TAL TECH

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

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.

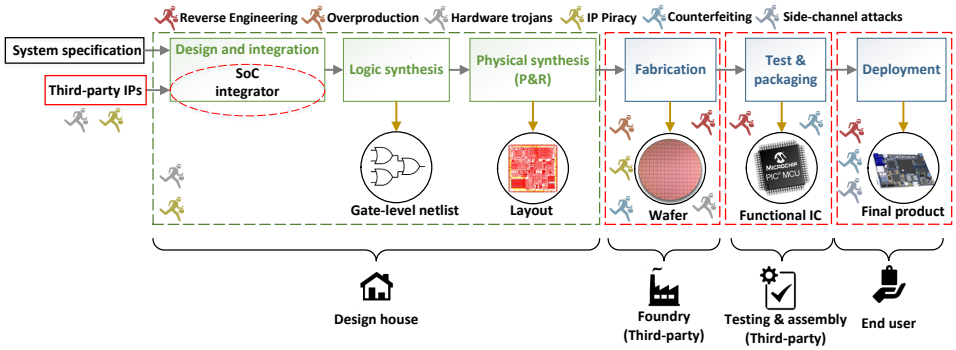


Figure 1: Typical stages involved in the globalized IC supply chain: untrusted stages are highlighted in red.

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.

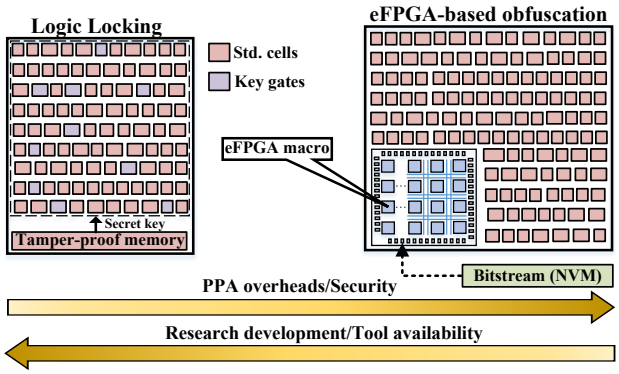


Figure 2: Comparison Diagram: LL vs. eFPGA-based obfuscation.

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.

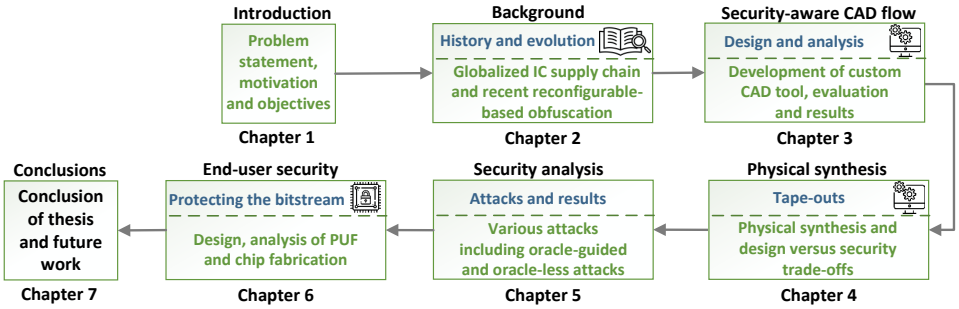


Figure 3: Organization of the thesis.

- **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.

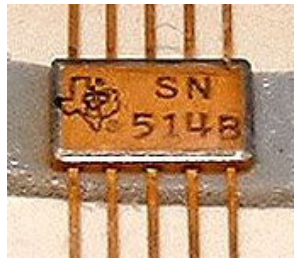


Figure 4: The SN514 IC released by Texas Instruments [112].

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].

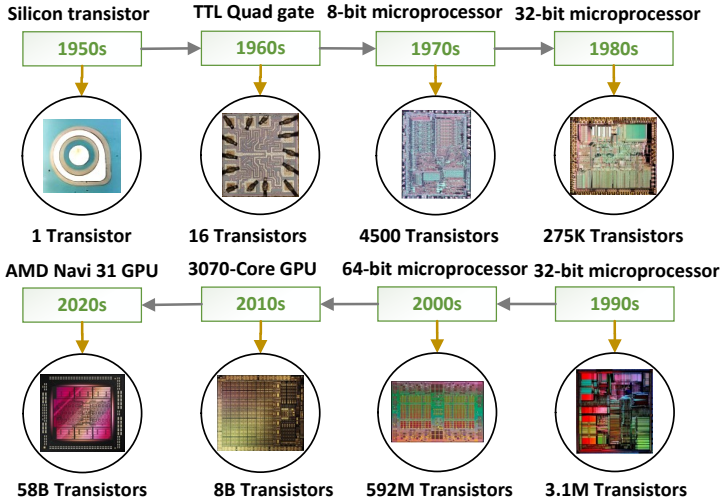


Figure 5: The timeline of the semiconductors in computers [114].

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.

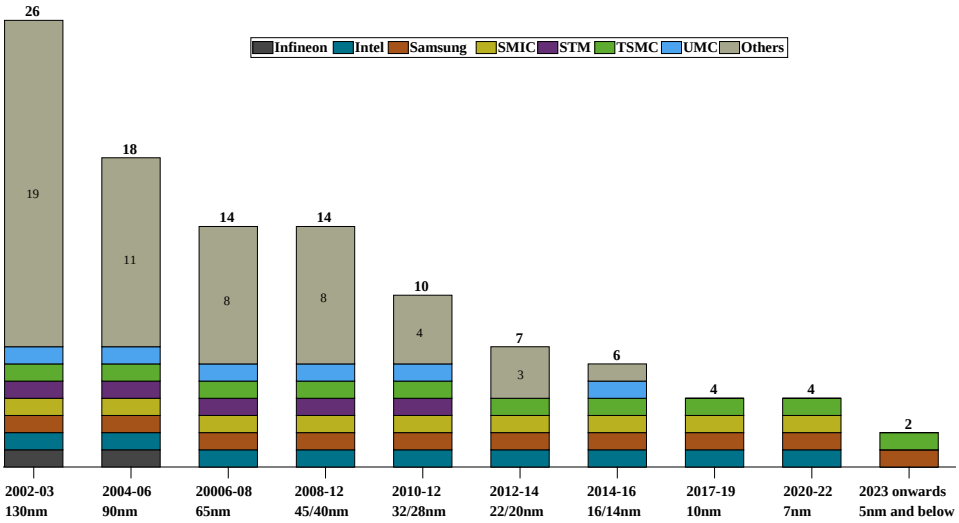


Figure 6: The semiconductor industry evolution up to 10nm [120].

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.

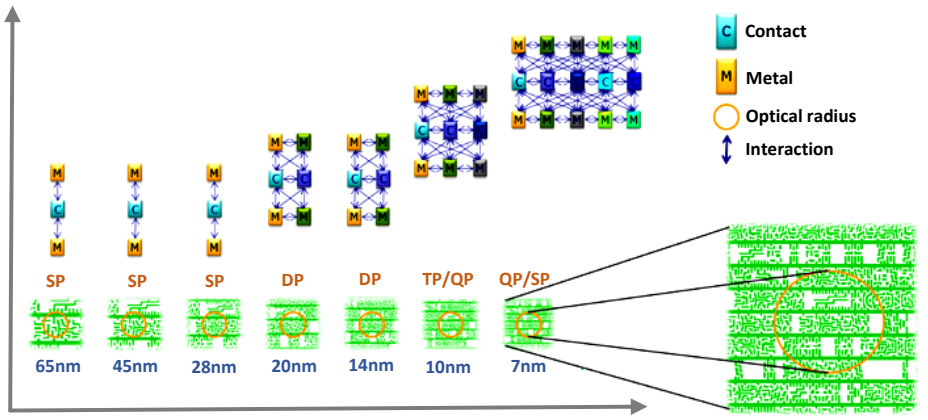


Figure 7: The complexity of device fabrication on the advanced technology nodes [126, 127].

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

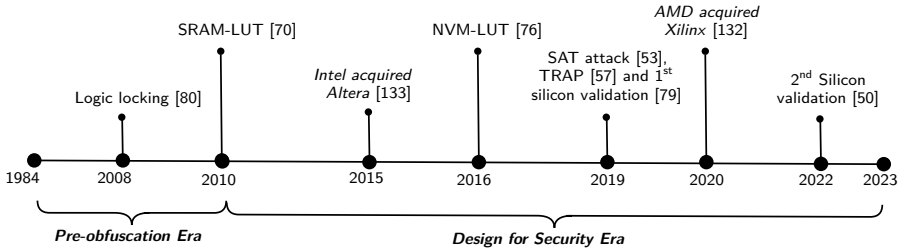


Figure 8: The transition from the pre-Obfuscation era to the security era.

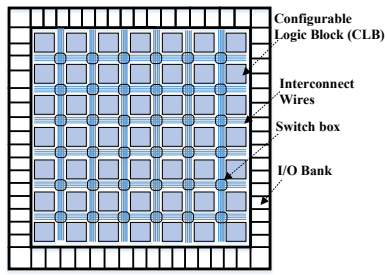


Figure 9: The conventional island-style architecture of FPGAs, adapted from [138].

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 $Y = 1$.

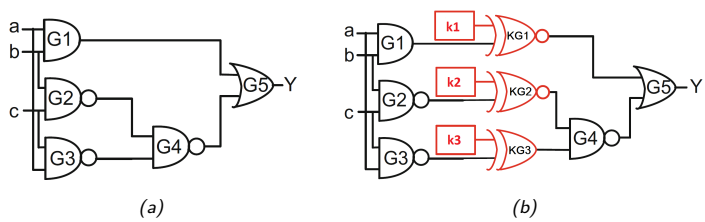


Figure 10: LL using XOR/XNOR gates [46].

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.

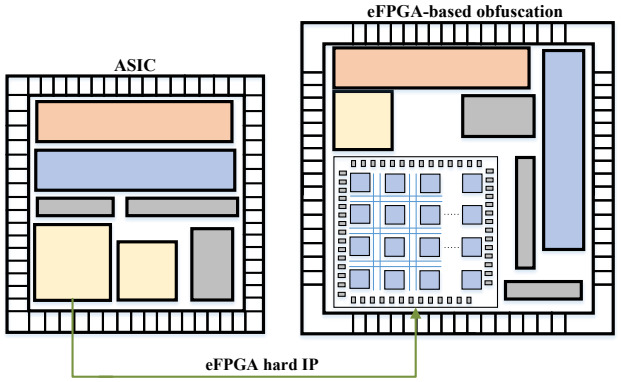


Figure 11: Understanding the eFPGA-based obfuscation technique [83].

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.

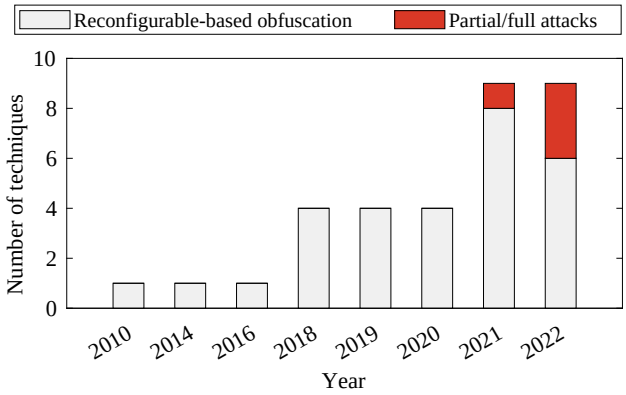


Figure 12: Publications trend for the techniques and attacks on reconfigurable-based obfuscation.

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	min: 0.7	min: 0.0
			avg: 6.4	avg: 24.9	avg: 28.4
			max: 20.6	max: 82.1	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		min: 0.0
			avg: 12.2	–	avg: 20.4
			max: 25.4		max: 41.8
	SOT-LUT-32i_G [77]	ISCAS/MCNC	min: 6.7		min: 0.0
			avg: 17.7	–	avg: 28.5
			max: 27.2		max: 47.5
	SOT-LUT-64i_G [77]	ISCAS/MCNC	min: 15.2		min: 4.2
			avg: 22.2	–	avg: 36.1
			max: 27.2		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.

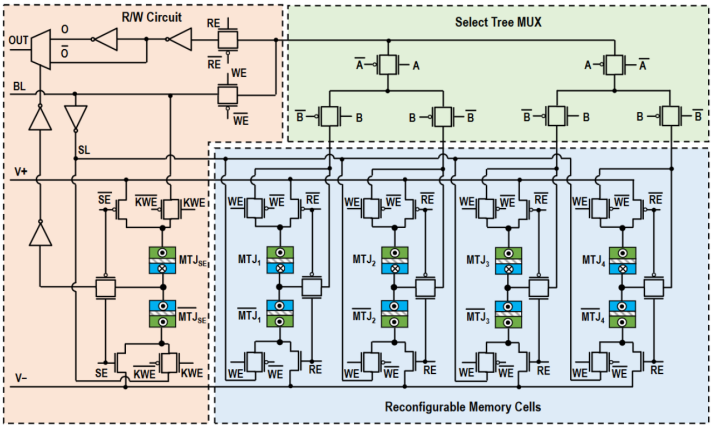


Figure 13: 2-input MRAM-based LUT that utilizes STT and MTJ technology [54].

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.

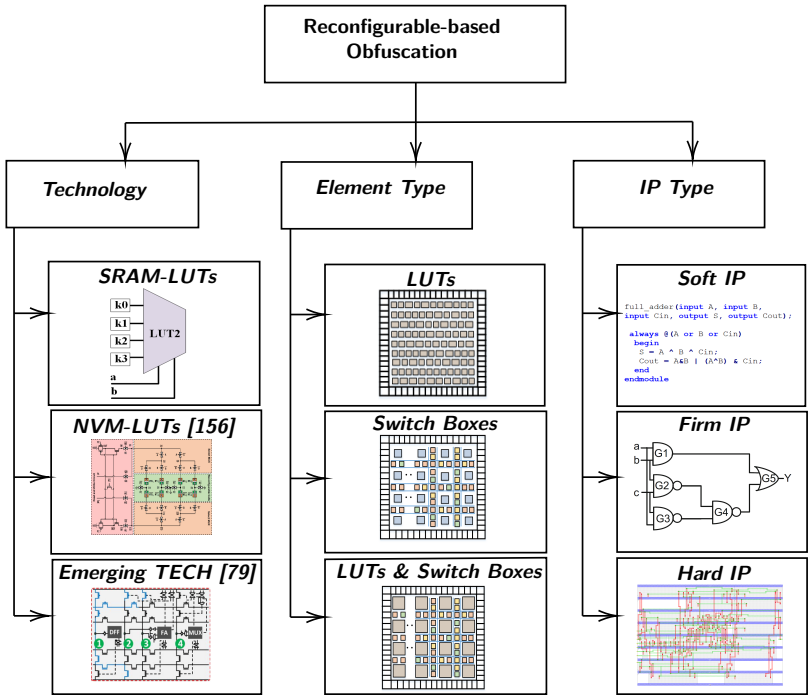


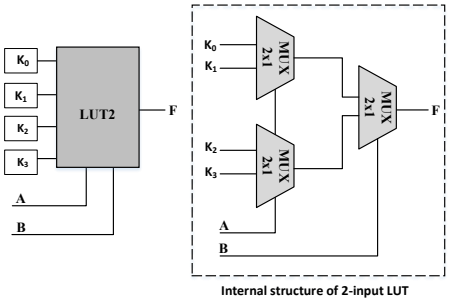
Figure 14: Classification of Reconfigurable-based obfuscation.

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},

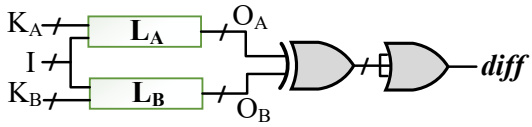


Figure 16: Circuit employed by the SAT attack.

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

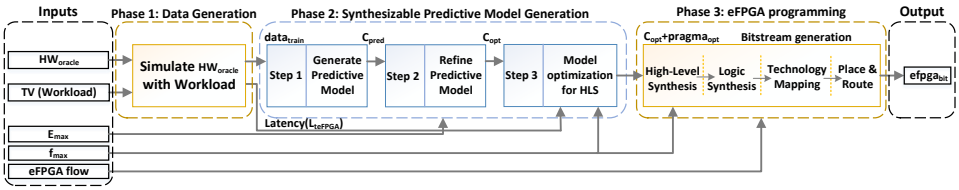


Figure 17: The three primary phases of predictive model attack, adapted from [162].

[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 $g(x,k^*) \equiv f(x)$

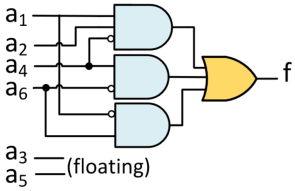
while *(True)* **do**
  $W \leftarrow$ SearchFeedbackSignals$(g(x,k))$
  // $W \leftarrow \{w_0, w_1, \ldots, w_m\}$
  for $w_i \in W$ **do**
    $F(w_i, w_i') \leftarrow$ BreakFeedback(w_i)
    $NCCNF(k) \leftarrow \bigwedge_{i=0}^{m} F(w_i, w_i')$
    // $NCCNF(k) \leftarrow$ `BreakFeedback`$(w_0) \wedge \cdots \wedge$ `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.

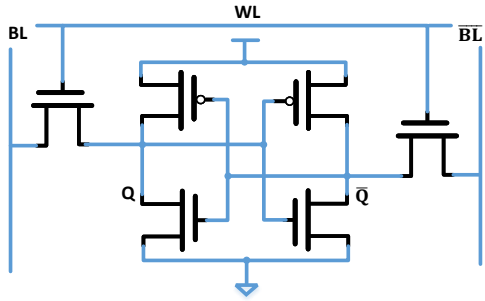


Figure 19: The internal architecture of 6T-SRAM bitcell, adapted from [181].

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.

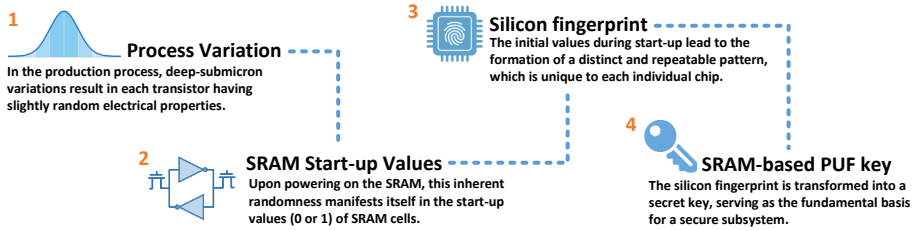


Figure 20: The procedure to harvest unique signature from SRAM-PUF [182].

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

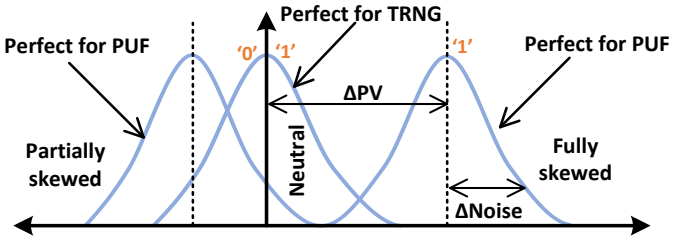


Figure 21: Characteristic of SRAM bitcells on power-up state, adapted from [183].

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.

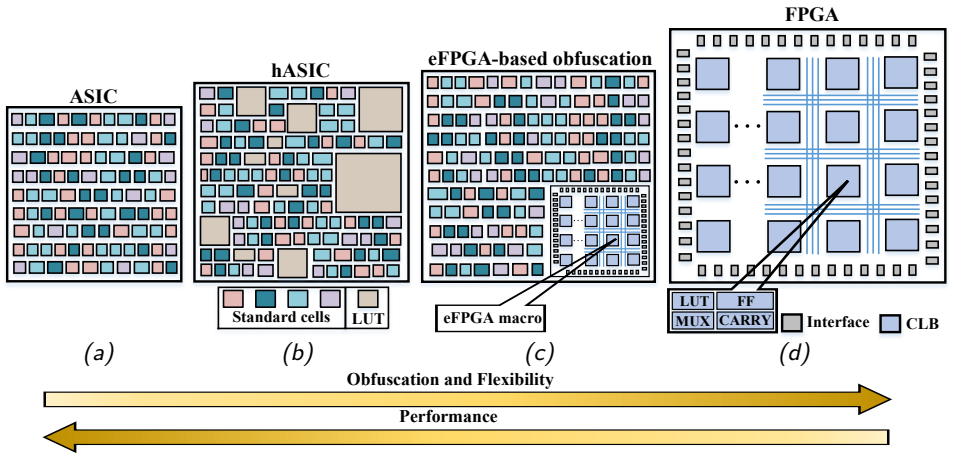


Figure 22: The design obfuscation landscape.

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.

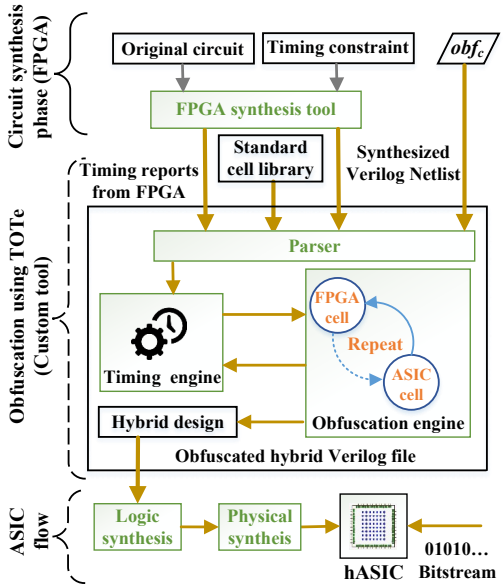


Figure 23: A security-aware CAD flow for hASIC.

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.

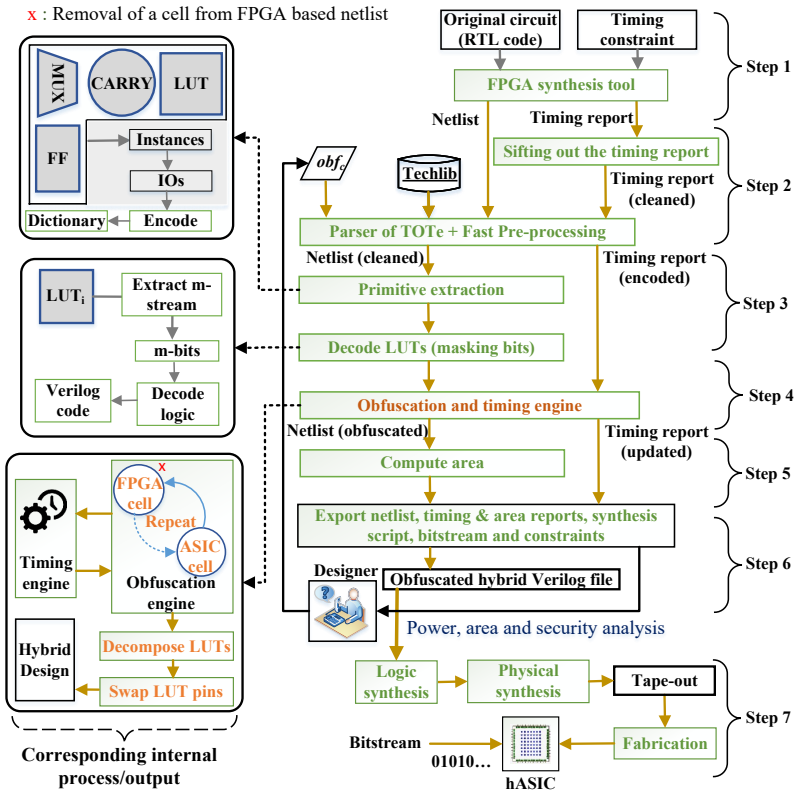


Figure 24: Overview of the obfuscation flow and its inner steps.

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 $L_{ST} \leftarrow \phi$, $L_{RE} \leftarrow L$
2 $K \leftarrow (1 - obf_c) \times SIZE\_OF(L_{RE})$
3 **while** *SIZE_OF* $(L_{ST}) \leq K$ **do**
4 $\quad path \leftarrow$ FIND_CRITICAL(P)
5 $\quad lut \leftarrow$ FIND_SLOWEST($path$)
6 $\quad$ **if** $lut \in L_{RE}$ **then**
7 $\quad\quad INSERT(lut, L_{ST})$
8 $\quad\quad$ REMOVE(lut, L_{RE})
9 $\quad\quad$ UPDATE_INSTANCENAME_TIMING(lut, P)
10 **for** *each* $lut \in L_{ST}$ **do**
11 $\quad Design_{ST} \leftarrow$ DECODE(lut)
12 **for** *each* $lut \in L_{RE}$ **do**
13 $\quad$ GEN_CASE_0_1(lut)
14 $\quad$ DECOMPOSE_OPT(lut)
15 $\quad$ SWAP_PINS(lut)
16 Design_RE $\leftarrow$ GEN_RE(L_RE)
17 **return** $hASIC \leftarrow Design_{ST} \cup Design_{RE}$

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

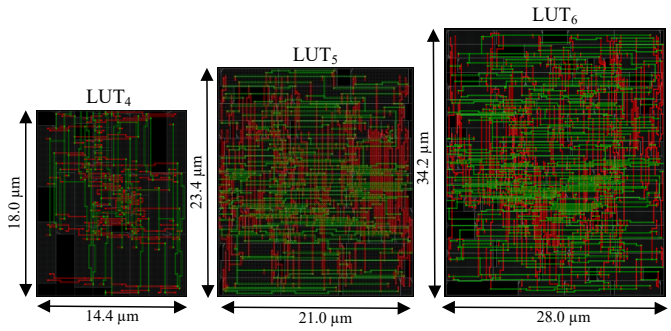


Figure 25: The layout of macros for LUT_4, LUT_5, and LUT_6 [73].

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.

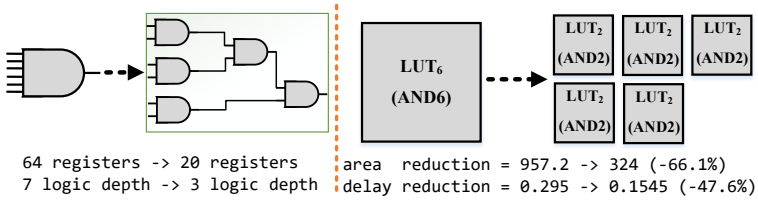


Figure 26: Logic conversion and decomposition of LUT_6 [73].

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 [$n2$, $n0$, $n1$].

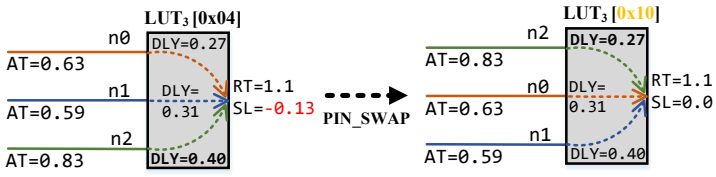


Figure 27: Example of a beneficial pin swap [73].

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.

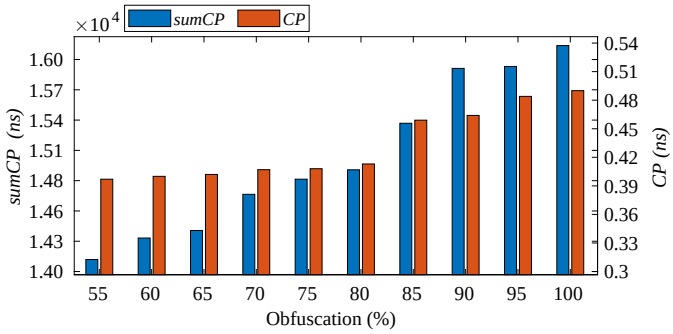


Figure 28: Obfuscation versus performance trade-off for SBM [74].

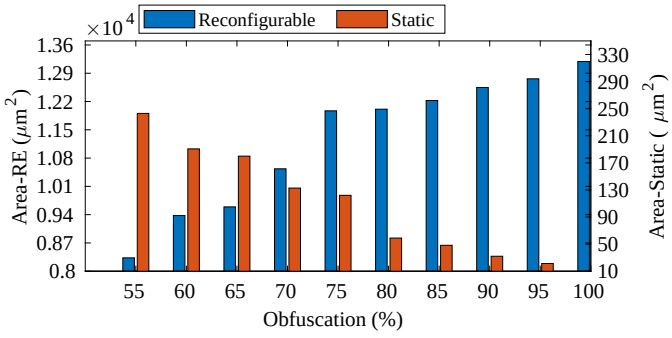


Figure 29: Obfuscation versus area trade-off for SBM [74].

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.

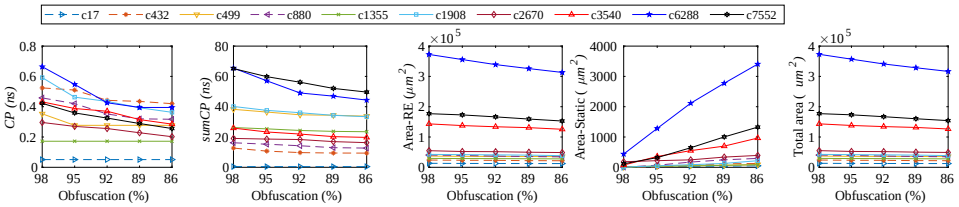


Figure 30: Obfuscation results for ISCAS'85 benchmarks [73].

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.

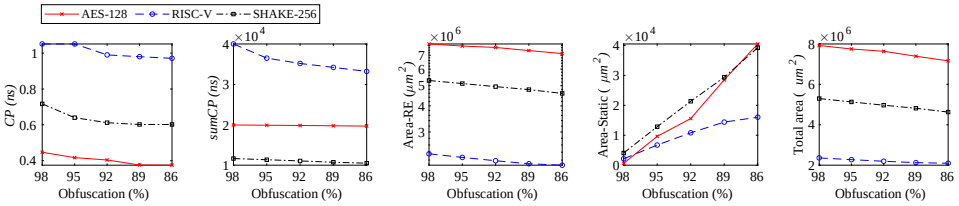


Figure 31: Obfuscation results for AES-128, RISC-V and SHAKE-256 [73].

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.

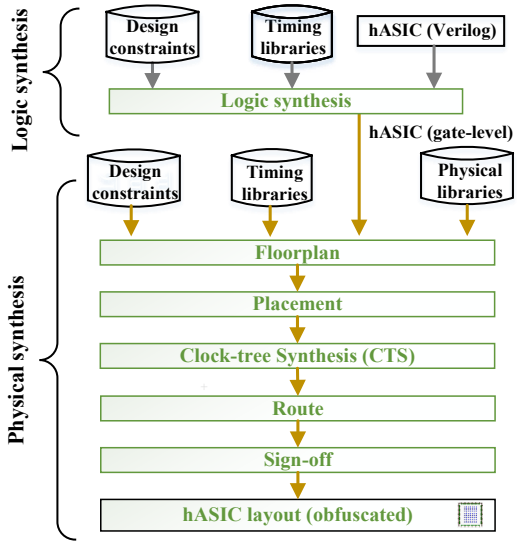


Figure 32: The steps involved in the physical synthesis of hASIC.

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

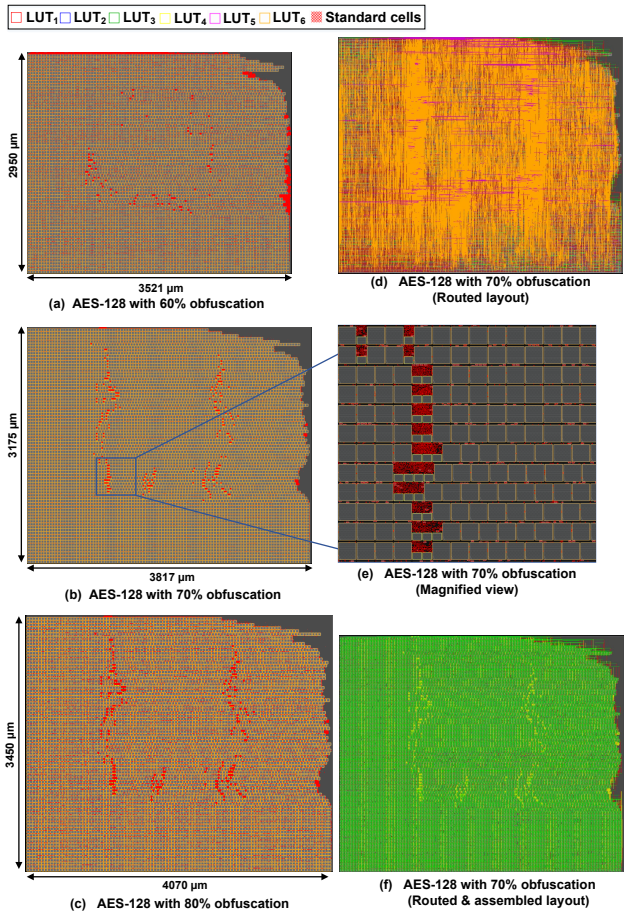


Figure 33: Implementation results for AES-128 with different obfuscation levels.

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

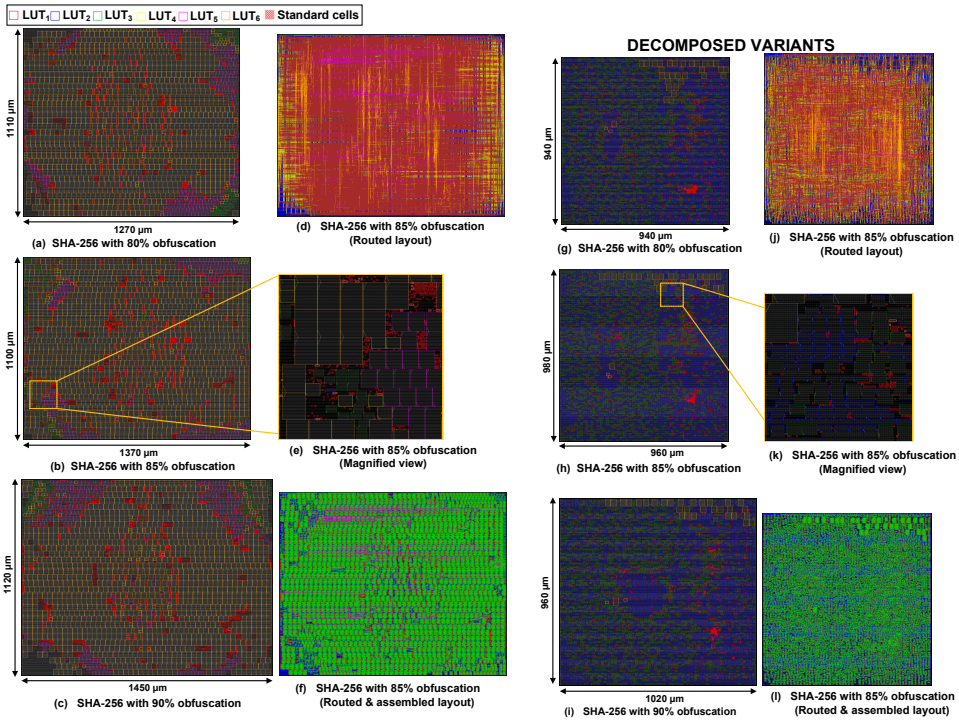


Figure 34: Implementation results for SHA-256 with different obfuscation levels [73].

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

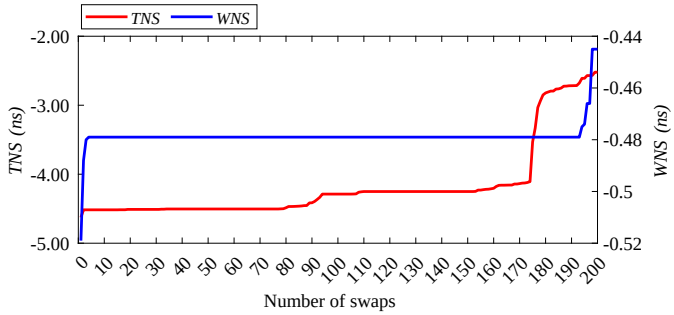


Figure 35: Change in the TNS/WNS concerning the swap of LUT pins [73].

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.

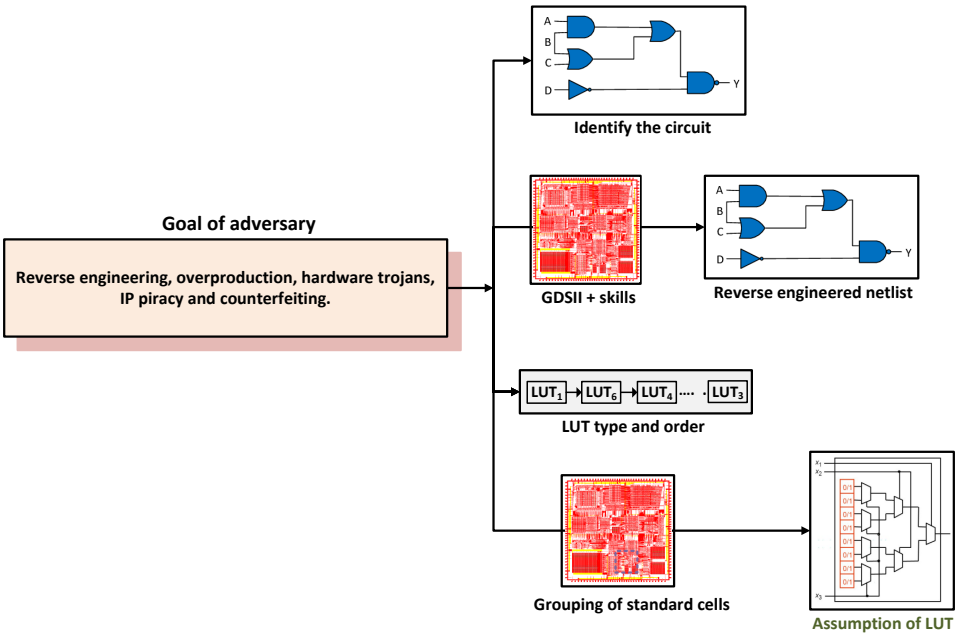


Figure 36: The summary of the threat model for hASIC.

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.

