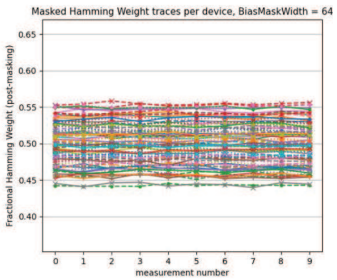
Figure 53: The PUF's characteristics of P_1_a and P_1_b.



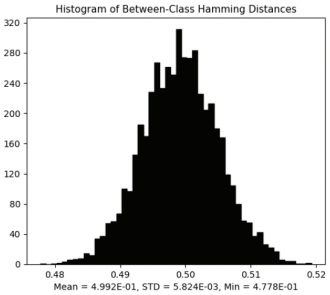
(a) WCHD



(b) HW

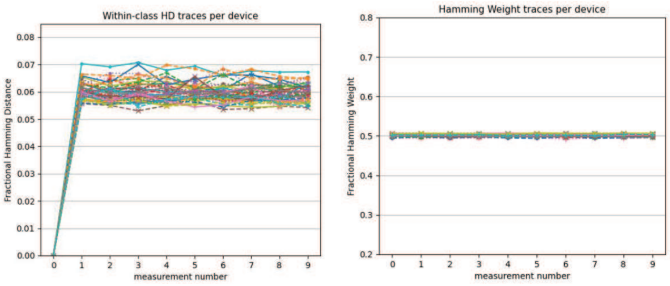


(c) MHW



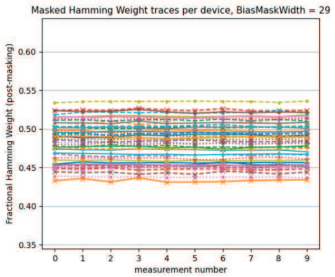
(d) BCHD

Figure 54: The PUF's characteristics of P_2_a and P_2_b.

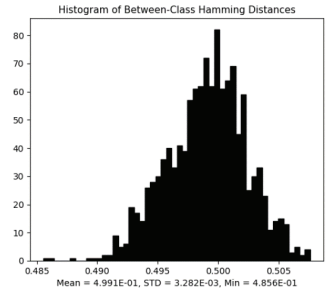


(a) WCHD

(b) HW

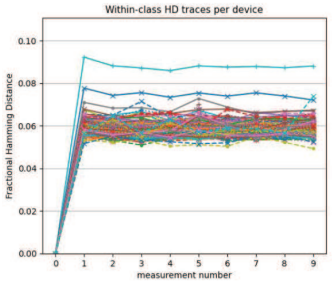


(c) MHW

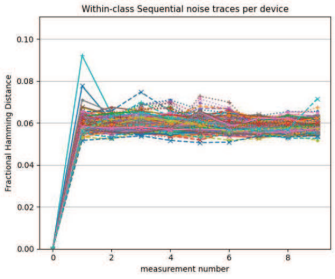


(d) BCHD

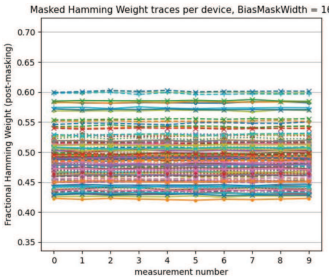
Figure 55: The PUF's characteristics of P_3.



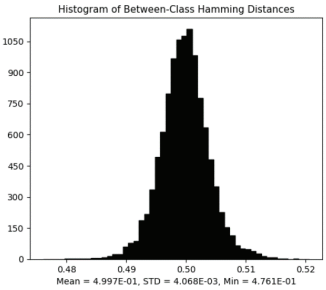
(a) WCHD



(b) WCSHD

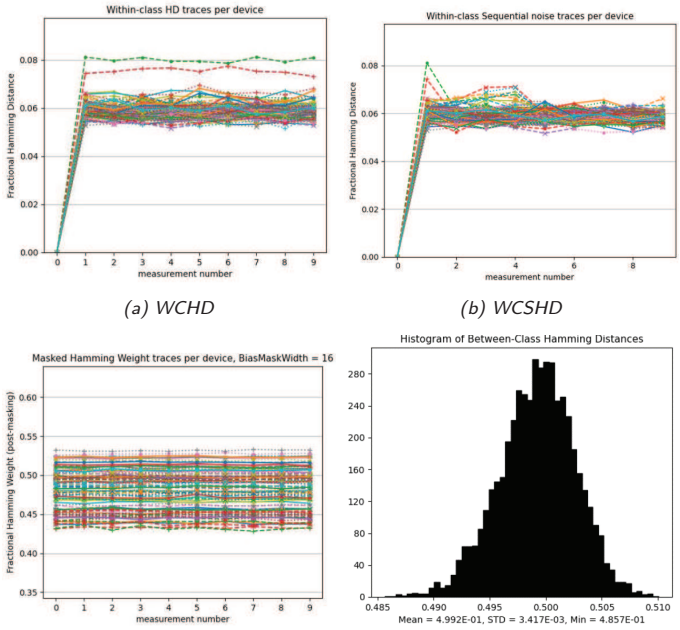


(c) MHW



(d) BCHD

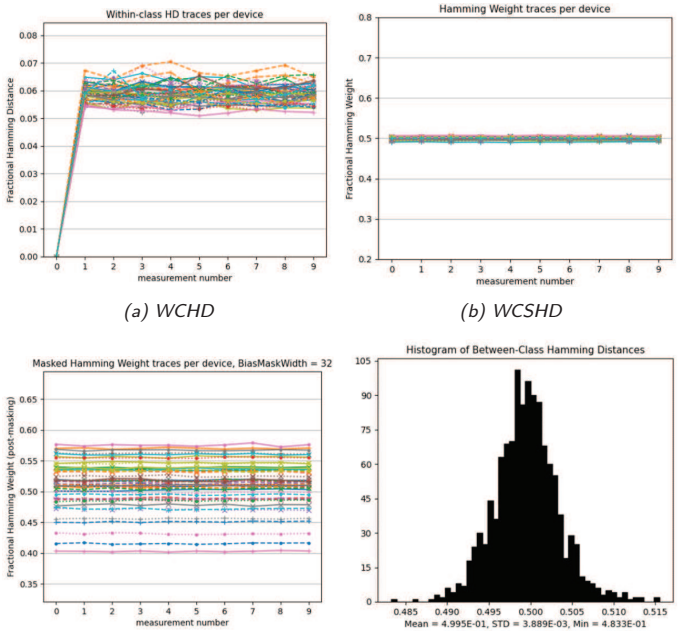
Figure 56: The PUF's characteristics of P_4_a and P_4_b.



(a) WCHD *(b) WCSHD*

(c) MHW *(d) BCHD*

Figure 57: The PUF's characteristics of P_5_a and P_5_b.



(a) WCHD *(b) WCSHD*

(c) MHW *(d) BCHD*

Figure 58: The PUF's characteristics of P_6.

0.1, except for P_1, which exhibits MHW values ranging from 0.38 to 0.62. Although a trend of few chips appears near 0.4, most of the chips were located closer to the ideal value, indicating good entropy. The uniqueness of the PUFs was evaluated in panel (d) of Figures 53, 54, 55, 55, 56, 57 and 58, which shows the BCHD for both fast and slow SRAM-based PUFs, with the probability density on the x-axis and the BCHD values on the y-axis. The distributions are centered around 0.5, and the more narrow the distribution, the better and closer to the ideal value. Overall, the PUFs demonstrate good uniqueness. The randomness of the PUFs was analyzed by computing HW, which yielded a fractional value of 0.5 ± 0.3 for all SRAM-based PUFs, indicating good randomness. Table 9 summarizes the findings. The table's first column lists memory types, while subsequent columns show WCHD, MHW, and BCHD results. The SRAM-based PUF P_4 demonstrated the highest reliability and uniqueness across all 50 chips. Conversely, the SRAM-PUF P_3 showed the highest entropy for all chips. Notably, all SRAM-PUFs exhibited randomness close to the expected 50% value, with most PUFs demonstrating good reliability (over 90%). The variance in entropy highlights that the uniqueness and randomness of all SRAM-PUFs are close to ideal values.

