FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Design and evaluation of a PUF implemented in advanced CMOS technology

Gonçalo Fagundes Silva Nunes

Mestrado em Engenharia Eletrotécnica e de Computadores

Supervisor: João Canas Ferreira

October 11, 2023

Abstract

This dissertation addresses the domain of hardware security, more precisely, the implementation of Physical Unclonable Functions (PUFs) in Integrated Circuits (ICs) as an alternative to current hardware security practices.

We begin our study by exploring existing PUF concepts and how they are implemented into practice. Subsequently, we will implement and test an Arbiter PUF out of the analyzed designs using advanced CMOS technology.

Afterward, diverse performance metrics are applied to the PUF to evaluate our chosen architecture's performance. Through the obtained results, we try to infer whether a PUF's functionality is or is not affected by the scaling down of advanced CMOS technologies.

Acknowledgements

This dissertation and the work developed within it signal the culmination of my time here at Faculdade de Engenharia da Universidade do Porto. The journey has been arduous, lengthy, and exhausting. However, throughout this unexpected, long, and treacherous route, I have been fortunate to receive support from many outstanding individuals who have been consistently available to me during the lows and the highs. The words expressed here serve as a simple, grateful, humble homage to those who stood by me.

First and foremost, I would like to express my heartfelt gratitude to my family. Their unwavering support has been crucial from the outset, enabling me to pursue the desired career and facilitating my relocation from the Azores to Oporto. Above all, I want to extend my most profound appreciation to my parents for their unwavering belief in my abilities and steadfast support.

To my friends, I consider myself incredibly fortunate to have crossed paths with such genuinely good-hearted and caring individuals. I am grateful for the invaluable explanations, corrections, laughter, and warm hugs you generously shared. Among the many wonderful people I have had the privilege to meet along my journey, I would like to extend my deepest appreciation to Carlos, Carminho, Helena, and Mariana. To every one of you, I wholeheartedly say thank you!

It is also more than deserving to mention Dra. Helena Lopes, her commitment throughout my hardships, and the constant noting of the bright side of the entire journey.

Lastly, I would like to express my heartfelt gratitude to my supervisor, Professor João Canas Ferreira, for his unwavering support throughout my academic journey, including completing my dissertation. I am truly grateful for your kindness, patience, and willingness to share your knowledge and offer assistance whenever needed.

Obrigado,

Gonçalo

"The credit belongs to the man ... who errs, who comes short again and again,
because there is no effort without error and shortcoming
... who at the best knows, in the end, the triumph of high achievement,
and who at the worst, if he fails, at least fails while daring greatly ...."

Theodore Roosevelt

Contents

List of Figures

List of Tables

Abbreviations and Symbols

ASIC	Application-specific integrated circuit
BEOL	Back end of line
CMOS	Complementary metal–oxide–semiconductor
CRP	Challenge-Response pairs
CS	Current Starved
DoS	Denial of Service
EEPROM	Electrically erasable programmable read-only memory
FEOL	Front end of line
FPGA	Field-programmable gate array
HD	Hamming Distance
HMAC	Hash Message Authentication Code
HS	Hardware Security
HT	Hardware Trojan
HW	Hamming Weight
IC	Integrated Circuit
INV	Inverter
I/O	Input/Output
IoT	Internet of Things
IP	Intellectual Property
LFSR	Linear-Feedback Shift Register
ML	Machine Learning
MUX	Multiplexer
NVM	Non-volatile Memory
POK	Physically Obfuscated Key
PUF	Physical Unclonable Function
RE	Reverse Engineering
RO	Ring Oscillator
RTL	Register Transfer Level
SCA	Side-channel Attack
SRAM	Static random-access memory
TRNG	True Random Number Generator
UNO	Unique Object
ZTC	Zero Temperature Coefficient Point

Chapter 1

Introduction

1.1 Context

Over the past decades, humanity has witnessed our modern world's astonishing technological momentum. The state-of-the-art technology today usually needs to be updated within a few years. Humanity has never been capable of creating and developing new technologies at such rates. Electronic devices are leading our technological evolution. These devices, born from the constant development of electronics, have transformed how we learn, work, and communicate—creating a new world where reliance on them is not just a choice but a natural progression of our modern existence.

At the forefront of this technological wonder, we have the ICs, a foundational component of the devices we have grown accustomed to using. With the remarkable growth in the number of transistors, surpassing even billions, the design of secure systems has become more complex. Verifying that these intricate systems perform their intended functions precisely—without deviation—has become indispensable. However, with the intricate nature of contemporary designs and the multitude of interactions between various on-chip components, ensuring the security of ICs presents itself as a highly challenging task [1]. Albeit challenging, we should recall how important it is, as, in the end, these devices store not only our personal information but also proprietary data, making them a lucrative and appetizing target for hackers' attacks [2].

So, as ICs continue to underpin our digital world, assuring the trustworthiness of the electronic systems they support becomes a critical endeavor [1]. From this premise, security can be defined as a means to enable trust [3]. And security relies on cryptography to protect sensitive information. Cryptography can be defined as the methods and procedures used to construct protocols and algorithms to achieve information security [3]. Therefore, cryptographic primitives become vital and the cornerstone that provides the foundation for executing algorithms and/or exchanging information securely, generating and storing secrets, and safeguarding data within these devices [3].

These cryptographic primitives shall uphold the following [3]:

- Data confidentiality: keeping information secret from unauthorized parties;

- Entity authentication: obtaining proof of the identity and the presence of the entity one is interacting with;
- Data integrity and authentication: aimed at preventing and detecting unauthorized alteration of data (integrity) and ensuring the origin of the data (authentication);

As society transforms into a digital information ecosystem and our dependence on digital information processing and communication systems escalates, implementing robust cryptographic primitives becomes paramount to fostering trust in this new paradigm [3][1].

A cryptographic primitive can be defined as a low-level algorithm with a security-specific function. These algorithms form the building blocks of a secure system, hence their careful design and rigorous checking. The more commonly used primitives are [1]:

- Symmetric ciphers: a deterministic algorithm that converts a stream of unencrypted data (aka plaintext) into a ciphertext, the latter in an illegible format hiding its information. A secret key, only known to authorized parties, is used to encrypt and decrypt.
- Asymmetric ciphers: unlike the previous, they do not use the same key for encryption and decryption. Instead, each user has a public key that can be widely available and a private key only known to them. Its security relies on solving prohibitively complex mathematical problems (e.g., computing discrete logarithms) without access to further information (i.e., the private key).
- One-way Hash Functions: this primitive maps arbitrary length inputs to a short fixed-length output (digest). One example of said primitive is HMAC (Hash-based Message Authentication Code).
- Random Number Generators: mainly used to generate a nonce (an arbitrary number used only once). The nonce is then employed as an initialization vector for encryption algorithms and authentication protocols to prevent replay attacks.

When conceiving cryptographic primitives, a typical design principle is to streamline security objectives by reducing them to the confidentiality of a single parameter - the key. The degree of protection is determined by the effort required to break it without knowledge of the said key. This should be an exponential function of the key's bit length [3].
These keyed primitives are often compared to a "black box" where their users and outsiders only see an input-output relation instead of seeing the box's internal workings. For the majority of them, we assume their capability to [3]:

- Secure key generation: to generate, for every instantiation of the primitive, a secure (random, unique, and unpredictable) key;
- Secure key storage: the key can be assigned to, stored, and retrieved by the instantiation without being revealed;

- Secure execution: the instantiation can execute the cryptographic algorithm without revealing any information about the key or about internal results and without an outsider being able to influence the internal execution in any possible way;

Although convenient and logical, these assumptions are difficult to attain in practice, requiring physical measures to implement them. Information security objectives are reduced to physical security requirements and their given implementations. However, these are still vulnerable to malicious attacks [3].

Lenstra et al. [4] showed how a shortage in randomness within a large collected set of actually used public keys could jeopardize secure key generation. Torrance and James [5], and Herder et al. [6] note how critical non-volatile digital memories on a silicon chip can still be vulnerable and endanger secure key storage. Not even the secure execution assumption can be relied upon as side-channel attacks, which abuse the fact that all actions on a digital platform leak information about their execution through so-called side channels, e.g., through their power consumption [7], become more pervasive.

Given this new paradigm of distributed devices and the computing flow of sensitive data they create, it becomes evident how the search for improved device security should be thorough and pay attention to both software and hardware security. Software security is a widely discussed and understandable topic for a greater audience. However, the same can not be said about hardware security. It then becomes interesting to categorize the various attacks these devices could face nowadays. To do so, we will follow the classification proposed by Halak [1]. By doing so, we can classify them into three categories. These being:

- Communication Attacks: the unauthorized access of a communication channel by an eavesdropping adversary, allowing him to steal sensitive information or maliciously manipulate the transmitted data;
- Software Attacks: aimed to maliciously modify the software running on a computing device to install malware and/or cause a system malfunction;
- Hardware Attacks: aimed to maliciously modify or tamper with the physical implementation of a computing device;

Out of the three, and noting the previously referred need for physical measures and their liabilities, we will focus on Hardware Attacks. These generally require direct access to the hardware or its design files and can occur at any time during the life cycle of the computing device. Consequently, they can be classified into two further categories [1]:

- (a) Attacks during design and fabrication: usually these occur either through Trojan insertion, where extra circuitry with malicious functionality is added to the design [8][9], or, due to IC counterfeiting [10][11], where a malicious fabrication facility produces more than required, subsequently selling the exceed in the black market;

- (b) Post-fabrication attacks: also known as Physical attacks, occur after the device is put into operation. They can be further classified into three categories:

  b1) Invasive attacks: that procure access to the internal structure of the integrated circuits, e.g., to reverse engineer and steal the intellectual property (IP) of a design;

  b2) Non-invasive attacks: where the adversary interacts with the hardware externally. One such example could be through a side-channel analysis where an attacker analyses the correlation between leakage current and the input patterns trying to infer the secret key [12];

  b3) Semi-invasive attacks: which require access to the surface of a device but not its internal logic. E.g., optical fault injection, where illuminating a device could cause some of its transistors to conduct current, which may trigger an error [13];

Despite typically requiring more resources and knowledge to wage, making them less frequent than communication or software attacks, the cost of protecting electronic systems against such attacks is high. However, having found this necessity, the urge to develop new and better primitives that, solely based on physical reasoning, can be trusted to resist physical attacks, consequently providing physical security, creates a search for what Maes calls physical roots of trust [3]. These are defined as trusted primitives rooted in the actual physical world. To truly understand what qualifies as a physical root of trust, consider using human fingerprints as an example and how they are used as a form of authentication. Possible candidates for physical roots of trust could be [3]:

- True Random Number Generators (TRNGs): hardware devices capable of generating random numbers based on physical sources of randomness, as a result, trusted to produce highly random keys for cryptographic purposes [14];
- Security-aware design styles: specific designs for digital silicon circuits have been developed which try to minimize, ideally eliminate, the feasibility of certain side-channel attacks [15];
- Physically Unclonable Functions: capable of producing unpredictable and instance-specific values usable for physically secure key generation and storage [16][17];

It is from within this need for physical roots of trust that PUFs have emerged as a possible solution in the search for better primitives ensuring to protect devices against physical attacks, counterfeiting, or cloning attempts [1].

These security primitives translate an input challenge into an output response through a physical system in a unique and unclonable way specific to the hardware instance [18]. They do so by attempting to harness the variability in the manufacturing process and its consequent implications on a chip, e.g., gate delay [18][3][1]. By deriving their cryptographic secret from these characteristics, a unique silicon "fingerprint" [3][1], almost impossible to replicate even though a manufacturer might have access to the chip's design [6], is created. Hence, the information retrieved

originates from within the system's physical characteristics rather than an attached memory [18], likely to be targeted [6]. Combined with appropriate processing algorithms [3], the system can authenticate any object or device connected to it or embedded within it [18][3][19]. It can also generate and store secrets securely [3]. This concept has become an object of study, development, and attempted use in multiple different applications [18].

1.2 Motivation

In the previous section, we underscored the critical importance of security in our interconnected world of devices. As we lean further toward connected systems and electronic devices, so does the urgency to safeguard their defenses against an ever-evolving landscape of threats.

The multifaceted nature of chip development means that to develop a new chip, companies can either produce or outsource IP designs, integrate them with their in-house systems, send the blueprints to a third-party foundry for fabrication, and test them, usually at another third-party facility. This multinational distributed production chain raises the opportunities for potential adversaries to tamper with designs, compromising them and/or implanting malicious functionalities [1].

One such example is the previously mentioned hardware Trojans [9], which can be used, for example, to leak information by creating parasitic antennas [1]. The Internet of Things (IoT) boom connected, and promises to connect, many more devices to the Internet [20]. This availability makes it easier for an attacker to perform a physical attack, consequently extracting sensitive data, injecting a fault, or reverse engineering its design [21]. This phenomenon is particularly challenging given the constrained resources of these devices, making traditional cryptographic methods impractical [22]. Moreover, we have also seen an increased use of what we can call secure hardware tokens, e.g., smart cards and staff cards. Cryptographic operations have increasingly relied on these tokens for security-sensitive applications in recent years, for example, through the implementation of electronic identity systems for tasks like paying taxes or managing retirement funds, such as "ID-porten" and "Telia ID" in Norway and Sweden, respectively [1]. Another usage is the secure hardware tokens some financial institutions offer their clients to confirm their identities and/or generate a digital signature to confirm the details of a transaction, enhancing the security of online banking beyond the usual login/password. [23][24][1].

To address these complexities and vulnerabilities, it is necessary to implement robust and strategic hardware security measures, thus strengthening the integrity, trustworthiness, and resilience of our technologically interconnected world.

Key-based cryptography systems have traditionally been used to protect IP and licensing applications [2]. However, saving keys using nonvolatile electrically erasable programmable read-only memory (EEPROM) or battery-backed static random-access memory (SRAM) is expensive. In terms of both design space and power usage [6]. Additionally, such nonvolatile memories, used to store and manage keys, are often vulnerable to invasive attack mechanisms and tampering [25][6]. Protection against these attacks requires active tamper detection/prevention circuitry, which must be continually powered. However, due to the diversity of attacks, the design of fully tamper-proof

ICs is a challenge and an inevitable overhead regarding performance, area, and power. Their high cost is yet another factor making them unappealing for commercial purposes. [2][6][11].

The quest for improved and feasible alternative solutions drove the academic and industrial communities alike. Such solutions should be [2]:

- Simpler than traditional cryptographic methods;
- Cost-effective;
- Inherently random (without a definite mathematical model);
- Intrinsically tamper-resistant;
- Easily implemented on resource-constrained platforms;
- Difficult to duplicate and reverse;

We are all aware that the fabrication processes of physical objects usually have some limitations. Attaining accurate control of the devices being manufactured is a task that poses difficulties, leading to slight variations in the dimensions of the resulting products. This holds especially true when it comes to the production of semiconductor devices in advanced technology nodes (below 90nm) [26][27][28]. Given these intrinsic process variations, the transistors devices' length, width, oxide thicknesses, and doping levels may vary. Despite using the same mask and going through the same manufacturing process, each IC is unique and different from others due to normal manufacturing variability. So, the electrical characteristics of a transistor may differ slightly when fabricated on different devices [1][6].

The exquisite solution that could potentially fulfill those prerequisites appeared always to have been right there, the process variation in IC fabrication [19][29][30].

PUFs exploit these inherent process variations to create a unique hardware device identifier. Therefore, their usage as a basic building block within the construction of security protocols and the design of secure systems presents itself as a viable and logical solution. As such, they could be applied in [1]:

- Secure key generation and storage: PUFs provide a different approach for key generation and storage. Usually, for encryption and authentication, most cryptographic algorithms require a pre-installed key, which acts as the root of trust. As stated, these tend to be vulnerable when stored in the device's memory. By utilizing a PUF's response to construct a root key, the need for storing the key in memory is removed, and better protection is provided [31];
- Low-cost authentication: Combining their relatively simpler, cheaper, and more energy-efficient architectures with their unique input/output response behavior, PUFs become an attractive choice for low-cost authentication schemes, especially for recourse-constrained systems such as IoT devices [22];

- Anti-counterfeiting design: Malicious factories can be prevented from overproducing ICs by embedding each chip with a PUF circuit that securely locks the design. After fabrication, the designer characterizes and authenticates each PUF's behavior, allowing the latter to generate a passkey to activate only authenticated chips [32];
- Secure hardware tokens: Once again, thanks to their complex input/output response behavior and their increased tamper-resistant, especially when compared to reliant on digitally stored information tokens (e.g.smart cards), PUFs can be used in hardware-assisted cryptographic protocols [33];
- Secure remote sensing: PUFs, like other circuits, tend to be susceptible to changes in environmental factors, such as temperature or power supply voltage. Consequently, their response depends not only on the applied inputs but also on these environmental factors. To accurately measure environmental changes with PUF sensors, firstly, it is crucial to understand how their responses correlate with said environmental factors. Only then is the sensor ready to deploy. Therefore, PUF implementation eliminates the need for extra encryption, making a remote sensing system cost-effective and secure [34];

Considering their aforementioned characteristics and potential applications, we find it interesting to further evaluate them, particularly in advanced CMOS technology.

1.3 Objectives

Having encompassed this work in a larger context and defined its motivation, the following section will present the main objectives to accomplish during its elaboration.

Focusing on digital circuits, first, we pretend to evaluate a state-of-the-art PUF implemented at the layout level in an advanced CMOS technology. Its evaluation shall be performed according to the following criteria:

- Standard area;
- Time;
- Power;
- Robustness;
- Temperature dependency;
- Inter-die variability;
- Inter-wafer variability;

Smaller CMOS technologies raise the variability within the fabrication process; therefore, it is crucial to understand how this impacts the functionality and applicability of a PUF. If possible,

an evaluation of the relationship between the implemented PUF and the technology used shall be conducted.

1.4 Methodology

To accomplish the proposed objectives, the following methodology shall be applied:

1. Create a State-of-the-Art about this topic, thus allowing familiarization with the theme and acquiring knowledge on the most frequent and modern PUF implementations;
2. Choose a suitable CMOS technology and design flow for the chosen PUFs;
3. Evaluate their performance according to previously defined criteria;
4. Document the results obtained;

1.5 Document structure

This first chapter briefly introduces the reader to the theme and presents its scientific context, explaining the motivation and objectives of the work to be developed.

Following into Chapter 2, we expand on our knowledge of PUFs, presenting them in greater detail. What exactly is a PUF, how it works, and their most common architectures were questions we aimed to answer.

In Chapter 3, we present the chosen architecture to design, justify its choice and the results of its evaluation.

Finally, we conclude with Chapter 4, where the main contributions of the developed work are presented along with suggestions for further improvements.

Chapter 2

State of the art

This chapter highlights the significant findings from previous research to build a solid basis for this dissertation's upcoming exploration and analysis. As such, these shall be the starting point for developing the work.