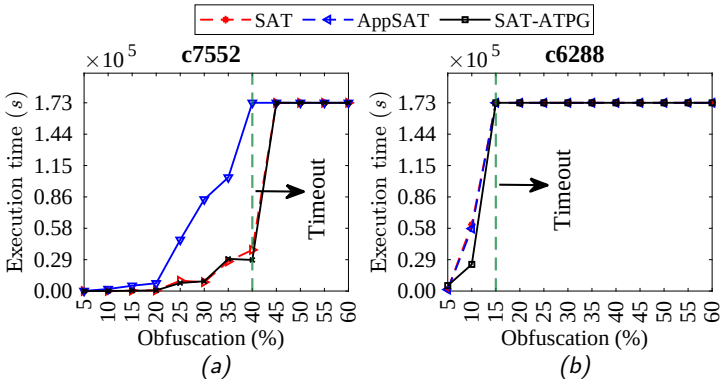


Figure 37: The execution time of SAT attacks.

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.

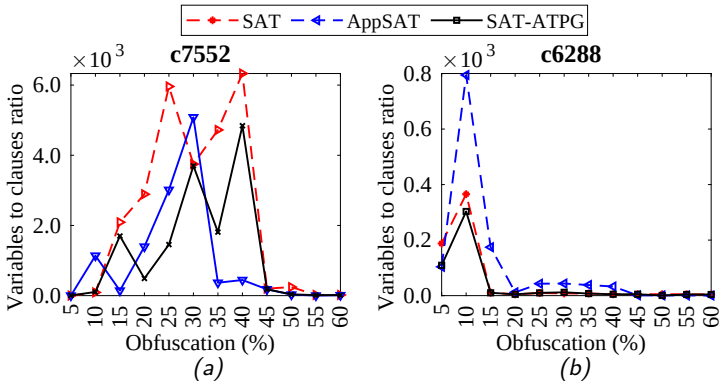


Figure 38: The variables to clauses ratio of SAT attacks for two different designs.

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].

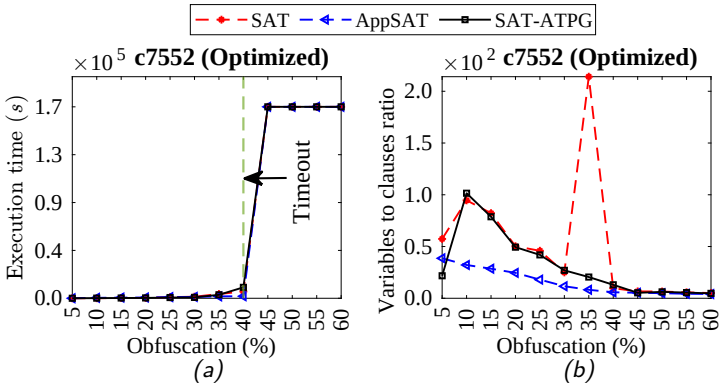


Figure 39: The optimized results for c7552 regarding the execution time and the ratio of variables to clauses in SAT attacks.

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.

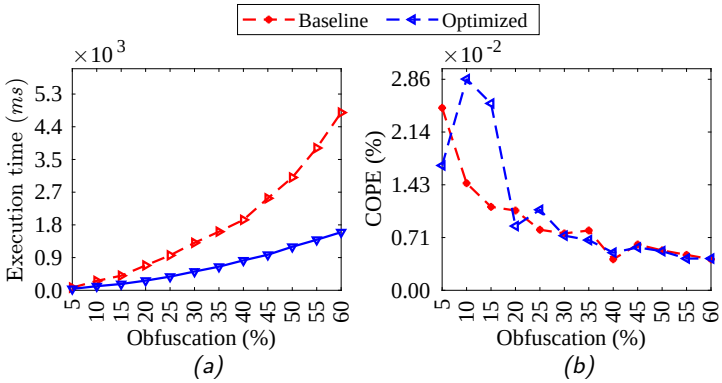


Figure 40: The comparison between the baseline and optimized design for the oracle-less SCOPE attack.

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

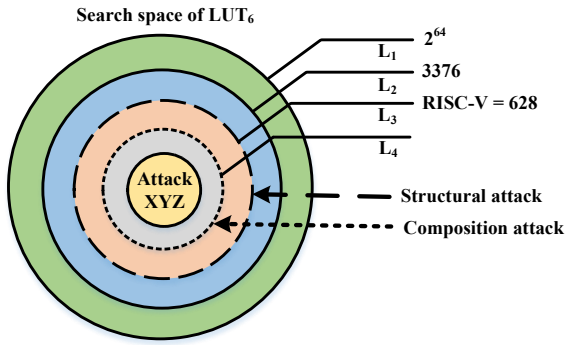


Figure 41: The search space of LUT_6 as it shrinks with different attacks [74]

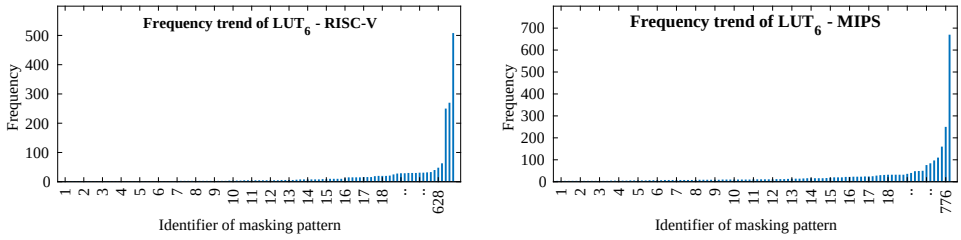
of commercial FPGA synthesis, such as Xilinx [204]. The decomposition method only applies to the reconfigurable part of the design, whereas the initial knowledge obtained by the adversary is from the static part, which remains unchanged regardless of the use of decomposition. The majority of the static 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.

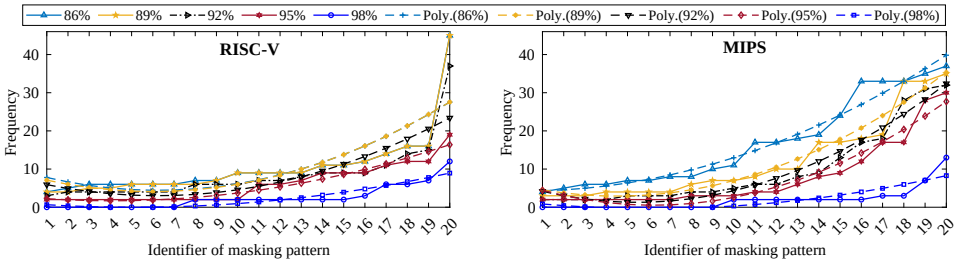


Figure 43: The structural analysis of RISC-V and MIPS [73].

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.

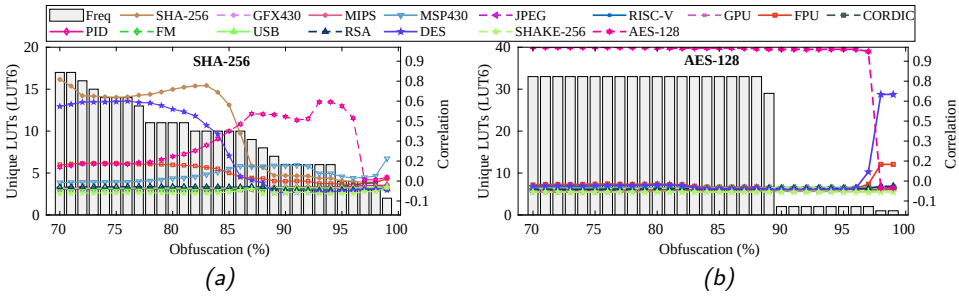


Figure 44: The correlation of SHA-256 and AES-128 versus numerous other designs [73].

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.

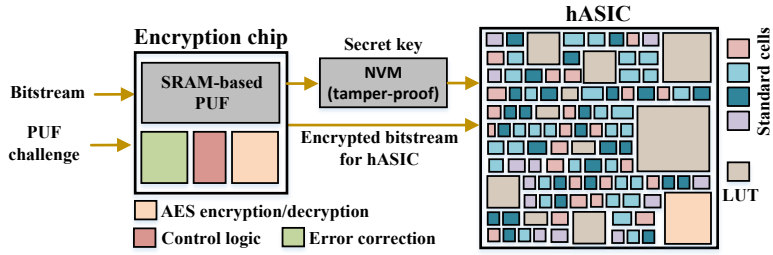


Figure 45: Encryption/decryption scheme of hASIC with SRAM-based PUF.

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.

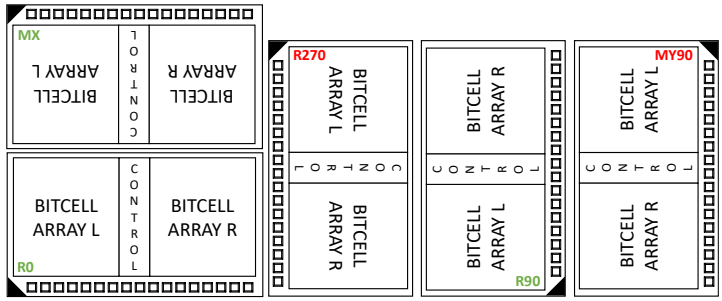


Figure 46: The simplified architecture and orientations of memory [220].

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.

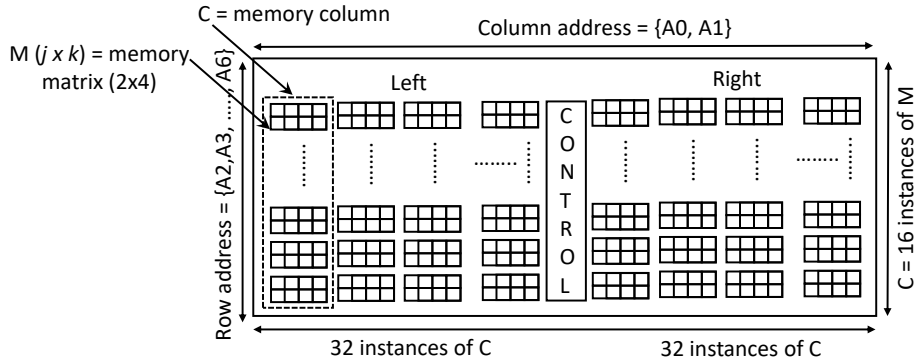


Figure 47: Architecture of low-speed memory with 64-bit datawidth and 128 location depth.

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.

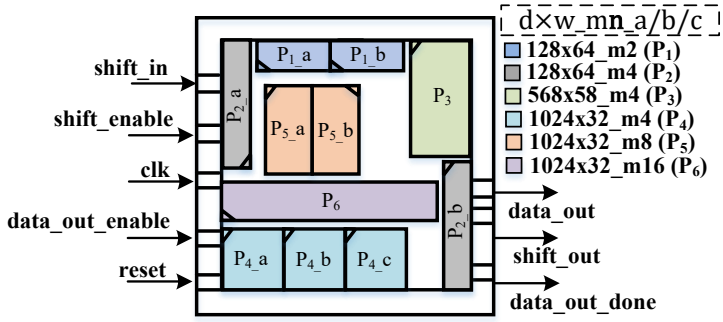
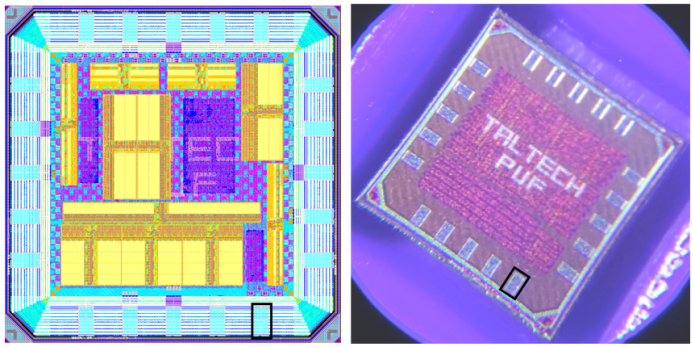


Figure 48: The simplified architecture of the SRAM-based PUFs.

[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.

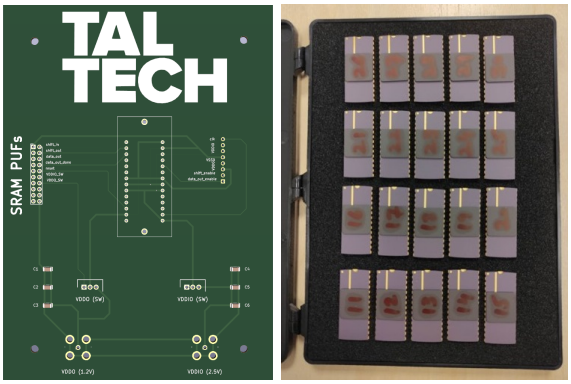


(a) *(b)*

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.

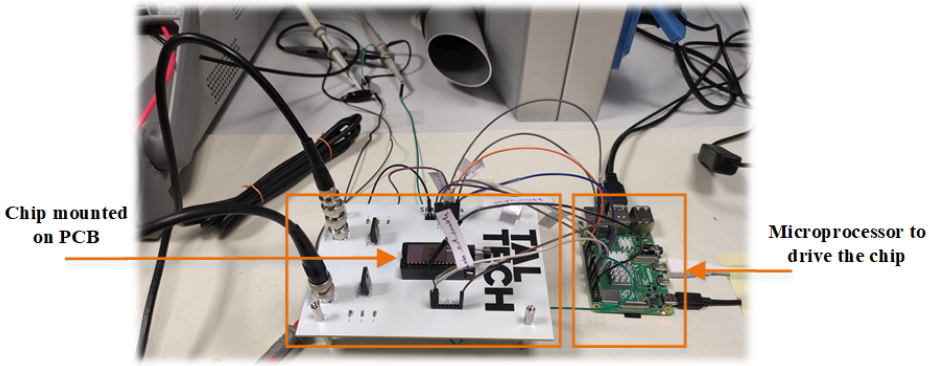


Figure 51: Testing setup for the chip.

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.

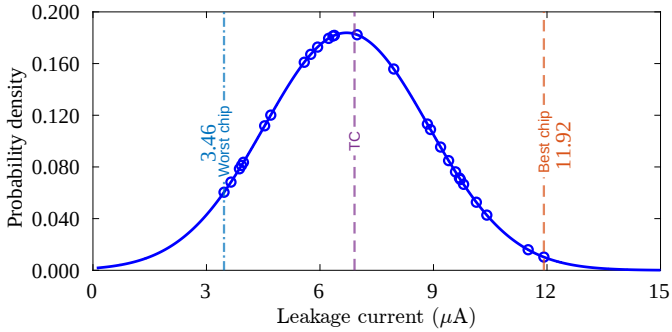


Figure 52: The distribution of leakage current across 50 chips and the typical corner (TC) from physical synthesis.

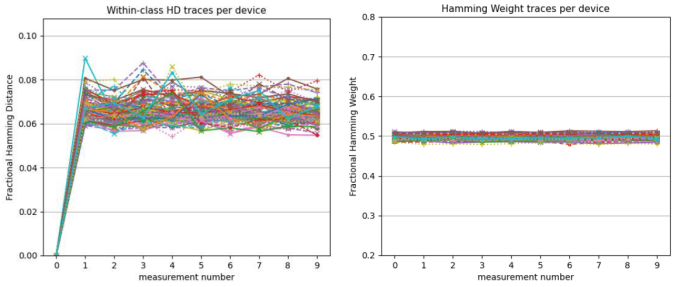
6.4 Results and Observations

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

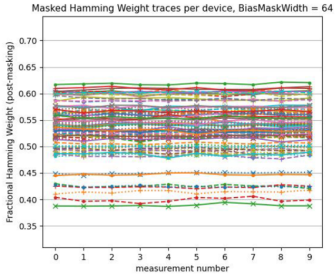
The panels in Figures 53, 54, 55, 56, 57, and 58 illustrate the trends of WCHD, HW, MHW, and BCHD for six distinct SRAM-based PUFs. The x-axis represents the measurement number (power cycles 0-9) and the y-axis shows the corresponding value. The WCHD of all the SRAM-based PUFs is less than 10%, indicating 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 $\pm$

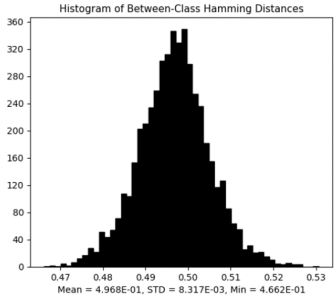


(a) WCHD

(b) HW

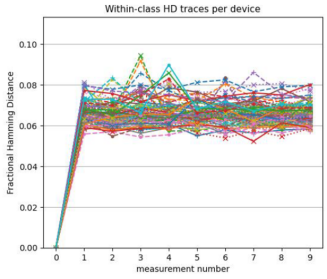


(c) MHW

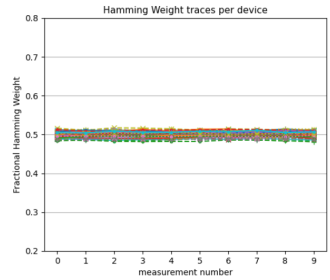


(d) BCHD

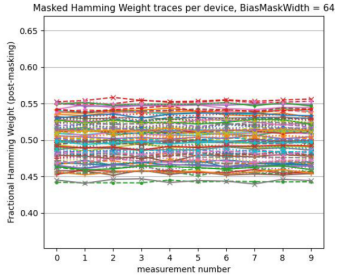
Figure 53: The PUF's characteristics of P_1_a and P_1_b.



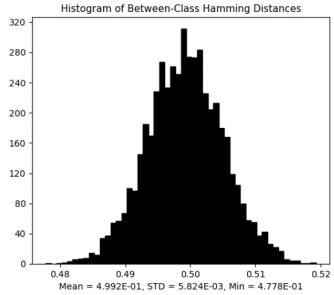
(a) WCHD



(b) HW

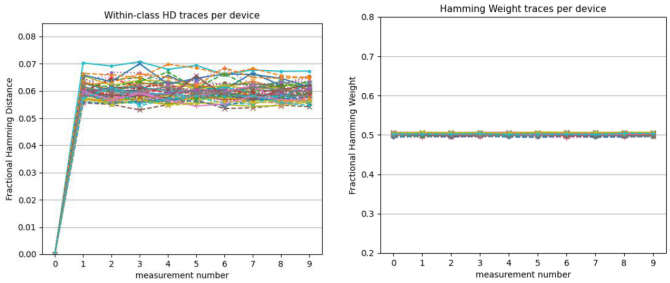


(c) MHW

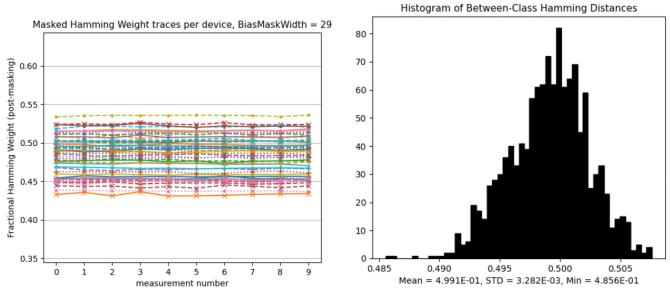


(d) BCHD

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

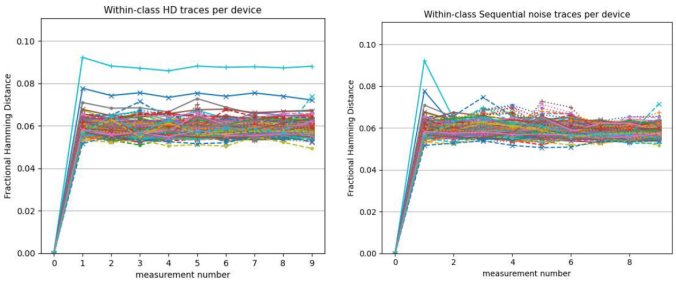


(a) WCHD *(b) HW*

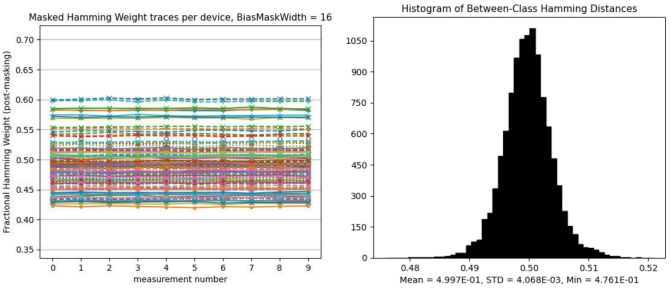


(c) MHW *(d) BCHD*

Figure 55: The PUF's characteristics of P_3.

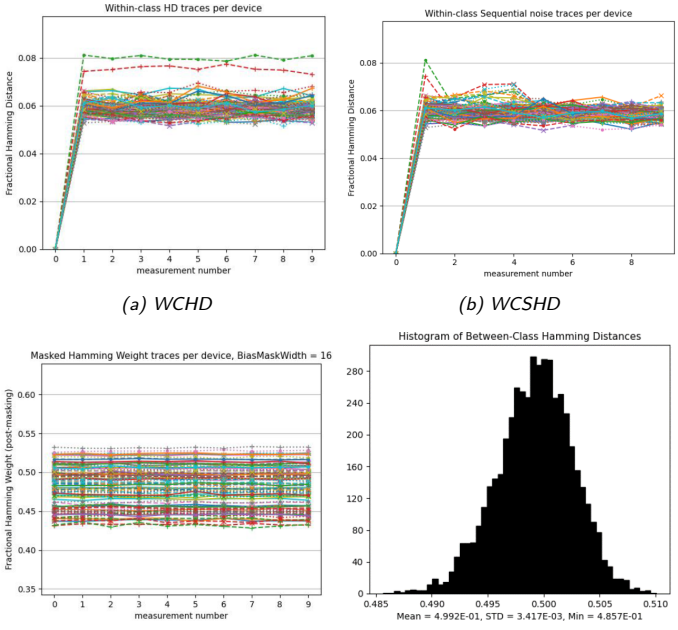


(a) WCHD *(b) WCSHD*



(c) MHW *(d) BCHD*

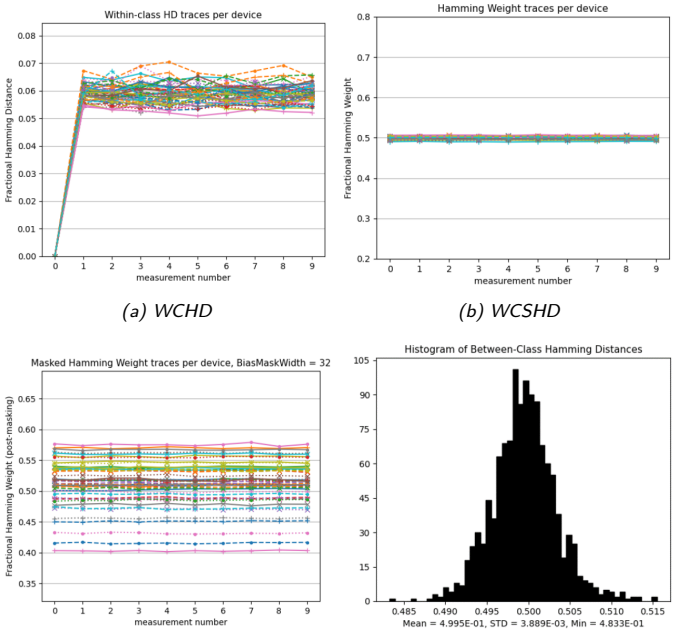
Figure 56: The PUF's characteristics of P_4_a and P_4_b.



(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

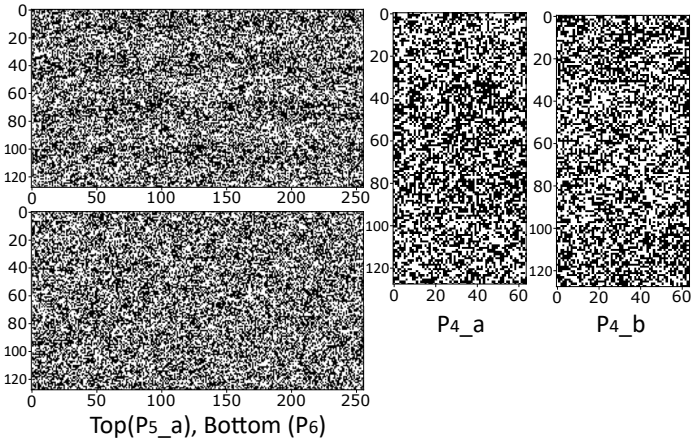


Figure 59: The start-up pattern of the P_5_a, P_6, P_4_a and P_4_b SRAM-based PUFs is represented by a sequence of bits, with white spaces denoting a logical one.

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.

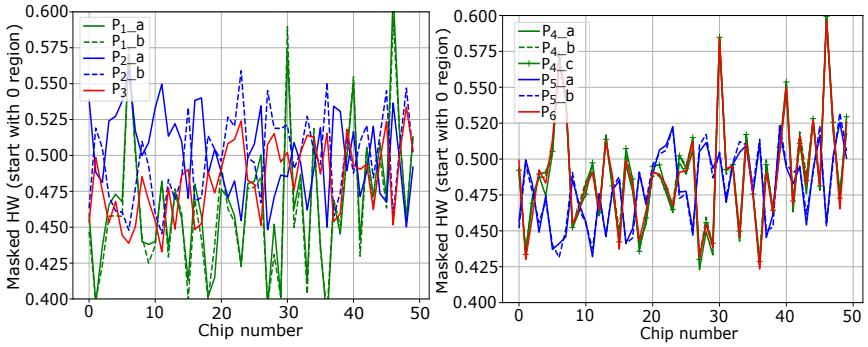


Figure 60: The correlation of SRAM-based PUFs for the bias pattern with baseline SRAM-based PUF (P_1_a) [220].

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 Conference 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.

Acknowledgements

I am deeply grateful to the Almighty Allah, who has given me abundant blessings and unwavering hope. I extend my heartfelt appreciation to Prof. Dr. Samuel Pagliarini, my esteemed Ph.D. mentor and the head of the Centre for Hardware Security. His consistent guidance, sincere efforts, and regular meetings have been instrumental in shaping my thesis. Moreover, I express my gratitude to him for his mentorship throughout my tenure at TalTech, where his timely actions and guidance have enabled me to contribute to prestigious journals relevant to my Ph.D. research. His expertise sharpened my technical skills and provided invaluable advice in various aspects of life. His caring nature and profound insights have undeniably shaped the outcome of my work.

I am deeply grateful to the Department of Computer Systems, the School of IT staff, and the esteemed faculty members for creating an exceptional academic environment that has nurtured my growth as a researcher. Their unwavering support and dedication have been instrumental in my development. I would also like to acknowledge the financial support I received during my Ph.D. studies. The generous funding provided by the Estonian Research Council through the MOBERC35 project, the European Commission through the SAFEST project, the IT Academy, the European Social Fund, and the Estonian Education and Youth Board through the EITSA18019 project, as well as the SMART4ALL program under the EU's Horizon 2020 R&D initiative, has enabled me to contribute to impactful publications.

My heartfelt thanks go to my colleagues, whose unwavering support, encouragement, and camaraderie made this journey memorable. I am immensely thankful to Dr. Levent Aksoy for his invaluable feedback, constructive criticism, and expertise in the field. His technical insights, comments, and suggestions have been crucial in structuring my work. I am deeply grateful Dr. Muayad Baqer Al-Jafar for their limitless support. I also thank my colleague and roommate, Mohammad Eslami, for sharing valuable research experiences and cherished moments. Furthermore, I thank Tiago Perez and Malik Imran for their continuous support. I am also grateful to Malik Imran for his culinary expertise, which has added joy to our time together.

My parents, sister, aunts, and uncles deserve my utmost gratitude. Their unwavering support, encouragement, and belief in me have been the pillars of my success as a researcher. I dedicate this thesis to my father as a token of my love and appreciation. May Allah grant him high ranks in Jannah. Finally, I cannot overlook the captivating beauty of Tallinn and Estonia, which have provided inspiration and a serene backdrop to my research endeavors.

Abstract
Leveraging FPGA Reconfigurability as an Obfuscation Asset

The smooth operation of IC-based systems depends significantly on using high-performance ICs. However, the production of these ICs requires the adoption of faster technology, leading to the increasing complexity of IC manufacturing and the higher cost of establishing and maintaining advanced technology foundries. Contemporary semiconductor companies adhere to a globalized IC supply chain that relies on numerous untrusted entities. The globalized IC supply chain poses several threats, including piracy, reverse engineering, overproduction, and malicious logic insertion. Various obfuscation techniques, split manufacturing, design camouflaging, and LL have been proposed to counter these threats. However, only LL can offer security across all untrusted stages of the globalized IC supply chain. The LL has been evolving for almost a decade and various defense and attack techniques have been proposed. In recent years, reconfigurable-based obfuscation techniques have been proposed using coarse or fine-grain reconfigurable blocks such as eFPGA or LUTs. Researchers typically redact the most sensitive circuit part on the reconfigurable part, keeping it as minimal as possible to avoid PPA overheads. Reconfigurable-based obfuscation requires integrating a custom tool into the traditional CAD flow, which can be challenging. To tackle this challenge, a platform or framework is necessary to enable designers to make informed decisions and manage trade-offs effectively. Such a platform would aid in evaluating design versus security trade-offs and assess the impact of implementing security measures. This thesis proposes a new obfuscation method in a custom tool that guarantees security in the globalized IC supply chain. To provide a concise overview of this thesis, the key contributions can be summarized as follows:

Tunable Design Obfuscation Technique (TOTe). A custom tool "TOTe" is developed to obfuscate the design with the given obfuscation rate. The CAD flow of this tool is security-aware and based on LUTs, but still compatible with the standard cell based physical synthesis flow. It explores the FPGA-ASIC design space and generates heavily obfuscated designs with only small parts of the logic resembling an ASIC. This specialized solution is called a "hybrid ASIC" or hASIC. Various design results, including ISCAS'85 benchmarks, are presented with different obfuscation levels to analyze trade-offs. TOTe includes decomposition and pin swapping algorithms for heavily LUT-dominated designs, enabling performance gains that can achieve performance levels comparable to ASICs. In analyzing the two selected cryptocores, AES-128 and SHA-256, they were implemented in 65nm commercial technology with different obfuscation levels. The design with 80% obfuscation for AES-128 achieved 248 MHz, while the same design for SHA-256 achieved a remarkable 368 MHz. This is compared to their respective FPGA implementations, which only reached 103 MHz and 77 MHz. Through oracle-guided attacks, it has been verified that the obfuscated designs are highly resilient against SAT attacks. The same applies to optimized designs which utilize LUT decomposition. In the case of oracle-less attacks, adversaries can only retrieve a maximum of 50% of the correct bits by guessing. In addition, customized attacks have revealed that the adversary can predict the circuit and narrow down the search space for further attacks.

The evaluation of SRAM-based PUF for bitstream security. After fabrication, threats to the end user's stage are countered using a secure bitstream in hASIC. A secure and reliable AES decryption key extracted from SRAM-based PUFs solely relied upon for the security of the hASIC bitstream.The reliability, bias pattern, entropy, uniqueness, and randomness are all factors that contribute to the robustness of SRAM-based PUF. The impact of memory orientation on the bias pattern of SRAM-based PUFs is investigated, and a 65nm CMOS chip that contains eleven SRAM macros exercising different memory- and chip-level parameters was designed and fabricated. Several parameters passed to the SRAM compiler are considered at the memory level, including the number of addresses, the number of words, the aspect ratio, and the chosen bitcell. Chip-level decisions are considered during the floorplan, including the location and rotation of each SRAM macro in the testchip. A comprehensive analysis of different memory orientations and their effect on the biasing direction is conducted, and it is revealed that specific memory orientations, namely R270 and MY90, exhibit a distinct negative biasing direction compared to other orientations. Importantly, this biasing direction remains consistent regardless of memory type, column mux ratio, memory size, or the utilization of SRAMs with different bitcells. Overall, this analysis highlights the significance of careful physical implementation and memory orientation selection in designing SRAM-based PUFs, substantially impacting their entropy and reliability. Physical measurements were performed on 50 fabricated chips to arrive at these conclusions.

In a nutshell, TOTe effectively addresses the global security challenge of design obfuscation. The analysis of SRAM-based PUF not only contributes to the security of hASIC, but also assists designers in selecting SRAM memories with properties that improve SRAM-based PUFs, thereby potentially reducing error correction efforts needed to compensate for instability.

Kokkuvõte
FPGA ümberkonfigureeritavuse rakendamine hägustamise vahendina

IC-põhiste süsteemide sujuv töö sõltub suuresti kõrgjõudlusega integraallülituste (IC) kasutamisest. Siiski nõuab nende IC-de tootmine kiirema tehnoloogia kasutuselevõttu, mis toob kaasa IC tootmise suureneva keerukuse ja täiustatud tehnoloogia tehaste rajamise, sealhulgas ka hooldamise kõrgemad kulud. Kaasaegsed pooljuhtide ettevõtted järgivad globaliseeritud IC tarneliini, mis sõltub mitmetest mitte usaldusväärsetest ettevõtetest. Globaliseeritud IC tarneliin esitab mitmeid ohte, sealhulgas piraatlus, pöördprojekteerimine, ületootmine ja pahatahtlik loogika lisamine. Mitmesugused hägustamistehnikaid on väljapakutud nende ohtude vastu võitlemiseks, nagu näiteks jagatud tootmine, disaini kamuflaaž ja loogika lukustus (LL). Kuid ainult LL saab pakkuda turvalisust kogu globaliseeritud IC ebausaldusväärsetes tarneliini etappides. LL on arenenud peaaegu kümnendi, mille jooksul on välja töödatud erinevaid kaitse- ja ründetehnikaid. Viimastel aastatel on väljatöödatud ümberkonfigureeritavaid obskureerimistehnikaid, kasutades kõrge- või madalatasemelist ümberkonfigureeritavat blokki, nagu näiteks eFPGA või LUT-d. Teadlased redigeerivad tavaliselt ümberkonfigureeritaval osal kõige tundlikumat ahela osa, hoides seda võimalikult minimaalsena, et vältida jõudluse, pinna ja energiakulu suurenemist. Ümberkonfigureeritavate obskureerimismeetodite rakendamine nõuab eriotstarbelise tööriista integreerimist traditsioonilisse arvutipõhisesse disaini (CAD) voogu, mis võib olla väljakutse. Selle väljakutsega toimetulemiseks on vajalik platvorm või raamistik, mis võimaldaks disaineritel teha informeeritud otsuseid ja haldada kompromisse tõhusalt. Selline platvorm aitaks hinnata disaini ja turvalisuse kompromisse ning hinnata turvameetmete rakendamise mõju. See väitekiri pakub välja uue hägustamismeetodi eriotstarbelises tööriistas, mis tagab turvalisuse globaliseeritud IC tarneliinis. Selle väitekirja olulisemad panused võib kokku võtta järgmiselt:

Häälestatav disaini obskureerimistehnika (TOTe). On välja töötatud eriotstarbeline tööriist "TOTe", et hägustada disain etteantud hägustamismääraga. Selle tööriista CAD-voog on turvalisust arvestav ja põhineb LUT-idel, kuid on siiski ühilduv füüsilise sünteesi voogudega, mis põhinevad standardsetel loogikaplokkidel. Antud tööriist uurib FPGA-ASIC disainiruumi ja genereerib tugevalt hägustatud disainid, mis vaid vähesel määral meenutavad algset ASIC disaini. Sellist spetsialiseeritud lahendust nimetatakse "hübriid-ASIC"-iks ehk hASIC-iks. Erinevad disaini tulemused, sealhulgas ISCAS'85 etalonid, on esitatud erinevate hägustamistasemetega, et analüüsida kompromisse. TOTe kasutab dekompositsiooni ja viigu-vahetuse algoritme LUT-idest domineeritud disainides, mis omakorda tagab jõudluse, mis on võrreldav ASIC-ide omadega. Kahe valitud krüptograafilise tuuma, AES-128 ja SHA-256, analüüsimisel realiseeriti need 65nm kommertstehnoloogias erinevate hägustamistasemetega. Disain koos AES-128 puhul saavutati 80% hägustamisega taktsageduseks 248 MHz, samas kui disain SHA-256 tuumaga saavutas märkimisväärse 368 MHz. Saadud tulemused võrreldi vastavate FPGA implementatsioonidega, mis saavutasid vaid 103 MHz ja 77 MHz. Oraklijuhitud rünnakute abil kontrolliti, et hägustatud disainid on vastupidavad SAT-rünnakutele. Sama kehtib ka optimeeritud disainide kohta, mis kasutavad LUT-dekompositsiooni. Orakli-vabade rünnakute korral arvasid vastased ainult maksimaalselt 50% õigetest bittidest ära. Lisaks on kohandatud rünnakud näidanud, et vastane suudab ennustada ahelat ja kitsendada otsinguruumi edasiste rünnakute jaoks.

SRAM-põhise PUF hindamine bittivooga seotud turvalisuse jaoks. Pärast tootmist ohud lõppkasutaja etapis lahendatakse turvalise bittivooga hASIC-is. Turvalise ja usaldusväärse AES-i dekrüpteerimisvõtme eraldamine SRAM-põhistest PUF-idest tugines üheselt hASIC-i bittivoo turvalisusele. SRAM-põhiste PUF-ide tugevusele aitavad kaasa usaldusväärsus, kallutatuse muster, entroopia, unikaalsus ja juhuslikkus. Uuriti mälupiagutusega kaasnevaid mõjusi SRAM-põhistele PUF-idele, mille jaoks kavandati ja valmistati 65 nm CMOS tehnoloogias kiip, mis sisaldab üheteistkümmet erinevat SRAM mälutüübi macrot ja kiibi taseme parameetrit. Mitmed parameetreid, mis edastatakse SRAM kompilaatorile, arvestatakse mälutasemel, sealhulgas aadresside arv, sõnade arv, kuvasuhe ja valitud bitiploki tüüp. Kiibtaseme otsused arvestatakse pinnalaotuse planeerimise käigus, sealhulgas iga SRAM makro asukoht ja suund testkiibil. Tehakse põhjalik analüüs erinevate mälupaigutuste kohta ja mõjust suunale, kus tuleb välja, et teatud mälupaigutused, nimelt R270 ja MY90, näitavad selgelt negatiivset tendentsi võrreldes teiste paigutustega. Tähtis on märkida, et antud tendents jääb püsima sõltumata mälu tüübist, veerude dekodeerimis suhtest, mälu suurusest või SRAM-ide kasutamisest erinevate bitiploki tüüpidega. Üldiselt antud analüüs rõhutab hoolika füüsilise rakenduse ja mälupaigutuse valiku olulisust SRAM-põhiste PUF-ide kujundamisel, mõjutades oluliselt nende entroopiat ja usaldusväärsust. Nende järelduste saamiseks viidi läbi füüsilised mõõtmised 50 valmistatud kiibil.

