October 2021

On Improving Robustness of Hardware Security Primitives and Resistance to Reverse Engineering Attacks

Vinay C. Patil
University of Massachusetts Amherst

ON IMPROVING ROBUSTNESS OF HARDWARE SECURITY PRIMITIVES AND RESISTANCE TO REVERSE ENGINEERING ATTACKS

A Dissertation Presented

by

VINAY CHANDRAKANTH PATIL

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2021

Electrical and Computer Engineering

ON IMPROVING ROBUSTNESS OF HARDWARE SECURITY PRIMITIVES AND RESISTANCE TO REVERSE ENGINEERING ATTACKS

A Dissertation Presented

by

VINAY CHANDRAKANTH PATIL

Approved as to style and content by:

Sandip Kundu, Chair

Wayne P. Burleson, Member

Daniel E. Holcomb, Member

Adam O'Neill, Member

Christopher V. Hollot, Department Chair
Electrical and Computer Engineering

DEDICATION

To my parents Leelavathi Patil and Chandrakanth Damodar Madiwal.

ACKNOWLEDGMENTS

I am most thankful to my advisor Prof. Sandip Kundu for his invaluable guidance since I first came to UMass for my Master's and through the pursuit of my doctoral degree. His dedication to supporting and guiding his students in both professional and personal arenas is a source of inspiration to us all. I would like to thank Prof. Wayne Burleson, Prof. Daniel Holcomb and Prof. Adam O'Neill for agreeing to be part of my committee. Their constructive inputs have played a great part in shaping my research.

I have had the pleasure of working with many talented collaborators on various projects through the years. I extend a special thanks to Prof. Holcomb for playing a crucial part as a sounding board along with my advisor to help direct the projects in the right direction during the early stages. I am greatly indebted to my colleagues from Brazil – Dr. Charles Prado of Inmetro and Dr. Leandro Santiago of UFF – for their contributions to my research and for their friendship. Also, I would like to thank my former lab-mates Arunkumar Vijayakumar and Nazmul Islam for their contributions to various projects. They made work feel like fun.

I extend my thanks to all current and past lab members for creating a wonderful work culture and ambiance in our lab. I will forever cherish my stay in Amherst due to the many happy experiences and friends I have made here. Their support and insights have played an important role in shaping me as a person.

Finally, I would like to thank my makers for the many sacrifices they have made without which I would not have been able to pursue my education. I will forever be thankful to their immense love and support.

ABSTRACT

ON IMPROVING ROBUSTNESS OF HARDWARE SECURITY PRIMITIVES AND RESISTANCE TO REVERSE ENGINEERING ATTACKS

SEPTEMBER 2021

VINAY CHANDRAKANTH PATIL

B.E., VISVESVARAYA TECHNOLOGICAL UNIVERSITY

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Sandip Kundu

The continued growth of information technology (IT) industry and proliferation of interconnected devices has aggravated the problem of ensuring security and necessitated the need for novel, robust solutions. Physically unclonable functions (PUFs) have emerged as promising secure hardware primitives that can utilize the disorder introduced during manufacturing process to generate unique keys. They can be utilized as *lightweight* roots-of-trust for use in authentication and key generation systems. Unlike insecure non-volatile memory (NVM) based key storage systems, PUFs provide an advantage – no party, including the manufacturer, should be able to replicate the physical disorder and thus, effectively clone the PUF. However, certain practical problems impeded the widespread deployment of PUFs. This dissertation addresses such problems of (*i*) reliability and (*ii*) unclonability. Also, obfuscation techniques have proven necessary to protect intellectual property in the presence of

an untrusted supply chain and are needed to aid against counterfeiting. This dissertation explores techniques utilizing layout and logic-aware obfuscation. Collectively, we present secure and cost-effective solutions to address crucial hardware security problems.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

Worldwide spending on Information Technology (IT) has been projected to reach $3.76 trillion in 2019, according to the latest forecast by Gartner, Inc [37]. Despite saturation in certain markets such as mobile devices or PCs, IT spending is set to further increase due to growth in younger sectors like cloud services, neural networking applications and, most of all, Internet of Things (IoT). The ubiquitousness of large, interconnected digital ecosystems, in critical areas such as banking or healthcare, presents an ever-growing attack surface for malicious entities to exploit. The complexity of the supply chain for the production and deployment of IT infrastructure also presents a security problem. Potential breaches in security can prove very costly in terms of loss of privacy, safety and revenue. Hence, developing robust solutions to ensure security of information being stored or exchanged, ability to identify/authenticate users or devices numbering in the millions and intellectual property (IP) protection in an untrusted supply chain has become vital.

1.1 Physically Unclonable Functions

Secure authentication and identification is crucial for many interconnected systems with, potentially, millions of users or edge devices. Often, a system's security goals have to be serviced using the deployed hardware and hence, integrating roots-of-trust into the design becomes essential. For example, resource-constrained systems like Smartcards or RFID tags can implement *lightweight* authentication via locally stored secret digital keys. However, such traditional non-volatile memory (NVM)

based storage of secret keys is susceptible to various forms of attacks when an attacker is able to gain physical access to the device. *Side-channel* attacks using power measurements, *fault injection* via overclocking or even *invasive* attacks such as decapsulation, de-layering and probing can be utilized to access the stored secret keys [21, 47]. An alternative to NVM-based secret key storage are *physically unclonable functions* (PUFs), which can *generate* secret keys when desired instead of storing them. In principle, PUFs leverage the physical disorder, say introduced during the complex manufacturing process for integrated circuits (ICs), and translate it into unique binary outputs. It is assumed that no one, including the PUF manufacturer, will be able to replicate the exact disorder of a PUF system and that any invasive attack will upset the disorder to render the PUF unusable.

Extensive research has explored construction of PUFs by exploiting various sources of physical disorder such as Optical PUF [93], coating PUF [121], phosphor PUF [63], RF COA [38] and LC-PUF [51]. Silicon transistor-based PUFs were first introduced by Gassend et al.[47]. Since then, a large number of silicon based PUFs like Arbiter PUF [75] and SRAM PUF [55] have been proposed.

Definition: A PUF P implements a unique mapping function $f(c)$ that maps any m-bit input challenge $c \in \{0,1\}^m$ to an n-bit output response $r \in \{0,1\}^n$. The tuple (c, r) is termed as a *challenge-response* pair (CRP).

Depending on the number of unique CRPs that a PUF system is able to generate, we can classify PUFs into *Weak* or *Strong* PUFs.

1.1.1 Weak PUFs

PUFs which can produce only a limited set of unique CRPs are classified as *Weak PUFs*. This limitation requires that the generated responses be *secret* as an attacker can clone the PUF easily if the CRPs are exposed. Weak PUFs are primarily useful

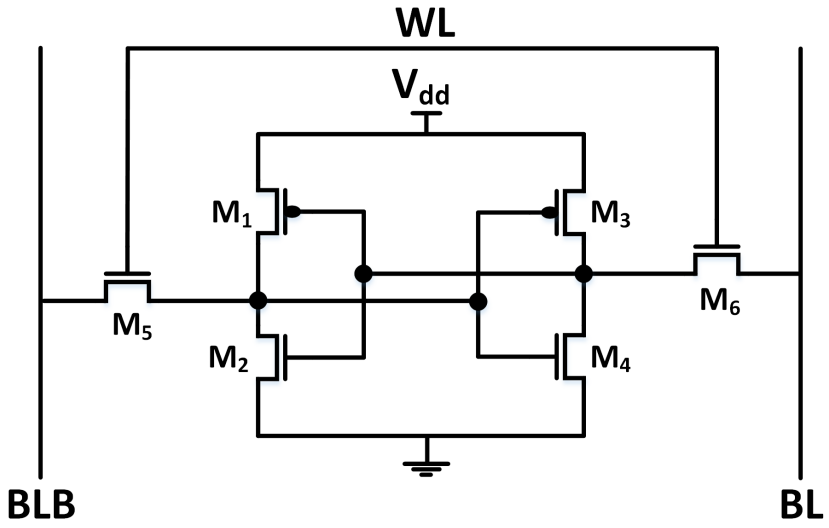


Figure 1.1: A six transistor SRAM cell

for generation of secret keys and can replace NVM-based key storage due to the previously mentioned security against invasive attacks.

The most widely studied Weak PUFs are SRAM PUFs [50, 55, 57] which utilize SRAMs in embedded memories. An SRAM cell typically consists of two cross coupled inverters whose outputs are connected to the bitlines via access transistors. Figure 1.1 shows a typical 6-transistor SRAM cell. Due to intrinsic process variations, the power-up state of an SRAM cell can randomly settle into either a *logic-0* or *logic-1* value. The state is determined by mismatch due to the process variations in the cell transistors from the manufacturing process. Settlement to *consistent*, yet *random* states allows processing multiple cells' outputs to be used as a key or identifier. Post-manufacturing, the Weak PUF key is recorded during the *enrollment* phase and this key becomes the reference once the device is deployed in the field.

Ideally, an SRAM PUF will produce the same key each and every time it is queried. However, noise can impact the SRAM state during start up and thus, make the PUF unreliable. Specifically, cells with *low* process variation induced mismatch between the cross-coupled inverters are highly sensitive to noise and their outputs can flip

from the correct value. Cells with greater mismatch produce sufficient differential drive to overcome any impact from noise. The sources of noise can be variations in environmental conditions, supply voltage changes and parametric changes due to aging of the transistors. Since Weak PUF outputs have to be reliable for use as secret keys, *error correction* techniques are critical for ensuring proper operation.

1.1.2 Strong PUFs

Strong PUFs, in contrast to Weak PUFs, are capable of producing an extremely large number of CRPs due to the more complex mapping between the challenges and responses. The large set of unique CRPs makes Strong PUFs a good candidate for authentication applications as an attacker, ideally, cannot clone the PUF by intercepting a few CRPs. Further, the authenticator need not repeat the same challenge between successful authentication events, preventing a *replay* attack using previously recorded CRPs. Also, the authentication is more resilient in the presence of unreliable responses as more responses can be queried and a threshold operation used to decide the outcome of the event. Ideally, the Strong PUF response from a previously unseen challenge cannot be predicted by an attacker. In practice, research has shown that utilizing machine learning (ML) algorithms on a limited set of CRPs, it is possible to build a *prediction model* which can simulate the PUF and output responses [107]. If the prediction accuracy is sufficiently high, the software model has successfully cloned the Strong PUF and can authenticate itself as the legitimate PUF with a large probability of success. This breaks the 'unclonability' property of a PUF. Hence, ensuring low learning accuracy is important for a Strong PUF.

One of the earliest Strong PUFs proposed was the Arbiter PUF [75], as shown in Figure 1.2. It consists of multiple delay elements in each stage, challenge inputs to select the signal path and an Arbiter at the end that outputs a 0/1. When a challenge is applied, two unique paths are chosen by the challenge bits at the switches of each

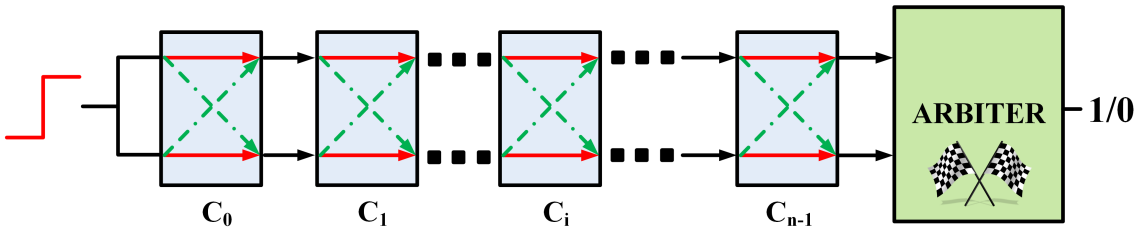


Figure 1.2: Arbiter PUF

stage and a common signal is allowed to race through these paths to the final Arbiter. The Arbiter resolves the response to *logic-0* or *logic-1* depending on which signal arrives first. An exponential number of path pairs can be created based on the input challenge and hence, produce an exponential number of unique challenge-to-response mappings are possible. As each delay element is affected by process variation, the same challenge can produce different outputs across different instances of the PUF. It should be noted that the final delay of the signals that arrive at the Arbiter is a linear sum of individual stage delays. Hence, machine learning techniques were able to model the Arbiter PUF with high accuracy [107]. A more complex challenge-response mapping is needed to protect against modeling attacks.

1.1.3 Quintessential PUF Properties

Irrespective of the PUF classification, PUF circuits are expected to exhibit certain salient features in terms of *uniqueness*, *reliability* and *unpredictability*. Along with these PUF related metrics, other standard circuit metrics such as area, power and speed should also be satisfied.

1.1.3.1 Uniqueness

Each PUF implementation has to exhibit a high degree of uniqueness across various PUF instances and across various challenges to the same PUF. Failure to do so can adversely affect the ability to identify a PUF uniquely among a large number of instances. Low uniqueness can stem from systematic bias and from the PUF

circuit's inability to harness the process variations effectively. However in practice, ensuring high uniqueness can incur a high yield loss for a manufacturer as bias in the manufacturing process cannot be completely eliminated.

1.1.3.2 Reliability

The circuit characteristics of a PUF can be affected by sources of noise such as environmental variations. This can result in erroneous responses to the same challenge across different queries. The reliability of a PUF response in the presence of noise is critical for Weak PUFs. In Strong PUFs, susceptibility to noise increases the number of responses needed to successfully authenticate a device and the authentication threshold will have to be less than 100%. Further, the accuracy requirements for machine learning algorithms is reduced for a more unreliable Strong PUF, adversely impacting the security of the PUF.

1.1.3.3 Unpredictability

Unpredictability or unclonability is the most important requirements for PUFs. In Weak PUFs, the generated key should not be exposed to unauthorized parties. In Strong PUFs, the security of the PUF is dependent on the complexity of modeling the challenge-response mapping using learning algorithms. If the mapping is easy to model, a software clone can be created to act as a rogue PUF, indistinguishable from the authentic PUF.

1.2 Hardware Obfuscation

Increasing costs for manufacturing has led to rise of fabless companies that design Integrated Circuits (ICs) and outsource manufacturing to foundries. Typically, for an IC, there is a non-recurrent design cost, incurred by the designer, and manufacturing cost, for as long as the IC is in production. The manufacturing cost is kept manageable as the foundry manufactures multiple designs from different intellectual property (IP)

owners for improved efficiency. Also, many products can contain multiple IPs from different vendors on the same die.

This fabless structure of the industry has led to an increased risk of IP theft as the designer has to reveal the complete design to a foundry for manufacturing. Another source of concern is that to further reduce costs the foundries are located in countries with inexpensive labor but, with ambiguous IP protection laws. Hence, this creates a situation for malicious actors to steal the IP, cutting down on design costs, and produce counterfeits for profit. Also, rival companies may attempt to learn about proprietary designs to gain advantage in the market. This leads to a significant loss of revenue for the designer. Current assessments estimate this loss to be to the tune of $4 billion annually [10]. There is also an increased security risk as the attackers may corrupt the original design for their own purposes before sale. Such a harmful black-market product can severely damage the reputation of the design company leading to further losses.

To protect their revenue stream and their products from tampering, fabless companies have proposed many solutions for increased security. On the manufacturing side, fabless companies and government agencies like Intelligence Advanced Research Projects Agency (IARPA) [13] have proposed using *split manufacturing* to secure against attacks. The design layout is split into Front End Of Line (FEOL) and Back End Of Line (BEOL) layers. FEOL layers are fabricated by foundries with advanced feature capabilities. The wafers are shipped to a trusted foundry where BEOL layers are fabricated. The trusted foundry costs less to build and maintain. Hence, it can be in-house or shared among a group of fabless companies. Many other solutions have been proposed like hardware metering[106], watermarking or tamper-proofing. One area of IP security research that has received much attention is *hardware obfuscation*. The goal of obfuscation is to hide the true functionality of a design. Obfuscation techniques have been proposed at various levels of design abstraction and can vary

from introduction of additional gates to lock a circuit [106] to system level techniques [31]. Obfuscation can also involve the creation of camouflaged cells [33, 101] whose function can be hard to determine.

1.3 Scope of this Work

In this dissertation, we seek to explore improved PUF designs with particular focus on reliability of Weak PUFs and on unpredictability for Strong PUFs. We also explore hardware obfuscation techniques to aid in the prevention of theft of intellectual property.

In particular, our contributions include:

- Designed alternatives to simple SRAM PUFs that exhibit greater reliability in the presence of thermal noise. This results in significant savings in terms of error correction circuitry required to create the final robust keys.

- Proposed an intelligent accelerated aging methodology to further aid in increasing reliability of Weak PUFs while reducing the aging time overhead.

- Leveraged Weightless Neural Network (WNN) architecture and Weak PUFs to create a Strong PUF. Multiple variants were created and analyzed to realize a Strong PUF that exhibits a high resilience to machine learning attack while maintaining high uniqueness and reliability.

- Identified various avenues for visual information leakage from a physical IC layout and proposed techniques to obfuscate visual circuit information.

- Leveraged advantageous, intrinsic properties of FinFETs to create camouflaged cells capable of implementing multiple logic functions without change in cell layout and being resistant to reverse engineering.

1.4 Dissertation Outline

The dissertation is organized as follows: in Chapter 2, we discuss the SRAM circuit alternatives for increased reliability. In Chapter 3, an efficient accelerated aging methodology for increasing the reliability of multiple Weak PUF designs is explored. Chapter 4 illustrates various Strong PUF architectures based on Weightless Neural Networks and Weak PUFs and analyzes the proposed variants against PUF metrics. Chapter 5 describes a Strong PUF system built on reliable Weak PUFs as an entropy source and linear-feedback shift register (LFSR) based mechanism to provide a large set of robust keys. In Chapter 6, we explore protection against design reverse-engineering using visual information in physical layouts. Chapter 7 explores the utilization of multiple threshold FinFETs to create lightweight camouflaged cells. Finally, chapter 8 provides a conclusion for this dissertation work

CHAPTER 2

IMPROVING RELIABILITY OF WEAK PUFS VIA CIRCUIT TECHNIQUES

2.1 Introduction

As mentioned in Section 1.1, in practical applications, PUF characteristics are influenced by environmental factors and aging. The PUF outputs are susceptible to noise and hence, the reliability of a PUF cell must be addressed. As the principal use of Weak PUFs is in key generation/identification, post-processing is typically employed to create extremely reliable keys from PUF outputs. A host of techniques have been proposed, ranging from temporal majority voting [85] to fuzzy extraction and error correction schemes [80, 81]. These techniques come with significant costs in terms of additional circuitry like counters (majority voting), or require decoding blocks and a large number of initial Weak PUF cells to produce a stable 128-bit key (fuzzy extractors). These strategies for handling errors are often layered atop standard SRAM cell designs to generate reliable keys, but the basic SRAM cell error rate is treated as a given and unchangeable.

In this chapter, we explore alternative cross-coupled designs that can provide a greater sensitivity to intrinsic process variations and thereby, enhance the mismatch between the coupled elements. Enhancing the imbalance makes the design less susceptible to noise. We highlight the significant savings that can be achieved for ECC implementations on account of modifying the Weak PUF cells to be more reliable. We study circuit thermal noise as the main error mechanism as it can affect the circuit in a differential manner, causing an error in the outputs. We present results on the

reliability impacts of environmental factors such as supply voltage and temperature variations, and we show the advantages of using our designs.

2.2 Background

In this section, we discuss previous research into improving the reliability of keys generated by SRAM-based Weak PUFs that involve algorithmic and circuit-based error correction mechanisms.

2.2.1 Fuzzy Extraction and Error-Correction Codes

Fuzzy extraction was initially proposed to derive stable keys from biometric data and also, to authenticate such data [40]. Application of error-correction codes (ECCs) and fuzzy extractors to correct errors in SRAM PUFs involves the generation and use of helper data [28,39,80,81,83], which is made public. While the helper data aids in the recovery of a key from original data in the presence of noise, a stable key of specified length requires the use of a long starting string [28]. As a technology node matures, process mismatch reduces and this increases the intrinsic error of any SRAM PUF implemented in the mature technology node. Hence, the overhead of a longer initial string and larger amount of helper data for fuzzy extraction and error correction to compensate the high error rate becomes increasingly expensive in mature technologies. Thus, reducing the inherent error rate of SRAM cells can accrue significant area and computation savings by reducing reliance on costly error-correction.

2.2.2 Circuit and Fabrication Techniques

Alternatives to ECC involve the pursuit of circuit and device-level technology solutions that either improve the SRAM cell reliability or implement more efficient error detection and compensation schemes. Hofer et al.proposed pre-selecting SRAM cells that possess a greater degree of mismatch in the cell transistor threshold voltages [54]. However, this approach also involved the need for expensive Non-Volatile

Random-Access Memory (NVRAM). Bhargava et al.have proposed technique to improve reliability through Hot carrier injection (HCI) aging [26]. Similarly, effect of aging on PUF reliability has been explored by Garg et al.[46] and Maes et al.[82]. Jang and Ghosh [61] proposed an 8T SRAM PUF with PMOS latch and a low-power 7T SRAM cell with embedded Magnetic Tunnel Junction (MTJ) to enhance the reliability of the PUF in the presence of environmental fluctuations. Su et al. [116] proposed the use of SRAM-like cells with a common centroid layout and resettable logic to reduce the influence of systematic process variations.

The use of temporal majority voting (TMV), burn-in and dark bits evaluation for the purpose of reducing error rates was showcased by Mathew et al.[85]. Design changes were required to enable voting and synchronous design helped improve uniqueness. However, the approach can only correct error rates of $< 8\,\%$ and additional techniques are necessary for practical applications. Adapting voltage ramp-up time to ambient temperature to reduce the error rate of memory PUFs has been proposed by Cortez et al.[36]. The drawbacks are that the auxiliary circuits needed for voltage ramp-up can be area intensive and shaping supply voltage is expensive for designs with large power delivery network.

Ganta and Nazhandali explore alternate configurations of the inverters in the SRAM cell to improve the stability of SRAM cells with respect to variations in temperature and reduce the number of unreliable bits to save on ECC area [45]. Our goal is to study the PUF cell performance at a given temperature in the presence of thermal noise that can cause errors in its output. We seek to explore alternative configurations of the PUF cell that increase process sensitivity and hence, provide greater mismatch between the cross-coupled elements. Bucci and Luzzi [30] presented a differential circuit that captures the process mismatch and amplifies it to reduce the effect of noise. Once the difference has been sufficiently amplified, the differential outputs are

latched to generate stable bits. Our work seeks to create simpler differential circuit design alternatives that can also immunize the cells to noise.

2.3 Weak PUF Design

In this section, the process variation and noise modeling parameters utilized for this work are detailed. Next, we discuss a simple SRAM-like cell design consisting of two-cross coupled inverters and then, explore alternatives that have a greater process variation sensitivity.

2.3.1 Modeling Process Variation

We model manufacturing induced process variations as random parametric variations in threshold voltage (VTH) and channel length of each transistor in a circuit. Also, the parameters are random across PUF instances. The values are obtained from a normal distribution, $N(\mu, \sigma^2)$, where the mean (μ) and standard deviation (σ) are determined based on the technology node used. The geometry of a transistor decides the susceptibility of a device to process variations with larger devices experiencing lesser fluctuations. In terms of threshold voltage, the mean is the default transistor model value and the standard deviation is given by,

$$\sigma_{VTH} = \frac{\sigma_{VTH0}}{\sqrt{\frac{W*L}{W_{min}*L_{min}}}} \tag{2.1}$$

where (W_{min},L_{min}) are the minimum possible width and length of a device, respectively, and (W,L) are the sizes used in the design. σ_{VTH0} is the standard deviation of threshold voltage for the minimum sized device.

In this work, we instantiate the PUF cell designs with 45 nm NCSU FreePDK45 models [6]. Typically, for 45 nm node, a standard deviation of 53 mV for threshold voltage and 10 % channel length variation are considered [4]. However, one of our goals is to analyze PUF performance for *low process variation* corner where cells

have high error rates, providing a significant challenge for PUF design. The low variation scenario is also of great practical importance as it is *representative* of mature technology nodes which exhibit low manufacturing process variations. Hence, to obtain the amplified mean error rate for each design, only the threshold voltage for each transistor is varied with the base value provided by transistor models as the mean and the standard deviation of a minimum sized device set to 5 mV (chosen by us). Equation (2.1) is then used to calculate the standard deviation for any arbitrary sized transistor.

2.3.2 Thermal noise errors

Thermal noise in a transistor occurs due to the random motion of the charge carriers from thermal excitation and can create a random voltage fluctuation in conductors [64,92]. Thermal noise has no correlation among different sample across time and has a near uniform power spectral density. In advanced CMOS technology nodes, short channel effects [49] exacerbate the effects of thermal noise and significantly impact transistor noise performance [119]. The magnitude of thermal noise at any given node can be represented in terms of a normal distribution with 0 mean and standard deviation given by [111],

$$\sigma_{NOISE}(T) = \sqrt{\frac{k_B * T}{C}} \tag{2.2}$$

where k_B is the Boltzmann constant, T is the absolute temperature (Kelvin) and C is the node capacitance (Farads).