2.1 Unique Objects

Authentication has always played a crucial role in the transmission of reliable information, from the historical use of sealing wax and other methods to the realization expressed by Landauer that *"Information is physical"* [35], the need for reliable authentication has endured. As such, and continuing with the example of wax stamps, this notion of harnessing a physical object's intrinsic and hard-to-replicate irregularities is far from novel, even if its formal nomenclature was yet to emerge.

Embodying the age-old pursuit of information authentication through the integration of physical attributes, two prominent categories of disorder-based security systems can be considered: Unique Objects (UNOs) and PUFs [36].

A Unique Object is a physical system that exhibits a small set of unique analog properties upon measurement by an external apparatus. This system and its properties shall have the following characteristics [36]:

- Disorder: These properties, and the "fingerprint" they create, should be based on the unique physical disorder of the object;
- Operability: This "fingerprint" measurement shall be cost- and time-effective. It also should be stable over time, robust to aging, environmental conditions, and repeated measures;
- Unclonability: To produce another object with the same unique analog properties, thus creating an equal "fingerprint" when queried by the measurement device, should be prohibitively costly or impractical (even for the manufacturer). Although part of the system, the measurement device is not required to be unclonable. It can be mass-produced, and

for different measuring devices under the same conditions, the UNO's "fingerprint" should remain the same;

An example of the usage of UNOs occurred during the Cold War. Considering that each speckle pattern would be unique, a thin, random layer of light-scattering particles was sprayed onto nuclear missiles. It was then illuminated from different angles, and a certified inspector captured the interference patterns. Upon future inspections, these patterns would be measured again and compared to the previously recorded ones [36].

Despite being outside the scope of this work, given our focus on what the literature tends to call silicon PUFs [16], we considered this mention of UNOs interesting given their usage of concepts familiar to PUFs such as uniqueness, physical disorder, and unclonability. One could point out three common metrics typically applied to categorize the uniqueness and robustness of both PUF responses and UNO fingerprints, namely entropy, intra and inter-distances [36]. A brief explanation of each of them is given in the following subsections: 2.1.1, 2.1.2, and 2.1.3.

2.1.1 Entropy

For these applications, the entropy is determined by the total count of distinct identifications produced by the device architecture [36].

2.1.2 Intra-distance

Also known as intra-chip or intra-die of a PUF response, it is given by a random variable that describes the distance between two responses from the same PUF instance and the same challenge. So, taking two evaluations $R_i(c)$ and $R'_i(c)$ of the same PUF instance, i, and the same challenge, c, let dist[. , .] be any distance metric over the response set R. The intra-distance of a PUF i is given by Equation 2.1 [37]:

$$Intra-distance \triangleq HD[R_i(c), R'_i(c)] \tag{2.1}$$

Figure 2.1, taken from [37], gives us a visual representation of intra-distance. In it, we can see how the same challenge, c, is evaluated by the same PUF instance, PUF_i, at two different moments, originating two different responses, $R_i(c)$, and $R'_i(c)$. These responses would likely consist of a bit string in an actual application. As such, while comparing two binary strings of equal length, the Hamming distance (HD) is the number of bit positions in which the two bits are different [38]. Given a range from [0,1], an $Intra-distance_i$ close to "zero" means the PUF is highly reliable. Oppositely, a result closer to "one" means the PUF is the least reliable. For example, the intra-distance between two responses generated from the same challenge with the same PUF instance under the same environmental condition should be smaller (ideally zero) than the intra-distance between the same responses generated under two different conditions. This denotes PUFs' susceptibility to the environment they operate in and its variations, e.g., in supply voltage or temperature. An ideal PUF should have a small intra-distance [37].

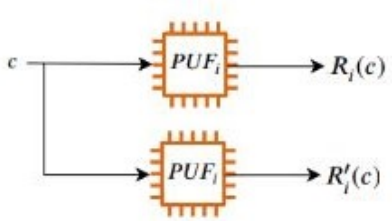


Figure 2.1: Intra-distance of the same PUF instance [37]

2.1.3 Inter-distance

Also known as inter-chip or inter-die of a PUF response, it is given by a random variable that describes the distance between two responses, $R_i(c)$, and $R_j(c)$, generated by two different PUF instances (with the same architecture), PUF_i, and PUF_j, for the same challenge c. Once again, HD is used as a distance metric, as the inter-distance for a given PUF is measured through $R_i(c)$, and $R_j(c)$, as described by Equation 2.2 [37]:

$$Inter-distance_{R(c)} \triangleq HD[R_i(c), R_j(c)] \tag{2.2}$$

Figure 2.2, taken from [37], gives us a visual representation of this metric. In it, we can see how the same challenge, c, is evaluated by two distinct instances, PUF_i, and PUF_j, originating $R_i(c)$, and $R_j(c)$.

Again, for a range of [0,1], if Equation 2.2 results in a value closer to "zero", it means the PUF is less unique. Conversely, a value closer to "one" means the PUF is highly unique. It is important to note that this metric is also prone to variations in environmental conditions. An ideal PUF should have a sizeable inter-device distance [37].

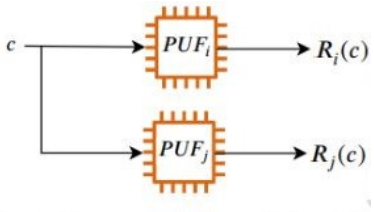


Figure 2.2: Inter-distance for two different PUF instances [37]

2.2 Security Landscape

In sections 1.1 and 1.2, we briefly introduced the reader to some concepts within the domain of Hardware Security (HS). This section presents these concepts more broadly without delving deeply into them, as a rigorous survey of them is not the goal of this work. However, it is essential to encompass PUFs as a possible solution within this security landscape.

Hardware is commonly seen as a trusted party supporting the computer system while running instructions from the software layer. As a result, hardware-based security is often referred to as hardware implementations of cryptographic algorithms where hardware is utilized to enhance the performance and efficiency of cryptographic applications [39].

For a long time, the IC supply chain was thought to be well-protected behind the barriers of expert knowledge and high cost, so attackers could not easily compromise the fabricated chips. However, as mentioned in section 1.2, the IC supply chain has become global due to the high cost of cutting-edge foundries and the increasing design complexity of modern systems. In line with this trend, third-party fabrication services and IP usage prevailed in contemporary circuit design and fabrication. This might alleviate the workload, lower the fabrication cost, and reduce time-to-market, but it also invalidates the idea of a well-protected supply chain [39][40][41][21][1].

2.2.1 Hardware Trojans

At first, HS was mainly concerned with discovering and containing Hardware Trojans (HTs). Compared to their software counterparts, they are more dangerous as updating the firmware cannot easily remove them. HTs, potentially inserted at any phase of the IC design, have a malicious design to alter or add circuit elements, which can result in changes to functionality, reduced reliability, or information leaks. Their insertion can occur at the register transfer level (RTL) or through dopant manipulation, while their design depends on the attacker's goals and available resources. They consist of two parts: trigger and payload. The trigger, acting like a sensing circuitry, activates it to perform a specific task while the payload carries out malicious actions. A Trojan can be triggered to carry out a specific task or be always on. Always on HTs solely contain a payload [21][39][40][41].

As shown in Figure 2.3 [21], HTs are a vast theme in itself. With this being said, it becomes clear that PUFs are not the miraculous solution to prevent or detect them. However, given their capabilities for authentication, it becomes an exciting vector for further research.

2.2.2 Counterfeiting

The challenges for HS are only increased due to the nature of this multinational and distributed supply chain and the reliance on, sometimes untrusted, third parties. This creates opportunities for attackers to counterfeit, clone, overproduce, or reverse engineer a given IP core or the entire IC design, ensuring an illegal revenue of billions of dollars [39][41][42]. Both [42] and [43] define counterfeited as an umbrella encompassing the concepts naming this subsection. However, we believe each concept has a very tangible and unique definition, hence our separation.

In IP/IC counterfeiting, attackers create counterfeit versions of original designs and sell them under a genuine supplier's brand name [41]. In [43], the author mentions the recycling and resale of discarded chips as new ICs, labeling them as counterfeit. Labels aside, a standard solution is on-chip aging sensors, namely ring oscillators (ROs), anti-fuse memory, and fuse memory.

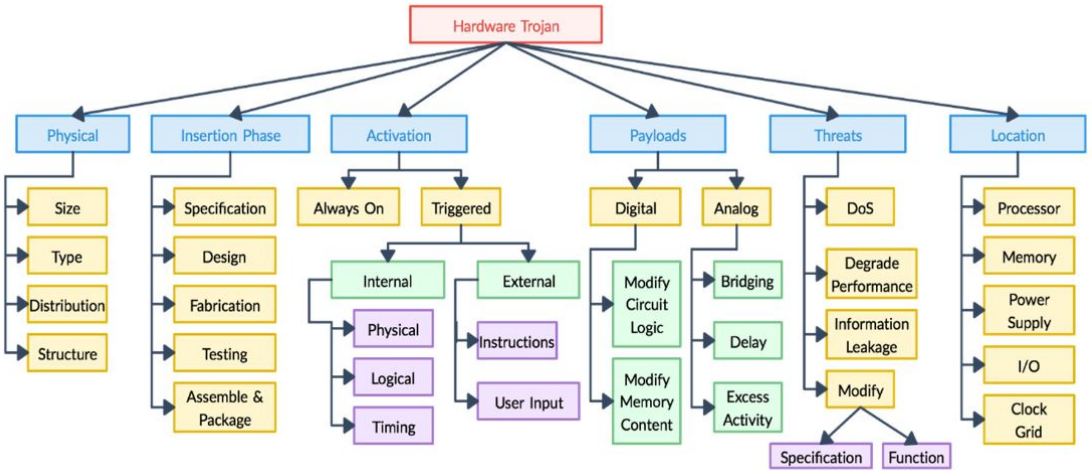


Figure 2.3: A taxonomy of Hardware Trojans [21]

Another proposed solution is a secure, distributed, and tamper-proof database, e.g., blockchain, which allows for the verification of whether an IC has been sold more than once [39][43].

ROs used as aging sensors tend to be isolated from the rest of the IC's components, making them an ideal target for either modifications or, in some cases, removal. Although tamper-resistant, as they are written only once and cannot be deleted, the usage of anti-fuse memory is limited due to large area requirements and the needed match between the frequency of its write operations versus the maximum number of planned write operations, as the IC cannot be turned off before the next memory write function. Fuse memory, albeit extremely compact, is also susceptible to manipulations. Lastly, to provide each chip with a unique ID, we rely on manufacturing process variation [43]. Having presented all these vulnerabilities, it becomes clear how the introduction of a PUF could help to improve or even allow some of the previously shown solutions [32].

Watermarking and hardware metering are two other commonly used techniques to prevent counterfeiting. Watermarking consists of a hidden signature embedded at design time, checked against its intended attributes for authenticity verification. These watermarks will identify a given design, not individual IC instances, hence, useful for tracking stolen design IPs in the supply chain [32][44].

Hardware metering enables post-fabrication tracking of ICs to differentiate genuine hardware from counterfeits through tools, methodologies, and protocols. It can be either passive or active. Passive identification refers to the specific recognition of ICs based on their function or unique identifiers. These identified ICs can be cross-checked against a pre-existing database to detect any unregistered or overbuilt ICs (in case of collisions at the database). Active metering empowers designers and/or IP rights owners to access, lock (disable), or unlock (enable) particular aspects of the chip's functionality through advanced design knowledge not shared with the foundry, besides the ability to distinctly identify ICs [32][44].

Metering can be further classified into intrinsic or extrinsic. Intrinsic uses process variation to

create unique fingerprints based on existing device properties or side channels, such as delay and power. These methods are inherently passive. Extrinsic uses extra hardware or software to create a unique ID for each device. There are passive and active methods for this. Once identified by hardware metering techniques, the foundry or other reliable sources can be contacted to obtain more information about the IC [44][32]. Again, establishing a connection between these techniques and using PUFs for their implementation and/or improvement is almost natural.

2.2.3 Cloning

In the semiconductor industry, cloning involves attackers creating cloned versions of original IPs/ICs under their labels. These are not only a liability for the original creator's revenue and reputation but may lead to the integration of fake IPs/ICs in electronic systems. This might seem manageable at first sight, but these can affect the reliability and performance of the systems they might integrate. Also, they could potentially contain malicious or backdoor logic, which is undesirable, especially in critical systems [41].

2.2.4 Over-production

Over-production refers to the intentional production of excess ICs. After the design is passed on to the foundry, the designer has limited control over the production process. Therefore, the foundry could produce the quantity requested or create an excess. To address this issue, a technique called split manufacturing is employed. This involves dividing the design between trusted and untrusted foundries so the designer retains complete control over the design data. The design is split into "front end of line" (FEOL) and "back end of line" (BEOL) components, which are fabricated by trusted and untrusted foundries, respectively. This technique can also prevent HTs [41][39].

2.2.5 Reverse Engineering

Reverse engineering (RE) toolsets are being improved and used more frequently for circuit and IP core duplication. A reverse engineering attack aims to uncover the design structure or functionality using various forms like RTL, netlist, layout, mask, or a manufactured IC. These attacks, if successful, could lead to the insertion of backdoors, HTs, or counterfeiting. CMOS camouflaging is commonly used for safeguarding IP, which entails obfuscation at the layout level. Nevertheless, such an approach can result in a substantial performance burden. Other potential solutions involve design obfuscation and logic encryption, which, although robust given the attacker's need to find both netlist and obfuscation keys, negatively impact performance and necessitate a layout overhaul [41][39].

2.2.6 Side-channel

HS may also be at risk of side-channel attacks (SCAs). ICs emit signals similar to human heat signatures during their regular operation. Such unintentional information "leakages" allow attackers to obtain internal signals, typically without physical intrusion, through static or differential analysis of these side channels. This requires the observation of certain functional or parametric behaviors at runtime. For example, event timing, power consumption, supply current, electromagnetic emissions, etc., could be used to leak on-chip secrets. Fault injection attacks are a highly effective SCA. In these instances, the attacker intentionally induces a fault in the hardware to disrupt its execution, potentially exposing sensitive information, such as key bits. Despite efforts to optimize design, countering these threats can still significantly impact performance [41][39]. As previously discussed, PUFs have the potential to mitigate certain security risks. However, safeguarding against SCAs poses a challenge, as shown by Tajik et al. [45] who shows how a fault injection could be used to model a PUF, replicating it successfully.

After conducting a brief survey on the various risks associated with hardware security, we recognize the critical importance of continuously exploring and developing advanced techniques to address safety, protection, and authentication issues in hardware devices. HS research is evolving towards establishing physical roots of trust as a solid foundation for secure systems [39].

2.3 Standard Pratices

Having utilized the previous section 2.2 to present and describe the threats, it is now essential to understand the current practices regarding safety, protection, and authentication for hardware devices.

Standard security techniques have traditionally involved placing secret keys in non-volatile memory (NVM), such as electrically erasable programmable read-only memory (EEPROM) or battery-backed static random-access memory (SRAM), while bolstering their protection with additional hardware cryptographic operations(e.g., secure hash algorithms). Nevertheless, this conventional approach presents some hurdles, including increased area, power consumption, and the requisite for active tamper detection to thwart invasive attacks against the NVM. These drawbacks have prompted the exploration of alternative solutions, from where silicon PUFs emerged as a promising alternative. Their tamper-resistant keys, unique to each instance and produced on demand instead of stored in NVM, offer a unique advantage, as they're based on nanoscale structural disorder, assumed impossible to reproduce, even by the original manufacturer. This also means that any attempt to access or probe the system physically would likely interfere with the entropy source and change the readout during the subsequent evaluation. Such an anti-tampering feature also helps to prevent certain SCAs on the PUF's electronics. So, a PUF can serve as the physical root of trust, making it a cost-effective and attractive option, particularly for devices with limited resources [6][21][37][33] [45] [18]. Compared to standard secure digital storage [6]:

- PUFs are a simpler and easier-to-fabricate solution, translating into a reduction in power consumption and area compared to EEPROM/SRAM solutions with anti-tamper circuitry;
- To generate a key, the PUF must be powered, so any attempt of a physical attack must happen while it's still on;
- The need for continually powered active anti-tamper mechanisms ceases to exist
- Cheaper when compared to expensive NVM;

2.4 The model of a PUF

A model can aid in comprehending how a PUF circuit works. At its core, a PUF is a physical process that translates external electrical stimuli - termed *challenges* - into unpredictable, unique, but repeatable internal reactions, or *responses*. The said model could be defined as

$$P : C \rightarrow R$$

such that $P(c) = r$ expresses the relationship between a challenge c and its response r, where $c \in C$ and $r \in R$. The connection between a challenge and its corresponding response is commonly known as a Challenge-Response pair (CRP). These pairs are generated by the "function" $P(.)$, which represents internal manufacturing variability unknown to user and manufacturer alike. As a result, each CRP is unique and can serve as a distinctive feature to identify and characterize a specific PUF [6][21][37][46].

While utilizing the mathematical definition of a "function", it comes as no surprise to say that for a given challenge c, a PUF should always produce the same response r. As helpful as this definition is to perceive their behavior, it is crucial to acknowledge the reasoning for using quotation marks. As with any IC, environmental conditions impact a PUF's operation, so for an identical challenge c, a different response r could be produced, affecting the stability and dependability of its response. However, it is also worth noting that a change in the CRP is not inherently wrong. After all, it should ideally change due to process variability when creating a new PUF instance. Though the new instance will operate similarly, utilizing the same design and blueprints, its internal components vary during manufacturing, resulting in (ideally) unique responses from each PUF instance, reinforcing the role CRPs take as the PUF's fingerprint. Hence, a conclusion can be drawn on how the security and liability of a PUF rely not only on the difficulty of manufacturing two identical chips but also on the difficulty in measuring/estimating the parameters within $P(.)$ [6][21][33][37][45][46]. Availing this topic, it is essential to address a common misconception. When examining the CRPs, one might assume that having more would enhance security. However, since the number of responses increases proportionally to the number of evaluated challenges, this linear relationship does not yield a substantial enhancement in safety. To beneficially increase the CRP space of a PUF and the difficulty of possible cloning, we should increase the

number of challenge properties involved in the PUF evaluation process. Growing it, e.g., by evaluating two unique PUF properties instead of only one, will compound multiplicatively with each other to produce the output since each permutation of the properties leads to its response, thus scaling the CRP space polynomially. However, there will be a trade-off since this increases the difficulty when evaluating the PUF [18].

Figure 2.4 [37] helps us to understand better these three concepts of challenge, response, and CRP, as well as their interactions within the PUF system. In the upper part of it, we see how $P(X)$, the inherent process variability of a given PUF_X, generates for every challenge C_X a response R_X, therefore creating the unique CRP of PUF_X, named CRP_X. In the middle of Figure 2.4 [37], it is possible to observe how for each $PUF \in [1,n[$, $P(n)$ ensures that for each unique PUF_n, despite being subjected to the same challenge C_X, R_n is unique, hence different from previous and future responses given by other PUFs. At the bottom, we can see how a given PUF_X should obey the "function" model and, for different challenges, generate different responses. An interesting alternative to the strict boundaries imposed by this model can be found in [47], where a PUF is defined as a procedure with input-output functionality rather than a function.