Kokkuvõtvalt, TOTe käsitleb tõhusalt ülemaailmse turvalisuse väljakutset, rakendades selleks disaini hägustamist. SRAM-põhise PUF-i analüüs mitte ainult ei aita kaasa hASIC-i turvalisusele, vaid aitab ka disaineritel valida SRAM-mälu omadusi, mis täiustavad SRAM-põhiseid PUF-e, vähendades seeläbi potentsiaalselt vajalikke vigade parandamise jõupingutusi ebastabiilsuse kompenseerimiseks.

Appendix 1

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

From FPGAs to Obfuscated eASICs: Design and Security Trade-offs

Zain Ul Abideen, Tiago Diadami Perez, Samuel Pagliarini
Centre for Hardware Security, Tallinn University of Technology (TalTech), Estonia
{zain.abideen, tiago.perez, samuel.pagliarini}@taltech.ee

***Abstract*—Threats associated with the untrusted fabrication of integrated circuits (ICs) are numerous: piracy, overproduction, reverse engineering, hardware trojans, etc. The use of reconfigurable elements (i.e., look-up tables as in FPGAs) is a known obfuscation technique. In the extreme case, when the circuit is entirely implemented as an FPGA, no information is revealed to the adversary but at a high cost in area, power, and performance. In the opposite extreme, when the same circuit is implemented as an ASIC, best-in-class performance is obtained but security is compromised. This paper investigates an intermediate solution between these two. Our results are supported by a custom CAD tool that explores this FPGA-ASIC design space and enables a standard-cell based physical synthesis flow that is flexible and compatible with current design practices. The results after physical implementation are generated for the obfuscated circuits in a 65nm commercial technology, demonstrating the attained obfuscation quantitatively. Furthermore, our security analysis revealed that for truly hiding the circuit's intent (not only portions of its structure), the obfuscated design also has to chiefly resemble an FPGA: only some small amount of logic can be made static for an adversary to remain unaware of what the circuit does.**

***Index Terms*—Hardware Obfuscation, Secure ASIC Design, CAD, Reconfigurable obfuscation, Reverse engineering**

I. Introduction

Shipment of semiconductor devices is forecast to surpass one trillion units in the year 2021, the third time this mark is surpassed in a calendar year since 2018 [1]. The majority of those devices are being manufactured by foundries that subscribe to the fab-for-hire model. Many potential threats regarding third-party foundries have been studied in recent years, include tampering, counterfeiting, reverse engineering, and overproduction. On the other hand, many techniques have been devised to mitigate threats from untrusted fabrication. Countermeasure techniques to increase the IC security against not only third-party foundries but also from the end-user have been recently demonstrated. Notable examples include IC Camouflaging [2]–[4], Logic Locking [5]–[7], and Split Manufacturing [8], [9].

Generally speaking, all of the aforementioned countermeasures attempt to "hide" the design from adversaries and can be classified as obfuscation techniques. Unfortunately, none of these techniques is currently adopted in large-scale production of ICs, for reasons that include (lack of) practicality [8] and insufficient security guarantees [10]. Another approach towards obfuscation is the use of an FPGA (or FPGA-like) design, where the configuration *bitstream serves as a key* to unlock the functionality of the circuit [11]. Our paper too explores this possibility. The fabric in an FPGA contains reconfigurable elements, but this flexibility incurs a limited performance. On the other hand, ASIC requires one-time placement, it is static (non-reconfigurable), but it provides best-in-class performance. As shown in Fig. 1, performance increases if we move from right to left. Contrarily, area, obfuscation, and flexibility increase if we move from left to right.

This work has been partially conducted in the project "ICT programme" which was supported by the European Union through the European Social Fund. It was also partially supported by the Estonian Research Council grant MOBERC35.

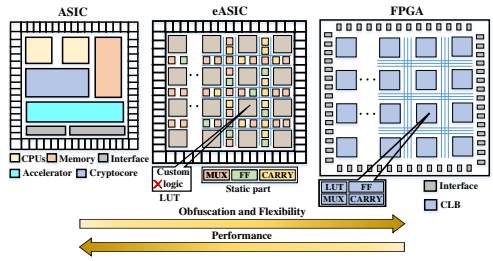


Fig. 1: The design obfuscation landscape, from ASICs to FPGAs. The relative sizes are notional.

Design obfuscation concept: In this work, we propose to obfuscate a design by exploiting the best of both worlds. The generated device is a hybrid which includes *reconfigurable elements* (analogous to the FPGA) and also includes ASIC cells as *static elements*, i.e., gates with fixed functionality after fabrication. Previous research on obfuscation by reconfigurable elements has focused on keeping the reconfigurable portion as small as possible [12], [13], which is logical if the goal is to keep overheads under control. However, we later show that true hiding of the circuit's intent requires a *high degree of obfuscation* that is usually not explored in the state of the art. We term our in-between solution an "embedded ASIC" (**eASIC**). Thus, our eASIC device is largely non-functional until it is programmed. Our main contribution is a tool for automatically obfuscating a design in the form of eASIC, where the obfuscation range can be from 0 to 100%. Furthermore, its physical synthesis flow is standard-cell based that is compatible with current design and fabrication practices.

II. A CAD flow for eASIC

Our CAD flow is centered around a tool named **T**uneable Design **O**bfuscation **T**echnique using **e**ASIC, or **TOTe** for short. This section explains the CAD flow of eASIC and TOTe's main features. TOTe generates a hybrid design with static and reconfigurable elements, which we refer to as eASIC. For the reconfigurable elements, we implement the logic utilizing the notion of programmable LUTs (Look Up Tables) - same as in FPGAs. The complete TOTe design flow for generating an eASIC is shown in Fig. 2 and it consists of three phases. In the **first phase** of our flow, the design under obfuscation, described in register-transfer level (RTL) form, is synthesized using a commercial FPGA synthesis tool. As a result, the netlist contains all typical FPGA primitives, i.e., FFs, MUXs, and LUTs. The input design requires no special annotations, synthesis pragmas, or any other change in its representation.

Next, in the **second phase**, TOTe requires the ASIC standard cell library of choice. As highlighted in the center of Fig. 2, the core idea of TOTe is to **replace reconfigurable logic for static logic**. For TOTe, only the LUTs are treated as reconfigurable logic, and, any other primitives from the FPGA synthesis are automatically transformed into static logic. TOTe utilizes its own **obfuscation and**

2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST) | 978-1-6654-4185-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/ASIANHOST53231.2021.9699758

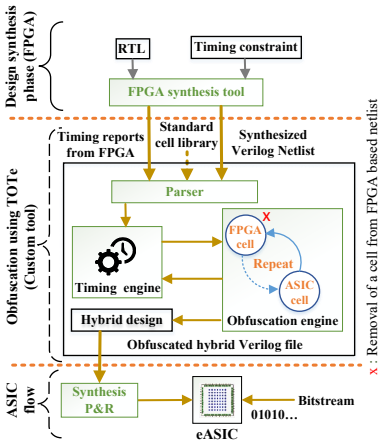


Fig. 2: The CAD flow for eASIC, combining FPGA/ASIC synthesis.

timing engines. These engines drive the security vs. performance trade-offs of the tool. For this phase, the designer provides an obfuscation target in terms of percentage, which determines the portion of logic that will remain reconfigurable as LUTs. TOTe builds a tree representation of the circuit where primitive types are annotated for every instance. For LUTs, in particular, the tool also annotates their masking patterns (i.e., the portion of the bitstream associated with an individual LUT). By using truth tables populated by the masking patterns, TOTe builds combinational logic that is equivalent to the LUT's intended usage (static logic). Finally, TOTe exports an obfuscated hybrid Verilog file (eASIC), timing report, and area report. A designer can repeat this procedure until he achieves his obfuscation and performance targets.

In the **third and final phase**, the obfuscated netlist from TOTe is synthesized using any commercial ASIC CAD tool and implemented using an also commercial tool where traditional P&R, CTS, DRC, LVS, etc. steps are executed. Finally, the tapeout database is sent to the foundry for fabrication.

III. Experimental Results using TOTe

This section reports the analysis of security versus performance, security versus area for selected designs and reports the results for numerous designs after obfuscation. For all experimental results that follow, FPGA synthesis was executed in Vivado and the targeted device is Kintex-7 XC7K325T-2FFG900C, which contains only 6-input LUTs. For the ASIC flow, the implementations are done using a commercial 65nm PDK with three standard cell flavors (LVT/SVT/HVT) and tools from Cadence (i.e., Genus and Innovus).

Custom standard-cell based LUTs: The premise of eASIC is to have reconfigurable and static elements that can be integrated transparently. For this reason, we have designed our own custom LUTs (LUT_1, LUT_2,..., LUT_6) by following VPR's template [14]. Different from FPGAs that generally implement only one LUT size, for eASIC we have the flexibility to implement more than one size because our design intent will not change. By doing this, we preserve area and potentially increase the performance of eASIC. These blocks are highly compact since the main design goal for them was area/density. Each LUT has its own registers for storing the configuration, a functionality that is enabled by including three extra configuration pins ($serial\_in$, $serial\_out$, and $enable$). The LUTs are connected to one another in a daisy chain that is analogous to a scan chain.

TABLE I: Detailed results for selected designs using TOTe

Design	Obf. (%)	*sumCP* (ns)	*CP* (ns)	Area-RE (μm^2)	Area-ST (μm^2)	LUT (RE)	LUT (ST)
SBM	98	16088.690	0.490	13190.04	0	29	0
	95	15895.826	0.484	12762.00	21.40	28	1
	92	15877.962	0.464	12547.80	32.11	27	2
	89	15458.506	0.461	12438.72	37.56	26	3
	86	15370.682	0.459	12224.52	48.27	25	4
SHA-256	98	7425.731	0.962	1313150.76	10291.86	2195	44
	95	7354.593	0.871	1275984.00	28875.24	2128	111
	92	7322.155	0.871	1233448.56	50142.96	2060	179
	89	7301.945	0.871	1179674.64	77029.92	1992	246
	86	7164.025	0.871	1125799.56	103967.46	1925	313
FPU	98	2909.063	0.707	1031676.84	1250.028	2487	50
	95	2734.008	0.650	1003225.68	2672.586	2412	126
	92	2572.952	0.650	966715.20	4498.11	2336	202
	89	2478.732	0.650	935060.04	6080.868	2259	279
	86	2410.211	0.650	893005.56	8183.592	2183	355

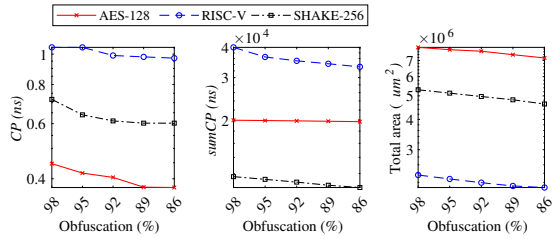


Fig. 3: Obfuscation results for AES-128, RISC-V and SHAKE-256.

Design Space Exploration in TOTe: For our first experiment, we selected a small but representative design which covers all possible FPGA primitives: a schoolbook multiplier (SBM), which is a bit-serial polynomial multiplication circuit. For a SBM design that is synthesized targeting a very high frequency, the CP and $sumCP$ become, as calculated by TOTe, 0.490 ns and 16088.69 ns, respectively. These values correspond to a design obfuscated at 100%, i.e., the design has only reconfigurable logic. The absolute accuracy of these values is not relevant since final timing analysis is performed using a commercial physical synthesis engine later.

While the SBM design is an interesting motivational example, it showed that CP tends to saturate while the $sumCP$ continues to improve as the obfuscation is reduced. Next, we wanted to determine if the same saturation trend appears for other designs and the results are reported in Table I. From these experiments, it is possible to conclude that the performance of numerous designs saturates incredibly fast as we decrease the obfuscation level, even when the obfuscation range is limited to 86-100%. Moreover, the results for AES-128, RISC-V, and SHAKE-256 have been depicted in Fig. 3. Several other designs, including ISCAS'85 benchmarks and known opencores, have been evaluated. The complete set of results can be found in our git repository [15].

IV. Security Analysis

As compared to conventional logic locking, the LUTs introduced in eASIC are the key-gate equivalents. In principle, a single LUT_n ought to be equivalent to 2^n XOR/XNOR key-gates. In practice, the LUT logic has similarities to a run of key-gates (see [6]) due to the n-to-1 multiplexing nature of it, which reduces the search space and may possibly make eASIC vulnerable to well-known oracle-based attacks

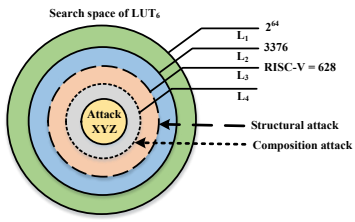


Fig. 4: The search space of LUT_6 as it shrinks with different attacks.

(e.g., SAT). However, notice that we are considering designs with target obfuscation rates higher than 86%, which results in bitstreams with thousands of bits. Even for a small and combinational design as the ISCAS'85 c7552, 50% of obfuscation requires a bitstream with approximately 11k bits. The SAT attack is not able to find the correct key, even running for more than 60 hours. We make the following assumptions to build a threat model:

- The adversary goal is to identify the circuit intent, even in the presence of obfuscation. For this goal, the adversary **does not need** to recreate the bistream.
- The adversary has access to the GDSII file of the eASIC design. He or she is skilled in IC design and has no difficulty in understanding this layout representation.
- The attacker can recognize the standard cells, thus the gate-level netlist of the obfuscated circuit can be easily recovered [16].
- We assume that the attacker can differentiate between design inputs and reconfiguration pins [10], [17].
- We assume the adversary can group the standard cells present in the static logic and convert them back into reconfigurable logic (i.e., LUT representation)[1].

In order to evaluate the security hardness of eASIC, we propose two different attacks: one based on the *structure* of design and another based on the *composition* of known different circuits.

Structural Analysis Attack: The goal of this attack is to decrease the key search space and attempt to recover the bitstream. As we mentioned before, the key search space is 2^{64} for a single LUT_6. But this assumption only holds if the FPGA synthesis is actually capable of exercising the entire key search space, which our results reveal that is far from possible. We have synthesized a large number of representative designs (>30) and counted how many unique LUT_6 masking patterns appear in the corresponding netlists. Designs of varied complexity, size, and functionality where added until the combined number of unique masking patterns appears to settle, forming a set of $m = 3376$ elements. This result alone, albeit being empirical, reduces the global search space from L_1 to L_2 as illustrated in Fig. 4.

We utilize tuples of $\langle pattern, frequency \rangle$ for tracking how often masking patterns repeat. The tuples are referenced by integer identifiers and ordered by frequency. Our analysis reveals that the RISC-V netlist has 628 unique LUTs and only 3 occur more than 100 times.In practice, if the attacker could know for a fact that the obfuscated circuit is indeed RISC-V, the search space would shrink further. The shrunk search spaces are labeled L_3 in Fig. 4. The question then becomes whether the static portion of the circuit is large enough for the adversary to be confident that the circuit under attack can be labelled as circuit C_1, C_2, or C_n. We investigate this possibility by

[1]This is a very generous concession since the static logic is repeatedly optimized during logic and physical synthesis.

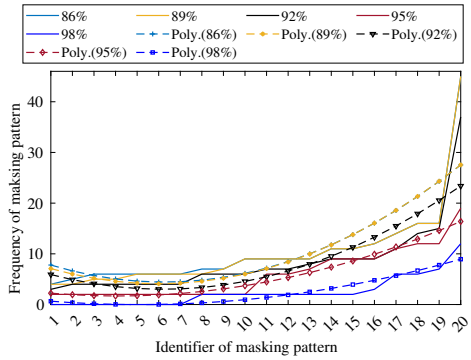


Fig. 5: The structural analysis of MIPS and RISC-V.

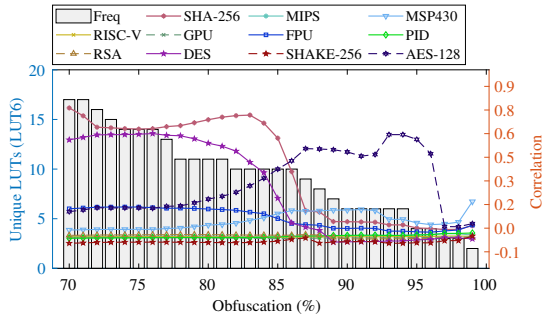


Fig. 6: The correlation of SHA-256 versus numerous other designs.

further analysing the behaviour of the frequency of masking patterns, as depicted in Fig. 5. For this, we utilized polynomial trendlines for a portion of identifier of masking pattern, considering netlists generated by TOTe at 98%, 95%, 92%, 89%, and 86% obfuscation levels. It is noteworthy that the trendlines become better frequency predictors as the obfuscation level is decreased. For RISC-V, in particular, the adversary can guess a small number outliers and the best guess (when obfuscation is 86%) is far from the original frequencies (>100).

Composition Analysis Attack: The goal of this attack is to identify the circuit by correlation to known circuits. This attack also exploits the frequency of the LUT_6, but here we correlate entire designs (instead of pattern-frequency tuples) based on their composition. We consider that the attack is successful if the adversary is able to identify the circuit (see threat model, 1st bullet). In this experiment, we performed correlation analysis for the well-known SHA-256 crypto core as shown in Fig. 6. The objective of this experiment is to analyze the leaked information from the static part of an obfuscated design against a database[2] of circuits. We have obfuscated SHA-256 in the 70-100% range and then correlated the static portion of the design with the database of known circuits. In Fig. 6, we show the results where the x-axis shows the obfuscation percentage and the y-axis shows correlation (right) and number of unique LUTs (left). For this circuit, three regions of interest can be defined: 97-100% (no correlation), 86-96% (strong correlation to another circuit), and 70-85% (correlation to itself). This attack reveals that if the adversary goal is solely to identify the circuit's intent,

[2]We assume the adversary can obtain samples of open source cores from repositories and execute FPGA synthesis on them with his tool of choice.

TABLE II: Results for the implementation of SHA-256 for different obfuscation levels

CAD Flow	Obf. (%)	Density	Area (μm^2)	Freq. (MHz)	T. Power (mW)	# LUT	Comb.	Seq.
FPGA	100	–	–	77	191	2238	–	1830
TOTe	100	46%	1412227	166.7	274.24	2238	82756	105128
TOTe	90	45%	1274690	178.6	262.21	2015	83452	94876
TOTe	85	46%	1215328	200	277.82	1904	79626	90420
TOTe	80	54%	1135752	200	262.77	1792	74000	83790
TOTe	0	71%	43097	200	6.93	0	3165	1806
ASIC	0	71%	60563	769	33.55	0	3165	1806

eASIC can be as vulnerable as an ASIC design. To mitigate this undesirable effect, obfuscation levels should remain relatively high. Otherwise, if the obfuscation lies between 70 and 84%, the search space would shift from L_3 to L_4 as shown in Fig. 4.

V. Physical Synthesis for eASIC

This section contains the physical implementation results for an obfuscated SHA-256 core. We have selected SHA-256 as it is popular and widely used in cryptography. The variants of the design with different obfuscation levels are implemented with the aid of the LUTs defined in Section III. The results obtained after implementation are focused on performance vs. area trade-offs for the 80-100% obfuscation range as determined by the security analysis of Fig. 6. Initially, we synthesized and implemented the SHA-256 core on FPGA. This implementation achieves a frequency of only 77 MHz (for reference, the Kintex-7 family is produced on a 28nm CMOS technology). To start the analysis, we select 100% obfuscation as a baseline design because it is fully reconfigurable and somewhat analogous to an FPGA design.

The implementation results for 0%, 80%, 85%, 90%, and 100% obfuscation are listed in Table II, obtained after physical synthesis and are for the worst process corner (SS) and a nominal temperature of 25°C. It is noteworthy that the performance of the design is increasing as we decrease the level of security. This behaviour is clearly depicted in the fourth column of Table II and matches the goal we set from the beginning: to trade performance for security. Here we also show that performance saturates rather quickly, as predicted by TOTe in Section III. The area of the design is proportional to the obfuscation level which means that increasing the security of design will cause area overhead. The results obtained from the physical synthesis justify trade-offs and Table II show the resource requirements.

VI. Comparison and discussion

From the many results, we conclude that obfuscation levels should be relatively high to achieve a considerable security, thus the majority of the eASIC logic should be reconfigurable logic (i.e., LUTs). Having a large portion of reconfigurable logic provides an opportunity to correct the issues/bugs that could be easily fixed during the reconfiguration phase. Naturally, there are limitations since a portion of the system consists of static logic and cannot be modified. This limitation could be eased if the eASIC layout were to include spare LUTs. To some degree, those spare LUTs could also be used to make side-channel attacks less successful.

A recent trend in obfuscation research is the use of embedded FPGA (eFPGA) [18], [19]. A very similar approach is also found in [20], where authors perform obfuscation with transistor-level granularity. While there are advantages to this practice, it has been used selectively to only protect key portions of a design and therefore keep the performance penalty as low as possible. The challenge is in determining which portions of the circuit merit protection and which ones do not. Our eASIC approach bypasses this question almost completely by only revealing (portions of) critical paths when they are selected to become static logic, which we consider an advantage if the ASIC-equivalent performance can be sacrificed.

VII. Conclusions

In this paper, we have developed a custom tool (TOTe) that obfuscates a design and transforms it into an eASIC device. Our eASIC solution contrasts with the current practice of eFPGA for obfuscation and this is not by coincidence: our experimental results show that obfuscation rates have to be high to protect not only the bitstream but also the design's intent. This is a key finding of our research which we hope can help to steer current obfuscation practices in the literature. Our findings are also validated in a commercial physical synthesis tool with industry-strength timing and power analysis, from which we confirm that TOTe's trade-off analysis is sufficiently accurate.

References

[1] IC Insights, "Semiconductor units forecast to exceed 1 trillion devices in 2021," [Online]. Available at: https://www.icinsights.com/news/bulletins/Semiconductor-Units-Forecast-To-Exceed-1-Trillion-Devices-Again-In-2021/.

[2] M. Yasin *et al.*, "Removal attacks on logic locking and camouflaging techniques," *IEEE Transactions on Emerging Topics in Computing*, vol. 8, no. 2, pp. 517–532, 2020.

[3] R. P. Cocchi *et al.*, "Circuit camouflage integration for hardware ip protection," in *DAC*, 2014, pp. 1–5.

[4] M. Li *et al.*, "Provably secure camouflaging strategy for ic protection," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 38, no. 8, pp. 1399–1412, 2019.

[5] K. Zamiri Azar *et al.*, "Threats on logic locking: A decade later," in *GLSVLSI '19*, 2019, p. 471–476.

[6] M. Yasin *et al.*, "On improving the security of logic locking," *IEEE TCAD*, vol. 35, no. 9, pp. 1411–1424, 2016.

[7] J. Sweeney *et al.*, "Latch-based logic locking," in *2020 IEEE HOST*, 2020, pp. 132–141.

[8] T. D. Perez *et al.*, "A survey on split manufacturing: Attacks, defenses, and challenges," *IEEE Access*, vol. 8, pp. 184 013–184 035, 2020.

[9] J. Rajendran *et al.*, "Is split manufacturing secure?" in *2013 DATE*, 2013, pp. 1259–1264.

[10] P. Subramanyan *et al.*, "Evaluating the security of logic encryption algorithms," in *2015 IEEE HOST*, 2015, pp. 137–143.

[11] B. Liu *et al.*, "Embedded reconfigurable logic for asic design obfuscation against supply chain attacks," in *DATE*, 2014, pp. 1–6.

[12] H. Mardani Kamali *et al.*, "Lut-lock: A novel lut-based logic obfuscation for fpga-bitstream and asic-hardware protection," in *2018 IEEE ISVLSI*, 2018, pp. 405–410.

[13] S. D. Chowdhury *et al.*, "Enhancing sat-attack resiliency and cost-effectiveness of reconfigurable-logic-based circuit obfuscation," in *2021 IEEE ISCAS*, 2021, pp. 1–5.

[14] K. E. Murray *et al.*, "Vtr 8: High-performance cad and customizable fpga architecture modelling," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 13, no. 2, 2020.

[15] Z. U. Abideen *et al.*, "TOTe (Tuneable Design Obfuscation Technique using eASIC)," 2021. [Online]. Available: https://github.com/Centre-for-Hardware-Security/eASIC

[16] R. Torrance *et al.*, "The state-of-the-art in ic reverse engineering," in *CHES 2009*, C. Clavier *et al.*, Eds., 2009, pp. 363–381.

[17] M. Yasin *et al.*, "Provably-secure logic locking: From theory to practice," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, p. 1601–1618.

[18] B. Hu *et al.*, "Functional obfuscation of hardware accelerators through selective partial design extraction onto an embedded fpga," in *GLSVLSI '19*, 2019, p. 171–176.

[19] J. Chen *et al.*, "DECOY: DEflection-Driven HLS-Based Computation Partitioning for Obfuscating Intellectual PropertY," in *2020 IEEE DAC*, ser. DAC '20. IEEE Press, 2020.

[20] M. M. Shihab *et al.*, "Design obfuscation through selective post-fabrication transistor-level programming," in *DATE*, 2019, pp. 528–533.

Appendix 2

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

A Security-Aware and LUT-Based CAD Flow for the Physical Synthesis of hASICs

Zain Ul Abideen, *Graduate Student Member, IEEE*, Tiago Diadami Perez, *Graduate Student Member, IEEE*, Mayler Martins, and Samuel Pagliarini, *Member, IEEE*

***Abstract*—Numerous threats are associated with the globalized integrated circuit (IC) supply chain, such as piracy, reverse engineering, overproduction, and malicious logic insertion. Many obfuscation approaches have been proposed to mitigate these threats by preventing an adversary from fully understanding the IC (or parts of it). The use of reconfigurable elements inside an IC is a known obfuscation technique, either as a coarse grain reconfigurable block (i.e., eFPGA) or as a fine grain element (i.e., FPGA-like lookup tables). This article presents a security-aware CAD flow that is LUT-based yet still compatible with the standard cell-based physical synthesis flow. More precisely, our CAD flow explores the FPGA-ASIC design space and produces heavily obfuscated designs where only small portions of the logic resemble an ASIC. Therefore, we term this specialized solution a hybrid ASIC (hASIC). Nevertheless, even for heavily LUT-dominated designs, our proposed decomposition and pin swapping algorithms allow for performance gains that enable performance levels that only ASICs would otherwise achieve. On the security side, we have developed novel template-based attacks and also applied existing attacks, both oracle-free and oracle-based. Our security analysis revealed that the obfuscation rate for an SHA-256 study case should be at least 45% for withstanding traditional attacks and at least 80% for withstanding template-based attacks. When the 80% obfuscated SHA-256 design is physically implemented, it achieves a remarkable frequency of 368 MHz in a 65-nm commercial technology, whereas its FPGA implementation (in a superior technology) achieves only 77 MHz.**

***Index Terms*—Hardware obfuscation, hybrid ASIC (hASIC), LUT-based obfuscation, reverse engineering, secure ASIC design.**

I. Introduction

NOWADAYS, high-performance and energy-efficient integrated circuits (ICs) are enablers in a variety of application domains. However, this demands the fabrication of ICs in advanced technology nodes. Current predictions are that the sales of semiconductor devices will rise to $680B in 2022, the first time this mark has been surpassed in a calendar year since 2020 [1]. In tandem, the majority of IC design houses are adhering to a globalized supply chain to outsource fabrication from pure-play foundries. Even very large semiconductor companies rely on the so-called fab-for-hire model [2], [3], a framework that originates from the technological and financial challenges of developing and maintaining a foundry. The estimated cost to build a 3-nm production line is $15B–$20B [4]. The trend is clear: more than ever, fabless design companies rely on outsourcing the manufacturing of their ICs.

While this business model enables design houses to have access to high-end manufacturing, the integrity and trustworthiness of the ICs are potentially affected. For manufacturing an IC, the design house must share a blueprint of the IC with the foundry. This blueprint inevitably exposes all aspects of the IC and its many parts. A rogue element within the foundry can entirely or partially copy the design, i.e., the foundry and its employees are considered *potential adversaries*. Many potential threats are associated with the untrusted fabrication aspect of a globalized IC supply chain [5]. Such threats include tampering, counterfeiting, reverse engineering, and overproduction.

Numerous techniques have been devised to protect against the aforementioned security threats. Countermeasures to secure an IC also apply to a malicious end user that can be interested in reverse engineering a design. Noteworthy examples of countermeasures are Logic Locking [6], [7], [8], [9], [10], IC Camouflaging [11], [12], [13], Split Manufacturing [14], [15], and FPGA-like obfuscation approaches [16], [17], [18], [19], [20], [21], [22], [23], [24]. The latter style of obfuscation attempts to exploit an FPGA (or FPGA-like) fabric, where the functionality of the circuit is hidden by the configuration and the *bitstream serves as a key to unlock the design*.

Generally, the fabric in an FPGA device contains many reconfigurable blocks that can be leveraged for obfuscation purposes. The ability to reconfigure a device does incur performance penalties (i.e., FPGA versus ASIC). Being so, custom solutions where only a small portion of the design is reconfigurable have been sought, a solution typically termed eFPGA. This work also takes advantage of this possibility. A visualization of the obfuscation landscape is given in Fig. 1. As illustrated, performance increases if we move from right to left. Contrarily, obfuscation and flexibility increase if we move from left to right. However, we argue that *neither extremes of the landscape are a good design point* for circuits with stringent security and performance constraints. A midpoint solution is a better tradeoff, which is precisely the motivation for our work. We term our midpoint solution a hybrid ASIC (hASIC).

In [25], we have described an initial attempt at exploring and automating the design spaces captured in Fig. 1. In this

Manuscript received 13 July 2022; revised 25 October 2022 and 3 February 2023; accepted 10 February 2023. Date of publication 14 February 2023; date of current version 20 September 2023. This work was supported by the Project "ICT Programme" which was supported by the European Union through the ESF. This article was recommended by Associate Editor J. Rajendran. *(Corresponding author: Zain Ul Abideen.)*

Zain Ul Abideen, Tiago Diadami Perez, and Samuel Pagliarini are with the Department of Computer Systems, Centre for Hardware Security, Tallinn University of Technology, 12616 Tallinn, Estonia (e-mail: zain.abideen@taltech.ee; tiago.perez@taltech.ee; samuel.pagliarini@taltech.ee).

Mayler Martins is with SRG, Synopsys Inc., Mountain View, CA 94085 USA (e-mail: mayler.martins@synopsys.com).

Digital Object Identifier 10.1109/TCAD.2023.3244879

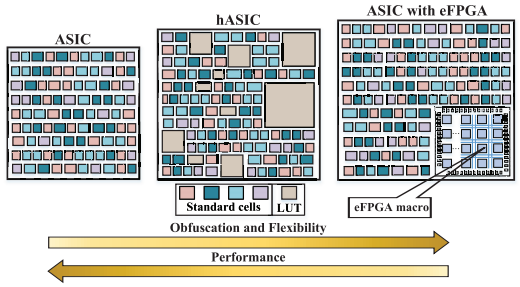


Fig. 1. Design obfuscation landscape.

work, we extend and improve our results considerably while keeping the same general theme: we seek to obfuscate a circuit by generating a hybrid design that consists of a reconfigurable portion and static logic. The *reconfigurable* part provides the obfuscation while the *static* logic provides performance benefits. We perform our design space exploration at the block level. Finally, the architecture of the generated block is a mix of *reconfigurable* and *static* cells. The reconfigurable part is implemented with programmable LUTs; the circuit is largely nonfunctional until it is programmed.

Earlier obfuscation techniques utilizing reconfigurable elements have focused on keeping the reconfigurable part as small as possible. Understandably, the goal would be to avoid large performance and area overheads. However, we emphasize (and later provide results) that proper hiding of the circuit's intent requires a *high degree of obfuscation* that is generally not investigated in the state-of-the-art. For this reason, in [25], we have proposed a CAD tool for automatically obfuscating a design, thus, generating a specialized solution called hASIC that is compatible with standard-cell-based flows and current design and fabrication practices. In this work, we markedly extend the CAD tool from [25]. The main contributions of this work are as follows.

1) Specialized algorithms for performance improvement of hASIC designs, including LUT decomposition and pin swapping approaches.
2) An analysis of performance versus obfuscation and area versus obfuscation tradeoffs for numerous designs, including known benchmarks.
3) A detailed analysis (physical synthesis) of performance, power, and area versus obfuscation for SHA-256, including tapeout-ready layouts in a 65-nm commercial technology.
4) Thorough analysis of hASIC's security against custom attacks and known oracle-based and oracle-less attacks.

II. CAD Flow for hASIC

Our CAD flow utilizes a custom tool named tuneable design obfuscation technique using hASIC (TOTe). Our custom tool produces an hASIC design with reconfigurable and static logic. For the reconfigurable portion, we implement the logic utilizing the notion of programmable lookup tables (LUTs)—same as in FPGAs. The complete process for obfuscating a design is fully automated and infers a marginal increase in design time (when compared to a traditional ASIC flow).

At its core, TOTe looks for critical paths and replaces "slow" reconfigurable elements with "fast" static ones. From this point of view, TOTe's design decisions are decoupled from security decisions. Later, in Section VI, we introduce a well-defined threat model and provide insights into the security of an hASIC design. A designer using TOTe only has to define an obfuscation target obf_c which is the percentage of LUTs that should remain reconfigurable (obfuscated).

A. Overview of TOTe

Initially, a commercial FPGA synthesis tool is utilized to synthesize the design under obfuscation (DUO), described in the register-transfer level (RTL) form. The DUO does not require any particular change in its representation. Then, the commercial FPGA synthesis tool generates a synthesized netlist and a timing report. This netlist includes all the typical FPGA primitives, i.e., LUTs, MUXs, and FFs.

Next, TOTe takes the ASIC standard cell library of choice as well as the outputs generated by the FPGA synthesis. The parser of TOTe reads the elements from the netlist and the paths from the timing report, which are then processed by a timing engine. The primary goal of TOTe is to *replace FPGA cells for ASIC cells*. More precisely, TOTe selects LUTs in the critical path (i.e., the path with the highest delay) and replaces them with standard cells that implement the same logic (except the programmability aspect is taken out). This process is repeated until enough LUTs have been converted to standard cells according to a user-provided obfuscation target (obf_c). The obfuscation target determines the ratio of the LUTs that must remain programmable. Replacing LUTs for static logic reduces the area, power, and delay (thus, improving the performance of the design). Finally, an obfuscated *hybrid* Verilog file containing reconfigurable and static LUTs is generated as the output.

In order to finalize the hASIC design, a commercial physical synthesis tool is used to implement it. Then, the foundry receives the layout and fabricates the design.

B. Detailed Flow and Internal Architecture of TOTe

The complete design flow for obfuscating a design, generating an hASIC along with logical and physical synthesis, is illustrated in Fig. 2 and comprises a total of seven steps, which we represent as circled numbers in the text that follows.

In Step ①, the DUO's RTL is synthesized using a commercial FPGA synthesis tool. The DUO requires no special annotations, no synthesis pragmas, nor any other change in its representation. Outputs from Step ① are in the form of a synthesized netlist and a timing report. The netlist comprises all the typical FPGA primitives, i.e., MUXs, LUTs, and FFs. We note that, at this point, the logic of the design is 100% obfuscated since it is entirely captured by LUTs. In very short words, the next steps of TOTe will find LUTs that are good candidates for being replaced by static logic. This is the *core functionality* of TOTe and is illustrated in the bottom left corner of Fig. 2.

Next, in Step ②, preprocessing takes place. This step aims to filter and interpret the timing report and Verilog netlist. The parsing of the timing report is a relatively trivial task. The

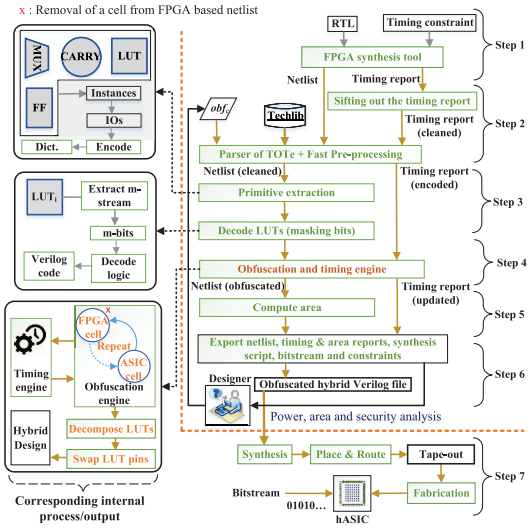


Fig. 2. Overview of TOTe's obfuscation flow and its inner steps.

Algorithm 1: TOTe's Obfuscation Procedure

Input: L (*list of LUTs*), P (*list of paths*), obf_c (*obfuscation criterion*)
Output: *hASIC* $\leftarrow f(input)$

1 $L_{ST} \leftarrow \phi$, $L_{RE} \leftarrow L$
2 **while** *SIZE_OF* $(L_{ST}) \leq obf_c$ **do**
3 $path \leftarrow$ FIND_CRITICAL(P)
4 $lut \leftarrow$ FIND_SLOWEST($path$)
5 **if** $lut \in L_{RE}$ **then**
6 INSERT(lut, L_{ST})
7 REMOVE(lut, L_{RE})
8 UPDATE_TIMING(lut, P)
9 **else**
10 REMOVE($path$, P)
11 **for** *each* $lut \in L_{ST}$ **do**
12 DECODE(lut)
13 **for** *each* $lut \in L_{RE}$ **do**
14 GEN_CASE_0_1(lut)
15 DECOMPOSE_OPT(lut)
16 SWAP_PINS(lut)
17 *hASIC* $\leftarrow L_{ST} \cup L_{RE}$

timing report contains information that should be discarded (empty lines, headers, etc.) for which a bash script has been written. After filtering the timing report, every analyzed path may now contain four FPGA primitives: FF, CARRY, LUT$_i$, and MUX. TOTe encodes (hashes) the instance names to avoid lengthy string representations. The preprocessing step ends when TOTe produces a list of timed paths, where each path contains a list of hashed instances and associated delay values. Note that an instance can appear in many paths and also can appear in many paths under different timing arcs. Finally, the list of timed paths is sorted in ascending order. As a result, the path that has the highest delay (critical path) is referred to as CP and the sum of all CPs is referred to as sumCP.[1]

Step ③ is the process of primitive extraction and LUT decoding. TOTe builds a graph representation of the netlist to keep track of port connections such that the circuit structure can be preserved once optimizations are applied. Under the graph representation, primitive types are annotated for every instance; For LUTs, in particular, the tool also annotates their masking patterns (i.e., configuration bits of an individual LUT). In practice, TOTe is able to interpret the LUT encoding scheme utilized in the netlist coming from FPGA synthesis. For the case of a LUT$_6$, the 64-bit masking pattern extracted from the netlist is converted to a truth table with six inputs and one output. The masking pattern determines which combinations of inputs generate outputs as 1s and 0s. The process is identical for smaller LUTs, which then have smaller truth tables. By using truth tables populated by the masking patterns, TOTe builds combinational logic that is equivalent to the LUT's logic. The truth tables are exported as synthesizable Verilog code. Other primitives, such as FF and MUX, require no decoding and are directly translated to their ASIC equivalents.