2.3.3 Simple SRAM-like Weak PUF (*Ref*)

PUF cells produce random outputs due to mismatch in the cross-coupled inverters' strengths due to manufacturing process variations and noise during evaluation. A Weak PUF cell needs to have reliably identical behavior across multiple evaluations. However, thermal noise affects each node in the circuit differently and hence, can drive one of the cross-coupled inverter's NMOS to an ON state, inducing a logic-0 state

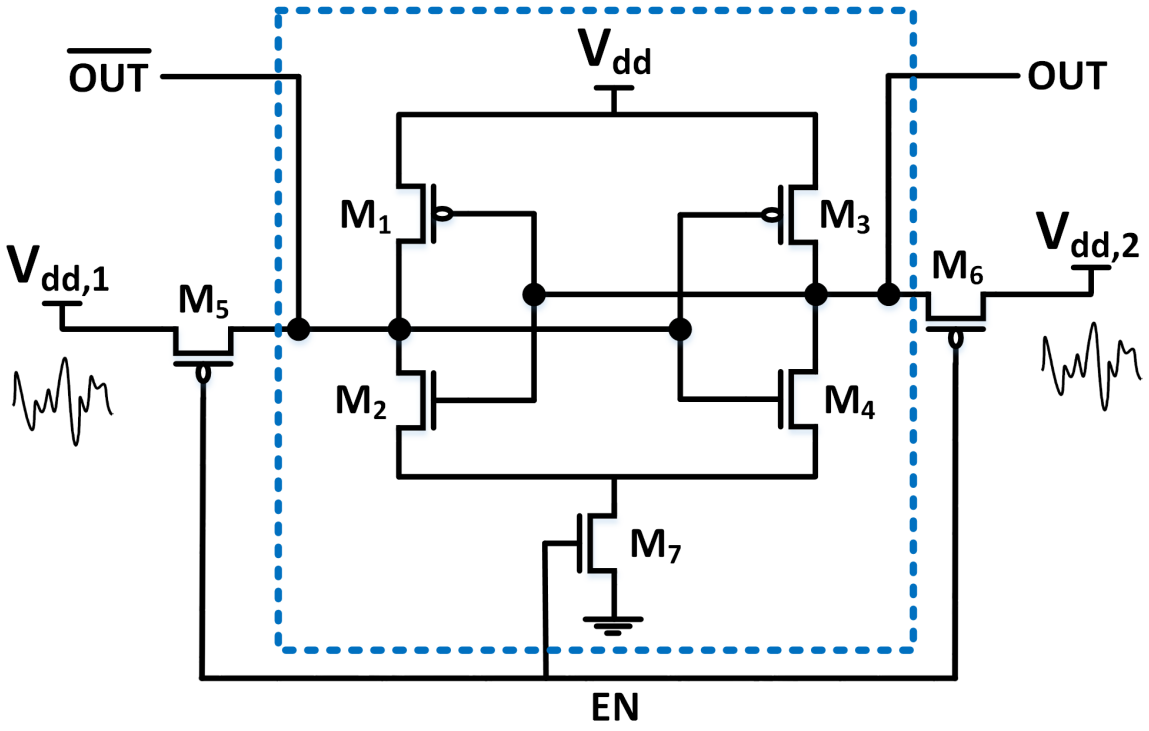


Figure 2.1: Standard cross-coupled inverter PUF cell (*Ref*)

its output. Across multiple evaluations, a cell with inadequate mismatch between the cross-coupled inverters can produce different results and hence, be unreliable as a Weak PUF.

In this work, to facilitate a resettable PUF, we make minor modifications to the basic cross-coupled devices by implementing additional pre-charge/discharge circuitry, as shown in Figure 2.1. The enable signal (EN), at logic-0, first pre-charges the OUT and $\overline{OUT}$ to V_{dd} while keeping the footer NMOS (M_7) in OFF state. The evaluation phase begins by setting EN to logic-1 and depending on process variation, the cell will settle into a particular state. To mimic thermal noise effects, we set the OUT and $\overline{OUT}$ to $V_{dd,1}$ and $V_{dd,2}$. The voltages are obtained from a Gaussian distribution with a mean of V_{dd} and variance given by (2.2). This design is an adaptation of similar circuits proposed to enable TMV [85], Sense-Amplifier PUF [25], and a PUF based on cross-coupled NOR cells [116].

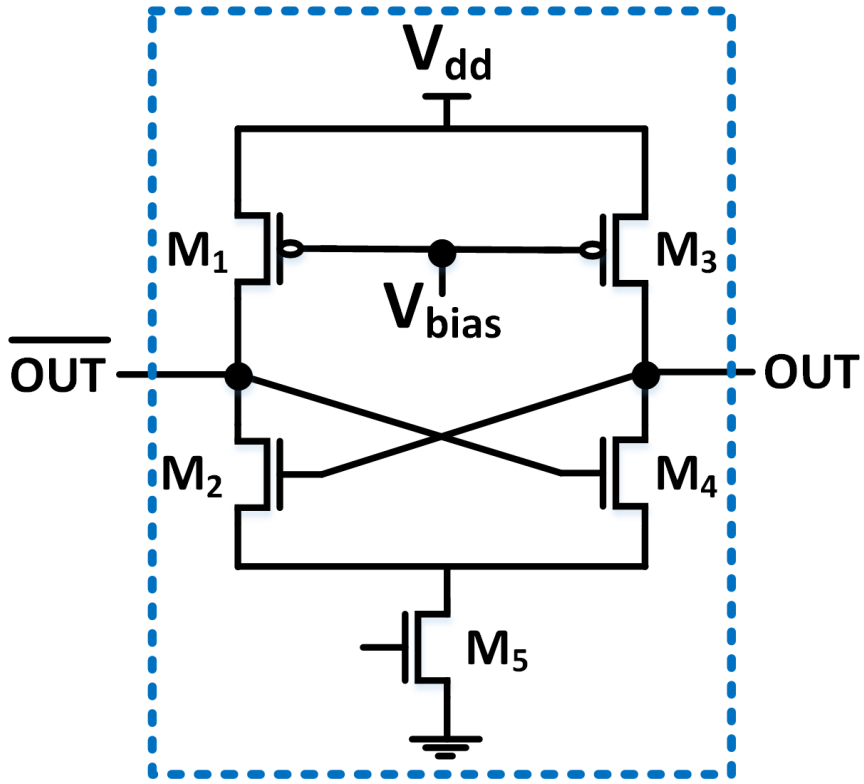


Figure 2.2: PUF cell with only pull-down network and active resistive loads ($D1$)

2.3.4 Study of various Cell designs

Here, we explore alternative PUF cell designs that are more sensitive to process variations, resulting in greater mismatch between the cell cross-coupled elements. The salient features and drawbacks of each alternative are also discussed. Only the cross-coupled elements (dashed boxes) are altered with respect to *Ref* while the overall circuit operation remains same.

2.3.4.1 Simple active loads ($D1$)

The first alternative we consider modifies the PUF cell pull-up network of the previous inverter configuration by connecting a DC source, V_{bias}, to the PMOS gate terminals. This converts the PMOS transistors into active loads whose current outputs depend on their threshold and input voltages. Figure2.2 shows this cross-coupled design, but omits the precharge transistors for conciseness.

In a simple inverter configuration, the currents in both the pull-up and pull-down networks are affected by the inputs and process variation. In a low process variation scenario, thermal noise will affect both networks in a differential manner and has a greater chance of introducing error. By removing the input dependence in one of the networks of each cross-coupled element (M_1 or M_3 in Figure 2.2), we make the current through that network purely dependent on the process variation. The constant source, V_{bias}, is shared by both cross-coupled elements and any noise associated with this source will become common mode noise. The complimentary network is used to drive the feedback loop. Hence, the circuit becomes more process sensitive. Additionally, the footer NMOS prevents high static current flow due to the pull-up network by breaking connection to the ground when the cell is not in use.

2.3.4.2 Parallel active loads ($D2$,$D3$)

We study the case of using parallel active loads, as shown in Figure 2.3. Applying Kirchhoff's Current Law (KCL) at the output node, OUT, we see a linear additive relationship for the currents through the active loads. Taking a case of two active loads in parallel at OUT node, let the currents in the two loads have a normal distribution with unique means (μ_1, μ_2) and variances (σ_1^2, σ_2^2), due to process variations, across a population of PUF cells. The addition of such currents at OUT realize a current with a variance that is the sum of the two variances ($\sigma_1^2 + \sigma_2^2$). This concept can be extended to multiple parallel loads. Hence, theoretically, this approach provides a greater sensitivity to process variation than just a single active load ($D1$), which is the simplest case of the parallel active loads configuration.

The size of the input connected NMOS transistors may need to be larger to effectively sink enough current to drive one of the outputs to a logic-0 in the presence of multiple parallel PMOS loads. Also, complex biasing for the active loads may be needed to further reduce the current that the pull-down network needs to handle.

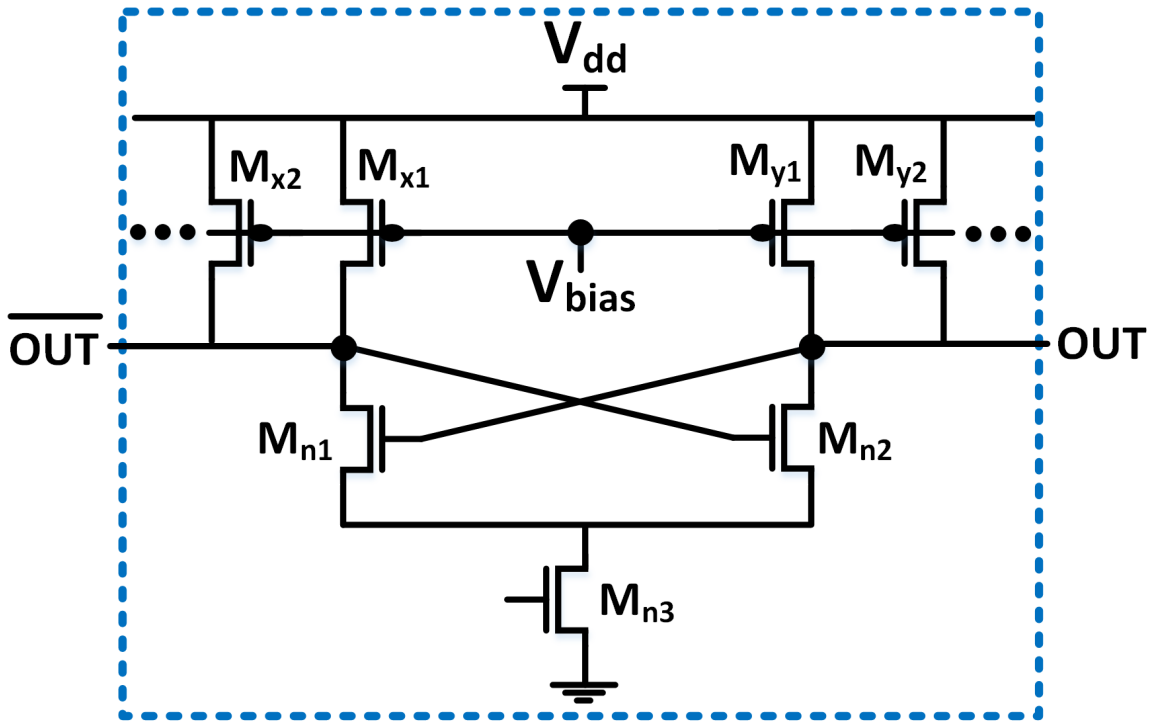


Figure 2.3: PUF cell with parallel active loads ($D2$,$D3$)

The multiple parallel loads and the need for a larger pull-down transistor adds to the area overhead compared the simple case ($D1$).

2.3.4.3 Current Mirror loads ($D4$)

A current mirror has the property of providing a multiplier effect based on the sizing of the mirror transistors. Hence, we explore replacing the simple load with a current mirror load, as shown in Figure 2.4. The current mirror provides two additional venues to boost the process variation effects in the PUF cell: (a) the process variation in the mirror transistors (say M_{x1}, M_{x2} in Figure 2.4) affects the current mirroring factor; (b) the process variation of the bias transistor (M_{x3}) of the current mirror influences the base current that will be mirrored. These combined effects can help increase the mismatch between the cross-coupled elements of the PUF cell and hence, reduce error.

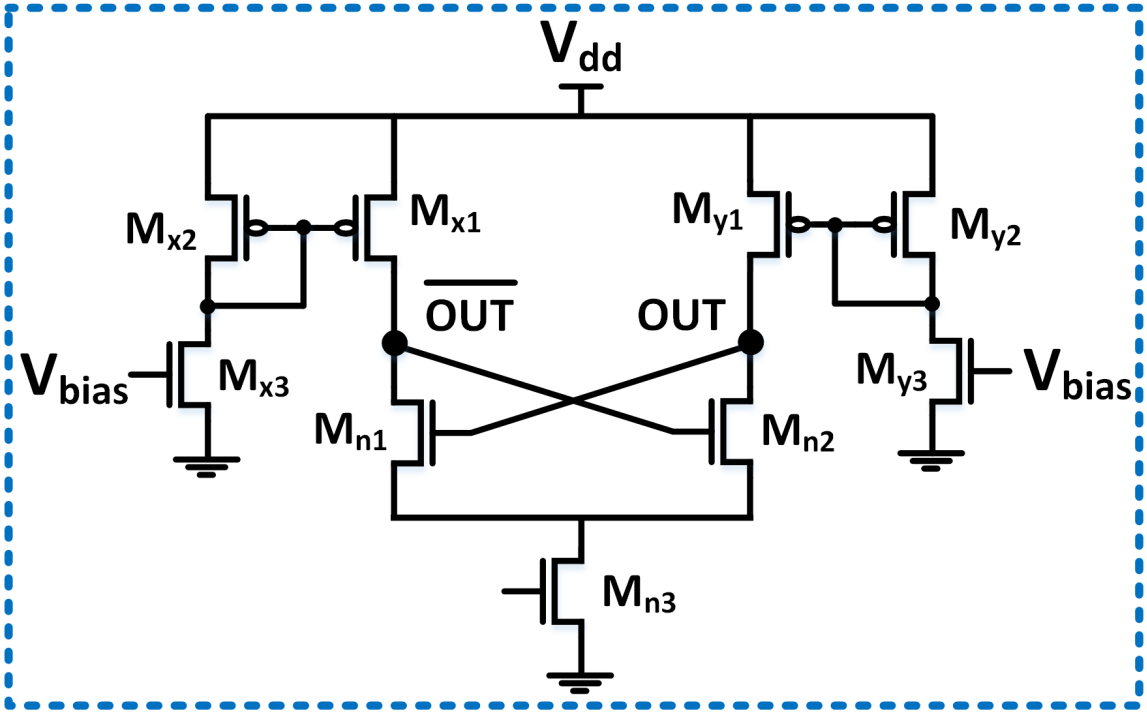


Figure 2.4: PUF cell with current mirror loads ($D4$)

Similar to the aforementioned active load configurations, we need to generate the appropriate bias voltages.

2.4 Results and Discussion

In this section, we will first contrast the reliability performance of various alternatives explored in section 2.3.4 with the simple cross-coupled inverter PUF cell using error rate as the metric. Then, the design *Ref* and select configurations ($D2$, $D4$) are analyzed in greater detail illustrating the advantages of our approach, specifically, in terms of reduction in ECC complexity and the associated area savings.

2.4.1 Error rate comparison

Our primary goal is to compare the reliability of each proposed alternative ($D1 \rightarrow D4$) with respect to *Ref*. For this purpose, we assume room temperature (25 °C) and obtain the output node capacitance for the cross-coupled elements of each design and

calculate the standard deviation of thermal noise to be used for simulation according to (2.2).

We simulate a cell instance over $1,000$ thermal noise value pairs to mimic $1,000$ power-ups of an SRAM-based Weak PUF cell. The output voltages are then classified as either logic-0 or logic-1 based on a certain threshold, $\frac{V_{dd}}{2}$ V in our case. The error rate for each instance is the percentage of trials in which the logic output of the cell flipped compared to the base case with no thermal noise.

The footer NMOS is sized larger allowing the circuits to achieve near ground-level voltage. The pull-up and pull-down transistor are sized to maximize process sensitivity, according to (2.1), and the sizes for each design configuration are listed in Table 2.1. The bias voltage, V_{bias}, is set to 0 V for design $D1$ and the bias is $\frac{V_{dd}}{2}$ V for the rest.

The mean error rates obtained by simulating $10,000$ instances of each design, are given in Table 2.1. As explained in Section 2.3.1, we use a low process variation scenario ($\sigma_{VTH} = 5\,\text{mV}$) to generate the results. All the design alternatives perform significantly better than the design *Ref* with $4\times$ to $9\times$ reduction in error rates. Furthermore, the parallel active loads designs ($D2$, $D3$) showcase the gains from increasing the number of parallel transistors.

2.4.2 Flipping point analysis

To further illustrate the improvements afforded by the new designs over the simple cross-coupled inverter cell, we conduct an experiment to compare the *Ref* design against the alternatives $D2$ and $D4$, in terms of the amount of noise needed to change the base state of a cell. First, we find the state of the cell outputs without the presence of noise and then, add a DC offset to the corresponding pre-charge supply of one side until finding the minimum voltage that can flip the cell outputs. We term this voltage as the *flipping voltage*. We repeat the experiment for each design across $10,000$ cell

Table 2.1: Mean Error Rate of PUF cell configurations considering Low Process Variation

Configuration (*Identifier*)		**Transistor Sizing (nm)**		**Mean Error Rate (%)**
		PMOS	**NMOS**	
Simple (*Ref*)		90	90	20.79
Single Active Load (*D1*)		90	180	5.14
Parallel Loads	**2 (*D2*)**	90	180	4.53
	3 (*D3*)	90	180	2.23
Current Mirror Load (*D4*)		180	$M_{(x,y)3}$: 90 $M_{n(1,2)}$: 180	3.79

instances using the full process variation parameters and a 300 mV maximum offset threshold. The histogram of the recorded *flipping voltages* for both *Ref*, $D2$ and $D4$ designs are shown in Figure 2.5. The distributions highlight that, on average, a much larger noise perturbation is needed to affect the output states of $D2$ and $D4$ than the *Ref* design. Consequently, more instantiated cells of $D2$ and $D4$ are likely to have zero or very low error rates compared to *Ref* case. We should note that this experiment offers more of a qualitative insight into the noise resilience of the circuits as it is extremely unlikely that the differential thermal noise values will reach over 20 mV for any of the instantiated designs.

2.4.3 Voltage and Temperature Variations

To assess the impact of variations in supply voltage and temperature, we instantiate 100,000 instances, each, of *Ref*, $D2$ and $D4$ using the full process and channel length variation parameters [4]. For each design, we consider 25° C and 1 V as the nominal temperature and supply voltage settings, respectively. The outputs are obtained by sweeping either the temperature or the voltage, keeping the other constant. We do not introduce circuit thermal noise in these simulations in order to sensitize the

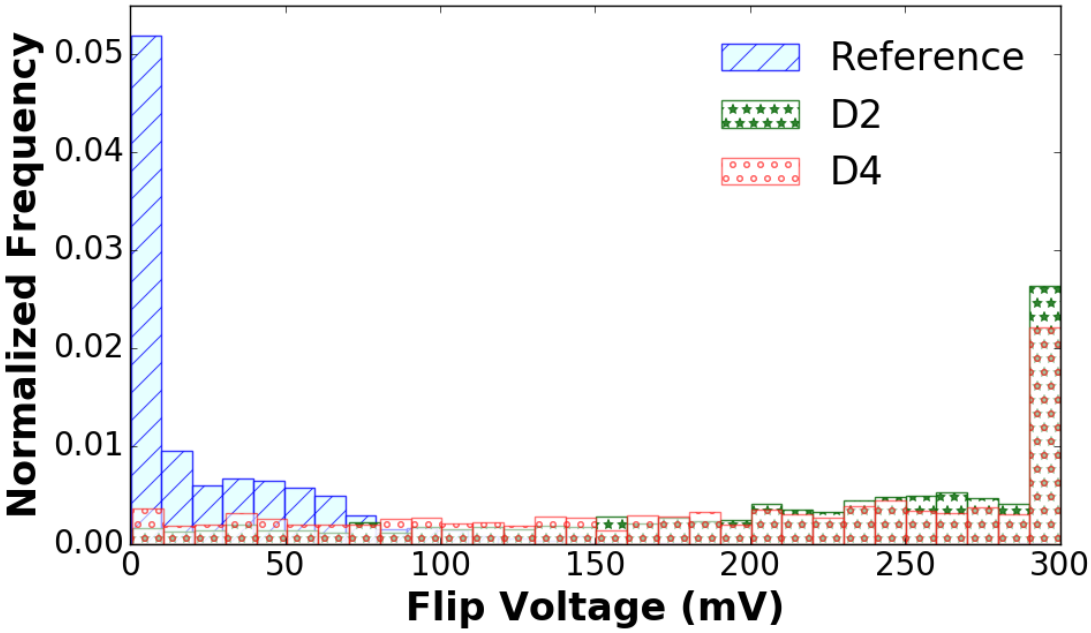


Figure 2.5: Flipping voltage comparison between *Ref*, *D2* and *D4*

outputs only to variations in voltage/temperature. For each design, the $100,000$-bit binary result at each temperature/voltage is compared to the binary string obtained under nominal conditions. The percentage of errors for temperature and voltage sweeps are plotted in Figure 2.6. We see that $D2$ and $D4$ have a lower number of bit errors compared to *Ref* design across various voltages/temperatures. In addition to being highly resistant to thermal noise, this result shows the proposed PUF alternatives to perform better than the *Ref* design under environmental variations.

2.4.4 Reducing ECC circuitry overhead

As the inherent cell error rate is reduced considerably by using the alternative cell designs, we can make appreciable gains by reducing the amount of post-processing required to make a highly reliable Weak PUF implementation (error rate $\leq 10^{-6}$). To obtain an idea of the amount of savings to be expected, we examine the *Ref*, $D2$ and

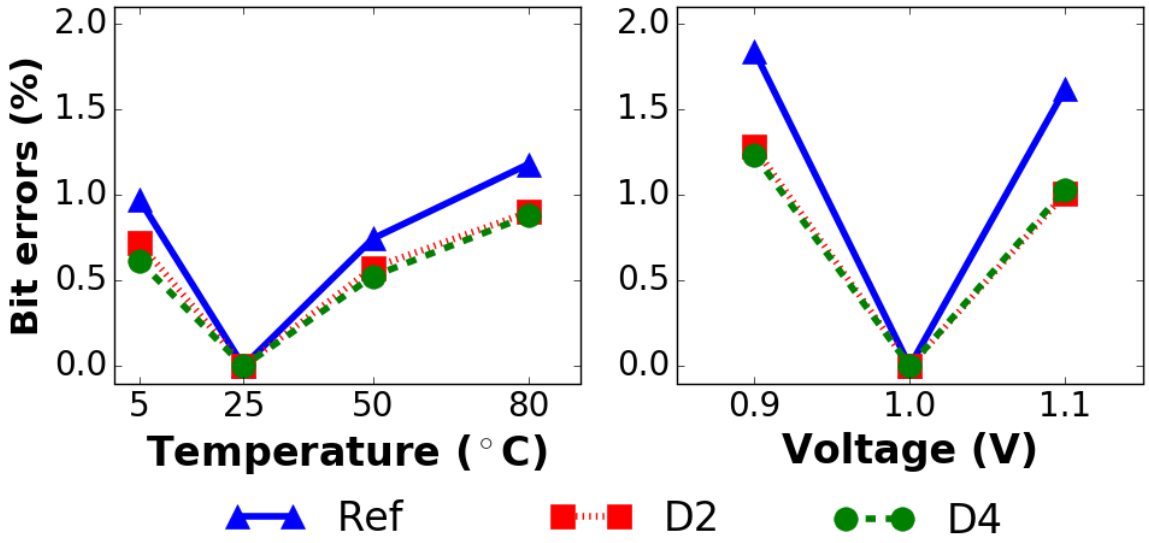


Figure 2.6: Percentage of bit errors in 100,000-bit strings for *Ref*, *D2* and *D4*, compared with (25° C, 1 V) case

$D4$ designs. We instantiate 10,000 different cells of each design and obtain the error rates as described in section 2.4.1 under thermal noise and with full process variation.

Bösch et al.[28] have conducted extensive studies on the implementation of ECC in hardware for the purpose of extracting stable keys from SRAM-based Weak PUFs. The authors explore the use of various error correction codes to correct errors in a binary string with certain error probability (p_b). However, Roel Maes presented a new 2-parameter error model that highlighted the fact that a population of SRAM cells will possess a distribution of error rates [78]. The author shows that the more accurate 2-parameter model can produce surprisingly different results from the simple homogeneous error rate assumed in earlier works when used to estimate the key failure rates for an SRAM-based Weak PUF to generate a stable 128-bit key. Following the same procedures detailed in their work, we estimate the parameters of the 2-parameter error model by using the error rates obtained from the selected cell designs. The curve-fitted probability mass functions (PMFs) are shown in Figure 2.7, using continuous lines for clarity. We see that $D2$ and $D4$ each have over 95 % of the cells producing 0 errors in 1000 trials, compared to 83 % for *Ref* design.

