Trabajo de Fin de Máster
"Máster Universitario en Microelectónica: Diseño y Aplicaciones de Sistemas Micro/Nanométricos"

Design of aging-resilient SRAM-based Physical Unclonable Functions (PUFs)

Author:	Andrés Santana Andreo
Advisors:	P. Brox, E. Roca and F. V. Fernández
Date:	February 19, 2025

Acknowledgements

First of all, I would like to thank those who contributed to the success of this work. I am very grateful for the dedication, guidance and expertise offered by my advisors Dr. Piedad Brox Jiménez, Dr. Elisenda Roca Moreno and Dr. Francisco V. Fernández Fernández. I have been able to learn a lot thanks to them.

Furthermore, I would also like to express my appreciation for my colleagues and friends Eros Camacho Ruiz, Pablo Sarazá Canflanca and Héctor Carrasco Lopez, who provided very helpful advice whenever I needed, as well as sharing their insight in this topic and providing a great enviroment to work in.

Moreover, I would like to thank the University of Seville and its teachers for the opportunity to study in a field that I am truly passionate about and that I hope to keep working on.

Last but not least I want to thank my mother, my sister, my girlfriend and my friends for supporting me during my studies and through this complicated year.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Every year, connectivity and digitization increases in almost every aspect of our lives. This growing tendency has transformed many fields, such as communications, healthcare, finance or transportation, providing new solutions and a massive increase in efficiency and productivity. The deployment of IoT networks, where multiple devices– from simple sensors to smartphones and wearables – are connected together for the purpose of exchanging data over the network, have been accelerated during the last years. However, there is a lack of security in digital interconnectivity, thus making the exploitation of sensitive data through device impersonation very profitable for potential attackers.

A way to protect a digital device is the integration of a hardware Root-of-Trust (RoT). Conventional implementations of hardware RoT employ Non-Volatile Memories (NVMs) to store secret keys. This is an expensive solution due to their high design area and power consumption. Moreover, NVMs are vulnerable to physical attacks being necessary to add active circuitry that detects tampering, increasing the cost even more.

Furthermore, several IoT scenarios often operate on resource-constrained devices, which makes the use of NVMs non-affordable. Physical Unclonable Functions (PUFs) have emerged as an alternative to solve this problem, thus being one of the dominant topics in the hardware security domain. A PUF is a physical implementation of a function that maps an input (challenge) to an output (response). Specifically, silicon PUFs are based on exploiting small variations present during semiconductor manufacturing, so that the PUF creates a unique challenge-response mapping and thus it cannot be cloned. PUFs can be used as the foundation of a large set of security related tasks such as authentication and generation of cryptographic keys. These features make PUFs key elements to build RoT. PUFs usually imply simple circuitry and they generally do not need anti-tamper protection, since physical attacks on PUFs are very difficult to perform without modifying the characteristics from which the RoT is derived.

There are multiple implementations for silicon PUFs, but SRAM-based PUFs are one the most popular ones due to their low-cost of implementation [1, 2]. For this reason, this type of PUFs will be addressed in this work. In SRAM PUFs, the power-up values of the SRAM memory cells are used as unique identifiers. Commercial implementations of SRAM PUFs are already available through hardware security solution suppliers such as NXP [3], intrinsic ID [4] and Microsemi [5].

The main drawback of SRAM PUFs is their limited reliability, as they do not always return the same response to a challenge, an essential feature of a PUF. Reliability is further diminished by environmental conditions during operation and device aging. This problem can be solved by utilizing a series of pre- and post-processing techniques, which, together, form the helper data algorithm (HDA) [6]. Different techniques are available, and the development of new ones has been an important topic of research for the past years [7, 8, 9]. Particularly, a common post-processing technique is the use of error correction code (ECC) circuits, to ensure a correct response. However, this circuitry implies a high cost in terms of area, power and the requirement of redundant bits out of the PUF, a cost that increases linearly with the unreliability of the PUF. As one of the main advantages of PUFs is their low cost, there is a great incentive to increase reliability as much as possible with the goal of reducing the complexity of ECC.

One way to increase reliability in SRAM-PUFs before applying the ECC is through bit selection, which reduces the incorrect responses by selecting the most reliable cells of the array. The main goal of this project is to exhaustively validate through experimental results a new bit selection technique. It performs a better selection than previously reported techniques, making possible a significant reduction of the ECC circuitry needed. The method is experimentally validated against voltage and temperature variations and aging effects in an SRAM array designed in a 65nm CMOS technology.

In chapter 2 an introduction to PUFs is presented, including some of the available post-processing techniques to improve their reliability. Then, a more detailed look into SRAM PUFs is done in chapter 3, including the effects of environmental variations and aging mechanisms on reliability. Afterwards, in chapter 4, a variety of SRAM-specific pre-processing techniques are explained, which serve as point of reference for the new pre-processing technique, Maximum Trip Supply Voltage (MTSV). This technique is validated at different operating conditions, demonstrating its benefits even in the case of aging of the SRAM cells. Finally, in chapter 5, the process by which an SRAM PUF's unreliable response is transformed into a reliable one is evaluated in detail using a fully characterized SRAM PUF chip. Through this process, the performance of a selection of cells based on MTSV is compared to other selections, which will illustrate the degree of improvement achieved thanks to MTSV. Afterwards, the ability of these selections to generate a key with an ECC is tested. Finally, the reduction in cost achieved by MTSV is shown by comparing the required ECC for a selection based on MTSV and the requirements for other selections.

Chapter 2

Physical Unclonable Functions

This chapter presents an introduction to the concept of Physical Unclonable Functions (PUFs). After a brief description of their main characteristics and applications, the most common implementations are reviewed. The metrics selected to evaluate PUFs are then defined and the HDAs used to enhance their performances are introduced.

2.1 What is a PUF?

The term Physical Unclonable Function (PUF) was coined by Tuyls et al. in [10] and has become the prevalent term, but the notion of "Physical one way function" was first introduced by Pappu in [11]. There are multiple definitions given for PUFs [12, 2], but, in general, it is an entity that takes advantage of process variations on a measurable physical property to generate an unpredictable and unique response to a device. Process variations are the naturally occurring disparities between components with identical design during manufacture. This disparity is small and can not be modelled or replicated even by the designer or the manufacturer of the device.

PUFs can be thought of as a black-box challenge-response system [13]. If a challenge c is issued, it returns a response $r = f(c)$. This forms a challenge-response pair (CRP). The fundamental aspect of PUFs is that f (Input/output relations in the PUF) is hidden from the user or the potential attacker.

The essential characteristics that must be present in a PUF's response are **uniqueness** (how unique is the response of a particular instance among a group of instances of the same type), **unpredictability** (how unpredictable the response is even with knowledge of part of the response or of another PUF's response) and **reliability** (how reproducible the PUF response is when the same challenge is applied at different points in time). Considering three PUFs named A, B and C and their corresponding CRPs, these PUFs characteristics are schematically depicted in fig. 2.1.

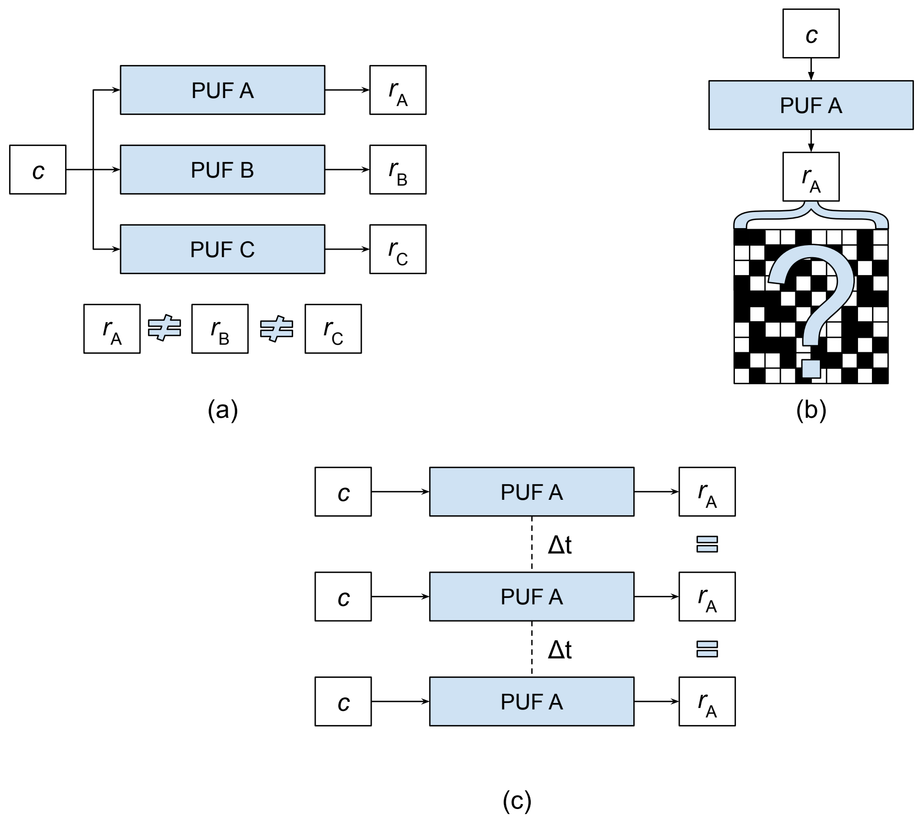

Figure 2.1: Desired characteristics in a PUF's response: (a) represents uniqueness, (b) unpredictability and (c) reliability.

PUFs can be distinguished between weak or strong according to the amount of CRPs available [13]. A weak PUF will support a small amount of challenges. Therefore, the CRPs must be kept secret, otherwise the PUF can be emulated. In some cases there may be only one challenge and accordingly only one CRP. Strong PUFs on the other hand have such a large amount of CRPs that they cannot be mapped in a reasonable time frame. Due to this, its output does not need to be private. It should not, however, reveal information about the functionality of the PUF since, otherwise, responses to different challenges could be predicted.

2.2 PUF applications

PUFs have a variety of applications but almost all of them are geared towards security, particularly for digital devices [14, 2, 12]. They are used as one of the main components to build RoT or hardware anchors to reinforce security in the device in which the RoT is incorporated. The PUF response acts as the foundation to derive trust for the rest of components of the device.

Several ways to design and develop a RoT are presented in literature. A rough classification can be provided according to the implementation nature. A RoT can be implemented in hardware, software or a hybrid hardware/software version. The main advantage of hardware RoT compared to software RoT is being invulnerable to malware attacks. One of the most popular technologies to implement hardware RoT is the Trusted Platform Module (TPM) [15]. It is a standard for a secure cryptoprocessor, which is a dedicated microcontroller with the required security primitives to ensure the integrity of the device. This includes the generation and storage of cryptographic keys and true random number generation, and requires physical security mechanisms to provide anti-tamper protection. However, they involve a large cost in terms of area and power which is suitable for in small, resource constrained applications such as those of Internet of Things (IoT) or embedded devices.

PUFs offer two advantages in this field compared to the conventional hardware RoT solutions: cost reduction and an increased level of security. Regarding cost, secret information to derive cryptographic keys are stored in a secure NVM for traditional systems. Meanwhile, a PUF's response is used for generation of keys on-the-fly without being necessary their storage in NVMs. Furthermore, the PUF existing implementations are not expensive in terms of energy and area. Some PUF implementations even use already existing circuitry of the device, which further reduces these costs. Regarding security, PUFs are practically impossible to attack through reverse engineering methods due to their unclonability. As mentioned before, the response is generated on-chip. This is an advantage in terms of security as well, since the traditional method of transferring secure information to the chip is an additional vulnerability against eventual attacks. Furthermore, since PUFs rely on small process variations, they have an innate protection against physical tampering as any meddling may modify the response and turn the PUF unusable.

In the next subsections different security applications of PUFs are presented.

2.2.1 PUF as a unique identifier

Since PUFs generate a unpredictable and unique response, they can be used directly for identification [1, 14, 16]. Each unique PUF is assigned to one physical system. In this way, PUF identification is similar to biometrical identification. The PUF output constitutes the "digital fingerprint" of the system.

A simple example of the use of a PUF as an unique identifier is shown in fig. 2.2. Two phases can be distinguished when using a PUF as an unique identifier: enrollment and identification. During enrollment, shown in fig. 2.2 (a), a number of CRPs from every PUF are stored in a database coupled with the physical system associated with the PUF. This step is done only once. Later, during identification, shown in fig. 2.2 (b), the verifier selects one challenge from the database and challenges the PUF with it. If the resulting response matches the stored one, the identification is positive, and, otherwise, it fails.

In strong PUFs, where multiple CRPs are stored, there can be a degree of tolerance with the discrepancies between the responses [14]. This is due to the fact that each CRP is used only once to prevent replay attacks. In weak PUFs, there is only a small number of possible challenges so the verification response should match perfectly the enrollment response. The PUF must be examined in an environment where an authenticating party is present to ensure that the PUF itself is being evaluated and there are no replay attacks.

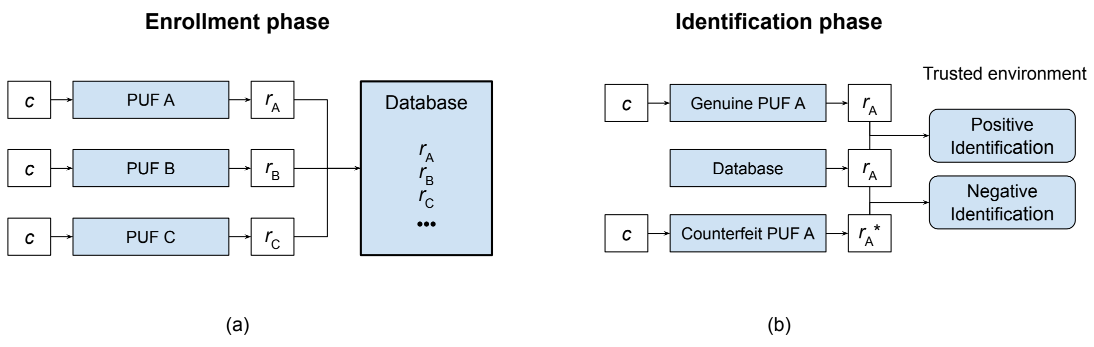

Figure 2.2: Simple implementation using a PUF as an unique identifier. (a) Enrollment: occurs only once. (b) Identification: occurs any time the system has to be identified.

2.2.2 PUF as a secret key generator

PUFs can be used as well for key generation in cryptographic encryption and decryption algorithms [17, 18, 13]. Traditionally, the key is permanently stored in a NVM, which opens up multiple points of attack. However, PUFs generate the key whenever required instead of storing it, providing better security. This application has strict requirements for the response as it must be perfectly reliable and unpredictable to ensure the security of the cryptographic scheme. Weak PUFs that provide only one response are optimal for secret key generation [14], as only one key is required in cryptographic algorithms.

An important distinction can be made between symmetric and asymmetric cryptographic algorithms. In symmetric key algorithms, shown in fig. 2.3, the same key is used for both encryption and decryption. This means that the key must stay secret during the whole process. The key has to be provided to the receiver in advance for authentication through a secure channel, which in some applications is complicated and opens up more points of attack [19].

On the other hand, asymmetric or public key algorithms use different keys for encryption and decryption. One of the keys can be public, knowing it will not compromise the security of the system. Having a public key algorithm instead of a symmetric one is useful when secure channels are not available, the number of participants in the encryption is high, or when keys are frequently changed but the security requirements are higher.

In a public key algorithm, first the PUF device generates a key used to create a private and public key pair (fig. 2.4 (a)). The public key is broadcast on a public

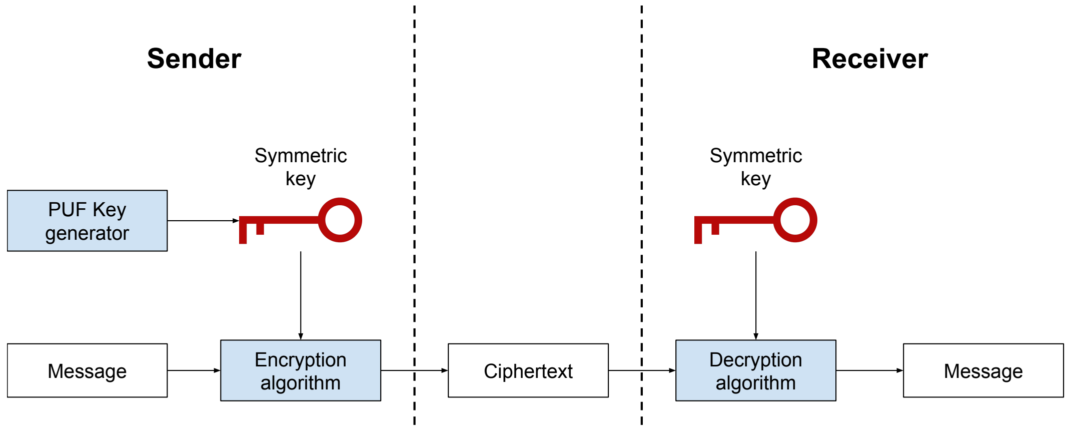

Figure 2.3: Use of a PUF as a key generator in a symmetric cryptographic algorithm.

key server that any party can access. From here, there are different applications. The PUF device can be authenticated by sending certain data to be encrypted by the private key. If the original data is recovered through the public key, the PUF device is correctly verified. Similarly, any transaction signed by this private key can be verified through the public key. Device-to-device secure communication is another possibility, where a certain message encrypted by the sender through the public key can only be decrypted and read by the intended receiver through the private key (fig. 2.4 (b)).

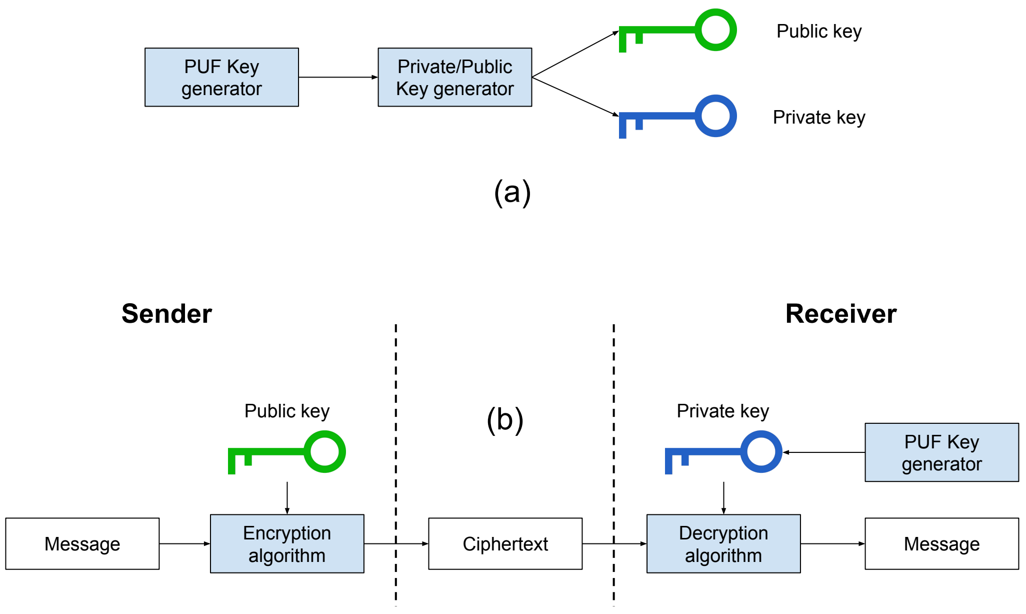

Figure 2.4: Use of a PUF as a key generator in a public key algorithm.

The requirements in terms of minimum key length for these algorithms can vary greatly, from 128 bits for the symmetric algorithm AES [20] to 2048 for the asymmetric RSA algorithm [21]. The chosen algorithm and entailed level of security will require different lengths, but the bare minimum is 128 bits [8].

2.2.3 PUF as a source of entropy

True random number generators (TRNG) generate random numbers from unpredictable physical processes. Although their main field of application is in cryptography, they are used in gambling as well as in situations were randomness is useful, such as juror selection or military draft lotteries. One example of their use in cryptography is to generate nonces [8], an arbitrary number employed just once in a cryptographic communication to avoid replay attacks [22]. These nonces can be used as initialization vectors (IVs) to prevent a sequence of text that is identical to a previous sequence from producing the same exact encrypted message. PUFs that can perform both as a key generator and as a source of entropy are specially useful in cryptographic algorithms [23, 24].

PUFs used as a source of entropy take advantage of the unpredictability of PUF implementations (fig. 2.5). In terms of reliability they ideally require the exact opposite of the applications presented so far: a completely random response. To quantify the entropy or randomness of a PUF response, NIST tests [25], developed by the National Institute of Standards and Technology, can be used.

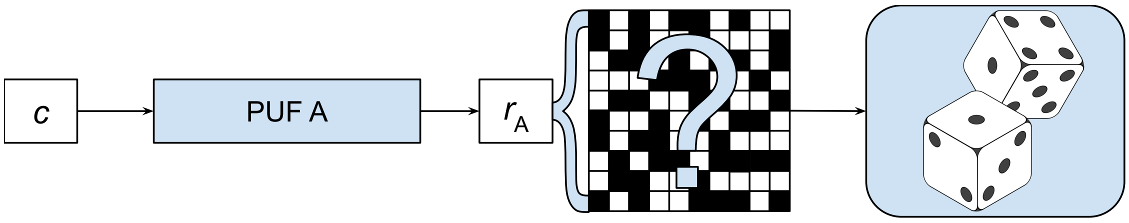

Figure 2.5: Use of a PUF as a source of entropy.

2.3 PUF Implementations

In this section a brief overview of different PUF implementations is provided. Many different approaches have been published, since the concept of PUFs can be applied to plenty of physical phenomena. Some implementations take advantage of already existing properties of the system they secure, while others are embedded. The implementations presented here cannot be taken as a complete list, only some of the most significant ones are considered. A more comprehensive listing can be found in [1, 14].

2.3.1 Optical and non-electric PUFs

The original optical PUF presented in [11] relies on the interaction of a laser beam with a scattering medium in its path. As shown in fig. 2.6, the input is the position and polarization of the laser (strong PUF) and the output is the ”speckle” of light exiting the scattering medium, which is recorded by an imaging device. A hash is applied to this recording to obtain a string of bits. The scattering path depends heavily on the input and the structure of the scattering medium, and unless this medium is very simple it is practically impossible to model and predict the output. In the mentioned implementation, microscopic refractive glass spheres embedded in a transparent epoxy plate are used as scattering medium. This implementation is not practical due to the complexity of the system, it served as a proof of concept. A more integrated design was proposed in [26] and more recently in [27]. Optical PUFs provide an unmatched input/output complexity which cannot be obtained by electrical PUFs, but their high cost make them hard to justify in real implementations.

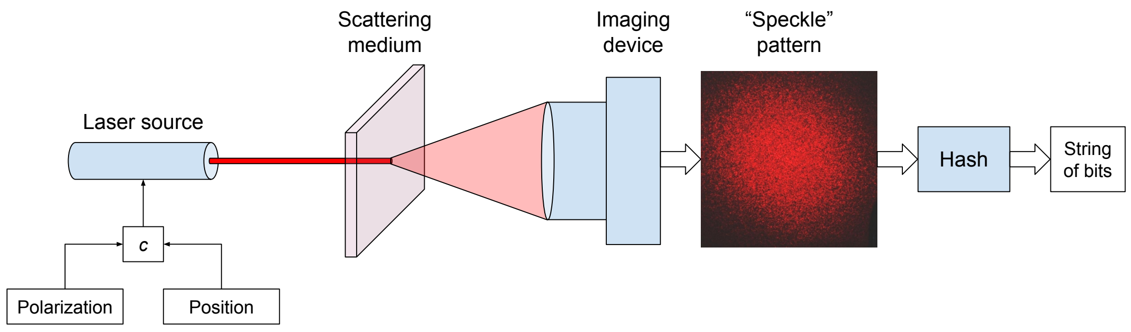

Figure 2.6: Schematic illustration of an optical PUF implementation.