[1]CP and SumCP are analogous to WNS and TNS in traditional static timing analysis (STA), except all paths, here, are assumed to pass timing checks. For simplicity, no negative values are, therefore, considered in this analysis.

TOTe comes with obfuscation and timing engines that drive the security versus performance objectives of the tool. These engines are utilized in Step ④ and are responsible for different important tasks, including timing analysis, critical path identification, and replacement of reconfigurable cells for static cells. Algorithm 1 describes the different operations inside the obfuscation main loop of the tool, where L is a list of LUTs, P is a list of timed paths, and obf_c is the obfuscation criterion. The internal variables L_{ST} and L_{RE} are lists of LUTs in static and reconfigurable form, respectively. Initially, all LUTs are considered (line 1). Then, the obfuscation engine executes until the desired number of LUTs is made static (line 2), where the SIZE_OF function returns the size of a list. Inside the obfuscation inner loop, the critical path is identified (line 3) using the FIND_CRITICAL function, then the slowest LUT on that path is identified using the FIND_SLOWEST function (line 4). If the identified LUT is a reconfigurable LUT (line 5), the lists of LUTs are updated (lines 6 and 7) and the timing engine recalculates the affected paths (line 8). If the identified LUT is not reconfigurable (line 9), the path is removed (line 10) and the loop continues (line 2). The INSERT and REMOVE functions update the lists as hinted by their names.

A few additional steps take place after the obfuscation criterion has been met (lines 11–17). These steps are already related to the implementation of hASIC, but we list then, here, for completeness. The DECODE function operates on every LUT that was assigned to be static. From Step ③, TOTe already possesses their description in Verilog as truth tables. TOTe then executes the ASIC synthesis of the truth tables to obtain netlists composed of standard cells. The function GEN_CASE_0_1 generates the "force logic" to be used for timing and power analysis during physical synthesis; otherwise, each LUT would be timed for its worst timing arc instead of the actual implemented timing arc when the LUT is programmed. DECOMPOSE_OPT decomposes the larger LUTs into smaller LUTs. Due to the complexity of this operation, we

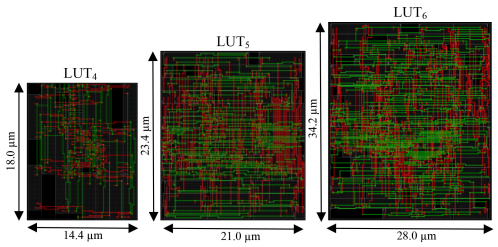


Fig. 3. Layout of macros for LUT_4, LUT_5, and LUT_6. Implementation was executed in Cadence Innovus.

dedicate an entire section to it (see Section III). SWAP_PINS performs a final timing optimization that is an attempt to swap the LUT pins in order to improve the delay, also discussed later in Section III-C. Finally, the algorithm merges L_{ST} and L_{RE} to generate hASIC and returns.

In Step ⑤, area estimation is performed. The estimated area of the hASIC design is calculated as $A = A_{re} + A_{st}$, where A_{re} is the area of the reconfigurable part and A_{st} is the area of the static part. For calculating A_{re}, we sum the area of the LUTs that remain reconfigurable. For calculating A_{st}, we sum the area of the standard cells of the static LUTs. Later, a very precise area estimation is done in an industry-strength physical synthesis tool, where congestion is properly accounted for.

Step ⑥ mostly relates to the generation of files that describe the hASIC intent. This step exports an obfuscated hybrid Verilog file, a timing report, and an area report. A designer can repeat this procedure until he/she achieves his obfuscation (security) and performance targets. Finally, in Step ⑦, the obfuscated netlist is implemented in a commercial physical synthesis tool where traditional P&R, CTS, DRC, etc., steps are executed and the resulting tapeout database is sent to the foundry for fabrication. Once the fabricated parts are delivered, they have to be programmed for the hASIC design to be functional. The programming step requires a bitstream, the same as in an FPGA design.

III. LUT-Specific Approaches to Improve QoR

In this section, we discuss LUT-related optimizations and decisions taken in order to make an hASIC design display the high-performance characteristics of an ASIC and the obfuscation capability of an FPGA fabric.

A. Custom Standard Cell-Based LUTs

We have designed our own custom LUTs (LUT_1, LUT_2, ..., LUT_6) out of *regular standard cells* and by following VPR's template [26]. The layouts for LUT_4, LUT_5, and LUT_6 macros are shown in Fig. 3. Table I shows the average delays and other characteristics of the implemented LUTs.

Commercial FPGAs typically implement only one LUT size, but hASIC provides the flexibility to implement the design with different LUT sizes. This is because the generated hASIC solution is design-specific, meaning that the reconfigurability notion of an FPGA is no longer sought. Moreover, our LUT macros are highly compact, which helps placement to achieve high-density designs. Every single LUT contains a number of flip-flops for storing the configuration bits that serve as a lock for the obfuscated design. Each LUT also makes use of three extra pins (serial_in, serial_out, and enable) to configure these registers. The LUTs are serially connected to one another, forming a daisy chain that is analogous to a scan chain. The choice of a flip-flop-based implementation makes our framework technology-agnostic while making the floorplan and placement almost effortless. Moreover, the LUTs themselves are also treated as regular standard cells during physical synthesis. This allows us to take full advantage of placement algorithms from commercial EDA tools, thus, eliminating the need for any extra custom scripts for placing the LUT macros.

TABLE I
Block Implementation Results for LUT_i

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

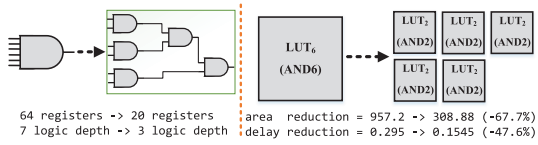


Fig. 4. Logic conversion and decomposition of LUT_6.

B. LUT Decomposition

The area and delay of a LUT are directly related to its number of inputs: the area is mainly bounded by the number of sequential elements used to store the LUT's truth table, whereas the delay is proportional to the LUT's internal MUX tree. However, not all 6-input functions require a LUT_6 to be implemented. For instance, an AND6 can be decomposed in 5 AND2s, as presented in Fig. 4. Referring to Table I, it is clear that the area almost doubles for each input added. Also, the delay vastly increases, where a LUT_6 has almost 6× more average delay than LUT_2. The previous example shows that a LUT_6 decomposition can reduce the area to less than one-third. Moreover, the delay is reduced to approximately half. This example presents a promising approach to improving both area and timing of the circuit.

To decompose our LUTs, we will use functional composition (FC) [27], an approach that can perform bottom-up association of Boolean functions and control the costs in the composition process. Such capability contrasts with traditional top-down functional decomposition, which does not provide a final cost until the decomposition is complete.

1) Functional Composition for LUTs: A summary of the FC paradigm and its application for LUTs will be presented. Readers can obtain more details from [27], [28]. FC is a bottom-up paradigm that has five principles: 1) it uses bonded pairs (BPs) that have a functional part (canonical implementations of a Boolean function, i.e., BDDs or truth tables) and an implementation part (the structure that is being optimized, i.e.,

Algorithm 2: FC-OPT-LUT Algorithm

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

Algorithm 3: FC-HEUR-LUT Algorithm

```
   Input: F (target function), C (cost function)
   Output: IMP
1  ALL_IMP ← φ, B ← φ
2  B.add (CREATE_INITIAL_FUNCTIONS (F, ALL_IMP))
3  IMP ← ALL_IMP(F)
4  if IMP ≠ φ then
5  |  return IMP
6  ALL_COF ← EXTRACT_ALL_COFACTORS (F)
7  foreach cofactor COF ∈ ALL_COF do
8  |  6 COF_IMP ← FC_HEUR_LUT(COF, C)
9  |  ALL_IMP ← [COF, COF_IMP]
10 ALL_IMP ← [F, GET_NAIVE_SOLUTION(F)]
11 COMBINE_COFACTORS(ALL_COF, ALL_IMP)
12 ASSOCIATE_FUNCTIONS (ALL_COF, ALL_IMP, C)
13 IMP ← ALL_IMP(F)
14 return IMP
```

a fanout-free LUT circuit in this article); 2) every BP association performs independently the functional/implementation operations, allowing for more complex implementations with simple functional operations; 3) using partial ordering and dynamic programming, all BPs with the same cost are stored together in a set (bucket), allowing the use of intermediate solutions as subproblems and to perform associations in a cost-increasing fashion; 4) to start any FC algorithm, initial BPs are required, i.e., constants and single input variables; and 5) it allows the heuristic selection of a subset of allowed functions to reduce the composition search space.

2) Exhaustive LUT FC Method: FC can be applied exhaustively, providing fanout-free implementations that have optimal cost. Algorithm 2 generates all minimal LUT fanout-free implementations for functions up to four variables. The algorithm to generate functions with N inputs consists of generating all implementations using LUT($N-1$) with a smaller cost than the LUT(N). Functions with up to two inputs are by definition not decomposable. For functions containing N inputs, a set of functions that will serve as Boolean operators are required. The Boolean operators are the NPN class functions from 2 up to $N-1$ inputs and their negated and permuted variants.

We take area values from the LUT macros' bounding box for the cost functions, and delay values are the average delay of all timing arcs. If a more sophisticated timing analysis is done, some permutations can generate different delays, yielding different results. This difference can be mitigated at the end of the flow by the SWAP_PINS capability later presented in Section III-C.

The FC-OPT-LUT algorithm takes the number of LUT inputs N and the cost function C, which accounts for area and delay. The result is ALL_IMP, a map of functions and LUT implementations. Lines 1–3 initialize the variable B, which is the bucket list containing all functions already implemented, and MAX_COST, which will provide the single LUT-N cost. Also, B is initialized with constants and single variables through the method CREATE_INITIAL_FUNCTIONS. In line 4, the association of tuples and arrival time (AT) is computed, which consists of 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 $B[i-1]$. Still, at the same time, it is the smaller one of all possibilities. As an example, if the candidate ATs have cost 14, 10, 10, and 12 and $B[i-1]$ cost is 9, the candidate ATs with cost 10 will be selected.

The while loop in lines 5–8 checks, at each iteration, if the cost of AT is not higher than MAX_COST. In such cases, it is better to use the naive solution. If the cost is smaller, the method ASSOCIATE will process the list of AT, combining them 2 by 2 or 3 by 3 (depending on the tuple size), using the cost function for breaking ties. The result is added to the bucket list. A new list of ATs is computed, which will continue or break the loop. Finally, in lines 9 and 10, if there are remaining functions, they are considered not decomposable (i.e., the cost to decompose them is higher than the naive solution). The method CREATE_NAIVE_IMPS will look for Boolean functions that do not have an implementation on ALL_IMP and add a naive one, guaranteeing that all Boolean functions of up to N inputs are present on the ALL_IMP map, returned in line 11.

3) Heuristic LUT FC Method: The method described in the previous section can only generate optimal LUTs of up to four inputs. A heuristic is required to deal with more complex LUTs. A LUT decomposition can be thought of as a factorization problem, where there is a direct conversion from a factored form to a LUT tree (i.e., a fanout-free) structure. With some modifications, we can apply the Boolean factoring method presented in [28] to perform decompositions. This approach is called FC-HEUR-LUT. These modifications are necessary to derive LUT decompositions that can have a better cost than the naive solution.

FC-HEUR-LUT is presented in Algorithm 3. The algorithm takes the target function F and the cost function C. The result is the LUT implementation IMP, which is the LUT circuit containing the decomposed naive solution. Lines 1–3 initialize the variables ALL_IMP, which represents a map storing all known implementations for the functions already decomposed, and B, which contains the buckets. The method CREATE_INITIAL_FUNCTIONS remains the same as in FC_OPT_LUT. Lines 4 and 5 check if it is a trivial case and returns if so. In line 6, the method EXTRACT_ALL_COFACTORS is executed. This method computes all the cofactors and cubecofactors (excluding constants) from F and stores them in the ALL_COF set.

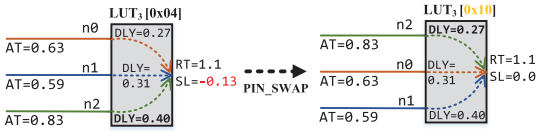


Fig. 5. Example of a beneficial pin swap.

Lines 7–9 are a recursive call to the algorithm, providing a LUT implementation to all cofactors and cubecofactors. With the cofactors and cubecofactors derived, the combination of cofactors takes place in line 11, using the same strategies presented to expand the "allowed functions" set, as explained in [28]. This expansion guarantees at least 2 factored subfunctions that, when associated in the next step, will provide at least one functionally equivalent solution. 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. If the association is functionally equivalent to F, 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 and 14 will collect the resulting implementation IMP and return.

Some techniques applied to greatly speed-up FC-HEUR-LUT include using FC-OPT-LUT results to aid FC-HEUR-LUT. The ALL_IMP map is used at the beginning of the algorithm to quickly return the optimal implementation if the function F has a support of 4 or fewer inputs while also improving the decomposition quality of results (QoR). Another technique applied is the limit on the number of associations of not comparable Boolean functions because those can be a considerable number (i.e., more than 100 thousand). This limit avoids substantial runtime trying to decompose more complex functions, which generally have worse costs when decomposed.

C. Pin Swap Approach

The method SWAP_PINS takes advantage of the fact that a LUT function can have an arbitrary input pin swap if the truth table is permuted accordingly. So, our method takes a LUT function and timing information as input and provides the permuted truth table and the new order of input pins/nets. The example presented in Fig. 5 shows an effective pin swap that improved the slack of the design. The pin swap algorithm takes the LUT function, the AT of each input net (termed [$n0$, $n1$, $n2$]), the cell arc delay (DLY) associated with each input, and the required time (RT) at the output. In the example, RT = 1.1, and the critical arc is $n2$, with a total delay of 1.23. The algorithm initially tries all the input permutations, trying to minimize WNS. If two or more arcs have negative slack, it also tries to reduce TNS. Once all permutations are tried, and a new order improves WNS and/or TNS, the truth table is permuted accordingly to keep the same functionality. The algorithm returns the truth table $0x10$ and the new net order [$n2$, $n0$, $n1$].

IV. Experimental Results Using TOTe

Without loss of generality, for all experimental results, we have executed FPGA synthesis in Vivado and the target is a Kintex-7 XC7K325T-2FFG900C device which contains 6-input LUTs [29]. Following, Cadence Genus is used for the logic synthesis with three flavors of a commercial 65-nm standard cell library (LVT/SVT/HVT). However, we emphasize that TOTe is *agnostic with respect to PDKs, libraries, and tools*.

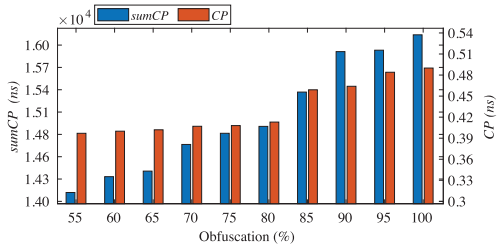


Fig. 6. TOTe's obfuscation versus performance for SBM.

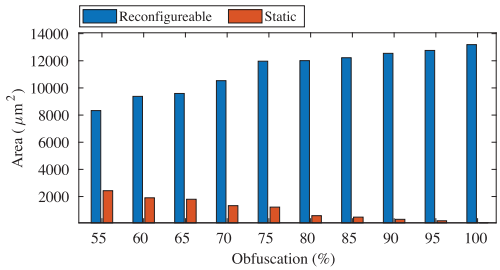


Fig. 7. TOTe's obfuscation versus area for SBM.

For our first experiment, we considered a small but pragmatic design that covers all possible FPGA primitives. We selected a schoolbook multiplier (SBM) design as DUO [30]. We obfuscated an 8-bit SBM by varying obf_c from 55% to 100% and evaluated the obfuscation versus performance and obfuscation versus area trends. We have synthesized the SBM design targeting a challenging frequency of 540 MHz. As calculated by TOTe's timing engine, the CP and sumCP values become 0.490 and 16088.69 ns, respectively. These values correspond to a design obfuscated at 100%, i.e., all LUTs are reconfigurable. At this stage, these values represent a simplistic timing analysis, realistic timing values will be obtained when the final timing analysis is performed using a commercial physical synthesis tool. However, as we move along with the obfuscation process, CP and sumCP remain consistent in relative terms, which is sufficient to generally determine critical paths to target.

After performing the obfuscation for different levels, the timing characteristics for the 8-bit SBM are illustrated in Fig. 6. The analysis of CP and sumCP shows that performance is decreasing as we increase the level of obfuscation. Conversely, decreasing the level of obfuscation increases the performance of the design. The trend depicted in Fig. 6 is that CP improves inversely with the obfuscation, but it is saturated when the obfuscation is below 80%. This fact is not true for sumCP, the decrease in obfuscation causes continuous improvement as expected. Similarly, the performance versus area profile of the 8-bit SBM is illustrated in Fig. 7.

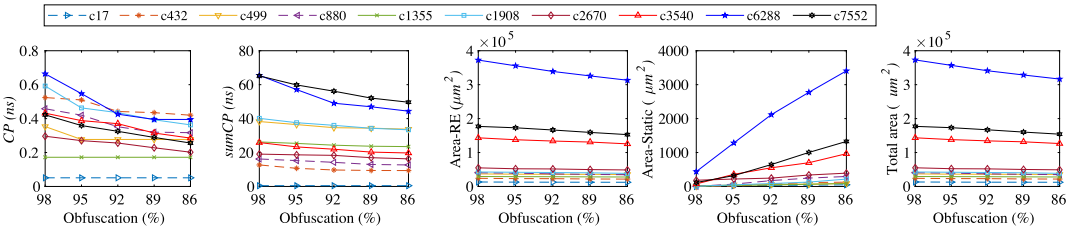


Fig. 8. Obfuscation results for ISCAS'85 benchmarks using TOTe.

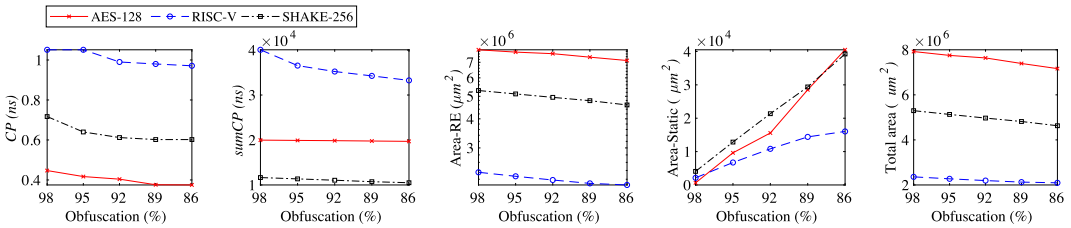


Fig. 9. Obfuscation results for AES-128, RISC-V, and SHAKE-256 using TOTe.

Next, we investigate whether the same saturation would appear for other designs. We first selected the ISCAS'85 benchmarks and the results are depicted in Fig. 8. These relatively outdated combinational benchmarks were selected for the reason that they have a single stage of logic, so the CP and sumCP correlation is easy to follow (i.e., the critical path does not change from different reg2reg paths). Even in these simplistic designs, saturation occurs remarkably fast.

Naturally, we have also obfuscated more representative designs. In [25], detailed results are provided for SBM, SHA-256, and FPU [31] designs. For the sake of brevity, we do not repeat those results here. Additional results are also provided in graphical form in Fig. 9 for AES, RISC-V, and SHAKE-256 designs so the trends are easy to visualize.

In summary, the results presented in this section confirm that TOTe is a generic tool for obfuscation and it can obfuscate a design regardless of its complexity. It also becomes clear that the reliance on a LUT-based representation of the circuit, akin to an FPGA, has different implications for delay and area. For area, the trend is clear: the lesser is the obfuscation target, the more compact the circuit becomes. However, for delay, it appears that hASIC brings performance penalties that cannot be overcome by simply reducing the targeted obfuscation level. Therefore, other strategies are needed for achieving better performance. In the next section, we will present a more detailed analysis of physical synthesis. We will also apply the optimization methods described in Section III for improving performance.

V. Physical Synthesis for hASIC

This section contains the physical implementation results for an obfuscated SHA-256 [33] design. Cadence Innovus is utilized for physical synthesis, together with a commercial 65-nm PDK. We have selected SHA-256 as it is popular and widely used in cryptography. The variants of the design with different obfuscation levels are implemented with the aid of the LUTs defined in Section III-A. The results obtained after implementation are focused on performance versus area tradeoffs for the 80%–100% obfuscation range, thus, avoiding the saturation trend highlighted in Section IV.

Initially, we synthesized and implemented SHA-256 on FPGA, the target device being a Kintex-7. The FPGA implementation achieves a frequency of only 77 MHz (for reference, the Kintex-7 family is produced on a 28-nm CMOS technology). To start the analysis, we select 100% obfuscation as a baseline design because it is fully reconfigurable and somewhat analogous to an FPGA design. The implementation results for 80%, 85%, 90%, and 100% obfuscation are given in Table II. The timing results are obtained after physical synthesis and are for the worst process corner (SS), VDD = 0.9*$\text{VDD}_{\text{nominal}}$, and a temperature of 125 °C.

From the results, it is clear that the level of obfuscation does not affect the utilization density of the design (i.e., the ratio of placement sites that are occupied versus empty). For all designs, we achieved around 80% utilization density, which is very high considering a large number of macros. In other words, our macros do not compromise global routing resources. It is noteworthy that the performance of TOTe-generated designs is increasing as we decrease the level of obfuscation; our baseline hASIC design is running at 223 MHz (as shown in Freq. column of Table II) and it increases as obf_c decreases. This behavior matches the goal we set from the start: to establish a tradeoff between performance (ASIC) and security (FPGA).

The area of the design is proportional to the obfuscation level, which means that increasing the security of the design comes with an area penalty. As we only exploit LUT primitives for promoting obfuscation, the number of LUTs increases with the obfuscation level. In the same manner, leakage and dynamic power figures are proportional to security as reconfigurable logic is less efficient than static. This is mainly because

TABLE II
RESULTS FOR THE IMPLEMENTATION OF SHA-256 FOR DIFFERENT OBFUSCATION LEVELS

CAD flow	Obf.	Density	Area (μm^2)	Freq. (MHz)	Leakage (mW)	Dynamic Power (mW)	# LUT	# Buffer	# Comb.	# Inv.	# Sequential	Total Wirelength (μm)
FPGA	100%	–	–	77	2.4	191	2238	–	–	–	1830*	–
TOTe	100%	81%	1751500	223	14.85	505.05	2238	5846	93470	6175	105128	9247654
TOTe	90%	77%	1638500	234	12.23	438.47	2015	4626	84107	5017	94876	7505590
TOTe	85%	80%	1507000	241	12.10	430.98	1904	4846	80304	5585	90420	7207023
TOTe	80%	80%	1409700	248	11.05	386.89	1792	4406	75083	4564	83790	6724434
ASIC	NONE	92%	34208	248	0.18	9.37	–	167	3244	190	1806	158003
ASIC	NONE	91%	40804	550	0.299	23.86	–	675	7981	1456	1806	181441

* Vivado performs flip-flop cloning for solving high fanout buffering. Thus, there is an increase in the number of registers w.r.t. ASIC.

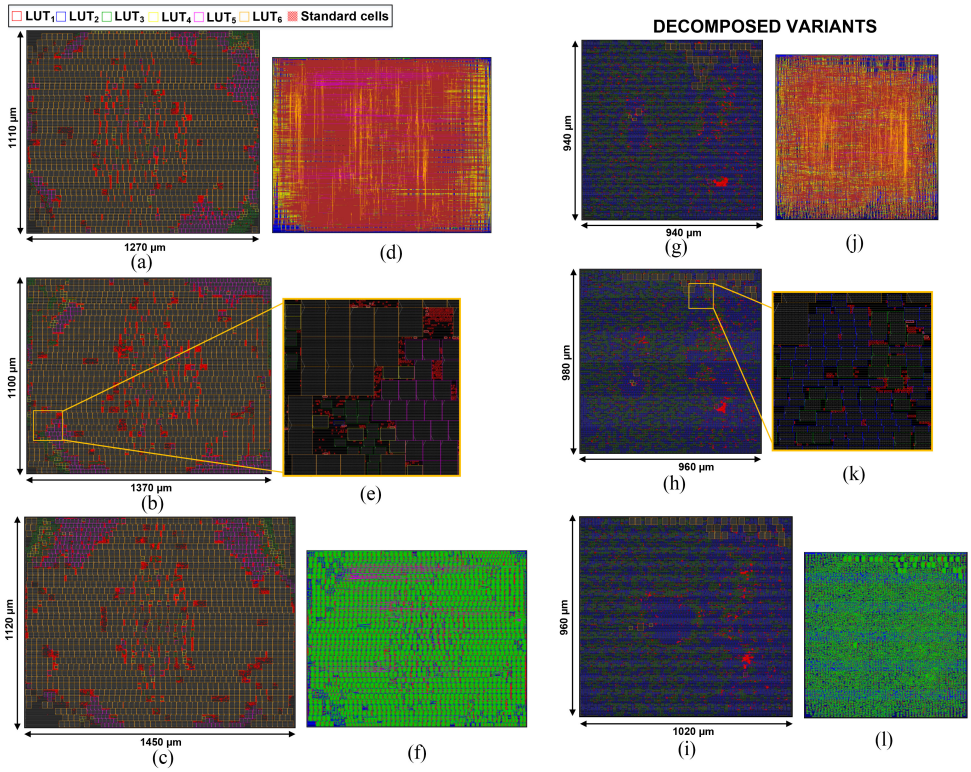


Fig. 10. Implementation results for SHA-256 with different obfuscation levels. (a) SHA-256 with 80% obfuscation. (b) SHA-256 with 85% obfuscation. (c) SHA-256 with 90% obfuscation. (d) SHA-256 with 85% obfuscation (routed layout). (e) SHA-256 with 85% obfuscation (magnified view). (f) SHA-256 with 85% obfuscation (routed and assembled layout). (g) SHA-256 with 80% obfuscation. (h) SHA-256 with 85% obfuscation. (i) SHA-256 with 90% obfuscation. (j) SHA-256 with 85% obfuscation (routed layout). (k) SHA-256 with 85% obfuscation (magnified view). (l) SHA-256 with 85% obfuscation (routed and assembled layout).

of the use of flip-flops to store the LUT truth tables. The last five columns of Table II show the resource requirements for hASIC (number of buffers, combinational cells, inverters, sequential cells, and the total wirelength).

In Fig. 10, we show many different views of the SHA-256 layouts under different obfuscation targets. The considered metal stack has seven metals assigned to signal routing. Panels (a)–(c) of Fig. 10 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. All six variants of LUTs are highlighted with different colors and the static part of hASIC is highlighted in red—notice that, as expected, the design remains primarily a sea of LUTs. The majority of those LUTs are LUT_6, thus, the layouts appear to be dominated by yellow boxes.

Panel (d) of the same figure demonstrates the final post-route layout of hASIC. Notice how the post route design contains mostly vertical orange lines which correspond to M6. Panel

TABLE III
RESULTS FOR THE IMPLEMENTATION OF SHA-256 FOR DIFFERENT OBFUSCATION LEVELS WITH DECOMPOSED LUTS

CAD flow	Obf.	Density	Area (μm^2)	Freq. (MHz)	Leakage (mW)	Dynamic Power (mW)	#LUT	#Buf.	#Comb.	# Inv.	#Seq.	Total Wirelength (μm)
TOTe	100%	61%	1155000	307	8.00	301.49	10182	3583	29352	15261	53868	3391742
TOTe	90%	65%	979200	312	7.55	273.54	9127	1797	27115	13538	49016	3242970
TOTe	85%	67%	940800	322	7.03	256.36	8676	1882	26011	13136	46796	2982627
TOTe	80%	64%	883600	357	6.44	278.37	8124	1726	24614	12340	43830	2889253
TOTe (Swap)	80%	64%	883600	368	5.93	283.35	8124	1726	24614	12340	43830	2889760

(e) of Fig. 10 shows the magnified view of the placement in an hASIC design. The mixed structure of LUT macros and standard cells clearly depicts the placement pattern and the spacing between the macros is usually filled with standard cells. Notice how the LUT macros align with the standard cell rows, allowing for the entire design to have a uniform power rail and power stripe configuration. In panel (f) of the same figure, we illustrate the same design but filter out some routing layers (only M2, M3, and M4 are shown). As depicted in Fig. 3, the implemented LUTs utilize the aforementioned metal layers, therefore, the assembled view of a panel (f) represents how visually regular the hASIC structure is.

In the results shown in panels (g)–(l) of Fig. 10, we have utilized the same design and conditions but applied LUT decomposition for improving performance. Notice that the layouts are drawn to scale to highlight the area reduction brought by decomposition. In particular, when comparing panels (f) and (l), we note that regularity is still present even after decomposition, as expected.

The detailed results for these designs are listed in Table III. With decomposition, the baseline frequency increased significantly, from 223 to 307 MHz. As in the nondecomposed version, the performance increases inversely with the obfuscation level. On top of that, the area was reduced by more than half, along with the power consumption. However, due to a large number of small LUTs (mostly LUT_2s), placing and routing become slightly more challenging. For this reason, the maximum utilization density across the optimized designs is approximately 65%. Nevertheless, decomposition is very beneficial: the gain in PPA when compared with the nonoptimized versions is significant. We argue that since decomposition has a negligible impact on the runtime of the physical synthesis flow, it should always be applied. For instance, the runtime to apply the decompositions in the SHA-256 circuit with 100% obfuscation containing 2238 LUTs was 11 min in an Intel Core i7-6700K. The decomposition achieved improvements of 50% in the LUT area and 32% in the total LUT delay.

While the LUT decomposition brings significant performance improvement, we seek to achieve performance levels that are as close to the ASIC implementation as possible. For that reason, after the decomposition, we also applied the pin swapping technique. As we noted earlier, this is possible because the same logic function can be generated with different input orders and different masking bits (truth table). Therefore, we can search for LUTs that appear on the critical path(s) and swap their pins to reduce the total delay.

For illustrating the capability of pin swapping, we first create an artificial scenario where we increase the target frequency of the design until several paths violate setup timing. The frequency increase determines the number of violating paths that indirectly determines the number of LUTs that will be considered for pin swap purposes.[2] All LUTs from violating paths are chosen as candidates and saved in a list. Then, iteratively, starting with the worst violating path, the pins of the LUTs are swapped until the critical path is improved (i.e., WNS). The number of swaps performed versus the TNS and WNS is illustrated in Fig. 11. From this figure, the first few swaps improved the WNS while the TNS remain the same. The continuing swapping starts to improve the TNS without any change in the WNS. If the TNS is improving, that means there is a chance to reach a better WNS. Thus, we performed the swapping until the next jump in the WNS. After attempting 200 swaps, WNS improved by approximately 80 ps and TNS by 2 ns, thus, making the design 11 MHz faster.

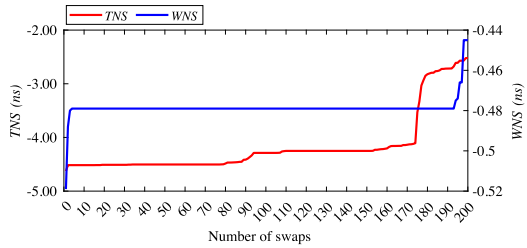


Fig. 11. Change in the TNS/WNS with respect to the swap of LUT pins.

VI. SECURITY ANALYSIS

A. Threat Model

In our considered threat model, the primary adversary is the *untrusted foundry*. We make no distinction whether the adversary is institutional or a rogue employee. Assuming the security of an hASIC design is a function of its static logic (fully exposed) and reconfigurable logic (protected by a bitstream that serves as a key), we make the following assumptions.

1) The main adversary goal is to reverse engineer the design in order to pirate its IPs, overproduce the IC, or even insert sophisticated hardware trojans. For this goal, the adversary *must* recreate the bitstream.
2) The adversary goal might also be to identify the circuit intent, even in the presence of obfuscation. For this goal, the adversary *does not need* to recreate the bitstream.

[2]Here, we establish a runtime versus QoR tradeoff. The more aggressive the frequency target is, the more LUTs are considered for pin swap.

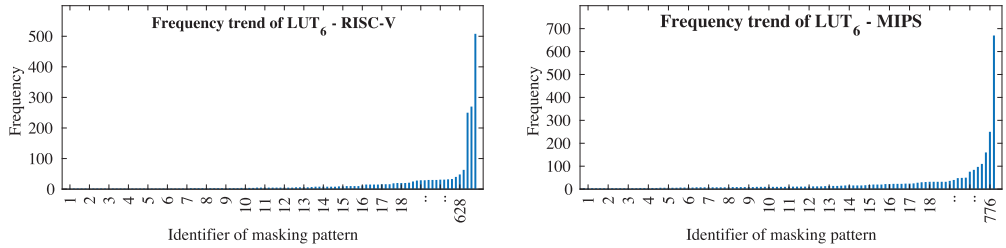


Fig. 12. Frequency of masking patterns for RISC-V and MIPS.

3) The adversary has access to the GDSII file of the hASIC design sent for fabrication. The adversary is skilled in IC design and has the knowledge and tools required for understanding this layout representation.
4) The adversary can recognize the standard cells, therefore, the gate-level netlist of the obfuscated circuit can be easily recovered [35].
5) The adversary can identify reconfiguration pins [36], [37], thus, being able to effortlessly enumerate all LUTs and their programming order.
6) The adversary can group the standard cells present in the static logic and convert them back into a LUT representation.[3]

We have proposed two different attacks to evaluate the security hardness of hASIC: one based on the *structure* of design and another based on the *composition* of known different circuits. We assert that an adversary can learn and extract information by exploiting the static portion of the design, including the frequency of specific masking patterns. This capability would allow an adversary to shrink the search space for the key that unlocks the design.

Similarly, the notion of masking pattern frequency can be utilized as a template to compare different designs. In other words, the composition of the LUTs in a design would allow for a template-based attack. Moreover, we have also evaluated the security hardness of hASIC for conventional oracle-guided and oracle-less attacks borrowed from logic-locking attacks. All the experiments reported in this section were run on a server equipped with 32 processors (Intel Xeon Platinum 8356H CPU @ 3.90 GHz) with 1.48 TB of RAM.

B. Structural Analysis Attack

Goal: By statistical analysis means, decrease the key search space before attempting to recover the bitstream.

We recall again that TOTe's obfuscation engine utilizes six variants of LUTs. However, the majority of the LUTs are LUT_6 due to the packing algorithm executed during FPGA implementation. The decomposition only applies to the reconfigurable part of the design. The initial knowledge that the adversary acquires is from the static part, which remains unchanged whether decomposition is used or not used. The static part is composed mostly of LUT_6, so the adversary ought to keep his/her analysis geared at LUT_6 too. Therefore, we consider this scenario and present our analysis of it.

For a LUT_6, the possible number of keys is 2^{64}. But this number is only realistic if the FPGA synthesis tool is genuinely able to exercise the entire key search space. This does not appear to be true: We have synthesized a considerable number of representative designs (>30) and extracted all unique LUT_6 masking patterns from the corresponding netlists. We term these values m_i. We considered designs of varied size, complexity, and functionality until the combined number of unique masking patterns forms a set of $M = \sum m_i = 3376$ elements that appear to settle. This result alone, albeit being empirical, reduces the global search space from 2^{64} to $3376 = 2^{11.72}$.

With this information at hand, we hypothesize that an attacker can exploit the frequency at which LUTs appear in a netlist in order to mount attacks, thus, the name structural analysis attack. In other words, the adversary is interested in finding the values of m_i for a given circuit C_i. However, the adversary only has partial knowledge of the design. The question then becomes whether the adversary can estimate m_i by performing statistical analysis on a portion of C_i. To this end, we targeted two processor designs in our statistical analysis: MIPS and RISC-V. For each circuit, we utilize tuples of ⟨pattern, frequency⟩ for tracking how often masking patterns repeat. The pattern is a 64-bit hexadecimal number. The tuples are referenced by integer identifiers and ordered by frequency as shown on the bar charts in Fig. 12. Notice that the MIPS netlist has 776 unique LUTs and that there are very few outliers that occur more than 50 times. Similarly, for RISC-V, there are 628 unique LUTs and only 3 occur more than 100 times.

We investigate this by analyzing the behavior of the frequency of masking patterns as depicted in Fig. 13. For this, we utilized netlists generated by TOTe at 98%, 95%, 92%, 89%, and 86% obfuscation levels. Therefore, for this experiment, we assume the attacker only has visibility over 2%, 5%, 8%, 11%, and 14% of the LUTs, respectively. The adversary then attempts to predict the distribution of actual masking patterns in the design from his/her observation of the small percentage of LUTs that are exposed in the static portion of hASIC. In Fig. 13, the adversary's guessing attempt is performed with the aid of polynomial trendlines. For MIPS and RISC-V, it appears that the adversary can estimate to some degree what masking patterns are the outliers. The actual number of unique patterns, m_i, is not trivial to determine from extrapolation since many patterns appear a single time or very few times (see Fig. 12). We clarify that several circuits studied in this article (e.g., PID, IIR, GPU, SHA-256, etc.) have a

[3]This is a very generous concession since the static logic is repeatedly optimized during synthesis. Nevertheless, we assume the adversary can achieve a perfect reconstruction of LUTs.