Table 9: Results for the robustness evaluation of SRAM-PUFs

SRAM-PUF	WCHD (%)	MHW	BCHD	Entropy by one-probability
P_1	5.8-8.5	0.391-0.622	0.468-0.526	0.685-1
P_2	5.8-8.8	0.440-0.564	0.479-0.520	0.826-1
P_3	5.5-7.0	0.430-0.539	0.486-0.508	0.811-1
P_4	5.0-9.1	0.435-0.541	0.480-0.519	0.824-1
P_5	5.2-8.0	0.430-0.575	0.486-0.510	0.798-1
P_6	5.1-7.0	0.390-0.580	0.483-0.515	0.713-1

Two additional experiments were conducted to study the behavior of the SRAM-based PUFs placed adjacent to each other in a specific region of the chip. Three identical SRAM-based PUFs (1024×32_m4_a/b/c) were placed in the bottom left corner, as shown in Figure 48. The power mesh of the entire chip was symmetrical and carefully planned for balanced power distribution, except for this region. In the first experiment, the impact of IR drop on the PUF's characteristics was examined, and the results showed that it did not have a measurable effect on the behavior of the SRAM-based PUF. The second experiment analyzed the behavior of the SRAM-based PUF under varying voltage conditions, and the results indicated that the robustness of the SRAM-based PUF was not significantly affected by these voltage conditions. Experiments were also conducted where the two power supplies were turned on at different speeds and in different orders, but there were no measurable changes in SRAM-based PUF quality, due to the chip's power-on-control functionality on its IO cells. This functionality cannot be bypassed, and the core of the chip is only provided power when both VDD and VDDIO are provided. Figure 48 depicts the placement of the three identical SRAM-based PUFs

Table 10 provides a summary of state-of-the-art research on SRAM PUFs. In [101], the authors assessed the uniqueness and WCHD of different PUF architectures, including four identical instances of SRAM-based PUFs, on a 65nm technology node for 192 devices. The findings on WCHD indicated a 95% similarity with similar studies such as [101, 102, 103, 104, 105, 106], with the exception of [101], which considered a lower number of chips. In this study, entropy values were consistent with those reported in [105]. While the entropy values in [107] were slightly higher, a fair comparison was difficult due to their reliance on an older technology node. The work demonstrated good uniqueness, similar to previous studies such as [101, 102, 103, 104, 105, 106, 107], being very close to the ideal value. The results on randomness were also close to the

ideal case and in line with prior research. The purpose of comparing the outcomes is to ensure their consistency with previous research, thereby validating the findings before proceeding with the next analysis. Overall, the study confirmed that SRAM-PUFs behaved as expected and in line with prior studies.

Table 10: Comparison of results

Ref.	# ICs	Tech. (nm)	WCHD (%)	BCHD	Entropy
[101]	192	65	5-5.5	–	–
[102]	40	65	10	0.489	–
[103] †	11	65	15.98	0.495	–
[104]	1	65	15.42	–	–
[105]	10	65	3.3-5.3	–	0.960-1
[106]	17	90	2-4	0.435	–
This work (P_4)	50	65	5-9.1	0.480-0.519	0.824-1

† Authors evaluated 22000 2-bit cells (each die has 2000 cells).

6.4.2 Impact of the Bias Pattern

Next, the analysis focuses on assessing how various factors like sizes, mux selection ratios, memory types, and orientations impact the bias pattern of SRAM-based PUFs. The memory macros are pre-designed, pre-verified memory modules that are crucial to a SoC. They can be embedded in the chip to provide fast and efficient data storage for various tasks. The pins are usually placed on a single edge of the memory. Placing pins along the edges of the memory module makes it easier to connect within the SoC. In this regard, memory orientation plays a vital role in SoC design to maximize performance, minimize power consumption, and reduce the physical footprint of the chip. Physical designers often rotate and flip memory macros within the SoC design to optimize memory placement and routing. Achieving these objectives requires careful consideration of every aspect of chip layout, including memory organization. The memory orientation on the circuit's floorplan does not affect its functionality but impacts the Bias Direction (BD).

The start-up pattern of SRAMs P_5_a, P_6, P_2_a, and P_2_b is illustrated in Figure 59 to visualize the biasing patterns. The response vectors are concatenated into binary data to determine the bias pattern. Each auto-correlation is computed for each chip, but only on the first measurement. Figure 60 depicts the correlation between the MHW for all 50 chips. Notably, the direction of correlation varies from one SRAM-based PUF to another when considering P_1_a as the baseline. For instance, the correlation peaks for the baseline and P_2_b are opposite. Thus, the bias direction is reported as negative. Table 11 summarizes the relationship between memory orientation and bias correlation direction in the last two columns.

Observation 1: Table 11 presents the data width, bias pattern, and number of instances for various SRAM-based PUF instances. It is important to note that the data width affects the bias pattern, but its direction remains unchanged. For instance, consider two identical memories, P_2_a and P_2_b. Despite their identical nature, each memory exhibits a different bias direction, with one having a positive bias direction and the other a negative bias direction.

Observation 2: The change in mux ratio results in different aspect ratios for SRAM-based PUFs, which in turn causes bias patterns to vary between 1024×32_mux8 and 1024×32_mux16. Thus, the bias pattern is impacted by a larger mux ratio. Identical SRAM-based PUFs exhibit an identical bias pattern. However, the column mux ratio

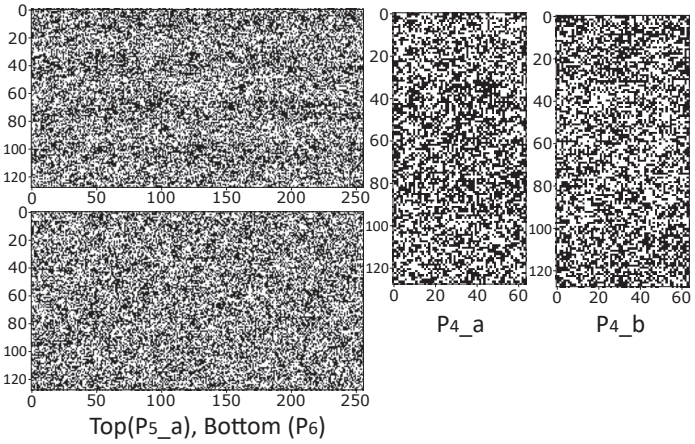


Figure 59: The start-up pattern of the P_5_a, P_6, P_4_a and P_4_b SRAM-based PUFs is represented by a sequence of bits, with white spaces denoting a logical one.

Table 11: The biasing pattern of different SRAM-PUF instances

SRAM-PUF	Bias pattern	Orientation	BD
P_1_a, P_1_b	$0(32)1(64)0(64),...$	R0, R0	+, +
P_2_a, P_2_b	$0(32)1(64)0(64),...$	R90, R270	+, -
P_3	$0(29)1(29),...$	R270	-
P_4_a, P_4_b, P_4_c	$0(16)1(16),...$	MX, MX, MX	+, +, +
P_5_a, P_5_b	$0(16)1(16),...$	R270, MY90	-, -
P_6	$0(16)1(32)0(32),...$	R0	+

does not affect the direction of the bias pattern.

Observation 3: The width and direction of the bias pattern do not correlate with the fast and slow memories. This is also true for using SRAMs with different sizes, bitcells and column mux ratios. Therefore, it can be concluded that different bias widths or directions cannot be achieved by utilizing different bitcells and column mux ratios.

Observation 4: It should be noted that SRAM-based PUFs can exhibit an alternating bias pattern due to the internal structure of SRAMs. As explained in Section 6.3.1, the SRAM macro comprises two halves: one on the left and the other on the right. This arrangement leads to an interesting observation: the initial 32 bits of P_1_a and P_1_b tend to skew towards zero, followed by an alternating pattern of 64 bits.

Observation 5: The study has confirmed that two memory orientations, namely R270 and MY90, exhibit a negative biasing direction that is distinct from the other memories. This bias direction is independent of various memory attributes such as speed, column mux ratio, size, and utilization of SRAMs with different bitcells.

Observation 6: The direction of bias pattern in SRAM-based PUFs remains unaffected by power planning and overall floorplan of the chip, except for factors related to orientation.