Electrical or silicon-based PUFs are naturally easier to implement in an electronic medium and less costly, so the rest of the designs considered are electrical. There are, however, other proposed non-electrical PUFs such as Paper PUFs (using the unique fiber structure mainly for currency notes) [28], CD PUFs (using the lands and pits on a compact disk for identification) [29] or Acoustical PUFs (using the characteristic frequency spectrum of an acoustical delay line) [30]. The difference between these goes to show how versatile and universal the concept of PUF is, and more possible implementations come up as time goes by [1].

2.3.2 Arbiter PUFs

The initial proposal for an arbiter PUF was made in [31]. It is considered a delay PUF. This type of PUFs exploit the random variations in delays of wires and gates on silicon. Arbiter PUFs introduce a digital race condition on two paths on a chip. An arbiter circuit decides which path is faster. They are designed in an identical way so that the resulting difference in delay is a product of manufacturing variations which cannot be controlled. The initial design uses *switch* blocks connected in series with two states, where each state uses a different connection (*straigh* or *switched*). The challenge of the PUF will be the setting of the switches. In this way, the number of CRPs depends exponentially on the number of switch blocks used and it constitutes a strong PUF. Fig. 2.7 shows a schematic for an arbiter PUF.

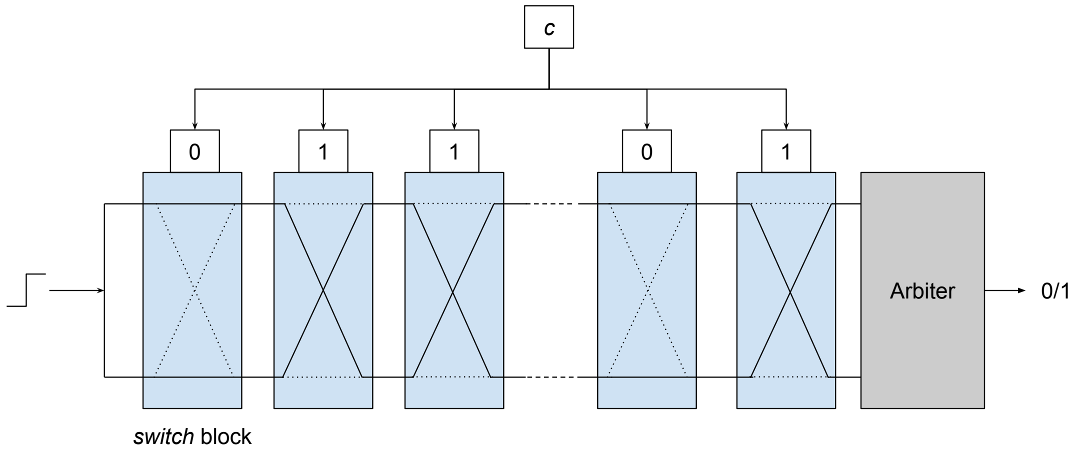

Figure 2.7: Schematic of an arbiter PUF implementation.

These responses can however be noisy. If the offset between the two paths is too small, the setup-hold time of the arbiter is violated and the result will depend on random noise, causing unreliable bits. Environmental factors such as temperature, supply voltage or aging play a significant role in which path is faster as well. Delays in each switch are linear and independent from each other, so arbiter PUFs with this design are vulnerable to the use of standard linear system analysis to model them and predict their response [32]. To solve this, non-linearities are introduced. These come at the cost of even more noisy results and even if the response is less predictable it can still be modelled [14].

2.3.3 Ring oscillator PUFs

A Ring Oscillator (RO) is a device composed of an odd number of inverters attached in a chain where the output of the last inverter is fed into the first one. Accordingly, the output oscillates between high and low voltage. Its oscillation frequency (ν) depends on the sum of the gate delays of all inverters, and their structure is very simple (composed only of standard logic gates), so it can be easily implemented in a field-programmable gate array (FPGA) or an application-specific integrated circuit (ASIC).

The RO PUF, first proposed by Gassend et al. in [33], measures the difference in oscillation frequencies between pairs of identically designed ROs caused by manufacturing variations. Like arbiter PUFs, they fall under the delay PUFs category. Figure 2.8 illustrates a standard RO-PUF implementation. It includes N identically laid out ROs, two N-bit multiplexers, two counters and a comparator. Since the delay of each inverter will be different due to process variability, the frequency of each RO will be unique. The outputs of these ROs are fed into the multiplexers, and the PUF challenge determines which ROs are compared. The frequency is measured through the counters, which count the number of oscillations for a fixed interval of time. Finally, the output bit is determined based on which oscillator has a higher frequency, i.e. has a greater count.

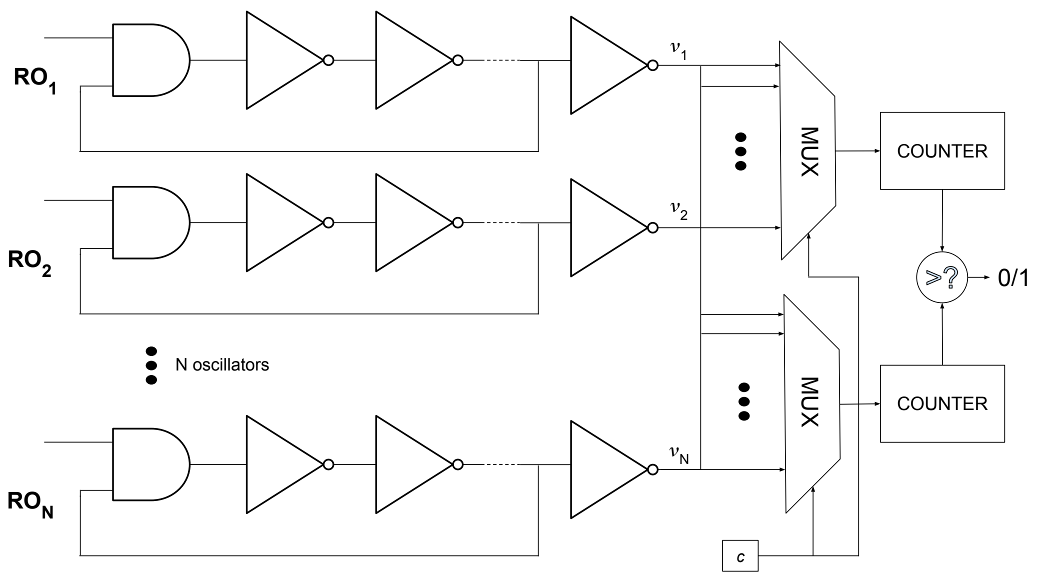

Figure 2.8: Schematic of a conventional RO PUF implementation.

For N oscillators, up to $log(N!)$ bits can be extracted by comparing frequencies. More pairings are possible but they would start to correlate with each other, compromising the security. This is a weak PUF, as there is a limited number of bits that can configure the PUF's operation. As in the case of the arbiter PUFs, RO PUFs are very susceptible to environmental variations since they are based on delay, and responses can be noisy.

2.3.4 SRAM PUFs

The first SRAM PUF was proposed separately by Guajardo and Holcomb in 2007 [34, 35]. They use the start-up value of SRAM cells, which consist of cross-coupled inverters and are used as static memory. Before the cell powers up, the SRAM does not have any value stored since it is a volatile memory. Once it powers up, it theoretically exists in a metastable state where the two inverters are identical. However, due to mismatch from process variation, one of the inverters starts conducting before the other. Due to the positive feedback present in the cross-coupled inverters the cells settles in either "1" or "0" depending on which inverter powers up faster, as shown in 2.9. This state is used as the response. Whether one loop or the other is stronger depends on the relative threshold voltage between the transistors. This behaviour can be exploited to obtain a PUF. A response can be generated by using a number of cells, one per each desired bit. This is an extreme example of a weak PUF , since there is only one challenge (powering up) and one response.

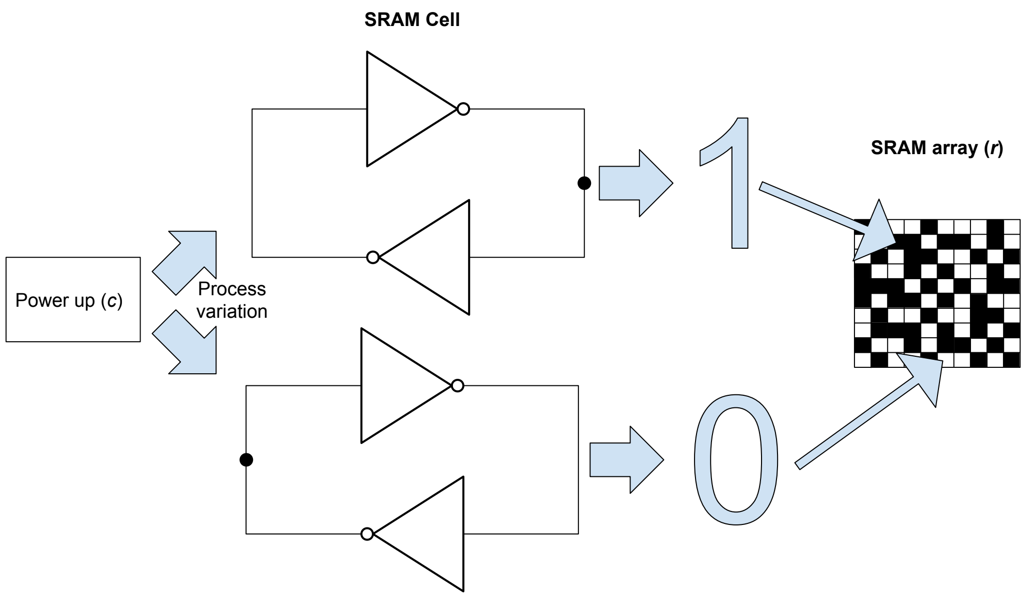

Figure 2.9: Schematic of an SRAM PUF implementation.

SRAM PUFs are quite popular compared to other implementations [36, 37, 38, 39] because they provide multiple advantages. SRAM memories are a ubiquitous device and a small fraction of a general purpose SRAM memory cells can also be used as PUF, in which case the PUF would come at almost no implementation cost. They have a low area and power consumption when compared to other PUFs and are easily scalable. Since SRAM memories have been a staple of electronic systems for a long time they are efficient, well characterized and understood. However, their main drawback is their limited reliability due to noisy responses. These PUFs will be the focus on this work so a more detailed description on the use of SRAM as PUFS is provided in the next chapter. Many studies have been published regarding the different aspects of SRAM PUFs and how to improve them due to the potential of the technology.

2.4 Metrics

Proper metrics are fundamental to evaluate the performance of PUFs and ensure their functionality in the applications shown in sec. 2.2. Their goal is to properly measure **Uniqueness**, **Reliability** and **Unpredictability** without redundant metrics. These characteristics have been explained qualitatively but they must be characterized quantitatively. There is a wide range of possibilities on which ones to use. A comprehensive list of metrics specifically created for PUFs can be found in [40]. PUF implementations usually provide a binary response either due to the physical property they measure or by applying some postprocessing after that measurement. Accordingly, the metrics used for PUFs are for binary strings of variable length.

2.4.1 Probability

A fundamental concept in PUFs as a statistical process is the probability of occurrence of a “1” response, p. If the elements of a string of bits are random variables, their response corresponds to a binomial distribution [2]. This distribution is defined as:

$$P(x = k) = \begin{pmatrix} n \\ k \end{pmatrix} p^k (1-p)^{n-k} \tag{2.1}$$

where n is the number of bits in the bit string and k the number of “1” responses. When the number of bits is sufficiently large, it can be approximated by a normal distribution which significantly simplifies obtaining the probability. The probability density function for the normal distribution is:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{2.2}$$

where μ is the mean of the distribution and σ is the standard deviation. The conversion from the binomial to the normal distribution is as follows:

$$\begin{aligned} \mu &= np \\ \sigma^2 &= npq \end{aligned} \tag{2.3}$$

Lastly, the mean of the normal distribution for N binomial distributed data sets (DS) can be approximated as [2]:

$$\mu_{all} \approx \frac{\text{\#of 1's in all DS}}{n \cdot N} \tag{2.4}$$

And the probability of each cell is derived from here by using eq. 2.3:

$$p = \frac{\mu_{all}}{n} \approx \frac{\text{\#of 1's in all DS}}{N} \tag{2.5}$$

2.4.2 Bit Error Rate

The Bit Error Rate (BER) of a response is defined as the ratio between the number of false bits e and the whole number of bits in a bit string N [2]:

$$\text{BER} = \frac{e}{N} \cdot 100 \tag{2.6}$$

It gives a clear measure of how error prone the PUF is. Accordingly, it helps to measure **Reliability**. The stability (number of correct bits) is complementary to the BER and usually only one of the two is given. To determine whether a bit is correct or false, first, there must be a string with the ideal or golden response for reference. This reference mask is calculated from the probability of each bit, if $p \geq 0.5$ the ideal value will be 1 and if $p < 0.5$ the ideal value will be 0. The mask can be determined directly from the evaluated dataset (intrinsically) or given as an input (extrinsically). The extrinsic reference is the golden response obtained under nominal conditions. It can be useful if, for example, the device is measured in varying conditions of voltage or temperature and a bit response switches from preferring 1s to 0s or vice versa. This would introduce a persistent error in the response when compared to the nominal conditions, but if the ideal string were determined intrinsically it would appear as if the cell was working properly.

The BER can be obtained bit-wise ($N = 1$) or response-wise (N depends on the number of bits in the response). The second one gives a clear idea of the response quality and is generally the one used, although the first one can be useful for graphical representations (as in fig 2.10) or to obtain more detailed information about the behaviour of each bit. The ideal value for BER would be 0% in either case.

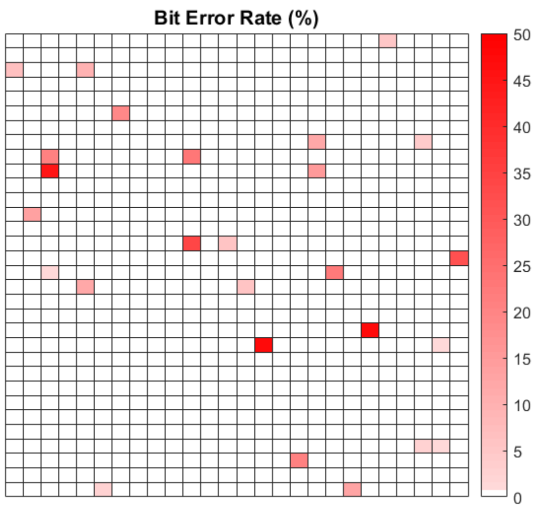

Figure 2.10: Bit-wise representation of BER for a given dataset

2.4.3 Maximum intra hamming distance

The hamming distance between two strings is the number of different bits, i.e. the bits that need to be flipped for the two strings to be equal. The maximum intra hamming distance is the highest value among the m hamming distances between each response to the same challenge and the golden response. This can be defined formally as:

$$\max_{\text{IntraHD}} = \max_{i=1,\ldots,m} \left[\frac{HD\left(R_{ideal}, r_i\right)}{n} \right] \cdot 100 \tag{2.7}$$

Where n is the number of memory cells and m the number of responses generated from the device. R_{ideal} is the golden response and r is one of m responses. As in the BER the ideal response can be determined intrinsically or extrinsically. It is a measure of **Reliability** as well, but gives an idea of how bad the worst case scenario is instead of the general reliability of the device. Zero is the ideal value for this metric, since it means all responses to the same challenge are equal.

2.4.4 Mean Inter Hamming distance

The mean inter hamming distance is calculated as the mean of the hamming distances between the golden responses of all chips. This is obtained through the following expression [40]:

$$\text{mean}_{\text{InterHD}} = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD\left(R_i, R_j\right)}{n} \times 100\% \tag{2.8}$$

Where k is the number of chips. The $2/k(k-1)$ factor computes the mean. This metric gives a measure of the **Uniqueness** of the PUF. In the ideal case it is 50%, meaning that the response of each chip is independent from each other. If uniqueness is low, knowing the response of one PUF would make predicting the response of other chips easier and security would be reduced.

2.4.5 Hamming Weight

Hamming weight simply measures the percentage of 1s in the m responses to the same challenge:

$$\text{HW} = \frac{1}{n \cdot m} \sum_{i=1}^{m} \sum_{l=1}^{n} b_{i,l} \cdot 100 \tag{2.9}$$

Where $b_{i,l}$ is the l-th binary bit of a n-bit response from a chip and m is the number of responses. This metric provides an insight into the **Unpredictability** of the response, as a non uniform one where a certain state is favoured will make predicting the response easier. Ideally the result is 50%, a balanced distribution of 1s and 0s.

2.5 Helper Data Algorithms

For the sake of better understanding, so far the PUF has been considered ideal so that its response meets the specifications described in fig. 2.1. However, due to environmental conditions (e.g. noise, polarization changes or temperature variations) and circuit aging, some response bits will not have a stable output and their value will change after repeating the same challenge, as shown in fig. 2.11. This is known as the bit-flipping problem [24, 41, 42]. Bit-flipping is particularly problematic for silicon PUFs and, specifically, SRAM PUFs. Weak PUFs used as unique identifiers or to generate cryptographic keys must have perfect reliability, otherwise a legitimate PUF may be considered a fraudulent one.

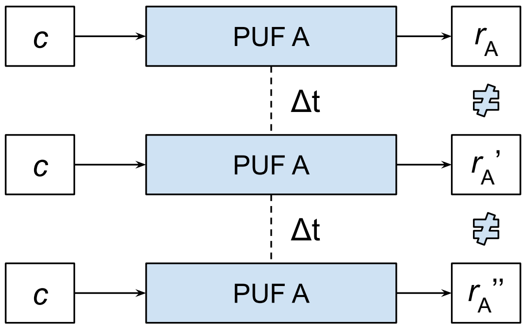

Figure 2.11: Illustration of the bit-flipping problem. The same PUF gives a different response to the same challenge at different points in time.

In order to avoid this problem, HDAs are applied so that PUFs responses meet the required reliability specifications for key generation in any implementation [6]. They transform noisy and non-uniform responses into reliable and uniformly distributed keys. To achieve this improvement, they require helper data. These specifications depend on the implementation's demands (such as the mean time between two successive accesses) but a target used thorough the literature is a failure rate of less than 10^{-4} % [2, 6]. This means that the key has a chance of one in a million to be erroneous. The failure rate is also known as Key Error Rate (KER). 10^{-4} % is a reasonable target for a mean inter access time of one minute [8].

HDAs are applied in three consecutive steps: bit selection, error correction and entropy compression [6]. Bit selection consists on using the most reliable bits out of the available ones. Error correction uses ECCs which lower the probability of failure by taking redundant bits. Finally, entropy compression uses a hash function to make the key uniform with perfect entropy and requires redundant bits as well. The need for these redundant bits means that the required PUF's response will be longer than the resulting key.

There is always a trade-off in terms of cost and improvement with each technique used. The goal when selecting a HDA is to meet the required specifications in the most efficient way possible. Bit selection, error correction and entropy compression are not mandatory, there are schemes that rely only on heavy error correction [43], that do not require any error correction [44] or that go without entropy compression [22].

The specific implementation of a HDA can take different forms [7]. Two common constructions are considered: Code-offset, presented in [45] and applied for example in [46, 43], and fuzzy commitment presented in [47] and applied in [48, 49]. They are capable of reaching a similar level of reliability [50] and employ the same type of ECCs.

In a code-offset construction two phases are required: Enrollment and Reconstruction. Enrollment, shown in fig. 2.12 (a), is performed only once in a secure environment and after manufacturing. In this phase, a codeword c is generated with the error correction encoder by using a random secret x. The PUF response r is employed to generate the key through a hash function. A XOR operation is performed on this response and the code word to generate the helper data h. In principle, helper data can be made public without compromising the security of the key, and may be stored in an internal NVM, an external memory, in a server, etc. Reconstruction, shown in fig. 2.12 (b), is in-the-field deployment, performed when the key is required. First, a XOR on the noisy PUF response r' and the stored helper data results in a codeword c'. Then, if the number of errors does not exceed the tolerance of the code, the error correction decoder fixes the mistakes in c' and returns c. Afterwards, the original PUF response r is recovered by performing a XOR on the codeword c and the helper data. The original key is obtained after r is hashed.

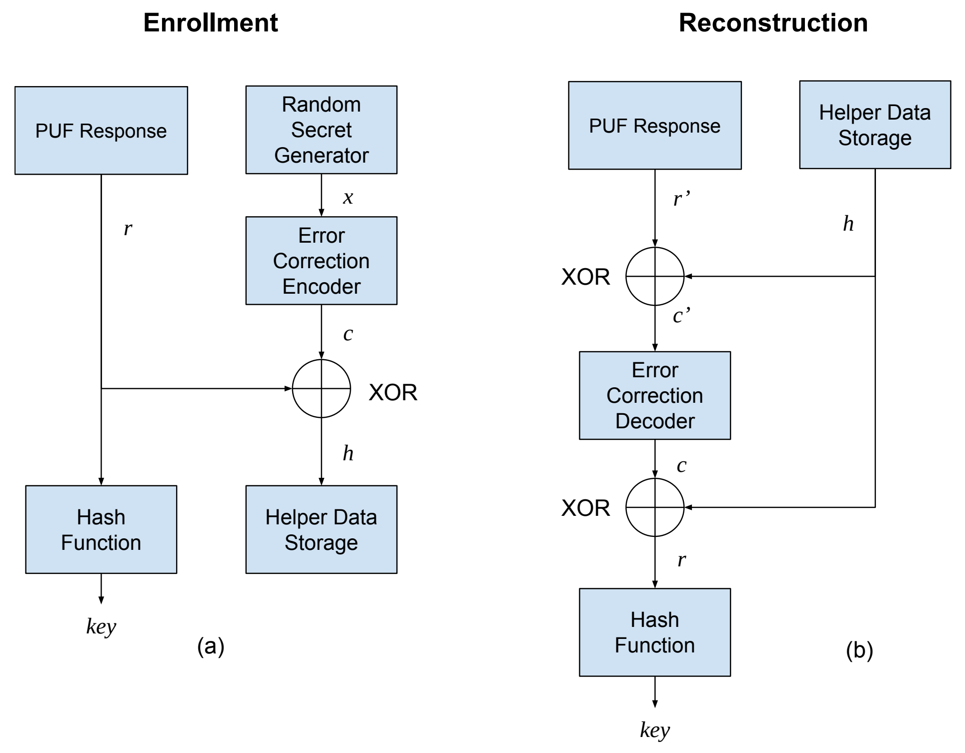

Figure 2.12: Helper data algorithm based on the code-offset construction. This algorithm has two phases: Enrollment (a) and Reconstruction (b).

In a fuzzy-commitment construction, the same two phases used in the code-offset construction are employed: Enrollment and Reconstruction. However, the approach in this construction is to obfuscate a randomly generated key or secret information through the PUF response instead of obtaining a key from the PUF response itself. In the Enrollment phase, shown in fig. 2.13, *key* is generated and then codified through the error correction encoder, obtaining c. Then, a XOR operation is performed on the codeword c and the PUF response r to obtain the helper data h, which is stored. During Reconstruction, c' is obtained by performing a XOR on the noisy PUF response r' and h. If the number of errors is low enough, *key* is successfully recovered.