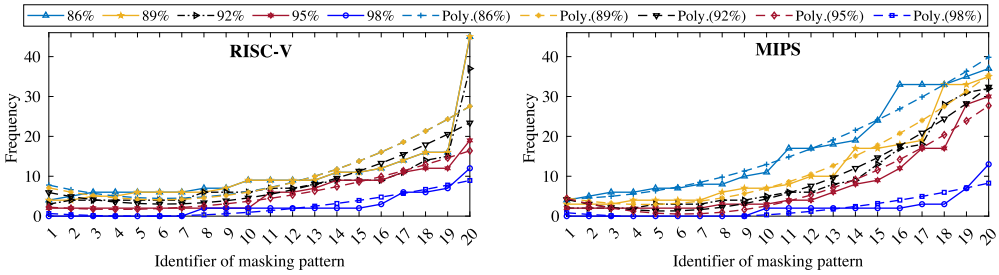


Fig. 13. Structural analysis of RISC-V and MIPS.

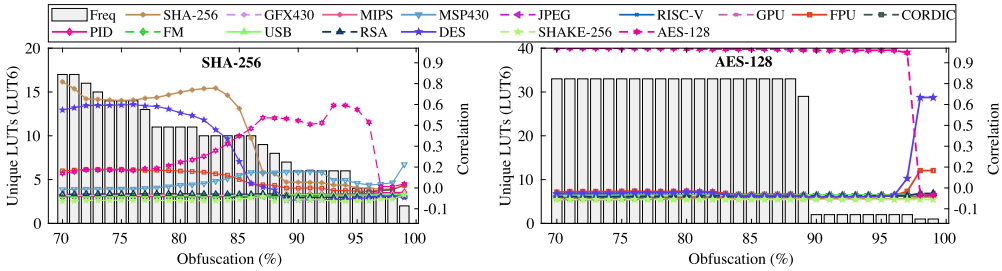


Fig. 14. Correlation of SHA-256 and AES-128 versus numerous other designs.

similar profile, where only a handful of high-frequency LUTs appear. As stated earlier, the attack exploits only the static part. But, when the decomposition is applied the adversary needs an in-depth knowledge of the decomposition algorithm to estimate the frequency of outliers for the reconfigurable part. Therefore, it remains to be studied if the knowledge gathered from this attack can be useful for some form of hill climbing attack (or even a biased version of SAT).

C. Composition Analysis Attack

Goal: identify the circuit by correlation to known circuits.

This attack also exploits the frequency of the LUTs, but, here, we correlate several designs against each other based on their composition, thus, the name. We suppose that the attack is already successful if the adversary is able to identify the circuit (i.e., breaking the key is not necessary).

In the experiment depicted in Fig. 14, we carried out correlation analysis for two different crypto cores: 1) SHA-256 and 2) AES-128. The goal of this analysis is to examine the leaked information from the static part against a database[4] of circuits that are known to the attacker. We perform obfuscation of SHA-256 and AES-128 in the 70%–100% range and then correlate their static portions with the designs in the database. The x-axis of Fig. 14 is the obfuscation level, and the y-axis is the number of unique LUTs (left) and Pearson correlation (right).

The correlation results reveal very interesting trends. For SHA-256, three regions of interest can be defined based on the degree of obfuscation: 97%–100% (no correlation), 86%–96% (strong correlation to another circuit), and 70%–85% (correlation to itself). A similar analysis has been performed for AES-128 as shown in the right side of Fig. 14. The correlation between obfuscated AES-128 versus AES-128 is almost one for obfuscation $< 97\%$. Conversely, the correlation for obfuscated AES-128 versus other designs is almost zero obfuscation in the same range ($< 97\%$).

Assuming that the adversary's objective is exclusively to recognize the circuit's intent ("what is this circuit?"), hASIC could prove as vulnerable as an ASIC design. This is the case for the AES-128 circuit, while the SHA-256 case reveals a contrasting trend: there are obfuscation ranges that can be targeted on purpose to confuse an adversary. For SHA-256, this range appears to be 86%–96%.

Finally, for an adversary that is interested in obtaining the bitstream, we hypothesize that the correlation analysis herein depicted might be useful to shrink the key search space further. In practice, if the attacker could know for a fact that the obfuscated circuits are indeed AES, or SHA-256, or C_i, his key guessing would be based on the m_i of the circuit with the highest correlation. It is noteworthy to mention that this attack leverages the design attributes and the frequency of LUTs derived from the static part of the design. Hence, the attack's success or failure depends upon three key points: First, the adversary's ability to reconstruct the LUTs from the static part; Second, the availability of enough datapoints in the database of known circuits; Third, the design has static parts, i.e., the obfuscation level is below 100%. Referring again to the example from the previous section, the search space would shrink further; from 3376 to 776 for MIPS and from 3376 to 628 for RISC-V. As stated in Section VI-B, this attack also exploits only the static part. Therefore, identifying the design has no relationship with decomposition, i.e., a decomposed design is not easier (or harder) to correlate. From this point

[4]We assume that the adversary can obtain samples of open-source cores from repositories and execute FPGA synthesis on them to create a database.

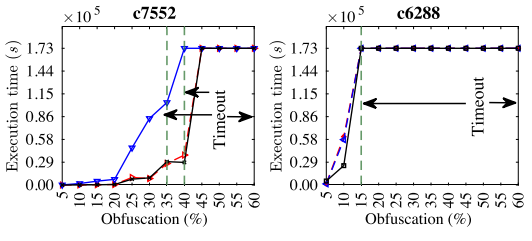


Fig. 15. Execution time of SAT attacks for two different designs.

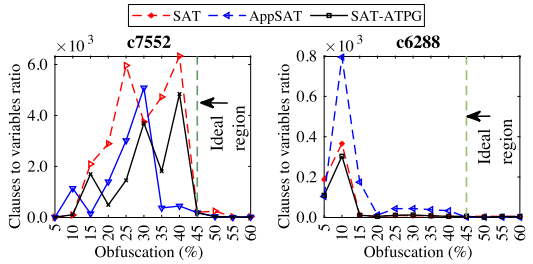


Fig. 16. Variables to clauses ratio of SAT attacks for two different designs.

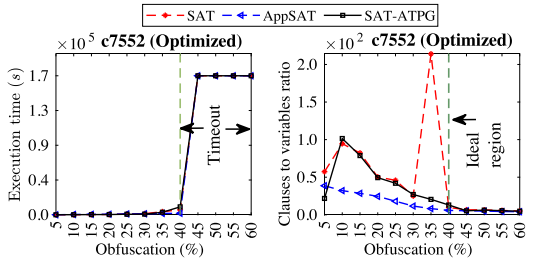


Fig. 17. Optimized results for c7552 with regards to the execution time and the ratio of variables to clauses in SAT attacks.

onward, in order to obtain the actual key, an adversary would still have to resort to other attacks (not specific to hASIC). We discuss such attacks in the text that follows.

D. Oracle-Guided Attacks

Goal: To retrieve a key or a key guess.

As compared to conventional logic locking [38], the LUTs introduced in hASIC are the elements that serve as key gates. A LUT_6, in theory, introduces 64 bits of key, akin to 64 XOR/XNOR gates in conventional logic locking. The very first circuit we introduced in Section IV, the SBM, has 25 LUTs (out of which 11 are LUT_6) when its obfuscation rate is 86%. In turn, the key search space would be $2^{11\times64}$ for LUT_6 alone. Such a large search space would discourage an adversary from performing SAT attacks on hASIC.

However, enumerating the key search space is a very simplistic/naive approach. One has to perform actual attacks in order to evaluate the security of the designs, especially, well-known satisfiability-based attacks. We have, therefore, considered three different SAT attacks to evaluate the security hardness of hASIC: 1) Conventional SAT [36]; 2) AppSAT [39]; and 3) ATPG-based SAT [40].

We have selected large combinational circuits (c7552 and c6288) from the ISCAS'85 suite to evaluate the security hardness of hASIC against SAT-based attacks. Importantly, we present the results for two different variants of hASIC, optimized and nonoptimized, for the selected designs. Concerning the nonoptimized variant, Fig. 15 illustrates the execution time for different SAT attacks and different obfuscation rates, where the x-axis is the obfuscation level and the y-axis is the execution time. As expected, the execution time increases as we increase the obfuscation level. The region to the left of the green line shows successful SAT attacks. However, the region on the right corresponds to unsuccessful attacks, where timeout (48 h) was achieved before the solver returned an answer. In principle, this is an encouraging result since even a very small and combinational-only circuit like c6288 leads to timeouts at relatively low obfuscation rates ∼15%.

More intriguingly, another combinational-only circuit c7552 led to timeout after reaching 40% obfuscation rate. Thus, the obfuscation rate where the SAT solver returns a timeout varies in different designs. Additional statistics about the behavior of the SAT solver for the obfuscated circuits are given in Fig. 16. The SAT solver determines when a Boolean formula is satisfiable or not. One approach to measuring the chance of convergence of the attack is to measure the ratio of clauses to variables of the SAT solver. With the help of this ratio, an obfuscated design can be labeled SAT-hard if the ratio is around 4.2 [41]. Fig. 16 shows the evolution of the number of clauses to variables with respect to the obfuscation level. The x-axis is the obfuscation level (%) and the y-axis shows the ratio of clauses to variables. We label the right region of Fig. 16 as "Ideal region" because the clauses to variables ratio are near the ideal value of 4.2.

TOTe automatically optimizes designs by decomposing LUTs into smaller LUTs. While power, area, and performance are improved, the decomposition reduces the size of the key for unlocking the design. We must, therefore, verify that this reduction does not make the hASIC designs vulnerable to existing attacks. For the optimized version of c7552 in Fig. 17, we can see that there is a reduction in the time that it takes for the successful attacks to complete. However, none of the attacks is successful beyond 40% obfuscation. Further details for c7552 are given in Table IV where we also list the key sizes for different designs and two obfuscation rates, namely, 55% and 60%. Note that, counter-intuitively, the decomposed designs have better variables-to-clause ratios. Our interpretation is that decomposition keeps keys that are less correlated to one another, thus, each individual key bit is more effective. Readers are directed to [41] for details on the SAT attack and to [40] for a discussion on key interference.

E. Oracle-Less Attacks

Goal: To retrieve a key or a key guess.

Oracle-less attacks do not require an oracle (i.e., a functional IC). Instead, they operate directly on the netlist of the obfuscated circuit. One of such attack is the synthesis-based constant propagation attack on logic locking (SCOPE) [42].

TABLE IV
ANALYSIS OF VARIABLES-TO-CLAUSES RATIO FOR $obf_c = 55\%$ AND 60%

Attack	Obf. (%)	*Key length* (bits)	*Variables*	Clauses	Iterations	Ratio
SAT [36]	55	7494	32318070	1597559	820	20.8
	60	8582	33315150	1898618	540	17.5
	55∗	4014	3434578	598599	139	5.7
	60∗	4406	2829008	564533	106	5.0
AppSAT [39]	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 [40]	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

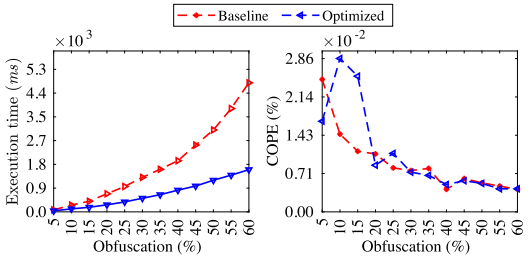


Fig. 18. Comparison between the baseline and optimized design for the oracle-less SCOPE attack.

This attack does not require any knowledge about the locking technique or the obfuscated design. SCOPE performs a synthesis-based analysis on a single key-input port and extracts important design features that may assist to derive the correct key bits. Fig. 18 illustrates the comparison of execution time, COPE metric for the baseline, and optimized design of the c7552 design. It is clear from the left panel that the execution time is exponentially increasing with the obfuscation level. Similar trends are seen for the baseline and optimized design, both are exponential but present different rates. The right panel of Fig. 18 shows the COPE metric, which decreases with the obfuscation level. The details for the calculation of COPE metric are available in [42]. For simplicity, we clarify that the COPE metric is a rough estimate of the level of vulnerability (%) to the SCOPE attack.

When SCOPE concludes, a key guess is produced. For each bit of the key, SCOPE assigns either a "1," a "0," or an "X" (undetermined). When matching the guess from SCOPE with our known key, the result is that about 50% of the key bits are correctly guessed. This percentage is not related to the obfuscation level, the result is always the same for baseline and optimized designs. In other words, SCOPE cannot perform better than a random guess for hASIC.

VII. DISCUSSION

A recent trend in obfuscation research is the use of embedded FPGA (eFPGA) [17], [18]. While there are advantages to this practice, it has been used selectively to only protect key portions of a design and, therefore, keep the performance penalty as low as possible. The challenge is in determining which portions/modules of the circuit merit protection and which ones do not. Our hASIC approach bypasses this question almost completely by only revealing (portions of) critical paths when they are selected to become static logic, which we consider an advantage. Mohan et al. [44] presented a top-down methodology to implement ASICs with eFPGAs. Their designs share many of the advantages of our hASIC solution while presenting more regularity than our designs (they make use of logic tiles as in commercial FPGAs). Our tile-free design trades this regularity for performance as evidenced by the layout in Fig. 10 and the corresponding results in Table II.

VIII. CONCLUSION

The main finding of our work is that an hASIC solution contrasts with the current practice of eFPGA obfuscation; our experimental results illustrate that obfuscation rates have to be high to secure the design's intent. To this end, we have presented a custom tool that obfuscates a design and generates an hASIC block. Our LUT decomposition, along with the pin swapping, improves the performance and reduces the area of hASIC designs. We have also validated our results in a commercial physical synthesis tool with industry-strength timing and power analysis. Our security analysis, anchored by the results from diverse attacks, confirms that obfuscation rates should be high.

REFERENCES

[1] "2022 semiconductor sales to grow 11% after surging 25% in 2021." IC Insights. 2022. [Online]. Available: https://www.icinsights.com/data/articles/documents/1424.pdf
[2] "Apple's M3 chips on track for 2023 as next-gen 3nm process begins." Mac World. 2021. [Online]. Available: https://www.macworld.com/article/557232/m3-chips-apple-devices-tsmc-3nm-process-2023.html
[3] "TSMC to kick off mass production of Intel CPUs in 2H21 as Intel shifts its CPU manufacturing strategies, says TrendForce." TrendForce. 2021. [Online]. Available: https://www.trendforce.com/presscenter/news/20210113-10651.html
[4] "Big trouble at 3nm." Semiconductor Engineering. 2018. [Online]. Available: https://semiengineering.com/big-trouble-at-3nm/
[5] M. Rostami, F. Koushanfar, and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proc. IEEE*, vol. 102, no. 8, pp. 1283–1295, Aug. 2014.
[6] K. Z. Azar, H. M. Kamali, H. Homayoun, and A. Sasan, "Threats on logic locking: A decade later," in *Proc. Great Lakes Symp. VLSI*, 2019, pp. 471–476.
[7] Y. Xie and A. Srivastava, "Delay locking: Security enhancement of logic locking against IC counterfeiting and overproduction," in *Proc. 54th ACM/EDAC/IEEE Des. Autom. Conf. (DAC)*, 2017, pp. 1–6.
[8] M. Yasin, J. Rajendran, and O. Sinanoglu, *Trustworthy Hardware Design: Combinational Logic Locking Techniques*. Cham, Switzerland: Springer, 2019.
[9] M. Yasin and O. Sinanoglu, "Evolution of logic locking," in *Proc. IFIP/IEEE Int. Conf. Very Large Scale Integr. (VLSI-SoC)*, 2017, pp. 1–6.
[10] J. Sweeney, V. M. Zackriya, S. Pagliarini, and L. Pileggi, "Latch-based logic locking," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust (HOST)*, 2020, pp. 132–141.
[11] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran, "Removal attacks on logic locking and camouflaging techniques," *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 2, pp. 517–532, Apr.–Jun. 2020.
[12] R. P. Cocchi, J. P. Baukus, L. W. Chow, and B. J. Wang, "Circuit camouflage integration for hardware IP protection," in *Proc. 51st ACM/EDAC/IEEE Des. Autom. Conf. (DAC)*, 2014, pp. 1–5.
[13] M. Li et al., "Provably secure camouflaging strategy for IC protection," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 38, no. 8, pp. 1399–1412, Aug. 2019.

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

[15] J. Rajendran, O. Sinanoglu, and R. Karri, "Is split manufacturing secure?" in *Proc. Des. Autom. Test Eur. Conf. Exhibit. (DATE)*, 2013, pp. 1259–1264.

[16] B. Liu and B. Wang, "Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks," in *Proc. Des. Autom. Test Eur. Conf. Exhibit. (DATE)*, 2014, pp. 1–6.

[17] B. Hu et al., "Functional obfuscation of hardware accelerators through selective partial design extraction onto an embedded FPGA," in *Proc. Great Lakes Symp. VLSI*, 2019, pp. 171–176.

[18] 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 *Proc. 57th ACM/EDAC/IEEE Des. Autom. Conf.*, 2020, pp. 1–6.

[19] J. Bhandari et al., "Exploring eFPGA-based redaction for IP protection," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, 2021, pp. 1–9.

[20] H. M. Kamali, K. Z. Azar, K. Gaj, H. Homayoun, and A. Sasan, "LUT-lock: A novel LUT-based logic obfuscation for FPGA-bitstream and ASIC-hardware protection," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI (ISVLSI)*, 2018, pp. 405–410.

[21] G. Kolhe, P. D. S. Manoj, S. Rafatirad, H. Mahmoodi, A. Sasan, and H. Homayoun, "On custom LUT-based obfuscation," in *Proc. Great Lakes Symp. VLSI*, 2019, pp. 477–482.

[22] G. Kolhe et al., "Security and complexity analysis of LUT-based obfuscation: From blueprint to reality," in *Proc. IEEE/ACM Int. Conf. Comput.-Aided Des. (ICCAD)*, 2019, pp. 1–8.

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

[24] G. Kolhe et al., "Breaking the design and security trade-off of look-up-table–based obfuscation," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 27, no. 6, pp. 1–29, 2022.

[25] Z. U. Abideen, T. D. Perez, and S. Pagliarini, "From FPGAs to obfuscated eASICs: Design and security trade-offs," in *Proc. Asian Hardw. Orient. Security Trust Symp. (AsianHOST)*, 2021, pp. 1–4.

[26] K. E. Murray et al., "VTR 8: High-performance CAD and customizable FPGA architecture modelling," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 13, no. 2, pp. 1–55, 2020.

[27] M. G. A. Martins, R. P. Ribas, and A. I. Reis, "Functional composition: A new paradigm for performing logic synthesis," in *Proc. 13th Int. Symp. Qual. Electron. Des. (ISQED)*, 2012, pp. 236–242.

[28] M. G. A. Martins, L. Rosa, A. B. Rasmussen, R. P. Ribas, and A. I. Reis, "Boolean factoring with multi-objective goals," in *Proc. Int. Conf. Comput. Des. (ICCD)*, 2010, pp. 229–234.

[29] "Xilinx Kintex-7 FPGA KC705 evaluation kit." Xilinx, Inc. 2021. [Online]. Available: https://www.xilinx.com/products/boards-and-kits/ek-k7-kc705-g.html

[30] M. Imran, Z. U. Abideen, and S. Pagliarini, "An open-source library of large integer polynomial multipliers," in *Proc. 24th Int. Symp. Des. Diagnos. Electron. Circuits Syst. (DDECS)*, 2021, pp. 145–150.

[31] J. Al-Eryani. "Floating-point unit (FPU) controller." 2017. [Online]. Available: https://opencores.org/projects/fpu100

[32] J. Carlos. "FPGA-based median filter." 2014. [Online]. Available: https://opencores.org/projects/fpu100

[33] S. Joachim. "SHA-256." 2020. [Online]. Available: https://github.com/secworks/sha256

[34] O. Kindgren and M. John. "FPGA-based median filter." 2015. [Online]. Available: https://github.com/openrisc/or1200

[35] R. S. Rajarathnam, Y. Lin, Y. Jin, and D. Z. Pan, "ReGDS: A reverse engineering framework from GDSII to gate-level netlist," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust (HOST)*, 2020, pp. 154–163.

[36] P. Subramanyan, S. Ray, and S. Malik, "Evaluating the security of logic encryption algorithms," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust (HOST)*, 2015, pp. 137–143.

[37] M. Yasin, A. Sengupta, M. T. Nabeel, A. Ashraf, J. Rajendran, and O. Sinanoglu, "Provably-secure logic locking: From theory to practice," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security*, 2017, pp. 1601–1618.

[38] J. A. Roy, F. Koushanfar, and I. L. Markov, "EPIC: Ending piracy of integrated circuits," in *Proc. Des. Autom. Test Eur.*, 2008, pp. 1069–1074.

[39] K. Shamsi, M. Li, T. Meade, Z. Zhao, D. Z. Pan, and Y. Jin, "AppSAT: Approximately deobfuscating integrated circuits," in *Proc. IEEE Int. Symp. Hardw. Orient. Security Trust (HOST)*, 2017, pp. 95–100.

[40] J. Rajendran, Y. Pino, O. Sinanoglu, and R. Karri, "Security analysis of logic obfuscation," in *Proc. Des. Autom. Conf.*, 2012, pp. 83–89.

[41] 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)*, M. Wallace, Ed. Berlin, Germany: Springer, 2004, pp. 438–452.

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

[43] "Using encryption and authentication to secure an ultrascale/ultrascale+ FPGA bitstream." Xilinx. 2022. [Online]. Available: https://www.xilinx.com/content/dam/xilinx/support/documents/application_notes/xapp1267-encryp-efuse-program.pdf

[44] P. Mohan, O. Atli, O. Kibar, M. Zackriya, L. Pileggi, and K. Mai, "Top–down physical design of soft embedded FPGA fabrics," in *Proc. ACM/SIGDA Int. Symp. Field-Program. Gate Arrays*, 2021, pp. 1–10.

Zain Ul Abideen (Graduate Student Member, IEEE) received the M.S. degree in computer engineering (Master in Integration, Security and Trust in Embedded Systems) from the Grenoble Institute of Technology, Grenoble, France, in 2019. He is currently pursuing the Doctoral degree with the Tallinn University of Technology, Tallinn, Estonia.

During his master's studies, he was associated with the Cybersecurity Institute, Univ. Grenoble Alpes, Grenoble. He worked on hardware security and side-channel attacks. His research work is mainly focused on hardware security and obfuscation-based ASIC design.

Tiago Diadami Perez (Graduate Student Member, IEEE) received the M.S. degree in electrical engineering from the University of Campinas, São Paulo, Brazil, in 2019. He is currently pursuing the Ph.D. degree with the Tallinn University of Technology, Tallinn, Estonia.

From 2014 to 2019, he was a Digital Designer Engineer with Eldorado Research Institute, São Paulo. His current research interests include the study of hardware security from the point of view of digital circuit design and IC implementation.

Mayler Martins received the M.S. (*summa cum laude*) and Ph.D. degrees in microelectronics from the Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil, in 2012 and 2015, respectively.

From 2016 to 2018, he was a Research Scientist with the ECE Department, Carnegie Mellon University, Pittsburgh, PA, USA. He worked as a Lead Engineer with Siemens EDA, Fremont, CA, USA, from 2018 to 2022. He is currently a Research and Development Engineer Staff with Synopsys, Sunnyvale, CA, USA. His current research interests include logic synthesis methods focusing on QoR optimization.

Samuel Pagliarini (Member, IEEE) received the Ph.D. degree from Telecom ParisTech, Paris, France, in 2013.

He has held research positions with the University of Bristol, Bristol, U.K., and Carnegie Mellon University, Pittsburgh, PA, USA. He is currently a Professor with the Tallinn University of Technology, Tallinn, Estonia, where he leads the Centre for Hardware Security. His current research interests include many facets of digital circuit design and hardware security.

Appendix 3

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

Orientation's Impact on the bias pattern of SRAM-Based PUFs

Zain Ul Abideen
Department of Computer Systems, Centre for Hardware Security, Tallinn University of Technology (TalTech), 12616, Tallinn, Estonia

Rui Wang
Intrinsic ID, HTC 83, 5656AG, Eindhoven, Netherlands

Tiago Diadami Perez
Department of Computer Systems, Centre for Hardware Security, Tallinn University of Technology (TalTech), 12616, Tallinn, Estonia

Geert-Jan Schrijen
Intrinsic ID, HTC 83, 5656AG, Eindhoven, Netherlands

Samuel Pagliarini
Department of Computer Systems, Centre for Hardware Security, Tallinn University of Technology (TalTech), 12616, Tallinn, Estonia

PHYSICALLY UNCLONABLE Function (PUF) modules are a useful hardware security primitive due to their uniqueness, non-reproducible and unclonable features. Generally speaking, a PUF is used for two applications: secret key generation for cryptography use and/or device authentication for Integrated Circuits (ICs) [1]. PUFs exploit process variation (e.g., gate oxide thickness, size, and threshold voltage) that occurs naturally during the fabrication of ICs. In spite of the fact that the circuits are fabricated with identical layouts, every transistor presents slightly random electric properties which aid to generate a unique identity [2]. Importantly, these random properties cannot be replicated even if an adversary has access to the full design, thus it provides a unique advantage for anti-counterfeiting measures [3].

SRAM-based PUF technology is, by far, the most studied form of a PUF [4], [5], [6], [7], [8], [9], [10]. The digital signature of an SRAM-based PUF is the raw entropy of the SRAM array which is converted into the form of digital bits. SRAM-based PUF offers a combination of simplicity, low cost, high reliability, scalability, and cryptographic strength, making them a preferred choice for commercial PUF solutions. Another advantage of SRAM-based PUFs is that they rely on standard SRAM IP that is typically available to designers. The same memory macros utilized for design purposes can also serve as a PUF, there is no need to have a customized memory macro. For these reasons, our work also focuses on SRAM-based PUFs.

In this work, we explore the impact that *designer's choice of orientation* have on the bias pattern of SRAM-based PUFs. We designed and fabricated a 65nm CMOS chip featuring eleven SRAMs with varying numbers of addresses, number of words, aspect ratio, and chosen bitcell. Secondly, we also considered rotation for this evaluation. This is the first study study which makes a significant contribution to explore and analyze the phenomenon of the bias pattern in SRAM-based PUFs. We aim to deepen our understanding of how orientation influences the observed bias patterns in SRAM-based PUFs.

Digital Object Identifier 10.1109/DNT.2023.Doi Number
Date of publication 00 xxxx 0000; date of current version 00 xxxx 0000

Copublished by the IEEE CEDA, IEEE CASS, IEEE SSCS, and TTT

Background

The key component of SRAM is the bitcell, which consists of two CMOS inverters interconnected in a positive feedback loop, forming a bistable storage element. The initial state of each bitcell in the SRAM is determined by the process variation that arises during the IC's manufacturing process. The stability of each bit is contingent upon the degree of threshold voltage mismatch between the local devices. The typical 6T-SRAM cell exhibits a preferred state due to stochastic variations in the threshold voltages of its transistors. This randomness in the initial values of 6T-SRAM results in a largely unpredictable, largely repeatable, and yet exclusive pattern of zeros and ones.

The architecture of the memory and different orientations are illustrated in Fig. 1. All SRAM memories are single-port and utilize 6 transistors per bitcell. Furthermore, we utilized two distinct types of memories. Those with a bitcell size of approximately $\sim 0.65 \mu m^2$ are labelled as low density and high speed, whereas those with a bitcell size of approximately $\sim 0.50 \mu m^2$ exhibit high density but low speed. The high speed memory adopts Standard-vt for both periphery and bitcells, while the low speed memory utilizes Mixed-vt for periphery and High-vt for bitcells. The SRAMs are organized in an array of memory locations, where each memory access involves reading or writing all the bits in a single location.

The memory system comprises a specific number of arrays, each containing a memory matrix that exhibits variations between high speed and low speed memory types. These distinctions in the memory matrix design reflect the differing characteristics and performance attributes associated with each memory type. By considering these variations, we gain a comprehensive understanding of how the memory system is structured to accommodate and optimize for different speed requirements. As shown in Fig. 1 (a), both the high speed and low speed memory configurations have half of the bits located on the right and the other half on the left. In the case of low speed memory, we will consider an example with a datawidth of 32 bits and a depth of 1024 locations, as depicted in panel (b) of Fig. 1. Both low-speed and high-speed memory implementations utilize a memory matrix consisting of $2 \times n$ bitcells, where n represents the number of adjacent bits within the matrix. Specifically, the architecture in Fig. 1 features two adjacent memory matrices (2×2) arranged within the memory array, forming an array. There is an equal amount of data on both sides of the figure, with C1 representing the number of memory matrices on each side. As a result of this symmetry, a balanced representation of data is achieved throughout the entire memory system, contributing to its efficiency.

The column mux ratio (m) plays a crucial role in a memory array as it represents the number of memory cells that are connected to a shared bitline (column). This ratio determines the number of memory cells associated with a single read/write bitline, thereby impacting the density and performance characteristics of the SRAM. The selection of the column mux ratio not only affects the density and performance of the SRAM but also has implications for the aspect ratio (width and height of the memory array). A lower column mux ratio means that fewer cells are connected to a bitline, which can impact both the width and height of the array.

The placement of the memory within the chip is determined by the overall floorplan. In our test chip, we have utilized 11 SRAMs with different possible orientations, as depicted in Figure 1. The baseline orientation is denoted as R0, representing a rotation angle of zero degrees. The notation MX signifies mirroring along the x-axis, R270 refers to a rotation of 270 degrees, R90 represents a rotation of 90 degrees, and MY90 signifies mirroring along the y-axis followed by a rotation of 90 degrees (all rotations mentioned here are anti-clockwise).

In SRAM-based PUFs, the combination of bitcells forms the response data. This response data is a collection of responses generated by the SRAM-based PUF. 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 ideal PUF should produce the same output in response to a given challenge, regardless of the environmental and operational conditions. The reliability of a PUF can be assessed by evaluating its with-in class hamming distance (WCHD), which is the fractional hamming distance between measurements taken at various conditions during reconstruction and a reference initial measurement (enrollment).

SRAM-based PUFs leverage *entropy* from process variations to build various kinds of unique fingerprints for each identically fabricated chip. For *entropy*, one important parameter is the *bias pattern* that is linked with the PUF's physical characteristics. The response

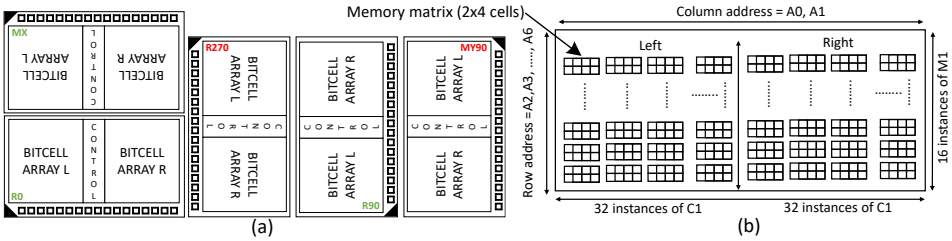


Figure 1: Architecture and orientations of memory: low speed memory with 32-bit datawidth and 1024 location depth.

data often displays a consistent bias towards either a logical zero or one. This bias cannot be easily determined by calculating the fractional hamming weight, which measures the percentage of ones in the raw output response. This recurrent phenomenon is usually referred to as the *bias pattern*.

The biased SRAM-based PUFs will show higher auto-correlation values compared to non-biased devices. We select the SRAM-based PUF with the highest auto-correlation per SRAM-based PUF's design and run Fast Fourier Transform (FFT) on the auto-correlation. The auto-correlation has some periodic effect, and the period corresponds to the width of the bias pattern. 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 [11].

ASIC Demonstration

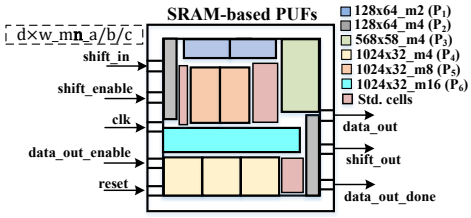
The simplified internal architecture of the chip is depicted in Fig. 2a. The chip contains 6 distinct SRAM-based PUFs, some of which are replicas denoted by underscored letters (a, b, c, etc.). This results in a total of 11 SRAM-based PUFs within the chip. [1]. The designed chip consists of a simple serial interface and eleven different SRAM-based PUFs At the input of the serial interface, 15 bits are employed for the addressing and selecting of eleven SRAM-based PUFs. At the output of the serial interface, 70 bits are retrieved, including 64 bits of data along with 3 start bits and 3 stop bits for data alignment. The chip includes both slow and fast memories, with the smaller memories (128×64) being classified as fast memories while the rest of the memories are classified as slow.

[1]The d denotes the depth or number of addresses, the w denotes the datawidth, and m denotes the column mux.

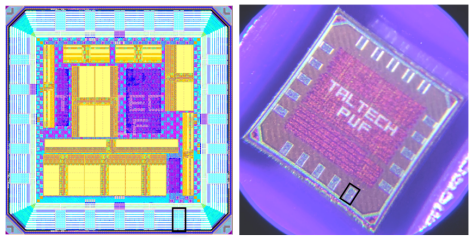
We have manually placed all memories multiple times to maximize the area usage. Next, we executed P&R to generate the final layout of the chip. The control logic inside the chip is minimal while the majority of the area is occupied by memories, making buffers, inverters, and sequential cells less than 5% of the circuit. The number of buffers, combinational cells, inverters, and sequential cells are 233, 640, 881, and 54, respectively.

Fig. 2b shows the bare die in the right panel and its layout is illustrated in the left panel. One can recognize the placement of memories and IO cells as illustrated in the layout. A black rectangle has been added to facilitate the identification of the lower right corner of the chip. To validate the design, we have packaged 50 samples of the chip and all of the packaged samples were confirmed to have full functionality. We have designed a custom printed circuit board (PCB) for the purpose of testing our ASIC prototype which contains a chip socket, relays, and passive components to assist with measurements and filtering noise from the power supplies. The chip is controlled by a Raspberry Pi 3 Model B to facilitate the testing. The relays are used to turn on and off the VDD (1.2V) and VDDIO (3.3V) power supplies.

For the experiments, we recorded the PUF's responses from the chips by power cycling every chip 10 times. The purpose of using relays became clear as we utilized them to turn on and off the chip to complete one power cycle. The Raspberry Pi sent serial bits to the chip and stored the corresponding PUF response in a text file. The Pi repeated this process by turning on and off the chip for the subsequent experiments. We repeat this experiment ten times to analyze the stability of the PUF's response. We calculated the total number of bits stored by multiplying the datawidth of each memory with its respective depth. In this way, the



(a) The simplified architecture of the SRAM-based PUFs.



(b) Layout (left) and die micrograph (right) of the fabricated chip. The highlighted pin marks the lower-right corner.

Figure 2: The simplified architecture and chip design of the SRAM-based PUFs.

summation of these bits amounts to a total of 557.232 thousand bits, collected from a single chip using this approach.

Results and Discussions

Prior to presenting our analysis, we will examine three important parameters: WCHD, MHW, and entropy. These parameters provide valuable information on the behavior and characteristics of the SRAM-based PUF. To start this, we merged data from 50 chips and evaluated the corresponding metric for each SRAM measurement. A summary of the results is presented in Table. 1 where the first column lists the type of memory, and the subsequent columns display WCHD, MHW, and entropy. The WCHD values, which are below 10%, indicate a high level of reliability in the SRAM-based PUFs. Ideally, the MHW value should be around 0.5, and the results in the second column of Table 1 closely align with the ideal value. The entropy values are derived from the MHW values and also align well with the expected entropy levels. Furthermore, the entropy values obtained in our study were found to be comparable to those reported in [9]. The entropy values reported in [10] are slightly higher but their analysis is based on an older technology node, making a fair comparison challenging. Overall, the results confirmed that our SRAM-PUFs behave as expected and in line with previous studies.

Next, the analysis seeks to determine the impact of bias pattern of the SMRAM-based PUFs over varying sizes, mux selection ratios, memory types and orientations. The orientation of the memory in the circuit's floorplan has no impact on its functionality. A designer may choose to rotate a memory to obtain better routability. However, memory orientation affects the bias direction (BD). To start the analysis, the

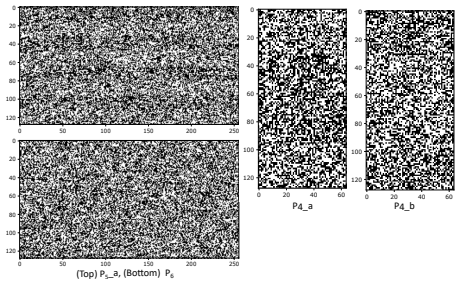


Figure 3: The start-up pattern of the P_5_a, P_6, P_4_a and P_4_b SRAM-based PUFs is represented by a sequence of bits, with white spaces denoting a logical one.

start-up pattern of SRAMs P_5_a, P_6, P_2_a and P_2_b is plotted in Fig. 3. One can visualize the biasing patterns that we will explain in the following results. The response vectors are concatenated into valid binary data from each row into a one-dimensional vector to determine the bias pattern. Next, we compute each auto-correlation for each chip (but only on the first measurement). The correlation between the MHW for all 50 chips is depicted in Fig. 4. Notably, the direction of correlation varies from one SRAM-based PUF to another when considering P_1_a as the baseline. For example, if we compare the correlation peaks for the baseline and P_2_b in Fig. 4 then the peaks are opposite; thus, we report the bias direction as negative. A summary of the relationship between the memory orientation and the bias correlation direction is listed in the last two columns of the Table. 1.

Observation 1: The bias pattern between different sizes is influenced by the data width, but it does not affect the direction of the bias pattern. The data width (word size), bias pattern, and the number of instances for different SRAM-based PUF instances

Table 1: Results for the robustness evaluation of SRAM-PUFs

SRAM-PUF	WCHD (%)	MHW	Entropy by one-probability	Bias pattern	Orientation	BD
P_1_a, P_1_b	5.8-8.5	0.391-0.622	0.685-1	$0(32)1(64)0(64), \ldots$	R0, R0	+, +
P_2_a, P_2_b	5.8-8.8	0.440-0.564	0.826-1	$0(32)1(64)0(64), \ldots$	R90, R270	+, -
P_3	5.5-7.0	0.430-0.539	0.811-1	$0(29)1(29), \ldots$	R270	-
P_4_a, P_4_b, P_4_c	*5.0-9.1	0.435-0.541	0.824-1	$0(16)1(16), \ldots$	MX, MX, MX	+, +, +
P_5_a, P_5_b	*5.2-8.0	0.430-0.575	0.798-1	$0(16)1(16), \ldots$	R270, MY90	-, -
P_6	5.1-7.0	0.390-0.580	0.713-1	$0(16)1(32)0(32), \ldots$	R0	+

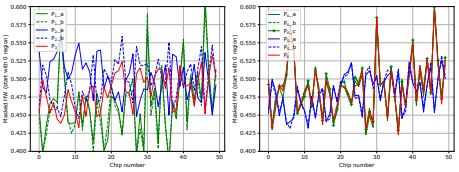


Figure 4: The correlation of SRAM-based PUFs for the biasing pattern with baseline SRAM-based PUF (P_1_a).

are listed in the last three columns of Table. 1. For instance, consider two identical memories, P_2_a and P_2_b. Despite their identical nature, these memories exhibit different bias directions. One memory may have a positive bias direction, while the other may have a negative bias direction.