Observation 7: The intra-die process variation manifests as uniqueness among individual dies, and therefore, it does not affect the bias pattern's orientation. Additionally, all chips originate from a single wafer and have not undergone rotation on the Multi-Project Wafer (MPW) reticle.

Based on observations 1-7, it has been concluded that the direction of the bias pattern remains unaffected by changes in sizes, mux ratios, memory types, memory

structure, or process variations. The orientation of the pattern is solely dependent on specific factors. With these observations in mind, a hypothesis was developed to validate the direction of the bias pattern.

Hypothesis 1: The effect observed is due to the orientation of the bitcells themselves. In the R90 orientation of the SRAM, the bitcells are positioned vertically compared to the R0 orientation. When the R90 orientation is taken as a reference, the R270 and MY90 orientations flip the left and right bitcell arrays, as shown in Figure 46. This implies that the direction of the bitcell placement associated with the first address of the SRAM gets reversed in these orientations. In simpler terms, the orientation of the bitcell placement corresponding to the first address of the SRAM gets reversed in these orientations.

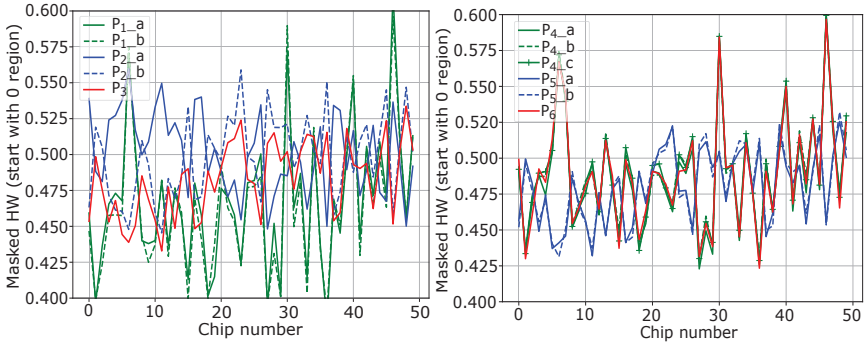


Figure 60: The correlation of SRAM-based PUFs for the bias pattern with baseline SRAM-based PUF (P_1_a) [220].

Hypothesis 2: Analyzing and mitigating the effects of doping variations in SRAM-based PUFs requires a comprehensive understanding of their impact on the bias pattern. The doping of transistors during the lithography process affects their behavior, and variations can occur due to fabrication equipment limitations [221]. The machine moves along the x-axis, doping the transistors from left to right or right to left, then steps up until all transistors are treated. These doping variations impact the SRAM cells' initial state and stability, evaluated using the Static Noise Margin (SNM) concept, which represents the minimum noise voltage required to flip the state of the bitcell. The width-to-length (W/L) ratios of the load and access transistors are set to be as close to 1.0 as possible, while the cell ratio determines the cell's stability and size [221]. Doping variation directly affects the W/L ratio, potentially leading to variations in the SNM value. Transistors on the vertical axis experience more significant variations than adjacent transistors on the x-axis. All SRAM-based PUFs are affected by doping variations, but certain orientations exhibit a distinctive negative bias pattern, such as R270 and MY90, where the transistor arrangement becomes reversed or opposite to the baseline SRAM-based PUF's orientation.

Regarding Figure 45, the error correction block is placed on the other chip which is executed in the trusted environment. There are two phases in error correction techniques for PUF: enrollment and reconstruction [94]. The PUF response is converted into a codeword during enrollment using an error correcting code [222, 223]. The mapping information is stored in the helper data, which is designed not to leak any information about the key. In the encryption scheme, the error correction IP contains the helper data necessary for key reconstruction. However, any modifications to the helper data,

whether malicious or not, will prevent key reconstruction. Additionally, the helper data is only valid for the chip on which it was created [219]. During the reconstruction phase, the hASIC performs a new noisy PUF measurement and extracts the PUF key (without noise) from the helper data and the new PUF response. Figure 45 presents a scheme that explains the complete encryption and decryption of a bitstream. The analysis of SRAM-based PUFs suggests a way to select the most robust SRAM-based PUF from the various ones included in this study. Generating a key from these PUFs provides a reliable and cost-effective solution. While any SRAM can be used as a PUF, the designer should consider using the most robust SRAM-based PUF to generate a secret key.

7 Conclusions and Future Directions

In the current era of technology incorporating AI and IoT, high-performance ICs have become essential for accomplishing everyday tasks. Nonetheless, these applications necessitate the fabrication of ICs that are based on advanced technology. Establishing and maintaining an advanced foundry that can fabricate ICs at this level demands billions of dollars. As a result, semiconductor vendors outsource the fabrication process to untrusted foundries. Moreover, other aspects of the manufacturing process, such as testing and packaging, are also outsourced. This has resulted in the globalization of the entire supply chain. The globalized IC supply chain faces many potential threats. Over time, various countermeasures have been proposed, such as LL, to enhance security. However, these solutions are susceptible to numerous attacks and the field is still evolving. New solutions, such as reconfigurable-based obfuscation, have been emerging.

The fundamental contribution of this thesis is developing a specialized obfuscation tool named "TOTe" that generates a hybrid solution called "hASIC", using reconfigurable elements. hASIC comprises reconfigurable LUTs and static standard cells. The current reconfigurable-based solution minimizes the PPA overheads by keeping the reconfigurable part as minimal as possible. However, the finding of this thesis is that an hASIC solution contrasts with the current practice of reconfigurable-based obfuscations. TOTe's design-security trade-off affects performance according to experimental results from various designs. Performance increases with decreased obfuscation level and area overheads. The results were validated in a commercial physical synthesis tool with industry-standard timing and power analysis. The initial analysis performed by TOTe is confirmed in the physical synthesis, where a similar trend is observed in the design versus security trade-offs. The FPGA implementation achieved 103 MHz and 77 MHz for AES-128 and SHA-256, respectively. However, the physical synthesis of the AES-128 and SHA-256 designs indicates a performance of 240 MHZ and 248 MHZ. TOTe has also incorporated LUT decomposition and pin swapping to enhance performance and reduce the size of hASIC designs. After optimization and pin swap, the frequency of SHA-256 was increased to 368 MHz. Based on the results obtained from SHA-256, it is evident that optimization improves the performance between 40-50 % and reduces the design size by almost 35%, resulting in a significant boost in performance and a compact design.

When it comes to security, an adversary can distinguish between the static and reconfigurable elements of hASIC. However, they must also be capable of reverting LUTs back to standard cells and determining the correct sequence of the LUTs. Based on the results of various attacks, a thorough security analysis has confirmed that high obfuscation rates are necessary. Oracle-guided attacks have confirmed that the obfuscated designs are resistant to SAT attacks. The same holds for optimized designs utilizing LUT decomposition. In oracle-less attacks, adversaries can only retrieve up to 50% correct bits through guessing. Furthermore, the customized attacks confirm that the adversary can predict the circuit and narrow down the search space for further attacks. In a nutshell, designs obfuscated by hASIC are highly secure against these attacks.

To fully utilize the fabricated hASIC chip, a bitstream is required. To ensure security at the end-user level, the bitstream is encrypted. The encryption of the bitstream is dependent on the reliability and security of the key extracted from the SRAM-based PUFs. To find a suitable and robust PUF for hASIC, a study was conducted on various SRAM-based PUFs. The experiments related to SRAM-based PUF revealed that orientation affects the bias direction significantly. This observation could prove useful for designing smart error correction algorithms for SRAM-based PUFs. The

study on the evaluation characteristics of SRAM-based PUFs, including reliability, bias pattern, entropy, uniqueness, and randomness, aligns with prior research and confirms that SRAM-based PUFs are representative. However, the study also emphasizes the significance of meticulous physical synthesis in the design of SRAM-based PUFs.

Overall, this research adds valuable insights to the field of obfuscation research. For future work, the research on reconfigurability in hASIC can be extended to reap its benefits. This includes fixing design bugs, achieving potential side-channel resilience, and further optimization. When most of the hASIC is reconfigurable, it allows to correct bugs and feed the correct bitstream after fabrication. Analyzing side-channel leakage and resistance is beneficial for enhancing the security of hASIC. Performance is bottlenecked when a significant portion of reconfigurability is used, thus further optimization techniques can be useful in balancing the design-security trade-offs.