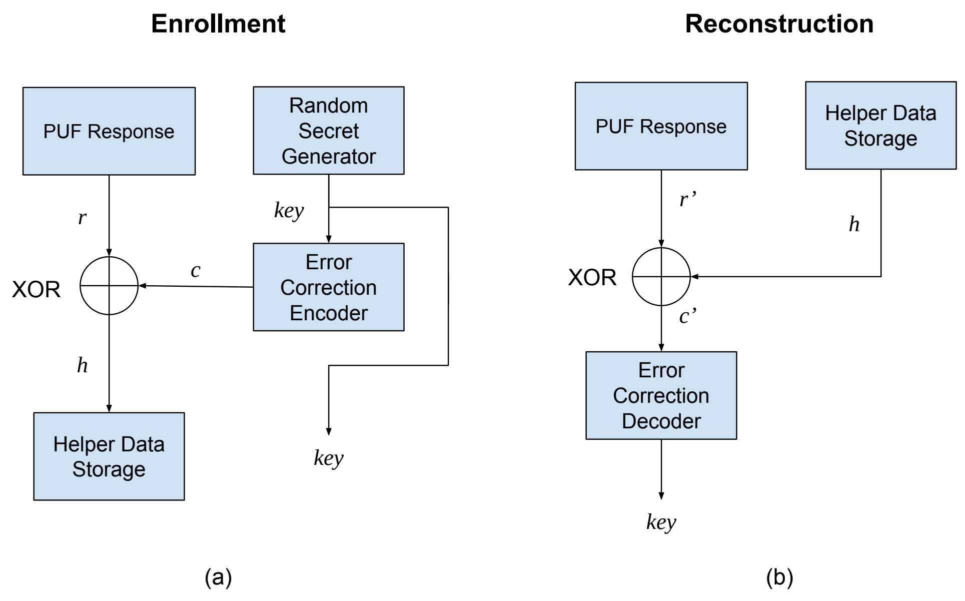

Figure 2.13: Helper data algorithm based on the fuzzy commitment construction. This algorithm has two phases: Enrollment (a) and Reconstruction (b).

Comparing both techniques, fuzzy commitment is simpler and easier to implement than code-offset. However, since it does not have a hash function, the helper data can leak information if the PUF response is non-uniform [45].

In the next subsections bit selection techniques, ECCs and entropy compression are examined one by one.

2.5.1 Bit selection

Bit selection improves the output reliability by using as PUF only a subset of the available bits. Their goal is to reduce the requirements for the ECCs. In some cases this selection simply removes the unstable bits out of the pool of available bits and in others it tries to rank the bits according to their reliability so the best selection possible is used.

Bit selection techniques vary according to the type of PUF employed. Since this work focuses on SRAM PUFs, only techniques applicable for SRAMs are considered. In these devices, the selection is done on the cells. For example, Multiple evaluation [24] is a widely used technique where all cells are powered up a number of times and those that do not always produce the same response are discarded. This is a straightforward technique but presents some drawbacks, as it requires many power ups and only discards unstable bits without making a classification of the stable ones. New bit selections techniques are sought, and some of them will be evaluated in chapter 4, where a new selection technique is presented.

2.5.2 ECCs

ECCs are a powerful and easy to scale tool to achieve any degree of reliability from a PUF response. They are well established since ECCs are of great importance in multiple fields such as computing, telecommunication or information theory to control errors in data over unreliable or noisy communication channels [51]. ECCs are the most common way to implement forward error correction (FEC) [51]. The central idea of FEC is to correct errors without requesting retransmission of the original information by introducing a known structure into a data sequence.

ECCs can be designed to correct bit-flips, bit-insertions and bit-deletion. The concern for PUFs is correcting bit-flips, so only ECCs geared towards this purpose are considered. In this work the PUF response is assumed to be a binary string, so only binary codes are evaluated.

The drawback of ECCs is their area and energy cost. These are around two orders of magnitude greater than the PUF itself [8] for common silicon PUF implementations and they increase linearly with the number of correcting bits [52]. On top of that, requiring redundant bits from the PUF comes at an expense that varies depending on the implementation. The rise in cost due to the increased hardware complexity provides a great incentive to reduce the unreliability of the response before applying the ECC.

As explained before, ECCs require certain helper data obtained during the enrollment phase of the HDA. This helper data differs from one ECC to another, and many are available [51]. To effectively choose one, an ECC can be evaluated through three criteria [12]:

1. **Error reduction:** Measured simply as the reduction in BER achievable through the ECC.
2. **Redundancy:** It is defined as the ratio of the number of bits required from the PUF to the length of the resulting key. As redundancy increases, so do the error correction capabilities, as well as the area and energy costs. Low redundancy is important if there is a constraint in the number of available bits in the PUF.
3. **Complexity:** Some ECCs are harder to implement in hardware than others or require more computational time. This criteria is more complicated to evaluate as it is heavily dependent on the specific implementation, but it is nonetheless important to consider. If, for example, a specific code performs better in redundancy or error correction compared to another one but the cost of implementing it increases significantly, the benefits from better performance may not be worth.

Each type of ECC offers advantages and disadvantages compared to the rest in terms of these criteria, and there is not a universally good choice. It is important to select the ECC that meets the requirements in the most efficient way possible, as the ideal choice may vary between implementations. Improving one of the criteria usually represents a trade-off with the others.

There are two categories of ECCs: block codes and convolutional codes [51]. Block codes add fixed size block of bits to the initial message. A sequence of message bits is followed by a sequence of redundant bits, which forms the codeword. In convolutional codes, the redundant bits are spread along the codeword. The choice between block code and convolutional code depends on the application [53]. Convolutional codes require a memory for encoding and decoding while block codes do not, which may be a limitation in certain implementations. Block codes are more established and easier to implement than convolutional codes [51], although the full potential of convolutional codes has not been reached yet [54]. Convolutional codes allow for on-the-fly decoding, which is very useful in continuous communications. Block codes require waiting for the entire codeword to start decoding. However, since PUF responses have a fixed length, they do not make use of this property, and convolutional codes are not as popular for PUF implementations since block codes seem like an easier and more obvious solution [46]. For this reason, block codes will be examined in more detail.

A block code C obtains a codeword c from a message m:

$$c = C(m) \tag{2.10}$$

In this way, the block code maps the message to a unique codeword (encoding). The length of message m is represented by k, while n represents the length of codeword c. The redundancy can then be expressed as $R = k/n$.

ECCs have a set of valid codewords in a certain space. The rest of possible codewords in that space are considered non-valid codewords. They are able to correct errors by changing non-valid codewords to their closest valid codeword and then decoding c back to m. Accordingly, a key parameter in a block code is the distance d. It is defined as the minimum hamming distance between two valid code words. This distance provides a direct measure of the error correcting capabilities of the code: up to $d - 1$ errors may be detected and up to $(d - 1)/2$ corrected, rounded down.

The most important and commonly used ECCs are linear codes [51], which will be the ones considered here. A linear code is one where the linear combination of two valid codewords is a valid codeword. Linear binary block codes are represented with the notation $[n, k, d]$, which sets the parameters of a specific block code.

There are plenty of possibilities regarding block codes, but some of the most important ones and their strengths will be evaluated now in increasing order of complexity. After that, one convolutional code implementation for PUFs will be briefly examined. Lastly, different ECCs implemented in PUFs and their requirements will be compared.

2.5.2.1 Repetition Code

The repetition code (R) is is the simplest code available. It obtains one bit of the key from n bits of the PUF's response through majority vote, so n must be odd and the redundancy is n. The distance in a repetition code is simply n. Some examples of PUF implementations that use this code can be found in [55, 46, 22].

To illustrate how this code work, a simple implementation using $R[3, 1, 3]$ for each bit of a 4-bit string will be examined in fig. 2.14. Initially the original message is encoded into four codewords by simply repeating each bit three times (fig. 2.14 (a)). Two different examples are considered for decoding. In the first one, fig. 2.14 (b), the codewords have three errors. However, they are spaced from each other and there is a majority of correct bits in each codeword. After doing a majority vote on each one, the original message is decoded. In the second one, fig. 2.14 (c), the codewords have seven errors. In this case, two of them have a majority of errors and the resulting message is incorrect.

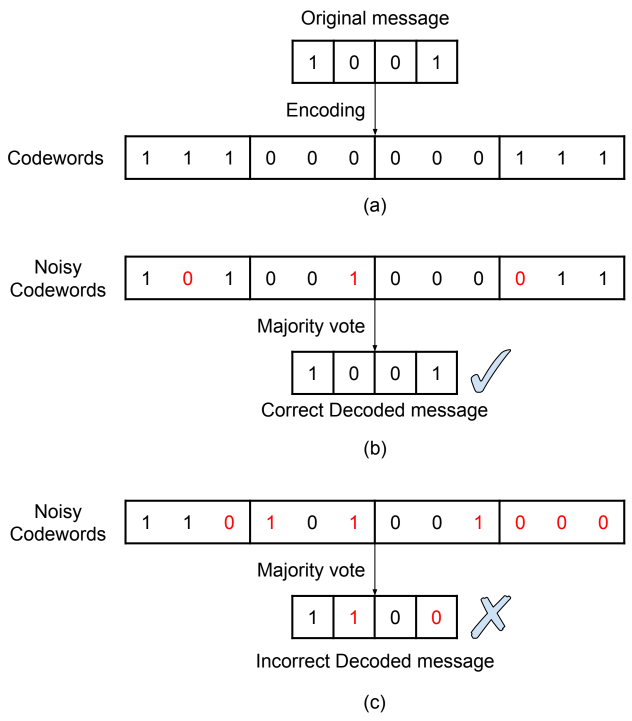

Figure 2.14: Repetition code $R[3, 1, 3]$ example for four bits. (a) shows the error correction encoding, (b) a successful decoding and (c) a failed decoding.

In any case, if less than half of the n bits of each codeword come with no error the original value will be recovered since the majority of bits will have the correct answer, otherwise it will fail. The error reduction capabilities of this code mostly depend on the initial BER. For a BER of $15 - 50\%$, the n required to meet specifications is prohibitively high, e.g. $n = 33$ for a BER of 15% [2]. However, if the BER is previously reduced, a repetition code with a small n is pretty effective.

This ECC has the highest redundancy possible, but its complexity is the lowest possible which makes it very easy to implement in any microcontroller. Repetition codes have entropy leakage issues, as pointed out by [6, 56]. These are specially significant for responses that are not perfectly uniform.

2.5.2.2 Hamming Code

The hamming code was one of the earliest ECCs developed, in 1950 by Richard Hamming [57]. It employs parity checks through embedded parity bits to detect up to two-bit errors and correct up to one ($d = 3$). Parity checks consist in evaluating the parity of a set of bits and comparing it to a reference value, the parity bit. In binary, evaluating the parity of a set of bits is equivalent to performing a XOR operation. The most common hamming code is $Hamming[7, 4, 3]$, which has three parity bits (p_1, p_2, p_3) and four message bits (d_1, d_2, d_3, d_4). One way to graphically represent these bits is shown in fig. 2.15.

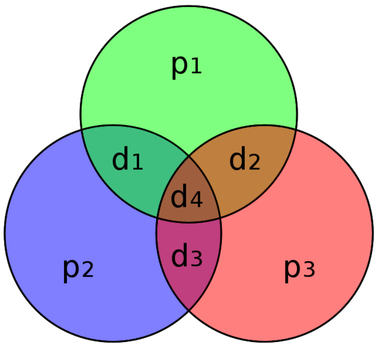

Figure 2.15: Graphical depiction of the 4 data bits d_1 to d_4 and 3 parity bits p_1 to p_3 and which parity bits apply to which data bits [58].

Each parity bit checks three different combinations of three data bits. Error correction is performed by evaluating the parity bits, and the erroneous parity bits determine which bit has to be corrected. Suppose that there is one bit error. If one of the parity bits is incorrect, then the error is in that bit. If two are incorrect, either d_1, d_2 or d_3 (determined by the two parity bits) have a bit flip. If all three are incorrect, d_4 has the error. This is illustrated by a practical example in fig. 2.16.

First, in fig. 2.16 (a), the original message is encoded by simply computing the parity bits and adding them to the message. Two different examples are considered for decoding. In the first one, in fig. 2.16 (b), the codeword has one error. Evaluating the parity bits reveals that p_1, p_2 and p_3 are wrong. If all three are incorrect, then d_4 has an error and gets corrected. The codeword is recovered and the message is correct. In the second one, in fig. 2.16 (c), the codeword has two errors. Evaluating the parity bits reveals that p_3 is wrong, so that bit gets corrected. Error correction failed because hamming codes can correct only one error, so the message is incorrect.

Regarding their performance, redundancy in hamming codes is very good, as they use the minimum amount of parity bits for their length [59]. This ratio becomes even better as the length of the message k increases. Their complexity is high compared

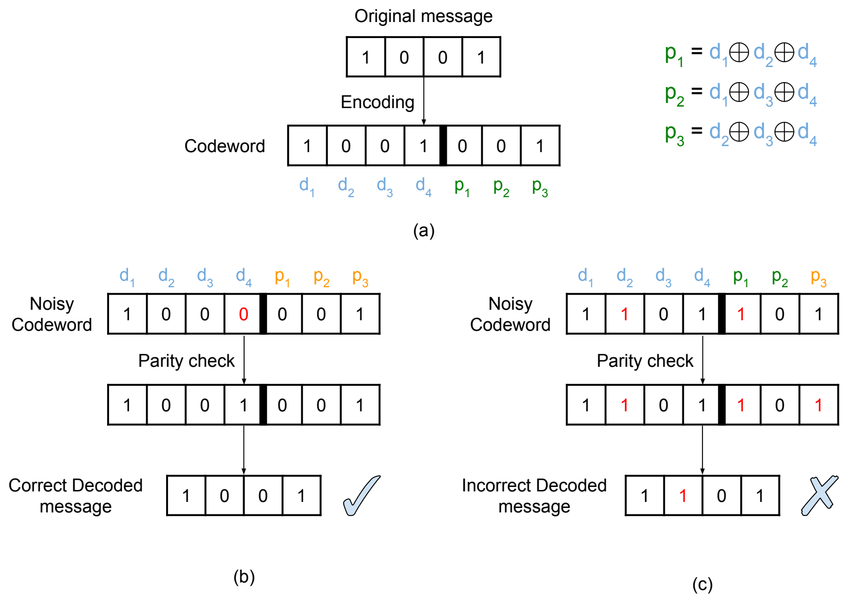

Figure 2.16: Hamming code $Hamming[7, 4, 3]$ example for four bits. (a) shows the error correction encoding, (b) a successful decoding and (c) a failed decoding.

to repetition codes but low compared to more advanced codes as they only require a few XOR operations to perform parity checks. However, they have low error reduction capabilities, as they can only correct one error. Since most PUFs do not have low error rates, Hamming codes are not popular.

2.5.2.3 Reed-Muller code

Reed-Muller (RM) codes are a big family of ECCs. They can be described in multiple ways. One clear explanation is provided in [60]. The code construction can be described as a greedy procedure, where the best set of codewords for each message m with a length k and a desired distance d are sought. Consider building a binary linear code; it must contain the all-0 codeword. If one has to pick a second codeword, then the all-1 codeword is the best choice under any meaningful criteria. If now one has to keep these two codewords, the next best choice to maximize the code distance is to include the half-0 half-1 codeword and to continue building codewords sequentially. Once saturation is reached at relative distance d, it is less clear how to pick the next codeword, but one can simply take linear combinations of the previously picked codewords, and iterate this after each saturation. This procedure results in the valid set of 2^k RM codewords, so that each possible combination of k bits has one codeword associated.

RM codes are often expressed as $RM(r, t)$, which corresponds to $RM\left[2^t, k, 2^{t-r}\right]$ in the standard notation. In this way, $n = 2^t$ and r represents the order of the code, an order by which the RM codes are classified. Reed-Solomon codes are one popular example of a high order RM code. However, for PUFs only RM codes of the first order are considered [43], also known as augmented Hadamard codes. These codes are relatively easy to encode and decode by using majority-logic circuits, unlike higher order codes which require complex decoders. They are represented as $RM[2^{(k-1)}, k, 2^{(k-2)}]$, for a message m of length k. These codes offer a middle ground between hamming and repetition in terms of redundancy and error correction capabilities, while being more complex than either of them. RM codes used in PUF implementations are found in [43, 48, 61].

2.5.2.4 BCH Code

BCH stands for Bose–Chaudhuri–Hocquenghem, their inventors. They are cyclic codes, meaning that codewords generated through circular shifts of a valid codeword are a valid codeword. In a string of bits, a circular shift implies moving every bit to the left one position and the last bit to the beginning. Its capabilities are based on certain algebraic properties obtained from identifying each string of bits with a polynomial. However, an explanation of these properties is complex and beyond the scope of this work, so the focus will be on its practical features as ECCs.

One of the main advantages of BCH codes is that they provide a wide variety of block lengths and corresponding code rates. All codewords will have a size $n = 2^i - 1$, where i is an arbitrary integer greater than two. The corresponding k and d parameters have to be calculated for each possible n. For example, for $n = 31$ the possible codes are $BCH[31, 26, 1]$, $BCH[31, 21, 2]$, $BCH[31, 16, 3]$, $BCH[31, 11, 1]$ and $BCH[31, 6, 7]$.

BCH codes use similar decoding algorithms as RM [62] so their complexity is similar. They offer better redundancy, close to Hamming codes, and the error correction capability depends on the parameters chosen. All of these facts make BCH codes a popular choice for PUFs [6, 43, 63].

2.5.2.5 Viterbi

In contrast with the proposed block codes so far, the Viterbi algorithm is a decoding algorithm for convolutional codes. In convolutional codes, the output code bits are continuously determined by logic operations (generally shifts and XORs) on the present input bit in a stream and a small number of previous bits, so a memory is required for encoding and decoding. Viterbi algorithms find the most likely sequence of hidden states (the message) that result in a sequence of observed events (the measured code bits). Error correction is based on the fact that a low number of erroneous bits are not influential enough to change the most likely sequence of hidden states.

PUFs implementations of convolutional codes with Viterbi algorithms are found in [64, 65]. These show promising results, outperforming block-code implementations in error correction capabilities and redundancy. Their main drawbacks are higher complexity and response times [64].

2.5.2.6 Comparison of ECCs in PUFs

In essence, there are many possibilities regarding ECCs. There has been a lot of effort by the scientific community to try and find progressively better codes, due to how promising PUFs are. A combination of ECCs in two or more stages (concatenated code) can make use of the strengths of two different codes [2, 43] and are quite popular. For two codes, C_1 and C_2, the message is codified by applying first C_2 and then C_1. The message is later decoded by applying first C_1 and then C_2.

Table 2.1 shows different ECC implementations found in the literature. All of these extract a 128-bit key from a PUF response that is presumed to have an average BER of 15 %, a very safe estimate even under the worst corners of reliability [46]. A fair comparison in terms of how costly the hardware implementation will be is hard to make as it is heavily dependant on the specific technology used, so only the required PUF size and final KER are shown. However, it is an important factor to keep in mind. It is important as well to distinguish between soft- and hard-decision decoding [64]. For hard decision decoding, all codeword bits have the same weight and contribute equally to decoding. Meanwhile, soft decision decoding uses reliability information about each bit to increase efficiency, increasing the error correction capabilities at the cost of higher complexity. Soft decision decoding is employed in [46, 48].

Reference	C_1	C_2	PUF bits	Redundancy	KER (%)
[43]	$R[33, 1, 33]$	-	5643	33	1 E-4
[43]	$BCH[1023, 46, 439]$	-	4092	23.92	1.85 E-6
[43]	$R[3, 1, 3]$	$BCH[127, 7, 31]$	2286	13.37	8.48 E-4
[43]	$R[7, 1, 7]$	$RM[16, 5, 8]$	3920	22.92	3.47 E-4
[46]	$R[3, 1, 3]$	$RM[64, 22, 16]$	1536	8.98	1 E-6
[48]	$R[14, 1, 14]$	$RM[8, 4, 4]$	4816	28.16	3.3 E-5
[66]	Viterbi	-	974	7.61	7.94 E-5

Table 2.1: Comparison of different ECC approaches with aimed KER 1 E-4 % showing required PUF bits and redundancy for a key length of 128 bits and a presumed BER of 15%.

It is clear that a large amount of PUF bits are required to obtain a 128-bit key, around five to fifty times the actual key size. This helps illustrate how costly ECCs are, and why there is such a motivation to preemptively reduce the PUF's BER by any means possible.

2.5.3 Entropy compression

The entropy of the response is non-maximum due to intrinsic correlation and bias introduced in the PUF, as well as leakage from the HDA [6]. By using this information, an attacker can reduce the search space required to obtain the secret key. A hash function solves this problem, as first proposed in [67]. This step is also known as privacy amplification. Hash functions require more input than output bits and work by compressing the entropy of the input into a shorter output. In this way, the total amount of entropy remains constant but the entropy per bit increases. The hash function breaks the link between responses and physical details of the PUF as well, adding one more layer of security and making model-building attacks much more difficult. Entropy compression further increases the amount of bits required out of the ECC and, accordingly, the amount of bits required out of the PUF.

In [34, 68], the entropy per bit of an SRAM PUF is estimated to be at around 0.75. To achieve a perfect entropy per bit of 1, $N/0.75$ bits are required [34] where N is the length of the key. This would be 171 bits for a 128-bit key or 342 bits for a 256-bit key. There are a variety of hash functions available with their advantages and drawbacks. A popular implementation is the SPONGENT lightweight hash function [65, 69]. An implementation of SHA-256 that is lightweight as well is found in [70]. The more bulky universal Toeplitz hash is used in [46].

Chapter 3

SRAM PUFs

This chapter presents a more detailed revision of SRAM-PUFs, so that contributions presented in the next chapters can be better understood. This starts by considering the architecture of the standard SRAM array, followed by a description of the 6T cell and a basic overview of its operation. Afterwards, its use as a PUF is evaluated with the advantages and challenges it presents. Finally, the environmental conditions and aging mechanisms that may affect the SRAM PUF's reliability are considered.

3.1 SRAM architecture

SRAM stands for static random-access memory. It is static since it does not need to be refreshed periodically, the data will be held as long as the memory is powered up, as opposed to a DRAM (dynamic random-access memory). When it powers down, it does lose its data, which makes it a volatile memory. SRAM memory is faster and more expensive than DRAM; it is typically used for CPU cache while DRAM is used for a computer's main memory.

An SRAM memory cache consists of an array of SRAM memory cells along with peripheral circuitry, such as row and column address decoders, sense amplifiers or write drivers [71]. An SRAM cell is a type of memory that uses a latch to store a bit. The cell is bistable (has two stable states), one corresponds to the value zero and the other to the value one.

Commercial SRAM chips include millions of memory cells distributed in columns and rows, conforming a two dimensional array. For example, a 4 MB SRAM memory will include 4,194,304 memory cells. To illustrate how the array is organized, a simple 4x6 SRAM array, corresponding to 24 bits, is shown in fig. 3.1. It is illustrative since a commercial SRAM array uses a similar structure, with larger address decoders and arrays. Cells are distributed in rows and columns. All cells in a row share a line, called wordline (WL), that is activated by the row decoder output. Likewise, all cells in a column share a pair of lines, called bitlines (BL and $\overline{BL}$), that are activated by the column decoder output. Encoding the address instead of specifying a row and a column reduces the number of interconnections by a factor of $log_2 N$, where N is the number of cells. This is of critical importance to make large SRAM arrays viable.