Observation 2: Overall, the change in mux ratio causes the different aspect ratios, which have different bias patterns from 1024×32_mux8 and 1024×32_mux16. So, the larger mux ratio has an impact on the bias pattern. The identical SRAM-based PUFs have an identical bias pattern. On the other hand, the column mux ratio does not influence the direction of the bias pattern. The direction of bias direction is independent of the column mux ratio.

Observation 3: Fast and slow memories do not have any connection with the width of the bias pattern. Additionally, they are also not related to the direction of the bias pattern. In other words, the use of different bitcells does not translate into different bias widths or directions.

Observation 4: However, it is important to note that SRAM-based PUFs may display an alternating bias pattern, which is a result of the internal structure of SRAMs. As discussed in Section , the SRAM macro consists of 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: It is confirmed that two memory orientations, namely R270 and MY90, show a negative biasing direction when compared to the other memories. The bias direction is found to be independent of memory type (either low speed or high speed), column mux ratio, memory size and the utilization of SRAMs with different bitcells.

Observation 6: It has been confirmed that the bias pattern direction observed in SRAM-based PUFs is not influenced by factors such as power planning and the overall floorplan of the chip, except for considerations specifically related to orientation.

Observation 7: Intra-die process variation exhibits uniqueness between individual dies, and thus, it does not contribute to the direction of the bias pattern. We further confirm that all chips come from the same wafer and have not been rotated on the MPW reticle.

According to observations 1-7, the direction of the bias pattern cannot be affected by varying sizes, mux ratios, memory types, memory structure, or process variations. The orientations are the only factors that determine it. We make the following assumptions to justify the direction of the bias pattern.

Hypothesis 1: This effect emanates from the orientation of the bitcells themselves. The R90 orientation of the SRAM positions the bitcells vertically compared to the R0 orientation. When comparing the R90 orientation as a reference, the R270 and MY90 orientations flip the left and right bitcell arrays as illustrated in Fig. 1. This means that the placement direction of the bitcells associated with the first address of the SRAM becomes reversed. In other words, the direction of the bitcell placement linked to the first address of the SRAM becomes the opposite in these orientations.

Hypothesis 2: Understanding the impact of doping variations on the bias pattern is crucial for analyzing and mitigating their effects in SRAM-based PUFs. During the lithography process, the doping of transistors in SRAM cells plays a crucial role in their

behavior. Although the process aims for uniformity, variations in doping can occur due to the limitations of the fabrication equipment [12]. The machine moves along the x-axis, doping the transistors as it progresses from left to right or right to left. It then steps up and continues the doping process until all transistors are treated. These doping variations have an impact on the initial state and stability of the SRAM cells. The stability of an SRAM cell is typically evaluated using the concept of static noise margin (SNM), which represents the minimum noise voltage required to flip the state of the bitcell. In SRAM cell design, the width-to-length (W/L) ratios of the load transistors and access transistors are often set to be as close to 1.0 as possible. The ratio of the driver transistor's W/L to the access transistor's W/L is known as the cell ratio, which determines the cell's stability and size [12]. The doping variation directly affects the W/L ratio, leading to variations in the SNM value. When considering the doping effect, transistors along the vertical axis experience more significant variations compared to adjacent transistors along the x-axis. These doping variations impact all SRAM-based PUFs to some extent. However, specific orientations, such as R270 and MY90, exhibit a distinctive negative bias pattern. In these orientations, the transistor arrangement becomes reversed or opposite to the baseline SRAM-based PUF's orientation.

Conclusions

Our experiments revealed that the orientation of the SRAMs affects the biasing direction significantly. This observation could prove useful for designing smart error correction algorithms for SRAM-based PUFs. Regarding the evaluation characteristics of the SRAM-based PUFs, our findings are consistent with prior research, indicating that our SRAMs are representative. Nonetheless, our study presents a useful contribution to the field of PUF research and highlights the importance of careful physical implementation when designing SRAM-based PUFs.

ACKNOWLEDGMENTS

This work has been partially conducted in the project "ICT programme" which was supported by the European Union through the ESF. It was also supported by EU's Horizon 2020 research and innovation programme under Grant Agreement No 872614 (SMART4ALL).

REFERENCES

1. G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *2007 44th ACM/IEEE Design Automation Conference*, 2007, pp. 9–14.
2. 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*, ser. CCS '02. Association for Computing Machinery, 2002, p. 148–160.
3. W. Che, F. Saqib, and J. Plusquellic, "Puf-based authentication," in *2015 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, 2015, pp. 337–344.
4. 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)*, 2017, pp. 1–2.
5. 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.
6. 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)*, 2012, pp. 486–489.
7. 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. 79 947–79 960, 2021.
8. 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.
9. 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)*, 2012, pp. 1319–1324.
10. 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)*, 2020, pp. 7–12.
11. NIST, "Recommendation for the entropy sources used for random bit generation," 2018. [Online]. Available:

https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-90B.pdf

12. B. Cheng, S. Roy, and A. Asenov, "The impact of random doping effects on cmos sram cell," in *Proceedings of the 30th European Solid-State Circuits Conference*, 2004, pp. 219–222.

Zain Ul Abideen received his M.S. degree in computer engineering (Master in Integration, Security and TRust in Embedded systems) from Grenoble Institute of Technology, Grenoble, France, in 2019. During his master's studies, he was associated with the Cybersecurity Institute Univ. Grenoble Alpes. He worked on hardware security and side-channel attacks. He is currently pursuing his doctoral studies at Tallinn University of Technology (TalTech), Tallinn, Estonia. His research work is mainly focused on hardware security and obfuscation-based ASIC design.

Rui Wang received his M.S. degree in electrical engineering from Eindhoven University of Technology in 2016. He is working for Intrinsic ID as a technology engineer since 2017, focusing on various product research topics with PUF technology.

Tiago Diadami Perez received the M.S. degree in electric engineering from the University of Campinas, São Paulo, Brazil, in 2019. He is currently pursuing a Ph.D. degree at Tallinn University of Technology (TalTech), Tallinn, Estonia. From 2014 to 2019, he was a Digital Designer Engineer with Eldorado Research Institute, São Paulo, Brazil. His current research interests include the study of hardware security from the point of view of digital circuit design and IC implementation.

Geert-Jan Schrijen received the M.S. degree in electrical engineering from the University of Twente in 2000. In 2001 he joined the security group of Philips Research in Eindhoven where he worked on various topics including Physical Unclonable Functions (PUFs), which resulted in the spin-off of Intrinsic ID in 2008. As a senior algorithm designer within Intrinsic ID, Geert-Jan focused on the development of signal processing algorithms and security architectures for embedded systems using PUF technology. In 2011 Geert-Jan became responsible for all development and engineering work at Intrinsic ID in his role of VP Engineering. In 2016 he was appointed as CTO of the company.

Samuel Pagliarini (M'14) received the Ph.D. degree from Telecom ParisTech, Paris, France, in 2013. He has held research positions with the University of Bristol, Bristol, UK, and with Carnegie Mellon University, Pittsburgh, PA, USA. He is currently a Professor at Tallinn University of Technology (TalTech) in Tallinn, Estonia where he leads the Centre for Hardware Security. His current research interests include many facets of digital circuit design and hardware security.

Appendix 4

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

An Overview of FPGA-inspired Obfuscation Techniques

ZAIN UL ABIDEEN, Centre for Hardware Security, Tallinn University of Technology, Estonia
SUMATHI GOKULANATHAN, Centre for Hardware Security, Tallinn University of Technology, Estonia
MUAYAD J. ALJAFAR, Centre for Hardware Security, Tallinn University of Technology, Estonia
SAMUEL PAGLIARINI, Centre for Hardware Security, Tallinn University of Technology, Estonia

Building and maintaining a silicon foundry is a costly endeavor that requires substantial financial investment. From this scenario, the semiconductor business has largely shifted to a fabless model where the Integrated Circuit supply chain is globalized but potentially untrusted. In recent years, several hardware obfuscation techniques have emerged to thwart hardware security threats related to untrusted IC fabrication. Reconfigurable-based obfuscation schemes have shown great promise of security against state-of-the-art attacks – these are techniques that rely on the transformation of static logic configurable elements such as Look Up Tables (LUTs). This survey provides a comprehensive analysis of reconfigurable-based obfuscation techniques, evaluating their overheads and enumerating their effectiveness against all known attacks. The techniques are also classified based on different factors, including the technology used, element type, and IP type. Additionally, we present a discussion on the advantages of reconfigurable-based obfuscation techniques when compared to Logic Locking techniques and the challenges associated with evaluating these techniques on hardware, primarily due to the lack of tapeouts. The survey's findings are essential for researchers interested in hardware obfuscation and future trends in this area.

CCS Concepts: • **Security and privacy → Systems security**; **Hardware security implementation**; **Hardware reverse engineering**; **Hardware attacks and countermeasures**; • **Hardware → Programmable logic elements**; **Programmable interconnect**; **Physical design (EDA)**; **Application-specific VLSI designs**; **Application specific integrated circuits**; **Analysis and design of emerging devices and systems**.

Additional Key Words and Phrases: Hardware security, Trustworthy hardware, Logic obfuscation, FPGA redaction, reconfigurable logic, LUT-based obfuscation

ACM Reference Format:
Zain Ul Abideen, Sumathi Gokulanathan, Muayad J. Aljafar, and Samuel Pagliarini. 2023. An Overview of FPGA-inspired Obfuscation Techniques. *ACM Comput. Surv.* 1, 1 (June 2023), 30 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Integrated Circuit (IC)-based systems have been used in both consumer and military electronics for several decades, enabling a range of devices, from smartphones to satellites. The continued

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

Authors' addresses: Zain Ul Abideen, zain.abideen@taltech.ee, Centre for Hardware Security, Tallinn University of Technology, Akadeemia tee 15a, Tallinn, Harju, Estonia, 12611; Sumathi Gokulanathan, sumathi.gokulanathan@taltech.ee, Centre for Hardware Security, Tallinn University of Technology, Tallinn, Estonia; Muayad J. Aljafar, muayad.al-jafar@taltech.ee, Centre for Hardware Security, Tallinn University of Technology, Akadeemia tee 15a, Tallinn, Harju, Estonia, 12611; Samuel Pagliarini, samuel.pagliarini@taltech.ee, Centre for Hardware Security, Tallinn University of Technology, Akadeemia tee 15a, Tallinn, Harju, Estonia, 12611.

0360-0300/2023/6-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX

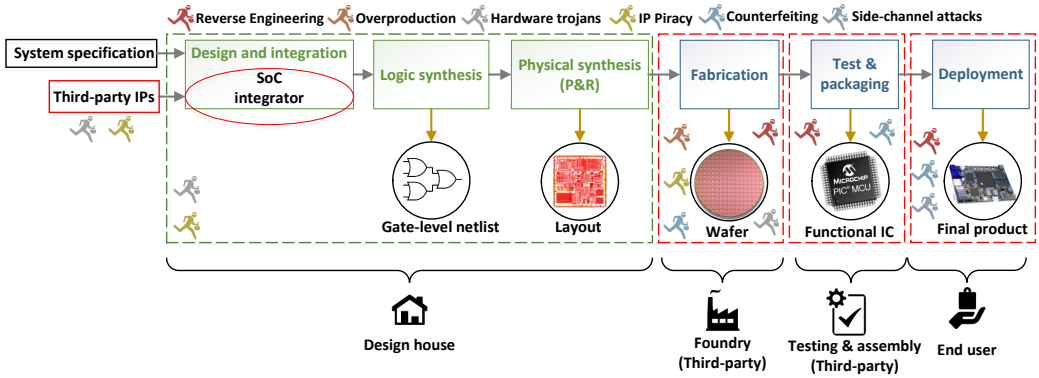


Fig. 1. Typical stages involved in the IC design and in the supply chain, untrusted stages are shown in red.

advancements in technology have also led to the adoption of IC-based systems in newer domains like the Internet of Things (IoT) and multi-cloud environments [71]. In every domain, the demand for high-performance ICs is increasing. The reason behind this trend is the complexity of modern systems and the need for faster speeds to handle larger amounts of data being processed. As a result, the semiconductor industry is experiencing a surge in demand for products such as memory chips, microprocessors, and sensors. For example, the global IC market is forecast to grow from $489 billion in 2021 to $1.136 trillion in 2028 [30]. On the other hand, ICs require advanced manufacturing processes and specialized equipment, which are only available in a limited number of foundries.

As the industry continues to evolve, the complexity of building and maintaining a foundry increases, resulting in skyrocketing costs. As an example, the estimated cost of building a 3nm foundry is in the range of $15-20B [67]. Thus, contemporary semiconductor vendors are increasingly adopting a *fabless model*, where a globalized IC supply chain allows the production of high-performance ICs without the requirement of heavy investment in specialized foundry equipment.

The globalized IC supply chain enables design houses to have access to high-end semiconductor manufacturers [64], but the exposure of the layout design to untrusted entities poses significant security threats as shown in Fig. 1. Losses due to security threats could be severe, including service interruption, damage to public data integrity, monetary losses, etc. For instance, the EU and US warned about the dangers to national security that scammers exploited in the recent IC supply chain crunch [6]. Similarly, the International Telecommunication Union (ITU) and the European Union Intellectual Property Office (EUIPO) reported in 2015 that 12.9% of the total sales of smartphones were lost due to counterfeit electronics. The sales of counterfeit devices in the market caused a loss of EUR 45.3 billion to legitimate industries – a significant monetary loss. The green color in Fig. 1 illustrates the steps performed in a trusted environment. In practice, all involved parties provide assurances but cannot provide guarantees related to the integrity and trustworthiness of the ICs. This lack of guarantees is primarily due to the concept of zero-trust in which one must assume that the foundry and its employees are potential adversaries. The fabrication phase holds the utmost importance as the foundry has access to all low-level details of the design. The related security threats include reverse engineering, overproduction, insertion of hardware Trojans, IP piracy, and counterfeiting [51].

Reverse engineering is extensively demonstrated in the literature, as a method to extract the design and/or technology details of an IC with the help of tools and imaging techniques. It involves a complex process of removing the package of an IC, extracting all the layers, stitching the

individual layers, and analyzing the obtained images to recover the netlist of a design [77]. Reverse engineering also provides an opportunity for IP piracy and counterfeiting. Reverse engineering could be exploited in conjunction with other techniques to extract secret information, i.e., cryptographic keys. Reverse engineering becomes more onerous and time-intensive after fabrication, packaging, or even deployment has occurred, but a skilled adversary can still perform it.

As stated earlier, third-party entities are involved in the design, packaging, and testing processes. The design process also typically includes outsourcing third-party IPs, **piracy** may be evident in different degrees in the form of IP theft, overbuilding at an untrusted foundry, or illegal ownership claims. An unauthorized individual inside the foundry has the potential to steal information through reverse engineering or unlawfully sell the IPs without the authorization of the owner. On the other hand, the untrusted foundry could also be interested to overproduce ICs and sell them in the grey (or black market) at cheaper prices. This is possible because a foundry typically incurs only a marginal increase in costs when manufacturing additional ICs from the same masks [46], i.e., the design house that owns the IP bares all the design-related NRE costs .

In particular, **hardware Trojans** are modifications in the form of small and hard-to-detect logic for malicious purposes. Hardware Trojans are utilized to interrupt the service of an IC [37] or to extract secret information [55]. In the context of the layout, the footprint of the Trojan could be very small which might become invisible to identify and test, especially when there is no reference (golden) design available to cross-verify the functionality. The source of the malicious logic could be a third-party IP or it could also be mounted during manufacturing. As mentioned earlier, the foundry has complete access to the layout therefore it can easily identify potential locations for Trojan insertion [55]. The Trojans/backdoors in third-party IPs may also contain hidden functionalities to expose restricted parts of the design and/or extract some secret information.

Counterfeit ICs are fraudulently made in such a way to appear almost identical to the original ICs. Counterfeit ICs are divided into seven different classes: recycled, remarked, overproduced, out-of-spec/defective, cloned, forged documentation, and tampered (See Fig. 3 in [31]). Recycled, remarked, out-of-spec/defective, and forged documentation are post-fabrication issues that appear when the counterfeit ICs deployed in a product are outsourced from non-authorized vendors or duplicate IC sellers. On the contrary, overproducing, cloning, and tampering are fabrication-time issues that could be completely tackled if (and only if) all the stages in Fig. 1 were fully executed in a trusted environment. Recalling again, the design house has to share the layout of the design thus it exposes all the minor details to untrusted foundries. On the other hand, plenty of techniques have been developed as countermeasures to these threats. This process is evolving with time and there is a race to bring a novel technique that is resistant and practical [29].

Countermeasure techniques to increase the IC security include Logic Locking [75, 81, 87, 89, 90], Camouflaging [26, 47, 86], Split Manufacturing [56, 59], and reconfigurable-based obfuscation techniques [3, 4, 12, 14, 17, 22, 23, 25, 32, 40–44, 49, 50, 53, 54, 60, 65, 66, 68, 69, 76, 80, 85]. Reconfigurable-based techniques typically draw inspiration from Field-Programmable Gate Array (FPGA) devices, thus the title of this article. In general, the aforementioned techniques aim to provide security against the threats that occur during IC fabrication. Some techniques also offer degrees of protection for post-fabrication attacks.

Regarding the reconfigurable-based obfuscation techniques, one of the most formative works came from Microsoft and Iowa State University authors as describe in [14]; the authors appropriately identified that reconfigurable logic could be leveraged as an obfuscation asset. Subsequently, a series of research studies emerged that utilize reconfigurable-based obfuscation schemes to protect digital designs. These techniques include a variety of reconfigurable elements, i.e., static random access memory (SRAM)-LUTs [3, 4, 14, 25, 44, 49, 50], NVM-LUTs [12, 40–43, 80, 85], and Others [17, 22, 23, 32, 53, 54, 60, 65, 66, 68, 69, 76]. These reconfigurable techniques are very promising and

demonstrate potential assurances against almost all hardware security threats. Overall, this paper is the *first survey* to focus on *reconfigurable-based obfuscation techniques* that combat security threats. Our intention is to present a detailed study of obfuscation trends, trade-offs, and recent attacks. This work provides a comprehensive overview of the reconfigurable-based obfuscation landscape, classifying the techniques based on three important factors: technology used, element type, and IP type. The efficiency of these techniques is evaluated and compared in terms of Power-Performance-Area (PPA) overheads. Finally, we discuss the benefits of reconfigurable-based obfuscation when compared to Logic Locking techniques and the challenges of evaluating obfuscation on hardware.

The structure of this paper is given as follows: Section 2 classifies the reconfigurable-based obfuscation techniques and provides in-depth explanations. Section 3 elaborates on the comparison and analysis between numerous reconfigurable-based obfuscation techniques. Then, a comprehensive study of attacks and their evaluation of various obfuscation techniques is given in Section 4. In Section 5, a rich discussion is provided as well as a comparison to Logic Locking. Future trends and challenges are also discussed in Section 5. Finally, we conclude in Section 6.

2 BACKGROUND AND CLASSIFICATION OF RECONFIGURABLE-BASED OBFUSCATION

As previously mentioned, in order to have access to advanced technologies, design companies often outsource IC manufacturing to third-party foundries. Fabless design houses are concerned with protecting their designs against potential threats that could arise during manufacturing at an untrusted foundry. In this context, reconfigurable-based obfuscation techniques can safeguard designs by addressing nearly all of the threats outlined in Fig. 1 [1].

2.1 Background

Over the last few decades, reconfigurable devices such as FPGAs and FPGA-based SoCs have been widely used as stand-alone solutions. It was only recently that the ability to reconfigure a design has been considered a form of obfuscation. In Fig. 2, we frame the evolution and usage of reconfigurable devices into two phases: the **pre-obfuscation era** and the **design for security era**. In 1984, Xilinx introduced the first FPGA, called the XC2064 [36]. This FPGA had 64 logic cells and was programmed using a hardware description language (HDL) called ABEL. In the late 1980s and early 1990s, FPGAs became more popular as their capacity increased and their price decreased [78]. Xilinx and Altera were the two leading FPGA manufacturers at that time.

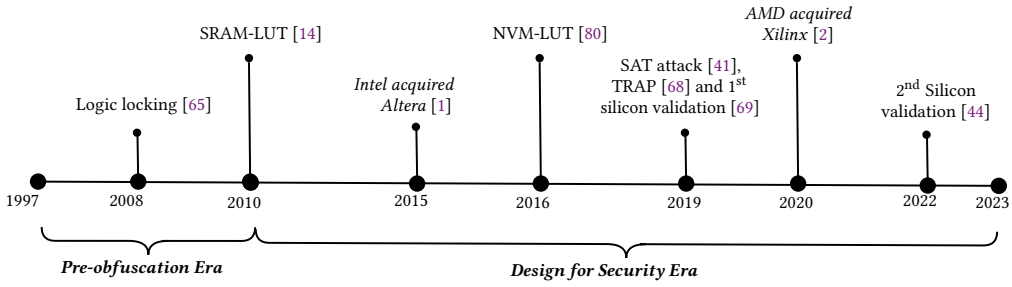


Fig. 2. Reconfigurable Logic: Navigating the Shift from Pre-Obfuscation to Security Era

The traditional island-style architecture of an FPGA is illustrated in Fig. 3. The architecture of an FPGA consists of several basic building blocks such as interconnect wires, configurable logic

[1]The effectiveness against side-channel attacks is not always present.

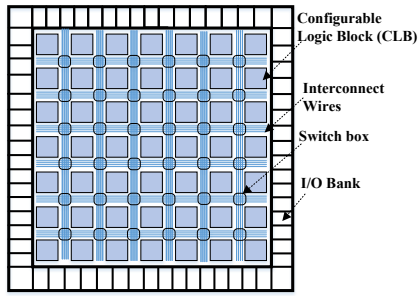


Fig. 3. The traditional island-style architecture of FPGA (adapted from [35]).

block (CLB), switch matrix, and I/O bank. A CLB consists of a small number of logic gates and can be configured to implement a specific logic function. The input/output blocks (IOBs) provide the interface between the FPGA and the external world. The interconnect network is used to connect the various building blocks together, and can be programmed to implement different routing configurations depending on the specific application. A CLB is typically composed of a LUT, a flip-flop, and a multiplexer. The LUT is used to implement combinational logic functions, while the flip-flop is used to store the state of a sequential logic circuit. The multiplexer is used to select between different inputs to the logic element. However, modern FPGAs are characterized by more complex and non-uniform device grids, with I/Os often placed in columns instead of around the perimeter [33]. It has been observed that the size of LUTs in FPGAs has evolved over time. Accordingly, the Virtex 4 family of FPGAs had 4-input LUTs, whereas the Virtex 5 and Virtex 6 families had 5-input and 6-input LUTs, respectively [82, 83]. It is even possible to find some manufacturers offering FPGAs with even larger 8-input LUTs [34]. With the evolution of LUT sizes, FPGAs are able to support a wide range of increasingly complex and sophisticated digital logic designs. Except for minor terminology differences among different vendors, the architecture depicted in Fig. 3 is representative of an FPGA.

In the early 2000s, new application domains emerged and the underlying process technology changed while the architecture of FPGAs continued to evolve. The ability to partially program and dynamically reconfigure FPGA architectures enables digital circuit design and implementation to be more flexible. Modifying only a small portion of the fabric allows for the design to continue to operate while some small parts are programmed. In practice, the same board can be used to switch from one design to another design, allowing for greater design flexibility [39]. In FPGA-based system on chips (SoCs), partial and dynamic reconfiguration [48] are possible for the FPGA fabric as well as for periphery IPs.

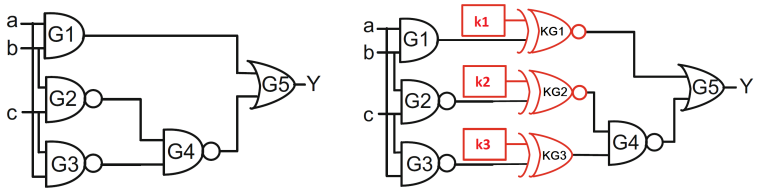
Moving forward with the evolution of FPGA architectures, modern FPGAs are comprised of a range of modules including memory, DSP, PLLs, clocking, networking, and more [9, 84]. Looking ahead, these blocks are expected to continue evolving and expanding in capability. Examples of these architectures include large blocks such as hardware accelerators [11]. A hybrid architecture emerged in 2003 when Xilinx embedded its FPGA technology into IBM's Application-Specific Integrated Circuits (ASICs), enabling designers to add programmable logic into their designs without developing a separate board for the FPGA [28]. At the same time, FPGA-based SoCs targeting digital signal processing applications extended their computational power through reconfigurable solutions often involving a matrix of computational elements with programmable interconnections [15, 18, 70, 91]. In practice, FPGA-based SoC are complex devices that employ much more than hard

and soft embedded processors [10]. In a very recent trend, FPGA-based SoCs are being shipped with a Network-on-Chip (NoC) subsystem [74] to interconnect all of its modules.

Over the next two decades, the boundaries between ASIC and FPGA design became less clear. The embedded-Field Programmable Gate Arrays (eFPGAs) emerged as a small FPGA module that can be integrated into ASIC. The eFPGA IP can be licensed for use in a similar manner to any other IP. For each application, eFPGA IP designers can specify exactly how many logic units, digital signal processing (DSP) and machine learning processing units (MLP) they provide. By eliminating unnecessary FPGA features, this results in more flexibility, lower costs, and smaller eFPGA IP area. Furthermore, if a custom or special FPGA architecture is needed, it can also be implemented as an eFPGA for greater reconfigurability. Several IP providers are offering eFPGA blocks of various granularities and architectures.

Concerning the security side, a landmark development in the field of obfuscation is the Logic Locking concept [65] that came to light in 2008 (see timeline in Fig. 2). Logic locking is a technique used at design time to safeguard ICs against supply chain threats and it involves the insertion of additional logic into a circuit and securing it with a secret key. Key inputs, driven by an on-chip tamper-proof memory, are added to the locked circuit along with the original inputs. The additional logic may consist of combinational logic like MUX, AND, OR, and XOR gates [87]. The design functions correctly and produces the correct output only when the correct key value is applied. Otherwise, its output differs from that of the original design.

In Fig. 4a, an example of a circuit that consists of three inputs and one output is depicted. Fig. 4b shows the locked version of the circuit, which includes three additional XOR/XNOR key gates. Each key gate has one input driven by a wire of the original design, while the other input, called the key input, is driven by a key bit stored in the tamper-proof memory. In the locked circuit of Fig. 4b, when the correct key value 110 is loaded into memory, all key gates behave as buffers and produce the correct output for any input pattern. However, applying an incorrect key value, such as 010, causes certain key gates to behave as inverters, leading to an error injection into the circuit. For instance, the key gate K1 acts as an inverter in the case of input pattern 000 and key value 010, producing an incorrect output Y = 1.



(a) An example circuit that consists of three inputs.

(b) Three XOR/XNOR gates should use the key value of 110.

Fig. 4. Logic locking using XOR/XNOR gates [87].

This initial period in the evolution of reconfigurable devices can be considered as **the pre-obfuscation era**. Then, the first reconfigurable-based obfuscation technique was proposed in 2010. This marks the beginning of the **the design for security era** which continues to this day. Several techniques have been proposed since.

Recent years were also marked by significant acquisitions in the semiconductors market. One of such remarkable events is Intel's acquisition of Altera, a leading provider of FPGA technology, in 2015 [20]. By acquiring Altera, Intel gained access to the company's industry-leading FPGA

technology, which is widely used in data centers, networking, and embedded systems applications. In 2020, AMD's acquisition of Xilinx was a strategic move to expand its market reach and strengthen its position in the high-performance computing (HPC) and data center markets. Xilinx FPGAs and adaptive SoC solutions complement AMD's portfolio of central processing units (CPUs), graphics processing units (GPUs), and other accelerator technologies.

In 2016, Menta introduced the first commercially available eFPGA IP, which allows designers to integrate eFPGA IP into their own ASICs. This approach enables the integration of a reprogrammable logic fabric into a wide range of ASICs. Another reconfigurable-based obfuscation was introduced in the same year, which utilized LUTs to obfuscate the design. Several reconfigurable-based obfuscation techniques which exploit LUTs to hide circuits were proposed from 2018 to 2019. The year 2019 marked a significant breakthrough, with the introduction of another reconfigurable-based obfuscation technique where the transistors are programmed to recover the functionality of the circuit. In the same year, the first Boolean satisfiability problem (SAT) attack on reconfigurable-based obfuscation was also introduced, which marked a notable development in the field.

In that same year, a novel reconfigurable-based obfuscation technique was proposed that utilized eFPGAs for obfuscation purposes [23]. Researchers referred to this as"eFPGA redaction" which is another term used to describe reconfigurable-based obfuscation techniques that utilize eFPGAs as an obfuscation asset. An example of eFPGA redaction is illustrated in the right panel of Fig. 5 which leverages an eFPGA macro to obfuscate the circuit. This level of obfuscation is inserted during the physical synthesis of a design, therefore this type of technique demands certain skills in physical implementation that are far more complicated than the insertion of XOR/XNOR in a netlist (as illustrated in Fig. 4). In summary, the final layout will look like the one illustrated in the right panel of Fig. 5. Recalling again, the generated layout must be shared with the untrusted foundry for manufacturing. However, the bitstream for the eFPGA IP will not be shared with the untrusted foundry. The example highlighted in Fig. 5 demonstrates how a single module is turned into a reconfigurable part to offer obfuscation.

Between 2010 to 2020, there were multiple reconfigurable-based obfuscation techniques that were offering obfuscation with a high level of security. However, none of them have demonstrated silicon validation. In 2020, the authors of [44] demonstrated what is likely the first proof of concept in silicon. It is noteworthy that the traditional FPGA uses SRAM bit-cells to reconfigure logic. In this work, however, authors utilized daisy-chained flip flops to keep the bitstream.

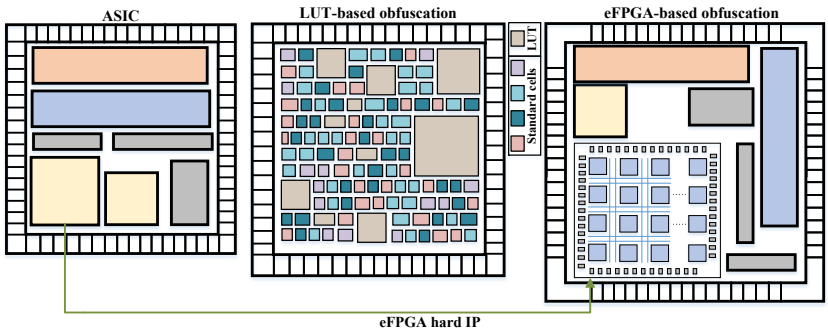


Fig. 5. Interpretation of eFPGA-based obfuscation technique [76]. Notice the increase in die area and the modified floorplan.

In order to summarize the advances in reconfigurable-based obfuscation, we present the publication trend of techniques and attacks in Fig. 6. It is clear from the trend that a great number of

reconfigurable-based obfuscation techniques have been proposed to protect the IP against various hardware security attacks [12, 14, 49, 50, 54, 80, 85]. The trend of defense techniques is continuously growing, and the research community has recently displayed significant interest in the attacks. Therefore, there are initial attempts at breaking obfuscation schemes based on reconfigurable devices. In most cases, even when using state-of-the-art attacks, adversaries appear to only be able to analyze the behavior of obfuscated circuits without much success. We elaborate on the details of these attacks in Section 4. Moreover, the substantial amount of publications has enabled us to classify the reconfigurable-based obfuscation techniques, which we further interpret in Section 2.2.

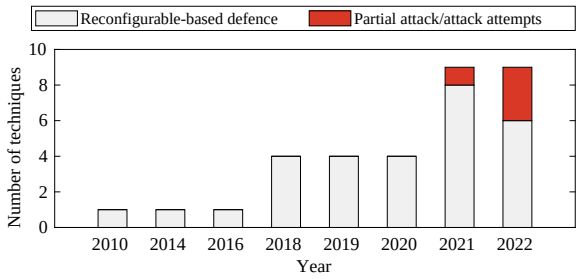


Fig. 6. Publication trend and attacks for reconfigurable-based obfuscation techniques.

2.2 Classification of reconfigurable-based obfuscation techniques

As discussed in the previous subsection, the research community has shown a growing interest in implementing obfuscation using reconfigurable logic – we can expect significant progress in this technique due to the apparent robust security guarantees it offers. It is thus important to adopt a unified classification and terminology the technique. As shown in Fig. 7, by carefully analyzing all the proposed techniques, we broadly classify them based on three important factors: 1) The technology used; 2) Element type; and 3) IP type.

2.2.1 Technology Used. Numerous reconfigurable-based obfuscation techniques have been proposed to date. In a general sense, it is the LUT that actually enables logic to be obfuscated via reconfigurability. During the obfuscation process, the selected internal gates from the design are mapped onto LUTs. As depicted in Fig. 7, there are several technologies available to store the key bits of the aforementioned LUTs. As shown in Fig. 7, we categorize them into SRAM-based LUTs [14, 25, 44, 49, 50], non-volatile memory (NVM)-based LUTs [12, 40–43, 80, 85], and Others [3, 4, 17, 22, 23, 32, 53, 54, 60, 65, 66, 68, 69, 76]. In Others, various technologies are available for programming the LUTs, including the spin transfer torque (STT)-based LUTs that exploit magnetic technology, flip-flop (FF)-based LUTs, programming pf individual transistors such as in a TRAnsistor-level Programming (TRAP) fabric, and programming of eFuses.

Among these technologies, SRAM-based LUT has garnered significant attention to store the key bits. SRAM-based LUTs are often considered due to their programmability, reconfigurability, fast access time, low power consumption, smaller area, scalability, and ease of testing. These characteristics make them a preferred choice for implementing logic functions in FPGA designs and therefore a natural choice for obfuscation as well. Fig. 8 showcases an instance of a 2-input LUT and the various feasible functions it may implement. The 2-input LUT has the capacity to implement 16 distinct functions, as enumerated in the table presented in Fig. 8.

In contrast to SRAM-based LUTs, the implementation of NVM-based LUTs relies on Non-Volatile Memory technology. The obvious advantage is that the programming remains even if the device

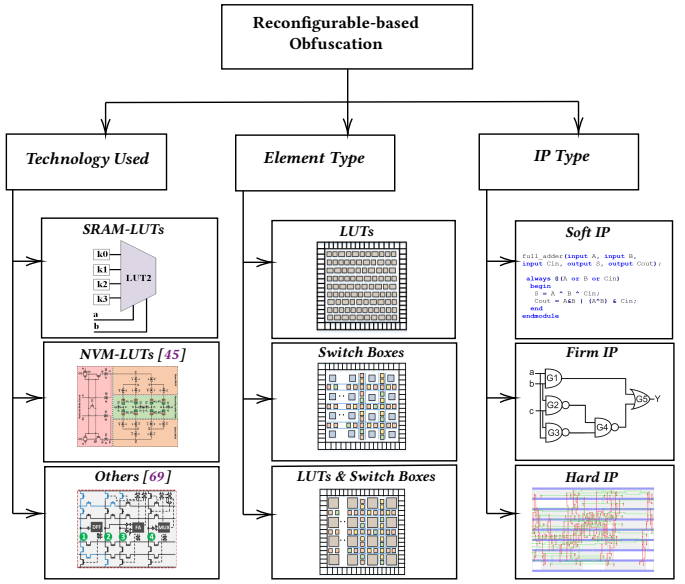
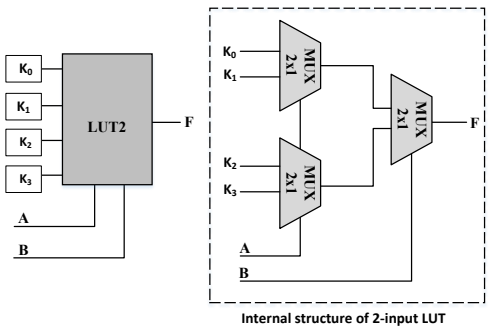


Fig. 7. Classification of Reconfigurable-based Obfuscation

is powered off. Compared to SRAM-based LUTs, however, they have some drawbacks, such as slower access time, and limited, complex programmability. However, it offers high-density storage elements. STT-based LUTs have been considered to design very robust and reverse engineering resilient LUTs. Contrary to NVM-based solutions, utilizing an FF-based LUT implementation renders the framework technology-agnostic, simplifying the process of floorplanning and placement tremendously. However, it does not achieve the same bit density as an SRAM-based solution does.

The TRAP fabric is a special case; it was introduced to obfuscate the design's intent by programming a sea of transistors. Efuses are essentially one-time programmable fuses that are blown to permanently program a specific configuration of the LUT. This differs from SRAM-based LUTs, which require reconfiguration each time the device powers up. Efuses offer different security properties, while they disallow reprogramming, they potentially expose the programmed values to reverse engineering by an end-user (but not by the foundry).



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

Fig. 8. The 2-input LUT realizes all 2-input logic functions depending on the configuration bits K_0-K_3.

2.2.2 Element Type. Concerning the element type, the reconfigurable-based obfuscation technique can leverage solely LUTs as in [3, 4, 12, 14, 40–42, 44, 49, 50, 80, 85]. The majority of the surveyed works take this approach, where the logic elements are obfuscated by LUT use only. As an example, the layout of this type of technique is illustrated in the top center of Fig. 7. This demonstrates a regular structure where the majority of the circuit consists of LUTs. On the other hand, there are some techniques that rely solely on exploiting switch boxes to obfuscate the design [17, 69] by obfuscating connections between elements, as shown in the center of Fig. 7. Here, the layout is a combination of switch boxes and standard logic, which is indicated with blue color. Finally, the reconfigurable-based obfuscation techniques that utilize both LUTs and switch boxes are commonly referred to as FPGA redaction techniques [22, 23, 25, 32, 43, 53, 54, 60, 65, 66, 68, 76].