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] PHYSEC, "Digitalization of critical infrastructure," last accessed on Sep 11, 2023. Available at: https://www.physec.de/en/manufacturer/industry/digitalization-of-critical-infrastructure/.

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

[4] IEEE Digital Reality, "The impacts that digital transformation has on society," last accessed on Feb 02, 2023. Available at: https://digitalreality.ieee.org/publications/impacts-of-digital-transformation.

[5] International Roadmap for Devices and Systems, "Semiconductors and artificial intelligence," last accessed on Mar 02, 2023. Available at: https://irds.ieee.org/topics/semiconductors-and-artificial-intelligence.

[6] ElectronicsHub, "Integrated circuits-types , uses," last accessed on Mar 30, 2023. Available at: https://www.electronicshub.org/integrated-circuits-types-uses/.

[7] International Roadmap for Devices and Systems, "High-end performance packaging 2022 – focus on 2.5d/3d integration," last accessed on Mar 03, 2023. Available at: https://www.yolegroup.com/product/report/high-end-performance-packaging-2022--focus-on-25d3d-integration/.

[8] F. Alan, "TSMC's new 2nm chip production fab will cost it how much?," last accessed on Jun 20, 2022. Available at: https://www.phonearena.com/news/tsmc-to-spend-fortune-on-2nm-production-fab_id140626.

[9] MacRumors, "Apple books nearly 90% of tsmc's 3nm production capacity for this year," last accessed on Jun 24, 2022. Available at: https://www.macrumors.com/2023/05/15/apple-tsmc-3nm-production-capacity/.

[10] 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.

[11] M. Yasin, J. J. Rajendran, and O. Sinanoglu, "The need for logic locking," in *Trustworthy Hardware Design: Combinational Logic Locking Techniques*, pp. 1–16, Cham: Springer International Publishing, 2020.

[12] S. Agam, "Europe, US warn of fake-chip danger to national security, critical systems," last accessed on Feb 02, 2023. Available at: https://www.theregister.com/2022/03/18/eu_us_counterfeit_chips/.

[13] EUIPO-ITU, "EUIPO-ITU report: The economic cost of IPR infringement in the smartphones sector," last accessed on Mar 02, 2023. Available at: https://www.itu.int/en/ITU-D/Regulatory-Market/Pages/Counterfeiting/SmartphonesStudy.aspx.

[14] SECLORE, "Intellectual property (IP) theft in the semiconductor industry: Innovation at risk," last accessed on Sep 20, 2023. Available at: https://www.seclore.com/wp-content/uploads/2023/08/Seclore-Intellectual-Property-IP-Theft-in-Semiconductor-Industry.pdf.

[15] A. Matthew, "Supply chain threats against integrated circuits," last accessed on Jan 02, 2023. Available at: https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/supply-chain-threats-v1.pdf.

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

[17] 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.

[18] P. Subramanyan, N. Tsiskaridze, K. Pasricha, D. Reisman, A. Susnea, and S. Malik, "Reverse engineering digital circuits using functional analysis," in *Proceedings of the Conference on Design, Automation and Test in Europe*, DATE '13, (San Jose, CA, USA), p. 1277–1280, EDA Consortium, 2013.

[19] N. Albartus, M. Hoffmann, S. Temme, L. Azriel, and C. Paar, "DANA universal dataflow analysis for gate-level netlist reverse engineering," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 4, pp. 309–336, 2020.

[20] R. Kibria, M. Sazadur Rahman, F. Farahmandi, and M. Tehranipoor, "RTL-FSMx: Fast and accurate finite state machine extraction at the RTL for security applications," in *2022 IEEE International Test Conference (ITC)*, pp. 165–174, 2022.

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

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

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

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

[25] F. Koeune and F.-X. Standaert, "A tutorial on physical security and side-channel attacks," in *Foundations of Security Analysis and Design III: FOSAD 2004/2005 Tutorial Lectures* (A. Aldini, R. Gorrieri, and F. Martinelli, eds.), pp. 78–108, Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[26] B. Yang, K. Wu, and R. Karri, "Scan based side channel attack on dedicated hardware implementations of data encryption standard," in *2004 International Conferce on Test*, pp. 339–344, 2004.

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

[28] Rambus Press, "Side-channel attacks explained: everything you need to know," last accessed on Aug 27, 2023. Available at: `https://www.rambus.com/blogs/side-channel-attacks/#what`.

[29] S. Engels, M. Hoffmann, and C. Paar, "A critical view on the real-world security of logic locking," *Journal of Cryptographic Engineering*, vol. 12, pp. 229–244, Sep 2022.

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

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

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

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

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

[35] 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.

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

[37] 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.

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

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

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

[41] Y. M. Alkabani and F. Koushanfar, "Active hardware metering for intellectual property protection and security," in *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium*, SS'07, (USA), USENIX Association, 2007.

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

[43] F. Koushanfar, "Hardware metering: A survey," in *Introduction to Hardware Security and Trust* (M. Tehranipoor and C. Wang, eds.), pp. 103–122, New York, NY: Springer New York, 2012.

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

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

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

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

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

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

[50] G. Kolhe, T. Sheaves, K. I. Gubbi, T. Kadale, S. Rafatirad, S. M. PD, A. Sasan, H. Mahmoodi, and H. Homayoun, "Silicon validation of LUT-based logic-locked IP cores," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 1189–1194, 2022.

[51] S. D. Chowdhury, G. Zhang, Y. Hu, and P. Nuzzo, "Enhancing SAT-attack resiliency and cost-effectiveness of reconfigurable-logic-based circuit obfuscation," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 2021.

[52] G. Kolhe, S. M. PD, S. Rafatirad, H. Mahmoodi, A. Sasan, and H. Homayoun, "On custom LUT-based obfuscation," in *Proceedings of the 2019 on Great Lakes Symposium on VLSI*, GLSVLSI '19, p. 477–482, Association for Computing Machinery, 2019.

[53] G. Kolhe, H. M. Kamali, M. Naicker, T. D. Sheaves, H. Mahmoodi, P. D. Sai Manoj, H. Homayoun, S. Rafatirad, and A. Sasan, "Security and complexity analysis of LUT-based obfuscation: From blueprint to reality," in *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–8, 2019.

[54] G. Kolhe, S. Salehi, T. D. Sheaves, H. Homayoun, S. Rafatirad, M. P. Sai, and A. Sasan, "Securing hardware via dynamic obfuscation utilizing reconfigurable interconnect and logic blocks," in *2021 58th ACM/IEEE Design Automation Conference (DAC)*, pp. 229–234, IEEE, 2021.

[55] B. Hu, T. Jingxiang, S. Mustafa, R. R. Gaurav, S. William, M. Yiorgos, C. S. Benjamin, and S. Carl, "Functional obfuscation of hardware accelerators through selective partial design extraction onto an embedded FPGA," in *Proceedings of the 2019 Great Lakes Symposium on VLSI*, p. 171–176, 2019.

[56] J. Chen, M. Zaman, Y. Makris, R. D. S. Blanton, S. Mitra, and B. C. Schafer, "DECOY: DEflection-Driven HLS-Based Computation Partitioning for Obfuscating Intellectual PropertY," in *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference*, DAC '20, IEEE Press, 2020.

[57] M. M. Shihab, J. Tian, G. R. Reddy, B. Hu, W. Swartz, B. Carrion Schaefer, C. Sechen, and Y. Makris, "Design obfuscation through selective post-fabrication transistor-level programming," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 528–533, 2019.

[58] P. Mohan, O. Atli, J. Sweeney, O. Kibar, L. Pileggi, and K. Mai, "Hardware redaction via designer-directed fine-grained eFPGA insertion," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1186–1191, IEEE, 2021.

[59] J. Chen and B. C. Schafer, "Area efficient functional locking through coarse grained runtime reconfigurable architectures," in *Proceedings of the 26th Asia and South Pacific Design Automation Conference*, pp. 542–547, 2021.