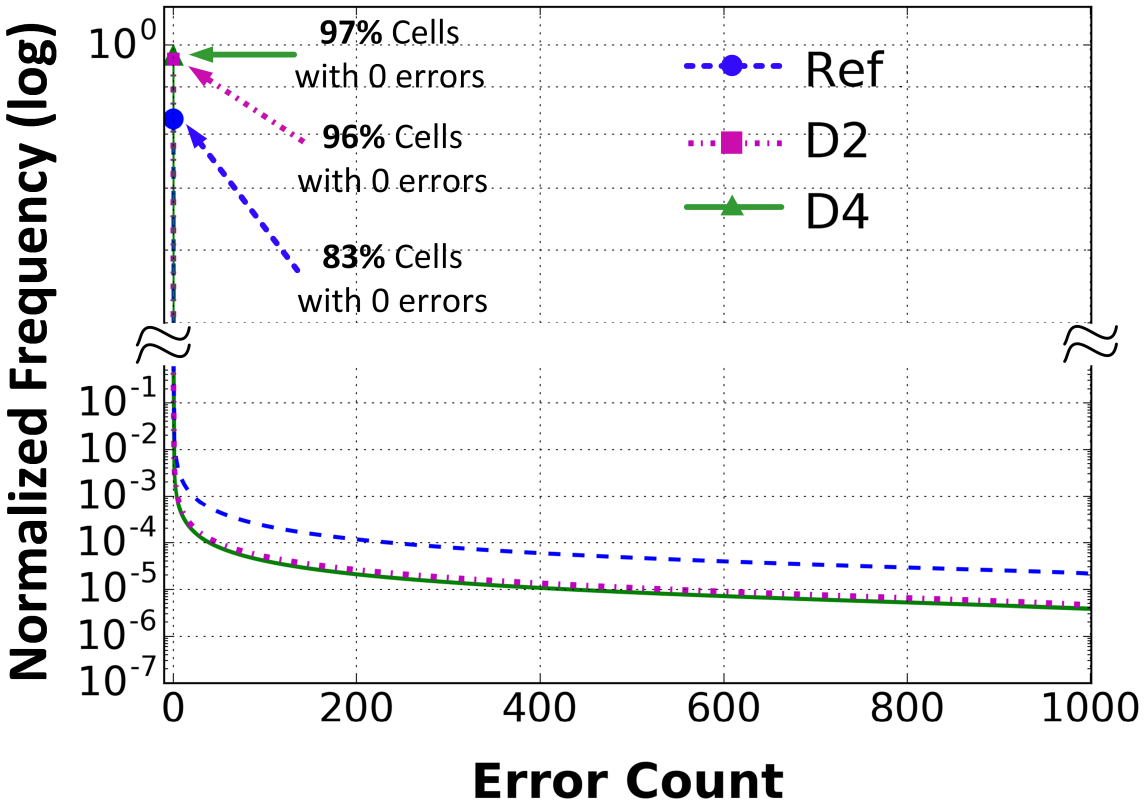


Figure 2.7: Curve-fitted PMFs for *Ref*, $D2$ and $D4$

Our final target is to obtain a robust key of 128 bits (with error rate $\leq 10^{-6}$). We consider the Reed-Muller (RM) ECC codes for which we can estimate the corresponding areas using gate counts provided by Bösch [28] and the areas of the relevant gates from the Nangate 45 nm cell library [5]. By simulating 1 *million* key generation systems and altering the ECC code parameters we found the smallest codes that would enable more than 99 % of the simulated keys to have an error rate $\leq 10^{-6}$. Then, the areas are calculated for the best code implementations. The unit PUF cell area for the *Ref*, $D2$ and $D4$ designs were found to be $1\,\mu m^2$, $1.25\,\mu m^2$ and $1.5\,\mu m^2$, respectively. The final results and relevant details are tabulated in Table 2.2. The alternatives are able to reduce the area requirements by more than 60 %.

The above simulation demonstrates the advantages of our approach in terms of ECC implementations. It may be possible to achieve better results with different

Table 2.2: Area savings from ECC implementations

Design	**ECC code [*n*,*k*,*t*]**	**# PUF cells needed**	**Area (μm^2)**	**Area Savings (%)**
Ref	RM[64,7,15]	1408	2286.3	—
*D*2	RM[16,5,3]	352	824.1	**63.96**
*D*4	RM[16,5,3]	352	912.1	**60.11**

ECC implementations. Although different error correction schemes and codes may be used, the results shown are not unique to this scheme, and it is expected that the increase in cell reliability should produce similar area savings in any such techniques.

2.5 Conclusion

Weak PUFs, especially SRAM-based, have gained popularity for application in key generation. However, such keys can suffer from reliability issues due to system noise. To address this, various ECC and fuzzy extraction techniques have been proposed. These schemes depend on a large number of initial bits sourced from Weak PUFs to generate stable keys due to the unreliability of the PUF cells. In this work, we address and improve the Weak PUF reliability by presenting new designs that harness inherent process variations to a greater degree than conventional SRAM-like cells. Our designs perform well in mature technology nodes with low process variations, and with respect to various noise sources. Also, lowering the error rate of the PUF cell allows us to scale down ECC resources required to generate a stable key.

CHAPTER 3

IMPROVING RELIABILITY OF WEAK PUFS VIA INTELLIGENT ACCELERATED AGING

3.1 Introduction

Among the prior approaches to improve PUF reliability, accelerating device aging or burn-in has received increasing attention [26, 46, 85]. To accelerate aging, devices are subjected to temperature and voltage stress in a burn-in chamber. For SRAM-like Weak PUFs, the burn-in process can be beneficial towards increasing the inherent mismatch between the cross-couple inverters so that a PUF output attains greater immunity to noise.

While burn-in is beneficial, it can significantly inflate production costs due to long baking times to maximize the number of reliable integrated circuit (ICs). The straight-forward way to determine the bake times is to account for the worst-case design corners. This can prove detrimental to utilization of PUFs in *low-cost* applications, like Smart Cards, and for high volume manufacturing. Hence, there is a compelling need to reduce the burn-in time. Integrating a mechanism for the IC to provide certain information to the manufacturer to aid in calculating the *minimum* burn-in time, without compromising the security of the PUF, can prove advantageous and offer considerable increase in manufacturing throughput.

In this chapter, we present a method to reduce the burn-in time by quantifying the minimum burn-in requirements for each Weak PUF cell. A low-cost proxy is used to represent the inherent cross-coupled inverter mismatch in terms of the PUF error rate. The error rates of the instantiated PUFs in an IC are measured and used to decide

the burn-in requirements. A *low-overhead* architecture is presented to automate the collection of necessary data. Also, the effect of alternate SRAM-like PUF designs and different transistor technology implementations on the burn-in process are analyzed.

3.2 Background and Motivation

In this section, we discuss some relevant background with regards to device aging and burn-in. We also discuss temporal majority voting (TMV) technique in detail as it will be utilized in our methodology. Prior research regarding previous techniques for improving Weak PUF reliability has been discussed in Section 2.2.

3.2.1 Temporal Majority Voting

The area overhead from implementing traditional ECC blocks and the number of initial PUF bits required scales superlinearly as the error rate increases. A simple, circuit-based way to reduce the error rate using Temporal Majority Voting (TMV) has been explored by Mathew et al.[85] and Xiao et al.[129]. Mathew et al.[85] also explored burn-in and dark bits evaluation for the purpose of reducing error rates. Design changes were required to enable voting and synchronous design helped improve uniqueness. However, the approach can only correct error rates of $< 8\,\%$ and additional techniques are necessary for practical applications.

In this work, we will assume a simple counter-based TMV for error correction during regular operation of the PUF. As an example, a simple 4-bit counter based TMV counts from 0 to 15 and hence, can be used for 15-way voting. If the resultant value after 15 evaluations of the cell's response is greater than 8, then the final value can be classified as 1 or else it can be classified as 0. The mathematical model of the TMV is a binomial counting process and hence, the reduction in error rate can be calculated analytically. For example, a PUF cell whose error rate is $1 - p$ produces a final error rate $P_e(N)$ given by,

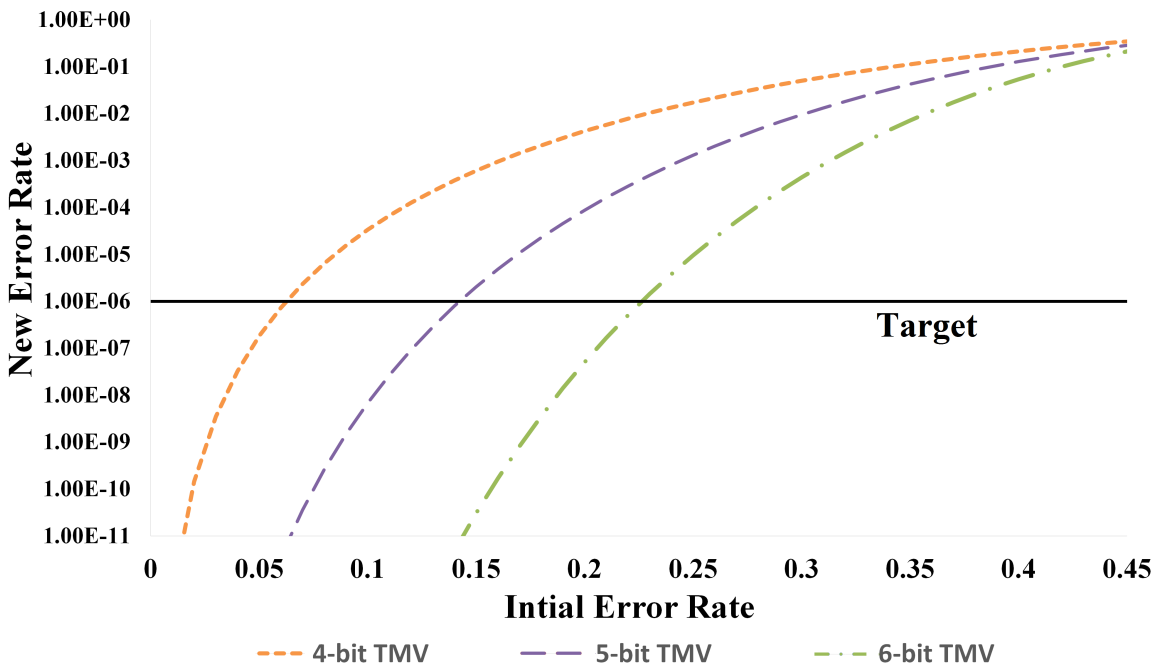


Figure 3.1: Error rate reduction due to Temporal Majority Voting

$$P_e(N) = \sum_{m=k}^{N} \binom{N}{k} p^m (1-p)^{N-m} \tag{3.1}$$

where $k = (N+1)/2$ (as N is odd) when an N-way voting is used [74]. The circuit implementation of the TMV typically consists of an n-bit counter where $N = 2^n - 1$. The counter is incremented by 1 if the response from the PUF cell is 1.

Using (3.1), we plot the initial and final error rates for 4-bit (15-way), 5-bit (31-way) and 6-bit (63-way) TMVs in Figure 3.1. The selection of the TMV would be based on the maximum error rate that the system needs to correct (final error rate $\leq 10^{-6}$). For this work, we utilize a 4-bit TMV for regular operation, noting that it can correct a maximum error of 6 %.

3.2.2 Negative Bias Temperature Instability

Transistor aging has become a significant reliability concern for current CMOS technology. Among various aging mechanisms, Bias Temperature Instability (BTI)

is considered the dominant aging mechanism, causing the threshold voltage of the transistor to increase. There are two BTI mechanisms: (i) Negative BTI (NBTI) affecting the PMOS transistors and (ii) Positive BTI (PBTI) affecting the NMOS transistors. NBTI results from continuous trap generation in Si-SiO_2 interface when a negative voltage is applied to the PMOS gate (stress). Under stressed operating conditions (i.e., On-state, negative gate bias at elevated temperature and supply voltage), Si-H bonds near the interface continue to break and generate interfacial traps which contribute to an increase in V_{th}. Due to the V_{th} degradation, NBTI results in poor drive current, lower noise margin and shorter device lifetime.

The NBTI-induced threshold voltage shift is a function of supply voltage, temperature and many technology parameters. Various models have been proposed in the literature to accurately estimate the threshold voltage degradation ΔV_{th} due to NBTI. Kang et al.proposed a compact threshold voltage degradation model considering the temporal NBTI variation in short channel devices [67]. Paul et al.showed that the maximum circuit delay degradation due to NBTI closely follows the same fractional power dependency to time as the V_{th} degradation with an appropriate scale factor [97]. Kumar et al.have proposed an efficient AC NBTI model for circuit simulations [73]. Vattikonda et al.have proposed a further improved circuit compatible NBTI model to consider AC relaxation effects and technology dependent parameters [123].

FinFET: The NBTI model for FinFETs is the same as (3.2.2). The effect of NBTI and mitigation techniques for FinFET SRAMs has been explored in detail by Wang et al.[128].

According to the Reaction-Diffusion (RD) model [23], the BTI-induced threshold voltage shift is:

$$\Delta V_{th}(t) = \left(\frac{\sqrt{K_v^2 \alpha T_{clk}}}{1 - \beta_t^{1/2n}} \right)^{2n}$$

$$\beta_t = 1 - \frac{2\xi_1 t_e + \sqrt{\xi_2 C(1-\alpha) T_{clk}}}{2t_{ox} + \sqrt{Ct}} \tag{3.2}$$

$$K_v = f(V_{dd} - V_{th}, T)$$

where α is the duty cycle, T is the temperature, T_{clk} is the clock cycle, and other parameters are technology parameters previously defined in [23].

3.2.3 Burn-In (Accelerated Aging)

IC designers and manufacturers are concerned about quality and reliability over a product's lifetime. To ensure economic viability, it is desirable to remove defective devices from the population before shipping them to the customer. Consequently, many ICs undergo a *burn-in* process after fabrication to accelerate failures that manifest in early-life which are primarily caused by process and manufacturing defects. However, under burn-in conditions, increased junction temperature (average temperature of the silicon substrate) increases the leakage current and increased leakage current further increases the junction temperature. Thus, manufacturers try to control the junction temperature by removing the heat from the IC. If the rate of heat generation becomes greater than the rate of heat removal, junction temperature starts increasing. This condition is called *thermal runaway* [122]. It has been shown that the setup for burn-in conditions must evolve by reducing either the ambient temperature or the thermal resistance or a combination of both of them. For example, in 130 nm process technology, the junction temperature should be kept below 110 °C with a thermal resistance of 0.5 °C/W and ambient temperature of 80 °C to avoid thermal runaway [122].

In the case of a Weak PUF, the response can be made more reliable by increasing the magnitude of the difference in the threshold voltages of the two PMOS devices in

the cross-coupled inverters (M_1, M_3), in Figure 3.4. One such method of improving the reliability is to exploit NBTI aging effects to *reinforce* the desired (or "golden") response of the PUF cell. This is done by finding the *golden* outputs (OUT, $\overline{OUT}$) of the PUF cell and forcing the opposite values onto them via the access transistors (M_5, M_6). Increasing the temperature and/or applying voltage stress for a certain amount of time accelerates the aging of the devices in a beneficial manner [26, 46, 85]. Hence, by improving the PUF reliability via burn-in, the number of defective ICs are reduced.

3.2.4 PUF Reliability using accelerated aging

The aging effect on PUF reliability has been studied extensively by Garg et al.[46] and Maes et al.proposed techniques to counter the effects [82]. Bhargava et al.[26] utilized aging via Hot Carrier Injection (HCI) aided PUF reliability.

3.3 Methodology

In this section, we discuss the PUF system design to enable the IC to output the maximum error rate observed. Later, we elaborate on a mechanism to correlate the error rate to the inherent mismatch in the PUF cell that allows us to find the burn-in time required to make all the PUF cells reliable.

3.3.1 Weak PUF System Design

In Figure 3.2, we describe the proposed system to measure the maximum error rate in a PUF. This was initially described in our paper [59]. The system consists of a PUF cell array that is connected to an array of multiplexers (*Muxes*). These muxes will direct the PUF outputs to the relevant counters based on the mode of operation. A *Central Control* unit oversees the entire operation of the PUF system. The circuit operates in two modes: (*i*) Design for Reliability (DfR) mode and (*ii*) Regular Operation (OP) mode.

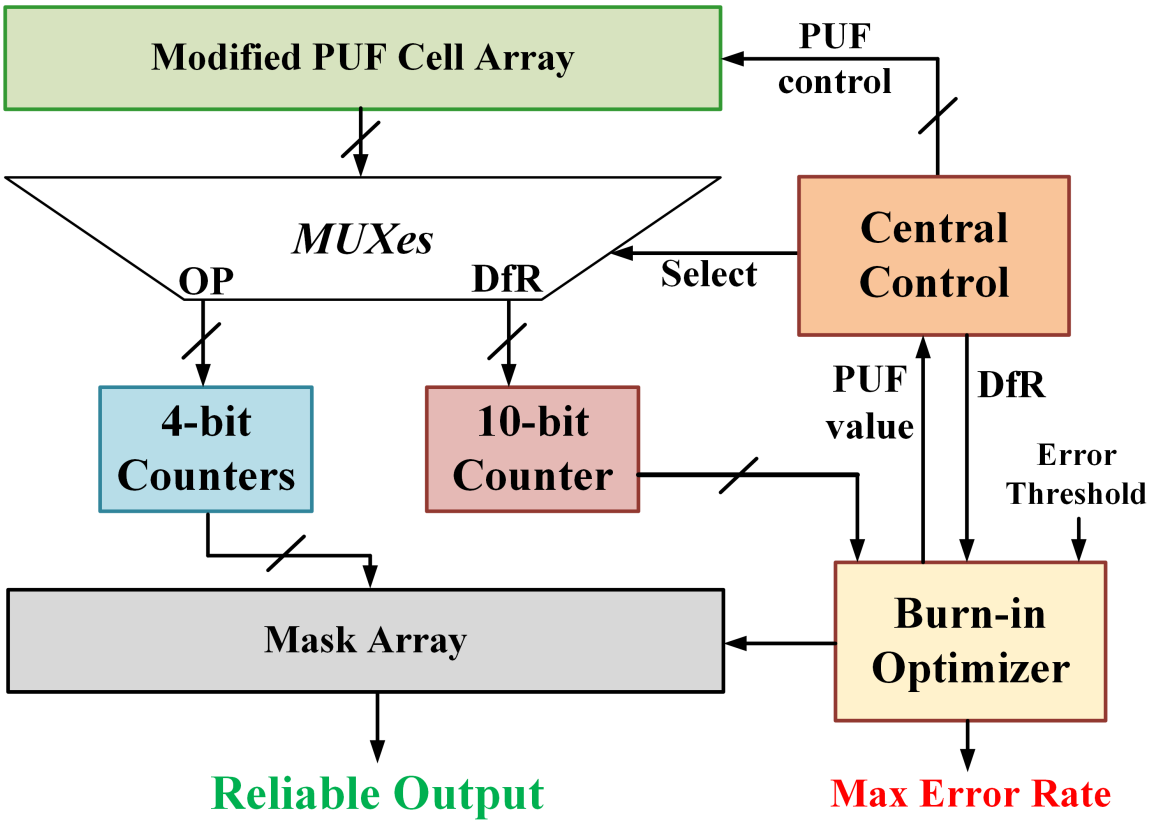


Figure 3.2: Block Diagram of the proposed reliability enhancement scheme (from [59], our earlier paper)

(*i*) *Design for Reliability (DfR) mode*: Prior to burn-in, the circuit is initialized into (*DfR*) mode to obtain the maximum error rate among the PUF cells. The control unit directs each PUF output via the MUX array to a 10-bit counter and performs 1024 pre-charge and evaluations cycles on each cell. During every evaluation, the counter increments if the output is *logic-1*. The large counter allows us to get an accurate measurement of the error rate of a cell. Also, a single 10-bit counter is enough as this process is carried out prior to burn-in and is not time intensive like burn-in. Even if a PUF cell takes 1 ns for each evaluation, then each cell error rate is obtained in $\approx 1\,\mu$s.

The counter value is fed to the *Burn-in Optimizer* unit that first determines the correct PUF cell output. If the count is below 512 (ideally 0), then the correct

output is determined to be logic-0. If the counter is between 0 and 512, then this count becomes the error rate for the cell (out of 1023). For logic-1, the count would be > 512 (ideally 1023). To find the error rate, we subtract the current count from 1023. The estimated logic value (PUF value) of the PUF cell is fed back to Central Control unit which then writes the opposite value to the PUF cell for the purpose of burn-in.

The *Burn-in Optimizer* also maintains an internal register that stores the current maximum error rate (initialized to 0 on start-up). Each calculated error rate is compared to current maximum using a comparator and set as the new maximum if the error rate is greater. All PUF cells are processed to obtain the final maximum error rate. A further optimization is possible by utilizing the maximum error, 6 %, that a 4-bit TMV can correct. The *Error Threshold* can be set to ≈ 62 (6 % of 1023) and each calculated error rate is compared against this before comparing with the stored maximum. The final IC output (*Max Error Rate*) is the maximum error only if it is greater than the error threshold and otherwise, output is a 0. Such ICs would not need burn-in as the TMV is sufficient.

In certain cases, the designer may wish to account for an acceptable yield loss and use more PUF cells than required. Hence, the *Mask Array* can be utilized to select the most reliable PUF cells. The *Burn-in Optimizer* is used to set the mask bits to indicate reliable cells. It can also possess an additional counter to keep track of cells with error rates of $\leq 6\,\%$ (for TMV). In case the system finds the required number of cells, it can output a 0 max error rate to further reduce burn-in requirements. After burn-in, the *Burn-in Optimizer* can be reused to set the final mask bits. Masking has been shown to help improve the reliability of the Weak PUF system [85]. The entire operation of the Burn-in Optimizer is illustrated as a flowchart, as shown in Figure 3.3.

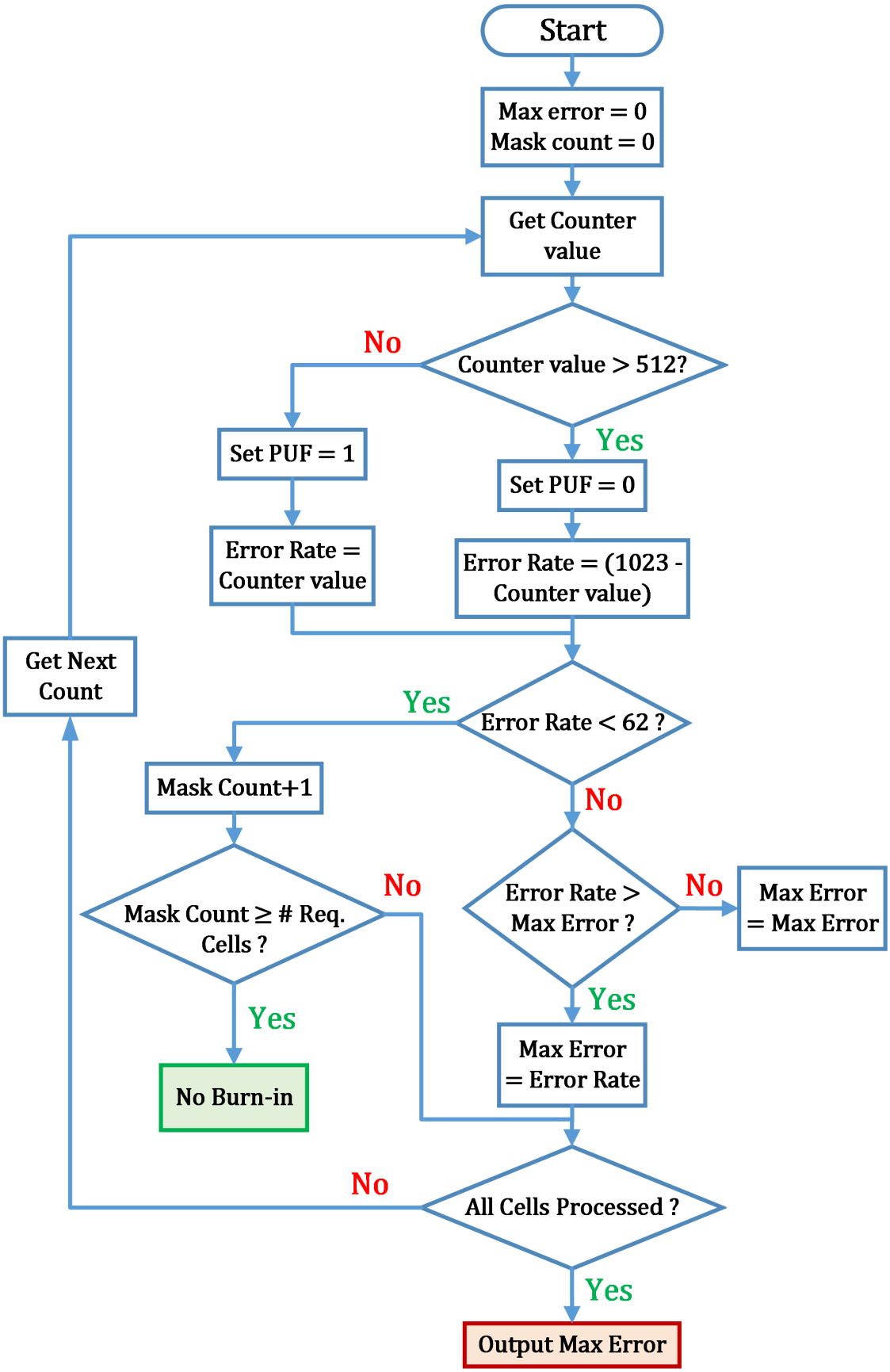


Figure 3.3: Flowchart illustrating the operation of *Burn-in Optimizer*

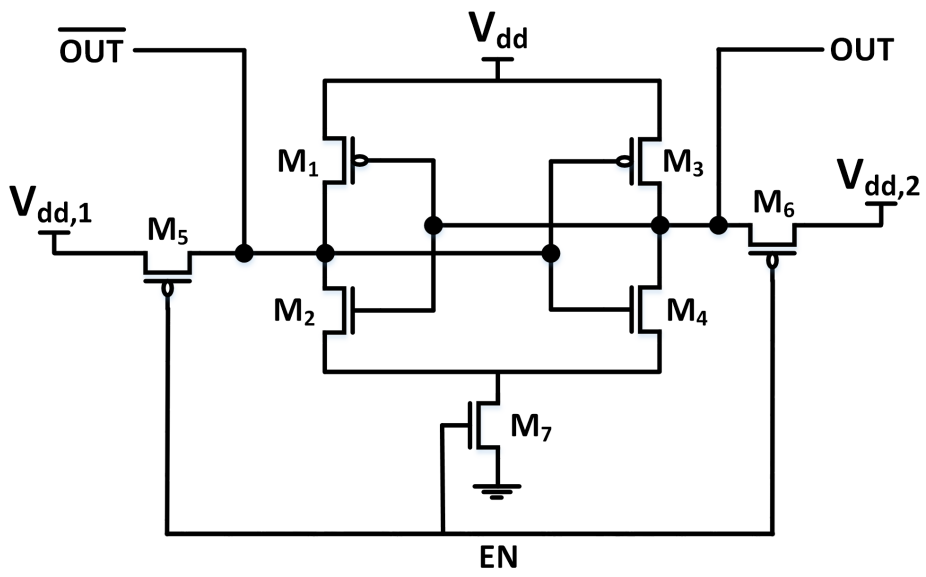


Figure 3.4: SRAM-like cross-coupled inverter PUF cell (*Ref*) [126]

(*ii*) *Regular Operation (OP) mode*: In this mode, the control unit queries and directs the PUF outputs to the 4-bit TMVs. We can use multiple TMVs to speed up the PUF evaluations for convenient real-time operation. As the burn-in process will have increased the mismatch of the PUF cells by an appropriate amount, the TMVs should be able to correct any observed errors as they will be well below the TMV threshold.