An example of reconfigurable-based obfuscation utilizing eFPGA is described in Section 2.1. It seems to be the case that obfuscating both LUTs and switch boxes has additional benefits in terms of enhancing the security of the design. By obscuring routing blocks, attackers cannot analyze routing patterns and infer functionality from them, thus offering another opportunity to thwart the otherwise powerful Boolean satisfiability (SAT)-based attacks. The reconfigurable-based obfuscation techniques can also be classified by the form they are delivered an intellectual property (IP), which we elaborate on in the next section.

2.2.3 IP Type. Reconfigurable-based obfuscation techniques require that the designers perform several additional steps in their design flow. Selecting the critical modules to be redacted or selecting suitable gates to be replaced using LUTs are examples of additional steps. Hence, the process of obfuscation can be carried out at any of the three forms of IP, namely soft IP, firm IP, or hard IP, as illustrated in Fig. 7. Obfuscation carried out at the register transfer level (RTL), such as in Verilog or VHDL code, or any high-level codes can be classified as obfuscation at the soft IP level [22, 23, 32, 53, 54, 60, 66]. The Verilog or VHDL IP is typically input by the designer and then modified in a way that results in design obfuscation. On the other hand, it can also be carried out with high-level codes, such as C/C++, followed by high-level synthesis. In this process, user-defined algorithms are executed to identify the optimal portions of modules to be obfuscated.

The next category involves implementing obfuscation at the firm IP level, which is the most commonly utilized approach [3, 4, 12, 14, 25, 40–44, 49, 50, 53, 65, 68, 69, 80, 85]. In this process, the post-synthesis netlist file is used as an input and it generates an obfuscated netlist as the output.

Finally, the obfuscation can also be performed at the hard IP level where a crucial part of the design is identified and mapped into a hard IP. This entails using eFPGAs [17, 76] as the means to carry out the obfuscations, one of the examples is illustrated in Fig. 5 and explained earlier at the end of Section 2.1.

3 EXISTING RECONFIGURABLE-BASED OBFUSCATION TECHNIQUES

The section highlights a few of the most prominent reconfigurable-based defense techniques. It also presents a comparison among them.

As the primary form of comparison among the various techniques, let us look into area, power, and delay overheads, as presented in Table 1. The majority of papers surveyed report their results considering PPA as a percentage increase over a baseline, which is also the approach we have followed. Table 1 provides valuable insights into the impact of different obfuscation methods on key design metrics and can help guide the selection of an appropriate obfuscation technique based on specific design requirements/constraints. The techniques presented in Table 1 are selected to highlight the extremes in PPA overheads. Moreover, the selected techniques also exhibit different flavors of obfuscation based on the technology used and element type. The results reported in Table 1 represent the techniques that were compared with the baseline designs. Instead of highlighting

the increase in PPA relative to baseline designs when reporting their results, the majority of these techniques simply focus on reporting the PPA of obfuscated designs.

Notice that Table 1 is divided into two parts: CMOS and Emerging Technologies. As far as technology is concerned, the majority of the techniques that utilize CMOS-based LUTs are referred to as CMOS technology (CMOS TECH) in Table 1. However, emerging technologies that are used to implement LUTs, such as spin-transfer torque (STT), magnetic tunnel junction (MTJ), spin-orbit torque (SOT), and magnetic-random access memory (MRAM), are called emerging technologies (Emerging TECH). As a general rule of thumb, these two are very distinct from one another, therefore, trends and comparisons are more meaningful within the same technology class.

It is noteworthy that the reconfigurable-based obfuscation techniques using CMOS technology provide an analysis of large designs or benchmarks which highlights an increase in the PPA. However, most of the emerging technology-based techniques have only been evaluated on small design or ISCAS benchmarks, they exhibit a significant increase in PPA, as seen in references [3, 4, 49, 53]. It is noticeable that some of the overheads are very large. For instance, in [12], the authors reported a 95.06x increase in the area of the c2660 circuit from the ISCAS'85 benchmark suite. The work in [14] suggests LUTs for obfuscation purposes and provides several replacement strategies to secure a netlist. On the other hand, [49] utilizes an SRAM-LUT structure as configurable logic for gate replacement, incorporating a 2^n-to-1 MUX and 2^n configuration memory cells. This approach facilitates the dynamic configuration of the replaced gates. Nonetheless, using SRAM for logic obfuscation generally incurs a relatively high area overhead, as shown in Table 1. In [17], the authors have employed an eFPGA to redact the design and they considered the integration of different fabric sizes in PicoSoC. All three variants present a non-linear percentage increase in PPA with respect to the fabric size. A similar type of approach is employed in [53], with substantially greater area and power overheads in comparison to other techniques. However, the delay overhead is significantly considerable. Additionally, it is essential to consider the security vs area and performance trade-offs, as given in [3].

Almost all the techniques aim to redact the most sensitive part of the design in a modular approach. The flexibility to obfuscate any part of the design, thereby crossing module boundaries, is presented in [3, 4]. The authors almost obfuscated the majority of the gates in the design, approximately 80-90%, which then yields high PPA penalties. Every technique that lies in the category of emerging technology could be referred to as a hybrid because they utilize standard CMOS technology along with one of the STT, MTJ, SOT, or MRAM devices. Table 1 provides an overview of the techniques that belong to these technologies, LUT being used as an element type designed using a hybrid of CMOS and emerging technology. A similar strategy could be employed for the combination of both switch box and LUTs as a hybrid technology. To further elaborate, we would like to emphasize a few insights about the hybrid technologies mentioned earlier. In order to realize a Spin Transfer Torque (STT) device, stacked multilayer sandwich structures are typically used. According to Fig. 9, there are a number of layers in the structure, including an oxide tunnel barrier, a free magnetic layer, and a pinned magnetic layer. An external magnetic field or a spin-polarized current J_{read} flowing through the junction can switch the magnetization direction of the free layer from a parallel to an antiparallel state (P to AP). These states, in turn, can represent a logic-1 or a logic-0. However, the most important aspect of the device depicted in 9 is that the MTJ itself does not compete with standard cells for area since it resides between two metal layers.

MTJ integration requires little die area, except for the CMOS circuits and contacts used to connect MTJs to MOS transistors. The MTJs does come with certain operational challenges. Asymmetry in write and read operations results in a difference in operation energy and delay, requiring a higher current for completing write operations. Spin-orbit interaction has recently been explored as an alternative write approach to overcome these bottlenecks, details area available from [19].

Table 1. Comparison of Reconfigurable Obfuscation Techniques

	Technique - Ref	Circuits	Area (%)	Power (%)	Delay (%)
CMOS TECH	eRECONF LOGIC - [49]	IDU	1595.0	942.8	165.0
		LEON2	34.7	6.7	131.0
	eFPGA REDAC - [17]	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 - [53]	RISC-V	89.0	40.0	136.0
		GPS	39.0	46.0	0.0
	hASIC - [4]	SHA-256	2231.8	717.0	353.8
	CAD-hASIC - [3]	SHA-256	4192.0	2051.9	150.0
	SILICON-LUT - [44] †	Multiple	LO: 7.0 MO: 14.0 HO: 262.0	LO: 0.0 MO: 3.5 HO: 17.8	0.0
Emerging TECH	Hybrid STT-LUT - [80]	ISCAS	min: 0.1 avg: 6.4 max: 20.6	min: 0.7 avg: 24.96 max: 82.11	min: 0.0 avg: 28.4 max: 82.3
	MTJ-STT-LUT - [40]	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 - [85]	ISCAS/MCNC	min: 2.5 avg: 12.2 max: 25.4	–	min: 0.0 avg: 20.4 max: 41.8
	SOT-LUT-32i_G - [85]	ISCAS/MCNC	min: 6.7 avg: 17.7 max: 27.2	–	min: 0.0 avg: 28.5 max: 47.5
	SOT-LUT-64i_G - [85]	ISCAS/MCNC	min: 15.2 avg: 22.2 max: 27.2	–	min: 4.2 avg: 36.1 max: 78.2
	CGRRA - [22]	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 - [68]	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).

The authors in [43] proposed reconfigurable logic and interconnects (RIL)-Blocks that leverage MRAM-based LUTs with MTJ technology and routing-based obfuscation. As illustrated in Fig. 10, each cell is accessed via A and B, while the write operation is controlled by $\overline{WE}$ signals. In each memory cell, the MTJs change complementary to each other during write operations. According to input signals A and B, output nodes O and O direct the appropriate output to the select tree

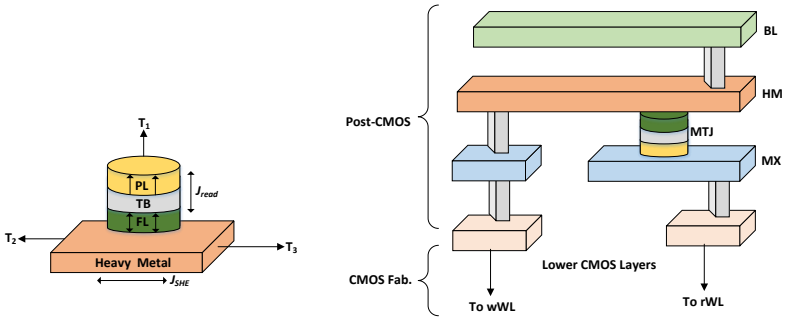


Fig. 9. Physical structure demonstration of SHE assisted MTJ switching mechanism (adapted from [85]).

MUX using the RE and RE signals. The RIL-Blocks are built using commercially available STT-MTJ technology to provide the desired obfuscation.

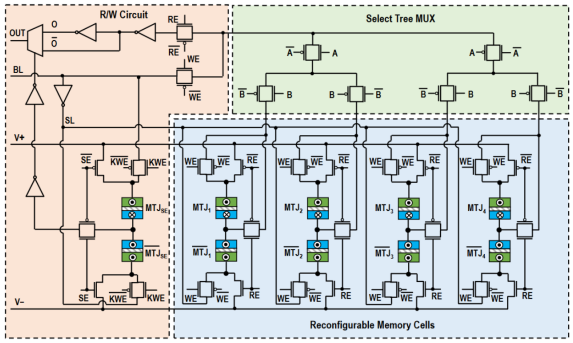


Fig. 10. The circuit diagram of the proposed 2-input MRAM-based LUT) that utilizes STT-MTJ devices [43]

The technique in [80] is the first one to propose hybrid-STT LUTs and its area overhead is pretty small but the power and delay overheads are around 82%. The approach given in [85] involves examining the internal gates of each class to identify the gate that can be obfuscated with minimal design overhead and path delay. In this approach, the authors present a cluster of 16, 32, or logic gates which are replaced with 2, 3, and 4-input LUTs. While the area of this technique is comparable to [80], its maximum overhead is nearly equal for groups of 16, 32, and 64. The approach in [40] does not incur any performance overhead and it exhibits almost 90% increase in area. The method given in [22] also validates their technique on small circuits but the area overhead is large even for a very small circuit that executes a sorting algorithm. The PPA overheads in [68, 69] are very small, this approach has the lowest PPA in reconfigurable-based obfuscation with the same remark of validation on small circuits.

Now, we will briefly explore how CMOS technologies can be used to obfuscate circuits. LUT-based obfuscation is the dominant technique used in CMOS technologies. In [43], the authors presented a novel approach to thwart various types of attacks by utilizing both reconfigurable interconnect and logic blocks. The researchers presented their security analysis without highlighting PPA overheads. The authors in [76] proposed RTL-based partitioning in tandem with eFPGA redaction for better obfuscation. However, the authors in [23, 32, 66] proposed a very similar partitioning scheme at the behavioral level, and demonstrated an automated partitioning flow for behavioral descriptions

for high-level synthesis (HLS). This technique is associated with high PPA overheads and the explanation of this technique is detailed in Section 2. A similar approach was implemented in [53] to obfuscate portions of the RISC-V control path and its overhead is mentioned in Table 1. In [25], the authors presented a LUT-based obfuscation algorithm that optimizes the replacement locations of LUTs and the input signals they receive in order to achieve resiliency against SAT attacks while minimizing overhead. The authors of [50] have proposed a different scheme called LUT-Lock that focuses the effort on a minimal set of primary output pins to increase the obfuscation difficulty. To accomplish this, they decided to focus on fan-in with a few primary outputs for obfuscation and selected gates that are connected to the smallest number of output pins. Additionally, gates that exhibit less control over primary inputs are preferable for obfuscation.

As the research on this type of techniques has matured, authors started to consider the trade-off space between security and PPA. The research in [41] has demonstrated that circuits with a small LUT input size (e.g., 2-input LUT) can be easily de-obfuscated. Their research demonstrates that the input size of LUT is the most influential and straightforward factor in achieving SAT resiliency. Though LUT-based obfuscation provides high-security levels, it results in prohibitive PPA overhead [49]. Recently, another LUT-based research work in [3] presents a security-aware computer-aided design (CAD) flow, which utilizes a combination of MUXs, LUTs, and FFs for obfuscation. It is compatible with the standard cell-based physical synthesis flow. The approach explores a midpoint between pure FPGA and pure ASIC design to generate heavily obfuscated designs that combine static parts with reconfigurable parts, which they them term a "hybrid ASIC" (hASIC). The results illustrate that better obfuscation could be achieved with slightly high hardware overhead.

Interestingly, the authors in [44] demonstrated a low overhead digital IC design obfuscation flow which is compatible with existing Electronic Design Automation (EDA) tools. They have also fabricated an IC to prove the concept. The proposed method was demonstrated for both non-volatile internal (efuse) and volatile external (SRAM) LUT key configurations to showcase its flexibility. Using the fabricated silicon, the actual design overhead, in terms of area, performance, and power, was measured. The chip was evaluated for various levels of security and showed that the SAT attack could be thwarted in the low obfuscation case with minimal overhead and in the medium obfuscation case with a maximum of 14% overhead, allowing for a significant SAT runtime margin.

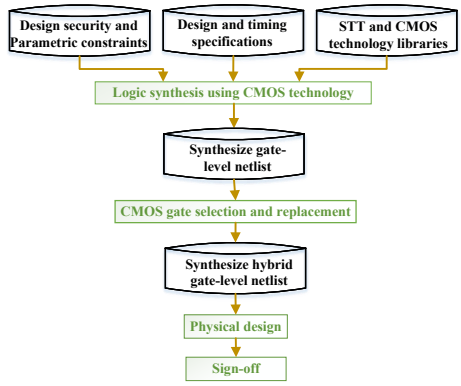


Fig. 11. Security-driven hybrid STT-CMOS design flow (adapted from [80])

Now, we will briefly explore how emerging technologies can be used to obfuscate circuits. In [42], the authors studied the previously proposed LUT-based obfuscation schemes and proposed a customized STT-LUT-based obfuscation with two different variants: LUT+MUX-based obfuscation,

and LUT+LUT-based obfuscation. This combination assists them in elevating security provided by logic obfuscation while concurrently creating SAT-hard instances. But, the work in [40] explored the design space for four crucial design factors that impact the design overhead and security of hybrid STT-LUT-based obfuscation: (1) LUT technology, (2) LUT size, (3) number of LUTs, and (4) replacement strategy as illustrated in Fig. 11. It is concluded in [40] that, among the four studied parameters, the input size of LUT is the most influential and straightforward factor in achieving SAT resiliency. The authors in [45] proposed a multi-layer defense mechanism using a combination of a Symmetrical MRAM-based LUT (SyM-LUT) and STT-MTJ-based LUT as shown in Fig. 12 [80]. SST structures require a high write current to switch the magnetization direction, SOT structures can achieve the same result with significantly reduced current. This makes them a promising candidate for low-power and high-speed data storage and processing applications. Therefore, the authors in [85] utilized the hybrid SOT-CMOS circuits to realize the reconfigurable logic with a lower write current among LUTs programming operations with smaller hardware overhead as shown in Table 1. As an example, Fig. 11 illustrates the STT-CMOS hybrid design flow. In order to incorporate hybrid technologies into the design flow, additional parameters and processes must be considered. This approach differs from conventional design flows in that it includes additional steps, such as replacing gates and synthesizing the netlist again.

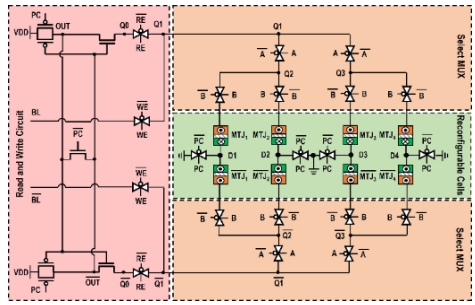


Fig. 12. The circuit-level diagram of 2-input SyM-LUT using STT-MTJ devices [45]

The research community has modified the research aspect of reconfigurable domains by incorporating dynamic morphing to resist SAT attacks. The Giant Spin Hall Effect (GSHE) [54] and Magneto-Electric Spin-Orbit (MESO) [60] concepts enable polymorphism which allows for runtime configuration of Boolean functions. Dynamically changing between states during runtime offers obfuscation, hence neutralizing the threat of a SAT attack. However, randomly morphing from one state to another limits the applicability of the obfuscation to applications like image processing that can tolerate a certain degree of errors [60]. In [22], a variant of an existing architecture called coarse-grained runtime reconfigurable array (CGRRA) is used to selectively map different portions of the design into a reconfigurable block. Different portions of a design, executed at different clock cycles, can therefore be mapped onto the same CGRRA without requiring additional area. The overall hardware overhead of this scheme is mentioned in Table 1. Examples of this architecture include the Stream Transpose Processor from Renesas Electronics [61] and Samsung's Reconfigurable Processor [38]. Another approach that offers reconfiguration at the transistor level is presented in [69]. The solution exploits "sea-of-transistor" architecture, supporting the implementation of custom cell libraries and facilitating fabric time-sharing. Here, the authors present a partitioning flow for RTL descriptions. As shown in Table 1, the results are promising to note that this solution not only results in an order of magnitude smaller PPA overhead but also increases the complexity of the

design [69]. However, this approach presents some drawbacks: it can make testing and simulation more challenging than other solutions, and since the configuration is done at the transistor level, the resulting bitstream size will become extremely large (see Fig. 1 of [68]).

Overall, the majority of reconfigurable-based techniques leverage either a reconfigurable element or the eFPGA redaction as depicted in the right panel of Fig. 5. The left panel of Fig. 5 represents the ASIC that requires obfuscation, while the center panel demonstrates the LUT-based obfuscation approach utilized for obfuscation. The eFPGA-based obfuscation is explained in Section 2.1. In this approach, one of the crucial modules is transformed into reconfigurable logic, and the other modules are converted into standard cells. Additionally, the reconfigurable elements are dispersed throughout the layout, whereas the eFPGA macro is placed in a specific place, resulting in a concentrated reconfigurable logic.

4 SECURITY ANALYSIS: THREAT MODELS AND EXISTING ATTACKS

This section presents an overview of state-of-the-art attacks and their associated threat models. Additionally, a comparison of different attacks and a thorough security analysis are presented. Furthermore, this section provides a comparative analysis of logic locking and reconfigurable-based obfuscation techniques.

4.1 Threat models

In the context of adversarial modeling, threat models can be classified into oracle-guided and oracle-less. This terminology has been borrowed from the field of Logic Locking, and it is equally applicable to attacks on reconfigurable-based obfuscation techniques. An oracle-guided attack involves a reverse-engineered netlist and a functional chip, which is commonly referred to as an oracle. As depicted in Figure 13, oracle-guided attacks typically employ Boolean satisfiability (SAT) techniques based on two copies of the locked netlist as part of a miter-like circuit.

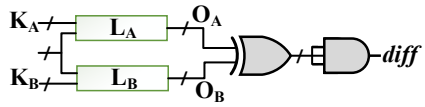


Fig. 13. The Miter circuit, which is utilized in the SAT attack to identify DIPs (adapted from [72]).

Algorithm 1 outlines the SAT attack methodology. A miter-like circuit is created using the locked netlist in the first step of the attack. In order to obtain the IC output O_d, SAT solvers are used to derive a DIP I_d using the CNF formula of the miter. With the constraint on the *diff* signal, the miter circuit is formulated as $L(I, K_A) = L(I, K_B)$. In each iteration of the attack, a DIP I_d is obtained by identifying a satisfying assignment to the miter CNF. The output O_d of the functional IC is recorded, and the CNF formula is enriched with additional clauses based on the (I_d, O_d) input-output pair. When UNSAT is returned by the SAT solver, the attack is concluded. The formula $L(I, K_A)$ is solved in the final step of the SAT solver, resulting in the precise key K_C. Repeatedly applying DIPs until the search space has been exhausted allows the attacker to obtain the correct key values.

Oracle-less attacks are a set of techniques that do not require accessing an oracle or a functional chip. There are many types of oracle-less attacks, and most of them are based on structural analysis [7, 8, 21]. Although Oracle-guided attacks, such as the SAT attack, are applicable to some scenarios, they may not be effective in situations where the search space for the key bits is large. Recently, a few attacks have been proposed for reconfigurable-based obfuscations, which will be discussed in the following paragraphs.

Algorithm 1: SAT attack algorithm [72]

Input : Locked netlist $L(I, K)$, functional IC $F(I)$
Output: Correct key K_C

while $I_d = SAT(L(I, K_A) = L(I, K_B))$ **do**
  $O_d = F(I_d)$; // Query the oracle
  $L(I, K_A) = L(I, K_A) \wedge (L(Id, K_A) = O_d)$; // Augment clauses
  $L(I, K_B) = L(I, K_B) \wedge (L(Id, K_B) = O_d)$
$K_C = SAT(L(I, K_A))$;

In [24], authors propose a "predictive model attack" that substitutes the exact logic mapped onto eFPGAs with a synthesizable predictive model, which attempts to replicate the behavior of the exact logic. This is an oracle-guided attack that relies on machine learning techniques to build a predictive model. The adversary is a proficient IC designer with the expertise and tools necessary for understanding this layout representation. The threat model of [24] is given below:

- Access to the inputs and outputs of an eFPGA can be facilitated by utilizing the scan-chain around the eFPGA. This approach is widely supported by all commercial eFPGA vendors.
- In the absence of a scan chain, a probing attack can be employed as an alternative method [79]. Given the regular structure of the eFPGA, a probing attack would be relatively straightforward.
- Access to the CAD flow required for programming the eFPGA can be reasonably assumed, since very likely the eFPGA is licensed from a known vendor (e.g., Achronix, Menta, or Quicklogic).

The next proposed attacks are oracle-less attacks, namely the "structural analysis attack" and the "composition analysis attack" [4]. Because [4] proposes a hybrid FPGA-ASIC solution, it is assumed that an attacker has access to a reversed engineered netlist where a portion of the logic is in the clear. The security of such style of reconfigurable-based obfuscation depends on how much information is exposed in this fully exposed portion of the logic. In [3], a follow-up work, authors present the threat model for their attacks as follows:

- The adversary may aim to learn the circuit's intent instead of recovering the entire IP (e.g., "Is this an AES cryptocore?"). To achieve this objective, the adversary does not need to recreate the correct bitstream, instead observations of a partially obfuscated logic give away hints about its intent.
- The adversary can recognize individual standard cells, therefore the gate-level netlist of the obfuscated circuit can be easily recovered following [58].
- The ability of the adversary to identify reconfiguration pins [73, 88] allows for easy enumeration of all LUTs and also their programming order.
- The adversary can form clusters of standard cells existing in the static logic and transform them back into a LUT representation[2].

The last attack proposed in the literature is called "break & unroll attack". It is capable of recovering the bitstream of eFPGA-based redaction schemes in a relatively short time even with the existence of hard cycles and large size keys [62] and, interestingly, this contrasts the common perception of eFPGA-based redaction schemes being secure against oracle-guided attacks [4, 16, 17, 27, 53]. In their threat model, it is assumed that all ICs are sequential circuits, thus the attacker can access the always-present scan chain. The attacker has complete access to the obfuscated netlist

[2]In other words, perfect LUT reconstruction is assumed.

and can obtain a functional circuit from the market as a black box, with the ability to derive correct outputs for given input vectors.

In general, the proposed FPGA-inspired obfuscation techniques were mainly evaluated against SAT-based attacks which are logical in nature. Physical attacks, such as side-channel attacks, are barely assessed [43, 45, 85]. Overall, there is a lack of security analysis for evaluating the FPGA-inspired obfuscation techniques against specific attacks. For well-established attacks, the analyses performed so far still have a large margin for improvement.

4.2 Attacks

FPGA-inspired obfuscation techniques, similar to the Logic Locking techniques, aim to protect IPs against different threats that appear at any phase in the globalized IC supply chain. The attack trend for the reconfigurable-based techniques is illustrated in Fig. 6. There is an obvious lack of developed attacks against reconfigurable logic mechanisms—currently, there are a few developed attacks presented in [24], [4], and [62].

The three main steps of a "predictive model attack" are illustrated in Fig. 14. In the first phase of the attack, the IC from the market is used to run applications that require the obfuscated hardware accelerator, and inputs and outputs of eFPGA are recorded for generating a predictive model. The latency of the obfuscated design is also recorded to ensure there are no timing issues when replacing the eFPGA portion with a predictive model. This text describes the second phase of the proposed attack, which involves searching for a predictive model that can be mapped in hardware on the eFPGA. The predictive model must fit within the eFPGA fabric, have outputs within the specified error threshold, and operate at the same frequency and latency as the original design. This phase is divided into three steps: model fitting, predictive model refinement, and automated MLP exploration. The output of this phase is a synthesizable C description of the predictive model that operates within the given constraints. The two predictive models considered are linear regression and multi-layer perceptron. The automated MLP explorer optimizes the MLP configuration to fit within the eFPGA and meet the error constraint. The predictive model generated in step 1 is further optimized in step 2 by obtaining the smallest possible model that operates within the specified error threshold.

The third and final step in the process of generating a predictive model for High-Level Synthesis (HLS) involves finding the smallest possible implementation of the predictive model with a latency equal to that of the exact version extracted in the first phase. This is achieved by setting different synthesis options combinations for the optimized synthesizable predictive model and generating the smallest implementation with latency L_{efpga}. The authors discuss the use of synthesis directives (pragmas) for synthesizing arrays, loops, and functions and how different combinations of these directives lead to a unique micro-architecture with specific area vs. latency trade-offs. The output of this stage is the pragma combination that leads to the smallest predictive model implementation (pragmaopt) and the newly optimized predictive model with exact latency as the exact obfuscated circuit (C_{Copt}). The final phase of the process generates an eFPGA bitstream to configure an eFPGA with the predictive model. The process includes HLS, logic synthesis, technology mapping, place and route, and eFPGA bitstream generation. The target synthesis frequency should be set to the same frequency at which the exact obfuscated circuit works to enable the unlocking of every manufactured IC. Despite the cleverness of the attack, it does bare a major limitation since it only applies to approximate computing. It is therefore unlikely that the eFPGA-obfuscated logic would implement any form of hardware-based cryptography since it has to be deterministic.

The authors of [3] proposed two different attacks: "structural analysis attack" and the "composition analysis attack". It is argued that, by exploiting the static portion of the design, including the frequency of specific LUT masking patterns, an adversary can extract information and reduce the

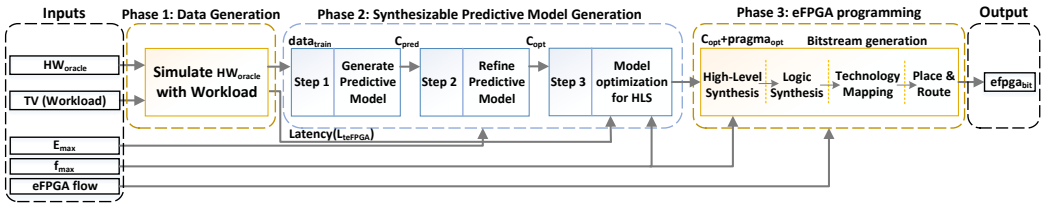


Fig. 14. The proposed attack flow composed of three main phases, the predictive model attack (adapted from [24]).

search space for the key that unlocks the design. They found that the combined number of unique masking patterns forms a set of 3376 elements, which reduces the global search space from 2^{64} to $3376 = 2^{11.72}$. They hypothesize that an attacker can exploit the frequency at which LUTs appear in a netlist to mount structural analysis attacks. The authors investigated this by analyzing the behavior of the frequency of masking patterns at different obfuscation levels. The results show that the adversary can estimate to some degree what masking patterns are the outliers. The results show that certain levels of obfuscation can either result in no correlation, a strong correlation to another circuit, or a correlation to itself. The study suggests that obfuscation can be targeted to confuse an adversary and shrink the key search space for obtaining the bitstream.

However, the success of the attack depends on the adversary's ability to reconstruct LUTs from the static part, the availability of enough datapoints in the database, and the design having static parts to begin with (eFPGAs would not have static parts). Additionally, the frequency of masking patterns can serve as a template for comparing different designs, as the composition of the LUTs within a design would enable a powerful template-based attack.

In [62], the "Break & Unroll attack" is proposed, which combines cycle breaking and unrolling to recover the bitstream of state-of-the-art eFPGA-based redaction schemes. The study highlights that the common perception that eFPGA-based redaction is secure against oracle-guided attacks is false and that additional research is required to secure eFPGA-based redaction schemes systematically. The overall flow of the Break & Unroll attack is given in Algorithm 2. The Break & Unroll algorithm consists of two main stages: breaking cycles and unrolling remaining cycles. If the first stage fails to reveal the correct key, the second stage, unrolling, is employed to neutralize the effect of hard cycles. The breaking phase involves breaking all cycles and adding a non-cyclic condition as a new constraint to the obfuscated circuit. The unrolling phase conquers the weakness of the breaking phase by unrolling a single cycle at a time, choosing a single feedback and adding a copy of every gate to the circuit. After unrolling one cycle, a new version of the obfuscated circuit is created and set W must be updated each time in the body of the attack algorithm.

4.2.1 State Space Reduction. The large search space of the bitstream makes reconfigurable-based obfuscation techniques inherently resilient to SAT attacks. However, a smart adversary could reduce the state space of a bitstream, as discussed in [4]. The authors explain that the synthesis tool may not explore all 2^n possible configuration patterns for n-input LUTs. Moreover, the adversary can exploit the frequency distribution of specific LUTs that are heavily preferred by synthesis engines. With the frequency characteristics of the LUT combined with statistical analysis, the search space can be reduced significantly. The significance of minimizing the search space lies in its criticality since it allows the adversary to identify the regular patterns and potential information inside the circuitry. Let us say the largest LUT in the design is a 6-input LUT. In this case, the search space is explored with the size of 2^{64} making patterns as illustrated in Fig. 15. The authors in [4] considered dozens of designs of varying complexity, size, and functionality. As a result,

Algorithm 2: Break & Unroll attack algorithm [62]

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

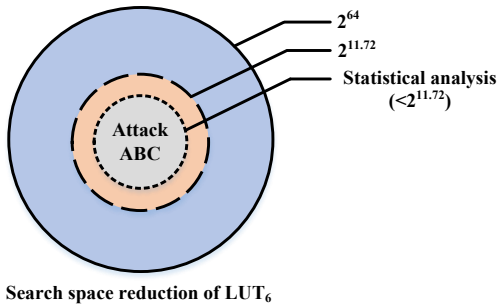


Fig. 15. The search space of LUT_6 as it shrinks with different attacks (adapted from [4]).

a total of $X = 2^{11.72}$ unique masking patterns were generated reducing the global search space significantly, as illustrated in Fig. 15. By incorporating synthesis hints along with statistical analysis, the adversary effectively narrows down the search space to another subsequent level. Furthermore, at that point, the adversary can apply his/her attack "ABC", whether it is a guided SAT-based attack or another clever attack. These considerations remain to be explored in future work.

4.3 Comparisons

Table 2 presents a list of reconfigurable-based obfuscation techniques, categorized according to the technology used in the implementations, as discussed in Section 3. The prohibitive costs of reconfigurable-based obfuscation techniques could make them impractical despite their attack

Table 2. Security comparisons of FPGA-inspired obfuscation techniques

Obf. Technique		Attack Resiliency			
		SA vs. AA	SAT	PSCA	Others
CMOS Tech	[14]	-	*no*	-	-
	[49]	AA	*no*	-	Hardware-Based Code and Injection Attack
	[50]	AA	*yes*	-	-
	[3]	AA	*yes*	-	AppSAT, Removal Attack, Composition Attack, Structural Attack, and SCOPE
	[44]	AA	*yes*	-	Removal Attack
	[76]	-	-	-	-
	[17]	AA	-	-	Icy-SAT
	[53]	AA	*yes*	-	-
	[32]	AA	*yes*	-	Brute Force Attack
	[16]	AA	*yes*	-	Icy-SAT, CycSAT, and Be-SAT
	[22]	SA	*yes*	-	Removal Attack
	[25]	AA	*yes*	-	Removal Attack
	[40]	AA	*yes*	-	Removal Attack, Scan-Based Attack, ATPG-Based Attack, Approximate Attack, and SMT-Based Attack
Emerging Tech	[68]	SA	*yes*	-	Brute Force Attack
	[54]	AA	*yes*	-	Double DIP
	[43]	AA	*yes*	*yes*	AppSAT, Removal Attack, Scan and Shift-Based Attack
	[45]	AA	*yes*	*yes*	Removal Attack, Scan and Shift-Based Attack
	[80]	SA	*no*	-	Brute Force Attack, Machine Learning-Based Attack
	[85]	SA	-	-	Brute Force Attack, Side-Channel Based Attack, Testing-Based Attack, Circuit Partition-Based Attack
	[42]	AA	*yes*	*yes*	-
	[41]	AA	*yes*	-	AppSAT, Removal Attack, Scan, and Shift-Based Attack
	[12]	-	*no*	-	-

SA and AA are abbreviations for Security Analysis and Applied Attack.

resiliency. As an attempt to decrease the overheads without compromising security, researchers have explored emerging non-volatile memory technologies for implementing reconfigurable logic. In addition, NVM technology can be used as a replacement for a tamper-proof memory *if* they lose their content upon invasive reverse engineering attempts [41]. Despite being promising technologies, these technologies are still under development and have not been widely adopted. For a detailed discussion, we refer our readers to [63].

As indicated in Table 2, some authors solely employed security analysis (SA) to assess the effectiveness of their defense techniques. If authors effectively attempted to attack their own defences, we marked them as applied attacks (AA). Here, we make a clear argument that both SA and AA have merits, but SA can be easily misinterpreted to suggest the techniques are more secure than they actually are. For instance, a classical SA discussion is the enumeration of the adversarial search space. But even large search spaces can be broken and this can only be shown through AA.

Furthermore, the resilience of the techniques was evaluated mainly against SAT-based attacks, as listed in the fourth column of Table 2. However, it should be noted that most of the techniques have not been validated against power side-channel attacks (PSCAs), especially in the case of

CMOS implementations. The authors of [85] and [68] discussed the resiliency of their techniques against testing-based attacks, circuit partition-based attacks, brute force attacks, SAT-attacks, and side-channel attacks. The column “Others” in the table shows the resilience of a given technique against other applied attacks. We have observed this trend over and over with Logic Locking mechanisms that were considered secure – some even proven to be secure – falling prey to *simple* attacks that were simply not initially considered.

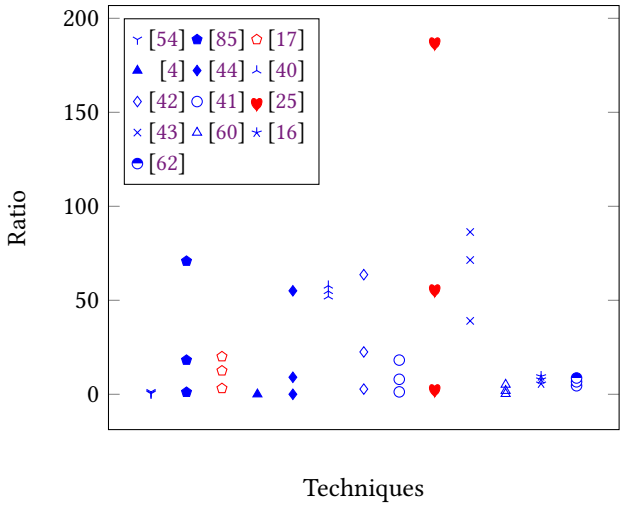


Fig. 16. Number of gates to the length of bitstream ratio in several techniques. The red color indicates that corresponding data was estimated, whereas the blue color indicates that the data was given in related references.

One of the major challenges is defining a security metric for reconfigurable-based obfuscation techniques. Currently, there is no specific or common security metric being adopted. To fairly assess the security of reconfigurable-based obfuscation techniques in this study, we adopted a common criterion: number of gates divided by the length of bitstream. Here the term “gates" refers to logic gates in the same was as in logic synthesis, meaning that a gate is a standard cell. For instance, if an author considered that a circuit with 1000 gates was adequately obfuscated with a bitstream of 128 bits, this circuit would have a metric of 7.81. Authors of defensive techniques are interested in maximizing the metric, such that a high number of gates can be protected with a small number of configuration/programming bits.

Often, authors report the area of obfuscated design, and from that, we can estimate the number of gates from the area using information about the technology used for implementation. Fig. 16 compares the gates count to the bitstream size ratio of benchmark circuits to be obfuscated as a security metric. We calculated the minimum, maximum, and average values of the ratio in most of the techniques[3] considering all benchmarks and bitstream sizes used originally by the developers for evaluating their techniques. As a rule of thumb, a large ratio is an indication of lower and costly security [50, 54, 68]. However, there are exceptions to this rule [50]. In addition, optimizing a design for desirable PPA overheads and security would complicate the security-bitstream size relation [16]. For some techniques, calculating the ratio was straightforward as the gates count and

[3]Not all surveyed papers provide enough information for this comparison, unfortunately.

the size of bitstream were provided by the authors [4, 41, 44]. However, for other techniques, the ratio either could not be calculated due to missing values [14], [50] or estimations were required for calculating it. For example, in [17], we needed to turn the area of a given circuit into the gate equivalent (GE) form to estimate the circuit gate counts.