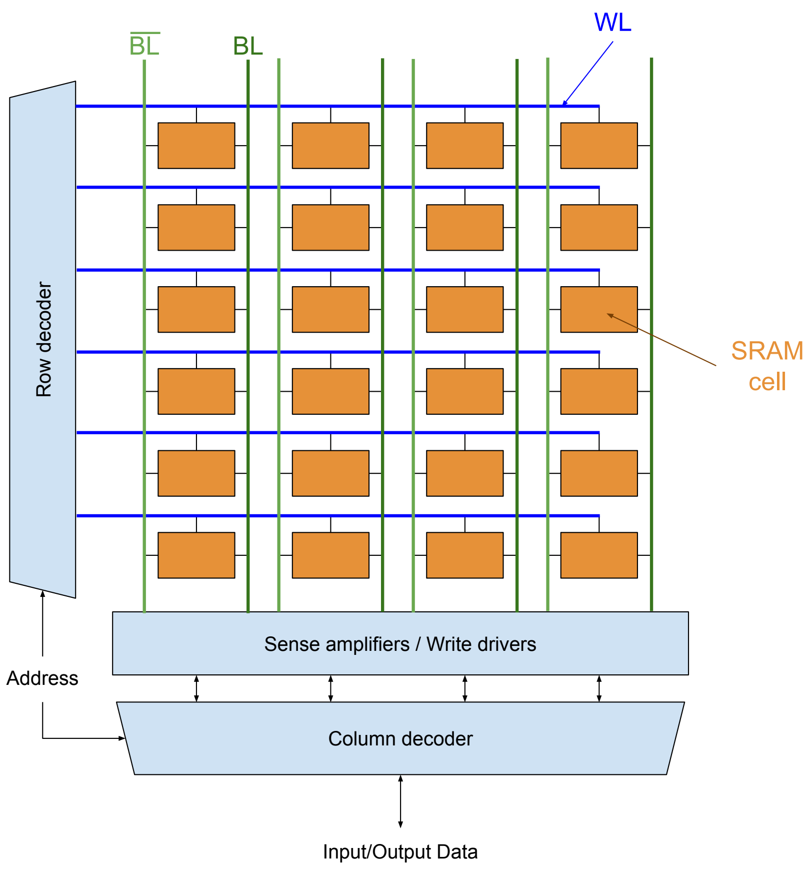

Figure 3.1: A simple 4x6 SRAM array structure

Each individual cell is addressed by selecting the appropriate wordline and bitline pairs. As mentioned before, these "lines" are selected by the row decoder and the column decoder, respectively, setting the specified wordline high and connecting the pair of bitlines to the sense amplifiers for reading, or to the write drivers for writing. Sense amplifiers amplify a small differential voltage to a full swing digital output signal, the value of the selected cell. Write drivers control the voltage level during write operation to change the state of the cell to the provided input.

In the following section the topology of the SRAM 6T cell, which is the one used in this work, will be examined in detail, along with how the read and write operations are performed.

3.2 The SRAM 6T cell

Although other designs exist, the standard topology for the SRAM cell is the 6T cell shown in fig. 3.2, which uses six transistors and is the cell generally employed in PUFs.

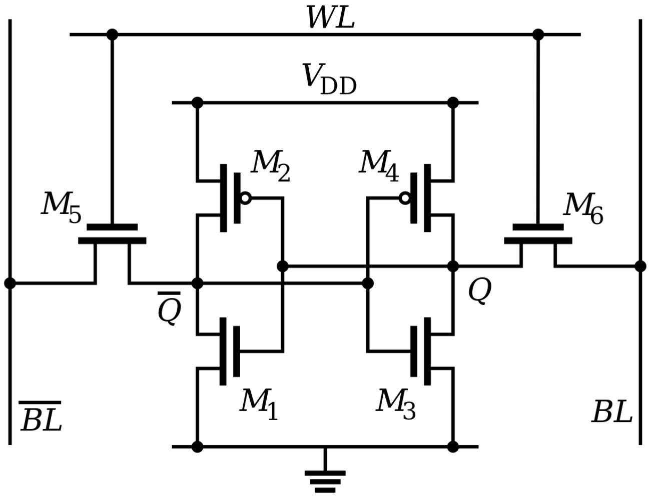

Figure 3.2: Schematic of a 6T SRAM cell

This cell is comprised of two cross-coupled inverters and two NMOS access transistors. The inverters set the voltage value of the internal nodes of the cell (Q and $\overline{Q}$) through one pull-up (PMOS) and one pull-down (NMOS) transistor each. When the system is in a steady state, one of the nodes is at a high voltage (V_{DD}) and the other at a low voltage (V_{SS} or ground). The values corresponding to V_{DD} or V_{SS} depend on the implementation. Accordingly, there are two possible states:

1. Q at high voltage and $\overline{Q}$ at low voltage.
2. Q at low voltage and $\overline{Q}$ at high voltage.

One of the states represents the logic value 1 and the other 0. The choice is arbitrary but must be consistent. Without loss of generality due to the symmetry of the cell, from this point onward logic 1 corresponds to the first state and logic 0 to the second state. Positive feedback forces the cell into one of the two states. The SRAM cell has three modes of operation: holding, writing and reading. The access transistors isolate the cell from the circuit during holding to preserve the written value, and allow access during writing and reading. Access transistors are controlled by the wordline (WL) signal. The other connections to the cell are the bit line signals (BL and $\overline{BL}$) which are connected to the internal nodes when the access transistors allow them. The bit line connection affects the internal nodes by introducing a small perturbation. This is useful for changing the state of the cell during the writing operation but must not be strong enough to change the state during the reading operation. These two modes of operation are examined now with more detail.

Reading

To perform the reading, both bit-lines are first driven to the high voltage value, V_{dd}. Then, the WL signal is activated. The process is illustrated in fig. 3.3 (b). One internal node will be at a low voltage and the other at a high voltage. The bitline connected to the node at low voltage discharges slightly, while the other bitline stays at roughly the same voltage. This difference is measured by a sense amplifier. The stored value is determined depending on which one of the bitlines has the higher voltage.

The internal node at low voltage rises its value as well when connected to the bitline. If this perturbation is small, the cell's feedback will be able to compensate once the reading is over. However, if it reaches a critical value it could alter the stored value and cause an error. The magnitude of this perturbation depends on the relative value between the width of the inverter's pull-down transistor (M_1 or M_3, depending on the state) and the width of the access transistors (M_5 and M_6 respectively).

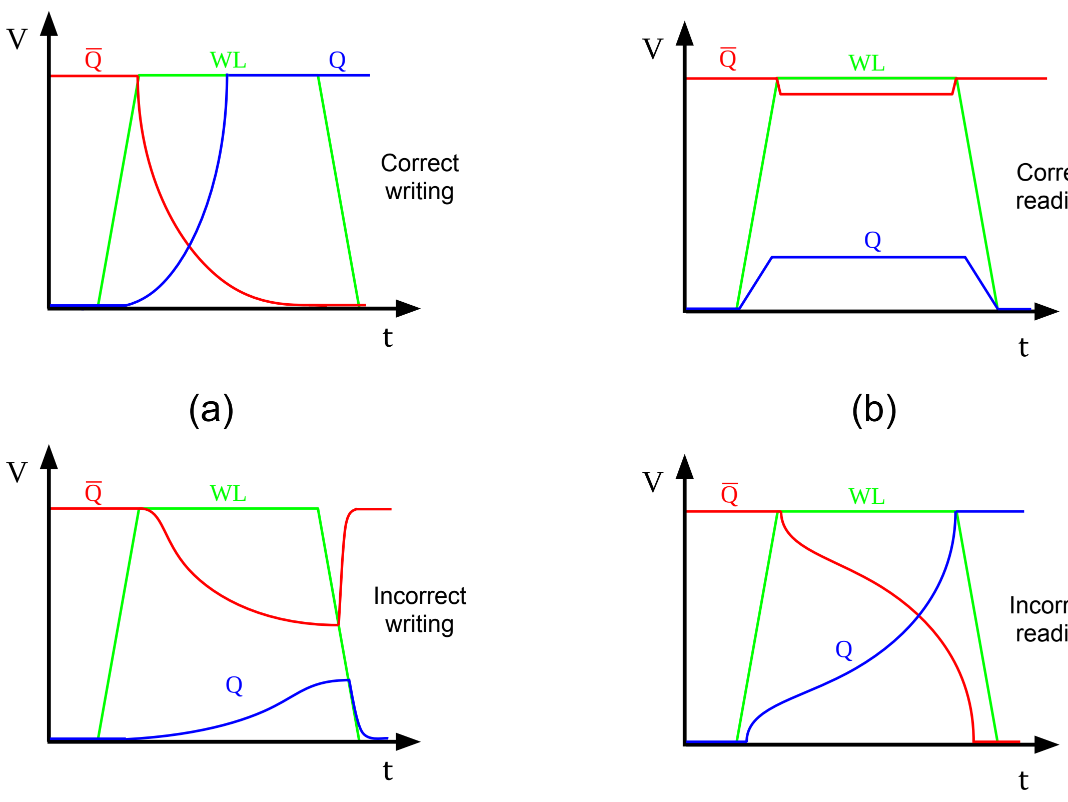

Figure 3.3: Voltage at the internal nodes and WL signal during a successful and unsuccessful writing (a) and reading (b) operation. The starting state is Q at low voltage and $\overline{Q}$ at high voltage, but the diagram would be similar for the other starting state.

Writing

This process is illustrated in fig. 3.3 (b). To perform the writing, each bit-line is driven to the desired voltage values of their respective internal nodes through the write drivers. The objective when writing is to change the state of the cell. To accomplish that, the node at low voltage has to increase to high voltage, and vice versa. The WL signal is then activated. The low voltage node increases its voltage at a rate which depends on the relative widths of the inverter's pull-up transistor

(M_2 or M_4, depending on the state) and the width of the access transistors (M_5 and M_6 respectively). The high voltage node decreases its voltage at a rate which depends on the relative widths of the inverter's pull-down transistor (M_1 or M_3) and the width of the access transistors (M_5 and M_6 respectively).

If the perturbance is not big enough, the voltage will not be flipped by the time the WL signal is deactivated. In this case, the nodes will go back to their original state and the writing operation will have failed. Accordingly, the widths of the transistors must be tuned to have a perturbation strong enough for writing and weak enough for reading.

3.3 SRAM as a PUF

An introduction to SRAM PUFs was provided in 2.3.4. Since SRAM PUFs are the focus on this work, a more detailed look into them is required. As explained before, SRAM PUFs exploit the mismatch due to fabrication process variability between the two inverters by measuring the start-up value of SRAM cells, determined by which one of the two inverters powers up faster.

Whether one inverter or the other powers up faster depends on the threshold voltage V_{th} of the different transistors. In order to simply illustrate this, we will consider that the strength of the cell depends only on the V_{th} values of the PMOS transistors M_2 and M_4 (using fig. 3.2 as reference). As an example, suppose that mismatch causes $|V_{th2}|$ to be smaller than $|V_{th4}|$. When the cell powers up, i.e. V_{dd} rises, M_2 will start conducting before M_4, causing $\overline{Q}$ to go up to high voltage and Q to low voltage, resulting in a power-up state of "0" by the criteria defined earlier.

Generalizing this example, if the difference between threshold voltages is $\Delta V_{th} = |V_{th2}| - |V_{th4}|$ the cell should power up to "1" for $\Delta V_{th} > 0$ and to "0" for $\Delta V_{th} < 0$. However, some cells (below 20%) with $|V_{th2}| \approx |V_{th4}|$ may change their power-up values when this is determined at different moments, due to operating temperature differences, supply voltage variations, aging or noise [24]. These are considered unstable and are largely responsible for the bit flips in the SRAM response. Most cells (above 80%) have a large enough difference to reliably power-up to the same state and are considered stable.

Figure 3.4 illustrates this if the distribution of the transistors' threshold voltage is assumed to be gaussian as in the mismatch models [39]. Cells with values in the middle region are considered unstable, shown in red. The bias does not have to be very strong to have a stable cell; so, generally, a majority of cells are considered stable, shown in light green.

Although only the mismatch of the PMOS transistors was considered regarding the power-up state of the cell, the mismatch of the NMOS transistors is important as well. The NMOS transistor that starts conducting faster pulls its node to low voltage, as they are connected to V_{ss}. Their contribution depends on how fast the SRAM powers up [72], i.e. the power supply ramp rate. This contribution ranges

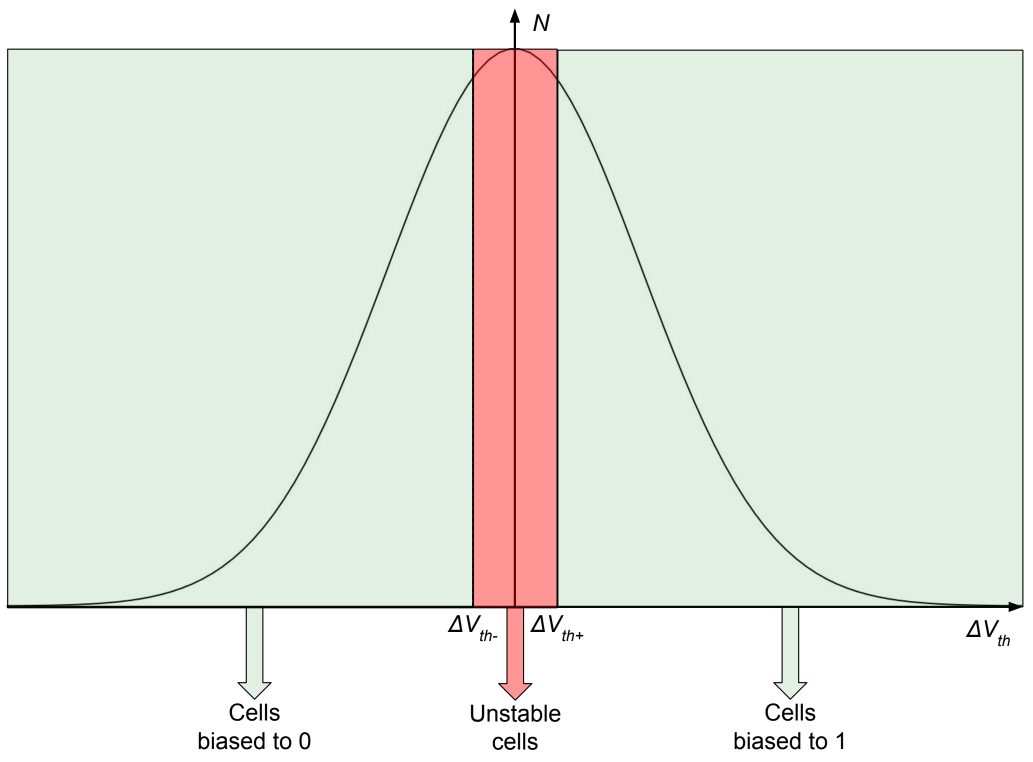

Figure 3.4: Number of cells N corresponding to a certain value of ΔV_{th}. Cells with $\Delta V_{th} \approx 0$ are unstable, while those to the left are biased to zero and those to the right to one.

from NMOS mismatch having almost no impact for near instant power-ups (1 ns) to NMOS mismatch having the same impact as PMOS for slow power-ups (1 s). In the second case, ΔV_{th} is calculated as $\Delta V_{th} = |V_{th2}| + |V_{th3}| - |V_{th4}| - |V_{th1}|$. In essence, depending on the difference in threshold voltages the cell will have more or less reliability working as a PUF. It is important to keep in mind that the stable cells still have some degree of unreliability over a large enough number of power-ups [44].

The standard solution for unstable cells in the context of PUFs used for key generation is employing helper data algorithms, but the requirements of heavy ECCs are quite costly as described in the previous chapter. Some proposed SRAM bit selection techniques which attempt to select only stable cells are evaluated in the next chapter. If the unstable cells are separated from the stable ones, they can still be useful for entropy generation [24].

Other approaches specific to SRAM PUFs have been studied to reduce this problem [24]. For example, in [16, 39, 9] they use a custom design of the SRAM cell. Specifically, in [16] write drivers and PMOS switches are added to the SRAM cell, in [39] circuitry to measure the threshold voltage is added so that only stable cells are used and in [9] the power supply of each inverter is routed independently to perform small independent variations and select stable cells based on their response to those variations and a Miller capacitor is added to increase stability. However, these methods are costly and renounce to one of the main advantages of SRAM PUFs, the ability to use generic SRAM cells already present in ICs.

3.4 Sources of reliability degradation

It is important as well to characterize how local temperature differences, supply voltage variations and aging affect PUF reliability. An overview is provided in the following subsections. From this understanding the reliability of the SRAM PUF can be properly measured and new techniques to boost reliability can be developed and tested.

3.4.1 Supply voltage

SRAM cells power up by rising their supply voltage from zero to V_{dd}. There are two different aspects concerning supply voltage and reliability.

On the one hand, possible deviations from the nominal value of supply voltage V_{dd}. This is usually tested by evaluating the circuit at supply voltages of $\pm 10\%$ of the nominal supply voltage [73]. However, the start-up value of the cell should be established during power-up and the supply level at the end of the ramp should not have a large impact on SRAM PUF reliability.

On the other hand, the ramp-up time has a pronounced effect on reliability [74, 72]. There is a wide range to choose from, from the order of μs to around one second. The issue is that faster ramps are better for high temperatures while slower ramps are better for low temperatures [75]. According to this, in [75] a proposal to dynamically change the ramp-up time is made. In any case, the effects vary according to the device design and a careful choice based on experimental data must be made to achieve optimal reliability.

3.4.2 Temperature

Temperature is quite influential on the performance of ICs. In SRAM PUFs it directly affects the characteristics of transistors, and, therefore, the PUF response [73]. Accordingly, it is of great importance to properly characterize its effects for reliability. This is done by measuring the behaviour of the chip in a certain temperature range. These ranges depend on the implementation. Although the specific range may vary, generally accepted ones are [76]:

- Commercial: 0°C to 70°C
- Industrial: −40°C to 85°C
- Military: −55°C to 125°C

Usually, PUFs are tested in the industrial range [70, 73, 75]. Depending on the SRAM design, the worst case reliability can be in either extreme of the temperature range [73].

3.4.3 Aging

As technology scaling continues in microelectronics, SRAM cells become smaller and time dependent degradation of electrical properties (aging) becomes more severe. Understanding aging is important to predict errors in any electronic implementation.

Aging can have a wide range of effects, from a change in threshold voltage, and the consequent increase in delay times, to permanent failure. Naturally, the change in threshold voltage is very important in SRAM PUFs since it may change the bias of one cell from one state to another during power-up. Aging depends in more variables than just time, being influenced by workload during operation, environmental conditions and electrical stress.

The aging mechanisms generally considered for SRAM cells are Bias Temperature Instability (BTI), Hot-Carrier Injection (HCI), Time-Dependent Dielectric Breakdown (TDDB) and Electromigration (EM) [77]. BTI, HCI and TDDB occur in the gate oxides of transistors while EM happens in the interconnect metal lines. BTI and HCI are the main concern regarding PUF reliability as they occur at nominal voltages. More studies have been performed for BTI than HCI but an exact model for either one has not yet been created [78]. In the subsequent subsections the effects of BTI and HCI will be considered qualitatively.

3.4.3.1 BTI

BTI includes NBTI (Negative-Bias Temperature Instability) effects, which occur in PMOS transistors, and PBTI (Positive-Bias Temperature Instability) effects, which occur in NMOS transistors. NBTI is generally dominant over PBTI for small technology nodes [77] and consequently NBTI aging is the main focus of study. NBTI increases the threshold voltage of the transistors and the consequent decrease in drain current and transconductance. It is caused by positive charges built up in the MOS gate insulator due to the application of negative gate bias. There is no consensus on a model for NBTI [79]. The Reaction-Diffusion model holds that NBTI is a diffusion-limited process and the Defect-Centric model holds that it is reaction-limited. According to the first one, interface traps are generated during operation and these interface states become positively charged when the PMOS device is biased in the ”on” state, i.e. with negative gate voltage. According to the second one, preexisting traps located in the bulk of the dielectric are filled with holes coming from the channel of PMOS.

Either the oxide electric field caused by negative gate voltages or elevated temperatures can produce NBTI, but a stronger and faster effect is produced by their combined action. Due to technology scaling, the magnitude of the electric field becomes greater, so NBTI is always going to be present and will be stronger as scaling keeps progressing [80]. This degradation phenomenon reveals a stochastic and discrete nature for technology nodes in the nanometer range, which means that identical transistors (and therefore circuits) may age differently [81]. The effects of NBTI have a permanent and a temporal component, since part of the degradation can be recovered over time when stress is removed (i.e. when the gate voltage is removed).

How does BTI affect an SRAM cell? The 6-T standard cell shown in fig. 3.2 is considered. Suppose it is biased so that Q is at the high voltage at power up. This corresponds to a state 1 according to the criteria expressed earlier. Accordingly, the left NMOS transistor has a smaller threshold voltage than its pair and/or the right PMOS transistor has a smaller threshold voltage than its pair. They will be the first to turn ON and will make the cell take the 1 logic value due to the feedback. If node Q is at a high voltage following the nomenclature of fig. 3.2 the NMOS transistor M_3 will be off and the PMOS transistor M_1 will be on and suffering PBTI. On the other hand, node Q will be at a low voltage, so PMOS transistor M_4 will be off and the NMOS transistor M_2 will be on and suffering PBTI. NBTI occurs in the PMOS transistor, so this transistor will be more affected. In any case, threshold voltage increases over time, consequently reducing the initial bias of the cell. The reasoning is the same if the cell is biased towards 0. If the cell is written to its non preferred value, BTI will instead increase the existing bias.

If the initial power-up bias of a cell is reduced over time, stable cells will turn unstable progressively. However, it can be solved by storing the non preferred value in the cell, which mitigates the stress caused during functional operation. This technique, called directed accelerated aging (DAA) [82], can be used to increase the initial bias and the resulting reliability of the cell. It can even be used to flip the initial skew of a cell and make it strongly biased to the other state. This is used in [83] to improve the uniformity of the PUF response as well.

3.4.3.2 HCI

HCI is a phenomenon by which the threshold voltage of a transistor is altered when high energy carriers are trapped in the gate oxide, as in BTI. However, HCI is caused by a flowing drain current, in which moving charge carriers (the hot carriers) break up the bonds between silicon and hydrogen at the boundary between the transistor channel and gate dielectric. The carriers need a certain kinetic energy to achieve this. When velocity saturates, electron's average velocity will not increase but the random kinetic energy of some electrons can achieve the required kinetic energy to break the silicon-oxide barrier due to random collisions. Once electrons start to break the barrier, a small leakage current appears and traps can occur.

The requirements for HCI to be present are high electric fields and long mean free path, i.e. the average distance travelled by an electron between collisions. As in BTI effects, technology scaling makes high electric fields common. Increasing the voltage will make degradation more pronounced. On the other hand, the mean free path for electrons to be accelerated is inversely proportional to temperature, so lower temperatures will increase HCI. In digital gates, it occurs only when the device switches, so the effect is proportional to the signal activity factor (ratio of average number of signal transitions to clock transitions and the clock frequency) [84]. The recoverable component of HCI degradation is very small compared to BTI

[85], so HCI has high permanence (does not lessen significantly over time).

Although BTI degradation is the most popular for DAA in the literature, in [85] HCI is used to boost reliability by applying stress through a V_{DS} increase. The advantages over BTI are shorter stress time, high permanence and requiring only a one-time reinforcement stress.

Chapter 4

A new method for bit selection in SRAM-PUFs

A naive approach to SRAM PUFs would be to choose a random selection of cells among the available cells. As pointed out in section 2.5, this would leave all the work to the ECC step and would come at a prohibitive cost considering the limited reliability of standard SRAM PUFs responses. There are a variety of criteria to choose the best cells in a SRAM. An ideal selection would improve the reliability of the response without being detrimental to uniqueness or uniformity. These cells should also provide a better response under the worst corners of performance, not just under nominal conditions.

In this chapter, some commonly used prevalent bit selection techniques are discussed. They serve as reference to compare the new bit selection technique, Maximum Trip Supply Voltage (MTSV), that tries to improve the existing techniques. Afterwards, this new technique is presented and validated with experimental data.

4.1 Multiple Evaluation

In Multiple Evaluation (ME) several power-up evaluations are performed on each cell, either to elaborate a response by majority vote [41], or to discard those cells that do not always power-up to a given value [24]. A key aspect of this method is that a fixed number of measurements must be selected: for instance, 20 measurements were chosen as a good trade-off in [24]. However, it is also shown in [24] that many cells that returned the same value for the first 20 power-ups, returned at least one "erroneous" value (i.e., their nonpreferred value) in the next 60 measurements, since 80 power-ups were performed to each cell. This is illustrated in fig. 4.1 for one of the chips used later in this work. In [41], no specific number of evaluations is set; however, the necessity of making this choice is clearly stated and the possibility of using from 10 to 100 evaluations is mentioned. Nevertheless, the appropriate number of evaluations depends on several factors, such as the technology node. Choosing well would thus require a tedious experimental study. In fact, this type of multiple evaluation may require a prohibitively high number of power-ups, thus delaying an on-line response, and significant hardware resources [44].