3.3.2 Process Variation and Error Rate

We utilize the SRAM cell design, whose operation is detailed in Section 2.3.3 and re-illustrated in Figure 3.4 as the reference design, termed *Ref*.

Utilizing the maximum error rate produced by a PUF system, we aim to find the current inherent mismatch, in terms of threshold voltages, that exists between the PMOS transistors (M_1, M_3), in Figure 3.4. This acts as a proxy that represents the inherent mismatch between the cross-coupled elements in the actual PUF cell and will aid us in determining the amount of NBTI aging needed to make the cell reliable.

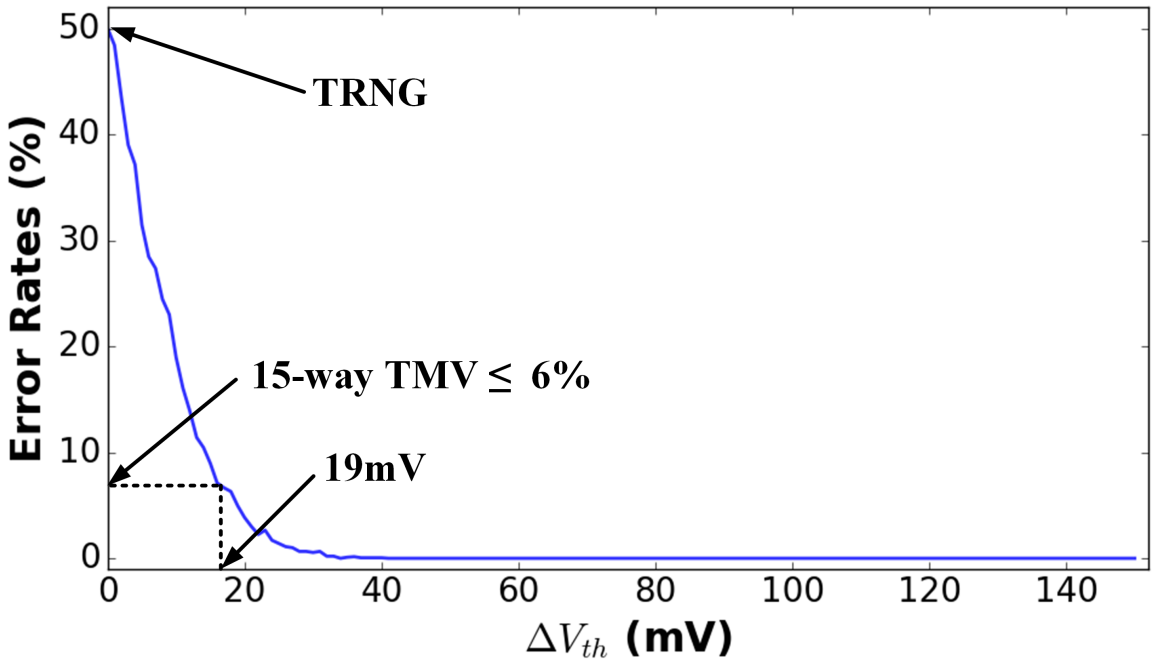


Figure 3.5: Error rate correlation with PUF cell threshold voltage mismatch

To correlate the error rate with a threshold voltage mismatch value, we perform a set of SPICE simulations on the Weak PUF cell. Circuit thermal noise is considered as the source of errors in the PUF output. The noise is applied to the circuit in a differential manner at $V_{dd,1}$ and $V_{dd,2}$, as shown Figure 3.4. The inherent mismatch of a PUF cell is *approximated* by varying the threshold voltage in one of the PMOS transistors (M_3 in our case) in steps of 1 mV up to a certain maximum. A large number of evaluations are performed under varying thermal noise for each step and the error rate is calculated by comparing the values with the output for zero thermal noise.

An example, as shown in Figure 3.5, plots the error rates against the increasing threshold voltage mismatches for a PUF cell, as shown in Figure 3.4, in 45 nm technology [6] using minimum sized transistors. This gives us an *approximate* correlation between the error rate of a cell and the inherent PMOS transistors mismatch. Knowing that a 4-bit (15-way) TMV can correct a maximum of 6 % error, we see that any

cell with an inherent mismatch of $\geq 19\,\mathrm{mV}$ can be handled by the TMV. The cells with lower mismatch are not TMV-correctable and become the target of our burn-in efforts. The worst-case threshold voltage shift, when the maximum error rate is 50 %, for the burn-in process is set at a higher value than 19 mV to account for the approximations made in our analysis. This value becomes the total target mismatch required to create a fully reliable PUF system. We also note that an error rate of 50 % would represent the ideal true random number generator (TRNG). Based on the observed maximum error, which represents the lowest mismatch observed among the PUF cells of a particular IC, we can calculate the threshold voltage shift required from the burn-in process to reach the target mismatch. In a real-world scenario, the maximum observed error rates are likely to be below 50 % and hence, the needed threshold voltage shift from burn-in will be less than the set target value. This translates into significant savings with regards to burn-in time.

3.4 Burn-in time reduction

In this section, we explore the advantage of using the proposed solution presented in Section 3.3 considering Weak PUF systems that need to generate 128 reliable bits. The reported results consider a *cumulative* burn-in time where we assume that only a single IC undergoes the accelerated aging at a time. In practical situations, different manufacturers can utilize different ways. Hence, it is difficult to assume just one arbitrary process.

3.4.1 Weak PUF Designs

Along with the simple SRAM-like Weak PUF design (*Ref*), we extend the analysis using other PUF designs proposed in Chapter 2. It was shown that connecting either the pull-up or pull-down transistors of the cross-coupled inverters in the PUF to a bias voltage can provide significant benefits towards reliability. Since we consider

Table 3.1: Implementation details for various Weak PUF design configurations

Configuration (*Identifier*)	**Transistor Sizing**		**2-parameter model values [78]**		
	PMOS	**NMOS**	λ**1**	λ**2**	**Cells with** 0 **error (%)**
Simple (*Ref*)	90 nm	90 nm	0.292	1.906	83
Two Parallel Loads (*D1*)	180 nm	90 nm	0.188	2.395	96
Current Mirror Load (*D2*)	180 nm	90 nm	0.191	2.491	97
FinFET (*F1*)	1 fin	1 fin	0.413	1.595	60

NBTI as the aging mechanism of interest for the burn-in process, we modify the circuits so that the pull-down transistors are connected to bias voltages while the pull-up transistor are cross-coupled to form the inverters. For this work, we choose the parallel active loads design with two parallel NMOS transistors (*D1*), as shown in Figure 3.6, and the current mirror-based design with NMOS current mirrors (*D2*), as shown in Figure 3.7. The access transistors and the pre-charge voltages connected to OUT and $\overline{OUT}$ are the same as the *Ref* design, shown in Figure 3.4, and are omitted in the circuit diagrams for clarity. The transistors are sized to allow maximum process variation sensitivity except for the footer (M_2), which is sized larger to allow proper operation of the circuit. V_{bias} was set to 0.5 V for *D1* and 0 V for *D2*.

Conventional CMOS scaling beyond the 45nm technology node is severely constrained by pronounced threshold voltage (V_{th}) fluctuations resulting from Short Channel Effects (SCE) and Random Dopant Fluctuations (RDF) due to process variations [41, 105, 115, 120]. Hence, FinFETs were developed to facilitate the continued scaling of technology nodes. FinFET devices exhibit better electrostatic characteristics with respect to SCE as the gate, or *fin*, wraps around a thin slice of silicon (channel) [104]. The greater control over the channel allows FinFETs to have lower

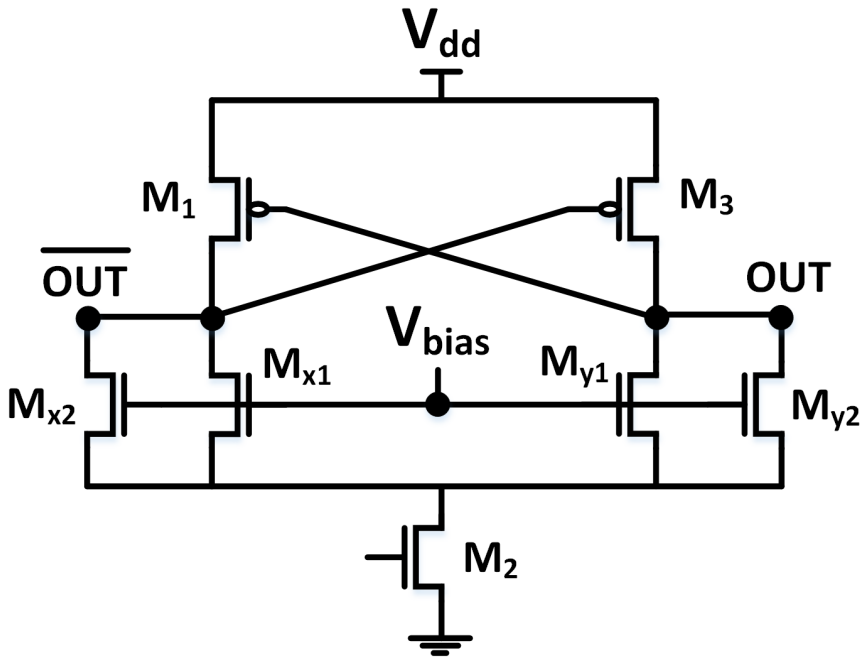


Figure 3.6: Modified parallel active loads-based PUF design (*D1*) [96]

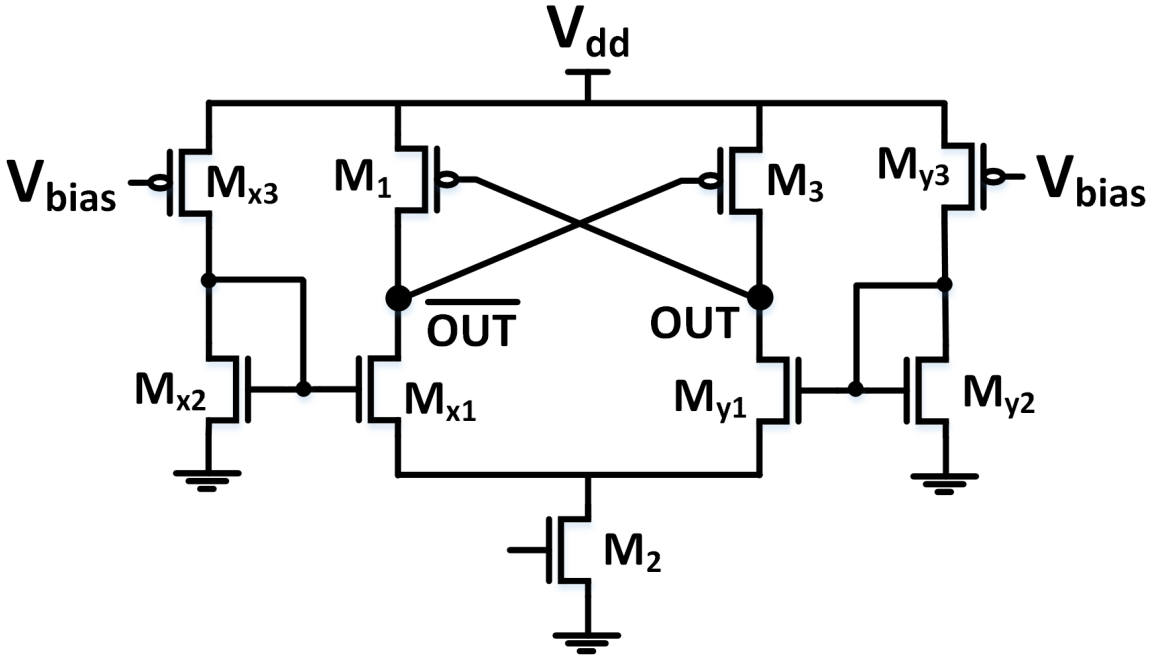


Figure 3.7: Modified current mirror-based PUF design (*D2*) [96]

leakage current and power consumption over bulk CMOS. We wished to study the effect on burn-in requirements when using FinFETs as the basis for constructing the

Weak PUF. We considered only the *Ref* design, as shown in Figure 3.4, and instantiated the PUF, termed as *F1*, using 20 nm FinFETs using predictive technology models [7]. The number of fins were fixed at 1 for all transistors except the footer (M_7 in *Ref*) which had 2 fins for proper current sinking. The supply voltage was set at 0.9 V.

The specifications for the various designs are tabulated in Table 3.1.

3.4.2 Thermal Noise Errors

The random motion of the charge carriers from thermal excitation induces thermal noise in transistors and can create random voltage fluctuations in conductors [64, 92]. Thermal noise has a near uniform power spectral density and there is no correlation among different samples across time. Short channel effects [49] can exacerbate the effects of thermal noise in advanced CMOS technology nodes, causing significant impact on transistor noise performance [119]. The magnitude of thermal noise at any given node can be represented in terms of a normal distribution with 0 mean and standard deviation given by [111],

$$\sigma_{NOISE}(T) = \sqrt{\frac{k_B * T}{C}} \tag{3.3}$$

where k_B is the Boltzmann constant, T is the absolute temperature (Kelvin) and C is the node capacitance (Farads).

We record the node capacitances at OUT and $\overline{OUT}$ in each design (*Ref, D1, D2, F1*), under no process variation condition, to determine the amount of thermal noise to be added to $V_{dd,1}$ and $V_{dd,2}$.

3.4.3 Error Rate vs Mismatch

Using the procedure described in section 3.3.2, we seek to find the correlation proxies for each of the designs being considered. For the planar MOSFETs (*Ref*, *D1*,

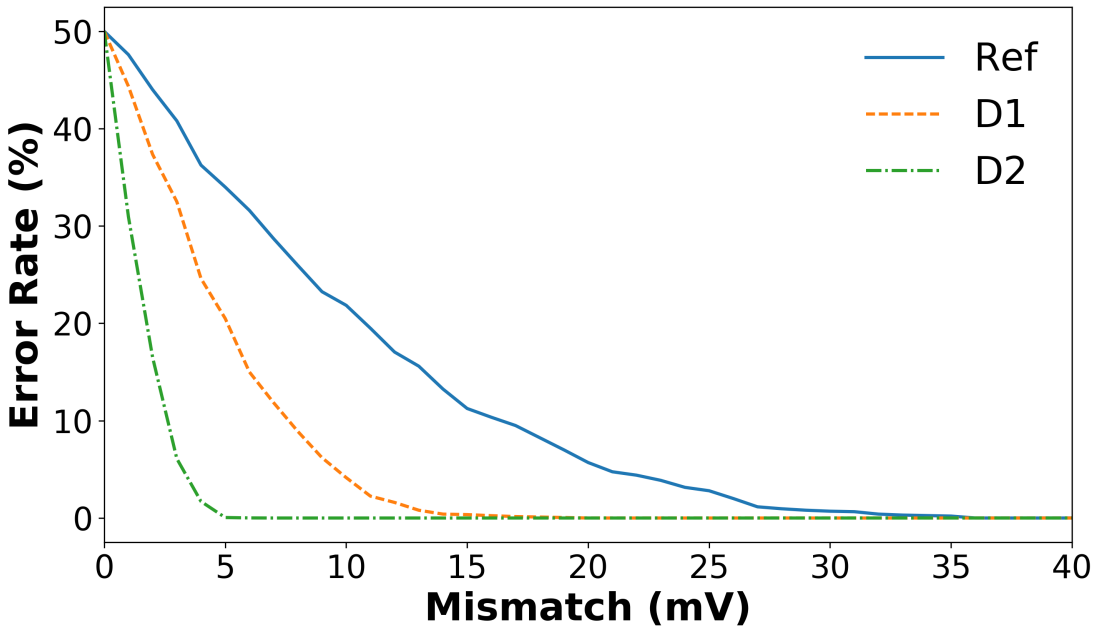


Figure 3.8: Error rate correlation with PUF cell threshold voltage mismatch for alternate Weak PUF designs ({*D1*,*D2*}) based on [96]

D2), we utilized the 45 nm technology [6] to instantiate the cells. The supply voltage (V_{dd}) is set to 1 V. We shift the threshold voltage of one of the PMOS transistors (M_3) in steps of 1 mV up to a maximum of 150 mV. This represents the proxy for the total mismatch and helps simplify further analysis. At each step, we perform 2000 evaluations of the cell under varying thermal noise conditions, using (3.3), and calculate the error rates. Figure 3.8 shows the results which highlight the fact that the alternate designs (*D1* and *D2*) have a greater process sensitivity than *Ref* as every step increase in mismatch reduces the observed error significantly and the error rate reaches 0 % with lower amount of mismatch.

In cases where incorporating the alternate designs may not be desired, we sought to explore a technique to improve the performance of the existing SRAM-based PUF (*Ref*). Boosting the supply voltage has been shown to be beneficial for the operation of an SRAM [98]. However, we must also be aware of increase in leakage power. For

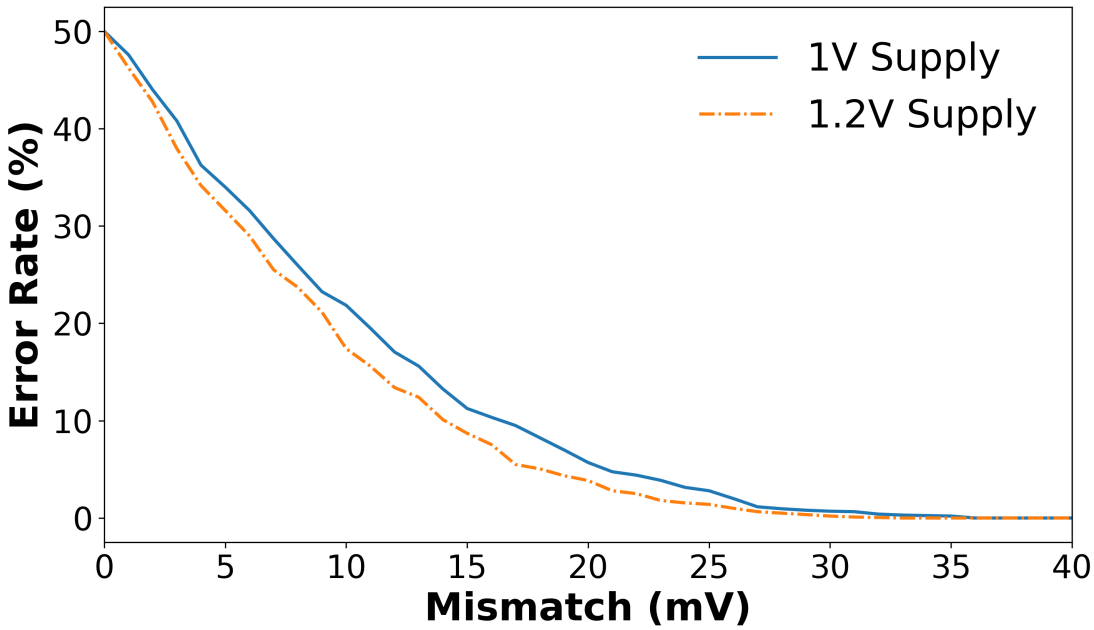


Figure 3.9: Error rate correlation with PUF cell threshold voltage mismatch for *Ref* under nominal and boosted supply voltage (1.2 V)

this work, we study the performance of *Ref* when the supply voltage is boosted to 1.2 V. The results, as shown in Figure 3.9, do indicate that there is a decrease in the error rate with increased mismatch under higher supply voltage. A designer can generate any number of such proxies at different voltages and use the data to adaptively choose what supply voltage needs to be set for the PUF block, which in turn decides the burn-in time. This is due to the fact that we only output the maximum error rate from an IC which is later used with the correlation data. However, such supply adaptability increases the complexity of the overall system design.

Replacing the planar MOSFET based PUF (*Ref*) with FinFET-based design (*F1*) provides better performance in cases where the inherent mismatch is higher, as shown in Figure 3.10. It should be noted that the FinFETs are more susceptible to thermal noise (from (3.3)) as the node capacitances are lower due to the smaller technology

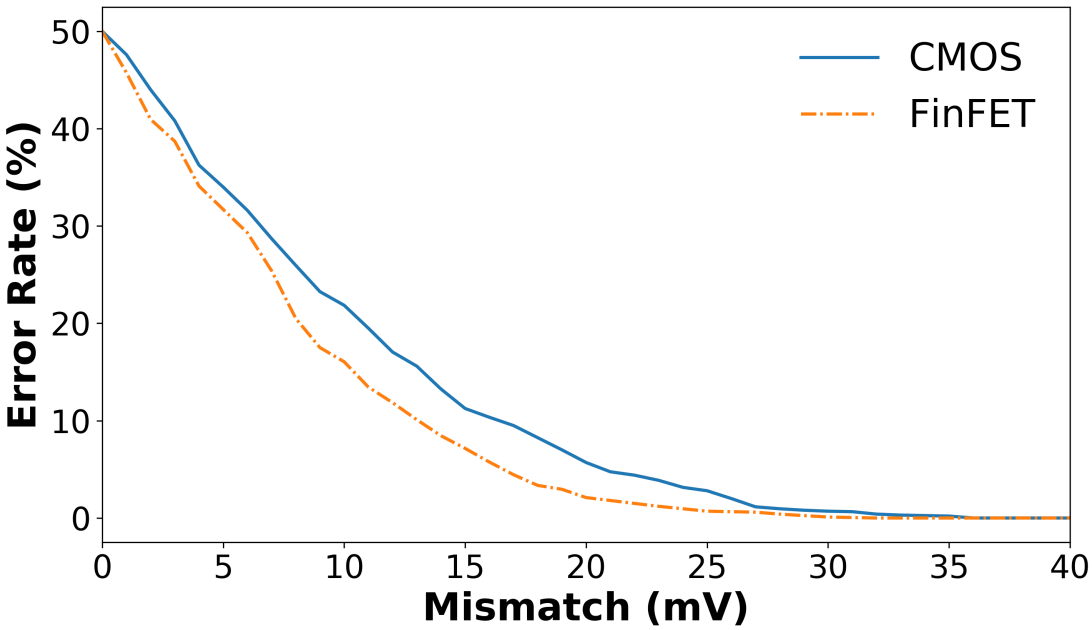


Figure 3.10: Error rate correlation with PUF cell threshold voltage mismatch for planar MOSFET (*Ref*) and FinFET (*F1*) designs

node (20 nm) compared to the planar MOSFET. The results show that FinFET based implementations can be viable as Weak PUFs even with the higher noise.