We noticed two contradictory solutions in the eFPGA context: First, in many techniques, obfuscating nearly 10%, 20%, or 30% of a circuit gates count could suffice to ensure security (i.e., SAT attacks would run out of time) [14], [50], [54], [68]. Second, in [4], a much higher obfuscation rate is required for protecting the bitstream and design intent, e.g., obfuscation rates higher than 86% are recommended. As a result, the first solution would result in a large ratio when compared to the second solution as this is evident in Fig. 16. Yet, the objectives of the techniques are different, and so are the underlying threat models.

We have also surveyed the CAD tools developed by the authors of obfuscation techniques. In 41% of the proposed techniques, researchers developed custom tools to implement their approaches, whereas, in other techniques, researchers only relied on the standard CAD tools for implementing their approaches. The developed tools are mostly closed source and are claimed to be compatible with standard CAD flow as illustrated in Fig. 17.

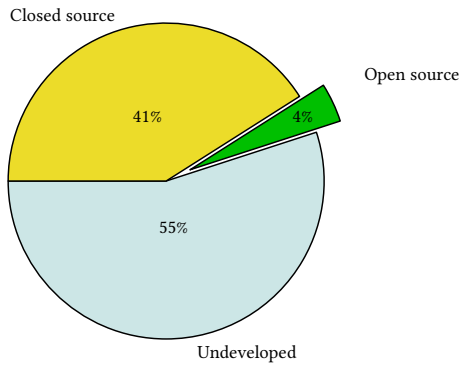


Fig. 17. Reconfigurable obfuscation techniques categorization based on developed/undeveloped CAD tools. The developed tools are split into open source/closed source.

5 DISCUSSION

In this section, we compare reconfigurable-based obfuscation and Logic Locking techniques and emphasize the lack of tape-outs demonstrating reconfigurable-based obfuscation techniques in silicon.

5.1 Reconfigurable-based obfuscation vs. Logic Locking

The Logic Locking concept has been around for more than a decade. In recent years, a cat-and-mouse game was established between developing and attacking Logic Locking techniques. This area of research has progressed rapidly, resulting in dozens of defense techniques and published attack strategies. The research line of reconfigurable-based obfuscation is not a new one, but it has received far less attention than Logic Locking. Both Logic Locking and reconfigurable-based obfuscation techniques share a common objective of protecting IP against supply-chain attacks.

Initially, Logic Locking techniques received more attention due to their practicality – the process of inserting XOR/XNOR gates in a netlist can be easily scripted. However, advances in Logic Locking

have coincided with emerging powerful attacks that have compromised design security. Recently, reconfigurable-based obfuscation has received more attention due to its high resiliency against attacks. However, this has raised questions about the security versus PPA trade-offs. As a result, researchers have been proposing solutions where the reconfigurable part is as small as possible [40]. Fig. 18 illustrates the conceptual difference between the Logic locking approach, shown on the left, that involves adding key gates to the original design, whereas the reconfigurable-based obfuscation approach, shown on the right, leverages reconfigurable logic elements. Logic Locking requires the correct configuration of the secret key while reconfigurable-based obfuscation techniques rely on loading the correct bitstream. These approaches are somewhat analogous and it is for this reason that attacks from Logic Locking domain can also be applied to reconfigurable solutions.

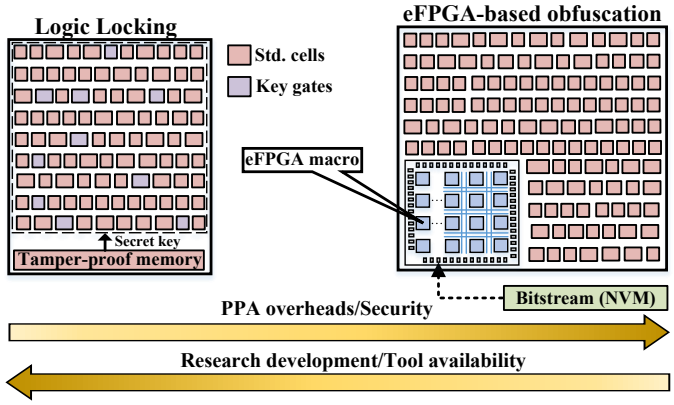


Fig. 18. Conceptual diagram of Logic Locking vs. eFPGA redaction

In Logic Locking, the locking mechanism is embedded in the netlist of design so the design is locked behind the secret key. This procedure could subject the design to several attacks, e.g., identifying and removing the lock, tampering with the lock, or identifying the secret key, compromising the secured design. The adversary can access the entire design consisting of the original intellectual property combined with the key gates. On the other hand, a selected portion of the design is hidden in reconfigurable-based obfuscation, meaning that a portion of the design is exposed to the adversary. This technique, which does not expose the entire design to the adversary, appears to offer a higher potential for securing ICs against supply-chain attacks.

In Logic Locking, circuit designers utilize the conventional CAD design flow. Reconfigurable-based obfuscation lacks a CAD tool flow that can support combined logic synthesis, timing analysis, and optimization of a mixed ASIC and FPGA-like design, unlike Logic Locking [53]. The right panel of Figure 18 shows an example of reconfigurable-based obfuscation, such as the eFPGA redaction technique and it employs custom tools that require significant effort to build. Reconfigurable-based obfuscation incurs higher security and overhead costs compared to Logic Locking, as shown by the arrows pointing to the right in Figure 18. It appears that Logic Locking techniques are more mature as well as more readily available tools. To summarize, reconfigurable obfuscation presents promising opportunities for securing digital integrated circuits against supply chain attacks, though many challenges remain.

5.2 Lack of Silicon Validation

There is a severe lack of frameworks to validate the security and functionality of obfuscated designs, as correctly highlighted in [44, 56]. Additionally, there is a minor effort toward silicon validation of

the techniques as a proof of concept. For example, only techniques presented in [44] and [68] were validated on silicon. Silicon validations of the techniques would provide an impetus to advance reconfigurable-based obfuscation as a promising solution for protecting digital ICs.

5.3 Future trends and Challenges

During our survey, an effort was made to present the findings of numerous research papers in a clear and unbiased manner. Yet, it is as evident as ever that the hardware security community lacks a unified benchmark suite and/or a common set of criteria. Frequently, researchers employ benchmark suites that enjoy popularity within the test community but are irrelevant to security. For instance, the ISCAS'85 suite lacks crypto cores and other real applications that are the cornerstone of the evaluation in this field. Furthermore, it is our belief that employing circuits that more accurately reflect current IC design practices, where IPs frequently comprise millions of gates and ICs house billions of transistors, would be highly beneficial for the community.

Within the research community, no standard criteria for assessing the security of reconfigurable-based obfuscation techniques is observed. Although it is confirmed that such techniques are resilient to various state-of-the-art attacks, the need for commonly accepted criteria to evaluate their security is an urgent need. While we utilized a ratio of gates to bitstream size, which is a reasonable but simple metric, it is noteworthy that for eFPGA estimations need to be converted into the number of gates. Concerning eFPGA macros available on the market, few companies have products available, including Achronix [5], Menta [52], and Quicklogic [57]. In tandem, there is also a discernible trend in the realm of ASIC design companies, such as AMD [2] and Intel [1], which both have acquired FPGA technology. This strategic move has enabled these entities to benefit from both ASIC and FPGA domains. With the convergence of these two technologies, there exists a possibility of integrating reconfigurable logic as an integral aspect of commercial production for security purposes. Today, the driver behind ASIC-FPGA hybrid solutions are design metrics, not security metrics.

It is also worth discussing the threat models proposed so far. The authors in [3] have established a robust threat model. Defining the capabilities of an attacker remains a complex task, requiring an understanding of their motivations, technical skills, and resource availability. Underestimating the attacker may result in ineffective defense strategies, whereas overestimating them could lead to unnecessary PPA overheads due to convoluted defense strategies. This challenge extends to reconfigurable-based obfuscation techniques and any other obfuscation-promoting approaches.

Another consideration in terms of attack is whether an attacker can leverage a partially recovered netlist. For instance, in a design that employs multiple instances of the same block, recovering one block correctly may enable the attacker to recover all other instances of the same block through a visual inspection of their structure [13]. This line of thinking is also applicable to datapaths and certain cryptographic structures that exhibit regularity. Consequently, a functional analysis of the recovered netlist can be combined with existing attacks to enhance correctly guessed connections.

Figure 17 clarifies that developing a custom tool is almost a mandatory part of the obfuscation process: 45% of the researchers have developed their custom tool, and 41% of them are closed source. This practice is a barrier to the research community and there must be an initiate to make the tools open-source for academic use. Future research could (and should) explore how these tools, along with reverse engineering and other methods mentioned previously, can be utilized to break the security of reconfigurable-based obfuscation. Logic Locking has become increasingly susceptible to SAT attacks. There have been many attacks and measures against these attacks, so the Logic Locking concept continues to evolve. In spite of the proposal of SAT-hard solutions for Logic Locking, perhaps reconfigurable-based obfuscation techniques will eventually take over due to its inherit security against SAT-attacks. It is predicted that eFPGA-based obfuscation will gradually

surpass Logic Locking in the coming years due to the fact that eFPGAs offers reliable security. Researchers in this field are likely to benefit from open-source macro generators for eFPGAs since only commercial solutions exist in the market. In the near future, there will probably be many publications based on this area of research.

6 CONCLUSION

Our study revealed a significant variation in the approaches to the reconfigurable-based obfuscation techniques among the surveyed works. However, we were able to classify the studies, providing a clear demonstration of the many interpretations of the technique, its attacks, and defenses. The study results were compiled to provide essential features, metrics, and performance results. Table 1 provides valuable insights into the impact of different obfuscation methods on key design metrics and can help designers to select an appropriate obfuscation technique based on specific design requirements and constraints. Design and fabrication are becoming more complex with emerging technologies. The research community needs to address the deficiency in the reconfigurable elements of these technologies, even though their developers claim these elements display high-reliability and low-area requirement.

The evaluation metrics and benchmark suites used varied widely, making direct comparisons challenging and, in some cases, impossible. The "ratio of gates to bitstream length" is a straightforward yet sound criterion for assessing the security of reconfigurable-based obfuscation techniques. Based on Fig. 16, most techniques fall within 0-50 range for this ratio, indicating varying degrees of security. Currently, there are a few developed attacks presented in [24], [4], and [62], and no attack has yet been found that can achieve perfect success. By leveraging state space reduction along with other attack strategies, likely further advances in attack results can be achieved.

It is an urgent need to conduct a comprehensive evaluation of critical security aspects to ensure the viability of the many proposed techniques. In the meantime, the research community must also validate the techniques on silicon to quickly eliminate approaches that are not viable, either from a technology standpoint of from a PPA overhead perspective.

REFERENCES

[1] 2015. *Intel Acquisition of Altera.* https://newsroom.intel.com/press-kits/intel-acquisition-of-altera/

[2] 2020. *AMD Acquires Xilinx.* https://www.amd.com/en/corporate/xilinx-acquisition

[3] Zain Ul Abideen, Tiago Diadami Perez, Mayler Martins, and Samuel Pagliarini. 2023. 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* (2023), 1–1. https://doi.org/10.1109/TCAD.2023.3244879

[4] Zain Ul Abideen, Tiago Diadami Perez, and Samuel Pagliarini. 2021. From FPGAs to Obfuscated eASICs: Design and Security Trade-offs. In *2021 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. 1–4. https://doi.org/10.1109/AsianHOST53231.2021.9699758

[5] Achronix Data Acceleration. 2022. Speedcore Embedded FPGA IP. https://www.achronix.com/product/speedcore

[6] Shah Agam. 2022. Europe, US warn of fake-chip danger to national security, critical systems. https://www.theregister.com/2022/03/18/eu_us_counterfeit_chips/

[7] Abdulrahman Alaql, Domenic Forte, and Swarup Bhunia. 2019. Sweep to the Secret: A Constant Propagation Attack on Logic Locking. In *2019 Asian Hardware Oriented Security and Trust Symposium (AsianHOST)*. 1–6. https://doi.org/10.1109/AsianHOST47458.2019.9006720

[8] Abdulrahman Alaql, Md Moshiur Rahman, and Swarup Bhunia. 2021. SCOPE: Synthesis-Based Constant Propagation Attack on Logic Locking. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 29, 8 (2021), 1529–1542. https://doi.org/10.1109/TVLSI.2021.3089555

[9] Altera Corporation. 2015. *Stratix V Device Handbook.* https://www.intel.com/programmable/technical-pdfs/683665.pdf

[10] AMD Xilinx. 2022. AMD Xilinx Adaptive SOCs. https://www.xilinx.com/products/silicon-devices/soc.html

[11] AMD Xilinx. 2023. Alveo U50 Data Center Accelerator Card. https://www.xilinx.com/products/boards-and-kits/alveo/u50.html

[12] Aliyar Attaran, Tyler David Sheaves, Praveen Kumar Mugula, and Hamid Mahmoodi. 2018. Static design of spin transfer torques magnetic look up tables for ASIC designs. In *Proceedings of the 2018 on Great Lakes Symposium on VLSI*. 507–510.

[13] Giorgi Basiashvili, Zain Ul Abideen, and Samuel Pagliarini. 2022. Obfuscating the Hierarchy of a Digital IP. In *Embedded Computer Systems: Architectures, Modeling, and Simulation*, Alex Orailoglu, Marc Reichenbach, and Matthias Jung (Eds.). Springer International Publishing, Cham, 303–314.

[14] A. Baumgarten, A. Tyagi, and J. Zambreno. 2010. Preventing IC Piracy Using Reconfigurable Logic Barriers. *IEEE Design Test of Computers* 27, 1 (2010), 66–75. https://doi.org/10.1109/MDT.2010.24

[15] Jürgen Becker and Manfred Glesner. 2001. A parallel dynamically reconfigurable architecture designed for flexible application-tailored hardware/software systems in future mobile communication. *The Journal of Supercomputing* 19, 1 (2001), 105–127.

[16] Jitendra Bhandari, Abdul Khader Thalakkattu Moosa, Benjamin Tan, Christian Pilato, Ganesh Gore, Xifan Tang, Scott Temple, Pierre-Emmanuel Gaillardo, and Ramesh Karri. 2021. Not All Fabrics Are Created Equal: Exploring eFPGA Parameters For IP Redaction. *arXiv preprint arXiv:2111.04222* (2021).

[17] Jitendra Bhandari, Abdul Khader Thalakkattu Moosa, Benjamin Tan, Christian Pilato, Ganesh Gore, Xifan Tang, Scott Temple, Pierre-Emmanuel Gaillardon, and Ramesh Karri. 2021. Exploring eFPGA-based Redaction for IP Protection. In *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*. 1–9. https://doi.org/10.1109/ICCAD51958.2021.9643548

[18] Michele Borgatti, Francesco Lertora, Benoit Forêt, and Lorenzo Calí. 2003. A reconfigurable system featuring dynamically extensible embedded microprocessor, FPGA, and customizable I/O. *IEEE Journal of Solid-State Circuits* 38, 3 (2003), 521–529.

[19] Arne Brataas and Kjetil M. D. Hals. 2014. Spin–orbit torques in action. *Nature Nanotechnology* 9, 2 (01 Feb 2014), 86–88. https://doi.org/10.1038/nnano.2014.8

[20] Business Wire. 2015. Intel to Acquire Altera. https://www.businesswire.com/news/home/20150601005864/en/Intel-to-Acquire-Altera

[21] Prabuddha Chakraborty, Jonathan Cruz, Abdulrahman Alaql, and Swarup Bhunia. 2021. SAIL: Analyzing Structural Artifacts of Logic Locking using Machine Learning. *IEEE Transactions on Information Forensics and Security* (2021), 1–1. https://doi.org/10.1109/TIFS.2021.3096028

[22] Jianqi Chen and Benjamin Carrion Schafer. 2021. Area Efficient Functional Locking through Coarse Grained Runtime Reconfigurable Architectures. In *Proceedings of the 26th Asia and South Pacific Design Automation Conference*. 542–547.

[23] Jianqi Chen, Monir Zaman, Yiorgos Makris, R. D. (Shawn) Blanton, Subhasish Mitra, and Benjamin Carrion Schafer. 2020. DECOY: DEFlection-Driven HLS-Based Computation Partitioning for Obfuscating Intellectual PropertY. In *Proceedings of the 57th ACM/EDAC/IEEE Design Automation Conference* (Virtual Event, USA) *(DAC '20)*. IEEE Press, Article 6, 6 pages.

[24] Prattay Chowdhury, Chaitali Sathe, and Benjamin Carrion Schaefer. 2022. Predictive Model Attack for Embedded FPGA Logic Locking. In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*. 1–6.

[25] Subhajit Dutta Chowdhury, Gengyu Zhang, Yinghua Hu, and Pierluigi Nuzzo. 2021. Enhancing SAT-attack resiliency and cost-effectiveness of reconfigurable-logic-based circuit obfuscation. In *2021 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.

[26] R. P. Cocchi, J. P. Baukus, L. W. Chow, and B. J. Wang. 2014. Circuit camouflage integration for hardware IP protection. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–5. https://doi.org/10.1145/2593069.2602554

[27] Luca Collini, Benjamin Tan, Christian Pilato, and Ramesh Karri. 2022. Reconfigurable Logic for Hardware IP Protection: Opportunities and Challenges. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 1–7.

[28] Electronic Design News (EDN). 2002. *Hybrid architecture embeds Xilinx FPGA core into IBM ASICs*. https://www.edn.com/hybrid-architecture-embeds-xilinx-fpga-core-into-ibm-asics/?utm_source=eetimes&utm_medium=relatedcontent

[29] Susanne Engels, Max Hoffmann, and Christof Paar. 2022. A critical view on the real-world security of logic locking. *Journal of Cryptographic Engineering* 12, 3 (01 Sep 2022), 229–244.

[30] Fortune Business Insights. [Online]. Available at: https://www.fortunebusinessinsights.com/integrated-circuit-market-106522. Integrated Circuit market Size, share and impact of COVID-19. ([Online]. Available at: https://www.fortunebusinessinsights.com/integrated-circuit-market-106522).

[31] U. Guin, K. Huang, D. DiMase, J. M. Carulli, M. Tehranipoor, and Y. Makris. 2014. Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain. *Proc. IEEE* 102, 8 (2014), 1207–1228. https://doi.org/10.1109/JPROC.2014.2332291

[32] Bo Hu, Tian Jingxiang, Shihab Mustafa, Rajavendra Reddy Gaurav, Swartz William, Makris Yiorgos, Carrion Schaefer Benjamin, and Sechen Carl. 2019. Functional Obfuscation of Hardware Accelerators through Selective Partial Design Extraction onto an Embedded FPGA. In *Proceedings of the 2019 Great Lakes Symposium on VLSI*. 171–176.

[33] Christophe Huriaux, Olivier Sentieys, and Russell Tessier. 2016. Effects of I/O routing through column interfaces in embedded FPGA fabrics. In *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*. 1–9. https://doi.org/10.1109/FPL.2016.7577376

[34] Intel Inc. 2015. *Intel Arria 10 Device Overview*. https://docs.xilinx.com/v/u/en-US/ds150

[35] Invent Logic. 2014. FPGA Architecture. https://allaboutfpga.com/fpga-architecture/

[36] Franz Kaitlyn. 2021. *History of the FPGA*. https://digilent.com/blog/history-of-the-fpga/

[37] Ramesh Karri, Jeyavijayan Rajendran, Kurt Rosenfeld, and Mohammad Tehranipoor. 2010. Trustworthy Hardware: Identifying and Classifying Hardware Trojans. *Computer* 43, 10 (2010), 39–46. https://doi.org/10.1109/MC.2010.299

[38] Changmoo Kim, Mookyoung Chung, Yeongon Cho, Mario Konijnenburg, Soojung Ryu, and Jeongwook Kim. 2014. Ulp-srp: Ultra low-power samsung reconfigurable processor for biomedical applications. *ACM Transactions on Reconfigurable Technology and Systems (TRETS)* 7, 3 (2014), 1–15.

[39] Dirk Koch, Jim Torresen, Christian Beckhoff, Daniel Ziener, Christopher Dennl, Volker Breuer, Jürgen Teich, Michael Feilen, and Walter Stechele. 2012. Partial reconfiguration on FPGAs in practice — Tools and applications. In *ARCS 2012*. 1–12.

[40] Gaurav Kolhe et al. 2022. Breaking the Design and Security Trade-off of Look-up Table-Based Obfuscation. *ACM Trans. Des. Autom. Electron. Syst.* (2022). https://doi.org/10.1145/3510421

[41] Gaurav Kolhe, Hadi Mardani Kamali, Miklesh Naicker, Tyler David Sheaves, Hamid Mahmoodi, P D Sai Manoj, Houman Homayoun, Setareh Rafatirad, and Avesta Sasan. 2019. Security and Complexity Analysis of LUT-based Obfuscation: From Blueprint to Reality. In *2019 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 1–8. https://doi.org/10.1109/ICCAD45719.2019.8942100

[42] Gaurav Kolhe, Sai Manoj PD, Setareh Rafatirad, Hamid Mahmoodi, Avesta Sasan, and Houman Homayoun. 2019. On Custom LUT-Based Obfuscation. In *Proceedings of the 2019 on Great Lakes Symposium on VLSI* (Tysons Corner, VA, USA) *(GLSVLSI '19)*. Association for Computing Machinery, 477–482. https://doi.org/10.1145/3299874.3319496

[43] Gaurav Kolhe, Soheil Salehi, Tyler David Sheaves, Houman Homayoun, Setareh Rafatirad, Manoj PD Sai, and Avesta Sasan. 2021. Securing Hardware via Dynamic Obfuscation Utilizing Reconfigurable Interconnect and Logic Blocks. In *2021 58th ACM/IEEE Design Automation Conference (DAC)*. IEEE, 229–234.

[44] Gaurav Kolhe, Tyler Sheaves, Kevin Immanuel Gubbi, Tejas Kadale, Setareh Rafatirad, Sai Manoj PD, Avesta Sasan, Hamid Mahmoodi, and Houman Homayoun. 2022. Silicon validation of LUT-based logic-locked IP cores. In *Proceedings of the 59th ACM/IEEE Design Automation Conference*. 1189–1194.

[45] Gaurav Kolhe, Tyler Sheaves, Kevin Immanuel Gubbi, Soheil Salehi, Setareh Rafatirad, Sai Manoj PD, Avesta Sasan, and Houman Homayoun. 2022. 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*. 85–90.

[46] Farinaz Koushanfar. 2011. 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* (Lausanne, Switzerland) *(GLSVLSI '11)*. Association for Computing Machinery, New York, NY, USA, 449–454. https://doi.org/10.1145/1973009.1973110

[47] M. Li et al. 2019. Provably Secure Camouflaging Strategy for IC Protection. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 8 (2019), 1399–1412. https://doi.org/10.1109/TCAD.2017.2750088

[48] Wang Lie and Wu Feng-yan. 2009. Dynamic Partial Reconfiguration in FPGAs. In *2009 Third International Symposium on Intelligent Information Technology Application*, Vol. 2. 445–448. https://doi.org/10.1109/IITA.2009.334

[49] B. Liu and B. Wang. 2014. Embedded reconfigurable logic for ASIC design obfuscation against supply chain attacks. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)*. 1–6. https://doi.org/10.7873/DATE.2014.256

[50] Hadi Mardani Kamali, Kimia Zamiri Azar, Kris Gaj, Houman Homayoun, and Avesta Sasan. 2018. 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)*. 405–410. https://doi.org/10.1109/ISVLSI.2018.00080

[51] Areno Matthew. 2020. Supply Chain Threats Against Integrated Circuits. https://www.intel.com/content/dam/www/public/us/en/documents/white-papers/supply-chain-threats-v1.pdf

[52] Menta. 2022. Embedded FPGA IP. https://www.menta-efpga.com/

[53] Prashanth Mohan, Oguz Atli, Joseph Sweeney, Onur Kibar, Larry Pileggi, and Ken Mai. 2021. Hardware redaction via designer-directed fine-grained eFPGA insertion. In *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 1186–1191.

[54] Satwik Patnaik, Nikhil Rangarajan, Johann Knechtel, Ozgur Sinanoglu, and Shaloo Rakheja. 2018. Advancing hardware security using polymorphic and stochastic spin-hall effect devices. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 97–102.

[55] Tiago Perez and Samuel Pagliarini. 2022. Hardware Trojan Insertion in Finalized Layouts: From Methodology to a Silicon Demonstration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2022). https://doi.org/10.1109/TCAD.2022.3223846

[56] T. D. Perez and S. Pagliarini. 2020. A Survey on Split Manufacturing: Attacks, Defenses, and Challenges. *IEEE Access* 8 (2020), 184013–184035. https://doi.org/10.1109/ACCESS.2020.3029339

[57] QuickLogic. 2022. eFPGA IP 2.0 – Enabling Mass Customization with FPGA Technology. https://www.quicklogic.com/products/efpga/efpga-ip2/

[58] Rachel Selina Rajarathnam et al. 2020. ReGDS: A Reverse Engineering Framework from GDSII to Gate-level Netlist. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 154–163. https://doi.org/10.1109/HOST45689.2020.9300272

[59] J. Rajendran, O. Sinanoglu, and R. Karri. 2013. Is split manufacturing secure?. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*. 1259–1264. https://doi.org/10.7873/DATE.2013.261

[60] Nikhil Rangarajan, Satwik Patnaik, Johann Knechtel, Ramesh Karri, Ozgur Sinanoglu, and Shaloo Rakheja. 2020. Opening the doors to dynamic camouflaging: Harnessing the power of polymorphic devices. *IEEE Transactions on Emerging Topics in Computing* (2020).

[61] Renesas Electronics. 2020. *Stream Transpose Processor.* https://www.renesas.com/us/en/products/power-management/pmic/stp-engine.html

[62] Amin Rezaei, Raheel Afsharmazayejani, and Jordan Maynard. 2022. Evaluating the Security of EFPGA-Based Redaction Algorithms *(ICCAD '22)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3508352.3549425

[63] Campbell Richard. 2023. The Radical Future of NVM. https://dl.acm.org/doi/fullHtml/10.5555/2380096.2380099

[64] Masoud Rostami, Farinaz Koushanfar, and Ramesh Karri. 2014. A primer on hardware security: Models, methods, and metrics. *Proc. IEEE* 102, 8 (2014), 1283–1295. https://doi.org/10.1109/JPROC.2014.2335155

[65] J. A. Roy, F. Koushanfar, and I. L. Markov. 2008. EPIC: Ending Piracy of Integrated Circuits. In *2008 Design, Automation and Test in Europe*. 1069–1074. https://doi.org/10.1109/DATE.2008.4484823

[66] Chaitali Sathe, Yiorgos Makris, and Benjamin Carrion Schafer. 2022. Investigating the Effect of different eFPGAs fabrics on Logic Locking through HW Redaction. In *2022 IEEE 15th Dallas Circuit And System Conference (DCAS)*. IEEE, 1–6.

[67] Semiconductor Engineering. [n. d.]. *Big trouble at 3nm.* https://semiengineering.com/big-trouble-at-3nm/

[68] M. M. Shihab et al. 2019. Design Obfuscation through Selective Post-Fabrication Transistor-Level Programming. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*. 528–533. https://doi.org/10.23919/DATE.2019.8714856

[69] M. M. Shihab et al. 2020. ATTEST: Application-Agnostic Testing of a Novel Transistor-Level Programmable Fabric. In *IEEE 38th VLSI Test Symposium (VTS)*. 1–6. https://doi.org/10.1109/VTS48691.2020.9107561

[70] Hartej Singh, Ming-Hau Lee, Guangming Lu, Fadi J Kurdahi, Nader Bagherzadeh, and Eliseu M Chaves Filho. 2000. MorphoSys: an integrated reconfigurable system for data-parallel and computation-intensive applications. *IEEE transactions on computers* 49, 5 (2000), 465–481.

[71] Watts Stephen. 2018. *How application-specific integrated circuits are powering the future of IT today.* https://www.cio.com/article/228545/how-application-specific-integrated-circuits-are-powering-the-future-of-it-today.html

[72] P. Subramanyan, S. Ray, and S. Malik. 2015. Evaluating the security of logic encryption algorithms. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 137–143. https://doi.org/10.1109/HST.2015.7140252

[73] Pramod Subramanyan, Sayak Ray, and Sharad Malik. 2015. Evaluating the security of logic encryption algorithms. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 137–143. https://doi.org/10.1109/HST.2015.7140252

[74] Ian Swarbrick, Dinesh Gaitonde, Sagheer Ahmad, Brian Gaide, and Ygal Arbel. 2019. Network-on-Chip Programmable Platform in Versal ACAP Architecture. In *Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays* (Seaside, CA, USA) *(FPGA '19)*. Association for Computing Machinery, New York, NY, USA, 212–221.

[75] Joseph Sweeney, V Mohammed Zackriya, Samuel Pagliarini, and Lawrence Pileggi. 2020. Latch-Based Logic Locking. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*. 132–141. https://doi.org/10.1109/HOST45689.2020.9300256

[76] Chiara Muscari Tomajoli, Luca Collini, Jitendra Bhandari, Abdul Khader Thalakkattu Moosa, Benjamin Tan, Xifan Tang, Pierre-Emmanuel Gaillardon, Ramesh Karri, and Christian Pilato. 2022. ALICE: An Automatic Design Flow for eFPGA Redaction. *arXiv preprint arXiv:2205.07425* (2022).

[77] Randy Torrance and Dick James. 2011. The state-of-the-art in semiconductor reverse engineering. In *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 333–338.

[78] Stephen Trimberger. 1995. FPGA Technology: Past, Present and Future. In *ESSCIRC '95: Twenty-first European Solid-State Circuits Conference*. 12–15.
[79] Huanyu Wang, Domenic Forte, Mark M. Tehranipoor, and Qihang Shi. 2017. Probing Attacks on Integrated Circuits: Challenges and Research Opportunities. *IEEE Design & Test* 34, 5 (2017), 63–71. https://doi.org/10.1109/MDAT.2017.2729398
[80] Theodore Winograd, Hassan Salmani, Hamid Mahmoodi, Kris Gaj, and Houman Homayoun. 2016. Hybrid STT-CMOS designs for reverse-engineering prevention. In *Proceedings of the 53rd Annual Design Automation Conference*. 1–6.
[81] Y. Xie and A. Srivastava. 2017. Delay locking: Security enhancement of logic locking against IC counterfeiting and overproduction. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*. 1–6. https://doi.org/10.1145/3061639.3062226
[82] Xilinx Inc. 2015. *Virtex-5 Family Overview*. https://docs.xilinx.com/v/u/en-US/ds100
[83] Xilinx Inc. 2015. *Virtex-6 Family Overview*. https://docs.xilinx.com/v/u/en-US/ds150
[84] Xilinx Inc. 2022. *Zynq UltraScale+ MPSoC Data Sheet*. https://www.xilinx.com/content/dam/xilinx/support/documents/data_sheets/ds891-zynq-ultrascale-plus-overview.pdf
[85] Jianlei Yang, Xueyan Wang, Qiang Zhou, Zhaohao Wang, Hai Li, Yiran Chen, and Weisheng Zhao. 2018. Exploiting spin-orbit torque devices as reconfigurable logic for circuit obfuscation. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 1 (2018), 57–69.
[86] M. Yasin, B. Mazumdar, O. Sinanoglu, and J. Rajendran. 2020. Removal Attacks on Logic Locking and Camouflaging Techniques. *IEEE Transactions on Emerging Topics in Computing* 8, 2 (2020), 517–532. https://doi.org/10.1109/TETC.2017.2740364
[87] M. Yasin, J. Rajendran, and O. Sinanoglu. 2019. *Trustworthy Hardware Design: Combinational Logic Locking Techniques*. Springer, Cham.
[88] Muhammad Yasin, Abhrajit Sengupta, Mohammed Thari Nabeel, Mohammed Ashraf, Jeyavijayan (JV) Rajendran, and Ozgur Sinanoglu. 2017. Provably-Secure Logic Locking: From Theory To Practice. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. 1601–1618.
[89] M. Yasin and O. Sinanoglu. 2017. Evolution of logic locking. In *2017 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. 1–6. https://doi.org/10.1109/VLSI-SoC.2017.8203496
[90] Kimia Zamiri Azar, Hadi Mardani Kamali, Houman Homayoun, and Avesta Sasan. 2019. Threats on Logic Locking: A Decade Later. In *GLSVLSI '19: Proceedings of the 2019 on Great Lakes Symposium on VLSI*. 471–476.
[91] Hui Zhang, Vandana Prabhu, Varghese George, Marlene Wan, Martin Benes, Arthur Abnous, and Jan M Rabaey. 2000. 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)*. IEEE, 68–69.

Received 25 May 2023

Curriculum Vitae

1. Personal data

Name	Zain Ul Abideen
Date and place of birth	01.01.1997, Pakpattan, Pakistan
Nationality	Pakistan

2. Contact information

Address	Tallinn University of Technology (TalTech), School of Information Technologies, Department of Computer Systems, Akadeemia tee 7/2-314, 12611, Tallinn, Estonia
E-mail	xaainulabideen@gmail.com, zain.abideen@taltech.ee

3. Education

2020–2024	Tallinn University of Technology, School of Information Technologies, Information and Communication Technology, Ph.D. studies
2018–2019	Institut polytechnique de Grenoble – ESISAR, Computer Engineering, MSc
2014–2018	University of Management and Technology, Department of Electrical Engineering, Electrical Engineering, BSc

4. Language competence

Urdu	native
English	fluent
French	beginner
Estonian	beginner

5. Professional employment

March 2020–January, 2024	Center for Hardware Security, Early Stage Researcher
January, 2019–August, 2019	Laboratoire de Conception et d'Intégration des Systèmes (LCIS), Research Assistant

6. Computer skills

- Operating systems: Windows and Linux
- Document preparation: LaTeX
- Programming languages: Verilog, TCL, VHDL, Python, C, SystemVerilog, MATLAB, Javascript, Intel Assembly, MIPS Assembly, Latex, Visual Basic
- Tools: Cadence Genus, Cadence Innovus, Siemes EDA Calibre, Cadence Xcelium Logic Simulator, Siemes EDA ModelSim, Xilinx Vivado IDE, LTSpice, Proteous, NI Multisim, KiCAD EDA, STM32Cube

7. Honours and awards

- 2016, Continuous Rector/Dean Merit awards for five semesters
- 2018, Prestigious Rector's medal upon completing BSc
- 2018, IDEX Master Scholarship for master studies
- 2022, First position in the International Hardware Security Competition (HeLLO: CTF)
- 2023, Young People Programme funding for DATE
- 2023, Young Fellows Programme funding for DAC
- 2023, Second place in Student Research Competition at ESWEEK
- 2023, Awarded with Keevalikku IT doctoral scholarship

8. Defended theses

- 2019, Development of a FPGA emulation-based fault injection tool for RTL designs, MSc, Prof. Dr. Vincent Beroulle, Institut polytechnique de Grenoble – ESISAR, France
- 2018, An Intuitive Approach for the Design and Implementation of Omni-Directional Tele-presence Robot, BSc, Mr. Hassan Tariq, University of Management of Technology, Lahore, Pakistan

9. Field of research

- Hardware Security
- Reconfigurable-based Obfuscation
- Secure-ASIC Design
- Physical Unclonable Function (PUF)
- Ring Oscillators
- Very Large Multipliers

Elulookirjeldus

1. Isikuandmed

Nimi	Zain Ul Abideen
Sünniaeg ja -koht	01.01.1997, Pakpattan, Pakistan
Kodakondsus	Pakistani

2. Kontaktandmed

Aadress	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Arvutisüsteemide instituut, Akadeemia tee 7/2-314, 12611 Tallinn, Eesti
E-post	xaainulabideen@gmail.com, zain.abideen@taltech.ee

3. Haridus

2020–2024	Tallinna Tehnikaülikool, Infotehnoloogia teaduskond, Informatsiooni-ja kommunikatsioonitehnoloogia, doktoriõpe
2018–2019	Grenoble'i polütehniline instituut – ESISAR, Arvutitehnika, MSc
2014–2018	Juhtimis- ja Tehnikaülikool, . . . Elektrotehnika osakond, Elektrotehnika, BSc

4. Keelteoskus

urdu keel	emakeel
inglise keel	kõrgtase
prantsuse keel	Algaja
eesti keel	Algaja

5. Teenistuskäik

märts 2020–jaanuar 2024	Riistvara Turvalisuse Keskus, doktorant-nooremteadur
jaanuar 2019–august 2019	Süsteemide Kavandamise ja Integreerimise Laboratoorium (LCIS), teadusassistent

6. Arvuti oskused

- Operatsioonisüsteemid: Windows ja Linux
- Dokumendi ettevalmistamine: LaTeX
- Programmeerimiskeeled: Verilog, TCL, VHDL, Python, C, SystemVerilog, MATLAB, Javascript, Intel Assembly, MIPS Assembly, lateks, Visual Basic
- Tööriistad: Cadence Genus, Cadence Innovus, Siemes EDA Calibre, Cadence Xcelium Loogikasimulaator, Siemes EDA ModelSim, Xilinx Vivado IDE, LTSpice, Proteous, NI Multisim, KiCAD EDA, STM32Cube

7. Autasud ja auhinnad

- 2016, pidev rektori/dekaani preemia viie semestri eest
- 2018, prestiižne rektori medal bakalaureuseõppe lõpetamisel
- 2018, IDEXi magistristipendium magistriõppeks
- 2022, esimene koht rahvusvahelisel riistvaraturbe konkursil (HeLLO: CTF)
- 2023, noorteprogrammi rahastus kuupäeval DATE
- 2023, noorte stipendiaatide programmi rahastamine DACile
- 2023, Teine koht Tudengiuuringute Konkursil ESWEEK'il
- 2023, pälvis Keevalikku IT doktoriõppe stipendiumi

8. Kaitstud lõputööd

- 2019, FPGA-emulatsioonil põhineva veainsertsioonitööriista arendamine RTL-kavandite jaoks, MSc, Prof. Dr. Vincent Beroulle, Grenoble'i Polütehniline Instituut – ESISAR, Prantsusmaa
- 2018, Intuitiivne lähenemine omni-direktiivse kaugjuhtimisroboti kavandamisele ja rakendamisele, BSc, Hr. Hassan Tariq, Tehnoloogia Juhtimise Ülikool, Lahore, Pakistan

9. Teadustöö põhisuunad

- Riistvara turvalisus
- Ümberkonfigureeritav hägustamine
- Turvaline ASIC disain
- Füüsiline mittekloneeritav funktsioon (PUF)
- Rõngasostsillaatorid
- Väga suured kordajad

ISSN 2585-6901 (PDF)
ISBN 978-9916-80-099-7 (PDF)