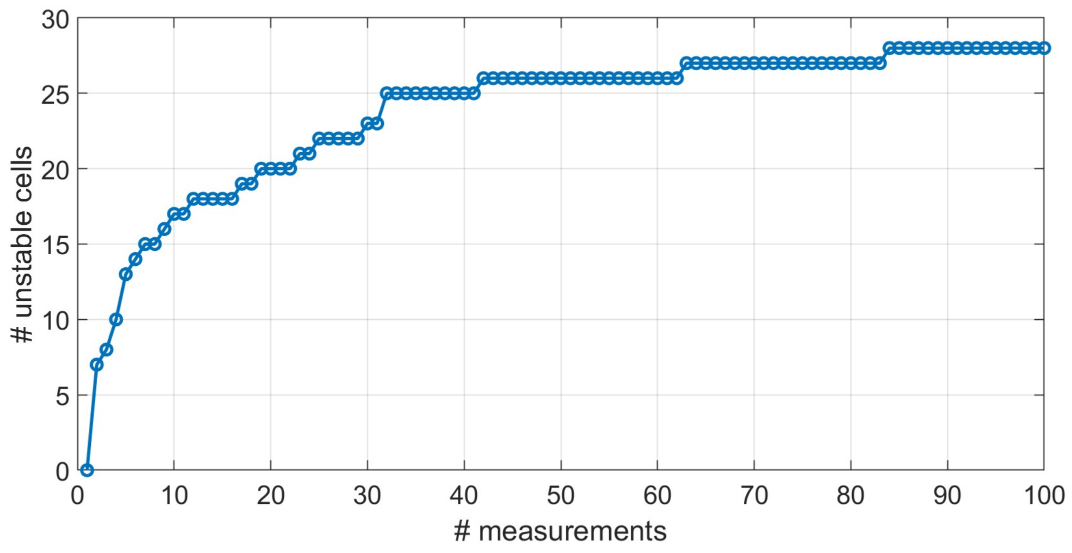

Figure 4.1: Number of unstable SRAM cells with respect to the number of power-ups for one of the chips.

4.2 Data remanence

The technique presented in [44] consists of two remanence tests: writing '1' (or '0') to the entire SRAM cell array and shutting down the power supply for a short time interval until a few cells flip. The technique exploits the fact that the cells that flip with shorter power-off durations have the strongest tendency towards the value that has not been written. This methodology has proven to be effective for experiments performed under different temperatures and power ramp times, and under device aging [44]. This approach has been tested in a design fabricated in an ultra-low leakage technology, which requires relatively long power-down times (in the order of hundreds of milliseconds) to observe those bit flips. However, much shorter data remanence times are expected for SRAMs built in advanced CMOS technologies (e.g., below microseconds [44]), which may turn this technique impractical to use on certain technologies. If power-on and power-off times in the order of microseconds have to be precisely controlled, it is clear that the ramp rates must be much faster than that. However, in [72], it is shown that the power-up states are especially sensitive to such factors like noise when ramp rates in the range of nanoseconds to microseconds are used. An option to overcome this issue is to operate these tests at low temperatures, at which longer remanence times are expected [86]. To this end, temperatures of tens of degrees Celsius below zero are necessary, which usually involve the utilization of liquid nitrogen as cooling method. This makes the method further unfeasible for practical applications. Moreover, to control the power-on and power-off times precisely, an additional microcontroller must be used. Therefore, this solution is not suitable for applications with resource-constrained devices.

4.3 Exploting the Power Supply Ramp Rate

In [72], a method to calibrate the strength of the SRAM cells is presented. Its authors claim that, if the SRAM is turned on by ramping the V_{DD} node from low to high at a rapid rate, the power-up of individual cells is decided entirely by the threshold voltage mismatch in the PMOS transistors in each cell. On the other hand, if the SRAM is powered up by rapidly ramping the V_{SS} node from high to low, the power-up state of each cell will depend on the threshold voltage mismatch between the NMOS transistors in the cell. Otherwise, if a slow ramp is used ('over several seconds' in [72]), both NMOS and PMOS pairs play a role in determining the state stored on the cell after the power-up.

From that theoretical starting point, the following method is proposed to identify the strongest cells: first, a fast V_{DD} ramp test is performed to investigate the preferred power-up state of each cell as determined only by its pair of PMOS core transistors. Then, an equivalent test is performed, this time ramping down the V_{SS} -node from high to low, to identify the preferred power-up state as determined only by its pair of NMOS core transistors. Considering both the preferences of the PMOS and NMOS transistors separately, there are four possible combinations ('0,0', '0,1', '1,0', '1,1'). Thus, around 50% of the cells are expected to power-up to the same state in both tests. By selecting these, a set of cells where both the NMOS and the PMOS transistor pairs individually bias the cell to the same power-up state is obtained. Finally, the strength of that set of cells is quantitatively calibrated. For that, the SRAM is written with a logic '1' (or '0') and then V_{DD} is slowly ramped down to a small voltage (e.g., 0.1V).

Then V_{DD} is ramped back again to its nominal value. This process is repeated while varying the lower limit of the supply voltage (e.g., 0.1V, 0.11V, 0.12V, etc.). If the initially written value was '1', cells that have '0' as their preferred value may flip to '0' during the above described process. While cells with a stronger tendency towards '0' may already flip their value at higher values of the decreased V_{DD} (e.g., 0.2V), cells with a weaker tendency will need a lower V_{DD} to flip their value (e.g., 0.1V).

However, there are some limitations to this technique. First, unlike other conventional methods to calibrate the strength of SRAM cells [41, 24, 44], it requires the realization of ramps in two different nodes of the SRAM cell (V_{DD} and V_{SS}). Second, it requires the utilization of two types of ramps, slow and fast, with ramping rates orders of magnitude apart. This translates into additional hardware resources if the method is to be implemented on-chip. Moreover, the utilization of fast ramps at both supply nodes is counterproductive. While fast ramps are performed in 1 nanosecond in the work presented in [72], it is also stated in that work that power-ups performed at such fast ramps are very sensitive to factors such as noise or device degradation. Therefore, the classification performed to select the potential candidates using ramps in the nanosecond scale may be incorrect. Finally, no silicon experimental results that validate the method under real operating conditions are presented in [72]: only simulation results using a Predictive Technology Model (PTM) for a 32-nm bulk technology, with estimated variability parameters, are provided. Moreover, background noise, supply voltage variations, circuit noise, temperature variations or circuit degradation due to aging effects were not considered, even though these are known to be critical factors in the performance of SRAM-based PUFs.

4.4 Maximum Trip Supply Voltage (MTSV) method

A new bit selection technique is now described. First, a description on how it works is provided. This technique is then tested experimentally against the most popular one available, multiple evaluation. The experimental setup used for these tests is described, and then the results are presented.

4.4.1 Description

To overcome some of the limitations that the above-discussed methods present, a new approach is described in this section. The idea behind this method, which will be referred here as MTSV method, is to evaluate the strength of a cell by writing its non-preferred value on it, then lowering V_{DD} for a given period of time, raising it again and checking if the cell has changed its value.

For the sake of illustration, two cells, *cell1* and *cell2*, which tend to go to ‘0’ but with different strength, will be considered. A ‘1’ is written in both cells and then V_{DD} is lowered and then raised (to its nominal value) repeatedly (with V_{DD} dropping to a lower value with each repetition) until the cell flips its power-up value to ‘0’. It is found that *cell1* flips at V_{DD1} while *cell2* does it at V_{DD2}. *Cell1* is said to have a stronger tendency to the ‘0’ state if $V_{DD1} > V_{DD2}$. Therefore, the idea is to classify all the cells in an SRAM by writing their least-preferred value on them, lowering their supply voltage to different values, and recording at which voltage value each cell starts flipping to its preferred value. This voltage value is their data retention voltage (DRV); strongest cells will thus be those with a higher DRV. Cells that, while lowering their V_{DD} at different values during the MTSV procedure, show a random behavior, for example returning ‘0’ and ‘1’ alternatively at different values of decreased power supply, do not have a strong power-up tendency. Therefore, they are labelled as unstable, no DRV value is assigned to them and do not even enter the power-up strength classification procedure.

While the Multiple Evaluation methodology introduced in the previous section may require a prohibitively high number of evaluations, the technique used in this work, on the other hand, requires a much lower number of iterations. For instance, if the range of supply voltages within which the cells experience a bit flip is bounded between 80 mV and 280 mV (as it is the case of this work), and assuming a step size of 10 mV for the evaluation, a total number of 21 steps is needed. Additionally, unlike [44], no precise timing of the power-off and power-on durations is needed, which becomes especially critical when these are in the range of nanoseconds to microseconds. Finally, unlike the method in [72], only one type of ramp in one of the nodes is needed. This is important in different aspects. First, only ramps at the SRAM V_{DD} node are performed (no ramp at the V_{SS} node is required). Second, only one ramp rate is needed, while in [72] two different ramp rates that are orders of magnitude apart were needed.

Furthermore, the ramp used in this work is in the order of a few milliseconds, which results in a reproducible PUF response, while the method in [72] used fast ramps in the range of nanoseconds, thus producing PUF responses that are very sensitive to factors such as noise or device aging. Finally, by not selecting the candidate cells only among those where both PMOS and NMOS transistor pairs produce a bias towards the same power-up state, potentially stable cells are not discarded, as it occurs in [72].

4.4.2 Chip

The circuit used in this work is an array of 832 SRAM cells, a photograph of which is depicted in Figure 4.2 [87]. These cells are distributed in 32 rows by 26 columns. The 6T SRAM cells have been sized with W = 80 nm and L = 60 nm for the access transistors and the PMOS pull-up transistors, and W = 160 nm and L = 60 nm for the NMOS pull-down transistors, following conventional sizing criteria. The chip has been designed so that individual access and control of every terminal of each SRAM cell is granted, as shown in fig. 4.3. This is necessary to properly perform the different tests on each SRAM cell, including normal operation tests, aging tests or simply switching off the cells.

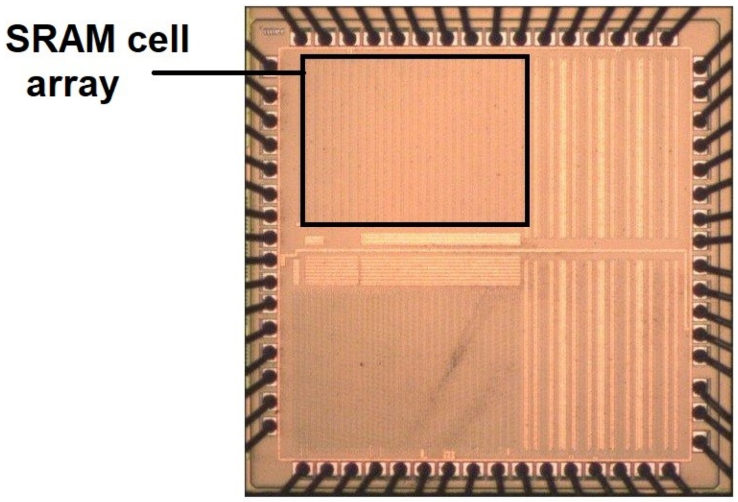

Figure 4.2: Photograph of the chip used in this work, with the part corresponding to the array of SRAM cells highlighted.

For the aging tests, as the degradation of the SRAM operation will extend along years, physical characterization is not possible under normal operation conditions. Therefore, accelerated aging conditions, e.g., by using higher operating voltages on SRAM terminals and/or higher operating temperatures, are applied during certain periods of time, in the range of seconds to hours. Some SRAM figure-of-merit must be measured before and after the stress cycle to compute its degradation.

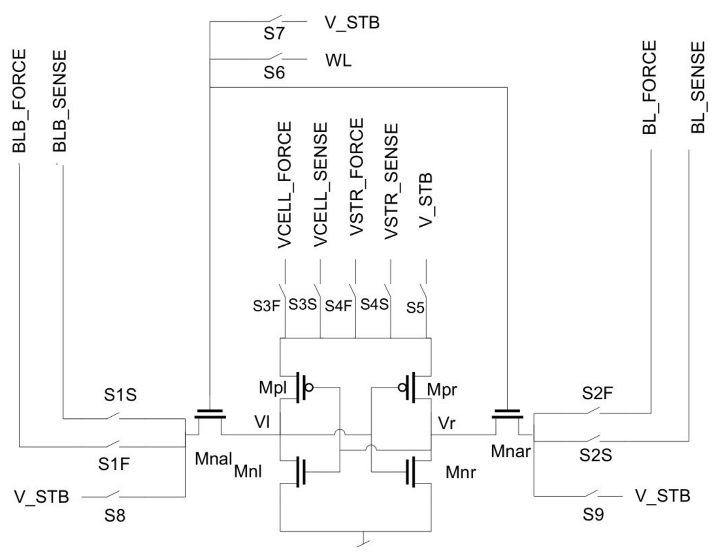

Figure 4.3: SRAM cell with transmission gates for appropriate terminal access and control.

It must be noticed that, as explained in section 3.4.3, in some aging phenomena, e.g., BTI, the stressed devices start recovering their original electrical parameters, once the stressed condition is removed. Therefore, accurate timing of stress and measurement stages must be established. Because of the stochastic nature of aging phenomena, stress and characterization tests must be performed over a large number of devices. The fact that a large number of cells have to be stressed during relatively long times implies a characterization bottleneck if the process is performed serially, cell by cell, especially if the recovery period has to be measured since cells are read-out serially. For this reason, each cell included digital control circuitry that allows parallelization of stress of several cells, while read-out of another cell is performed, strongly reducing the time of these experiments. As an example, if accelerated aging conditions are applied for 10,000 s, it would take around 97 days to characterize all 832 cells serially and around 28 hours to do so parallelly.

Force and Sense (F&S) techniques are needed for accurate measurements. Therefore, independent F&S paths are required to access those terminals where current is flowing and a voltage drop can occur.

The array includes row and column decoders to individually select each SRAM cell. Digital control circuitry is included in each cell as well as access circuitry (transmission gates, TG) that connect each terminal of each cell to output pads, as shown in Figure 4.3. Four control bits are used to control the TG´s at each node of the SRAM cell, defining 4 different operation modes for each cell:

1. **Measure mode:** This operation mode has been designed to connect the word line and bit line terminals to the word-line and bit-line analog paths, respectively, and the cell power supply terminal to the standard power supply path. In this operation mode, the selected cell can undergo any write or read operation, or hold some data. Voltages can be fully controlled so that any

SRAM figure-of-merit can be properly measured.

2. **Stress hold mode:** This operation mode has been designed to connect the power supply terminal of the SRAM to the stress power supply path. As shown in Fig. 3, this path is independent of the standard power supply. This feature enables the parallelization of the stress hold mode of one SRAM cell together with other operations in other cells.

3. **Stress AC mode:** This mode has been designed to perform write or read operation under stress conditions, up to 3.3V, applied to the word line, bit lines and power supply terminals.

4. **Stand-by mode:** This operation mode connects bit line, word line and power supply terminals to the stand-by analog paths (VSB). This operation mode has been designed to keep all cell terminals at 0V, and therefore under no aging process.

While this modified SRAM cell allows for accurate accelerated aging tests, its characteristics are not different in any way from the standard 6T SRAM cell considered in chapter 3. The experimental data and conclusions obtained from it can be generalized for a generic SRAM PUF. The MTSV technique itself does not need this particular design. Further information about the circuit design can be found in [87].

4.4.3 Experimental setup

The main components of the experimental setup used in this work are depicted in Figure 4.4. A full-custom Printed Circuit Board (PCB) has been used for the different tests, together with a power supply generator for the biasing of the PCB and the chip. A Keysight B1500 semiconductor parameter analyzer equipped with 4 High Resolution Sense Measurement Units (HRSMU) has been used for voltage application and measurement. The HRSMUs have a F&S system that, together with the F&S connection implemented in the circuit, allows the accurate application of voltages by avoiding undesirable voltage drops in the chip. The temperature tests have been performed using an ACS climatic chamber, which allows the stable and homogeneous application of temperature on the chip.

4.4.4 MTSV experimental performance

The adequacy of the MTSV method has been tested under four different scenarios: at nominal conditions, after the circuit has been subject to accelerated aging, under temperature variations, and under supply voltage variations. The metrics used to evaluate the quality of the responses are the ones presented in sec. 2.4. The values obtained without performing bit selection are the reference point to assess the effectiveness of ME and MTSV. The ramps used in this work are in the order of a few milliseconds, where the power-up state is expected to be scarcely impacted by factors such as noise [72].

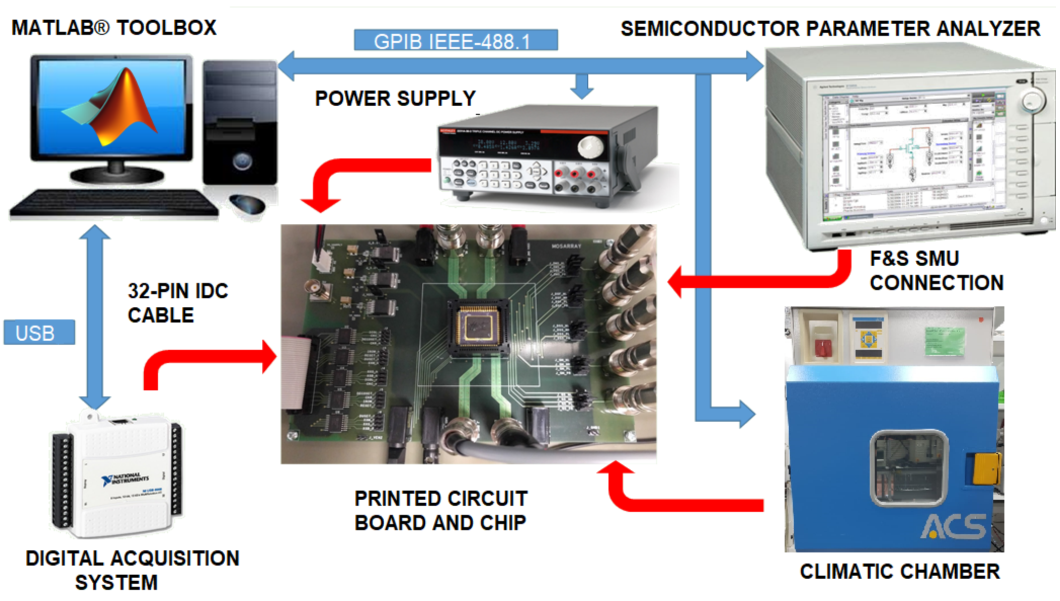

Figure 4.4: Representation of the experimental setup used in this work.

4.4.4.1 Nominal conditions

The first step is to measure the response of the chip under nominal conditions of temperature and supply voltage. Five different chips are used during these measurements to ensure that the results are valid under inter-chip variations. Since the inter hamming distance is measured across all chips, it is a good starting point. The result is obtained using eq. 2.4.4:

$$\text{mean}_{\text{InterHD}} = 49.9398\% \tag{4.1}$$

This result is good, close to the ideal 50%. It confirms that there is no relevant correlation between the power-up behavior of different chips' cells, and therefore uniqueness is expected from the different PUFs' responses.

The next step is to classify the cells in terms of their power-up response following the MTSV method. Figure 4.5 shows a flow diagram of how the classification of each cell is done, according to its power-up strength: first, its non-preferred power-up value was written on it. Then, its supply voltage was decreased to different values ranging from 280 mV to 80 mV and the content of the cell was read at each step. Cells that showed a random behavior, for example returning '0' and '1' alternatively at different values of decreased power supply, were labelled as "unstable". Among the five chips, the number of unstable cells ranged between 40 and 50 out of the 832 cells of each chip. Interestingly, these cells correspond roughly to the cells labelled as unstable by the Multiple Evaluation method when a large enough number of power-ups is performed (see Figure 4.1). Therefore, cells that did not show a random behavior during the MTSV procedure, that is, cells that flipped their content for one given VDD (their DRV value) and continued flipping their content for any VDD below their DRV value, correspond to the cells labelled as completely stable (0% erroneous power-ups) by the Multiple Evaluation procedure. These cells were assigned that

DRV value and were ordered from strongest (highest DRV value) to weakest (lowest DRV value). Note that the Multiple Evaluation method is not able to perform this classification to the cells it labelled as stable, since they all powered up to the same value at each evaluation.

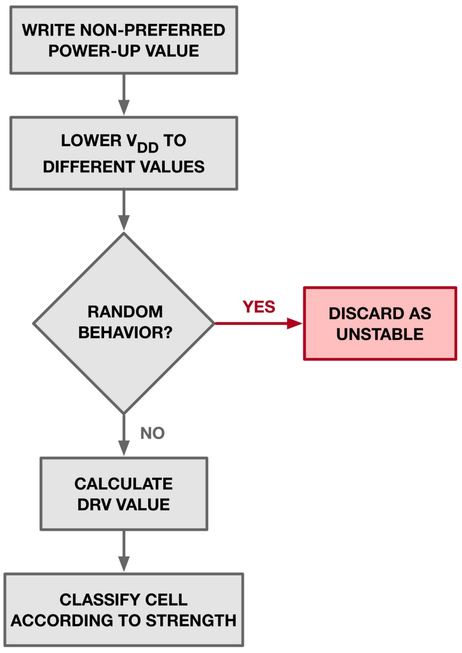
WRITE NON-PREFERRED POWER-UP VALUE
LOWER V_{DD} TO DIFFERENT VALUES
RANDOM BEHAVIOR?
YES
DISCARD AS UNSTABLE
NO
CALCULATE DRV VALUE
CLASSIFY CELL ACCORDING TO STRENGTH

Figure 4.5: Flow diagram of the procedure used to classify each cell according to its power-up strength.

In order to evaluate the method, the 50 strongest and the 50 weakest cells of each chip were selected, so that their strength could be compared between them and also to the rest of the array. In the following, “weakest cells” refers to the weakest cells among the ones that were not labelled as “unstable” during this first phase of the MTSV procedure.

It must be noted here that selecting 50 cells has been an arbitrary decision, since a larger or a smaller selection would serve as well for the purpose of this experimental validation. The goal here is not to create a specific PUF instance, but to prove that the MTSV method allows the selection of the strongest and more robust SRAM cells.

Figure 4.6 shows a histogram of the DRV values for the SRAM cells that display a tendency to power-up to one of the possible values. The histogram corresponds to an MTSV experiment with V_{DD} step size of 5 mV, performed on 832 cells of one of the chips. The strongest cells correspond to the right tail of the histogram, i.e., those with a higher DRV value, while the weakest cells correspond to the left tail of the histogram. Unstable cells do not appear in the histogram, since no DRV value could be assigned to them.