3.4.4 Modeling Process Variation

3.4.4.1 Planar MOSFET

Manufacturing induced process variations are modeled as *random* parametric variations in the threshold voltage (VTH) and channel length (L) of each transistor in a circuit. The values are obtained from a normal distribution, $N(\mu, \sigma^2)$, where the mean (μ) and standard deviation (σ) are determined based on the technology node used. Transistor geometry has an impact on the susceptibility of a device to process variations with larger devices experiencing less fluctuations. In terms of threshold voltage, the mean is the default transistor model value and the standard deviation is given by,

$$\sigma_{VTH} = \frac{\sigma_{VTH0}}{\sqrt{\frac{W*L}{W_{min}*L_{min}}}} \tag{3.4}$$

where (W_{min},L_{min}) are the minimum possible width and length of a device, respectively, and (W,L) are the sizes used in the design. σ_{VTH0} is the standard deviation of threshold voltage for the minimum sized device.

In this work, we instantiate the planar MOSFET PUF cell designs (*Ref*, *D1*, *D2*) with 45 nm NCSU FreePDK45 models [6]. Typically, for 45 nm node, a standard deviation of 53 mV for threshold voltage and 10 % channel length variation are considered [4]. Equation (3.4) is used to calculate the standard deviation for threshold voltage of any arbitrary sized transistor.

3.4.4.2 FinFET

While FinFETs are not affected by RDF due to undoped channel, they are susceptible to Work Function Variation (WFV) caused by irregularities in *fin* surface from the manufacturing process [86]. Hence, WFV has the greatest impact on threshold voltage variation. For this work, we consider a standard deviation of 30 mV for threshold voltage to represent the process variation in FinFET transistors [86].

3.4.5 Heterogeneous Error Model

In realistic scenarios, a collection of Weak PUF cells would have a distribution of errors. This heterogeneous error model is advantageous in modeling real PUF systems as a homogeneous error model can overestimate the required ECC resources [78]. Also, the homogeneous model would provide no useful information for driving the burn-in process as all cells would be assumed to have the same error rate and hence, the same amount of process mismatch.

For this work, 10,000 instances of the PUF cell, for each design, were simulated with the relevant process variation parameters, as described in sections 3.4.4. Each cell was evaluated 1000 times under varying thermal noise (as determined in section

3.4.2) at 25 °C. The error rate was obtained by comparing with a simulation with no noise. Utilizing the resulting data with the mathematical framework for the heterogeneous error model [78], we can generate error rates for an arbitrary number of PUF cells for each given design. The relevant 2-parameter model details for each PUF design are tabulated in Table 3.1. We also list the percentage of cells among an arbitrary population that would possess 0 % error for each design, given the 2-parameter model data. Due to the higher thermal noise susceptibility, FinFET implementation (*F1*) offers the lowest amount of 60 %. This is an indication that the FinFET based design will offer less savings than a comparable planar MOSFET design in terms of total burn-in time.

3.4.6 Cumulative Burn-in Time

Since our target is to obtain 128 stable bits from a PUF system, we first consider the system has 128 initial PUF cells and all the cells need to be made reliable. Next, we considered a scenario where a 2 % yield loss might be acceptable and *masking* is used. For this case, we reused (3.1) with $\{k = 128, p = 0.98\}$ to find the value of N that would result in $P_e(N) \leq 10^{-6}$ (failure rate of 1 ppm). The result showed that we needed 144 initial PUF cells.

For all the designs considered in this paper (*Ref, D1, D2, F1*), we assume that the burn-in temperature is 100 °C and the stress voltage is 1.1 V and utilize (3.2.2) to calculate the time required. The relevant device parameters for the NBTI equation are obtained from the NCSU FreePDK models [6] for planar devices and from PTM models for the FinFET [7].

For obtaining the cumulative burn-in time, 1 *million* PUF systems were generated for the 128 and 144 cells/system cases for each Weak PUF design using the heterogeneous error model. For each PUF system, the threshold voltage mismatch was found using the methodology described in section 3.3. To find necessary increase in device

Table 3.2: Results for reduction in Cumulative Burn-in time for various PUF configurations

Configuration	**# PUF Cells**	**Cumulative Burn-in Time (hours)**			**Target Mismatch (mV)**	**Natural Aging Time**
		Worst-case	**Optimized**	**Reduction (%)**		
Ref	128	41.52E6	17.93E6	***56.82***	25	700 days
	144		0.62E6	***98.50***		
Ref (1.2 V)	128	25.15E6	15.92E6	***36.70***	23	425 days
	144		0.42E6	***98.33***		
D1	128	0.49E6	0.114E6	***79.73***	12	9 days
	144		0	***100.00***		
D2	128	90.2	43.77	***51.48***	4	16 mins
	144		0	***100.00***		
F1	128	10.32E6	8.62E6	***16.47***	22	295 days
	144		0.18E6	***98.25***		

mismatch in each system, the data from section 3.4.3 and the mismatch representing the 6 % error rate (for 4-bit TMV) were used. Our target threshold voltage shift (also the worst-case shift) was set by increasing the TMV mismatch found for each design by 30 %. This is to account for the approximations in our methodology for correlating error rate and inherent mismatch. Table 3.2 also lists the amount of time needed to increase the PMOS threshold voltage due to NBTI by the target mismatch value.

For each system, the max error rate indicates the lowest amount of inherent mismatch and decides the amount of threshold voltage shift needed to reach the target. The shift needed is used in (3.2.2) to calculate the burn-in time. The cumulative burn-in time (*Optimized*) for 1 *million* PUF systems with 128 and 144 initial cells are recorded in Table 3.2 for each design. We also note that without the proposed solution, a manufacturer might need to assume that each IC needs to undergo the maximum burn-in for each design.

From the results, in Table 3.2, we see that intelligent burn-in offers significant savings compared to constant worst-case scenario driven burn-in. Also, using extra bits (144) and masking results in reduced burn-in time compared to using just 128 cells/system. *D2* offers the best results for any PUF system as the burn-in required is negligible compared to other designs. Additionally, combining the alternate designs (*D1* and *D2*) with masking and extra bits can allow us to forgo burn-in entirely. Consequently, these designs are attractive for low-cost applications where dedicated PUF circuits would make more sense in terms of resource utilization. The FinFET (*F1*) results show that we need to use extra bits for greater reliability, but the lower cell area for the technology allows us to easily incorporate more cells.

The results discussed here show the *relative* performance of various designs, but are ultimately dependent on the amount of error correction afforded to us by the TMVs used during regular operation. Vijayakumar et al.have shown that using an Up-Down counter [126] allows them to correct *twice* the error compared to a similarly

sized regular TMV. In actual designs, using better error correction reduces the target mismatch required by the burn-in process, but such considerations must be weighed by a designer against various constraints such as available area, power requirements and so on.

Resource Overheads: We considered a 144-PUFs system and calculated the area overhead from the *Burn-in Optimizer*, *Central Control*, *Mask Array* and 10-bit counter described in section 3.3. The rest of the circuitry, for TMV-based reliable bit generation is assumed to already be present in the system. Using the 45 nm standard cell library from Nangate [5], the area overhead was found to be $500\,\mu m^2$ (Burn-in optimizer costs $149\,\mu m^2$). This is a small overhead as the majority of the area ($\approx 2500\,\mu m^2$) for an efficient implementation is occupied by the TMV counters, PUF cells and the mux arrays.

The primary testing time overhead is from obtaining the maximum error rate which entails serially querying each PUF cell 1024 times (10-bit counter) and processing the results. For a 144-PUFs system, this requires $\sim 150,000$ cycles in total. At a test frequency of 10 MHz, for example, each chip will need 15 ms to generate the results. However, in the cases where no burn-in is required, the designer can directly proceed to final key enrollment phase. The Burn-in Optimizer is only used to set the Mask Array and the system operates at its final intended frequency, which is greater than the test frequency.

3.5 Conclusion

Weak PUFs have been extensively proposed for security applications such as key/ID generation. Such applications require the PUFs to be highly reliable even in the presence of noise. To ameliorate the noise susceptibility of the PUF outputs, many different techniques have been proposed. One such method is to utilize burn-in/accelerated aging for improving PUF reliability. Our work focuses on a method-

ology to create a PUF system that enables the calculation of minimum burn-in time for an IC. We obtain results for various SRAM-like Weak PUF designs which show a significant reduction in burn-in times, providing large savings during the post-silicon stages of the manufacturing process. Further, results show that FinFET based implementations can also be viable as Weak PUFs even in the presence of significant noise.

CHAPTER 4

REALIZING ROBUST STRONG PUFS USING WEIGHTLESS NEURAL NETWORKS

4.1 Introduction

As discussed in Section 1.1.2, *Strong* PUFs provide an exponential number of CRPs and can be suitable for authentication applications as *lightweight* hardware roots-of-trust. Ideally, PUFs need to possess high uniqueness and *reliability* (especially Weak PUFs) to provide robust security. Furthermore, Strong PUFs can be targeted with model-building attacks using machine learning techniques and thus, need to be resilient against cloning.

Unfortunately, practical Strong PUF implementations can suffer from reliability issues [65,124] and also, can be cloned with high accuracy by machine learning attacks [107, 125]. For use in applications like IoTs, unreliability can affect the number of CRPs required to properly distinguish a PUF-equipped integrated circuit (IC) from billions of other such devices [103]. Hence, there is still a need to design Strong PUFs that can be fully reliable and offer high resistance to machine learning attacks.

Artificial neural networks mimic biological neuron functions for the purpose of achieving effective pattern recognition capabilities. Weightless Neural Networks (WNNs) are a class of artificial networks that utilize excitatory/inhibitory signaling to simulate a neuron's dendritic tree. *Wilkie, Stonham & Aleksander's Recognition Device* (WiSARD) was the first WNN model to be distributed commercially [17]. It provides an efficient and simple implementation that can be realized using random-access memory (RAM).

This chapter explores adapting the simple WNN architecture and reliable Weak PUFs to create reliable Strong PUFs that can also exhibit high uniqueness and provide a high level of machine learning resistance. SRAM cells have persistent, yet random power-up values and are the most promising choice for creating Weak PUFs [50, 57]. Reliability of a Weak PUF is critical and has received extensive attention in research [29, 39, 80, 81, 85, 126]. In this work, we first combine the WiSARD WNN model and SRAM's PUF properties to realize Strong PUFs and later, utilize bits produced from an initial entropy source, consisting of a set of highly reliable Weak PUFs, to load the contents of the WiSARD WNN RAMs in order to provide a robust Strong PUF architecture. Since the initial Weak PUF bits are made reliable, this enables us to extend the reliability to the final Strong PUF. Using the WNN *classification* phase, we can conceive a Strong PUF by providing the challenge as the input pattern and extracting the output from the WNN as the response. We illustrate various architectures based on the WiSARD model to showcase multiple Strong PUF designs. Each architecture is analyzed to obtain uniqueness, reliability and machine learning resistance metrics. This work shows that it is possible to build a Strong PUF using neural networks obtaining high machine learning resistance while maintaining good uniqueness and reliability. Additionally, we explore the minimum size of the initial reliable Weak PUF entropy source that can still offer high uniqueness and machine learning resistance.

4.2 Background

In this section, we briefly explore the relevant background regarding PUFs, their reliability and machine learning resistance. Later, we discuss previous work on Weightless Neural Networks. The relevant prior research into improving the reliability of Weak PUFs has been detailed in Section 2.2.

4.2.1 PUFs

Pappu et al.introduced Physically Unclonable Function (PUF) as an one-way function to map challenges to unique responses [93]. The promise of Strong PUFs arose from the supposed complexity of the challenge-response mapping in each PUF and the uniqueness of such mappings across chips, which would ideally make the PUFs resistant to model building attacks. One of the earliest PUFs, called the Arbiter PUF [75], did not exhibit the salient properties and could be easily cloned [107]. Multiple alterations were proposed, like using XOR operations, to increase the resistance and yet, were still broken [107].

Unlike digital PUFs, analog circuits were proposed to harness non-linear behavior of CMOS transistors under certain operating conditions as a means to increase the attack resistance of Strong PUFs. Current-based [65,71] and voltage-based techniques [124] have been shown to be resistant against Support Vector Machine (SVM) learning algorithm, which had broken the previous digital PUF implementations. However, Vijayakumar et al.showed that ensemble meta-algorithms were a new class of machine learning algorithms that could effectively model even the analog PUFs with great accuracy [125]. Further, side-channel and fault based attacks have also been utilized to increase the modeling accuracy to break PUFs [70, 72, 130].

Irrespective of the Strong PUF models proposed, they all had reliability issues. While this can be mitigated by the fact that we have an exponential number of CRPs available and we can set a threshold of acceptable responses for practical use in authentication, the threshold level increases with a decrease in PUF reliability. This increases the number of CRPs needed to authenticate a device in real-world scenarios and thus, raises resource costs [103].

SRAM-based Weak PUFs have been studied extensively in previous research [50, 57]. In contrast, Holcomb and Fu proposed a Strong PUF using SRAM cells in a column of a memory block which are pre-loaded with values based on an input

challenge. The PUF output is generated by reading multiple cells in a column at once to create a contention at the sense amplifier, which produces the final response [56]. Bhargava et al.also explored the application of Weak PUFs to create Strong PUFs by extracting a stable secret key from Weak PUFs as an input to an AES block. The plain text input is considered as the challenge and the response is the cipher text generated by the AES block [24]. In this chapter, we utilize a similar concept of generating stable Weak PUF bits. The wealth of techniques available to improve the reliability of Weak PUFs, as detailed in Section 2.2, allows us to create an efficient implementation that yields the desired number of stable bits for later use in neural network architectures to realize robust Strong PUFs.

4.2.2 Weightless Neural Networks

Weightless Neural Networks (WNNs) [17] are abstract models of biological neurons where each neuron is represented by a Random Access Memory (RAM) node. This model offers an attractive practical solution to pattern recognition and artificial consciousness applications, due to its representation of neurons in a binary format and due to the capability to implement such networks using existing memory resources in devices.

WiSARD (Wilkie, Stoneham and Aleksander's Recognition Device) was one of the first WNN model developed [16] and was inspired by the n-tuple classifier [27]. Each class is represented by a structure called *Discriminator*, which comprises of a set of RAMs, composed of one-bit words, to store the relevant information during the *training* phase which will be used during the *classification* phase. The Discriminator inputs can be from an arbitrary source and only need to be converted to binary for use. A binary input of $N \cdot M$ bits is split in N tuples of M bits. Each Discriminator consists of multiple RAM blocks ($= N$), each containing 2^M locations, addressed by their respective M-bits. Tuples are connected to the binary input in a biunivocal

pseudo-random mapping and provide the address to each RAM block. A WiSARD system can have any number of such Discriminators and hence, any number of desired classes.

During the training phase, all RAMs of the WNN are initialized to zero (0). The training input is sent to each Discriminator, where the accessed RAM positions are set to one (1). During classification, an input is sent to all Discriminators and the Discriminator which presents the highest response is selected as representative class for the input. The discriminator response is calculated by summing all accessed RAM values.

The structure of WiSARD can be readily implemented in hardware using standard SRAM memory and address decoding to provide high generalization capabilities and real-time performance. The classification phase of WiSARD will be utilized to realize the PUFs in this work.

4.3 Strong PUFs based on Weightless Neural Network

In this section, we present a Strong PUF design inspired by the WiSARD model and examine some extended versions with the objective of improving the machine learning resistance. All architectures produce a 1-bit output response, given an m-bit input challenge.

4.3.1 WiSARD PUF

The first design is termed as *WiSARD PUF* and depicted in Figure 4.1. It is composed of a single Discriminator containing multiple RAM blocks. The input is a challenge string, which is sliced in a pseudo-randomic way into multiple tuples. Each tuple forms the address of a unique RAM block, Ri, and decides the size of the block. The 1-bit memory locations from the RAM blocks are accessed according to the input challenge and passed to a counter that accumulates the number of 1's. The final 1-

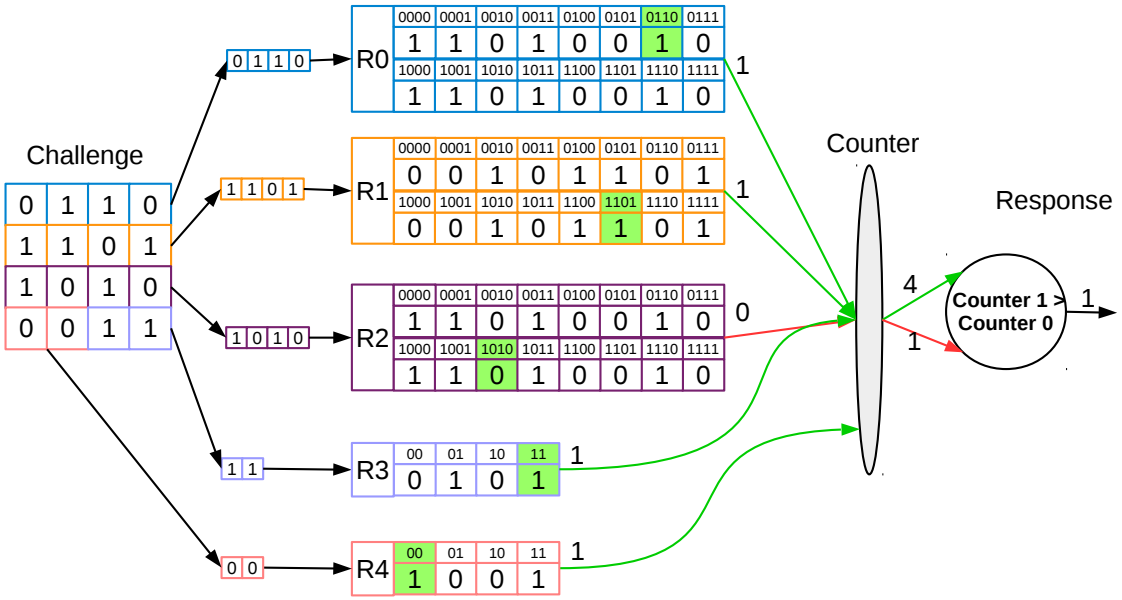


Figure 4.1: Example of WiSARD PUF architecture [108].

bit response is generated through the majority voting over the counter value. For the proper majority voting functionality, an odd number of RAM blocks are required. The challenge to tuple mapping is adapted accordingly to create odd number of RAM blocks, as shown in Figure 4.1 where 16-bits input challenge is mapped to three 4-bit tuples and two 2-bit tuples to complete the total of 5 RAM blocks.

The process to produce the final output response from Strong PUF is similar to the WiSARD *classification* phase, that applies counting to the accessed RAM bits instead of summing those bits as Discriminator response. Different from conventional WiSARD, there is no *training* phase. The RAM blocks consist of SRAM cells whose process variation dependence ensures to store the random values in each RAM content when powered on, like SRAM PUFs [50, 57]. Two WiSARD PUFs can have their own random RAM contents and also pseudo-randomic mapping of challenge to tuples.

SRAMs outputs are susceptible to noise resulting in erroneous behavior upon multiple power-ons [79]. This can affect the reliability of the WiSARD PUF which

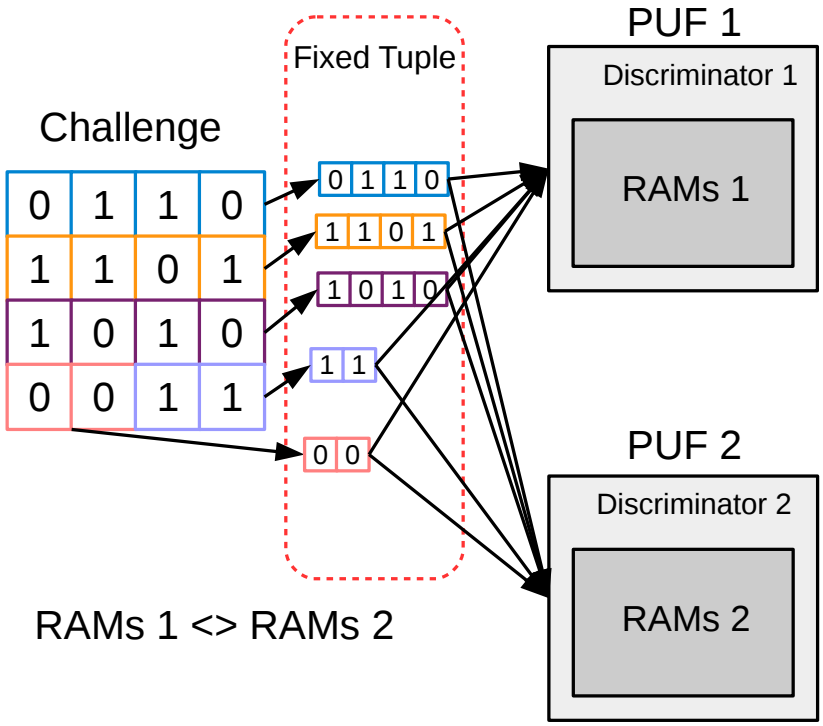


Figure 4.2: Example of WiSARD PUF with fixed tuples among PUFs [108].

comprises multiple SRAM cells. Consequently, it is crucial to consider the intra-class Hamming distance and ensure that the Strong PUF reliability is acceptable.

It is possible to have unique input-to-tuple mapping for each PUF. However, this can prove costly in terms of actual hardware implementation as it involves implementing additional circuitry to create the mapping. Multiple mapping techniques can be used, but are considered beyond the scope of this work. For simplicity, we will assume a case where the mapping is fixed across all WiSARD PUFs by the designer, as presented in Figure 4.2. All subsequent architectures that will be discussed assume a fixed mapping of the input challenge to tuples across all PUFs.

4.3.2 Extensions to WiSARD PUF architecture

Since using a WiSARD PUF with simple majority voting on the RAM block outputs can be sufficient to create a Strong PUF, we also explore possible extensions to the original design with the hope of improving machine learning resistance in

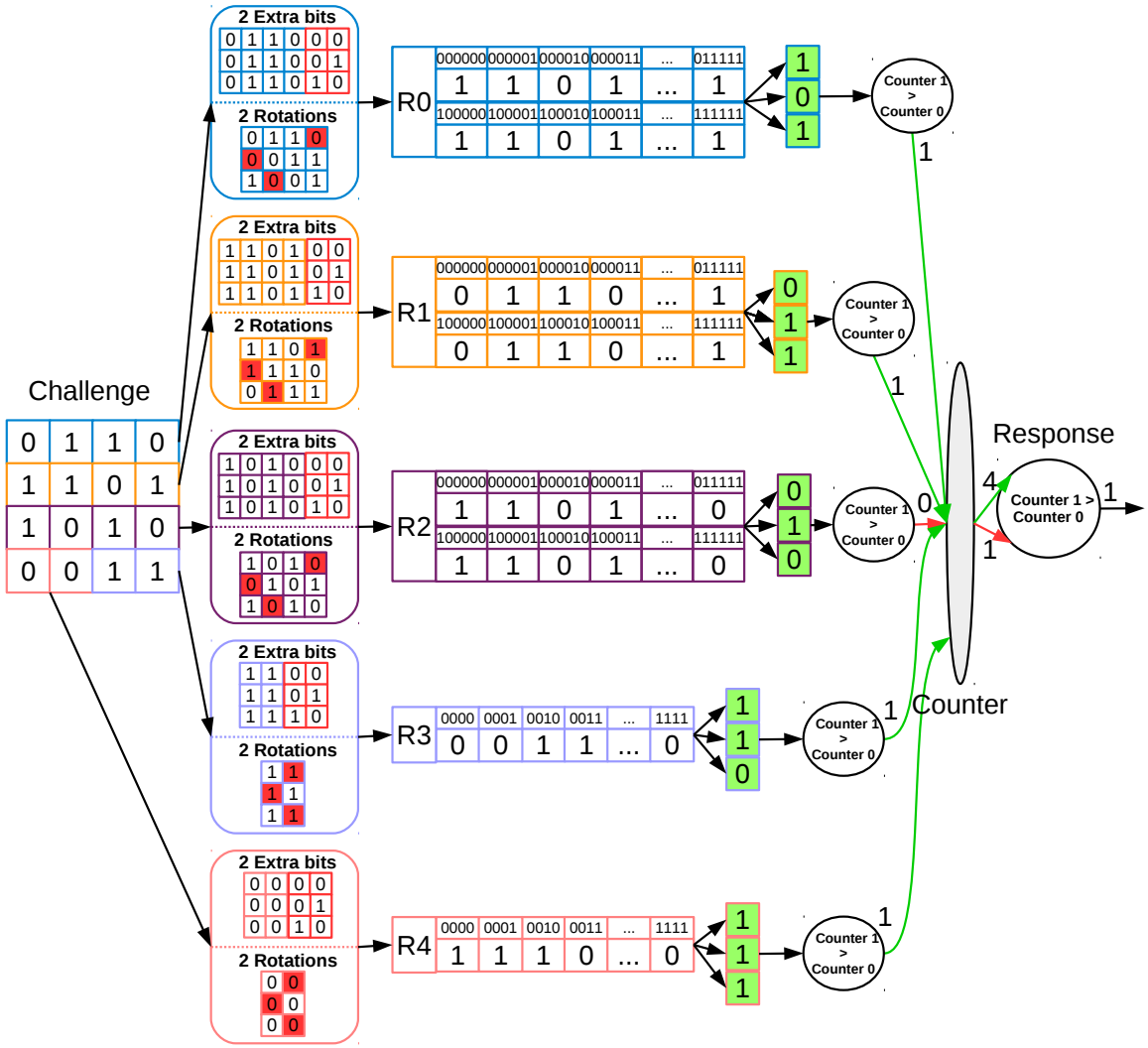


Figure 4.3: Example of WiSARD PUF architecture with extra bits and tuple rotations (circular shifts) [108].

comparison to the WiSARD PUF. In particular, we investigate to affect either the tuple generation for addressing the RAM blocks or the output process of the blocks in different ways to generate the final output response.