Having defined what a PUF is and how it behaves, one needs to gather some common properties to distinguish between what is considered a PUF and what is not. These can be described as [18][37][47]:

- Evaluatable: given P and c, it should be easy and feasible to produce a response r such that $r = P(c)$. To be "evaluatable" in theory, a PUF should be able to be evaluated in a reasonable amount of time and effort. In practice, this means that evaluation should not create excessive overhead. If a PUF is evaluatable, it is assumed to be realizable and constructible. However, the feasibility of their evaluation also depends on the specific application and whether the overhead is practical;

- Unique: with $P(.)$ containing some information about the identity of the physical entity embedding P, and this information varying from entity to entity, the CRPs can be used as a unique identifier, as no two PUFs should be the same;

- Reproducible: to ensure reproducibility, the PUF must be able to correct any divergence that may occur due to factors such as the physical environment or the system's characteristics. This ensures that the same response $r = P(c)$ can be consistently generated with minimal error, meaning it should be closely aligned with the distance metric being considered;

- Unclonable: a core property for a PUF translates the impossibility of finding a corresponding response r for challenge c without the physical PUF. This is due to *P(.)* and the process variation that's inherent in it. Therefore, it should be impossible for an adversary, even with physical access to PUF_i, with "function" *P(.)*, to create a procedure, with "function" *P'(.)*, such that $\forall\, c \in C : P'(c) \simeq P(c)$. It is important to note the word "procedure", as a distinction between physical and mathematical unclonability must happen. For true unclonability,

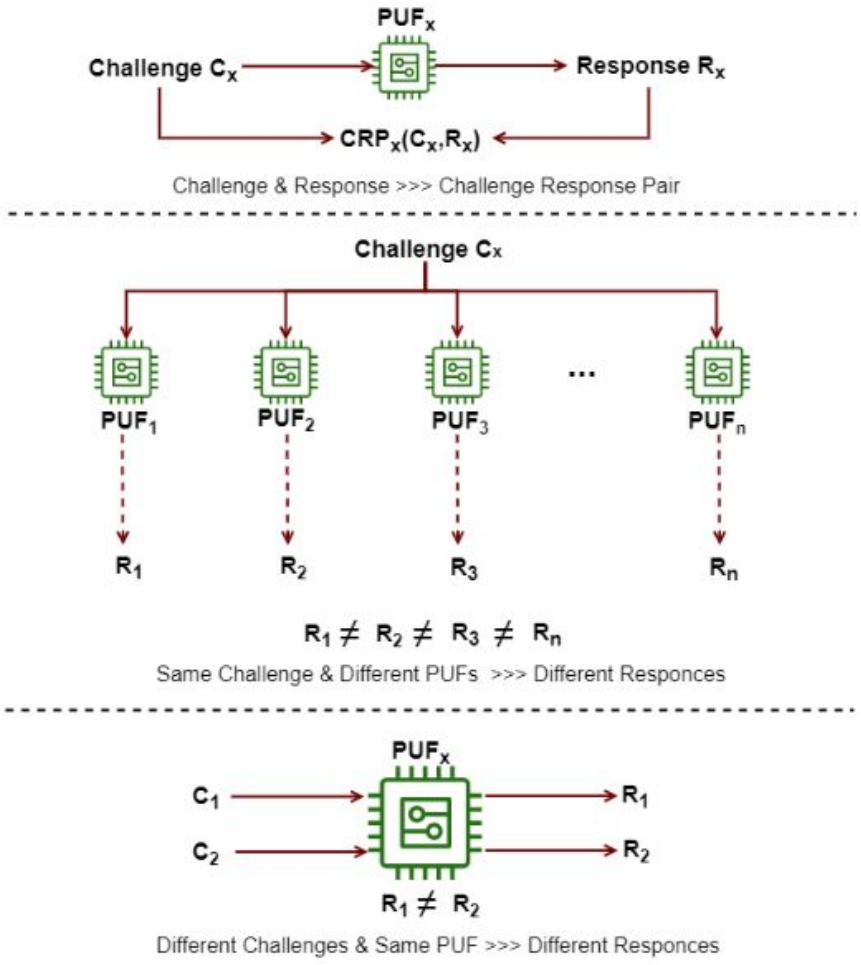


Figure 2.4: Expected behavior of a PUF's CRP [37]

P must possess both properties. A mathematical clone translates the possibility of finding a procedure f such that $\forall\, c \in C : f(c) \simeq P(c)$, emulating P;

- Tamper evident: the act of tampering involves permanently altering the integrity of a physical object. Two categories exist: tamper-proof systems, which provide no valuable information if tampered with, and tamper-evident systems, which may be tampered with but leave detectable evidence. Being embedded into a physical entity, any alteration to this entity will likely convert P into P', and, most likely, $\exists\, c \in C : P(c) \neq P'(c)$. Even with a small error, $P(c) \not\equiv P'(c)$. Therefore, PUFs are regarded as tamper-evident, as it is highly likely tampering will change the CRP;

As we delve deeper into the intricacies of PUFs, we must define the criteria by which they are evaluated. Determining these metrics can be challenging since no regulatory standard is in place yet for this relatively new concept. Consequently, the literature on PUFs reveals that while specific standardized metrics exist, variations in others can be observed depending on the author's

perspective. Hence, we use uniqueness, reliability, uniformity, and bit-aliasing. A definition of these can be seen in the following subsections 2.4.1, 2.4.2, 2.4.3, and 2.4.4, respectively.

2.4.1 Uniqueness

Ideally, each PUF should have a distinct CRP, even if produced using the same manufacturing process. The uniqueness of each PUF is the standard used to determine their individuality. Ideally, the degree of uniqueness ought to reach approximately 50%. This implies that when assessing the same challenge on diverse PUF instances, the responses obtained should differ significantly with a high likelihood. Such a characteristic empowers the distinction of a particular chip from a collection of chips of the same category. We can express it as an estimation of the inter-chip variation in terms of the PUF responses. So, having two different PUFs, i and j with n-bit responses, R_i and R_j, respectively, for the challenge c, the average inter-chip HD among k PUFs is defined as [18][37][48]:

$$Uniqueness = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD(R_i, R_j)}{n} \times 100\% \tag{2.3}$$

2.4.2 Reliability

It aims to measure the repeatability and consistency with which a PUF generates its response. After all, it is wishful that a PUF's response remains stable over time and whatever environmental variations it may encounter. One effective method of measurement is through intra-distance measurements of multiple responses. This involves taking an n-bit reference response R_i from PUF_i under normal operating conditions (room temperature and normal supply voltage) and comparing it with the same n-bit response under varying operating conditions, denoted as R'_i. If m samples of R'_i are taken, for PUF_i, the average intra-distance can be estimated as [18][37][48]:

$$PUF_{i_{INTRA}} = \frac{1}{m} \sum_{t=1}^{m} \frac{HD(R_i, R'_{i,t})}{n} \times 100\%$$

where $R'_{i,t}$ is the tth sample of R'_i. Through $PUF_{i_{INTRA}}$ we can quantify the average number of unreliable response bits, finally defining reliability as:

$$Reliability = 100\% - PUF_{i_{INTRA}} \tag{2.4}$$

The closer to 100%, the more reliable a PUF is [18][37][48].

2.4.3 Uniformity

To ensure that a PUF is genuinely random, its response bits should have an even distribution of '0's and '1's. This even distribution is known as uniformity and is measured by the percentage Hamming weight (HW) of an n-bit response. The HW calculates the number of non-zero symbols in the used alphabet, with a string like '11101' having an HW of 4, for instance. To meet the

criteria of genuine randomness, the proportion of '0s and '1s in a PUF must be 50%. For a PUF_i with a n-bit response r, uniformity is defined as:

$$Uniformity = \frac{1}{n}\sum_{l=1}^{n} r_{i,l} \times 100\% \tag{2.5}$$

where $r_{i,l}$ represents the lth bit of the n-bit response from PUF_i [37][48][49].

2.4.4 Bit-aliasing

Bit-aliasing is a metric used to assess the bias of a response bit in a group of PUF instances. To calculate the bit-aliasing for the lth bit of PUF_i when presented with a challenge c, we determine the percentage of the HW for this bit of the PUF's response across k devices. The desired outcome is for this value to be as close to 50% as possible, as any deviation can lead to similar PUF responses from different chips, which is not optimal. Bit-aliasing can be calculated by:

$$Bit-aliasing = \frac{1}{k}\sum_{i=1}^{k} r_{i,l} \times 100\% \tag{2.6}$$

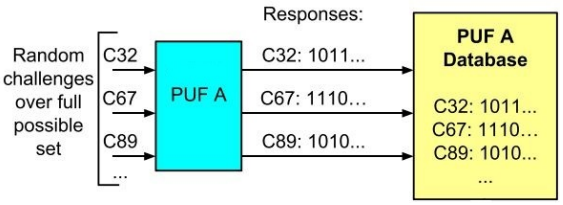
where, again, $r_{i,l}$ represents the lth bit of the n-bit response from PUF_i [37][48].

2.5 Operation of a PUF

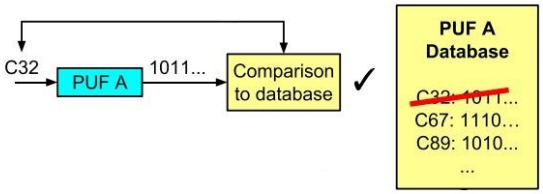
After presenting a model to explain the characteristics and evaluation metrics of a PUF in the previous section, it is now crucial to understand its operation in an actual application.

In various application scenarios, a PUF is presented with a challenge during verification or authentication processes, subsequently generating a response. To ensure the smooth continuation of the process, it is imperative to determine the correctness of the generated response accurately. So, during manufacturing, the system's CRPs are recorded and stored in a database. This process is called enrollment. Therefore, in a given application scenario, by leveraging the unique physical features of hardware, we establish a system where if the correspondence between a repeated challenge and its anticipated response aligns with the one saved in the database, a successful verification/authentication is confirmed. Any device attached or embedded with the PUF is, thus, uniquely authenticated [18][46].

Figure 2.5 [18] will help us illustrate how this process should play out. In Figure 2.5a, we can see that the PUF's responses to random challenges are recorded in a database. Later, in Figure 2.5b, when the PUF is presented with a repeated challenge, its response is compared to those stored in the database. If the response matches the saved one, the PUF is considered authenticated; if not, the authentication fails.



(a) Enrollment of a PUF



(b) Evaluation of a PUF

Figure 2.5: Operation of a PUF [18]

2.6 Classifying PUFs

When searching for information on PUFs, you may come across various concepts and designs that differ significantly. To better understand them, we will organize these findings using a classification method proposed by Mcgrath [18]. This method involves first categorizing PUFs based on their randomness's origin, then evaluating how this randomness is measured, and finally dividing the PUFs into different categories.

When it comes to a PUF's randomness and how it originates within its design, we can divide it into explicit or implicit randomization. Explicit, meaning that a PUF's randomness originates from an external source that adds her to the PUF afterward. Implicit means that its randomness naturally arises from uncontrollable effects during the fabrication process. Generally, implicit randomization is preferable since it does not require additional operations, and its randomness (at least physically) can not be cloned [3][18].

We can now look at how their evaluation is performed for further distinguishment. We will have either an internal evaluation or an external evaluation. A PUF whose features are internal and evaluated through mechanisms embedded within (tend to be more accurate, easier to use, and less prone to malefactor interference) is considered an internally evaluated PUF. Combined with implicit randomization, this characteristic originates what is defined as an intrinsic PUF. An example of such a PUF will be our object of study in Chapter 3. A PUF whose features are external and/or are evaluated through mechanisms outside the PUF is considered an externally evaluated PUF. These two definitions and the developed knowledge so far should lead the reader to conclude that intrinsic PUFs will offer the best solution for cryptographic applications. Their internal randomness and evaluation mechanisms make them tamper-resistant and more resistant to man-in-the-middle and side-channel attacks [3][18].

Having categorized the origin of their randomness and how it is evaluated, Mcgrath [18] divides the various PUF concepts into several categories within three grand schemes of classification, these being:

- Organic: a property-driven organization scheme
- Parametric: a parameter-driven organization scheme
- Chronological: a timeline-based organization scheme

We should note, however, that despite this classification, a PUF's concept can constantly be subjected to some degree of variation and specification according to its desired application and that a standardized classification system is still needed.

2.6.1 Chronological scheme

The least essential scheme when it comes to the scope of our work. Exposes a brief, straightforward timeline of different PUF concepts and their proposal dates.

2.6.2 Organic scheme

As the name suggests, this organic scheme is more arbitrary and relaxed than a parametric organization scheme. The author presents four levels: Application, Randomness Source, Family, and Concept [18]. The application level differentiates between the environments where a PUF would operate, dividing them into "All-Electronic" or "Hybrid". According to Mcgrath [18], a design is considered hybrid when the medium used to probe the PUF's source of uniqueness switches from an electronic signal to, for instance, emitted and detected light. The PUF's source of randomness is examined at the randomness source level, as already explained. In the third level, it is presented the family to which a given concept belongs. Here, the family works as an umbrella term for PUFs whose behavior is analyzed by measuring similar characteristics across different PUF concepts, e.g., the racetrack family includes PUFs that rely on comparing or analyzing time delays. Finally, at the fourth level, different PUF concepts are classified according to the more appropriate family. This scheme can be visualized in Figure 2.6 [18].

2.6.3 Parametric scheme

Within this second proposal scheme, there is a division into two levels. These are the Evaluation Mechanism and the Evaluation Parameter, respectively [18].

The first level organizes by what medium the signals that evaluate the PUF exist, e.g., an electronic signal, a laser, etc. The second level sorts the PUFs into categories where the most apparent property parameter examined in a PUF is used as a label, e.g., for an arbiter PUF, the comparison of propagation time between two signals [18]. Figure 2.7 [18] presents this scheme.

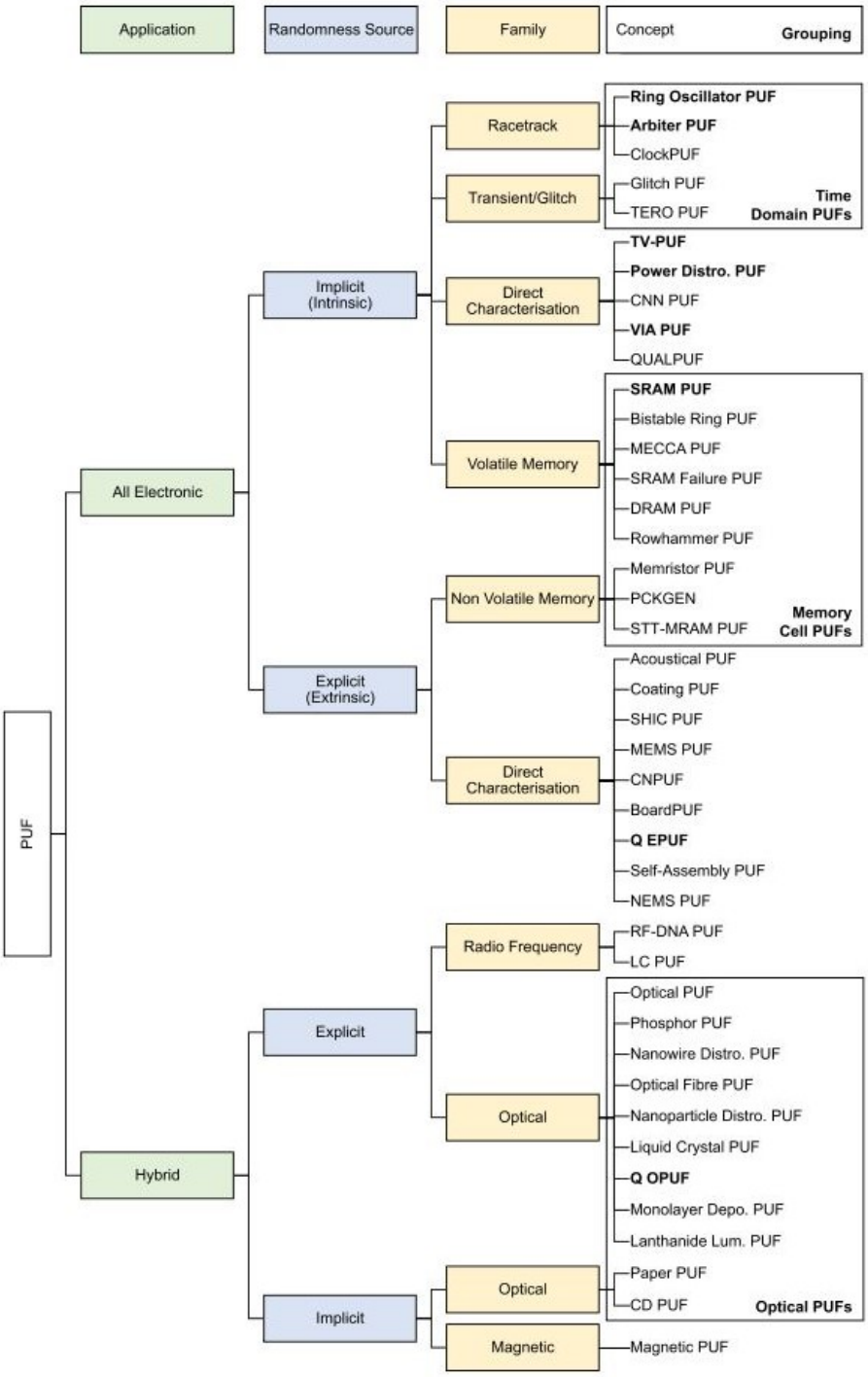


Figure 2.6: Organic scheme [18]

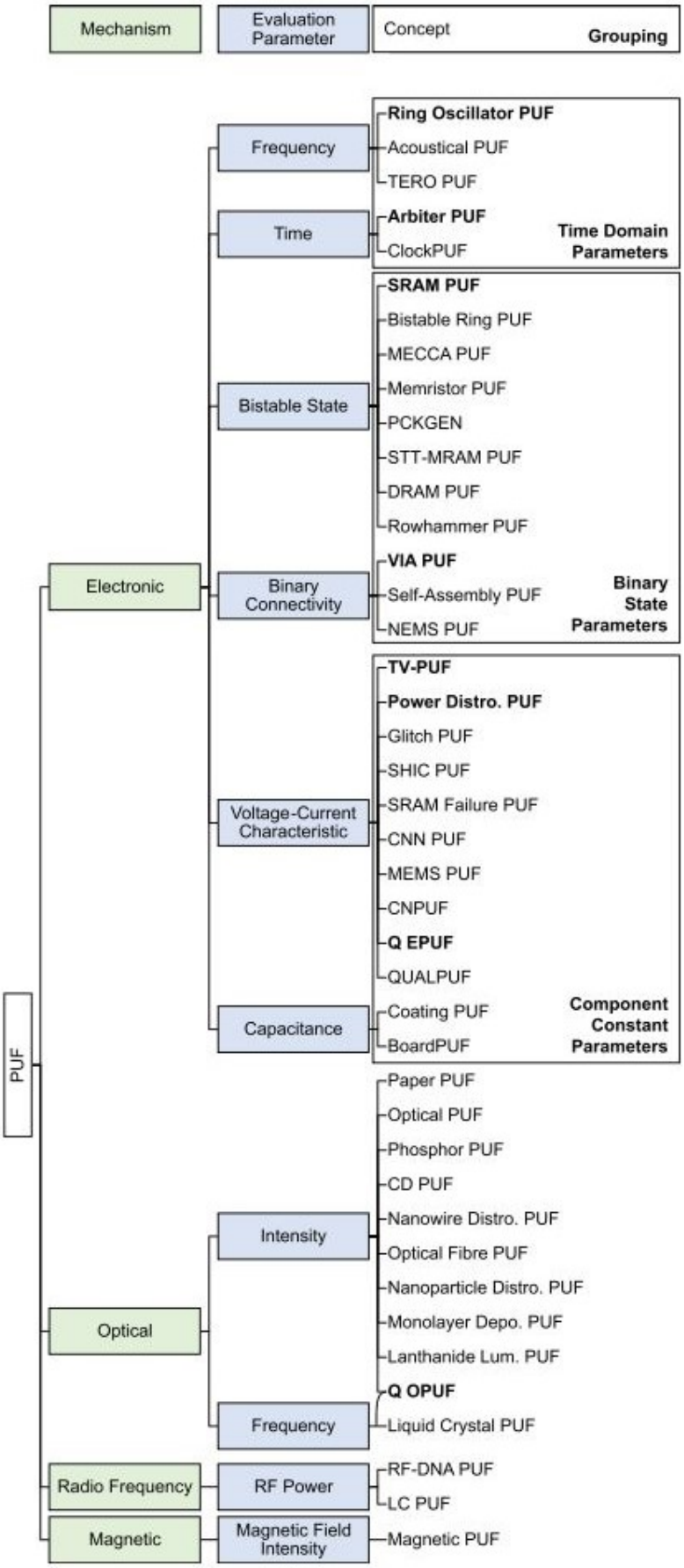


Figure 2.7: Parametric scheme [18]

2.7 Functionality

After discussing the operation of a PUF, it becomes apparent that the quantity of CRPs at one's disposal can affect their range of use. As such, PUFs are also distinct based on their strength, which can be either weak or strong. The strength of a PUF is ascertained by the number of unique challenges it can process and the resulting limit on the number of CRPs that can be generated from a single device [6][18][37].