Then, to evaluate the adequacy of the MTSV-based classification, 2,500 power-up evaluations are performed for the 50 strongest and the 50 weakest cells of each chip, and 200 power-ups for the rest of cells. The number of power-ups applied

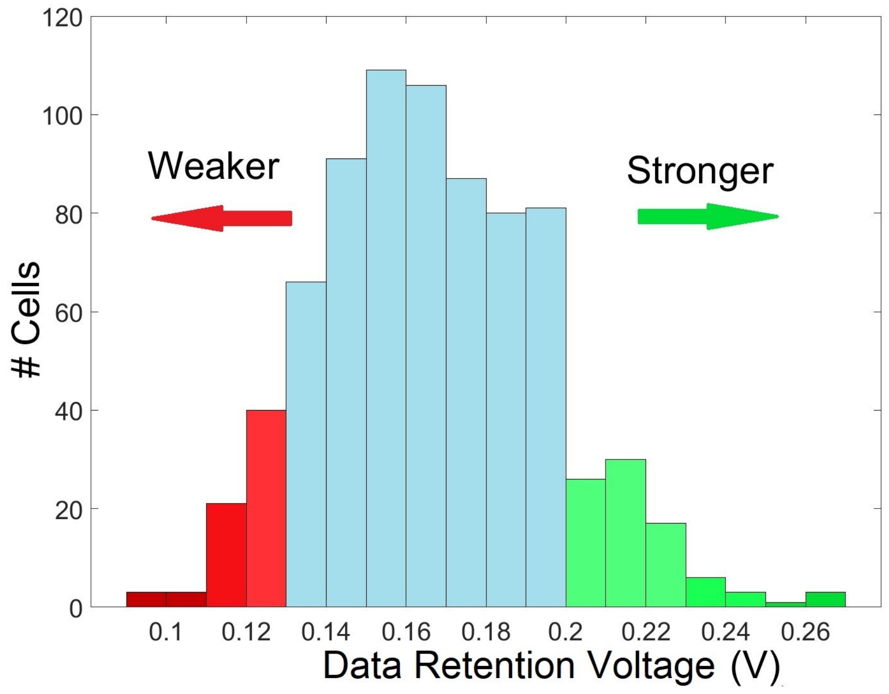

Figure 4.6: Histogram of the V_{DD} values at which cells (in one of the chips) start flipping from their non-favorite to their favorite value.

to the strongest and weakest cells is higher than the one commonly used in the Multiple Evaluation method, but for the purpose of the comparison made here, this number will provide a much more precise evaluation of the cell strength. Table 4.1 contains information about the mean BER obtained for the chips considering i) all the cells, ii) only the unstable ones, iii) the 50 strongest ones and iv) the 50 weakest ones. It is important to remember that "weakest" refers to the weakest cells once the unstable cells according to the MTSV method have been discarded. That is why the 50 weakest cells perform better than the complete array. The sets of strongest and weakest cells display proportionally fewer errors than the complete array or the set of unstable cells, so a larger number of evaluations was necessary to achieve the adequate accuracy. These results are displayed in Figure 4.7 in a more visual manner. Notice that, for a clearer visualization, Figure 4.7 has been plotted in logarithmic scale due to the large disparity of BER values, which span across several orders of magnitude. This occurs because cells classified as the strongest by the MTSV method have extremely low (i.e., good) BER values.

	All cells	Unstable cells	50 weakest cells	50 strongest cells
BER (%)	0.5265	8.8556	0.0765	0.0025

Table 4.1: Mean value of BER for the five chips

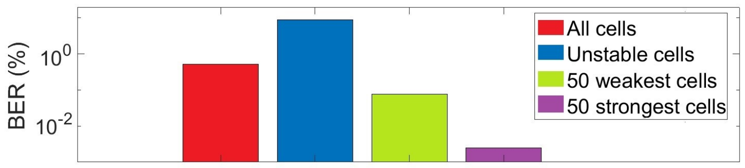

Figure 4.7: Average BER of the five chips taking into account all cells, only unstable cells, the 50 weakest cells and the 50 strongest cells.

The BER results indicate that the cells labelled as unstable during the first step of the MTSV method are those with a larger instability, and that, by just applying the first part of the MTSV procedure and discarding these cells, the reliability of the PUF is largely increased. Once the unstable cells have been discarded, the remaining cells are classified according to their DRV value, and the strongest ones are selected. Table 4.1 and Figure 4.7 show that this further improves the reliability of the PUF, since the 50 strongest cells according to their DRV have $\sim$ 30 times less power-up errors than the 50 weakest ones, and $\sim$ 210 times less power-up errors than when no classification is done and the whole array is considered.

As a final remark to highlight the strength of the cells selected by the MTSV method, a 0.0025% BER means that, in average, the strongest cells yield an erroneous power-up only once out of 40,000 times. It becomes thus evident that a typical Multiple Evaluation test with some tens of power-ups, like those in [41, 24], would not be able to achieve such a selection. This is the reason for choosing a very high number of power-up evaluations to test the cells that were not discarded during the first part of the MTSV test. In the following subsections, the capacity of the MTSV method to select strong cells in the presence of circuit aging and temperature and supply voltage variations will be evaluated. For this, the focus will be set on the strongest and weakest cells, since it has already been shown at nominal conditions that the cells labelled as unstable by the MTSV method are not suitable to be used in PUF identifiers.

4.4.4.2 Resilience to circuit aging

The next step was to evaluate the adequacy of the MTSV method to select aging-resilient cells. For this, the 100 selected cells (50 strongest and 50 weakest) of one of the chips (chip #1) were stressed (i.e., aged in an accelerated manner) to reinforce their non-preferred value. It must be clarified that this does not aim at emulating a real-case operation scenario, since SRAMs will age in different manners depending on the application they are used for. Therefore, the goal of this test is to represent a worst-case of aging degradation, in which its impact systematically reduces the cell power-up strength. To achieve this, their preferred value was written on each cell, and then the supply voltage was raised to 2.5 V during 10,000 s. As mentioned in chapter 3, the BTI aging suffered by an SRAM cell that stores a given value (e.g., '0') drives the preferred power-up value of that cell towards the opposite value (e.g., '1') [41]. This aging is V_{GS}-activated, and can be accelerated by increasing this voltage over its nominal value (1.2V for the technology used in this work). This is achieved by increasing the supply voltage of the cell. After this high voltage is removed, there are both a permanent and a recoverable component of degradation. For this reason, several measurements have been performed to investigate the resilience of the cells against degradation: one right after the stress ended, and three additional measurements 3 days, 10 days and 40 days later, to observe the evolution of the power-up behavior during the BTI recovery. The experiment performed right after the end of the stress consists of only 150 power-ups so that the recovery of the degradation throughout the test is not very significant. The other experiments, performed some days/weeks after the end of the stress, consist of 2,500 power-ups for each cell. The results obtained for each test were compared to the golden response obtained at nominal conditions for that chip before the stress was applied. The results for these experiments are collected in Table 4.2.

Time after stress	Right after stress	3 days	10 days	40 days
BER (%) 50 strongest	0	0	0.0016	0.0008
BER (%) 50 weakest	28.1200	21.0424	19.2640	18.5720

Table 4.2: Average BER of the 50 strongest and 50 weakest cells of chip #1 after accelerated aging

In this case, the difference in resilience against accelerated aging between the cells classified as strong and weak by the MTSV method is quite revealing. The 50 strongest cells according to that method remain unaffected by aging in terms of their power-up response, having a BER of 0 (which means that there was no power-up different to the one in the golden response) or very close to 0 in each experiment. On the other hand, the BER of the 50 weakest cells is much larger than the values obtained at nominal conditions before the stress was applied. Before the stress, this value for all chips was 0.08% (see Table 4.1), while now it is 28% right after the stress, and it slowly recovers down to 19%. This large increase in BER for the cells classified as weakest by the MTSV method, while those classified as strong maintain a very low BER, indicates that the MTSV method has correctly selected the cells that are more resilient to circuit aging in terms of power-up response. Notice that, according to the Multiple Evaluation method, both sets of weakest and strongest cells had been labelled as equally strong, since they both had 0% erroneous power-ups at nominal conditions. This demonstrates how the MTSV is able to foresee, before the cells have suffered any type of aging, which ones are going to be more robust against it, and therefore proves to be outstanding when selecting cells that are resilient to aging.

To gain insight into how the strength of the cells has changed after the stress, the DRV values measured 10 days after application of the stress are depicted in Figure 4.8. It can be seen how, although the average DRV of the 50 strongest cells, and with it their strength, decreases slightly after the application of the stress, it is still much larger than the fresh DRV of the 50 weakest cells. This explains the fact that, even after the stress, the cells classified as strongest by the technique proposed in this work still have a strong tendency towards the same value when powered up. On the other hand, the case of the 50 cells classified as weakest is very different. Only 33 out of the 50 weakest cells remain stable after the stress, while the remaining 17 cells lost their tendency towards a given value and, therefore, no DRV could be assigned to them. For this reason, only those 33 weakest cells are shown in the histogram at the bottom of Figure 4.8. As it can be seen in that histogram, there was also a decrease in their average DRV of a few milivolts. To shed some light on the behavior of the recoverable component of the degradation, the BER value measured in each of the tests shown in Table 4.2 is plotted against the time elapsed from the stress in Figure 4.9. Interestingly enough, the expected logarithmic recovery of degradation at device level (i.e., of the transistors' threshold voltage) [88], translates into a logarithmic recovery of BER after the application of the stress.

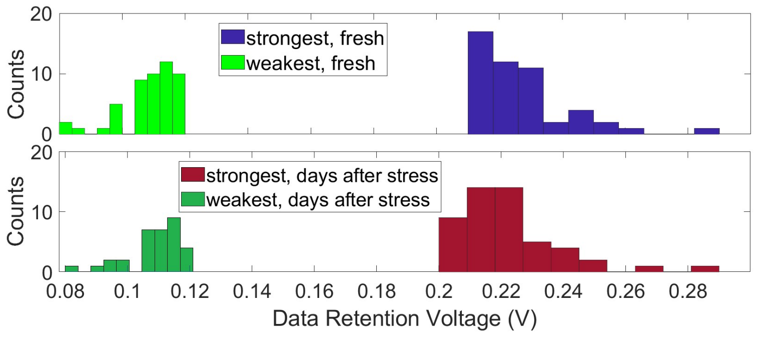

Figure 4.8: DRV value for the 50 strongest and 50 weakest cells measured before the stress (top), and 10 days after stress removal (bottom).

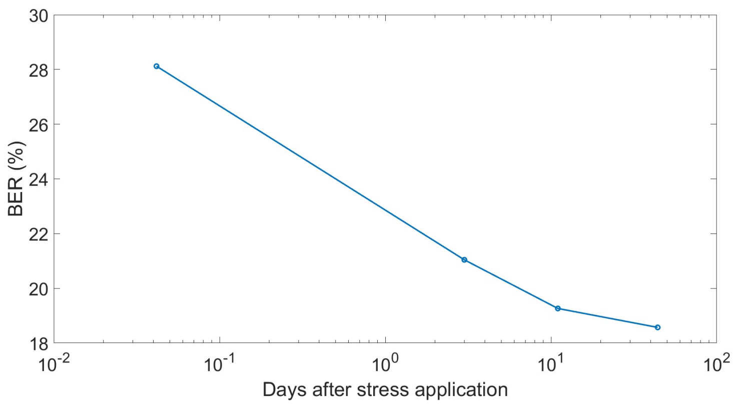

Figure 4.9: BER evolution for the 50 weakest cells of a chip with respect to the number of days since the application of the stress.

4.4.4.3 Resilience to temperature variation

To test the adequacy of the MTSV method under temperature variations, the 50 strongest and the 50 weakest cells of another chip (chip #2) were selected under nominal conditions. Then, 2,500 power-ups were performed for each cell at 0 °C, 10 °C, 20 °C, 30 °C and 40 °C. The power-up response of the cells obtained under the different temperatures was evaluated using as a reference the golden response of that same chip obtained at room temperature, i.e., 25 °C. The BER results are included in Table 4.3.

Again, the 50 strongest cells according to the MTSV method prove to be very resilient against environmental variations, in this case against temperature variations ranging from 0 °C to 40 °C. In fact, the worst BER obtained for the 50 strongest cells is 0.0024%, which is even slightly lower than the average BER for the 50 strongest cells obtained at nominal conditions of temperature and supply voltage, which was 0.0025%. The 0.0024% BER obtained in the worst case for temperature variations translates into roughly 1 erroneous power-up out of every 42,000 evaluations. On the other hand, the performance of the cells classified as weakest by the MTSV method worsens when temperature variations are considered. For instance, the BER increases more than 10 times at both 0 °C and 40 °C compared to the one found at 20 °C. The worst BER result for the 50 weakest cells is 1.7592% at 0 °C. This BER value corresponds to roughly 1 erroneous power-up out of every 57. This error rate is 700 times larger than that of the strongest cells. Similar to the circuit aging scenario discussed in the previous subsection, the results presented in Table 4.3 prove that the MTSV method performed at nominal conditions has been able to correctly select the SRAM cells with a power-up behavior that better tolerate temperature variations.

Temperature	0 °C	10 °C	20 °C	30 °C	40 °C
BER (%) 50 strongest	0	0.0024	0.0008	0.0024	0.0008
BER (%) 50 weakest	1.7592	0.4080	0.1256	0.2288	1.2128

Table 4.3: Average BER of the 50 strongest and 50 weakest cells of chip #2 tested under temperature variations

4.4.4.4 Resilience to Supply Voltage Variations

The final test that has been performed was to test the adequacy of the MTSV method to select cells with a power-up response that is reproducible when supply voltage variations are introduced, since this type of fluctuations may be present in real application scenarios. For this, the 50 strongest and 50 weakest cells of one of the chips (chip #2) according to the MTSV classification were powered-up 2,500 times each at room temperature and different supply voltages, i.e., raising their supply voltage from ground (0 V) to 1.08 V, 1.14 V, 1.2 V, 1.26 V and 1.32 V. This corresponds to a range of the nominal voltage 1.2 V ± 10%. As in the previous tests, the results were compared to the golden response of the chip measured at nominal temperature and supply voltage conditions. The BER values obtained for this test are displayed in Table 4.4. The worst BER value for the 50 strongest cells is obtained at $V_{DD} = 1.2$ V and is 0.0048%, which corresponds roughly to one erroneous power-up out of every 21,000 power-ups. For the 50 weakest cells, the worst BER value is obtained at $V_{DD} = 1.32$ V and is 0.204%, equivalent to one erroneous power-up out of every 490, which corresponds to an increase in the error rate of 70% compared to the one obtained at nominal supply voltage conditions. As in the previous tests, these results make clear that the MTSV classification performed at nominal temperature and supply voltage conditions is able to select cells that will have an extremely reproducible power-up behavior even under supply voltage variations

Supply voltage	1.08 V	1.14 V	1.20 V	1.26 V	1.32 V
BER (%) 50 strongest	0.0024	0.0016	0.0048	0.0008	0.0008
BER (%) 50 weakest	0.1160	0.1208	0.1216	0.1312	0.2040

Table 4.4: Average BER of the 50 strongest and 50 weakest cells of chip #2 tested under supply voltage variations

Chapter 5

PUF experimental implementation

After explaining the fundamental features of silicon PUFs, describing SRAM PUFs and presenting a new bit selection technique, this chapter is focused on presenting a real and complete PUF response. The objective is to verify the assertions made so far by turning the PUF with an unreliable response into a reliable key generator, using MTSV bit selection to alleviate the requirements of the ECC circuitry.

The chip design and the experimental setup used in this chapter are the ones described in subsections 4.4.2 and 4.4.3. In this case, only one chip (chip #6) is characterized. Instead of evaluating the 50 strongest and weakest cells according to MTSV, all the 832 cells of the chip will be measured.

First, different bit selections are considered to be used as a comparison with MTSV-based bit selection. Then, these selections are tested under nominal conditions, as well as under supply voltage variations, temperature variations and accelerated aging. An appropriate ECC will be selected based on the data for each one, and the fuzzy extractor construction will be implemented under different scenarios. Finally, a discussion on the exact ECC requirements for a KER of 10^{-4} % based on the measured BER values is presented.

5.1 Bit selections

The first step is to decide the key-length. It is important to remember that our chip has 832 cells to select from, while commercial SRAMs will generally have many more [73]. A length of 128 bits is the bare minimum for cryptographic algorithms, as explained in sec. 2.2.2, and it will be the benchmark used.

Next, the required BER out of each selection must be calculated. It is tied to the target for KER, which was set at lower than 10^{-4} % in sec. 2.5. The KER is understood as the probability of obtaining an erroneous key, which translates to the probability of an error in one or more bits. This probability can be derived by evaluating the binomial distribution (eq.2.1) term by term and performing approximations.

Naturally, $P(x = k) = 10^{-4}$ %. Using the approximation found in eq. 2.5, the probability of failure of each bit p can be approximated to the average BER of the response ($p \approx$ BER) [46]. This is a naive approach, since error rate will not be homogeneous across all cells [6]. For example, if one cell had 100% BER but the rest had 0%, the average BER of a 128-bit string would be 0.78% according to eq. 2.6. The failure rate would be 100% since one cell always fails, but it would not appear so by using the overall BER. However, it is an extreme case and in most cases this approach gives a good first estimation on the real KER.

The probability p must be low to meet the strict KER requirement, so presumably $p << 1$ and $(1-p) \approx 1$. k, equivalent the number of failures, can be taken as 1 as the probability of two mistakes is negligible for $p << 1$. In this case, the binomial coefficient is simply $\binom{n}{1} = n$. This results in a target of BER $\approx p = n^{-1}10^{-4}$ % i.e. $7.8 \cdot 10^{-7}$ % for a 128-bit key.

Once the length of the selection and the desired BER are established in an ideal and theoretical way, the next subsections explain the different selections used during the experimental tests.

5.1.1 Arbitrary selection

An arbitrary selection corresponds to a selection that does not follow any reliability criteria. It is meant to serve as a point of reference to compare the other bit selection techniques, and how much the required ECC is reduced by them. They should generally have a reliability close to the global reliability of all cells. Two arbitrary selections are considered: *First* and *Random*. *First* simply takes the first 128 cells of the SRAM array, which would be the fastest to read. *Random* takes a random selection of 128 cells out of the 832 ones available.

5.1.2 Multiple Evaluation selection

Multiple evaluation is the most popular bit selection technique in the bibliography [6, 24, 41], so it provides an appropriate comparison for MTSV bit selection. 20 power-ups were used for this selection as suggested by [24]. With these power-ups, 21 cells are classified as unstable, leaving 811 to choose from. Fig. 5.1 shows how the number of unstable cells increases with each power up.

This technique does not provide a classification of the available 811 "stable" cells. A random selection of 128 cells among these 811 ones is taken, *ME Stable*. Fig. 5.2 shows which cells in the array are considered unstable in red and which ones are chosen for *ME Stable* in green.

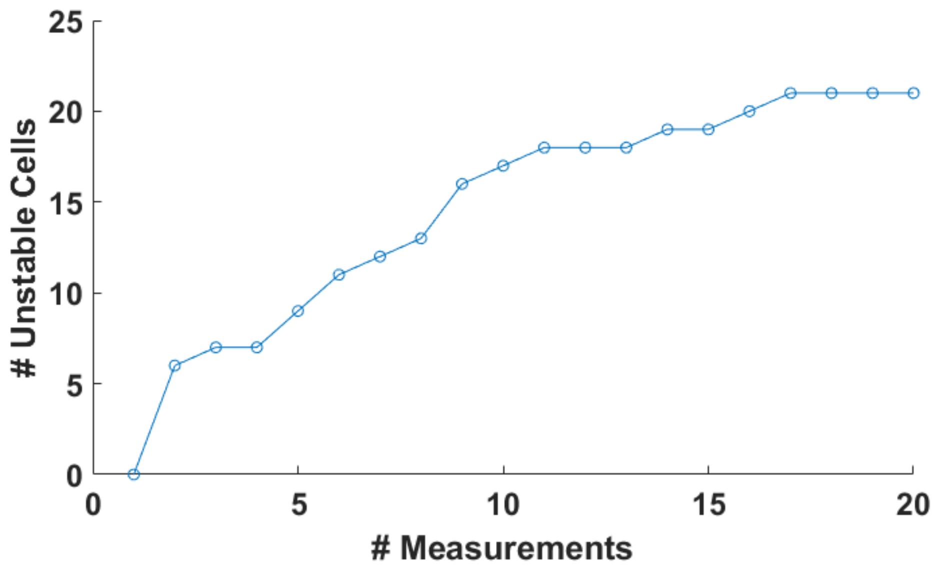

Figure 5.1: Number of unstable SRAM cells with respect to the number of power-ups.

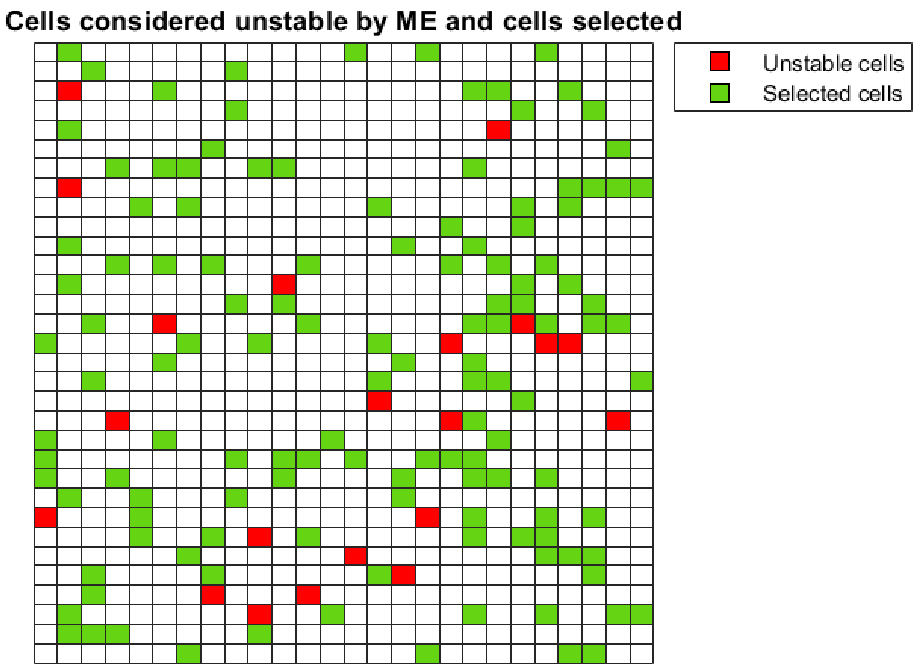

Figure 5.2: Cell-wise representation of the bit selection performed through ME. Cells in red are those considered unstable and cells in green are those selected randomly among the rest.

5.1.3 Maximum trip supply voltage selection

MTSV selection is explored in detail in section 4.4. First, cells that are labelled as unstable due to random behavior during characterization are discarded. These cells are shown in red in fig. 5.3. There are 48 in total, more than double compared to ME. Of the 21 cells labelled as unstable by ME, 17 are also labelled as unstable by MTSV (shown in black), i.e. 80%. The 4 cells labelled as unstable by ME but not by MTSV (shown in blue) are assigned a low DRV value compared to the rest, so they will be considered weak cells and not form part of any reasonably sized selection. In this way, MTSV discards the cells that ME considers unstable and expands on that considerably.

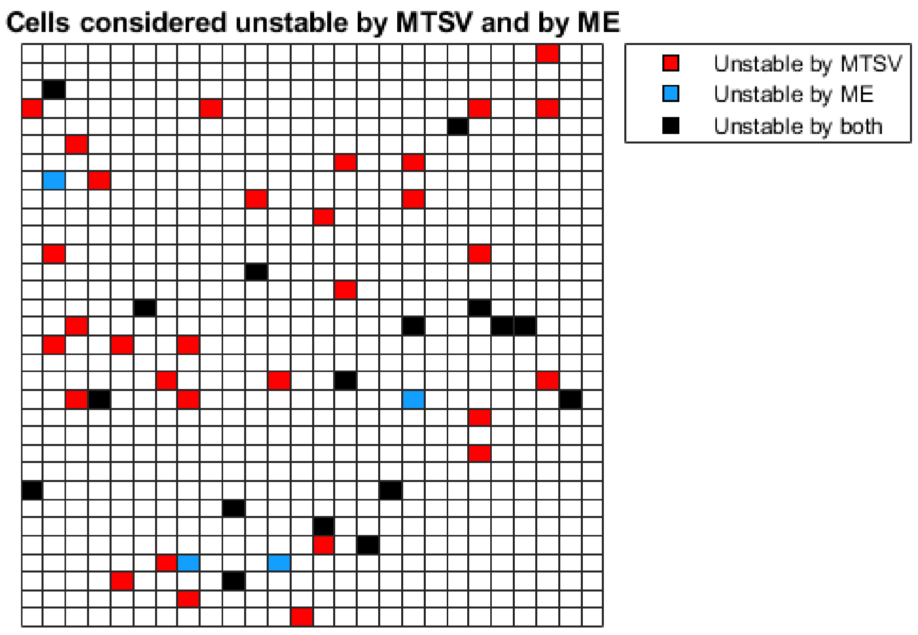

Figure 5.3: Cell-wise representation of the cells classified as unstable by ME or MTSV.