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

[61] M. El Massad, S. Garg, and M. V. Tripunitara, "The SAT attack on IC camouflaging: Impact and potential countermeasures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 8, pp. 1577–1590, 2020.

[62] 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.

[63] 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.

[64] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 137–143, 2015.

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

[66] 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.

[67] A. Mondal, M. Zuzak, and A. Srivastava, "StatSAT: A boolean satisfiability based attack on logic-locked probabilistic circuits," in *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2020.

[68] N. Limaye, S. Patnaik, and O. Sinanoglu, "Valkyrie: Vulnerability assessment tool and attack for provably-secure logic locking techniques," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 744–759, 2022.

[69] S. Patnaik, N. Limaye, and O. Sinanoglu, "Hide and seek: Seeking the (un)-hidden key in provably-secure logic locking techniques," *IEEE Transactions on Information Forensics and Security*, vol. 17, pp. 3290–3305, 2022.

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

[71] B. Liu and B. Wang, "Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks," in *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–6, 2014.

[72] H. Mardani Kamali, K. Zamiri Azar, K. Gaj, H. Homayoun, and A. Sasan, "LUT-lock: A novel LUT-based logic obfuscation for FPGA-bitstream and ASIC-hardware protection," in *2018 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pp. 405–410, 2018.

[73] Z. U. Abideen, T. D. Perez, M. Martins, and S. Pagliarini, "A security-aware and LUT-based CAD flow for the physical synthesis of hASICs," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, pp. 1–1, 2023.

[74] 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.

[75] A. Attaran, T. D. Sheaves, P. K. Mugula, and H. Mahmoodi, "Static design of spin transfer torques magnetic look up tables for ASIC designs," in *Proceedings of the 2018 on Great Lakes Symposium on VLSI*, pp. 507–510, 2018.

[76] T. Winograd, H. Salmani, H. Mahmoodi, K. Gaj, and H. Homayoun, "Hybrid STT-CMOS designs for reverse-engineering prevention," in *Proceedings of the 53rd Annual Design Automation Conference*, pp. 1–6, 2016.

[77] J. Yang, X. Wang, Q. Zhou, Z. Wang, H. Li, Y. Chen, and W. Zhao, "Exploiting spin-orbit torque devices as reconfigurable logic for circuit obfuscation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 1, pp. 57–69, 2018.

[78] G. Kolhe, T. D. Sheaves, S. M. P. D., H. Mahmoodi, S. Rafatirad, A. Sasan, and H. Homayoun, "Breaking the design and security trade-off of look-up table-based obfuscation," *ACM Trans. Des. Autom. Electron. Syst.*, 2022.

[79] M. M. Shihab, B. Ramanidharan, S. S. Tellakula, G. Rajavendra Reddy, J. Tian, C. Sechen, and Y. Makris, "ATTEST: Application-agnostic testing of a novel transistor-level programmable fabric," in *2020 IEEE 38th VLSI Test Symposium (VTS)*, pp. 1–6, 2020.

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

[81] J. Bhandari, A. K. Thalakkattu Moosa, B. Tan, C. Pilato, G. Gore, X. Tang, S. Temple, P.-E. Gaillardon, and R. Karri, "Exploring eFPGA-based redaction for IP protection," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, 2021.

[82] S. Patnaik, N. Rangarajan, J. Knechtel, O. Sinanoglu, and S. Rakheja, "Advancing hardware security using polymorphic and stochastic spin-hall effect devices," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 97–102, IEEE, 2018.

[83] C. M. Tomajoli, L. Collini, J. Bhandari, A. K. T. Moosa, B. Tan, X. Tang, P.-E. Gaillardon, R. Karri, and C. Pilato, "ALICE: An automatic design flow for eFPGA redaction," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, p. 781–786, 2022.

[84] N. Rangarajan, S. Patnaik, J. Knechtel, R. Karri, O. Sinanoglu, and S. Rakheja, "Opening the doors to dynamic camouflaging: Harnessing the power of polymorphic devices," *IEEE Transactions on Emerging Topics in Computing*, 2020.

[85] C. Sathe, Y. Makris, and B. C. Schafer, "Investigating the effect of different eFPGAs fabrics on logic locking through HW redaction," in *2022 IEEE 15th Dallas Circuit And System Conference (DCAS)*, pp. 1–6, IEEE, 2022.

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

[87] M. Yasin, B. Mazumdar, S. S. Ali, and O. Sinanoglu, "Security analysis of logic encryption against the most effective side-channel attack: DPA," in *2015 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFTS)*, pp. 97–102, 2015.

[88] A. Duncan, F. Rahman, A. Lukefahr, F. Farahmandi, and M. Tehranipoor, "FPGA bitstream security: A day in the life," in *2019 IEEE International Test Conference (ITC)*, pp. 1–10, 2019.

[89] Xilinx, Inc., "Using encryption and authentication to secure an ultrascale/ultrascale+ FPGA bitstream," last accessed on Oct 20, 2022. Available at: https://www.xilinx.com/content/dam/xilinx/support/documents/application_notes/xapp1267-encryp-efuse-program.pdf.

[90] A. Moradi and T. Schneider, "Improved side-channel analysis attacks on xilinx bitstream encryption of 5, 6, and 7 series," in *Constructive Side-Channel Analysis and Secure Design* (F.-X. Standaert and E. Oswald, eds.), (Cham), pp. 71–87, Springer International Publishing, 2016.

[91] F. Benz, A. Seffrin, and S. A. Huss, "Bil: A tool-chain for bitstream reverse-engineering," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, pp. 735–738, 2012.

[92] P. Swierczynski, "Bitstream-based attacks against reconfigurable hardware," last accessed on Oct, 10 2023. Available at: https://www.langer-emv.de/fileadmin/2017_Bitstream-basedattacksagainstreconfigurablehardware-34.pdf.

[93] National Institute of Standards and Technology (NIST), "Advanced Encryption Standard (AES)," last accessed on Aug 20, 2023. Available at: https://csrc.nist.gov/files/pubs/fips/197/final/docs/fips-197.pdf.

[94] Intrisic ID, "SRAM PUF: The secure silicon fingerprint," last accessed on Sep 11, 2023. Available at: https://www.intrinsic-id.com/wp-content/uploads/2023/03/2023-03-09-White-Paper-SRAM-PUF-The-Secure-Silicon-Fingerprint.pdf.

[95] S. Gören, O. Ozkurt, A. Yildiz, and H. F. Ugurdag, "FPGA bitstream protection with PUFs, obfuscation, and multi-boot," in *6th International Workshop on Reconfigurable Communication-Centric Systems-on-Chip (ReCoSoC)*, pp. 1–2, 2011.

[96] S. S. Mansouri and E. Dubrova, "Ring oscillator physical unclonable function with multi level supply voltages," in *2012 IEEE 30th International Conference on Computer Design (ICCD)*, pp. 520–521, 2012.

[97] K. Fruhashi, M. Shiozaki, A. Fukushima, T. Murayama, and T. Fujino, "The arbiter-PUF with high uniqueness utilizing novel arbiter circuit with delay-time measurement," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, pp. 2325–2328, 2011.

[98] J. Miskelly and M. O'Neill, "Fast DRAM PUFs on commodity devices," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 3566–3576, 2020.

[99] S. Zhang, B. Gao, D. Wu, H. Wu, and H. Qian, "Evaluation and optimization of physical unclonable function (PUF) based on the variability of FinFET SRAM," in *2017 International Conference on Electron Devices and Solid-State Circuits (EDSSC)*, pp. 1–2, 2017.

[100] K.-H. Chuang, E. Bury, R. Degraeve, B. Kaczer, D. Linten, and I. Verbauwhede, "A physically unclonable function using soft oxide breakdown featuring 0% native BER and 51.8 fj/bit in 40-nm CMOS," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 10, pp. 2765–2776, 2019.

[101] R. Maes, V. Rozic, I. Verbauwhede, P. Koeberl, E. van der Sluis, and V. van der Leest, "Experimental evaluation of physically unclonable functions in 65 nm CMOS," in *2012 Proceedings of the ESSCIRC (ESSCIRC)*, pp. 486–489, 2012.