Weak PUFs, also known as Physically Obfuscated Keys (POKs), are limited in terms of the number of challenges they can support, supporting a small number of challenges or even just one, resulting in a reduced number of CRPs. This number is directly related to the number of components that rely on manufacturing variation, leading to a unique digital signature suitable for cryptographic purposes like key derivation. For this purpose, their responses must be highly reliable and highly unpredictable. Moreover, keeping their responses confidential and concealed during operation is essential to prevent attackers from emulating the PUF's behavior by observing the CRPs. Their output can be used, for example, as a key in a keyed-hash message authentication code (HMAC) or as a secret key for encrypting/decrypting data on the device. In conjunction with other cryptographic hardware, they can also provide authentication capabilities [6][18][37].

Oppositely, strong PUFs support many challenges, meaning they can be directly authenticated without further cryptographic hardware, making them commonly used for authentication. Their number of CRPs is so large that even if an attacker gains access to it, it should be impossible to record them all within a fixed time (ideally, exponential in the number of challenge bits). Additionally, the collected sample should not enable the derivation of a response to a new, randomly selected challenge. Each CRP in the extensive repertoire might only be used once, preventing eavesdropping attackers and supporting secure communication protocols that utilize the PUF. While privacy is unnecessary for the responses, the PUF must not expose any information about its internal structure or operation. Also, it's essential to be aware of any attempts to completely list their responses when using them. If the CRPs of a PUF are known, it's considered a corrupted PUF. It's worth noting that the faster a PUF's response, the easier it is to enumerate its CRPs. Therefore, it's essential to strike a balance between the readout time of a strong PUF and the number of CRPs it has, even if a fast response is desired in certain applications [6][18][37].

The strength of a PUF is often measured by how the number of potential CRPs increases with the size of the PUF. A PUF is deemed strong when its CRP count rises exponentially with its size. Conversely, if the growth is linear or polynomial, it is usually considered weak. However, some PUFs with linear or high-order polynomial CRP growth are still referred to as strong [18].

While the previous classification may suggest that strong PUFs are more versatile than weak ones, it is crucial to acknowledge that a strong PUF cannot be vulnerable to modeling attacks. These attacks rely on machine learning techniques to forecast the missing information in a limited set of CRPs and create rules to calculate responses to future challenges [18][50].

Examples of weak and strong PUFs will be given in sections 2.9 and 2.10, respectively.

2.8 Error correction

Both weak and strong PUFs rely on the analog physical properties of the fabricated IC to derive secret information. Naturally, these analog properties have noise and variability associated with them. If these properties vary too much, the PUF's response might change, condemning the cryptographic operation to fail [6].

Although key generation has zero error tolerance, PUF authentication usually incorporates an allowable error threshold, decreasing the stability requirement and often obviating the need for error correction. A common mechanism to mitigate such errors involves adapting differential design techniques to cancel out first-order environmental dependencies, e.g., gate delay. Typically, PUFs that use this effect will not measure a single gate's delay but rather the difference between two identically designed but distinct gates on a die. Hence, any environmental factor should affect each gate equally [6].

However, despite the multiple error-correction techniques applied to these bits by modern PUF designs, many error-correcting methods have been shown to leak secret key bits since they require the computation and public storage of syndrome bits. So, an excess number of PUF bits are generated and then downmixed to produce a full entropy key [6].

Adding to these techniques, PUFs can also use soft-decision coding. This coding technique takes advantage of the reliability information of a given response bit to improve error correction performance. This reliability information is obtained through repeated PUF response readings in the case of SRAM PUFs or the magnitude of frequency difference values in the case of ring-oscillator PUFs.

2.9 Weak PUFs

They are usually considered PUFs that directly digitize some "fingerprint" into the circuit. Since this fingerprint should remain largely invariant, it can only be questioned by a few challenges, sometimes only one. This translates, as stated before, into $f(.)$ having a smaller domain and also a minimal scope given that a specific challenge should always result in the same response [46][6].

These so-called "fingerprints" are nothing less than the cryptographic keys spawned and stored by the PUF, which are then compared to an external database for identification/authentication or used as part of other protocols, such as secure communication or memory encryption [18].

A weak PUF can be characterized as having [6]:

- A smaller number of CRPs;
- A response that is stable and robust to environmental conditions;
- Being impractical to manufacture two devices with the same physical fingerprint;

Having a small number of CRPs, these must be kept secret since disclosing them would translate into a compromised PUF. Although impossible to copy, an attacker could still use another device to emulate its CRPs, rendering the PUF useless [6][18].

Weak PUFS are a good fit for key derivation processes with their randomness and secure storage, presenting an alternative to secure nonvolatile memories. Once recovered, the key derived from the PUF's response bits serves as a token for the continuation of the cryptographic process, be it authentication, encryption, or other cryptographic protocols. Note, however, that it is still important to store it in a secure volatile memory during the device's operation [6].

2.9.1 Examples of Weak PUFs

2.9.1.1 Ring Oscillator PUF

One such example of a Weak PUF is the Ring Oscillator PUF, whose architecture is comprised of n identically designed ring oscillators, which consist of a NAND gate in line with an odd number of NOT gates whose output feeds back as seen in Figure 2.8 [46].

Once fabricated, the ring oscillator frequency is set, meaning that the output bits of the PUF will always remain constant, despite that due to the variation in delay of the inverters in the ring oscillator, each will have a slightly different frequency [6].

It works by measuring and comparing the frequencies of two oscillators, thus revealing one of the PUF output bits. This means that for n oscillators, we have $n(n-1)/2$ possible pairings, so a maximum of $log(n!)$ bits can be extracted from the PUF [6].

Since it measures differences in gate delay, the ring-oscillator PUF is susceptible to environmental variations and noise sources. So, error correction is most important in this application. One way to do so is by recognizing that oscillators "closer" in frequency have a greater likelihood of causing an output error than oscillators "far apart" in frequency. This happens due to the tiny fluctuations in oscillator frequency due to noise or environmental variations. If the two oscillator frequencies are further apart, it becomes less likely that a bit flip might occur. So, correctly assessing and selecting the oscillator pairs that define the PUF output bits increase the PUF's robustness towards noise and environmental variations [6].

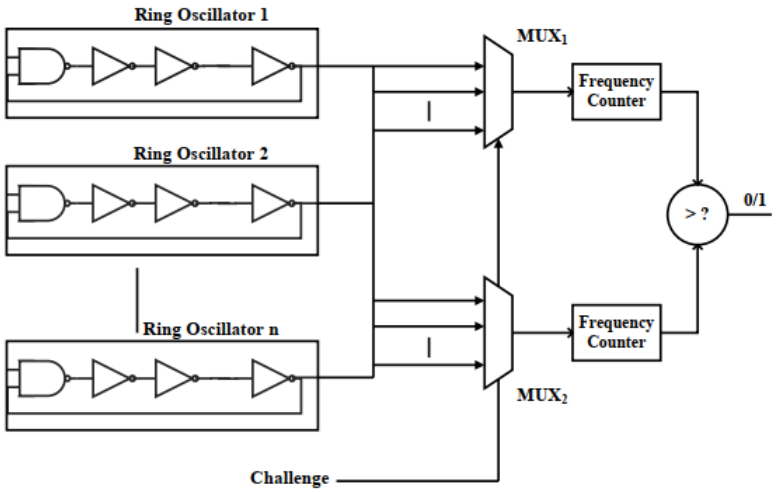


Figure 2.8: Architecture of a Ring Oscillator PUF [46]

2.9.1.2 SRAM PUF

A Static Random Access Memory (SRAM) cell, as shown in Figure 2.9 [18], consists of two inverters in conjunction with two access transistors. Each cell in an SRAM has two stable states, storing either 0 or 1, and positive feedback to force the cell into one of these, preventing an accidental transition. When power is applied, the cell can be written into either state [18].

An SRAM PUF exploits this positive feedback loop to uniquely characterize a system through the variation of otherwise symmetric transistor branches within SRAM elements. Again, the measurement is differential, meaning that the final state originates from the difference between two feedback loops. Hence, common-mode noise such as die temperature, power supply fluctuations, or common-mode process variations should not impact the transition [6][18].

Comparable to the ring-oscillator PUF, this architecture can be used to make intelligent decisions regarding error coding. Cells have a consistent bias towards 0 or 1, so by using repeated measurements, one can assess the stability of an SRAM PUF output bit and selectively use the most stable bits for the PUF's output. Combined with traditional coding techniques, this can mitigate the noise inherent to SRAM PUFs [6].

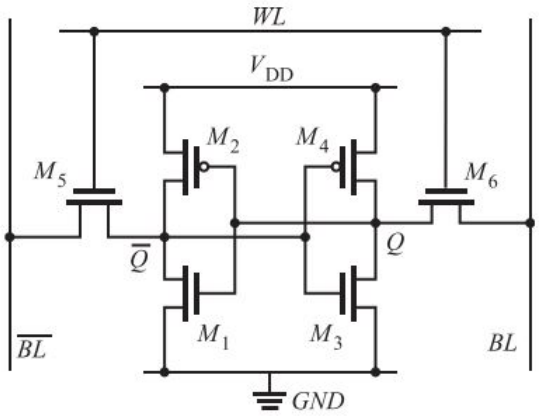


Figure 2.9: Circuit diagram for an SRAM cell [18]

2.10 Strong PUFs

After introducing weak PUFs and providing examples in Section 2.9, we will discuss strong PUFs, recalling their definition and presenting some examples to conclude this chapter.

In Section 2.7, we previously saw that oppositely to weak PUFs, strong PUFs support many CRPs. The requirements to be considered a strong PUF could be resumed in ample CRP space; the revelation of some CRPs should not be enough to infer the responses to newly presented challenges, and their readout should display only the responses, nothing else. A strong PUF can securely identify the embedded or attached device by replacing the secure memory and cryptographic hardware. Since the PUF doesn't need secure nonvolatile memory, anti-tamper circuitry,

or additional supporting cryptographic acceleration hardware, it requires less area, power, and mask layers than a traditional approach to secure authentication [6][18][37].

2.10.1 Examples of Strong PUFs

2.10.1.1 Arbiter PUF

One of the earliest proposals for implementing a strong PUF relied on the manufacturing variability of gate delay as a source of unclonable randomness in what would come to be known as an *Arbiter PUF* [16].

The concept behind an Arbiter PUF is to characterize a system by comparing the travel time of two electrical signals propagating down, theoretically, symmetrical paths, such as measuring individual gate delays. Again, possibly verified deviations originate from the inherent variability in CMOS fabrication. Since the effect of these variations is random per device but static for a given device, the delay difference and, therefore, the PUF's output will be device-specific [3][6][18][19][28].

As shown in Figure 2.10 [3], an arbiter PUF consists of several cells, named *switch blocks*, that connect two input signals to its two outputs. Each *switch block* (lighter gray) consists of two 2-1 multiplexers who either route both signals through a different signal line if the selection bit is one or maintain them in the same line if the selection bit is zero. By aggregating n switch blocks, a total of n selection bits are needed to configure any of 2^n possible pairs of delay paths. It is this *n-bit* setting that will act as a unique *n-bit* challenge for the arbiter PUF [3][18].

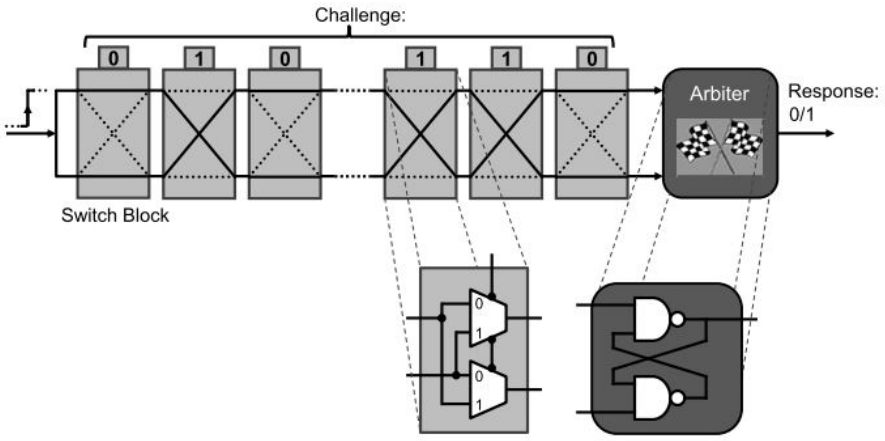


Figure 2.10: Architecture of an Arbiter PUF

Given that input-output delays within the two configurations of a *switch block* are slightly different and randomly affected by process variations, each of the 2^n challenges originates a new race condition between signals. This is where the *Arbiter* comes into play, evaluating which of the two input signals reaches first, returning the response from the PUF to a given challenge as a binary output. This circuit, possibly implemented in several ways, is commonly designed as an SR NAND latch, the best option given its unbiased behavior originating from its symmetric

construction [3][51]. Due to their stochastic nature, a draw among the racing signals can occur. If both signals reach the arbiter circuit virtually at the exact moment, the arbiter enters a metastable state, i.e., the logic output is neither 0 nor 1; it is temporarily uncertain. Once out of this state of brief but random duration, the PUF's output will be an arbitrary binary value independent of the race's outcome, neither device-specific nor static. This scenario and its occurrence are responsible for the origin of noise in the responses of an arbiter PUF [3].

An ideal response should be the outcome of the random effect of variability on the delay parameters of the circuit. For this to be verified, the following conditions should be met [3]:

1. Delay lines are designed to be nominally perfectly symmetrical, i.e., any differences in delay arise from the inherent variability
2. The arbiter circuit is completely fair

Failing to meet one of these conditions originates a biased PUF. If bias is unavoidable, some unbiasing techniques can be used [3].

In terms of PUF security, assuming, for instance, the comparison occurs between individual gate delays, it is safe to say that, even with physical access, an attacker would have difficulties performing such measures. Besides, it is also assumed that any possible invasive techniques utilized would destroy the gate delay properties [6]. Note, however, that the 2^n different configurations are not independent. They are based on several underlying delay parameters linearly added to produce the resultant response bit. As such, arbiter PUFs tend to be vulnerable to challenge-response modeling attacks [3][6][52]. Efforts to contrariety this tendency aimed to introduce non-linearity, difficulting the modeling, and came in the form of XOR Arbiter PUFs, Lightweight Secure PUFs, and Feedforward Arbiter PUFs [6].

2.10.1.2 XOR Arbiter PUF

Another example of a Strong PUF is the XOR Arbiter PUF, and as stated before, it originated from the attempt to add non-linearity to standard Arbiter PUFs. This architecture combines several rows of basic Arbiter PUFs whose outputs are xor'ed to form a single response bit. Usually, they are referred to as *X-XOR Arbiter PUF*, where *X* denotes the number of rows in the XOR Arbiter PUF [6][37].

Although they present a greater resilience against ML attacks, it has been shown that they are vulnerable to a combination of power SCAs and modeling attacks, which jeopardize their unclonability. It is crucial to note that while using output xor'ing can significantly increase modeling complexity, it also decreases stability exponentially. This can harm the efficacy of a PUF in an authentication setting and reduce the accuracy needed for an attack model, as the authentication protocol must tolerate a higher level of intrinsic PUF error [6][53].

Figure 2.11 [6] shows a possible implementation of this architecture. In it, we can see how the same n-bit challenge is passed to the n Arbiter PUFs, who construct the XOR Arbiter PUF, and how their responses are xor'ed at the end.

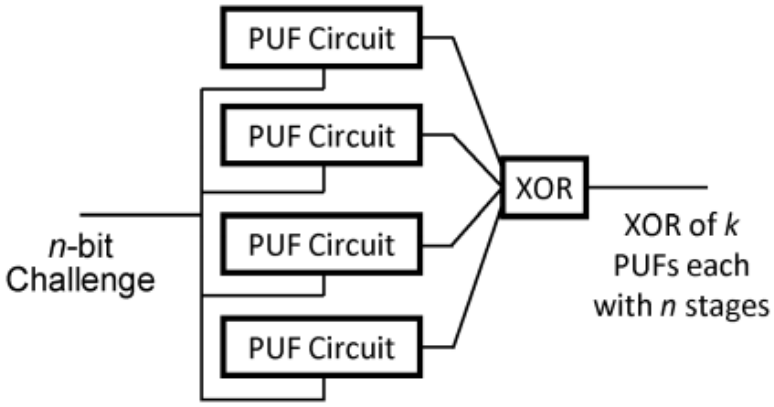


Figure 2.11: Architecture of an XOR Arbiter PUF [6]

Chapter 3

Chosen Design and Evaluation Results

This chapter is the culmination of our study into hardware security, PUFs, and their applications. In it, we will explore a selected PUF design and analyze its performance metrics. We aim to provide a comprehensive design overview, examine its effectiveness, and offer insightful conclusions for the evolution of Physical Unclonable Functions. We will begin by explaining our chosen design in Section 3.1, followed by the simulation results in Section 3.2.