Then, cells are classified according to their DRV value. The 128 cells with the highest value are selected, which results in the selection *MTSV Strongest*. Another possible scheme to have perfect uniformity is to select the 64 strongest cells biased to “1” and the 64 strongest cells biased to “0” since the preferred value of each cell is evaluated at enrollment. This selection is called *MTSV Balanced*. The bits selected by either one are shown in fig 5.4. Cells in red are those considered unstable, cells in blue correspond to *MTSV Strongest*, cells in purple to *MTSV Balanced* and cells in green to both selection. The selections are very similar, differing only in about 10% of cells.

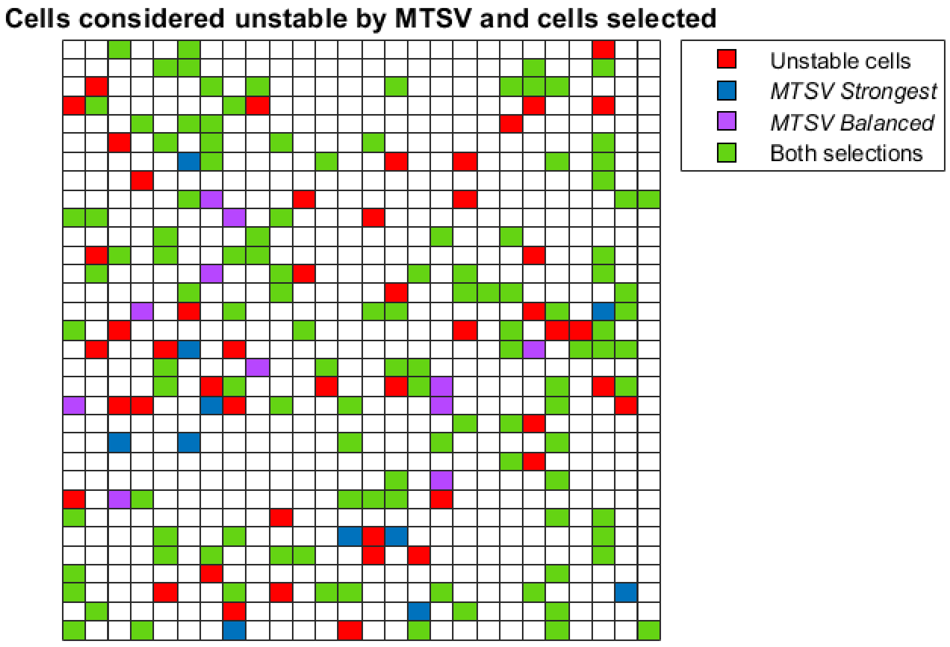

Figure 5.4: Cell-wise representation of the bit selection performed through MTSV.

5.2 Nominal conditions

Nominal conditions mean room temperature around 25 °C and a supply voltage of 1.2 V. Multiple tests with a varying number of power ups for all cells were performed. The results can be found in Table 5.1. BER and MAXintraHD are calculated intrinsically, i.e. using each dataset as its own reference.

Measurements	BER(%)	MAXintraHD(%)	Hamming Weight(%)
200	0.4140	1.0830	50.83
200	0.5102	1.0830	50.94
200	0.4693	1.2034	50.84
1000	0.4716	1.0830	50.94
2000	0.5221	1.4440	51.03

Table 5.1: Performance of all cells under nominal conditions.

BER varies between a value of 0.4140% and 0.5221%. This is for multiple tests and stays in a reduced bracket of values for a statistical metric. These values for nominal conditions are considerably low compared to other SRAM PUFs. In [73], seven different commercial and research SRAM designs are tested under nominal conditions as well as temperature and supply voltage variations. BER values go between 2.8% and 4.8% for nominal conditions in these tests. However, our results are still far away from the desired reliability.

Maximum intra hamming distance is low, with the highest value obtained for the 2000 measurements of 1.444 %. Hamming weight indicates a very slight bias towards one (Values between 50.83 % and 51.03 %), which is an acceptable range. Since the 2000 measurements are the most complete set they will be the ones used from this point onwards as measurements under nominal conditions and to obtain the chip's golden response. One interesting representation is the distribution of BER across the array to see if it follows any spatial pattern. This is shown in fig. 5.5, and there is no noticeable pattern. It seems clear that most of the errors come from a few cells. Another useful cell-wise representation shown in fig. 5.6 is the probability of each cell to power-up as either "1" or "0", obtained from the approximation of eq. 2.5.

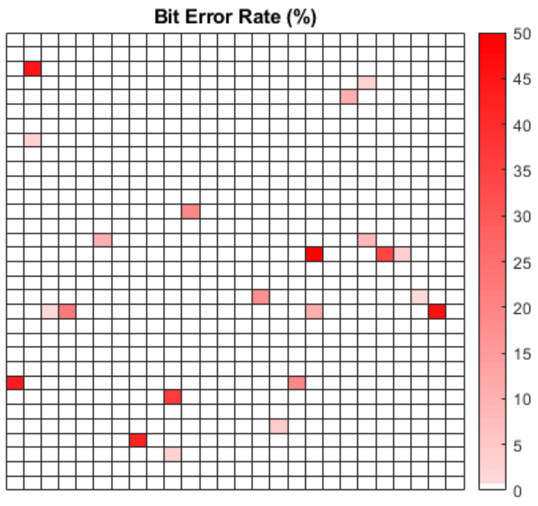

Figure 5.5: Cell-wise representation of BER for the 2000 measurements under nominal conditions.

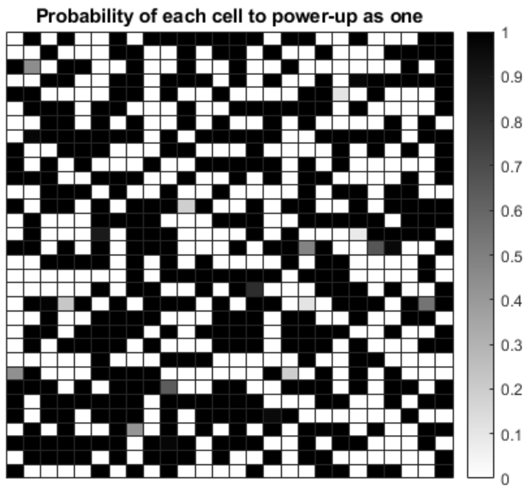

Figure 5.6: Probability of each cell to power up as "1" based on 2000 measurements under nominal conditions.

Again, no spatial pattern is observed and the distribution of values appears random.

The performance of the selections presented before under nominal conditions is now examined. BER, maximum intra hamming distance and hamming weight are considered in Table 5.2.

Selection	BER(%)	MAXintraHD(%)	Hamming Weight(%)
First	0.4625	2.3438	48.07
Random	0.6398	3.1250	45.11
ME Stable	0.0227	0.7813	52.32
MTSV Strongest	0.0012	0.7813	41.41
MTSV Balanced	0.0012	0.7813	50

Table 5.2: Performance of different selections of 128 bits when performing 2000 power-ups under nominal conditions.

As expected *First* and *Random* have BER values similar to the ones shown in Table 5.1. Their maximum intra hamming distance is higher due to taking a smaller sample size. *ME Stable* shows a substantial improvement over the values shown in Table 5.1. BER has reduced from around 0.4-0.6 % to 0.0227 %. Hamming weight indicates a result close to uniform, with 52.32 %. For *MTSV Strongest* and *MTSV Balanced*, BER is reduced even more, going as low as 0.0012 %. This is one order of magnitude better than *ME Stable*, proving the advantages of the proposed method.

It is important to keep this very small value for BER in perspective. 2000 power-ups per cell amount to $2.56 \cdot 10^5$ total power-ups for 128 cells. According to eq. 2.6, there were 3 total erroneous power-ups out of $2.56 \cdot 10^5$. However, these BER values are still not good enough to meet the target of $7.8 \cdot 10^{-7}$ %.

Regarding uniformity, there is a significant bias in favor of cells that have “0” as a preferred value for *MTSV Strongest*. A hamming weight of around 41 % is 10% lower than the hamming weight for all cells. This could be due to the cells’ layout, although it is fully symmetric. At this moment, this is an open topic that needs investigation. Further experiments and research are being performed to explain this difference.

MTSV Balanced solves this issue, which is important to avoid leakage of information by the helper data. Hamming weight naturally indicates perfect uniformity at 50 %. BER is the same as with *MTSV Strongest*, so taking this selection would not come at a significant cost in reliability, at least under nominal conditions.

5.3 Environmental variations

For measurements under environmental variations, only BER is considered since it is the most relevant metric for reliability. 200 power-ups were performed for each cell at each environmental condition. This is significantly lower than the 2,500 performed in the last chapter due to the fact that 832 cells are evaluated in each case, not just 100.

5.3.1 Resilience to supply voltage variations

In this set of measurements, cells were powered up 200 times at room temperature and different supply voltages, i.e., raising their supply voltage from ground (0 V) to 1.08 V, 1.14 V, 1.26 V and 1.32 V. This corresponds to a range of the nominal voltage 1.2 V ± 10%.

V_{DD} (V)	All	*First*	*Random*	*ME stable*	*MTSV Strongest*	*MTSV Balanced*
1.08	0.5355	0.3164	0.9492	0.1172	0.0078	0.0078
1.14	0.5788	0.4023	1.0469	0.1406	0	0
1.26	0.6444	0.3555	1.0117	0.043	0	0
1.32	0.7148	0.375	1.0508	0.0391	0.0039	0.0039

Table 5.3: Mean BER value for the different selections when performing 200 power-ups under supply voltage variations

BER increases slightly as V_{DD} increases according to this data. The change in reliability is not very significant, as expected. The worst BER is obtained for a supply voltage of 1.32 V. Bit selection techniques show a similar improvement to the one done under nominal conditions. *MTSV Strongest* and *MTSV Balanced* show low errors so these selections are robust under supply voltage variations.

5.3.2 Resilience to temperature variations

In these tests, instead of using the ACS climatic chamber employed in the previous chapter, a Thermonics T-2650BV is used. It allows for a larger range of temperatures by setting a determined temperature for the IC chip without affecting the whole PCB. 200 power-ups were performed for each cell, at −20 °C, 0 °C, 10 °C,

T (°C)	All	*First*	*Random*	*ME stable*	*MTSV Strongest*	*MTSV Balanced*
-20	2.4272	2.2578	1.6641	0.0117	0.039	0.0039
0	1.3267	0.8125	0.9844	0	0	0
10	0.9344	0.5234	0.9844	0.0078	0	0
20	0.8057	0.3789	0.9766	0.0195	0	0
40	0.9615	0.7656	1.2734	0.3711	0	0

Table 5.4: Mean BER value for the different selections when performing 200 power-ups under temperature variations

20 °C and 40 °C. The resulting BER is shown in Table 5.4. Measurements at higher temperature were attempted, but the results did not agree with the expected value found in literature and simulations. More experiments were planned, but the Thermonic T-2650BV broke down, and they could not be performed.

The changes in BER due to temperature variations are more noticeable. As temperature deviates from 20 °C upwards or downwards, BER increases. From −20 °C to 40 °C, *MTSV Strongest* and *MTSV Balanced* selections are able to provide practically error free results.

5.3.3 Resilience to circuit aging

In this section, the procedure described in section 4.4.4.2 is employed to evaluate the adequacy of the selections for cells under accelerated aging. In this case, all the cells were stressed. The measures performed are right after stress ended, 8 days later and 21 days later. It is important to point out that measurements right after stress are not instantaneous, it takes about 100 seconds to perform the 200 power-ups. The BER for each selection and point in time t are shown in Table 5.5.

t	All	*First*	*Random*	*ME stable*	*MTSV Strongest*	*MTSV Balanced*
Before	0.7858	0.6836	1.4453	0.043	0	0
Right after stress	7.4049	6.9492	6.4297	4.6523	1.0586	0.8477
8 Days	4.7918	5.8438	4.6914	1.9297	0.0039	0.0039
21 Days	4.7136	5.1406	4.6719	1.2227	0	0

Table 5.5: Mean BER value for the different selections when performing 200 power-ups at different points in time after accelerated aging.

The degradation of the cells is quite clear, with BER being around 10 times larger

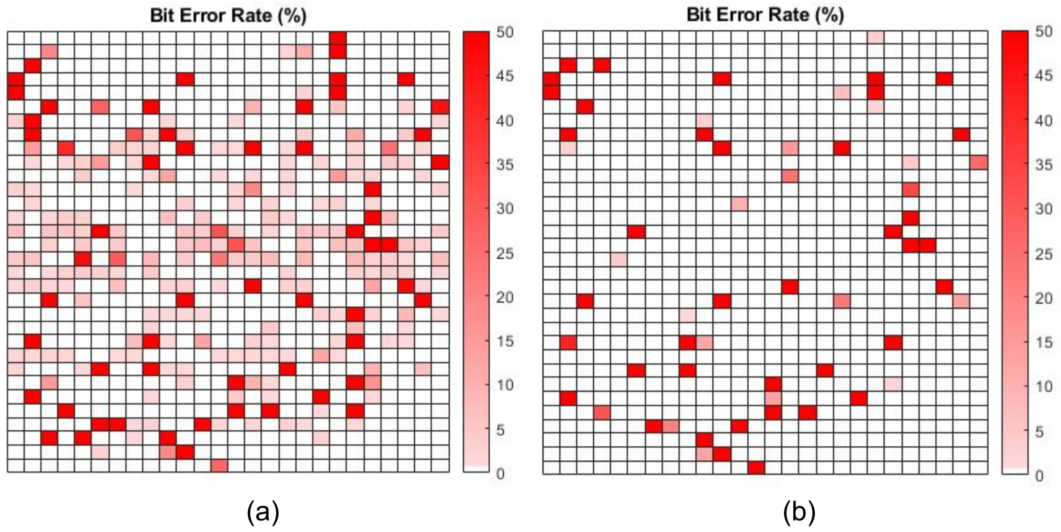

Figure 5.7: Cell-wise representation of BER for 200 measurements right after stress (a) and 8 days later (b).

before and after stress for all cells (from 0.7858 % to 7.4049 %). *First* and *Random* have values close to *All cells*, as expected. The temporal component of BTI is demonstrated by the decrease of about 3 % in BER for all cells after 8 days. A measurement 21 days later shows some recovery, but it is considerably smaller. The recovery can be observed graphically by comparing the cell-wise BER right after stress, as shown in fig. 5.7 (a) and 8 days later, as shown in fig. 5.7 (b).

ME stable is capable of reducing the BER by about 3%, but these values are still high. The results for either MTSV selection are outstanding. Right after stress BER is reduced to 1.0586 % for *MTSV Strongest* and to 0.8477 % *MTSV Balanced.* After 8 days it goes down to 0.0039 % for both of them, and after 21 the selected cells achieve 0% BER, the value obtained before stress. It clearly shows how MTSV is able to select those cells resilient to aging while the ME technique cannot.

5.4 Complete cryptographic solution

Once the PUF response has been fully characterized, a complete HDA scheme can be implemented with these measurements. The fuzzy commitment construction explained in section 2.5 will be employed, where a secret key is obfuscated through the PUF response. To measure the validity of this construction, each time the key is recovered successfully, it will be counted as a success. The Acceptance Rate (AR) will be the number of successful recoveries. In this way, it can range from zero, for no successful recoveries, to the number of attempts, where the key is successfully recovered each time.

As discussed in chapter 2.5.2, there are plenty of choices available as an ECC. Considering the low BER for both MTSV selections, a repetition code seems like the best choice. This code is the simplest to implement, essential for resource-constrained devices, has good error correction capabilities and will not require many bits since BER is already significantly reduced by the selection. To generate a 128-bit key, only two repetition codes use less than 832 bits: $R[3,1,3]$, which uses 384 bits and $R[5,1,5]$, which uses 640. Naturally, new selections are required for these sizes, performed by following the same criteria explained earlier. As MTSV classifies cells according to their strength, a larger selection will have higher BER.

First, the HDA will be implemented with the power-ups performed under nominal conditions for the different selections. Then, it should be tested in a bad corner for reliability. The measurement right after accelerated aging is not considered, as applying such a high stress for an extended period of time is not realistic in actual implementations. A better approach is to use the measurements at low temperature, -20°C, as well as those taken 8 days after accelerated aging, as they have a considerable increase in BER and can be trusted to represent a realistic degradation of reliability.

Afterwards, one interesting test is to try and recover the key with a response from another chip. There should be an AR of zero, as each PUF is unique. Finally, the theoretical requirements of the repetition code for each scenario based on the BER measurements are examined.

5.4.1 Key generation under nominal conditions

First, the 2000 power-ups performed under nominal conditions are used to test this construction. The helper data corresponding to each selection and repetition code is generated once (Enrollment) and each one of the 2000 power-ups are employed as individual attempts to recover the key (Reconstruction). The helper data generated during enrollment will be the one used in the other tests.

Results regarding AR are shown in Table 5.6, where n refers to the parameter of the repetition code. The AR is high for all selections, but it is expected as the number of measurements is low for a BER of around 0.5 %. In any case, for $R[3, 1, 3]$, *First* shows one error and *Random* seven. These errors no longer occur if the repetition code is expanded to $R[5, 1, 5]$, showing how a code with more redundancy has higher error correcting capabilities.

	First		*Random*		*ME stable*		*MTSV Strongest*		*MTSV Balanced*	
n	3	5	3	5	3	5	3	5	3	5
Required bits	384	640	384	640	384	640	384	640	384	640
BER (%)	0.2622	0.5447	0.6023	0.4178	0.0164	0.0133	0.0007	0.0016	0.0007	0.0016
AR	1999	2000	1993	2000	2000	2000	2000	2000	2000	2000

Table 5.6: BER and AR value for the different selections when performing 2000 power-ups under nominal conditions using two different repetition codes.

5.4.2 Key generation at low temperature

For $-20\,^{\circ}\mathrm{C}$ less power-ups were performed, 200, so the AR will go up to that value. Results are shown in Table 5.7. There are more erroneous responses this time, specially considering how the number of power-ups is 10 times lower. It is clear how the decrease in reliability, i.e. the increase in BER, translates into a decrease in AR, and why considering only measurements under nominal conditions is not enough. Only *MTSV Balanced* is able to recover the key 200 times for $R[3, 1, 3]$. There is one specific power-up with a larger number of bit-flips, due to the statistic nature of this process. This power-up is enough to cause one error in most selections. Even after applying a $R[5, 1, 5]$ only the selections based on MTSV are able to recover the key, showing once again its advantages versus multiple evaluation.

	First		*Random*		*ME stable*		*MTSV Strongest*		*MTSV Balanced*	
n	3	5	3	5	3	5	3	5	3	5
Required bits	384	640	384	640	384	640	384	640	384	640
BER (%)	1.5299	2.1696	3.4792	2.1273	0.7643	1.4500	0.0143	0.2203	0.0938	0.2453
AR	198	199	60	199	199	199	199	200	200	200

Table 5.7: BER and AR value for the different selections when performing 200 power-ups at −20 °C using two different repetition codes.

5.4.3 Key generation after accelerated aging

As in the previous test, 200 power-ups are employed. The AR obtained for each selection and repetition code is shown in Table 5.8. In this case, except for *First* and *Random* for a repetition code $R[3, 1, 3]$, the key is recovered successfully each time. This may seem counter-intuitive considering how AR was lower in the previous test while BER is higher in this one. However, it is explained by considering that the way BER is distributed matters greatly for key generation. In the previous test, there was one power-up that showed more bit-flips. Out of 200 power-ups, it does not have a large impact in the average BER, but it does translate in one failure to recover the key.

	First		*Random*		*ME stable*		*MTSV Strongest*		*MTSV Balanced*	
n	3	5	3	5	3	5	3	5	3	5
Required bits	384	640	384	640	384	640	384	640	384	640
BER (%)	4.4766	4.5484	5.4375	4.7492	2.8672	3.7188	0.0482	1.0211	0.0482	0.9695
AR	186	200	139	200	200	200	200	200	200	200

Table 5.8: BER and AR value for the different selections when performing 200 power-ups 8 days after accelerated aging using two different repetition codes.

5.4.4 Key generation using other chips

In this test, an attempt to recover the key with a different chip is performed. To do this, the helper data generated under nominal conditions for this chip is used in combination with the 200 power-ups measured for each of the five chips employed in the previous chapter, resulting in 1000 power-ups to work with. For any selection or repetition code, the AR obtained is 0, showing that a chip cannot be used to recover the key obfuscated by another chip. This verifies the uniqueness of the PUF, although on a small scale since only 6 chips are used in total.

5.4.5 Repetition code requirements for low KER

So far, different repetition codes have been tested with real measurements. However, the target for KER was set at 10^{-4} %. It is important to remember that this means around one error per million attempts. In our measurements, the largest number of power-ups performed was 2000 under nominal conditions, and to verify that the KER target is met with statistical significance power-ups in the order of a million would be required at a bad corner of reliability, which is not feasible. Instead, the error reduction capabilities of the repetition code can be estimated using the binomial distribution (equation 2.1). Since more than half of the bits need to fail, the probability of failure after the repetition code will be the cumulative probability $P\left(x \geq \frac{n+1}{2}\right)$ for $p = \text{BER}$ and $R[n, 1, n]$. This probability can be calculated through MATLAB, and the lowest n that meets the requirements is chosen. To differentiate the two stages, BER_0 is the BER before applying an ECC and BER_F the BER after ECC. As a reminder, the target BER_F is $7.8 \cdot 10^{-7}$ % for a KER of 10^{-4}%.

Leaving out the measurements performed right after applying stress and taking the values for 128-bit selections, the worst case for each selection is a BER of 5.8438 % for *First*, 4.6914 % for *Random* and 1.9297 % for *ME Stable*, where these three values were obtained in the aging test 8 days after accelerated aging and 0.0078 % for both *MTSV Strongest* and *MTSV Balanced*, obtained in the supply voltage test for 1.08 V. Since both *MTSV Strongest* and *MTSV Balanced* have the same BER_0, they will be simply referred as *MTSV* from now on. The results of this procedure are shown in Table 5.9.

Selection	BER_0 (%)	C	BER_F (%)	KER (%)	PUF bits
First	5.8438	$R[21, 1, 21]$	5.53 E-7	7.07 E-5	2688
Random	4.6914	$R[19, 1, 19]$	3.22 E-7	4.13 E-5	2432
ME Stable	1.9297	$R[13, 1, 13]$	1.54 E-7	1.93 E-5	1644
MTSV	0.0078	$R[5, 1, 5]$	4.75 E-10	6.07 E-8	640

Table 5.9: Required repetition code to meet the KER target for each selection.

The predicted reduction in required redundant bits is made apparent with these results. *First* and *Random* require a large amount of redundant bits, 2688 and 2432 bits respectively. While for *ME Stable* the required bits are reduced significantly, down to 1644, it is still a large number. The repetition code is perhaps not ideal for these selections, but as seen in Table 2.1 there is a limit on how efficient the code may be with higher complexity. For *MTSV*, a $R[5, 1, 5]$ code will achieve a KER of $6.07 \cdot 10^{-8}$ %, considerably lower than the rest. The required bits are significantly reduced. These results are consistent with the measurements at low temperature, where only a MTSV based selection with $R[5, 1, 5]$ is able to recover the key successfully each time.

Chapter 6

Conclusions

In this work, a new bit selection technique (MTSV), an experimental procedure to evaluate the reliability of SRAM cells for their utilization in power-up PUFs, is exhaustively validated. First, by itself, showing its ability to distinguish unstable, strong and weak cells. Cells selected as strong are consistently shown to be considerably more reliable than those classified as weak under nominal conditions, temperature and supply variations and after accelerated aging.