[102] S. Baek, G.-H. Yu, J. Kim, C. T. Ngo, J. K. Eshraghian, and J.-P. Hong, "A reconfigurable SRAM based CMOS PUF with challenge to response pairs," *IEEE Access*, vol. 9, pp. 79947–79960, 2021.

[103] Y. Shifman, A. Miller, Y. Weizmann, and J. Shor, "A 2 bit/cell tilting sram-based PUF with a BER of 3.1e-10 and an energy of 21 fj/bit in 65nm," *IEEE Open Journal of Circuits and Systems*, vol. 1, pp. 205–217, 2020.

[104] A. B. Alvarez, W. Zhao, and M. Alioto, "Static physically unclonable functions for secure chip identification with 1.9–5.8% native bit instability at 0.6–1 v and 15 fj/bit in 65 nm," *IEEE Journal of Solid-State Circuits*, vol. 51, no. 3, pp. 763–775, 2016.

[105] G.-J. Schrijen and V. van der Leest, "Comparative analysis of SRAM memories used as PUF primitives," in *2012 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1319–1324, 2012.

[106] G. Selimis, M. Konijnenburg, M. Ashouei, J. Huisken, H. de Groot, V. van der Leest, G.-J. Schrijen, M. van Hulst, and P. Tuyls, "Evaluation of 90nm 6t-sram as physical unclonable function for secure key generation in wireless sensor nodes," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, pp. 567–570, 2011.

[107] R. Wang, G. Selimis, R. Maes, and S. Goossens, "Long-term continuous assessment of SRAM PUF and source of random numbers," in *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 7–12, 2020.

[108] A. Van Herrewege, A. Schaller, S. Katzenbeisser, and I. Verbauwhede, "DEMO: Inherent PUFs and secure PRNGs on commercial off-the-shelf microcontrollers," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, (New York, NY, USA), p. 1333–1336, Association for Computing Machinery, 2013.

[109] NobelPrize.org, "Nobel lecture: Miniaturization of electronic circuits–the past and the future," last accessed on Apr 30, 2023. Available at: `https://www.nobelprize.org/prizes/physics/2000/kilby/lecture/`.

[110] J. Bardeen and W. Brattain, "The transistor, a semiconductor triode," *Proceedings of the IEEE*, vol. 86, no. 1, pp. 29–30, 1998.

[111] Industrial Alchemy, "Texas instruments SN514 integrated circuit," last accessed on Sep 20, 2023. Available at: `https://www.industrialalchemy.org/articleview.php?item=741`.

[112] HC (History Computer), "Integrated circuit (IC) explained — everything you need to know," last accessed on May 30, 2023. Available at: https://history-computer.com/integrated-circuit/.

[113] P. Chaturvedi, "Wafer scale integration: a review," *Microelectronics Journal*, vol. 19, no. 2, pp. 4–35, 1988.

[114] The Silicon Engine, "The timeline of semiconductors in computers," last accessed on Sep 2, 2023. Available at: https://www.computerhistory.org/siliconengine/.

[115] O. Doug, "Lessons from history: The 1980s semiconductor cycle(s)," last accessed on May 08, 2022. Available at: https://www.fabricatedknowledge.com/p/history-lesson-the-1980s-semiconductor.

[116] K. William W. and P. Louis W., "Crisis and adaptation in east asian innovation systems: The case of the semiconductor industry in taiwan and south korea," *Business & Politics*, vol. 2, no. 3, pp. 327–352, 2000.

[117] L. Alberto, "What is a fabless chip company?," last accessed on Mar 29, 2023. Available at: https://miscircuitos.com/fabless/.

[118] S. Stephen, "Moore's law: The rule that really matters in tech," last accessed on Mar 22, 2023. Available at: https://www.cnet.com/science/moores-law-the-rule-that-really-matters-in-tech/.

[119] F. Nate, "Who's going to pay for american-made semiconductors?," last accessed on Mar 20, 2023. Available at: https://builtin.com/hardware/american-made-semiconductor-costs.

[120] C. A. Johan, G. Dieter, H. Guido, H. Denis, and H. Arndt, "How does the semiconductor industry landscape look today?," last accessed on Mar 20, 2023. Available at: https://www.kearney.com/industry/technology/article/-/insights/how-does-the-semiconductor-industry-landscape-look-today.

[121] A. Majeed, "The truth about smic's 7-nm chip fabrication ordeal," last accessed on Mar 26, 2023. Available at: https://www.edn.com/the-truth-about-smics-7-nm-chip-fabrication-ordeal/.

[122] IC Insights, "U.S. chip suppliers continue to dominate R&D spending," last accessed on Mar 27, 2023. Available at: https://www.eetasia.com/u-s-chip-suppliers-continue-to-dominate-rd-spending/.

[123] D. Paula, "Process complexity means exponentially increasing data volumes and analysis challenges," last accessed on May 12, 2023. Available at: http://bit.ly/3t6IBwY.

[124] D. Shannon, "Shortage to surplus cycle hits semi but one segment escapes," last accessed on May 18, 2023. Available at: https://www.semiconductor-digest.com/shortage-to-surplus-cycle-hits-semi-but-one-segment-escapes/.

[125] S. Ed, "Design rule complexity rising," last accessed on May 30, 2023. Available at: https://semiengineering.com/design-rule-complexity-rising/.

[126] S. Steffen, "Using AI to pattern sub-10nm ICs," last accessed on Sep 1, 2023. Available at: https://www.ednasia.com/using-ai-to-pattern-sub-10nm-ics/.

[127] Y. Badr, A. Torres, and P. Gupta, "Mask assignment and DSA grouping for DSA-MP hybrid lithography for sub-7 nm contact/via holes," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 36, no. 6, pp. 913–926, 2017.

[128] WikiChip, "5 nm lithography process," last accessed on Sep 1, 2023. Available at: https://en.wikichip.org/wiki/5_nm_lithography_process.

[129] Wiki Wand, "Semiconductor manufacturing processes with a 3 nm GAAFET/FinFET technology node," last accessed on May 03, 2023. Available at: https://www.wikiwand.com/en/3_nm_process.

[130] F. Kaitlyn, "History of the FPGA," last accessed on Apr 20, 2022. Available at: https://digilent.com/blog/history-of-the-fpga/.

[131] S. Trimberger, "FPGA technology: Past, present and future," in *ESSCIRC '95: Twenty-first European Solid-State Circuits Conference*, pp. 12–15, 1995.

[132] Xilinx, Inc., "AMD acquires xilinx," last accessed on Mar 29, 2023. Available at: https://www.amd.com/en/corporate/xilinx-acquisition.

[133] Intel Corp., "Intel acquisition of altera," last accessed on Apr 15, 2023. Available at: https://newsroom.intel.com/press-kits/intel-acquisition-of-altera/.

[134] C. Huriaux, O. Sentieys, and R. Tessier, "Effects of i/o routing through column interfaces in embedded FPGA fabrics," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*, pp. 1–9, 2016.

[135] Xilinx Inc., "Virtex-5 family overview," last accessed on Jan 09, 2023. Available at: https://docs.xilinx.com/v/u/en-US/ds100.

[136] Xilinx Inc., "Virtex-6 family overview," last accessed on Jan 09, 2023. Available at: https://docs.xilinx.com/v/u/en-US/ds150.

[137] Intel Inc., "Intel Arria 10 device overview," last accessed on Jan 09, 2023. Available at: https://www.intel.com/content/www/us/en/docs/programmable/683332/current/device-overview.html.

[138] Invent Logic, "FPGA architecture," last accessed on Mar 30, 2023. Available at: https://allaboutfpga.com/fpga-architecture/.

[139] D. Koch, J. Torresen, C. Beckhoff, D. Ziener, C. Dennl, V. Breuer, J. Teich, M. Feilen, and W. Stechele, "Partial reconfiguration on FPGAs in practice — tools and applications," in *ARCS 2012*, pp. 1–12, 2012.

[140] W. Lie and W. Feng-yan, "Dynamic partial reconfiguration in FPGAs," in *2009 Third International Symposium on Intelligent Information Technology Application*, vol. 2, pp. 445–448, 2009.