3.1 Chosen Design

Given the many proposed designs currently in the subject literature, we found it essential to select a design capable of encompassing the evaluation of Physical Unclonable Functions so far and showing the future possibilities for this technology. Therefore, if we consider the work presented by Loftstrom et al. [29] in 2000 as the foundational idea to harness process variations and device mismatch as a way to identify an IC uniquely, one understands that PUFs are a relatively recent concept. Within this period, many authors point to the work developed by Gassend et al. [54][16] as the introduction of silicon PUFs and the results from Lim et al. [55][19] for the initial proposal of an Arbiter PUF. Given this context, we opted to study the PUF proposed by Cao et al. [56], as implementing a classic Arbiter PUF would allow for an interesting analysis of both the evolution of PUFs this far, as well as the implications of advanced fabrication technologies on them. The presented energy efficiency and higher temperature stability were also weighted in the choice of this specific architecture due to the importance these factors have when designing and creating PUFs capable of presenting a reliable alternative for secret key generation and device authentication.

The schematic of the proposed PUF can be seen in Figure 3.1 [56]. It consists of a double array of inverters (INVs) alternated with multiplexers (MUXs) stretching for a n-bit length, 64 in this particular configuration, terminating at the arbiter trusted to point which of the two signals first arrived. A further explanation of these three components can be found in Sections 3.1.1, 3.1.2, and 3.1.3, respectively.

Although not implemented in our experiment, we would like to point out that before passing on to each MUX, the challenge bits are pre-proceeded through a Linear-Feedback Shift Register

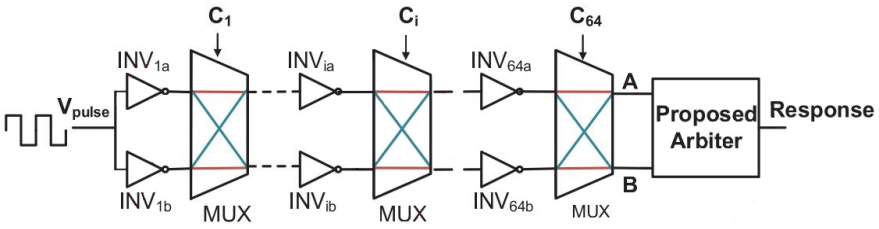


Figure 3.1: Schematic of the proposed Arbiter PUF [56]

(LFSR) acting as a stream cipher. Having them cycle a user-defined number of cycles through the LFSR would allow us to randomize and obfuscate the input challenge, increasing the difficulty of employing modeling attacks against the PUF [56].

3.1.1 Implemented inverter

The approach taken in this Arbiter PUF implementation sets itself apart from the traditional method (illustrated in Figure 2.10). Instead of using only conventional switch blocks, these are paired with current-starved (CS) inverters. An inverter, as the name implies, receives a signal and presents the inverse of it at its output. By incorporating CS-INVs, the delay distribution is broadened and enhanced around its average value, which is essential in maintaining a diverse range of delays in the delay chain. This improves uniqueness for each PUF instance, particularly in the face of different process variations [56, 57]. The CS-INVs are then biased in the zero temperature coefficient point (ZTC) to turn the propagation delay insensitive to variations in temperature [56]. The schematic for the proposed CS-INV is presented in Figure 3.2a [56] and is constructed by adding an NMOS transistor in a standard inverter, whose gate voltage is controlled through the bias circuit. The bias circuit and the bias voltage, V_b, originate from Figure 3.2b [56], and tuning it allows the inverter to reach not only different operating regions but also the ZTC, something verified for $V_b = 750mV$ and the dimensions given in Figure 3.2b. V_b will be shared among all the CS-INVs [56].

(a) CS Inverter

(b) Original Bias circuit

Figure 3.2: Schematic of the proposed CS Inverter and respective Bias circuit [56]

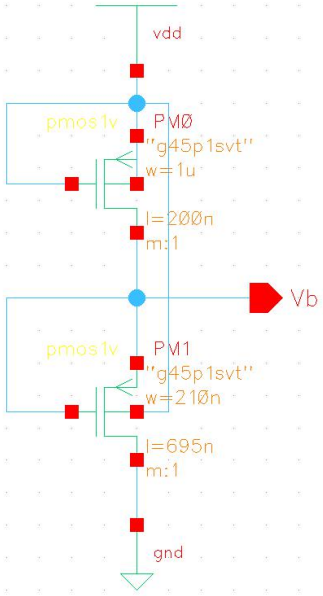


Figure 3.3: Schematic of the designed bias circuit

Cao et al. [56] utilized a 65*nm* process for the design and implementation of its Arbiter PUF; our work was developed using a 45*nm* technology, *gpdk045*, the Generic Process Design Kit 45*nm* Mixed Signal provided by Cadence Design Systems. This meant that all the components of this architecture had to be redimensioned for this technology. Therefore, for the circuit in Figure 3.2b, through experimental measurements, we defined for the upper PMOS the following dimensions, $L = 200nm$ and $W = 1um$ *(2 fingers)*, and the lower PMOS, $L = 695nm$ and $W = 210nm$. The schematic of our developed bias circuit can be seen in Figure 3.3. With these dimensions, V_b was measured at around $870mV$. Nevertheless, the impact of temperature variations on the PUF implemented at 45*nm* was not studied due to timing constraints. Therefore, it is unknown what impact V_{bias} would have or if further redimensioning would be required. The dimensions of the developed INVs were: the PMOS with $L = 45nm$ and $W = 240nm$, the NMOS with $L = 45nm$ and $W = 120nm$, and the V_b's NMOS with $L = 45nm$ and $W = 210nm$. The inverter's PMOS width was double that of the NMOS, following the usual rule of thumb to ensure a balanced operation of the inverter. The V_b's NMOS width was defined experimentally after realizing it needed to be increased. The designed schematic can be seen in Figure 3.4.

3.1.2 Implemented multiplexer

The proposed multiplexers are implemented utilizing four transmission gates, according to the schematic presented in Figure 3.5 [56]. A transmission gate is constructed using two transistors, one NMOS and one PMOS, creating a bilateral switch controlled by externally applied logic levels obeying the truth table in Table 3.1 [58]. If the control signal is asserted low, no signal is passed. If stated high, the signal goes through. Based on Figure 3.5, we designed the schematic presented

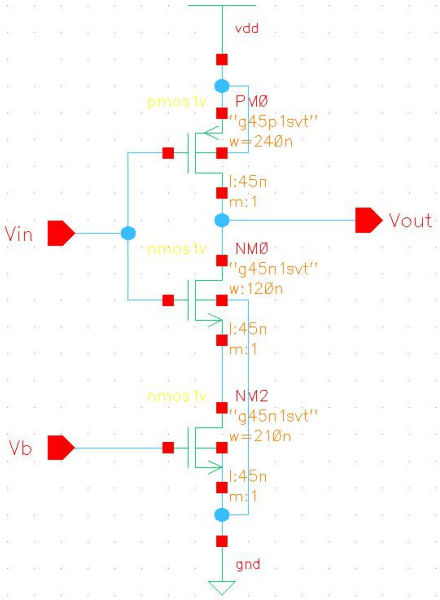


Figure 3.4: Schematic of the designed inverter

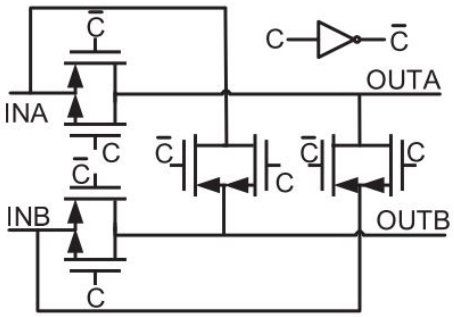


Figure 3.5: Schematic of the proposed multiplexer [56]

Control	A	B
0	0	Hi-Z
0	1	Hi-Z
1	0	0
1	1	1

Table 3.1: Truth Table of a Transmission Gate [58]

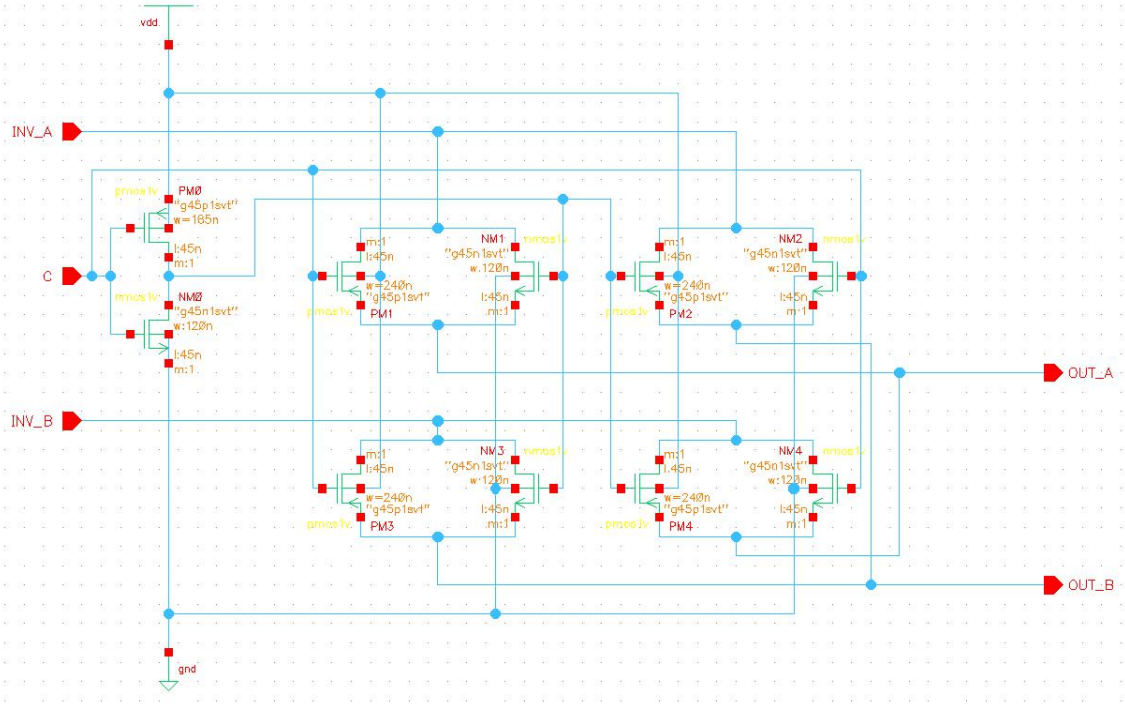


Figure 3.6: Schematic of the designed multiplexer

in Figure 3.6 containing the four transmission gates and an inverter to provide the negated version of the control signal C. Again, to ensure the balance of the transmission gates, the dimensions of the PMOS were $L = 45nm$ and $W = 240nm$, while the NMOS dimensions were $L = 45nm$ and $W = 120nm$. As for the inverter, we have $L = 45nm$ and $W = 185nm$ for the PMOS and $L = 45nm$ and $W = 120nm$ for the NMOS.

3.1.3 Implemented arbiter

The classical implementation of an Arbiter PUF utilizes a D flip-flop as the arbiter. However, as seen in Figure 3.7a [56], given the asymmetries in its internal paths from D to Q and CLK to Q, it may not be a fair arbiter, harming the reliability of the PUF. A better arbiter is achieved by utilizing an RS latch that has symmetric paths and occupies less area [56, 51]. Figure 3.7b [56] and Table 3.2 [59] present the schematic of an RS latch based on two NAND gates and its truth table, respectively. Noting on the first line of Table 3.2, we see that if both $\overline{S}$ and $\overline{R}$ are equal

$\overline{S}$	$\overline{R}$	Q	$\overline{Q}$
0	0	X	X
0	1	1	0
1	0	0	1
1	1	Q_0	$\overline{Q_0}$

Table 3.2: Truth Table of an RS Latch [59]

(a) Schematic of a D-flip-flop

(b) Schematic of an RS Latch

Figure 3.7: Schematics of typically used arbiters [56]

to zero, we reach a prohibited state. Not only both logic gates will output a 1, disrespecting the $Q \neq \overline{Q}$ condition, but there is also the possibility that both inputs might go high simultaneously afterward, creating a racing condition between the propagation time of both gates originating an unknown state with undetermined value [56].

To counter this problem, Cao et al. [56] propose the arbiter in Figure 3.8 [56], consisting of two RS latches and a NAND gate serving as enable to the second latch. Through the active low, enable input $\overline{EN}$, the latch in the second stage is only enabled when both inputs of the first latch are high simultaneously, improving the reliability compared to the conventional RS latch arbiter. Its truth table and proposed CMOS schematic can be seen in Table 3.3 [56] and Figure 3.9 [56], respectively. It is essential to note that this last schematic needed to be clearer to understand, and its implementation was somewhat challenging and time-consuming.

As it is possible to see, three NAND gates are easily identifiable (those with A and B inputs). However, if this schematic is supposed to translate the circuit in Figure 3.8, we should see at least two more NANDs (two from each latch plus the one creating the $\overline{EN}$ signal equals five NANDs total), excluding any additional logic required to implement the $\overline{EN}$ functionality at the second latch. Instead, we see two pairs of cross-coupled NORs, which, by our experiments, makes it impossible to implement the truth table in 3.3. After noting this, through several failed attempted implementations, the decision was made to design the proposed arbiter from scratch. So, maintaining every transistor, both NMOS and PMOS, in their standard dimensions, $L = 45nm$ and $W = 120nm$, we came up with the representation in Figure 3.10. In it, *PUF_ARBITER_RSLatch* corresponds to the first latch in Figure 3.8 while *PUF_ARBITER_2ndRSLatch* corresponds to the junction of the second latch in Figure 3.8 with the NAND gate. *PUF_ARBITER_RSLatch* has R and S as inputs, and Q and $\overline{Q}$[1] as outputs, creating a simple RS latch as presented by Fig-

[1] In the schematics created the notation "NOT_X" defines $\overline{X}$

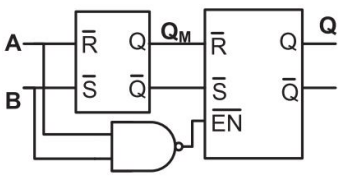


Figure 3.8: Schematic of the proposed arbiter [56]

$\overline{R}$	$\overline{S}$	Q_M	Q
0	0	Forbidden state	Previous state
1	0	Set	Previous state
0	1	Reset	Previous state
1	1	Previous state	Q_M

Table 3.3: Truth Table of the proposed arbiter [56]

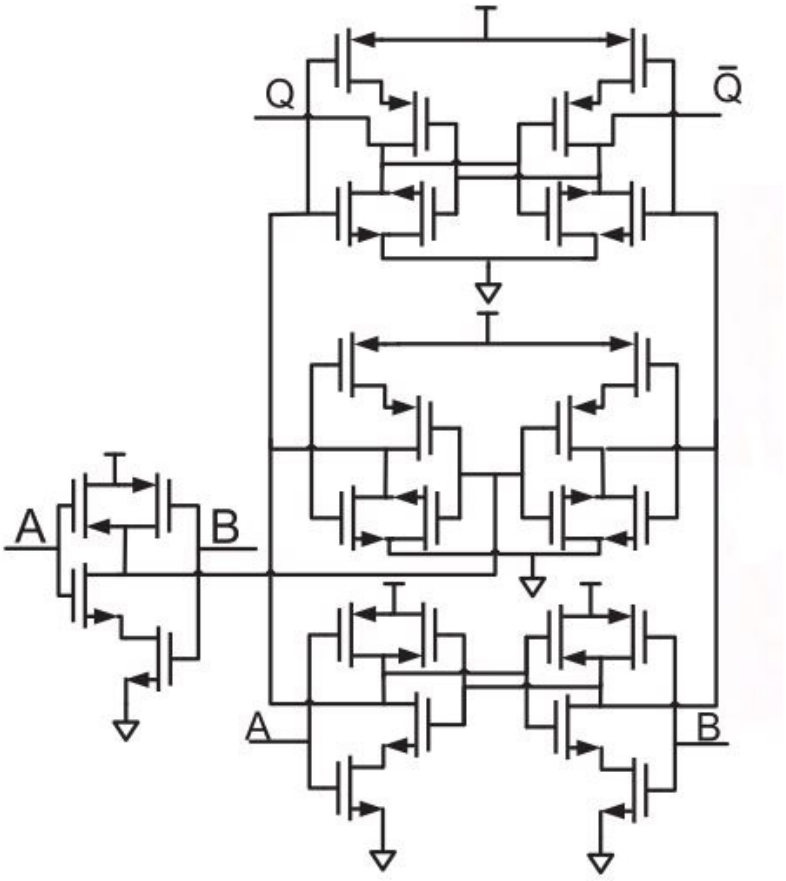


Figure 3.9: CMOS schematic of the proposed arbiter [56]

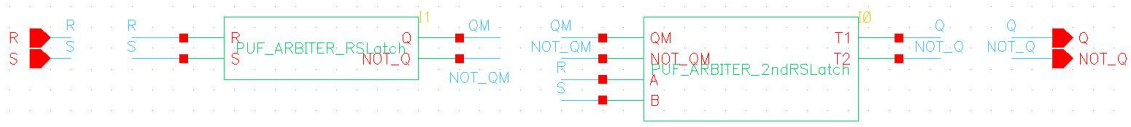


Figure 3.10: Newly designed proposed arbiter

ure 3.11. At the exit of Q, we will have a net named Q_M to represent the signal shown in Figure 3.8. As for *PUF_ARBITER_2ndRSLatch*[2], we constructed it utilizing Q_M, $\overline{Q_M}$, A, and B as inputs, $T1$ and $T2$, as outputs, as per Figure 3.10. For its construction, we utilized two sets of a "NAND+Transmission Gate" combination plus a "NAND+INV" combination, creating the schematic seen in Figures 3.12, 3.13, and 3.14.