4.3.2.1 Fuzzy logic based address generation

A popular way to deal with noisy data is to harness fuzzy extractor, as evidenced by its advantageous use in deriving stable keys from biometric data [40, 76]. Fuzzy logic has also been extensively used to improve reliability of Weak PUF in the pres-

ence of noise [29, 39, 80, 83]. Fuzzy extractor is usually split into *enrollment* and *reconstruction* phases. During enrollment, *helper data* is created manipulating the input data, say PUF response bits, in a trusted environment. Reconstruction assumes that the received data is noisy and uses the proper helper data to retrieve the error-free response originally used in the enrollment phase.

We utilize fuzzy extractor concepts to first generate more data from RAM blocks than in a normal WiSARD PUF, akin to helper data generation and then, reduce this data to obtain the final PUF response (Reconstruction). Two approaches were utilized to affect the challenge tuples so as to collect multiple outputs from each RAM block – (a) add *extra bits* in random locations to each tuple; (b) perform *rotation* (or *circular shift*) operation on each tuple.

In extra bits approach, we add e extra bits to each tuple allowing to generate all 2^e combinations of the extra bits and correspondingly, the same number of addresses and outputs from each RAM block. All combinations can be generated internally using a simple counter. For the second approach, the *original* challenge tuple is used to obtain the output from the corresponding RAM block. Then, we perform a predefined number of *right circular shift* (or rotation) operations on the tuple and generate multiple outputs using the new addresses.

Both methods help access more of the system entropy for the same input challenge to further help inoculate the PUF against machine learning attacks. An example for 16-bit input challenge and 2 extra bits (or 2 tuple rotations) is illustrated in Figure 4.3. Each RAM block generates 3 outputs which undergo majority voting to produce a single output. The new outputs undergo a second majority voting process to produce the final 1-bit response. We ensure that an odd number of inputs are utilized for the majority voting either by performing an even number of tuple rotations, or by adding $(2^e) - 1$ combinations of the extra bits, and using odd number of RAM blocks, similar to the basic WiSARD architecture.

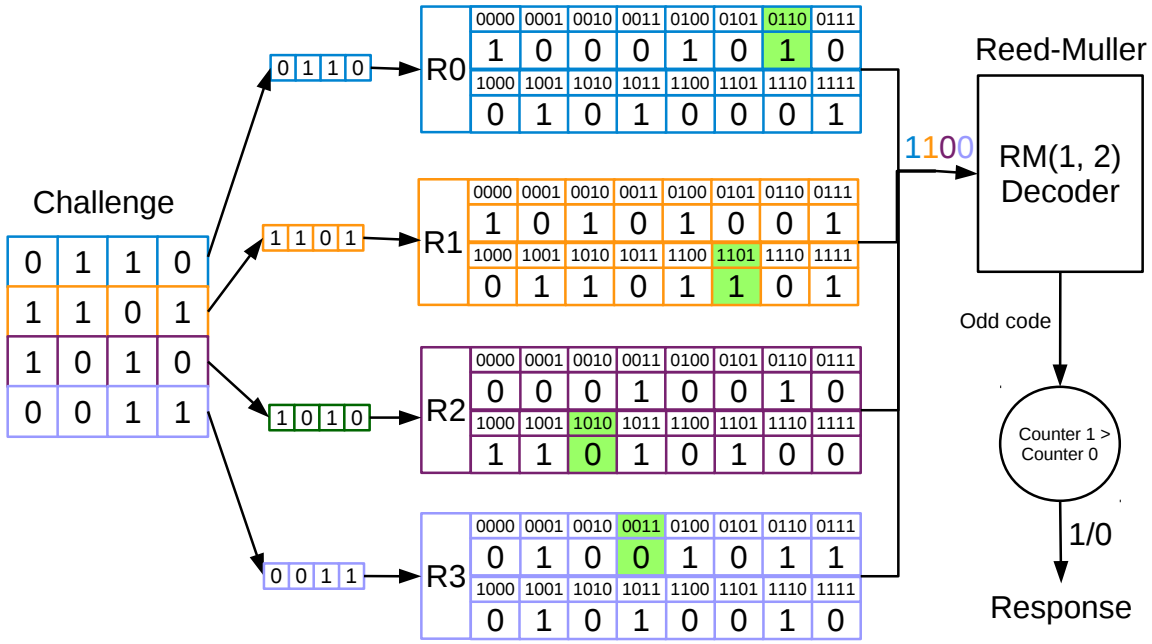


Figure 4.4: Example of RM-WiSARD PUF architecture [108].

4.3.2.2 Concatenated codes based response generation

The response generation in the basic WiSARD architecture, as shown in Figure 4.1, utilizes majority voting, which is akin to using a *repetition* code. Other error-correcting codes (ECC) can also be implemented for the purpose of generating the final response from the RAM block outputs. Based on the code scheme, it is possible to introduce a non-linear relationship between the RAM outputs and the response, in contrast to the linear relationship exhibited by majority voting. This can greatly benefit the machine learning resistance of the final Strong PUF. Christoph Bösch illustrated the advantages of using concatenated error-correcting codes (ECC) to improve Weak PUF reliability and also provided detailed hardware implementation of various ECC schemes [29]. We employ one of the concatenated ECC schemes and present an alteration to the basic WiSARD architecture discussed previously.

In this work, Reed-Muller (RM) code-based decoding is applied to the RAM block outputs, as illustrated in Figure 4.4. The final response is generated by using the repetition code on the output bits of the RM decoder. The new design is termed

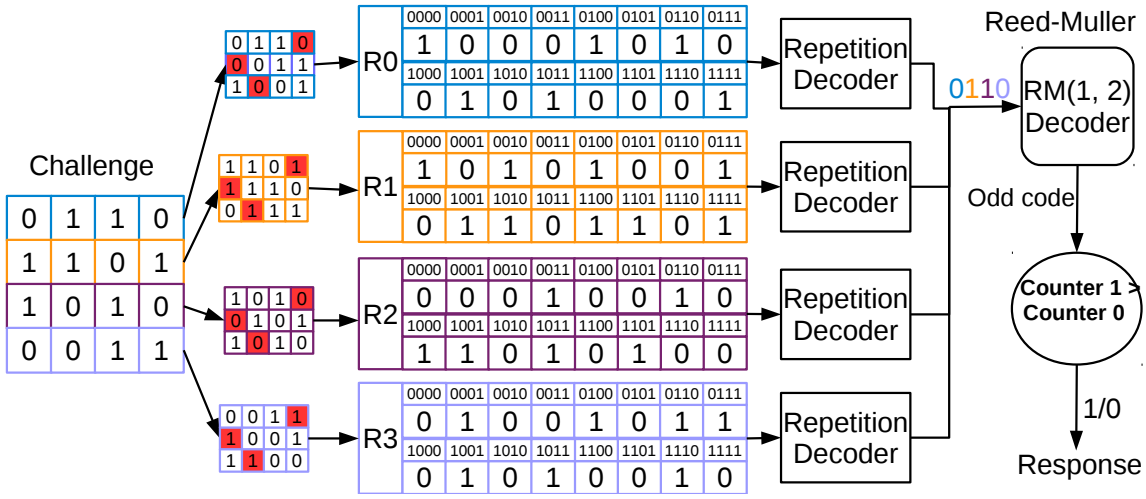


Figure 4.5: Example of RM-WiSARD PUF architecture with concatenated code (tuple rotations)[108].

as the *RM-WiSARD* architecture. The *RM decoder* used in our design accepts an input of length 2^m and generates an $(m+1)$-bit output. The decoder is referred to as RM$(1, m)$ Decoder. The architecture is changed to contain 2^m RAM blocks to produce the necessary outputs. The Hadamard transform algorithm is implemented by the decoder with a simple modification to extract a final code of odd length. A Reed-Muller decoder hardware implementation scales with the chosen value of m and has been detailed by Christoph Bösch [29].

RM-WiSARD implementation can be further modified to utilize extra bits or tuple rotations, similar to basic WiSARD architecture. The RM decoder is appropriately modified to accommodate the final PUF response generation. Figure 4.5 illustrates an example of the RM-WiSARD implementation where repetition code is applied to the outputs from tuple rotation of each RAM block. Then, the results are passed to the RM decoder which produces an output with odd number of bits. We use the repetition code again to get the final 1-bit response, as desired.

4.4 WNN PUF - Experimental Setup and Results

In this section, we evaluate the proposed designs in Section 4.3 to identify which one can provide the highest machine learning resistance. We also analyse the inter-class (uniqueness) and intra-class (reliability) Hamming distance metrics to further compare and contrast the suitability of various designs as Strong PUFs.

4.4.1 Setup

All PUF experiments are simulated in Python and the relevant SRAM circuit data is obtained from SPICE simulations using 45 nm transistor models [6]. Each PUF receives a 64-bit input challenge to produce a 1-bit response. The Discriminator SRAM cells of each PUF instance are filled with the random power-on states such that the average number of 1's and 0's are equal. The general details about the WNN designs are tabulated in Table 4.1. For designs that have 9 RAM blocks, 7 are addressed by 8-bit tuples while the remaining two use 4-bit tuples. The extended address generation designs assume 2 extra bits and 2 rotations for the respective implementations. All RM-WiSARD PUF design variants utilize the RM$(1, 3)$ decoder for response generation. Except original WiSARD PUF, all the designs assume a fixed input-to-tuple mapping across PUFs, as depicted in Figure 4.2.

4.4.2 Uniqueness

Uniqueness is a property wherein each PUF generates different responses compared to the others PUFs for the same challenge. Inter-class Hamming distance (HD) is the metric used to determine uniqueness. We calculate the Hamming distances between the responses of each pair of PUFs for the same challenge and average the results over many challenges and PUF instances. The average of inter-class HD, d_{inter}, can be defined as:

$$d_{inter} = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD(r_i, r_j)}{n} \tag{4.1}$$

Table 4.1: WiSARD PUF design architectures for 64-bit challenges [108]

	#RAM Blocks	***#Addresses***	***#SRAMs***
WiSARD PUF	9	$7 \cdot 2^8 + 2 \cdot 2^4$	$1,824$
WiSARD PUF (fixed tuple)	9	$7 \cdot 2^8 + 2 \cdot 2^4$	$1,824$
RM-WiSARD PUF	8	$8 \cdot 2^8$	$2,048$
WiSARD PUF + 2 Extra bits	9	$7 \cdot 2^{10} + 2 \cdot 2^6$	$7,296$
WiSARD PUF + 2 rotations	9	$7 \cdot 2^8 + 2 \cdot 2^4$	$1,824$
RM-WiSARD PUF + 2 Extra bits	8	$8 \cdot 2^{10}$	$8,192$
RM-WiSARD PUF + 2 rotations	8	$8 \cdot 2^8$	$2,048$

where k is amount of PUFs, n is the total of bit responses and $HD(r_i, r_j)$ is Hamming distance between responses of the PUF instances i and j to a particular challenge. A PUF is considered ideal when the normalized inter-class HD = 0.5.

For this experiment, (k =) 1000 PUF instances were evaluated for each design, (n =) 1000 challenges were applied to each PUF to produce the total of 1000 response bits. Table 4.2 summarizes the uniqueness results for the several PUF designs.

We notice that every designs have the mean close to the ideal 0.5 and the original WiSARD PUF (random input-to-tuple mapping) resulted in the highest normalized HD. Considering the extended tuple generation designs, the tuple rotation offered better results in comparison to extra bits scheme. Lastly, while we consider fixed

input-to-tuple mapping for the majority voting of the WiSARD PUF variants, it may be beneficial to explore efficient random challenge mapping in hardware in order to extract greater uniqueness from the system.

4.4.3 Reliability

Reliability is the PUF property to produce the correct response given the same challenges even in the presence of noise. Intra-class Hamming distance is the metric used to evaluate reliability. We calculate the intra-class HD by extracting the correct responses (r_i) from a PUF under ideal conditions and obtain a set (m) of noisy responses (r'_i) by introducing errors in RAM locations across multiple power-ons. Then, the Hamming distance between the correct and the noisy responses for each challenge is performed. An ideal PUF should have intra-class HD = 0 for any challenge, representing 100 % reliability. The average of intra-class HD, d_{intra}, is calculated as:

$$d_{intra} = \frac{1}{m} \sum_{t=1}^{m} \frac{HD(r_i, r'_{i,t})}{n} \tag{4.2}$$

where n is the number of CRPs collected from each PUF and $HD(r_i, r'_{i,t})$ is Hamming distance between responses of the right PUF instance i and its t-th noisy instance to a particular challenge.

To simulate the noisy conditions, we consider that each SRAM cell has an embedded inherent error rate across multiple power-ons. Roel Maes introduced heterogeneous error modeling with cell-specific error probabilities to evaluate the reliability of PUFs with high accuracy [79]. We utilized the proposed 2-parameter error model to assign the error rates to our SRAM cells. $10,000$ SRAM cells were simulated in 45 nm CMOS technology [6] in the presence of thermal noise and the error rates for each instance were obtained across 1000 power-ons. The data was curve-fitted to obtain the relevant parameters, found to be $\lambda_1 = 0.2916$ and $\lambda_2 = 1.9062$. Utilizing

Table 4.2: Uniqueness and Reliability results for WiSARD PUF variants [108]

PUF Type	**Uniqueness** (d_{inter})	**Reliability** (d_{intra})
WiSARD PUF	0.4995	0.0375
WiSARD PUF (fixed tuple)	0.4917	0.0385
RM-WiSARD PUF	0.4855	0.0477
WiSARD PUF + Extra bits	0.4771	0.0415
WiSARD PUF + Tuple rotation	0.4957	0.054
RM-WiSARD PUF + Extra bits	0.4775	0.0573
RM-WiSARD PUF + Tuple rotation	0.4861	0.0687

the methodology proposed by Maes [79], we can generate error rates for an arbitrary number of SRAM cells that will constitute the WNN PUFs.

The WiSARD PUF designs are simulated to obtain the responses across multiple power-ons applying the same challenges. For each design, 100 PUF instances were analyzed where each one received ($n =$) 1000 challenges and for each challenge ($m =$) 100 noisy responses were generated by simulating multiple power-ons for the SRAM cells. Table 4.2 presents the reliability results for the various PUF designs. We notice that the original WiSARD PUF performs better than all the other variants. Furthermore, in the extended tuple generation designs, the extra bits offered better reliability in comparison to tuple rotation scheme, in contrast with the uniqueness results. Possibly, utilizing different processing schemes for the RAM block outputs can further improve reliability. Those schemes will be targeted in the future works.

4.4.4 Machine Learning Resistance

Popular techniques like Logistic Regression (LR) and Support Vector Machine (SVM) have demonstrated the ability to model previous digital Strong PUF designs [107]. Sophisticated machine learning (ML) based on ensemble meta-algorithms, like Gradient Boosting, were shown to break even analog Strong PUFs, that were initially resistant to SVM, by Vijayakumar et al.[125]. In this work, we consider LR, SVM and Gradient Boosting (Grad Boost) to measure the attack resilience of the proposed designs.

The machine learning algorithms were implemented in Python using the scikit-learn tools [9]. Gradient Boosting was configured with the number of estimators set at 128 and learning rate of 0.01. For LR, we set the inverse of the regularization strength to a value of 10^{-5}. SVM utilizes radial basis function (RBF) kernel machines to model non-linearly separable functions as linearly separable in higher dimensions. We analyze 100 PUF instances for each design and collect $150,000$ CRPs for each PUF instance of which $100,000$ CRPs were used for *training* to obtain the cloned PUF model and $50,000$ CRPs were applied for *testing* the machine learning accuracy. An ideal Strong PUF will have machine learning accuracy of 50 %, which is akin to random guessing.

The *mean* and *standard deviation* of machine learning accuracy obtained through discussed ML algorithms for each PUF architecture are tabulated in Table 4.3. Gradient Boosting offered the best learning accuracy and all RM-WiSARD PUF variants showed high machine learning resistance across the various learning algorithms with 'RM-WiSARD PUF with Tuple rotation' providing the best results. We also observed that applying extra bits or tuple rotation modifications to the original WiSARD PUF worsened the *mean* modeling attack resistance for those PUFs. However, these results were generated for a fixed challenge-to-tuple mapping. It is possible that changing

Table 4.3: Machine Learning results for WiSARD PUF variants [108]

PUF Type	**Machine Learning Accuracy**					
	Grad Boost		**SVM**		**LR**	
	μ	σ	μ	σ	μ	σ
WiSARD PUF	0.79	0.016	0.690	0.020	0.637	0.246
WiSARD PUF (fixed tuple)	0.790	0.017	0.687	0.026	0.630	0.032
RM-WiSARD PUF	0.612	0.028	0.585	0.01	0.583	0.048
WiSARD PUF + Extra bits	0.815	0.018	0.722	0.028	0.652	0.033
WiSARD PUF + Tuple rotation	0.822	0.019	0.662	0.017	0.600	0.023
RM-WiSARD PUF + Extra bits	0.667	0.047	0.61	0.04	0.602	0.039
RM-WiSARD PUF + Tuple rotation	0.594	0.011	0.584	0.008	0.584	0.008

this mapping may affect the distribution of accuracies across the variants of WiSARD PUF. However, the mapping was not one of the control variables in our experiments.

4.4.5 Hardware Analysis

As all presented the designs in Section 4.3 integrate a Weightless Neural Network hardware and extra resources for concatenated code and fuzzy logic versions, the

main resource costs originate from the number of SRAM cells, area cost of final response bit generation and challenge-to-tuple mapping. The fixed mapping, shown in Figure 4.2, can be achieved with minimal resources and most designs require $\leq$ 2K SRAM bits aside from extra-bits schemes which require large number of SRAM bits, as seen in Table 4.1. Unit SRAM cell size of $0.346\mu m^2$, quoted by Intel [89], in 45 nm technology node gives us an area of $710\mu m^2$ for 2K SRAMs with additional area coming from interface circuitry. If an IC has the option of also utilizing the WNN for neural network applications, then we can amortize their resource costs over both applications. Reed-Muller decoder hardware implementation detailed by Bösch [29] achieved $248.976\mu m^2$ area for RM$(1, 3)$ and the repetition code decoder 31.92 μm^2 area, both costs determined by 45 nm standard cell library [5]. These resources represent a small overhead in comparison to the size of the RAM blocks and further reduction can be achieved by sharing decoder hardware and serializing the PUF operation.

4.5 Reliable Strong PUF Implementation

In this section, we explore the construction of a Strong PUF using reliable Weak PUFs as the initial entropy source and combining them with Weightless Neural Networks (WNNs), using the WiSARD model. Further, we explore alterations to the basic WNN architecture, presented in Section 4.3, with the intention of improving the machine learning resistance of the Strong PUF. All presented architectures produce a 1-bit output response, given an m-bit input challenge.

4.5.1 Reliable Weak PUF Entropy Source

To construct a reliable Strong PUF, we utilize Weak PUFs as the sources of entropy in the PUF system. Such an approach allows us to harness the extensive research carried out into making Weak PUFs reliable, as discussed in Section 2.2.

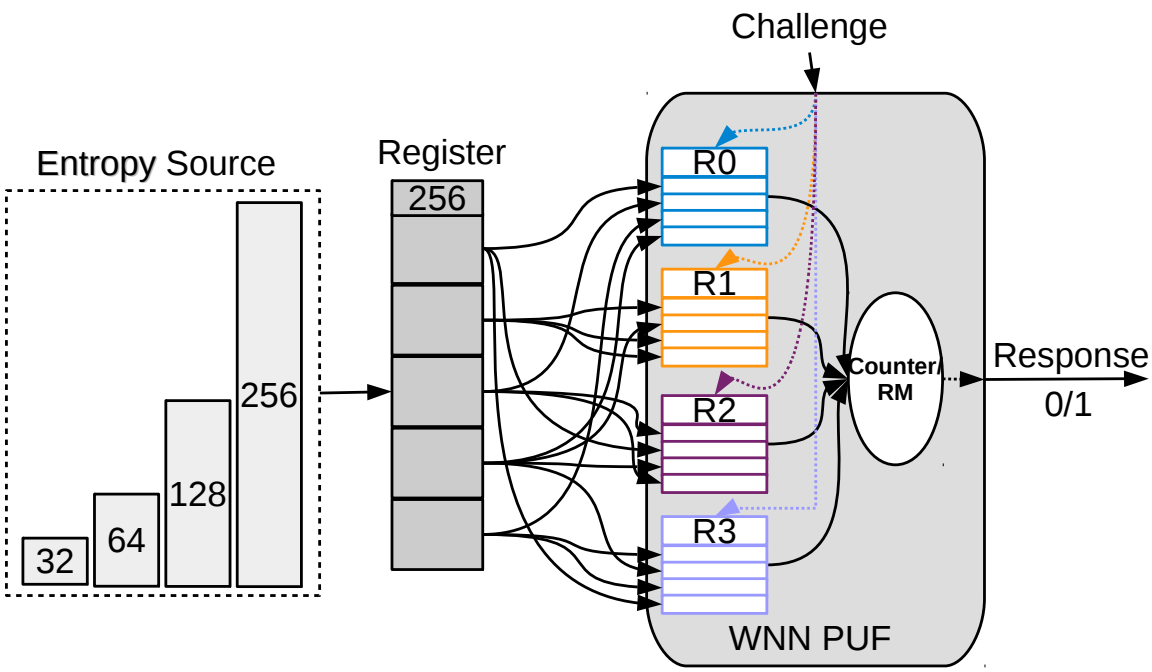


Figure 4.6: Reliable Strong PUF implementation

Using potent Weak PUF designs [96], robust error correction mechanisms [126] and techniques such as accelerated aging and masking [85] allows us to create a highly reliable binary string (or *secret key*) from an array of Weak PUFs. Such a binary string represents the process variation dependence of the final Strong PUF design and is unique to each IC.

In this work, we seek to find the smallest length of the binary string required for creating a Strong PUF that possesses high uniqueness and machine learning resistance. Having a smaller entropy source results in smaller area for the Strong PUF. For our work, we study the case where we have 256, 128, 64 or 32 initial reliable Weak PUF bits that will be used to fill the WNN RAM cells.

4.5.2 Complete Strong PUF architecture

The complete Strong PUF implementation requires that the RAM contents be filled with the reliable Weak PUF bits in an equi-probable manner to reduce bias

in the system. For example, if there are 2^{10} total RAM locations in the WNN and we obtain 2^8 reliable bits from Weak PUFs, then each bit needs to be randomly mapped to $2^{10}/2^8$ $(= 4)$ unique locations. The random mapping can be implemented in hardware by either using a crossbar network or hard-wiring each Weak PUF bit to the appropriate RAM cell.

The complete Strong PUF, as illustrated in Figure 4.6, assumes a 256-bit register is used to store the Weak PUF bits. The register itself obtains its data from an initial number of reliable Weak PUFs, ranging from 32 to 256 bits. When considering smaller number of initial reliable bits, we make the relevant number of copies to get a total of 256 bits. The contents of the register are, then, used to load the WNN RAM locations. The WNN processes the input challenges to produce the final 1-bit response.

4.6 Reliable Strong PUF - Experimental Setup and Results

In this section, we evaluate the proposed reliable designs discussed in Section 4.5 to find out how the size of the initial *entropy source* in the form of reliable Weak PUF bits influences the Strong PUF machine learning resistance and uniqueness. Next, we provide an analysis of the results with the purpose of identifying the minimum entropy source size that can provide higher machine learning resistance and high uniqueness. Since in Section 4.5 the Weak PUFs have high reliability, for these experiments we consider all architectures with 100% reliability.

4.6.1 Setup

All PUF experiments in this section are simulated in Python and the relevant SRAM circuit parameters are obtained from SPICE simulations using 45 nm transistor models [6]. Each PUF receives a 64-bit input challenge and produces a 1-bit response. The Discriminator SRAM cell contents are filled with the contents of the 256-bit

Table 4.4: Reliable WiSARD PUF Architectures with 64-bit Challenges

	#RAM Blocks	***# Addresses***	***# SRAMs***
WiSARD PUF	9	$7 \cdot 2^8 + 2 \cdot 2^4$	$1,824$
RM-WiSARD PUF	8	$8 \cdot 2^8$	$2,048$
WiSARD PUF + Tuple rotation	9	$7 \cdot 2^8 + 2 \cdot 2^4$	$1,824$
RM-WiSARD PUF + Tuple rotation	8	$8 \cdot 2^8$	$2,048$

register which, in turn, is loaded by the Weak PUF entropy source, as described in Section 4.5.2. The Weak PUFs in the entropy source are assumed to have equal number of 1s and 0s and are unique across PUF instances.