Afterwards, MTSV bit selection is tested in the context of using a SRAM PUF as a 128-bit key generator. This is done by comparing the performance of a selection based on MTSV with the performance of other possible selections. When measuring BER under nominal conditions, temperature and supply variations, as well as after accelerated aging, the MTSV-based selections consistently outperform the rest. Next, the ability of these selections to generate a key is tested by implementing a HDA based on the fuzzy commitment construction with a small repetition code. The results from this test demonstrate how MTSV-based selections are the only ones able to always recover the key successfully under different operating conditions. Finally, the required repetition code to achieve a KER lower than $10^{-4}\%$ for each selection is calculated. Although further measurements are still necessary to completely evaluate this method, specially at high temperatures, the results show quantitatively how much the requirements for the ECC are reduced, with a selection based on MTSV needing three to five times less redundant bits than the other selections. This considerably reduces the cost of implementation of SRAM PUFs, a matter of great importance due to their use in resource-constrained devices. The MTSV bit-selection technique represents one more step in the development of SRAM PUFs.

Future work will focus on implementing the MTSV technique in a commercial chip, including the enrollment and reconstruction phases, to demonstrate its capabilities in real implementations. Another step further would be to perform the hardware design to implement the complete crytographic solution on an IC, as well as evaluating the aging resilience of this design.

Bibliography

[1] "A PUF taxonomy". In: *Applied Physics Reviews* 6.1 (2019). ISSN: 19319401. DOI: 10.1063/1.5079407.

[2] Christoph Böhm and Maximilian Hofer. *Physical Unclonable Functions in Theory and Practice*. New York, NY: Springer New York, 2013, p. 300. ISBN: 978-1-4614-5039-9. DOI: 10.1007/978-1-4614-5040-5. URL: http://link.springer.com/10.1007/978-1-4614-5040-5.

[3] *Search — NXP*. URL: https://www.nxp.com/search?keyword=AN12292%7B%5C&%7Dstart=0 (visited on 11/17/2020).

[4] *QuiddiKey - Intrinsic ID — Home of PUF Technology*. URL: https://www.intrinsic-id.com/products/quiddikey/ (visited on 11/17/2020).

[5] *Libero SoC v11.6 Resources Archive — Microsemi*. URL: https://www.microsemi.com/product-directory/design-resources-archive/5086-libero-soc-v11-6-resources-archive (visited on 11/17/2020).

[6] Jeroen Delvaux et al. "Helper data algorithms for puf-based key generation: Overview and analysis". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34.6 (2015), pp. 889–902. ISSN: 02780070. DOI: 10.1109/TCAD.2014.2370531.

[7] Matthias Hiller, Ludwig Kürzinger, and Georg Sigl. "Review of error correction for PUFs and evaluation on state-of-the-art FPGAs". In: *Journal of Cryptographic Engineering* 10.3 (2020), pp. 229–247. ISSN: 21908516. DOI: 10.1007/s13389-020-00223-w. URL: https://doi.org/10.1007/s13389-020-00223-w.

[8] M. Alioto. "Trends in Hardware Security: From basics to ASICs". In: *IEEE Solid-State Circuits Magazine* 11 (2019), pp. 56–74.

[9] Yizhak Shifman et al. "A Method to Improve Reliability in a 65-nm SRAM PUF Array". In: *IEEE Solid-State Circuits Letters* 1.6 (2018), pp. 138–141. ISSN: 2573-9603. DOI: 10.1109/LSSC.2018.2879216. URL: https://ieeexplore.ieee.org/document/8519616/.

[10] Pim Tuyls, Boris Škorić, and Tom Kevenaar. *Security with noisy data: On private biometrics, secure key storage and anti-counterfeiting*. Springer London, 2007, pp. 1–339. ISBN: 9781846289835. DOI: 10.1007/978-1-84628-984-2.

[11] Ravikanth Pappu et al. "Physical one-way functions". In: *Science* 297.5589 (2002), pp. 2026–2030. ISSN: 00368075. DOI: 10.1126/science.1074376.

[12] Teng Xu and Miodrag Potkonjak. *Secure System Design and Trustable Computing.* 2015. ISBN: 9783319149714. DOI: `10.1007/978-3-319-14971-4_3`.

[13] Charles Herder et al. “Physical unclonable functions and applications: A tutorial”. In: *Proceedings of the IEEE* 102.8 (2014), pp. 1126–1141. ISSN: 00189219. DOI: `10.1109/JPROC.2014.2320516`.

[14] Roel Maes and Ingrid Verbauwhede. “Physically unclonable functions: A study on the state of the art and future research directions”. In: *Information Security and Cryptography.* 9783642143120. Springer International Publishing, 2010, pp. 3–37. DOI: `10.1007/978-3-642-14452-3_1`.

[15] *Trusted Platform Module (TPM) Summary — Trusted Computing Group.* URL: `https://trustedcomputinggroup.org/resource/trusted-platform-module-tpm-summary/` (visited on 11/06/2020).

[16] Shunsuke Okumura et al. “A 128-bit chip identification generating scheme exploiting SRAM bitcells with failure rate of 4.45 × 10-19”. In: *European Solid-State Circuits Conference* (2011), pp. 527–530. ISSN: 19308833. DOI: `10.1109/ESSCIRC.2011.6044938`.

[17] G. Edward Suh and Srinivas Devadas. “Physical unclonable functions for device authentication and secret key generation”. In: *Proceedings - Design Automation Conference* (2007), pp. 9–14. ISSN: 0738100X. DOI: `10.1109/DAC.2007.375043`.

[18] Md Tauhidur Rahman et al. “An Aging-Resistant RO-PUF for Reliable Key Generation”. In: *IEEE Transactions on Emerging Topics in Computing* 4.3 (2016), pp. 335–348. ISSN: 21686750. DOI: `10.1109/TETC.2015.2474741`.

[19] Yansong Gao, Said F Al-Sarawi, and Derek Abbott. “Physical unclonable functions”. In: *Nature Electronics* 3.2 (2020), pp. 81–91. ISSN: 2520-1131. DOI: `10.1038/s41928-020-0372-5`.

[20] *Advanced encryption standard (AES).* Tech. rep. Gaithersburg, MD: National Institute of Standards and Technology, 2001. DOI: `10.6028/NIST.FIPS.197`. URL: `https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf`.

[21] R. L. Rivest, A. Shamir, and L. Adleman. “A Method for Obtaining Digital Signatures and Public-Key Cryptosystems”. In: *Commun. ACM* 21.2 (Feb. 1978), pp. 120–126. ISSN: 0001-0782. DOI: `10.1145/359340.359342`.

[22] Macarena C. Martínez-Rodríguez et al. “VLSI design of trusted virtual sensors”. In: *Sensors (Switzerland)* 18.2 (2018). ISSN: 14248220. DOI: `10.3390/s18020347`.

[23] D. E. Holcomb, W. P. Burleson, and K. Fu. “Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers”. In: *IEEE Transactions on Computers* 58.9 (2009), pp. 1198–1210.

[24] Iluminada Baturone, Miguel A. Prada-Delgado, and Susana Eiroa. “Improved Generation of Identifiers, Secret Keys, and Random Numbers From SRAMs”. In: *IEEE Transactions on Information Forensics and Security* 10.12 (2015), pp. 2653–2668. ISSN: 15566013. DOI: `10.1109/TIFS.2015.2471279`.

[25] Andrew Rukhin, Juan Soto, and James Nechvatal. "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications". In: *Nist Special Publication* 22.April (2010), 1/1–G/1. URL: http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22rev1a.pdf.

[26] Pim Tuyls and Boris Skoric. "Physical Unclonable Functions for enhanced security of tokens and tags". In: Jan. 2006, pp. 30–37. DOI: 10.1007/978-3-8348-9195-2_4.

[27] Ulrich Rührmair et al. "Optical pufs reloaded". In: *Eprint. Iacr. Org* (2013).

[28] James Buchanan et al. "Forgery: 'Fingerprinting' documents and packaging". In: *Nature* 436 (Aug. 2005), p. 475. DOI: 10.1038/436475a.

[29] Ghaith Hammouri, Aykutlu Dana, and Berk Sunar. "CDs have fingerprints too". In: *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer. 2009, pp. 348–362.

[30] S Vrijaldenhoven. "Acoustical Physical Uncloneable Functions". MA thesis. the Netherlands: Technische Universiteit Eindhoven, 2005.

[31] J. W. Lee et al. "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)*. 2004, pp. 176–179.

[32] D. Lim. "Extracting Secret Keys from Integrated Circuits". MA thesis. USA: MIT, 2004.

[33] Blaise Gassend et al. "Silicon Physical Random Functions". In: *Proceedings of the 9th ACM Conference on Computer and Communications Security*. CCS '02. Washington, DC, USA: Association for Computing Machinery, 2002, pp. 148–160. ISBN: 1581136129. DOI: 10.1145/586110.586132.

[34] Jorge Guajardo et al. "FPGA intrinsic PUFs and their use for IP protection". In: *International workshop on cryptographic hardware and embedded systems*. Springer. 2007, pp. 63–80.

[35] Daniel E Holcomb, Wayne P Burleson, Kevin Fu, et al. "Initial SRAM state as a fingerprint and source of true random numbers for RFID tags". In: *Proceedings of the Conference on RFID Security*. Vol. 7. 2. 2007, p. 01.

[36] Kan Xiao et al. "Bit selection algorithm suitable for high-volume production of SRAM-PUF". In: *Proceedings of the 2014 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2014* (2014), pp. 101–106. DOI: 10.1109/HST.2014.6855578.

[37] Xu Zhang et al. "A novel SRAM PUF stability improvement method using ionization irradiation". In: *Electronics (Switzerland)* 9.9 (2020), pp. 1–16. ISSN: 20799292. DOI: 10.3390/electronics9091498.

[38] Haji Akhundov et al. "Public-Key Based Authentication Architecture for IoT Devices Using PUF". In: *6th International Conference on Computer Science, Engineering and Information Technology (CSEIT-2019)* (2019). DOI: 10.5121/csit.2019.91328.

[39] "An alternative to error correction for SRAM-like PUFs". In: *Lecture Notes in Computer Science* 6225 LNCS (2010), pp. 335–350. ISSN: 03029743. DOI: 10.1007/978-3-642-15031-9_23.

[40] Abhranil Maiti, Vikash Gunreddy, and Patrick Schaumont. "A systematic method to evaluate and compare the performance of physical unclonable functions". In: *Embedded Systems Design with FPGAs* 9781461413 (2013), pp. 245–267. DOI: 10.1007/978-1-4614-1362-2_11.

[41] Mudit Bhargava, Cagla Cakir, and Ken Mai. "Reliability enhancement of bistable PUFs in 65nm bulk CMOS". In: *Proceedings of the 2012 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2012* (2012), pp. 25–30. DOI: 10.1109/HST.2012.6224314.

[42] S. Eiroa et al. "Reducing bit flipping problems in SRAM physical unclonable functions for chip identification". In: *2012 19th IEEE International Conference on Electronics, Circuits, and Systems, ICECS 2012* (2012), pp. 392–395. DOI: 10.1109/ICECS.2012.6463720.

[43] Christoph Bösch et al. "Efficient helper data key extractor on FPGAs". In: *Lecture Notes in Computer Science* 5154 LNCS (2008), pp. 181–197. ISSN: 03029743. DOI: 10.1007/978-3-540-85053-3_12.

[44] Muqing Liu et al. "A data remanence based approach to generate 100% stable keys from an SRAM physical unclonable function". In: *Proceedings of the International Symposium on Low Power Electronics and Design* (2017). ISSN: 15334678. DOI: 10.1109/ISLPED.2017.8009192.

[45] Yevgeniy Dodis et al. "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. Technical Report 2003/235, Cryptology ePrint archive, http://eprint.iacr.org, 2006. Previous version appeared at EUROCRYPT 2004". In: (2004), pp. 79–100. URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.148.5971%7B%5C&%7Drank=10.

[46] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. "Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs". In: *Lecture Notes in Computer Science*. Vol. 5747 LNCS. 2009, pp. 332–347. ISBN: 364204137X. DOI: 10.1007/978-3-642-04138-9_24. URL: http://link.springer.com/10.1007/978-3-642-04138-9%7B%5C_%7D24.

[47] Ari Juels and Martin Wattenberg. "A fuzzy commitment scheme". In: *Proceedings of the 6th ACM conference on Computer and communications security - CCS '99*. New York, New York, USA: ACM Press, 1999, pp. 28–36. ISBN: 1581131488. DOI: 10.1145/319709.319714. arXiv: 0809.1318. URL: http://arxiv.org/abs/0809.1318%20http://portal.acm.org/citation.cfm?doid=319709.319714.

[48] Vincent Van Der Leest, Bart Preneel, and Erik Van Der Sluis. "Soft decision error correction for compact memory-based PUFs using a single enrollment". In: *Lecture Notes in Computer Science* 7428 LNCS (2012), pp. 268–282. ISSN: 03029743. DOI: 10.1007/978-3-642-33027-8_16.

[49] Lieneke Kusters et al. “Security of helper data schemes for SRAM-PUF in multiple enrollment scenarios”. In: *IEEE International Symposium on Information Theory - Proceedings* (2017), pp. 1803–1807. ISSN: 21578095. DOI: 10.1109/ISIT.2017.8006840.

[50] M. Yu et al. “Security and Reliability Properties of Syndrome Coding Techniques Used in PUF Key Generation”. In: *GOMACTech* (2013), pp. 1–4. URL: http://www.cosic.esat.kuleuven.be/publications/article-2323.pdf.

[51] Jorge Castiñeira Moreira and Patrick Guy Farrell. *Essentials of Error-Control Coding*. Chichester, UK: John Wiley & Sons, Ltd, 2006, pp. 1–361. ISBN: 9780470035726. DOI: 10.1002/9780470035726.

[52] Sachin Taneja and Massimo Alioto. “PUF-based Key Generation with Design Margin Reduction via In-Situ and PVT Sensor Fusion”. In: *ESSCIRC 2019 - IEEE 45th European Solid State Circuits Conference* (2019), pp. 61–64. DOI: 10.1109/ESSCIRC.2019.8902733.

[53] Rolf Johannesson and Kamil Sh Zigangirov. *Fundamentals of Convolutional Coding*. Ed. by Rolf Johannesson and Kamil Sh. Zigangirov. Hoboken, NJ, USA: John Wiley & Sons, Inc., 2015. ISBN: 9781119098799. DOI: 10.1002/9781119098799.

[54] C. Rachinger, J. B. Huber, and R. R. Müller. “Comparison of Convolutional and Block Codes for Low Structural Delay”. In: *IEEE Transactions on Communications* 63.12 (2015), pp. 4629–4638. DOI: 10.1109/TCOMM.2015.2488661.

[55] Christoph Böhm, Maximilian Hofer, and Wolfgang Pribyl. “A microcontroller SRAM-PUF”. In: *Proceedings - 2011 5th International Conference on Network and System Security, NSS 2011* (2011), pp. 269–273. DOI: 10.1109/ICNSS.2011.6060013.

[56] Patrick Koeberl et al. “Entropy loss in PUF-based key generation schemes: The repetition code pitfall”. In: *Proceedings of the 2014 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2014* (2014), pp. 44–49. DOI: 10.1109/HST.2014.6855566.

[57] R. W. Hamming. “Error detecting and error correcting codes”. In: *The Bell System Technical Journal* 29.2 (1950), pp. 147–160. DOI: 10.1002/j.1538-7305.1950.tb00463.x.

[58] *File:Hamming(7,4).svg - Wikimedia Commons*. URL: https://commons.wikimedia.org/wiki/File:Hamming(7,4).svg (visited on 11/12/2020).

[59] W. Cary Huffman and Vera Pless. *Fundamentals of Error-Correcting Codes*. 2003. ISBN: 9780521782807. DOI: 10.1017/cbo9780511807077.

[60] Emmanuel Abbe, Amir Shpilka, and Min Ye. “Reed-Muller Codes: Theory and Algorithms”. In: *arXiv* (2020). arXiv: 2002.03317. URL: http://arxiv.org/abs/2002.03317.

[61] Matthias Hiller et al. “Low-area reed decoding in a generalized concatenated code construction for PUFs”. In: *Proceedings of IEEE Computer Society Annual Symposium on VLSI, ISVLSI* 07-10-July-2015 (2015), pp. 143–148. ISSN: 21593477. DOI: 10.1109/ISVLSI.2015.31.

[62] Venkatesan Guruswami. *Course Notes, Introduction to Coding theory: Reed-Solomon, BCH, Reed-Muller, and concatenated codes Reed-Solomon codes.* 2010. URL: `https://www.cs.cmu.edu/~venkatg/teaching/codingtheory/notes/notes6.pdf`.

[63] Roel Maes, Anthony Van Herrewege, and Ingrid Verbauwhede. "PUFKY: A fully functional PUF-based cryptographic key generator". In: *Lecture Notes in Computer Science* 7428 LNCS (2012), pp. 302–319. ISSN: 03029743. DOI: `10.1007/978-3-642-33027-8_18`.

[64] Matthias Hiller et al. "Breaking through fixed PUF block limitations with differential sequence coding and convolutional codes". In: *Proceedings of the ACM Conference on Computer and Communications Security* (2013), pp. 43–54. ISSN: 15437221. DOI: `10.1145/2517300.2517304`.

[65] Matthias Hiller and Georg Sigi. "Increasing the efficiency of syndrome coding for PUFs with helper data compression". In: *Proceedings -Design, Automation and Test in Europe, DATE* (2014), pp. 4–9. ISSN: 15301591. DOI: `10.7873/DATE2014.084`.

[66] Matthias Hiller, Meng Day Mandel Yu, and Georg Sigl. "Cherry-Picking Reliable PUF Bits with Differential Sequence Coding". In: *IEEE Transactions on Information Forensics and Security* 11.9 (2016), pp. 2065–2076. ISSN: 15566013. DOI: `10.1109/TIFS.2016.2573766`.

[67] Blaise Gassend et al. "Silicon physical random functions". In: *Proceedings of the ACM Conference on Computer and Communications Security* (2002), pp. 148–160. ISSN: 15437221. DOI: `10.1145/586131.586132`.

[68] Mudit Bhargava and Ken Mai. "An efficient reliable PUF-based cryptographic key generator in 65nm CMOS". In: *Proceedings -Design, Automation and Test in Europe, DATE* 1 (2014). ISSN: 15301591. DOI: `10.7873/DATE2014.083`.

[69] Roel Maes et al. "Experimental evaluation of Physically Unclonable Functions in 65 nm CMOS". In: *European Solid-State Circuits Conference.* 2012, pp. 486–489. ISBN: 9781467322126. DOI: `10.1109/ESSCIRC.2012.6341361`.

[70] Vincent Van Der Leest et al. "Efficient implementation of true random number generator based on SRAM PUFs". In: *Lecture Notes in Computer Science* 6805 LNCS (2012), pp. 300–318. ISSN: 03029743. DOI: `10.1007/978-3-642-28368-0_20`.

[71] Jawar Singh, Saraju Mohanty, and Dhiraj Pradhan. *Robust SRAM Designs and Analysis.* Jan. 2013. ISBN: 978-1-4614-0817-8. DOI: `10.1007/978-1-4614-0818-5`.

[72] Wendong Wang et al. "Exploiting power supply ramp rate for calibrating cell strength in SRAM PUFs". In: *2018 IEEE 19th Latin-American Test Symposium, LATS 2018* 2018-January (2018), pp. 1–6. DOI: `10.1109/LATW.2018.8349685`.

[73] Geert Jan Schrijen and Vincent Van Der Leest. "Comparative analysis of SRAM memories used as PUF primitives". In: *Proceedings -Design, Automation and Test in Europe, DATE* (2012), pp. 1319–1324. ISSN: 15301591. DOI: `10.1109/date.2012.6176696`.

[74] Mathias Claes, Vincent Van Der Leest, and An Braeken. "Comparison of SRAM and FF PUF in 65nm technology". In: *Lecture Notes in Computer Science* 7161 LNCS (2012), pp. 47–64. ISSN: 03029743. DOI: 10.1007/978-3-642-29615-4_5.

[75] Mafalda Cortez et al. "Adapting voltage ramp-up time for temperature noise reduction on memory-based PUFs". In: *Proceedings of the 2013 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2013* (2013), pp. 35–40. DOI: 10.1109/HST.2013.6581562.

[76] *Operating temperature - Wikipedia*. URL: https://en.wikipedia.org/wiki/Operating_temperature (visited on 11/24/2020).

[77] Daniel Kraak et al. "Device aging: A reliability and security concern". In: *Proceedings of the European Test Workshop* 2018-May (2018), pp. 1–10. ISSN: 15581780. DOI: 10.1109/ETS.2018.8400702.

[78] Christian Schlünder et al. "HCI vs. BTI? - Neither one's out". In: *IEEE International Reliability Physics Symposium Proceedings* (2012), pp. 2–7. ISSN: 15417026. DOI: 10.1109/IRPS.2012.6241797.

[79] James H. Stathis, Souvik Mahapatra, and Tibor Grasser. "Controversial issues in negative bias temperature instability". In: *Microelectronics Reliability* 81.November 2017 (2018), pp. 244–251. ISSN: 00262714. DOI: 10.1016/j.microrel.2017.12.035.

[80] Dieter K. Schroder and Jeff A. Babcock. "Negative bias temperature instability: Road to cross in deep submicron silicon semiconductor manufacturing". In: *Journal of Applied Physics* 94.1 (2003), pp. 1–18. ISSN: 00218979. DOI: 10.1063/1.1567461.

[81] B. Kaczer et al. "Origin of NBTI variability in deeply scaled pFETs". In: *IEEE International Reliability Physics Symposium Proceedings* June (2010), pp. 26–32. ISSN: 15417026. DOI: 10.1109/IRPS.2010.5488856.

[82] Alec Roelke and Mircea R. Stan. "Controlling the reliability of SRAM PUFs with directed NBTI aging and recovery". In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26.10 (2018), pp. 2016–2026. ISSN: 10638210. DOI: 10.1109/TVLSI.2018.2836154.

[83] "Design of SRAM PUF with improved uniformity and reliability utilizing device aging effect". In: *Proceedings - IEEE International Symposium on Circuits and Systems*. Institute of Electrical and Electronics Engineers Inc., 2014, pp. 1941–1944. ISBN: 9781479934324. DOI: 10.1109/ISCAS.2014.6865541.

[84] Deepashree Sengupta and Sachin S. Sapatnekar. "Estimating Circuit Aging Due to BTI and HCI Using Ring-Oscillator-Based Sensors". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 36.10 (2017), pp. 1688–1701. ISSN: 02780070. DOI: 10.1109/TCAD.2017.2648840.

[85] Mudit Bhargava and Ken Mai. "A high reliability PUF using hot carrier injection based response reinforcement". In: *Lecture Notes in Computer Science* 8086 LNCS (2013), pp. 90–106. ISSN: 03029743. DOI: 10.1007/978-3-642-40349-1-6.

[86] Nikolaos Athanasios Anagnostopoulos et al. “Low-temperature data remanence attacks against intrinsic SRAM PUFs”. In: *Proceedings - 21st Euromicro Conference on Digital System Design, DSD 2018* (2018), pp. 581–585. DOI: 10.1109/DSD.2018.00102.

[87] P. Saraza-Canflanca et al. “Design Considerations of an SRAM Array for the Statistical Validation of Time-Dependent Variability Models”. In: *SMACD 2018 - 15th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design.* Institute of Electrical and Electronics Engineers Inc., 2018, pp. 73–76. ISBN: 9781538651520. DOI: 10.1109/SMACD.2018.8434900.

[88] Sanjay Rangan, Neal Mielke, and Everett C.C. Yeh. “Universal Recovery Behavior of Negative Bias Temperature Instability”. In: *Technical Digest - International Electron Devices Meeting* (2003), pp. 341–344. ISSN: 01631918.