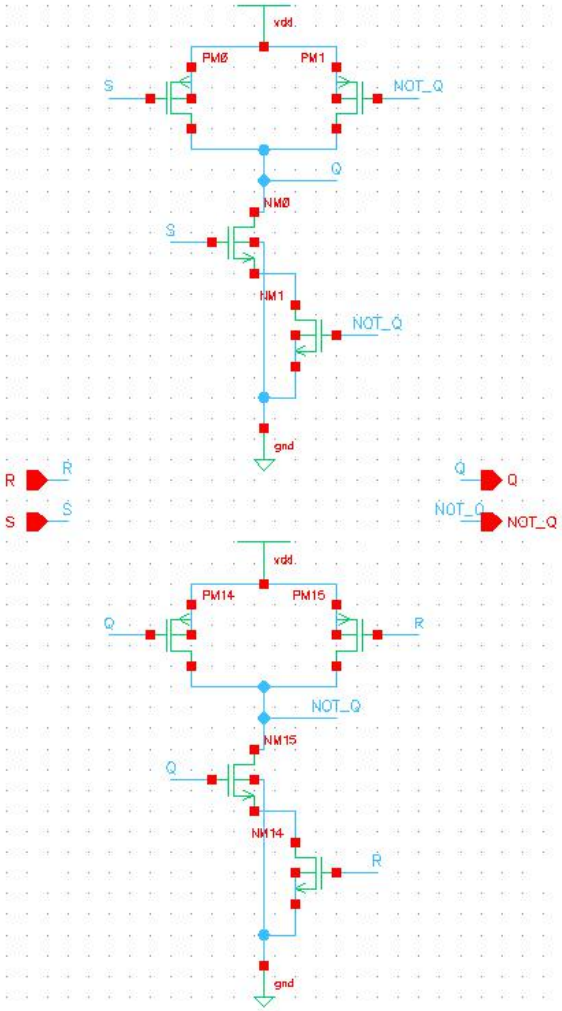


Figure 3.11: Schematic of *PUF_ARBITER_RSLatch*

[2]For a better resolution, its schematic will be presented in three parts, *I, II, and III* corresponding to Figures 3.12, 3.13, and 3.14, respectively

In Figure 3.12, $\overline{Q_M}|\overline{Q} = Q$, where $\overline{Q_M}$ represents the signal coming out of *PUF_ARBITER_RSLatch*'s output, $\overline{Q}$, and $\overline{Q}$ represents the output of the second NAND in *PUF_ARBITER_2ndRSLatch*. This originates Q, then forwarded to the transmission gate controlling *PUF_ARBITER_2ndRSLatch*'s output, $T1$.

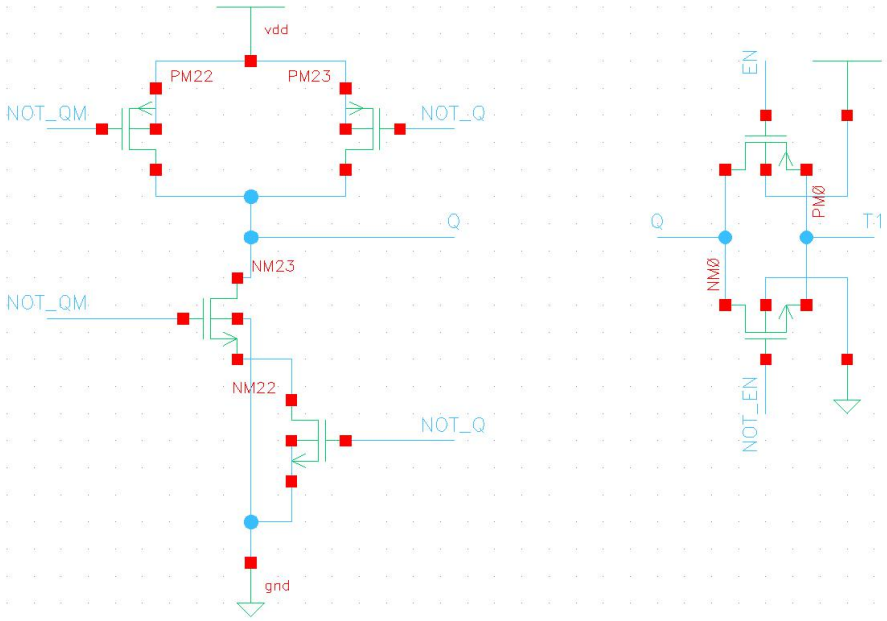


Figure 3.12: Schematic of *PUF_ARBITER_2ndRSLatch* (*I*)

In Figure 3.13, the output of the previous NAND, Q, is utilized as an input in the expression $Q|Q_M = \overline{Q}$, where Q_M represents one of *PUF_ARBITER_RSLatch*'s outputs. The result, $\overline{Q}$, is not only utilized as an input in the previously seen NAND but is again forwarded to another transmission gate, this one controlling the other of *PUF_ARBITER_2ndRSLatch*'s outputs, $T2$. Given the functionality pretended by utilizing $\overline{EN}$ (Figure 3.8), we decided to use two transmission gates depending on this signal to control the exits of the proposed arbiter. For this purpose, the inverter in Figure 3.14 is utilized. This figure presents the third and final part of the *PUF_ARBITER_2ndRSLatch*'s schematic and shows a NAND gate that will receive the signals R and S (A and B in the naming of the nets within *PUF_ARBITER_2ndRSLatch*). Thus, the result, EN, corresponds to the $\overline{EN}$ from Figure 3.8 and, along with its inverted signal, is routed to the corresponding nets to control the transmission gates.

According to Cao et al., the characteristics of this proposed Arbiter PUF can be summed into Table 3.4.

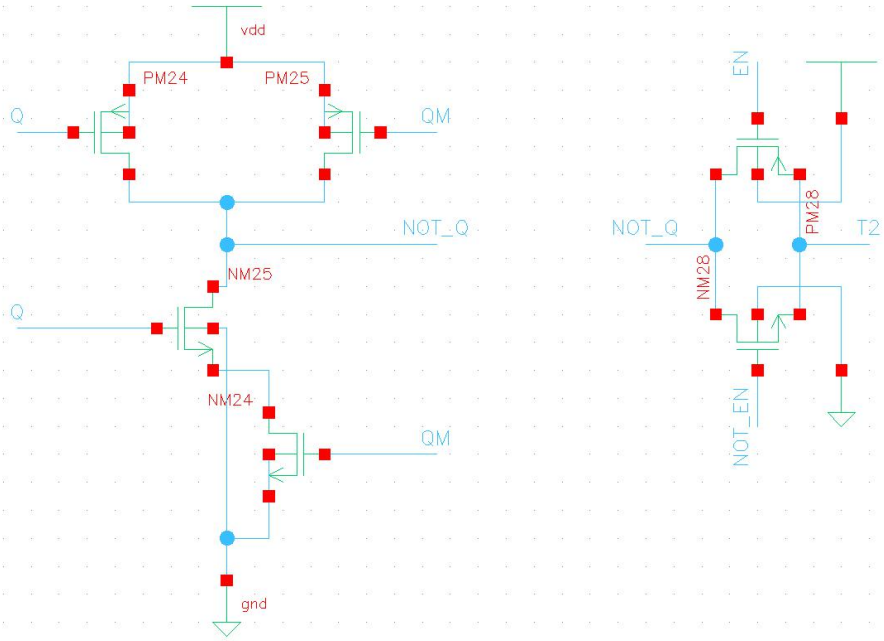


Figure 3.13: Schematic of *PUF_ARBITER_2ndRSLatch* (*II*)

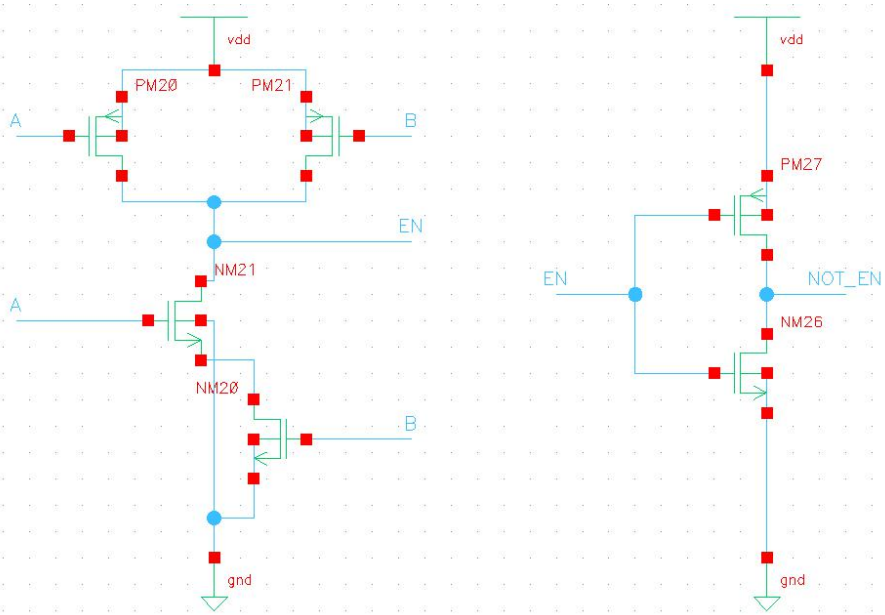


Figure 3.14: Schematic of *PUF_ARBITER_2ndRSLatch* (*III*)

Characteristics	Proposed PUF
Technology	65 nm
Nrº of possible CRPs	1.8×10^{19}
Core area ($\mu m^2/L^2$)	0.91 (normalized)
Supply Voltage (V)	$1.2V$
Voltage Range (V)	1.08 to 1.32
Temperature Range (ºC)	-40 to 150
Frequency	$25MHz$
Inter-PUF HD	46.86%
Intra-PUF HD	0.8%
Uniqueness	46.8%

Table 3.4: Characteristics of the proposed PUF [56]

3.2 Evaluation

Having developed and ensured the correct operation of the previously described components, it became time to employ them in a construction similar to the proposed PUF (Figure 3.1). So, using Cadence Virtuoso 6.1.8 software, we went ahead and constructed four test benches through which we would evaluate the performance of the chosen PUF.

The first two only had eight stages, meaning eight combinations of "2 INVs + MUX" spread throughout its delay chain, the first finishing without the arbiter (Section 3.2.1) while the second finished with the arbiter element (Section 3.2.2). The reasoning was firstly to understand if, through Monte Carlo simulations, we could simulate the manufacturing variability upon which the concept of a PUF rests. Therefore, we constructed the test bench with only eight stages to save on the simulation time.

It was verified that such simulations would only act as expected if "Mismatch" was the *Variation* selected on the ADE Assembler Monte Carlo form, as in Figure 3.15. As its name implies, this option mismatches each element within a given PUF instance instead of "Process", which would vary the process to the entirety of the instance. Our chosen design requires the delay of each path to be unique. This implies that each component in the delay chain of the proposed PUF should have a different delay from the same element in the same position of a separate PUF instance. For this reason, all our simulations were conducted utilizing "Mismatch" and not "Process". Then, given the previously mentioned difficulties in understanding Figure 3.9, and to avoid further delays in the workflow of the thesis, it was decided first to conduct the simulations without an arbiter, while its correct implementation was still under development. Also, the non-utilization of an arbiter allowed us to understand whether eight stages were enough for a noticeable delay.

The final two test benches were designed with the complete sixty-four stages as the proposed PUF. One of them without arbiter (Section 3.2.3), again aiming to understand if the total delay was noticeable or if more stages were needed. The other (Section 3.2.4) was constructed with the

arbiter to verify if it could correctly decide on which signal arrived first and how this decision would permit the identification of individual PUF instances.

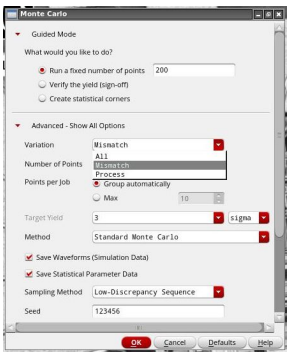


Figure 3.15: ADE Assembler Monte Carlo form

3.2.1 An 8 stages PUF without arbiter

All of our test benches are similar to each other. Obviously, different PUF instances will be evaluated, but, generally speaking, each test bench can be described as the instance under evaluation, a *vdc* cell to provide the test bench's supply voltage (*VDD*), *n-vpulse* cells to commute each of the *n*-challenge bits defining the path, a *vsource* generating the *V_PULSE* provided at the PUF's inputs, and two loads given by *cap* cells. All these cells belong to *gpdk045*'s *analogLib*. For a better understanding, we refer to Figures 3.16 and 3.17, which present the described test bench[3].

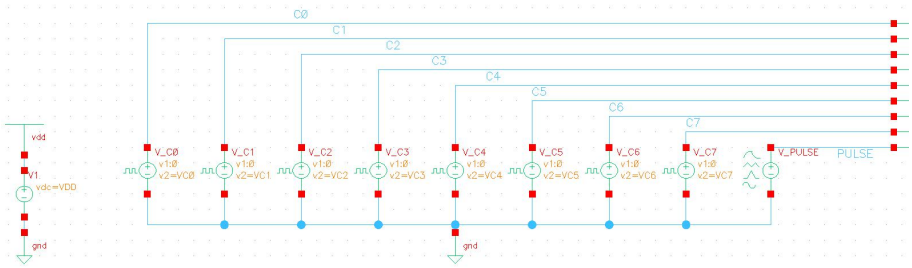


Figure 3.16: Inputs of the test bench

In Figure 3.18, we have the schematic of the instance under evaluation. If we count, we can see eight combinations of two inverters and their corresponding multiplexer, hence the eight stages. At the lower left corner, we see a green-lined rectangle representing the bias circuit from Section 3.1.1 connected to every pair of inverters along the path. The vertically paired green-lined rectangles represent the two inverters that feed the signal to the corresponding inverter. On top of each inverter, it is possible to see the input for its corresponding challenge bit. *PUF_8Stages_WITHOUT_ARBITER*, as the name implies, ends with two simple outputs and not an arbiter.

[3]The test bench was divided into two separate pictures for better resolution

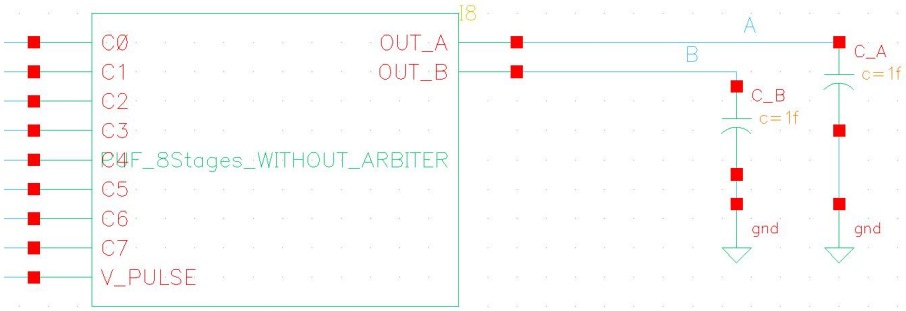


Figure 3.17: Outputs of the test bench

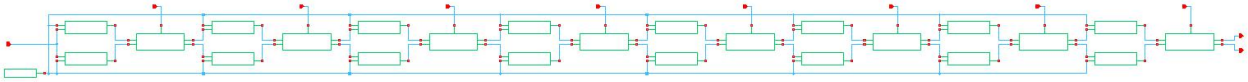


Figure 3.18: Construction of the *PUF_8Stages_WITHOUT_ARBITER*

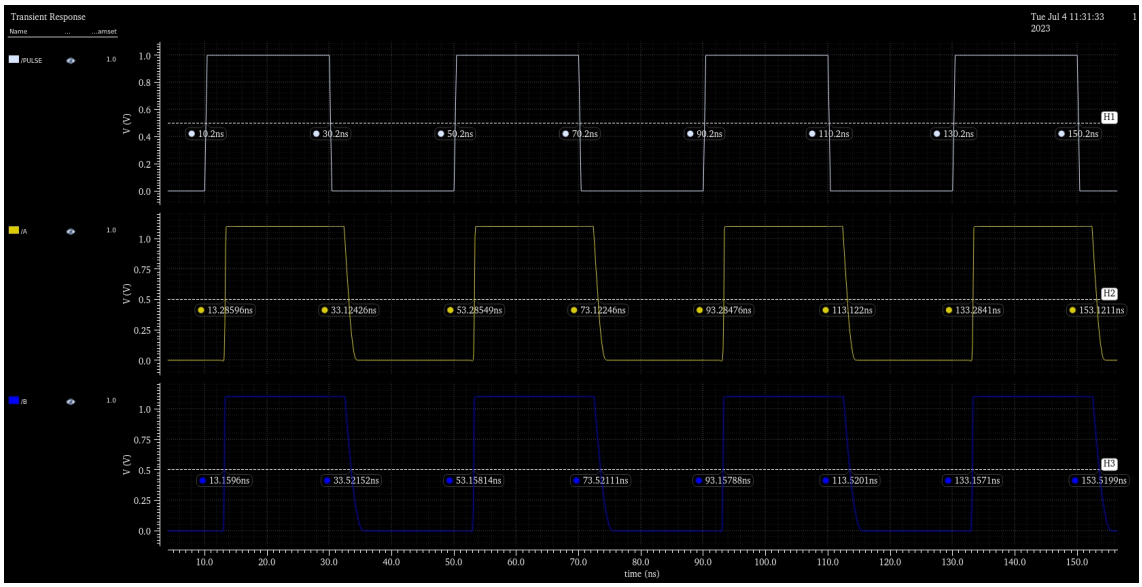


Figure 3.19: Example of a measurement in a Monte Carlo simulation

With the test bench concluded, we performed several Monte Carlo simulations whose specifications can be seen again in Figure 3.15. Their differentiation came either from the frequency of the *V_PULSE* utilized or which challenge bits were activated. Out of the two hundred points simulated, we collected data on points[4] number one, fifty, one hundred, one hundred and fifty, and two hundred. The data collected consisted of time values, at which the waveform crossed the 0.5 V line, measured at seven measuring points in each signal line in each *mcparamset*. Figure 3.19 presents an example of the data collected. We can see, for instance, that the first two measuring points of output *B* give us 13.1596 ns and 33.521 52 ns, respectively.

Before continuing, it is essential to note that Monte Carlo simulations are also utilized to understand a given circuit's yield after fabrication. This approach is unsuitable for our specific use case because we want a PUF to have a wide enough distribution of delays, enhancing the uniqueness of its instances. Narrower distributions (with higher yield) suggest that the performance characteristics of the produced ICs are consistent and predictable, resulting in less uniqueness. Given this fact, in ADE Assembler, the best way we found to understand what happened after each simulation was to go from *mcparamset* to *mcparamset* graphically checking their outputs. This process becomes inefficient, so we reduced the number of *mcparamsets* analyzed, selecting only five of the two hundred available. A helpful suggestion for Section 4.2 is to find a way to automatize this process.