The general details about WNN designs simulated in this section are tabulated in Table 4.4. Note that the extra bits schemes were not included for these experiments due to worst results discussed in Section 4.4. For designs that have 9 total RAM blocks, 7 are addressed by 8-bit tuples while the remaining two use 4-bit tuples. The designs with tuple rotations assume 2 rotations and both RM-WiSARD PUF design variants utilize the RM$(1, 3)$ decoder for response generation. Finally, all the designs assume a fixed input-to-tuple mapping across PUFs, as depicted in Figure 4.2.

4.6.2 Uniqueness

Uniqueness was defined in Section 4.4.2. For this experiment, 100 entropy source-to-WNN random mappings were generated for each PUF architecture. For each instantiated Strong PUF of any architecture, ($n =$) 1000 challenges were applied and ($k =$) 100 such instances were evaluated. The entropy size was varied to 32, 64, 128 and 256 Weak PUF bits in each PUF instance, as illustrated in Figure 4.6.

Table 4.5: Uniqueness Standard Deviation results for PUF architectures with varying Entropy Source sizes

PUF Type	**Standard Deviation of Uniqueness (σ)**			
	32	**64**	**128**	**256**
WiSARD PUF (fixed tuple)	0.0869	0.0586	0.0412	0.0314
RM-WiSARD PUF	0.0238	0.0194	0.0173	0.0164
WiSARD PUF + Tuple rotation	0.1413	0.0905	0.0581	0.0417
RM-WiSARD PUF + Tuple rotation	0.0319	0.0222	0.0185	0.0170

The mean inter-class HD was found to be close to the ideal 0.5 ranging between 0.4661 to 0.5001 for all designs and entropy source sizes. The minimum standard deviation results for the various PUF designs are tabulated in Table 4.5, representing the *best* entropy source random mapping among the 100 generated mappings. We observe that standard deviation decreases as the entropy source size increases in any given design. The original RM-WiSARD PUF with a 256-bit entropy source offered the lowest standard deviation with the mean inter-class HD of 0.4858.

4.6.3 Machine Learning Resistance

Machine Learning Resistance was defined in Section 4.4.4. In this experiment, we use Gradient Boosting (Grad Boost) to estimate the attack resistance of the proposed reliable designs as it consistently outperforms LR and SVM.

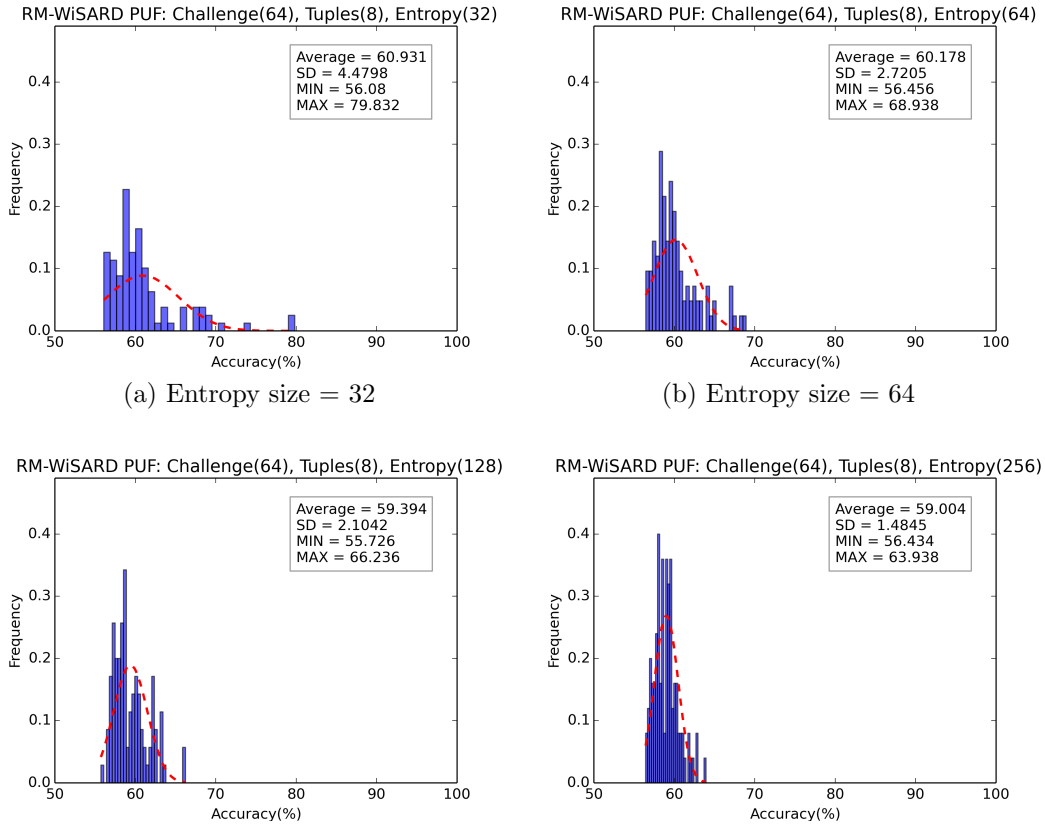
Gradient Boosting was implemented in Python using the scikit-learn tools [9] with the number of estimators set at 128 and learning rate of 0.01. For each scenario, the 10 best entropy source mappings were selected from the uniqueness results, representing

the 10 smallest standard deviations. These mappings help shortlist the candidates for machine learning analysis and reduce the number of experiments. $150,000$ CRPs were applied to each PUF instance out of which $100,000$ CRPs were used for *training* a model with Gradient Boosting. The trained model was tested using $50,000$ CRPs to measure the machine learning accuracy of the cloned PUF. For each selected design, 100 PUF instances were simulated.

Figure 4.7 presents the distributions of machine learning accuracies for varying entropy sizes for the RM-WiSARD PUF implementation considering the best mapping of the 10 shortlisted entropy source mappings. This implementation provided the best machine learning accuracy results compared to the other architectures. As with the uniqueness results, an increase in the entropy source size results in decreased machine learning accuracy and also, a smaller standard deviation. The average machine learning accuracy for each PUF architecture with varying entropy source sizes is summarized in Table 4.6. For faster analysis, the random entropy mapping is chosen for each design by considering the results from an entropy source size of 256 bits and then, changing the size while keeping the same mapping. We see that the RM-WiSARD PUF variants offered higher machine learning resistance than the simple WiSARD variants. Moreover, we see that we are still able to get $\leq 65\,\%$ machine learning accuracy by considering just 32 initial reliable Weak PUF bits. Hence, the RM-WiSARD PUF with an entropy size of 32 can offer high machine learning resistance while still exhibiting good uniqueness. This aids us in realizing a smaller Strong PUF.

4.7 Discussion

In this section, we discuss certain aspects of our PUF implementations and also, detail possible future improvements to the system.



(a) Entropy size = 32

(b) Entropy size = 64

(c) Entropy size = 128

(d) Entropy size = 256

Figure 4.7: Gradient Boosting machine learning accuracy distributions for RM-WiSARD PUF

Attack Scenario: For our work, we assume that the PUF is a *black box* from the perspective of an attacker and only the PUF CRPs can be intercepted for use in model building attacks. This requires that the designer take the necessary precautions: ensure that RAM block contents, required in the neural network implementation, are not accessible outside the system and prevent any data leakage from PUF system, especially the Weak PUF bits.

Further, the black box perspective offers protection, when utilizing just 32 initial reliable Weak PUF bits, against brute-force guessing of possible bit values. In reality, a designer may choose to have a larger entropy source to increase security. Future works will focus on studying the security of the PUF in case an attacker has knowledge

Table 4.6: Gradient Boosting-based Machine Learning Accuracy for WiSARD PUF variants

PUF Type	**Machine Learning Accuracy (%)**			
	32	**64**	**128**	**256**
WiSARD PUF	82.60	81.87	81.74	81.26
RM-WiSARD PUF	60.93	60.17	59.39	59.00
WiSARD PUF + Tuple rotation	85.22	83.40	82.32	81.74
RM-WiSARD PUF + Tuple rotation	64.87	62.44	60.93	59.15

of the PUF architecture due to possession of a physical device. Possible fault injection attacks will also be explored.

Hardware Implementation: One of the major resource costs for the reliable designs is the number of WNN memory cells required. As seen from Table 4.4, the WNN implementations require a maximum of 2K bits and the details of hardware resources is discussed in Section 4.4.5.

The other major area intensive unit is the Weak PUF block required to generate the reliable bits for the WNN. From Table 4.5 and Table 4.6, we can see that the RM-WiSARD PUF architectures allow us to achieve both high uniqueness and machine learning resistance by considering just 32 initial reliable Weak PUF bits. Using the details provided in the work by Vijayakumar et al.[126] and assuming a 45 nm standard cell library [5], we assume the 16 : 1 mux implementation and a 5 % observable yield loss that necessitates we use 48 ($\in 16\mathbb{Z}$) to obtain the final reliable 32 bits. This gives us an area of $\approx 550\mu m^2$, which is significantly less than the area that the WNN memory cells need.

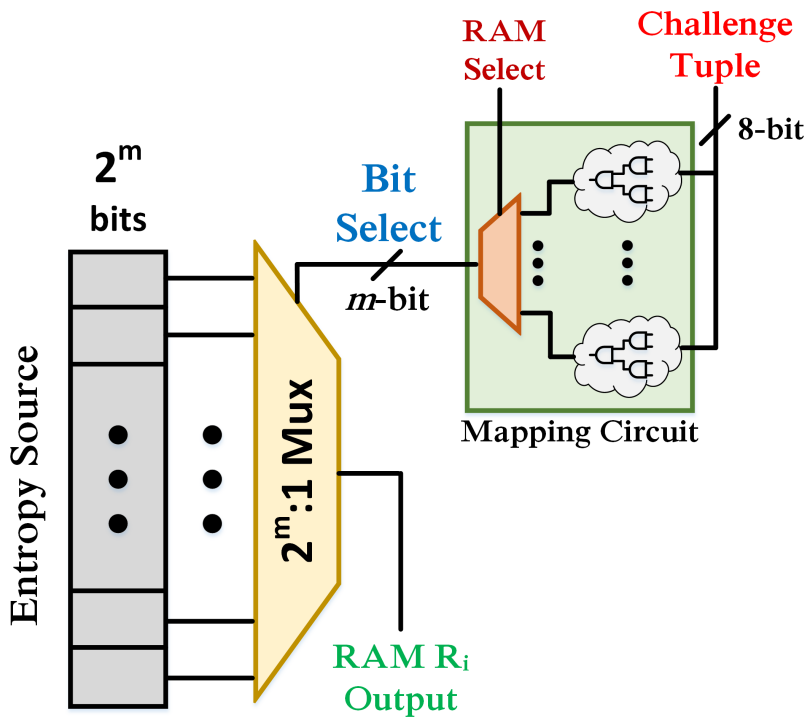


Figure 4.8: Combinational logic-based implementation of Strong PUF

We note that it is possible to use keyed hash functions, like Keccak [48], with Weak PUF bits to obtain a Strong PUF implementation. The area for an efficient hash implementation was found to be 2280 gate-equivalents [48] or $\approx 1900\mu m^2$ in 45 nm standard cell library [5]. To make a comparison with our designs, the area needed to generate the initial stable Weak PUF bits can be considered necessary to both the keyed hash and our PUF design and hence, excluded from comparison. Further, as noted in Section 4.4.5, the area for the RAM-based version of our design is highly dependent on practical implementation considerations.

An alternative PUF implementation is based on two mapping operations. The first mapping is between the input challenge to the tuple for addressing the RAM blocks. The second mapping is for the RAM content based on the Weak PUF entropy source. These maps can be implemented as a 2D crossbar or as combinational logic. Figure 4.8 shows the schematic for a combinational system. The *RAM Select* is used for virtual

Table 4.7: Combinational Logic implementations of PUF with varying Entropy Sources

Entropy Source (bits)	Area (μm^2)
32	1710
64	2060
128	2400
256	2650

RAM block (R_i) selection while the specific bit is selected via *Bit Select* control. Table4.7 summarizes the silicon areas, in 45 nm standard cell library [5], for varying sizes of the entropy source. We see that the *randomly mapped* implementation of the proposed Strong PUF compares favorably with the *optimized* Keccak design.

4.8 Conclusion

Strong PUFs have been promised to provide a low cost alternative to cryptography-based authentication. However, unreliability in Strong PUFs can increase the number of CRPs needed for proper authentication and low resistance against model building attacks using machine learning techniques can create a security risk. Hence, it is advantageous to develop a Strong PUF architecture that possesses high reliability and also, offers greater resistance to machine learning attacks. In this work, we seek to leverage the extensive work done to obtain high reliability for Weak PUFs and integration of neural networks in hardware to create new Strong PUF designs that can offer high reliability while maintaining high machine learning resistance and uniqueness. This work proposes a novel Strong PUF architecture composed of WiSARD Neural Network (WNN) and further explores variations of such a design. To obtain greater reliability, we extend the designs by using an initial, reliable Weak PUF entropy source mapped into the WNN and analyze the minimum entropy source needed to ensure a Strong PUF. Our results show that it is possible to create highly reliable

Strong PUFs with $< 65\,\%$ ML accuracy by using as few as 32 reliable Weak PUF bits.

CHAPTER 5

REALIZING ROBUST, LIGHTWEIGHT, MODELING ATTACK RESISTANT STRONG PUFS FROM WEAK PUFS

5.1 Introduction

Building on the work in Chapters 2 and 3, in this chapter, we propose a Strong PUF system that utilizes reliable SRAM-based Weak PUFs as the initial entropy source against which the states of a linear-feedback shift register (LFSR) are compared using Hamming distance (HD) as the metric. We find the LFSR state that is closest to the Weak PUF string, *i.e.* the state which yields the *minimum* HD. The number of iterations, termed the *index*, needed by the LFSR to reach the target state from its initial seed state is recorded as the output response. The LFSR seed state is set by an external challenge and this, in turn, changes the observed indices for the same Weak PUF source against different challenges. Our scheme handles a large Weak PUF string by dividing it into substrings of the length of a smaller LFSR and collecting a set of indices. A concatenation of these indices forms the output response of our PUF system. We also extend our mechanism to incorporate rounds where the response output of the first round becomes the challenge seed for the next round. Our results show that it is possible to create a *lightweight*, *robust* key generation system that exhibits ideal Strong PUF characteristics with respect to uniqueness and modeling-attack resistance while maintaining the high degree of robustness offered by the Weak PUFs. Our proposed system has small area overhead and produces large set of responses with the same length as the initial Weak PUF string. Also, the Weak PUF source is never exposed to unauthorized parties.

5.2 Motivation

We have explored relevant research relating to the reliability of Weak PUFs and on Strong PUFs with particular focus on modeling-attack resistance in Sections 2.2, 3.2 and 4.2. In this section, we briefly discuss the motivation for this work.

Paral and Devadas proposed a pattern matching key generation (PMKG) scheme that could alleviate the effects of noise on PUF outputs without the need for error correction codes, while providing a strong key that does not expose the PUF bits. Pattern matching is done using Hamming distance (HD). The initial PUF output is divided into substrings which are accessed based on input challenge. A substring is associated with an index and a set of indices are used to generate the final key while the PUF substrings are stored as helper data to aid in error correction [94]. Komano et al.proposed a more optimized PMKG solution offering greater security [68]. However, these PMKG schemes have significant area overheads and storage requirements for auxiliary data to produce a robust key.

Our goal is to utilize the lightweight hardware-based Weak PUF reliability enhancements to generate an initial set of robust bits. Then, we utilize the concepts from the previously discussed PMKG schemes to design a system that can utilize the robust Weak PUF bits and produce secure responses to input challenges without exposing the initial Weak PUF bits. We show that it is possible to implement a reliable and secure Strong PUF that is also lightweight compared to previous designs. Also, our design takes an n-bit input challenge and produce an n-bit output response.

5.3 Proposed Method

In this section, we will first explore the basic implementation of the LFSR-based key generation system. Based on this foundation, we will next discuss various modifications and extensions to the system to make it more robust in terms of security.

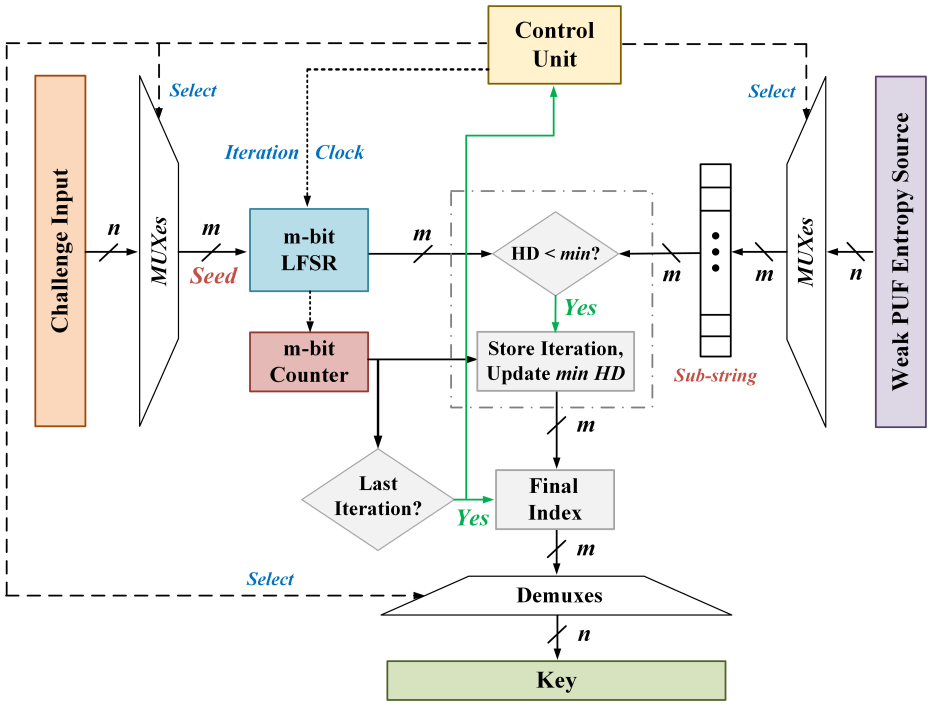


Figure 5.1: Illustration of the basic key generation system

5.3.1 Basic System

Figure 5.1 illustrates the operation of a basic LFSR-based key generation system with an initial n-bit Weak PUF entropy source. An m-bit Galois LFSR ($m <= n$) is used to generate the various states used for comparison with the Weak PUF entropy source. We record all the shift registers in the LFSR to produce the m-bit output or state per cycle. Using a smaller LFSR necessitates that the PUF string be divided into substrings of size m, which are then processed individually. For simplicity, we assume that m is a divisor of n. The seed to the LFSR is derived from an n-bit *challenge* input. Hence, multiplexers are used to select and forward the relevant seed/PUF substring to the LFSR during operation. If the seed input to the LFSR contains all zeros, we add 1 to the input LSB to prevent the LFSR from locking up. Substring selection is done by the *Control Unit*, which also produces the clock for the system.

In our work, we choose a primitive polynomial, or *taps*, for the LFSR feedback so as to ensure a *unique* output sequence of maximal length, *i.e.* $2^m - 1$ states. An *iteration* during operation is defined as obtaining the next m-bit LFSR state. An m-bit counter is used to track the number of iterations and is used to obtain the *index*. Once the LFSR is initialized with the seed, we obtain the next state for comparison with the PUF substring. Also, an *index register* is initialized to 0. Hamming distance (HD) between the LFSR state and the PUF substring is calculated and stored as the initial minimum HD in an *HD register*. During the subsequent iterations, the calculated HD is compared with the previously stored minimum HD and updates are made to the stored HD and index registers if the new HD is *lower*. As we use a maximal-length LFSR, the number of iterations needed to process all states is $2^m - 1$ and is checked using the *Last Iteration* condition. Using a maximal-length LFSR allows the index to cover the full range of values, *i.e.* $\in \{0, \ldots, 2^m - 1\}$. Otherwise, the LFSR states will have a cycle length $< 2^m - 1$, producing repeating states. Since we only update the index when a new HD lower that the current minimum is found, the final index can have most significant bits that are always 0. In contrast, updating the index register whenever the calculated HD $\leq minHD$ introduces bias in the least significant bits as the minimum state is likely to repeat at least once before the counter goes through all $2^m - 1$ cycles. Such bias is translated to the final output, reducing the entropy in the system.

Once each PUF substring is processed, the index output is directed towards the *Final Key* register using demultiplexers. Control Unit selects the next PUF substring and corresponding LFSR seed bits from the Challenge input. At the end of the key generation operation, we obtain an n-bit key after processing all the substrings.

In the aforementioned design, an attacker can recover the circuitry by reverse-engineering the hardware and figure out the taps utilized in the LFSR. Since the LFSR is a linear system, observing the input challenge seed and the output key can

allow the attacker to recover the initial Weak PUF bits, breaking the system. Even without hardware reverse-engineering, an attacker might be able to figure out the LFSR taps using the Berlekamp–Massey algorithm by just observing the inputs and outputs to the system. Hence, we need to modify our basic design to improve its robustness.

5.3.2 Key Generation with Rounds

A simple modification is to perform multiple rounds of the entire operation by using the key generated from one round as the input challenge in the next. The final output key will have a significantly high degree of non-linearity regarding its relationship with the initial Weak PUF entropy source and input challenge. Such recursive operations have been proven to be highly advantageous in the implementation of many cryptographic functions, such as Advanced Encryption Standard (AES). In such a case, the challenge input register can be re-used to store the key from the current round for the next round. For example, in the first round, once a particular challenge substring has been processed, the resulting index output can be written back to the location of that challenge substring. This allows us to save area needed for additional registers.

5.3.3 Tap Selection

The first possible modification is to introduce the ability to change the taps of the LFSR. An LFSR can have multiple primitive polynomials or tap selections that results in maximal length sequences and the order of states is unique to a sequence. Hence, we can choose between the polynomials based on certain conditions to realize different LFSRs. The LFSR is suitably modified to allow for such an operation. Given our goal that the initial Weak PUF entropy source bits should not be exposed to the outside world when a device is being authenticated, these *secret* bits can be used as inputs to a *selection function* that chooses one of the maximal-length LFSR taps.

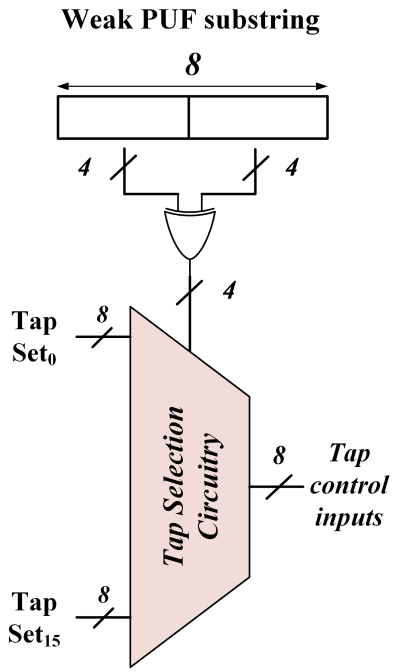


Figure 5.2: Illustration of the LFSR tap selection system

Hence, the LFSR behavior is now dependent on the system's secret and an attacker would have to guess the Weak PUF bits to find out the taps used in the LFSR, increasing the complexity of the attack. We note that the tap selection is specific to the PUF substring and does not change with respect to any other input. We term this as a *static* tap selection.

For the basic system described in Section 5.3.1, Figure 5.2 illustrates how we implement the tap selection system for $m = 8$. An 8-bit LFSR has 16 primitive polynomials. We take the first 4 bits (or nibble) and the last 4 bits of the 8-bit PUF substring and process them by performing bitwise exclusive OR operation to get a 4 bit output. Thus, we need 4 XOR gates in hardware. This becomes the input to the selection function, represented by a multiplexer array. The 8 bit output of this function sets the taps of the LFSR for that particular 8-bit PUF substring.

While the aforementioned tap selection process is *static* with respect to the input of a particular round, it is possible to make it *dynamic* by making the tap selection

input dependent on both the PUF substring and the input seed substring. As we perform rounds, the input seed changes and hence, will influence the selected taps for a particular substring. Here, the PUF substring can be considered as providing the *secret* static offset to the tap selection input. We process the seed substring in the same manner as in Figure 5.2 to get a 4-bit string which is then combined with the 4-bit string obtained from the PUF substring to get the final tap selection input. Hence, we will need 12 (or 3 sets of 4) XOR gates in total to realize the *dynamic tap selection*.

5.3.4 Coupling the Index output and Seed

Another possible modification is to perform bitwise *exclusive or* operation between the m-bit index output from processing one PUF substring and the m-bit LFSR seed input used for processing the next selected PUF substring. This affects the next substring's index output as the initial state of the LFSR is changed compared to using the initial seed bits directly. The attacker will need to know the order of PUF substring selection and then, work their way backwards. Combined with tap selection and employing multiple rounds, this increase the attack complexity significantly by introducing significant interdependence between the output bits.

The selection sequence of the next PUF substring to process during a given round can be set by the designer. In this work, we analyze two types of selection sequences.