[141] Xilinx Inc., "Zynq ultrascale+ mpsoc data sheet," last accessed on May 22, 2022. Available at: https://www.xilinx.com/content/dam/xilinx/support/documents/data_sheets/ds891-zynq-ultrascale-plus-overview.pdf.

[142] Altera Corp., "Stratix V device handbook.," last accessed on May 20, 2023. Available at: https://www.intel.com/programmable/technical-pdfs/683665.pdf.

[143] AMD Xilinx, "Alveo U50 data center accelerator card," last accessed on May 12, 2023. Available at: https://www.xilinx.com/products/boards-and-kits/alveo/u50.html.

[144] Electronic Design News (EDN), "Hybrid architecture embeds xilinx FPGA core into IBM ASICs," last accessed on Apr 25, 2023. Available at: https://www.edn.com/hybrid-architecture-embeds-xilinx-fpga-core-into-ibm-asics/?utm_source=eetimes&utm_medium=relatedcontent.

[145] H. Singh, M.-H. Lee, G. Lu, F. J. Kurdahi, N. Bagherzadeh, and E. M. Chaves Filho, "Morphosys: an integrated reconfigurable system for data-parallel and computation-intensive applications," *IEEE transactions on computers*, vol. 49, no. 5, pp. 465–481, 2000.

[146] H. Zhang, V. Prabhu, V. George, M. Wan, M. Benes, A. Abnous, and J. M. Rabaey, "A 1 v heterogeneous reconfigurable processor IC for baseband wireless applications," in *2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No. 00CH37056)*, pp. 68–69, IEEE, 2000.

[147] J. Becker and M. Glesner, "A parallel dynamically reconfigurable architecture designed for flexible application-tailored hardware/software systems in future mobile communication," *The Journal of Supercomputing*, vol. 19, no. 1, pp. 105–127, 2001.

[148] M. Borgatti, F. Lertora, B. Forêt, and L. Calí, "A reconfigurable system featuring dynamically extensible embedded microprocessor, FPGA, and customizable i/o," *IEEE Journal of Solid-State Circuits*, vol. 38, no. 3, pp. 521–529, 2003.

[149] AMD Xilinx Inc., "AMD xilinx adaptive SoCs," last accessed on Feb 20, 2023. Available at: https://www.xilinx.com/products/silicon-devices/soc.html.

[150] I. Swarbrick, D. Gaitonde, S. Ahmad, B. Gaide, and Y. Arbel, "Network-on-chip programmable platform in versal ACAP architecture," in *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, FPGA '19, (New York, NY, USA), p. 212–221, Association for Computing Machinery, 2019.

[151] Menta, "Embedded FPGA IP," last accessed on Apr 2, 2022. Available at: https://www.menta-efpga.com/.

[152] Business Wire, "Intel to acquire altera," last accessed on Mar 19, 2023. Available at: https://www.businesswire.com/news/home/20150601005864/en/Intel-to-Acquire-Altera.

[153] A. Brataas and K. M. D. Hals, "Spin–orbit torques in action," *Nature Nanotechnology*, vol. 9, pp. 86–88, Feb 2014.

[154] Renesas Electronics, "Stream transpose processor," last accessed on Mar 19, 2023. Available at: `https://www.renesas.com/us/en/products/power-management/pmic/stp-engine.html`.

[155] C. Kim, M. Chung, Y. Cho, M. Konijnenburg, S. Ryu, and J. Kim, "ULP-SRP: Ultra low-power samsung reconfigurable processor for biomedical applications," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 7, no. 3, pp. 1–15, 2014.

[156] G. Kolhe, T. Sheaves, K. I. Gubbi, S. Salehi, S. Rafatirad, S. M. PD, A. Sasan, and H. Homayoun, "Lock&roll: deep-learning power side-channel attack mitigation using emerging reconfigurable devices and logic locking," in *Proceedings of the 59th ACM/IEEE Design Automation Conference*, pp. 85–90, 2022.

[157] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately deobfuscating integrated circuits," in *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 95–100, 2017.

[158] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "SMT attack: Next generation attack on obfuscated circuits with capabilities and performance beyond the SAT attacks," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 97–122, 2019.

[159] P. Chakraborty, J. Cruz, A. Alaql, and S. Bhunia, "SAIL: Analyzing structural artifacts of logic locking using machine learning," *IEEE Transactions on Information Forensics and Security*, pp. 1–1, 2021.

[160] A. Alaql, D. Forte, and S. Bhunia, "Sweep to the secret: A constant propagation attack on logic locking," in *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*, pp. 1–6, 2019.

[161] A. Alaql, M. M. Rahman, and S. Bhunia, "SCOPE: Synthesis-based constant propagation attack on logic locking," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 8, pp. 1529–1542, 2021.

[162] P. Chowdhury, C. Sathe, and B. Carrion Schaefer, "Predictive model attack for embedded FPGA logic locking," in *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 1–6, 2022.

[163] A. Rezaei, R. Afsharmazayejani, and J. Maynard, "Evaluating the security of eFPGA-based redaction algorithms," ICCAD '22, (New York, NY, USA), Association for Computing Machinery, 2022.

[164] Z. Han, M. Shayan, A. Dixit, M. Shihab, Y. Makris, and J. Rajendran, "FuncTeller: How well does eFPGA hide functionality?," *arXiv preprint, arXiv:2306.05532*, 2023.

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

[166] L. Collini, B. Tan, C. Pilato, and R. Karri, "Reconfigurable logic for hardware IP protection: Opportunities and challenges," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–7, 2022.

[167] J. Bhandari, A. K. T. Moosa, B. Tan, C. Pilato, G. Gore, X. Tang, S. Temple, P.-E. Gaillardo, and R. Karri, "Not all fabrics are created equal: Exploring eFPGA parameters for IP redaction," *arXiv preprint, arXiv:2111.04222*, 2021.

[168] K. Ramesh, B. Tan, L. Collini, and T. M. A. Khader, "CSAW'21 logic locking," last accessed on Jul 27, 2023. Available at: `https://www.csaw.io/logic-locking`.

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

[170] O. Glamočanin, D. G. Mahmoud, F. Regazzoni, and M. Stojilović, "Shared FPGAs and the holy grail: Protections against side-channel and fault attacks," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 1645–1650, 2021.

[171] J. Szefer, "Survey of microarchitectural side and covert channels, attacks, and defenses," *Journal of Hardware and Systems Security*, vol. 3, pp. 219–234, Sep 2019.

[172] K. Tiri and I. Verbauwhede, "A logic level design methodology for a secure DPA resistant ASIC or FPGA implementation," in *Proceedings Design, Automation and Test in Europe Conference and Exhibition*, vol. 1, pp. 246–251 Vol.1, 2004.

[173] S. S. Ali, S. M. Saeed, O. Sinanoglu, and R. Karri, "Novel test-mode-only scan attack and countermeasure for compression-based scan architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 5, pp. 808–821, 2015.

[174] L. Alrahis, M. Yasin, N. Limaye, H. Saleh, B. Mohammad, M. Al-Qutayri, and O. Sinanoglu, "ScanSAT: Unlocking static and dynamic scan obfuscation," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 4, pp. 1867–1882, 2021.

[175] J. DaRolt, G. Di Natale, M.-L. Flottes, and B. Rouzeyre, "Scan attacks and countermeasures in presence of scan response compactors," in *2011 Sixteenth IEEE European Test Symposium*, pp. 19–24, 2011.

[176] N. Limaye and O. Sinanoglu, "DynUnlock: Unlocking scan chains obfuscated using dynamic keys," DATE '20, (San Jose, CA, USA), p. 270–273, EDA Consortium, 2020.

[177] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, p. 148–160, Association for Computing Machinery, 2002.

[178] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *2007 44th ACM/IEEE Design Automation Conference*, pp. 9–14, 2007.

[179] International Organization for Standardization, "ISO/IEC 20897-1:2020 information security, cybersecurity and privacy protection — physically unclonable functions — part 1: Security requirements," last accessed on Jun 30, 2020. Available at: `https://www.iso.org/standard/76353.html`.

[180] A. Shamsoshoara, A. Korenda, F. Afghah, and S. Zeadally, "A survey on physical unclonable function (PUF)-based security solutions for internet of things," *Computer Networks*, vol. 183, p. 107593, 2020.