Eight simulations were conducted, four at 25 MHz, the PUF's maximum operating frequency [56], and four at half of this frequency, 12.5 MHz, to try to understand the impact of frequency in its functionality. For both frequencies, the simulations comprised of having every challenge bit at logical zero, varying one challenge bit (V_C0 in Figure 3.16), running two challenge bits (V_C0 and V_C4 in Figure 3.16), and having them all at logical one. After each simulation, at the standard seven measuring points in each *mcparamset*, we calculated Δ, the difference between a measuring point in *B* and its value in *A*. Using Figure 3.19 as an example, this meant $13.1596 - 13.28596 = -0.12636$, $33.52152 - 33.12426 = 0.39726$, and so on. From this brief example, it is already possible to conclude that $\Delta < 0 \implies B$ "arrived" first then *A* while $\Delta > 0 \implies A$ "arrived" first then *B*. Having calculated each Δ, we performed their per-point simulation average, meaning we averaged them among the corresponding measurement points on the other four *mcparamsets*. These averages of Δ became the basis of our analysis. Figure 3.20 presents the plot of the resulting Δs from the four simulations performed at 12.5 MHz while Figure 3.21 presents them for 25 MHz.

Each colored line represents one of the simulations conducted, and the dots show the value of Δ (in ns) at one of the measuring points, marked on the graph's horizontal axis. Analyzing both graphs, it is possible to see that, at least with only eight stages, a frequency as long as it is within the PUF's operating range does not interfere with its response. It also becomes clear how the challenges can change this same response and how we can identify a given simulation based on its delays. Noting the similar design of both graphs, we can also conclude that, despite process

[4] Denominated *mcparamset* in the ADE Assembler environment

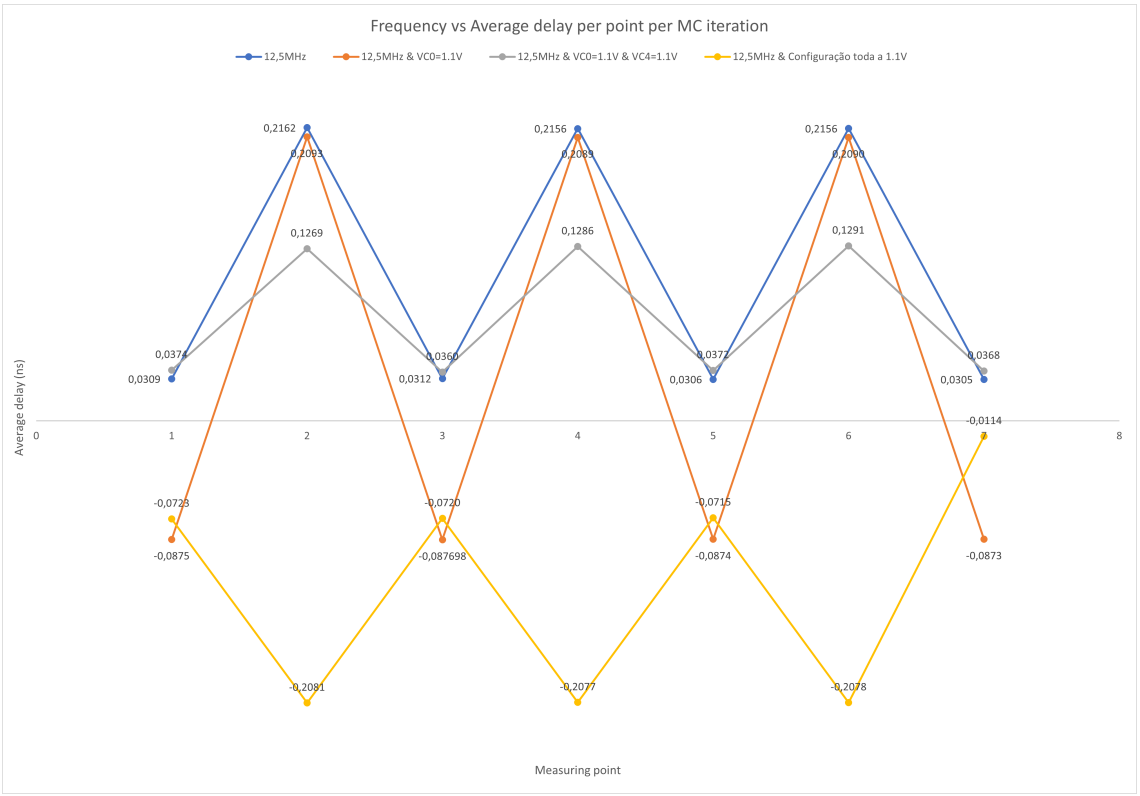


Figure 3.20: Frequency *versus* Average delay (Δ) at 12.5 MHz

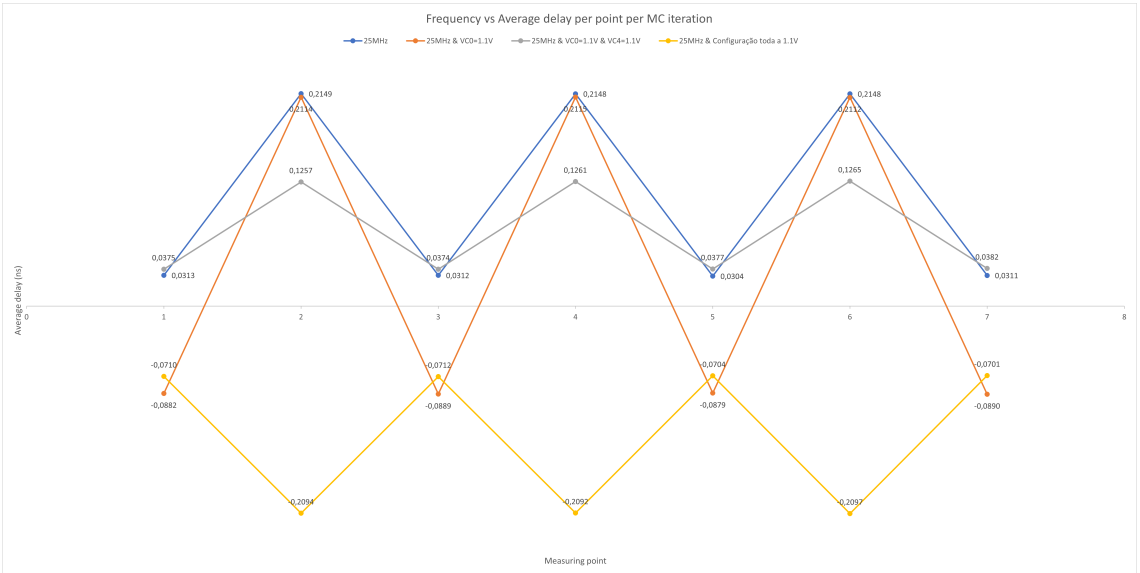


Figure 3.21: Frequency *versus* Average delay (Δ) at 25 MHz

variability not being a linear phenomenon, the addition of the delays along the PUF's delay chain can be modeled as such, as noted in Section 2.10.

Although our sample is reduced, the distribution of the delays was interesting to analyze for this purpose. With the help of the *Python* script in Appendix A, our data was fitted to a normal distribution, resulting in Figure 3.22 for 12.5 MHz and Figure 3.23 for 25 MHz. In both of them, we can see how each simulation has a very distinct curve, a good indicator of a PUF's ability to be uniquely identifiable. Albeit, it becomes clear, through the resemblance in these distributions, how more than eight stages are needed to ensure a greater uniqueness for each PUF instance.

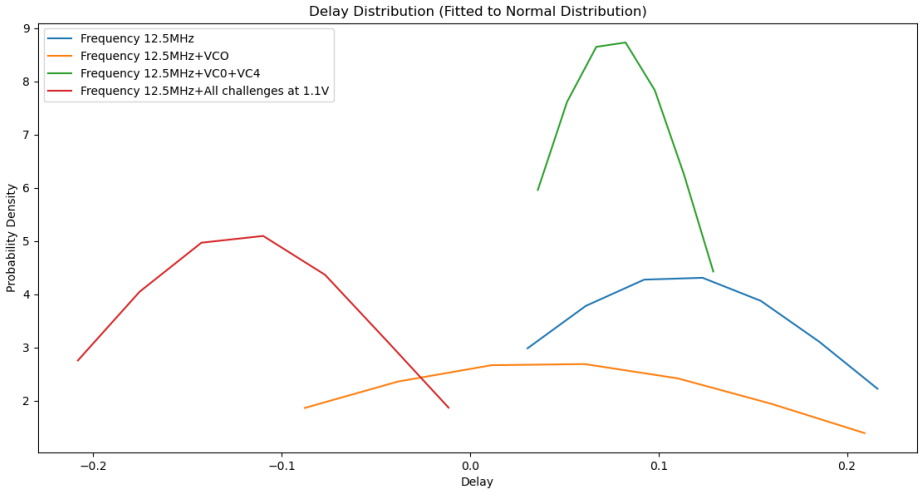


Figure 3.22: Delay distribution at 12.5 MHz

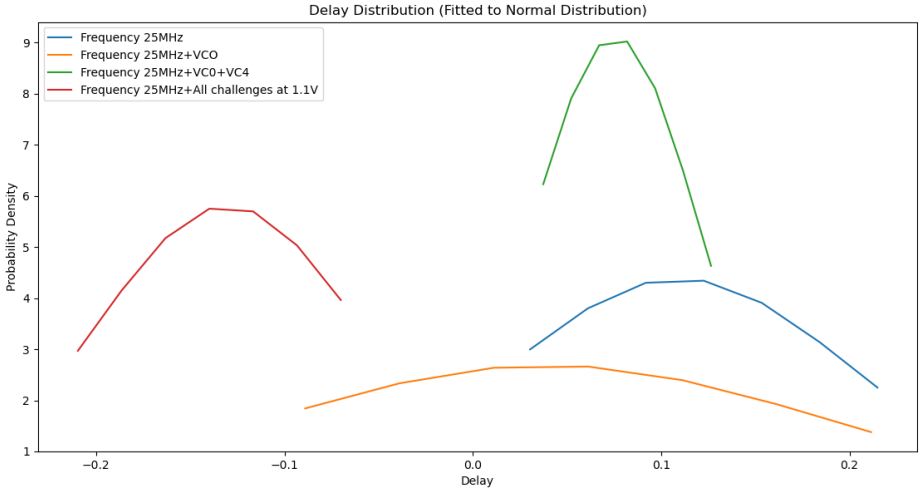


Figure 3.23: Delay distribution at 25 MHz

3.2.2 An 8 stages PUF with arbiter

The main difference between this and the previous test bench is that now, at the end of the PUF's delay chain, there will be an arbiter implemented as in Figure 3.10. The presence of an arbiter meant that at the output of this test bench, the expected response would be a bit accordingly to the signal that arrives first.

Maintaining the logic of the previous test bench, we conducted another four Monte Carlo simulations with the same specifications. This time, and noting the previously obtained frequency results, we only tested the PUF at 25 MHz. Again, the tests were conducted, having every challenge bit at logical zero, varying one challenge bit (V_C0) to a logical one, running two challenge bits (V_C0 and V_C4) at logical one, and having them all at logical one. Their results can be seen in Tables 3.5, 3.6, 3.7, and 3.8, respectively. The *mcparamsets* analyzed were also the same. Each table consists of three columns, *mcparamsets*, *MUX7_OUT*, and *ARBITER*. The first tells which of the *mcparamsets* was evaluated. The second presents the time value at which each signal waveform crossed the 0.5 V line exiting the last MUX. Hence, the third column should note which signal first arrived at the arbiter by presenting a logical one at its exit and a logic zero at the exit of the "losing" signal.

The combination of all the "wins" and "losses" resulting from each test was then compiled into Table 3.9. It presents the four conducted tests, all challenge bits at logical zero ("Zero" column), V_C0 at logical one ("V_C0" column), V_C0 plus V_C4 at logical one ("V_C0+V_C4" column), all challenge bits at logical one ("One" column), and the "winning" signal at each *mcparamset*. A "win" is portrayed as a "1" in the column belonging to the signal that arrived first at the arbiter. A loss is represented by an "-". It is important to note that these results represent an Arbiter PUF with only eight stages and without the LFSR. However, with this sample of *mcparamsets*, it is clear that the PUF has a bias towards B given that out of five *mcparamsets*, three, 1, 50, and 200 have B "winning" more times than A. This means that for this configuration, the PUF is not as random as it should be. This conclusion seems even more plausible if we note that for each test, the more challenge bits are commuted to a logical one, the more B tends to be the winner.

mcparamset	MUX7_OUT		ARBITER	
	A	B	A	B
1	13.024	12.913	0	1
50	13.136	12.874	0	1
100	12.890	13.274	1	0
150	13.085	13.235	1	0
200	13.278	13.242	0	1

Table 3.5: All challenge bits at logical zero @ 25 MHz

mcparamset	MUX7_OUT		ARBITER	
	A	B	A	B
1	12.841	13.116	1	0
50	13.140	12.860	0	1
100	13.149	12.991	0	1
150	13.107	13.217	1	0
200	13.443	13.074	0	1

Table 3.6: V_C0 at logical one @ 25 MHz

mcparamset	MUX7_OUT		ARBITER	
	A	B	A	B
1	13.130	12.821	0	1
50	12.969	13.017	1	0
100	12.831	13.326	1	0
150	13.049	13.275	1	0
200	13.391	13.099	0	1

Table 3.7: V_C0 + V_C4 at logical one @ 25 MHz

mcparamset	MUX7_OUT		ARBITER	
	A	B	A	B
1	13.066	12.873	0	1
50	13.171	12.817	0	1
100	12.946	13.181	1	0
150	13.169	13.167	0	1
200	13.290	13.235	0	1

Table 3.8: All challenge bits at logical one @ 25 MHz

mcparamset	Zero		V_C0		V_C0 + V_C4		One	
	A	B	A	B	A	B	A	B
1	-	1	1	-	-	1	-	1
50	-	1	-	1	1	-	-	1
100	1	-	-	1	1	-	1	-
150	1	-	1	-	1	-	-	1
200	-	1	-	1	-	1	-	1

Table 3.9: Distribution of *A* and *B* @ 25 MHz

3.2.3 A 64 stages PUF without arbiter

For this test bench, we scaled the PUF to its total number of stages minus the arbiter to try and follow the thought process behind Section 3.2.1. However, we concluded that the PUF could not deal with a pulse of 25 MHz as presented in [56]. We noticed how the signal, right at the exit of the first INV, was already highly distorted. This would only worsen as the signal propagated down the delay chain. Most likely, due to the parasitic capacities, the time required to discharge increased, meaning the operation frequency would have to be reduced. Augmenting the transistor sizes might be a solution. However, doing so would also increase the capacities. Through experimental measurement, we settled at a frequency of 2.5 MHz, a severe decrease.

Due to timing constraints, and after realizing the PUF's incapability to perform at 25 MHz, we decided to advance for Section 3.2.4 to evaluate the PUF as a whole.

3.2.4 A 64 stages PUF with arbiter

Performing the same tests as in the previously discussed test benches, we can resume the results in a table similar to Table 3.9. Hence, we present Table 3.10.

Analyzing it, we can infer that the PUF still maintains a bias to one of the paths, although the tendency seems to be more distributed than in Table 3.9 as this time three *mcparamsets* 1, 150, and 200 have A "winning" more times than B. However, *mcparamsets* 50 and 100 have B "winning" more than A. Even with 64 stages, the PUF is still not random enough. This time, commuting a challenge bit to a logical one does not impact the response as shown by Table 3.9. Nevertheless, we should be aware and note that the proportion of commuting two bits in eight is different from commuting two in sixty-four.

mcparamset	Zero		V_C0		V_C0 + V_C4		One	
	A	B	A	B	A	B	A	B
1	1	-	1	-	1	-	1	-
50	-	1	-	1	-	1	-	1
100	-	1	-	1	-	1	-	1
150	1	-	1	-	-	1	1	-
200	1	-	1	-	1	-	1	-

Table 3.10: Distribution of *A* and *B* @ 2.5 MHz

Chapter 4

Conclusions and Future Work

This final chapter will review the work developed within this dissertation, its contributions, and further improvements to be developed.

4.1 Summary

Our study started with a hypothesis. Could PUFs, within solid-state electronics, become the future "go-to" method for hardware security? With their emergence as a compelling avenue for advancing hardware security, our exploration started with somewhat high expectations. However, one should note that the PUF universe is relatively large, so we focused our research on a specific implementation of an Arbiter PUF. And in this implementation, its performance fell below initial expectations. With our technology, it proved incapable of operating at the expected frequency and was also highly biased. However, albeit not with the total number of stages as the proposed Arbiter PUF [56], we showed, as expected, that it could be uniquely identifiable through its response against a challenge.

Although our results did not meet expectations, a quick search on PUFs yields numerous potential architectures and industry use cases [60]. Therefore, the inherent potential of PUFs to enhance current security paradigms remains an exciting area of research and an interesting path in developing hardware security.

The results are open to being challenged, reevaluated, or even replicated by interested parties.

4.2 Future Work

Given the schedules that bound this dissertation, we leave a few topics as a suggestion for future work:

- **Automatization:** Conducting Monte Carlo Simulations is by itself a time-consuming process, even more so if we have to manually go from *mcparamset* to *mcparamset* visually analyzing each graph. A tool capable of analyzing graph parameters without the need to search graph by graph would be a great addition to the continuation of this work.

- **Frequency, randomness, and challenge bits:** It would be essential to understand why the PUF could not maintain its operating frequency when scaled from eight to sixty-four stages. This could help determine if the problem was in the dimensioning of the transistors or if this technology can not be used to implement this specific Arbiter PUF. Afterward, it would be essential to test the sixty-four stages PUF with a proportional quantity of commuted challenge bits to understand its behavior, especially if its randomness improved.

- **Bitstream:** At the beginning of this work and looking forward to the PUF's testing, a *Python* script capable of generating PWL files was created. Utilizing this script to create several files containing different binary pulses would be interesting to analyze and characterize this PUF's architecture, as it was only tested with a periodic pulse and not a proper bitstream. The script can be found in Appendix B.

- **Temperature and Voltage Characterization:** Given its described characteristics, a comprehensive evaluation of its performance under varying temperature and supply voltage conditions is needed. The obtained results should then be compared with those in [56]. These tests should measure its response reliability and investigate potential vulnerabilities derived from these variations. Doing so could also give an exciting overview of how manufacturing technology influences a PUF's behavior.

- **Layout and Post-Layout testing:** Having tested and characterized its schematic, the next step should be to design the layout of the proposed PUF. Post-layout, one should evaluate its ability to consistently produce the same responses as the schematic while subjected to the same temperature and supply voltage variations. It should also be interesting to study if layout variations originate PUFs with different responses.

- **Comparative Analysis with other PUFs:** A comparative analysis with different PUF architectures would be both insightful and desirable. A complete survey for the same manufacturing technology comparing the performance of diverse PUF architectures while highlighting their strengths and weaknesses would contribute to the broader advancement of PUF-based security solutions.

- **Hardware Implementation:** Finally, being able to fabricate a dedicated ASIC would provide real-world data, allowing us to conclude the practicality of this specific PUF architecture.

Appendix A

DataToNormal.py

```
import matplotlib.pyplot as plt
import numpy as np
from scipy.stats import norm

frequencies = ["12.5MHz", "12.5MHz+VCO", "12.5MHz+VC0+VC4", "12.5MHz+All challenges
     at 1.1V"]

# Each column of the "delays" matrix corresponds to a simulation in
# the "frequencies" array and presents the delay measured at the n-th measuring
# point for said simulation

delays = [[0.0309, -0.08746, 0.037406, -0.072338],
          [0.2162, 0.209346, 0.126912, -0.208112],
          [0.0312, -0.087698, 0.035958, -0.071988],
          [0.2156, 0.20894, 0.12856, -0.20772],
          [0.0306, -0.08736, 0.03724, -0.07148],
          [0.2156, 0.20898, 0.12906, -0.20782],
          [0.0305, -0.0873, 0.03678, -0.0114]
          ]

# Uncomment the following "frequencies" and "delays" and comment the previous ones
# to plot the delays at 25MHz

# frequencies = ["25MHz", "25MHz+VCO", "25MHz+VC0+VC4", "25MHz+All challenges at
    1.1V"]

# delays = [[0.0313, -0.0882, 0.0375, -0.0710],
#           [0.2149, 0.2114, 0.1257, -0.2094],
#           [0.0312, -0.0889, 0.0374, -0.0712],
#           [0.2148, 0.2115, 0.1261, -0.2092],
#           [0.0304, -0.0879, 0.0377, -0.0704],
#           [0.2148, 0.2112, 0.1265, -0.2097],
#           [0.0311, -0.0890, 0.0382, -0.0701]
#           ]

```

```
delays = np.transpose(delays)

plt.figure(figsize=(10, 4))

for i, delay_values in enumerate(delays):
    mu, std = norm.fit(delay_values)  # Fit my data to a normal distribution

    mean_val = mu
    std_val = std
    print(f"Frequency {frequencies[i]} - Mean: {mean_val:.4f}, Std Dev: {std_val:.4
    f}")

    x = np.linspace(min(delay_values), max(delay_values), len(delay_values))
    plt.plot(x, norm.pdf(x, mu, std), label=f'Frequency {frequencies[i]}')

plt.xlabel('Delay')
plt.ylabel('Probability Density')
plt.title('Delay Distribution (Fitted to Normal Distribution)')
plt.legend()
plt.show()
```

Appendix B

PWL_Creator.py

```
from array import *
from logging import ERROR
import numpy as np

# --- Initialization and user input
print()
print("Welcome to PWL file generator ! \n")

filename = input("Give a name for your file (terminated with .pwl): ")

n = int(input("How many bits the input has (n): "))

serial_input = list(input("Introduce your input: "))

if len(serial_input) != n:
    raise SyntaxError("n and input don't have the same length! \n")

time_unit = input("Define the prefix of the time unit on your PWL file: ")

bit_time = float(input("Define bit time: "))

delay = float(input("Define the input's delay (0 if none): "))
# ---

# --- Generation of the time values
pwl_times = array('f',[])

stop = float((bit_time*n) + (0.1*n) + delay)
# A bit has to last "bit_time", so for "n" bits => "bit_time * n"
# As the decimal values go from [0.0, 0.9], the next bit in the array moves 0.1
# So (0,1*n) is added to ensure "bit_time" is equal for all

for i in np.arange(0, stop, 0.1):
    i = round(i, 1)
    pwl_times.append(i)
```

```
# ---

# --- Open and write the PWL file
with open(filename, 'w') as pwlFile:

    final_time = 0
    count = 0
    x = 0

    for i in range(0, len(pwl_times)):

        if delay == 0:
            if count < (10*bit_time):
                print(f"{round(pwl_times[i],1)}{time_unit}{' '}{serial_input[x]}", file=pwlFile)
                count = count + 1
            elif count == (10*bit_time):
                print(f"{round(pwl_times[i],1)}{time_unit}{' '}{serial_input[x]}", file=pwlFile)
                count = 0
                x = x + 1
                final_time = round(pwl_times[i],1)
        elif delay != 0:
            print(f"{round(pwl_times[i],1)}{time_unit}{' '}{0}", file=pwlFile)
            delay = round((delay - 0.1), 1)

    print(f"{final_time+0.1}{time_unit}{' '}{0}", file=pwlFile) #Prints the final line as zero
# ---
```

References

[1] Basel Halak. *Physically Unclonable Functions: From Basic Design Principles to Advanced Hardware Security Applications*. Springer Cham, 2018. URL: https://doi.org/10.1007/978-3-319-76804-5, doi:10.1007/978-3-319-76804-5.

[2] Mark Tehranipoor, Nitin Pundir, Nidish Vashistha, and Farimah Farahmandi. *Hardware Security Primitives*. Springer Cham, 2022. URL: https://doi.org/10.1007/978-3-031-19185-5, doi:10.1007/978-3-031-19185-5.

[3] Roel Maes. *Physically Unclonable Functions: Constructions, Properties and Applications*. Springer Berlin Heidelberg, 2013. URL: https://doi.org/10.1007/978-3-642-41395-7_2, doi:10.1007/978-3-642-41395-7_2.

[4] Arjen K. Lenstra, James P. Hughes, Maxime Augier, Joppe W. Bos, Thorsten Kleinjung, and Christophe Wachter. Ron was wrong, whit is right. Cryptology ePrint Archive, Paper 2012/064, 2012. https://eprint.iacr.org/2012/064. URL: https://eprint.iacr.org/2012/064.

[5] Randy Torrance and Dick James. *The State-of-the-Art in IC Reverse Engineering*, page 363–381. Springer, Berlin, Heidelberg, 2009. doi:10.1007/978-3-642-04138-9_26.

[6] Charles Herder, Meng-Day Yu, Farinaz Koushanfar, and Srinivas Devadas. Physical unclonable functions and applications: A tutorial. *Proceedings of the IEEE*, 102(8):1126–1141, 2014. doi:10.1109/JPROC.2014.2320516.

[7] Paul Kocher, Joshua Jaffe, and Benjamin Jun. *Differential Power Analysis*, page 388–397. Springer, Berlin, Heidelberg, 1999. doi:10.1007/3-540-48405-1_25.

[8] Mohammad Tehranipoor and Farinaz Koushanfar. A survey of hardware trojan taxonomy and detection. *IEEE Design and Test of Computers*, 27(1):10–25, 2010. doi:10.1109/mdt.2010.7.

[9] Subhasish Mitra, H.-S. Philip Wong, and Simon Wong. Stopping hardware trojans in their tracks, Jan 2015. Accessed: July 5, 2023. URL: https://spectrum.ieee.org/stopping-hardware-trojans-in-their-tracks.

[10] Masoud Rostami, Farinaz Koushanfar, and Ramesh Karri. A primer on hardware security: Models, methods, and metrics. *Proceedings of the IEEE*, 102(8):1283–1295, 2014. doi:10.1109/jproc.2014.2335155.

[11] Ujjwal Guin, Domenic Forte, and Mohammad Tehranipoor. Anti-Counterfeit Techniques: From Design to Resign. In *14th International Workshop on Microprocessor Test and Verification*, pages 89–94, 2013. doi:10.1109/MTV.2013.28.

[12] Basel Halak, Julian Murphy, and Alex Yakovlev. Power balanced circuits for leakage-power-attacks resilient design. 2015. doi:10.1109/sai.2015.7237294.

[13] Sergei P. Skorobogatov and Ross J. Anderson. *Optical Fault Induction Attacks*, page 2–12. Springer, Berlin, Heidelberg, 2003. doi:10.1007/3-540-36400-5_2.

[14] Werner Schindler and Wolfgang Killmann. *Evaluation Criteria for True (Physical) Random Number Generators Used in Cryptographic Applications*, page 431–449. Springer, Berlin, Heidelberg, 2003. doi:10.1007/3-540-36400-5_31.

[15] Andrew J. Leiserson, Mark E. Marson, and Megan A. Wachs. *Gate-Level Masking under a Path-Based Leakage Metric*, page 580–597. Springer, Berlin, Heidelberg, 2014. doi:10.1007/978-3-662-44709-3_32.

[16] Blaise Gassend, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas. Silicon Physical Random Functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*, CCS '02, page 148–160, New York, NY, USA, 2002. Association for Computing Machinery. doi:10.1145/586110.586132.

[17] Blaise Gassend, Daihyun Lim, Dwaine Clarke, Marten Van Dijk, and Srinivas Devadas. Identification and authentication of integrated circuits. *Concurrency and Computation: Practice and Experience*, 16(11):1077–1098, 2004. doi:10.1002/cpe.805.

[18] Thomas Mcgrath, Ibrahim E. Bagci, Zhiming M. Wang, Utz Roedig, and Robert J. Young. A PUF taxonomy. *Applied Physics Reviews*, 6(1):011303, 2019. URL: https://doi.org/10.1063/1.5079407, doi:10.1063/1.5079407.

[19] J.W. Lee, Daihyun Lim, B. Gassend, G.E. Suh, M. van Dijk, and S. Devadas. A Technique to Build a Secret Key in Integrated Circuits for Identification and Authentication Applications. In *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525)*, pages 176–179, 2004. doi:10.1109/VLSIC.2004.1346548.

[20] Jeffrey Hojlo. Future of Industry Ecosystems: Shared Data and Insights, Jan 2021. Accessed: July 7, 2023. URL: https://blogs.idc.com/2021/01/06/future-of-industry-ecosystems-shared-data-and-insights/.

[21] Simranjeet Sidhu, Bassam J. Mohd, and Thaier Hayajneh. Hardware Security in IoT Devices with Emphasis on Hardware Trojans. *Journal of Sensor and Actuator Networks*, 8(3):42, 2019. doi:10.3390/jsan8030042.

[22] Wade Trappe, Richard Howard, and Robert S. Moore. Low-Energy Security: Limits and Opportunities in the Internet of Things. *IEEE Security & Privacy Magazine*, 13(1):14–21, 2015. doi:10.1109/msp.2015.7.

[23] Bank Millennium. New dimension of authentication. Accessed: July 10, 2023. URL: https://www.bankmillennium.pl/en/corporate/e-banking/internet-banking/security/hardware-token-with-scanner.

[24] Thales Group. Banking tokens - OTP authentication and signature devices. Accessed: July 10, 2023. URL: https://www.thalesgroup.com/en/markets/digital-identity-and-security/banking-payment/digital-banking/tokens.

[25] M Tanjidur Rahman, Qihang Shi, Shahin Tajik, Haoting Shen, Damon L. Woodard, Mark Tehranipoor, and Navid Asadizanjani. Physical Inspection & Attacks: New Frontier in Hardware Security. *IEEE 3rd International Verification and Security Workshop (IVSW)*, 2018. doi:10.1109/ivsw.2018.8494856.

[26] Kerry Bernstein, Ching-Te Chuang, Rajiv Joshi, and Ruchir Puri. Design and CAD Challenges in Sub-90nm CMOS Technologies. In *Proceedings of the 2003 IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '03, page 129. IEEE Computer Society, 2003.

[27] David J. Frank, Ruchir Puri, and Dorel Toma. Design and CAD Challenges in 45nm CMOS and beyond. In *Proceedings of the 2006 IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '06, page 329–333. Association for Computing Machinery, 2006. URL: https://doi.org/10.1145/1233501.1233567, doi:10.1145/1233501.1233567.

[28] Mohamed H. Abu-Rahma and Mohab Anis. Variability in VLSI Circuits: Sources and Design Considerations. In *2007 IEEE International Symposium on Circuits and Systems*, pages 3215–3218, 2007. doi:10.1109/iscas.2007.378156.

[29] K. Lofstrom, W.R. Daasch, and D. Taylor. IC identification circuit using device mismatch. In *2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No.00CH37056)*, pages 372–373, 2000. doi:10.1109/ISSCC.2000.839821.

[30] Daihyun Lim, J.W. Lee, B. Gassend, G.E. Suh, M. van Dijk, and S. Devadas. Extracting secret keys from integrated circuits. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 13(10):1200–1205, 2005. doi:10.1109/TVLSI.2005.859470.

[31] Meng-Day Yu, Richard Sowell, Alok Singh, David M'Raïhi, and Srinivas Devadas. Performance metrics and empirical results of a PUF cryptographic key generation ASIC. In *2012 IEEE International Symposium on Hardware-Oriented Security and Trust*, pages 108–115, 2012. doi:10.1109/HST.2012.6224329.

[32] Yousra Alkabani, Farinaz Koushanfar, and Miodrag Potkonjak. Remote activation of ICs for piracy prevention and digital right management. In *2007 IEEE/ACM International Conference on Computer-Aided Design*, pages 674–677, 2007. doi:10.1109/ICCAD.2007.4397343.

[33] Ulrich Rührmair. Oblivious Transfer Based on Physical Unclonable Functions. In *3rd International Conference on Trust and Trustworthy Computing*, page 430–440, 2010. doi:10.1007/978-3-642-13869-0_31.

[34] Yansong Gao, Hua Ma, Derek Abbott, and Said F. Al-Sarawi. PUF Sensor: Exploiting PUF Unreliability for Secure Wireless Sensing. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 64(9):2532–2543, 2017. doi:10.1109/TCSI.2017.2695228.

[35] Rolf Landauer. Information is Physical. *Physics Today*, 44(5):23–29, 1991. doi:10.1063/1.881299.

[36] Ulrich Rührmair, Srinivas Devadas, and Farinaz Koushanfar. *Security Based on Physical Unclonability and Disorder*, chapter 4, page 65–102. In Tehranipoor and Wang [61], 2012.

[37] Fahem Zerrouki, Samir Ouchani, and Hafida Bouarfa. A survey on silicon PUFs. *Journal of Systems Architecture*, 2022. doi:10.1016/j.sysarc.2022.102514.

[38] Nitya Raut. What is Hamming Distance?, Jun 2020. Accessed: July 17, 2023. URL: https://www.tutorialspoint.com/what-is-hamming-distance.

[39] Yier Jin. Introduction to Hardware Security. *Electronics*, 4(4):763–784, 2015. doi:https://dx.doi.org/10.3390/electronics4040763.

[40] Ramesh Karri, Jeyavijayan Rajendran, and Kurt Rosenfeld. *Trojan Taxonomy*, chapter 14, page 325–338. In Tehranipoor and Wang [61], 2012.

[41] Wei Hu, Chip-Hong Chang, Anirban Sengupta, Swarup Bhunia, Ryan Kastner, and Hai Li. An Overview of Hardware Security and Trust: Threats, Countermeasures, and Design Tools. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(6):1010–1038, 2021. doi:10.1109/tcad.2020.3047976.

[42] Brett Daniel. Counterfeit Electronic Parts: A Multibillion-Dollar Black Market, Jul 2020. Accessed: July 11, 2023. URL: https://www.trentonsystems.com/blog/counterfeit-electronic-parts.

[43] Mark Tehranipoor, Nitin Pundir, Nidish Vashistha, and Farimah Farahmandi. Counterfeit and recycled ic detection. In *Hardware Security Primitives*, page 281–300. Springer, Cham, 2022. doi:10.1007/978-3-031-19185-5_16.

[44] Farinaz Koushanfar, Saverio Fazzari, Carl Mccants, William Bryson, Matthew Sale, Peilin Song, and Miodrag Potkonjak. Can EDA combat the rise of electronic counterfeiting? In *Proceedings of the 49th Annual Design Automation Conference*, DAC '12, page 133–138. Association for Computing Machinery, 2012. doi:10.1145/2228360.2228386.

[45] Shahin Tajik, Heiko Lohrke, Fatemeh Ganji, Jean-Pierre Seifert, and Christian Boit. Laser Fault Attack on Physically Unclonable Functions. In *2015 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC)*, pages 85–96, 2015. doi:10.1109/fdtc.2015.19.

[46] Farhana Sheikh and Leonel Sousa. *Circuits and Systems for Security and Privacy*, chapter 6. CRC Press, Boca Raton, Florida, 2016. URL: https://doi.org/10.1201/b19499.

[47] Roel Maes and Ingrid Verbauwhede. *Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions*, page 3–37. Springer Berlin, Heidelberg, 2010. doi:10.1007/978-3-642-14452-3_1.

[48] Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. *A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions*, page 245–267. Springer New York, NY, 2012. doi:10.1007/978-1-4614-1362-2_11.

[49] Hamming Weight. Accessed: July 17, 2023. URL: https://pt.farnell.com/en-PT/hamming-weight-definition.

[50] Ruhrmair et al. PUF Modeling Attacks on Simulated and Silicon Data. *IEEE Transactions on Information Forensics and Security*, 8(11):1876–1891, 2013. doi:10.1109/tifs.2013.2279798.

[51] Lang Lin, Dan Holcomb, Dilip Kumar Krishnappa, Prasad Shabadi, and Wayne Burleson. Low-Power Sub-Threshold Design of Secure Physical Unclonable Functions. In *Proceedings of the 16th ACM/IEEE International Symposium on Low Power Electronics and Design*, ISLPED '10, pages 43–48, New York, NY, USA, 2010. Association for Computing Machinery. doi:10.1145/1840845.1840855.

[52] Gabriel Hospodar, Roel Maes, and Ingrid Verbauwhede. *Machine learning attacks on 65nm Arbiter PUFs: Accurate modeling poses strict bounds on usability*. 2012. doi:10.1109/wifs.2012.6412622.

[53] Ahmed Mahmoud, Ulrich Rührmair, Mehrdad Majzoobi, and Farinaz Koushanfar. Combined Modeling and Side Channel Attacks on Strong PUFs. 2013. URL: https://eprint.iacr.org/2013/632.

[54] Blaise Gassend. *Physical Random functions*. Master's thesis, MIT, MA, USA, 2003.

[55] Daihyun Lim. *Extracting Secret Keys from Integrated Circuits*. Master's thesis, MIT, MA, USA, 2004.

[56] Yuan Cao, Wenhan Zheng, Xiaojin Zhao, and Chip-Hong Chang. An Energy-Efficient Current-Starved Inverter Based Strong Physical Unclonable Function With Enhanced Temperature Stability. *IEEE Access*, 7:105287–105297, 2019. doi:10.1109/ACCESS.2019.2932022.

[57] Raghavan Kumar, Vinay C. Patil, and Sandip Kundu. Design of Unique and Reliable Physically Unclonable Functions Based on Current Starved Inverter Chain. In *2011 IEEE Computer Society Annual Symposium on VLSI*, pages 224–229, 2011. doi:10.1109/ISVLSI.2011.82.

[58] Transmission Gate. Accessed: July 19, 2023. URL: https://www.electronics-tutorials.ws/combination/transmission-gate.html.

[59] SR NAND latch. Accessed: July 19, 2023. URL: https://electronics-course.com/sr-nand-latch.

[60] Ed Peterson. Developing Tamper-Resistant Designs with Zynq UltraScale+ Devices, Aug 2018. Accessed: July 22, 2023. URL: https://docs.xilinx.com/v/u/en-US/xapp1323-zynq-usp-tamper-resistant-designs.

[61] Mohammad Tehranipoor and Cliff Wang, editors. *Introduction to Hardware Security and Trust*. Springer New York, NY, 2012.