[181] G. Srinivasan, P. Wijesinghe, S. S. Sarwar, A. Jaiswal, and K. Roy, "Significance driven hybrid 8T-6T SRAM for energy-efficient synaptic storage in artificial neural networks," in *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pp. 151–156, 2016.

[182] Intrinsic ID, "SRAM PUF technology," last accessed on Jul 01, 2022. Available at: `https://www.intrinsic-id.com/sram-puf/`.

[183] M. T. Rahman, A. Hosey, Z. Guo, J. Carroll, D. Forte, and M. Tehranipoor, "Systematic correlation and cell neighborhood analysis of SRAM PUF for robust and unique key generation," *Journal of Hardware and Systems Security*, vol. 1, pp. 137–155, Jun 2017.

[184] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198–1210, 2009.

[185] M. Cortez, S. Hamdioui, V. van der Leest, R. Maes, and G.-J. Schrijen, "Adapting voltage ramp-up time for temperature noise reduction on memory-based PUFs," in *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pp. 35–40, 2013.

[186] S. Elgendy and E. Y. Tawfik, "Impact of physical design on PUF behavior: A statistical study," in *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2021.

[187] National Institute of Standards and Technology (NIST), "Recommendation for the entropy sources used for random bit generation," last accessed on Oct 20, 2018. Available at: `https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf`.

[188] K. E. Murray, O. Petelin, S. Zhong, J. M. Wang, M. Eldafrawy, J.-P. Legault, E. Sha, A. G. Graham, J. Wu, M. J. P. Walker, H. Zeng, P. Patros, J. Luu, K. B. Kent, and V. Betz, "VTR 8: High-performance cad and customizable FPGA architecture modelling," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 13, no. 2, 2020.

[189] M. G. A. Martins, R. P. Ribas, and A. I. Reis, "Functional composition: A new paradigm for performing logic synthesis," in *Thirteenth International Symposium on Quality Electronic Design (ISQED)*, pp. 236–242, 2012.

[190] M. G. A. Martins, L. Rosa, A. B. Rasmussen, R. P. Ribas, and A. I. Reis, "Boolean factoring with multi-objective goals," in *Computer Design (ICCD), 2010 IEEE International Conference on*, pp. 229–234, IEEE, 2010.

[191] M. Imran, Z. U. Abideen, and S. Pagliarini, "An open-source library of large integer polynomial multipliers," in *2021 24th International Symposium on Design and Diagnostics of Electronic Circuits Systems (DDECS)*, pp. 145–150, 2021.

[192] FreeCores, "Infinite impulse response (IIR) filter," last accessed on Dec 25, 2021. Available at: `https://github.com/freecores/all-pole_filters`.

[193] T. Zhu, "PID (proportional integral derivative) controller," last accessed on Dec 26, 2022. Available at: `https://opencores.org/projects/pid_controller`.

[194] J. Carlos, "FPGA-based median filter," last accessed on Feb 19, 2023. Available at: `https://opencores.org/projects/fpu100`.

[195] S. Joachim, "SHA-256," last accessed on Jan 20, 2023. Available at: `https://github.com/secworks/sha256`.

[196] J. Al-Eryani, "Floating-point unit (FPU) controller," last accessed on Feb 15, 2023. Available at: `https://opencores.org/projects/fpu100`.

[197] O. Kindgren and M. John, "OpenRISC 1200 implementation," last accessed on Feb 21, 2023. Available at: `https://github.com/openrisc/or1200`.

[198] H. Hsing, "AES-128," last accessed on Jan 22, 2023. Available at: `https://opencores.org/projects/tiny_aes`.

[199] P. Mohan, O. Atli, O. Kibar, M. Zackriya, L. Pileggi, and K. Mai, "Top-down physical design of soft embedded FPGA fabrics," in *The 2021 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, p. 1–10, 2021.

[200] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp. 137–143, 2015.

[201] 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*, p. 1601–1618, 2017.

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

[203] E. Nudelman, K. Leyton-Brown, H. H. Hoos, A. Devkar, and Y. Shoham, "Understanding random sat: Beyond the clauses-to-variables ratio," in *Principles and Practice of Constraint Programming – CP 2004* (M. Wallace, ed.), (Berlin, Heidelberg), pp. 438–452, Springer Berlin Heidelberg, 2004.

[204] A. Taneem, D. K. Paul, and H. A. Jason, "Packing techniques for virtex-5 FPGAs," last accessed on Sep 25, 2023. Available at: `https://janders.eecg.utoronto.ca/pdfs/trets_taneem.pdf`.

[205] M. Hansen, H. Yalcin, and J. Hayes, "Unveiling the ISCAS-85 benchmarks: a case study in reverse engineering," *IEEE Design & Test of Computers*, vol. 16, no. 3, pp. 72–80, 1999.

[206] OpenCores, "DES cryptocore," last accessed on Sep 10, 2023. Available at: https://opencores.org/projects/basicdes.

[207] OpenCores, "Basic RSA encryption engine," last accessed on Sep 10, 2023. Available at: https://opencores.org/projects/basicrsa.

[208] OpenCores, "OpenGFX430," last accessed on Sep 10, 2023. Available at: https://opencores.org/projects/opengfx430.

[209] OpenCores, "Classic 5-stage pipeline MIPS," last accessed on Sep 10, 2023. Available at: https://opencores.org/projects/mips32.

[210] OpenCores, "JPEG decoder," last accessed on Sep 10, 2023. Available at: https://opencores.org/projects/jpeg_core.

[211] OpenCores, "USB host core," last accessed on Sep 11, 2023. Available at: https://opencores.org/projects/usb_host_core.

[212] OpenCores, "CORDIC core," last accessed on Sep 11, 2023. Available at: https://opencores.org/projects/cordic.

[213] OpenCores, "Simple all digital FM receiver," last accessed on Sep 11, 2023. Available at: https://opencores.org/projects/all_digital_fm_receiver.

[214] OpenCores, "2nd order sigma-delta DAC," last accessed on Sep 11, 2023. Available at: https://opencores.org/projects/sigma_delta_dac_dual_loop.

[215] OpenCores, "OpenMSP430," last accessed on Sep 11, 2023. Available at: https://opencores.org/projects/openmsp430.

[216] H. Hsing, "SHAKE-256," last accessed on Mar 22, 2023. Available at: https://opencores.org/projects/sha3.

[217] U. Embedded, "BiRiscV - 32-bit dual issue RISC-V CPU," last accessed on Feb 01, 2022. Available at: https://opencores.org/projects/biriscv.

[218] Q. Audrey, "A simple guide to AES 256-bit encryption," last accessed on Sep 16, 2023. Available at: https://www.azeusconvene.com/articles/a-simple-guide-to-aes-256-bit-encryption.

[219] J.-P. Linnartz and P. Tuyls, "New shielding functions to enhance privacy and prevent misuse of biometric templates," in *Audio- and Video-Based Biometric Person Authentication* (J. Kittler and M. S. Nixon, eds.), (Berlin, Heidelberg), pp. 393–402, Springer Berlin Heidelberg, 2003.

[220] Z. U. Abideen, R. Wang, T. D. Perez, G.-J. Schrijen, and S. Pagliarini, "Impact of orientation on the bias of SRAM-based pufs," *IEEE Design & Test*, pp. 1–1, 2023.

[221] B. Cheng, S. Roy, and A. Asenov, "The impact of random doping effects on CMOS SRAM cell," in *Proceedings of the 30th ESSCIRC*, pp. 219–222, 2004.

[222] J. Delvaux, D. Gu, D. Schellekens, and I. Verbauwhede, "Helper data algorithms for PUF-based key generation: Overview and analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 889–902, 2015.

[223] R. Maes, P. Tuyls, and I. Verbauwhede, "A soft decision helper data algorithm for SRAM PUFs," in *2009 IEEE International Symposium on Information Theory*, pp. 2101–2105, 2009.

[224] Z. U. Abideen, S. Gokulanathan, M. J. Alijafar, and S. Pagliarini, "An overview of FPGA-inspired obfuscation techniques," *arXiv preprint, arXiv:2305.15999*, 2023.