5.3.4.1 Consecutive Selection

The simplest sequence is to select consecutive PUF substrings for processing. For the round-based scheme, the output of the last substring processed in a particular round influences the seed utilized for the first substring selected in the next round. In the first round, the seed for processing the first substring is taken directly from the challenge inputs as there is no previous index output to use.

5.3.4.2 AES-type Selection

We explore an alternate selection scheme to see if the byte selection order has an effect on the PUF metrics. In particular, the byte selection order of the *ShiftRows* step of the AES is used as a reference. In a 128-bit AES, 16 bytes need to be processed in a round. If the bytes are numbered as $\{0, 1, 2, \ldots, 15\}$, then we can use (5.1) to produce the necessary byte sequence in any round. Each round will select all 16 bytes in a unique order. Such an approach has the advantage of the selection sequence changing in each round compared to consecutive selection.

$$select_{i+1} = (select_i + 1 + round * 4) \,\mathtt{mod}\, 16 \tag{5.1}$$

where $select_{i+1}$ indicates the next substring to choose based on the current substring $select_i$ and the *round* number ($\in \{0, 1, 2, \ldots\}$). The pattern generated in a particular round repeats every 4 rounds.

For processing 8 substrings, (5.1) can be modified by replacing 16 with 8 and 4 with 2. This will also have a 4 rounds cycle.

5.3.5 Multi-word Entropy Source

Previously, both the input challenge and the initial Weak PUF entropy source were considered to be of the same length, n bits. Now, we consider a case where the entropy source affords us a larger number of bits than the input challenge such that we can split the entropy source into multiple n-bit *words*, as shown in Figure 5.3. This allows us to select a different word from the entropy source in each consecutive round, further increasing the complexity for an attacker. If we can split the entropy source into k n-bit words, then it is possible to use an n-bit input challenge to create $n \times k$-bit output key in k rounds while utilizing the entire entropy source instead of having to re-use it as in the case of basic system with rounds. We can also output a regular n-bit key after each round if so desired.

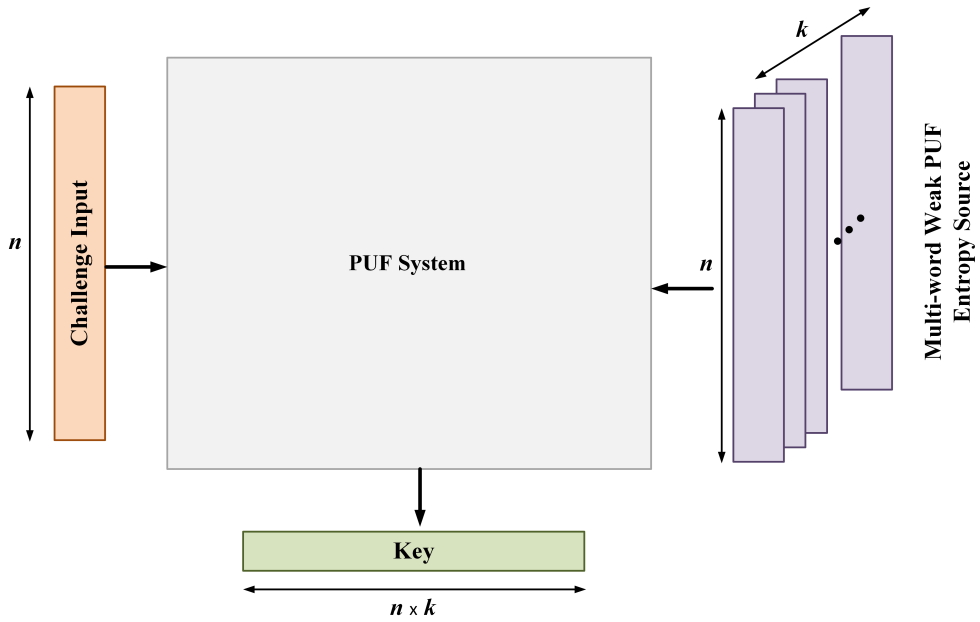


Figure 5.3: Illustration of the key generation system with Multi-word Entropy Source

5.4 Experimental Setup and Results

In this section, we describe the PUF system specifications used for generating the results. Next, we analyze the results from implementation and simulation of our system with respect to uniqueness and modeling attack resistance.

5.4.1 Setup

In this work, we first assume that the initial Weak PUF entropy source can produce (n =) 64 reliable bits. We utilize an (m =) 8-bit *Galois* LFSR to generate the states for Hamming distance (HD) comparison. So, the 64-bit initial Weak PUF and input challenge strings are divided into 8 8-bit sub-strings. Using a maximal-length/primitive polynomial provides us with 255 unique states and there are 16 such polynomials that can be utilized. The mapping function, as described in Section 5.3.3, takes a 4-bit input to select among the 16 possible tap combinations and outputs one such combination as an 8-bit vector to configure the LFSR.

For utilizing rounds-based scheme, as described in Section 5.3.2, we chose to consider a total of 4 rounds and record the outputs at the end of each round as a possible key for a particular challenge input. This allows us to analyze the effect of rounds on the PUF metrics. For all the analyzed design schemes, we perform the round-based operation. Consequently, the results for the first round will represent a design that does not use rounds.

For the results, the basic system, describe in Section 5.3.1, is termed as *Basic*. The byte selection schemes detailed in 5.3.4 are studied with the consecutive selection scheme termed as *Consec* and the AES-based scheme termed as *AESshift* in the results. For the byte selection schemes, we include the *static tap selection* modification, as discussed in Section 5.3.3 as part of the design. For the *Consec* variant, we also explore using dynamic tap selection, described in Section 5.3.3, instead of static selection and call this variant *Consec_DynTap* in the results.

5.4.1.1 Multi-word Entropy Source

For the case of a PUF system with a multi-word entropy source, as shown in Figure 5.3, we consider the sizes of the input challenge and a *word* to be $n = 32$. This allows us to analyze how our design scales compared to the 64-bit version. The value of $k \in \{1, 2, 4\}$ allows us to create keys with $(n \times k) \in \{32, 64, 128\}$ bits, respectively. For brevity, we analyze this design for the consecutive selection scheme with dynamic taps. We obtain results for $k + 1$ rounds and concatenate the results as necessary to obtain the final key. The rest of the parameters remain the same as for the aforementioned 64-bit design.

5.4.2 Attack Scenario

For our work, we assume that the attacker has knowledge of the PUF system design, but does not know the Weak PUF bits. Information about the Weak PUFs is only known to the authentication agent and can only be accessed once, during post-

manufacturing PUF testing. So, a man-in-the-middle type attacker has to rely on using machine learning algorithms to try and guess the internal Weak PUF bits using the observed challenge-response pairs (CRPs). Additionally, we assume the attacker can feed challenges to the system and observe the responses.

5.4.3 Uniqueness

Uniqueness is a property wherein, for the same challenge, each PUF generates unique responses compared to others PUFs. Inter-class Hamming distance (HD) is used for determining uniqueness. We calculate the Hamming distances between the responses of a pair of PUFs for the same challenge and average the results over many challenges and all possible PUF instance pairs. The average of inter-class HD, d_{inter}, can be defined as:

$$d_{inter} = \frac{2}{s(s-1)} \sum_{i=1}^{s-1} \sum_{j=i+1}^{s} \frac{HD(r_i, r_j)}{t} \tag{5.2}$$

where s is number of PUFs, t is the total number of 64-bit responses and $HD(r_i, r_j)$ is Hamming distance between responses of the PUF instances i and j to a particular challenge. A PUF is considered ideal when the normalized inter-class HD = 0.5 or 50 %.

For this work, ($s =$) 1000 64-bit strings were randomly generated and designated as the initial Weak PUF entropy source utilized in 1000 Strong PUF instances for each design variant considered. A set of 1000 ($= t$) challenges were randomly generated and applied across all PUFs. Hence, each PUF produces a total of 1000 64-bit responses.

For the purpose of analysis, we consider two uniqueness metrics. As we generate a 64-bit response string for each challenge, first we calculate the *string uniqueness* between pairs of response strings by performing bitwise exclusive-or operation on the strings. Next, we consider the *bitwise uniqueness* by considering each bit of a response separately to obtain 64 values representing the uniqueness metric for each bit. Then, we analyze the uniqueness of each bit individually across all challenges

Table 5.1: String Uniqueness Mean (%) for 64-bit Strong PUF design variants

	Round 1	**Round 2**	**Round 4**
Basic	50.09	49.87	49.88
Consec	49.98	49.99	49.99
AESshift	49.98	49.99	49.99
Consec_DynTap	49.99	50.01	50.02

and the entire PUF population. The *bitwise uniqueness* provides a fine-grained metric that can highlight potential biases in our PUF system by mirroring such biases in the bit positions.

Table 5.1 tabulates the normalized *string uniqueness* results, obtained using (5.2), for the first, second and last rounds. These results show that our basic design and all variants maintain a close to ideal inter-class HD across multiple rounds for a 64-bit signature response. *Bitwise uniqueness*, as illustrated in Figure 5.4, represents the normalized average inter-class HD for each bit in the selected rounds. The *Basic* design experienced a greater variation across bit positions compared to other design variants in the first round. Subsequent rounds served to smooth the variations closer to the ideal value of 50 %. The design variants were able to produce all bits with the same uniqueness across all rounds. Hence, we conclude that it is desirable to use a minimum of two rounds and that our design variants can realize a Strong PUF with high uniqueness.

5.4.3.1 Multi-word Entropy Source

For the 32-bit input challenge PUF system with varying k, as described in section 5.4.1.1, we generate 1000 keys of size = $\{32, 64, 128\}$ using 1000 randomly generated 32-bit input challenges. The population of PUFs is 1000 for each case.

The *bitwise uniqueness* results, as shown in Figure 5.5, show that our proposed design produces a normalized uniqueness close to the ideal value of 50 % across all bit

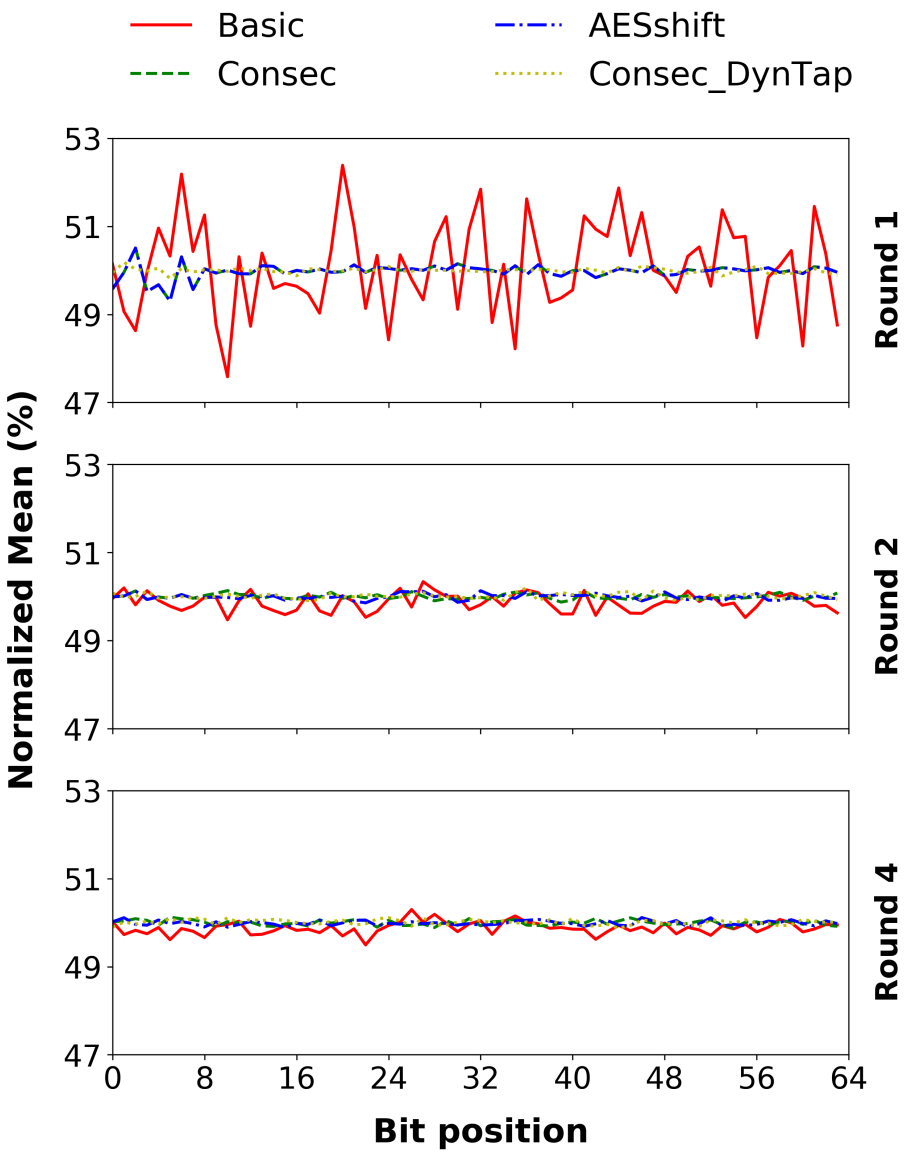
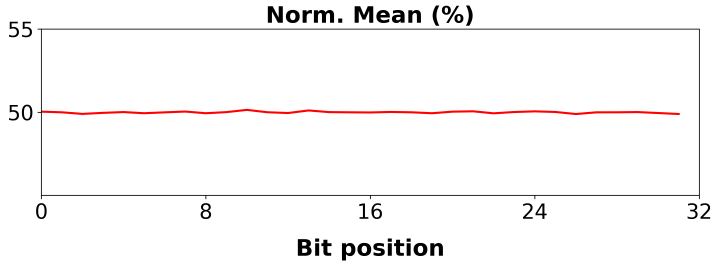
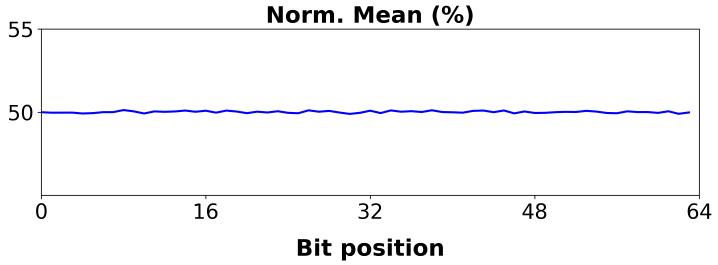


Figure 5.4: Bitwise Uniqueness across select Rounds

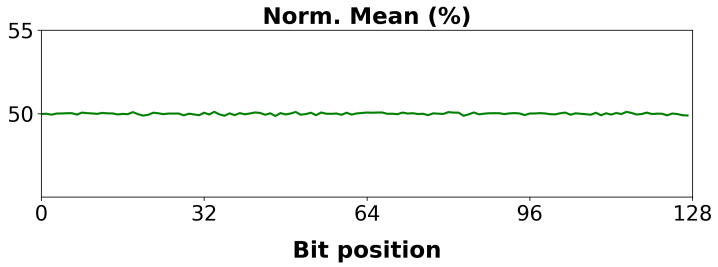
positions. Furthermore, for the case of $k = 1$ where the final key is of the same length as the input challenge (32-bit), we observe that our design scales well compared to the 64-bit PUF system described previously.



(a) Source = 32 bits



(b) Source = 64 bits



(c) Source = 128 bits

Figure 5.5: Bitwise Uniqueness for PUF with 32-bit Challenge and varying Entropy Source sizes

5.4.4 Machine Learning Accuracy

Popular techniques like Logistic Regression (LR) and Support Vector Machine (SVM) have demonstrated the ability to model previous digital Strong PUF designs [107]. However, they fail to target more resilient designs.

5.4.4.1 Gradient Boosting

Sophisticated machine learning (ML) based on ensemble meta-algorithms, like Gradient Boosting, were shown to provide significantly better machine learning accuracy by Vijayakumar et al.[125] and Santiago et al.[109]. Boosting algorithms iteratively learn several *weak* classifiers and assign weights to them based on their performance in terms of learning accuracy. This helps build the final *strong* classifier. Once a weak classifier has been weighted and assigned towards the final classifier, the misclassified datapoints are given higher priority while correct ones have lowered priority. Hence, future classifiers will focus on the misclassified points. With each iteration, the final classifier is strengthened and the prediction accuracy increases significantly.

Furthermore, the work by Ganji [44] has shown that Boosting algorithms can efficiently achieve probably approximately correct (PAC) learning of PUFs without explicit knowledge of their mathematical model. Hence, in this work we consider only Gradient Boosting as the primary choice for an attacker to clone the PUF.

The Gradient Boosting algorithm was implemented in Python using the scikit-learn [9] and lightgbm [14] tools . Gradient Boosting was configured with the number of estimators set at 200 and learning rate of 0.01. Since we have a 64-bit response output, we build 64 machine learning models to separately attack each bit. This is due to the fact that the full 64-bit response cannot be analyzed by current machine learning algorithms as this involves the algorithm being able to build a model that can output 2^{64} possible classes during Training, which is infeasible.

We create a subset of 100 instance from the 1000 Weak PUF instances used in Section 5.4.3. For each design, we collect $150,000$ CRPs for each PUF instance of which $100,000$ CRPs were used for *training* to obtain the cloned PUF model and $50,000$ CRPs were utilized for *testing* to obtain the machine learning accuracy. We obtain the accuracy metric for each bit position using the cloned model and the testing

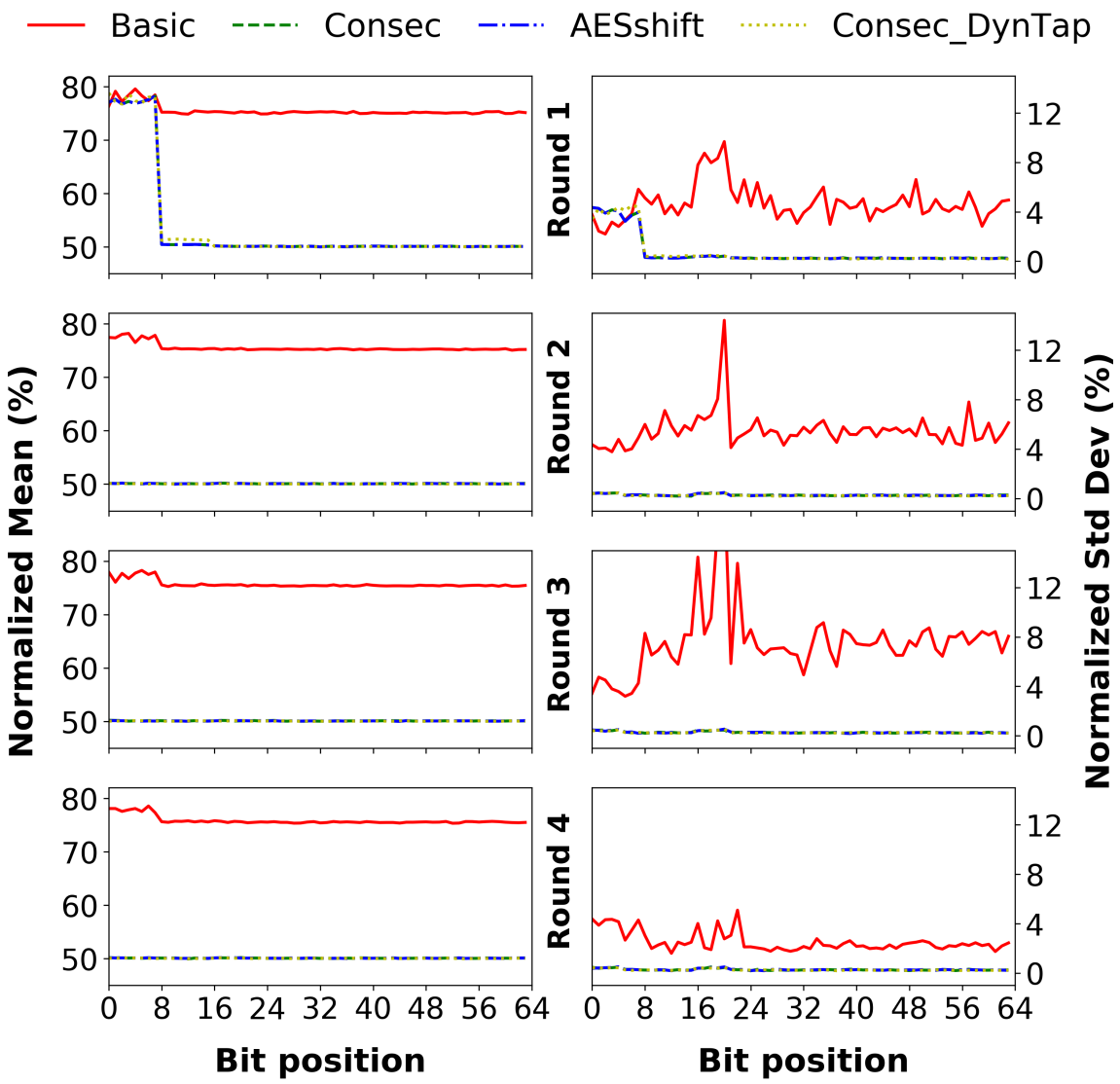


Figure 5.6: Machine Learning accuracy statistics for 4 rounds

CRP set. Hence, we get 64 distributions of accuracies over the entire population of PUFs considered for each design variant. Furthermore, such metrics are obtained separately for each round. An ideal Strong PUF will have a bitwise machine learning accuracy with a distribution possessing $\mu = 50\,\%$ and $\sigma = 0$, which represents a truly random entropy system.

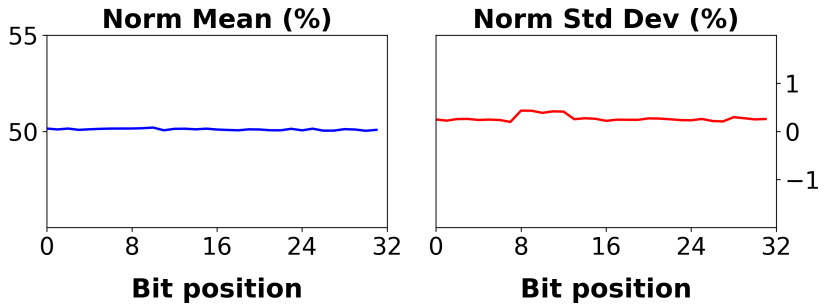
As shown in Figure 5.6, we see that the *Basic* design variant performs the worst across all bits and across rounds in terms of the mean and standard deviation ob-

served. All variants perform poorly for the first 8-bit substring processed in the first round. This illustrates that the coupling mechanism, described in Section 5.3.4, is crucial to ensure greater security as the first substring in the first round does not couple with any prior output. The order of byte selection is shown to not have an adverse effect of PUF metrics. Also, we note that we need to perform a minimum of two rounds to get close to the ideal machine learning accuracy metrics across all response bits. Particularly, we should not expose the response from the first round during authentication due to low ML resilience. We note that performing more rounds does not adversely affect the accuracy of the obtained response. Hence, to further improve security, the system can output the responses from every two rounds instead of every round. This will prevent an attacker from gleaning any useful information by trying to correlate the responses across rounds as they will be missing the responses from intermediate rounds.

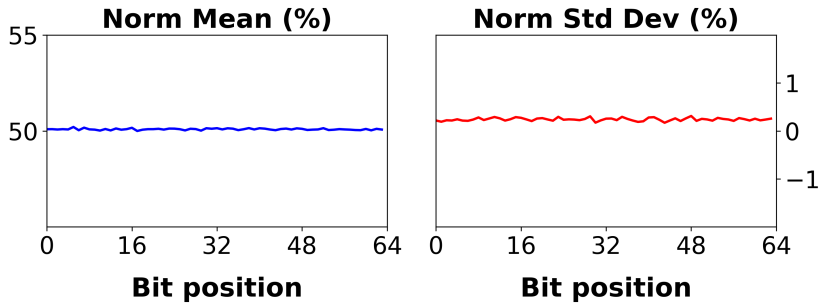
5.4.4.2 Multi-word Entropy Source

Similar to the 64-bit PUF system, we consider a population of 100 PUFs each for 32-bit input challenge designs with varying Weak PUF entropy sizes $\in \{32, 64, 128\}$. We collect 150K keys, of which 100K are used for training and 50K are used for testing. Gradient Boosting was used a the machine learning algorithm and the parameters remain the same as for the 64-bit version.

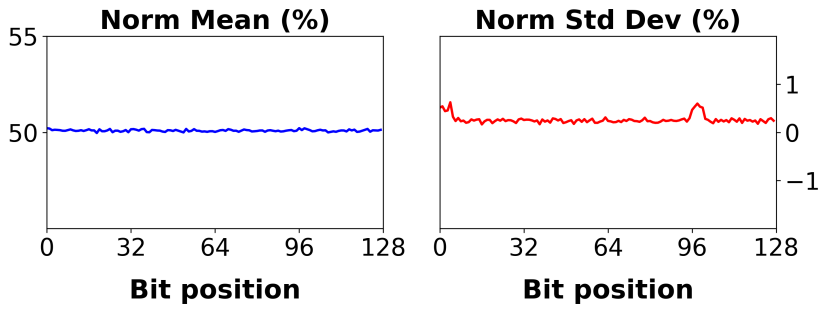
The simulations were run for $k + 1$ rounds to ensure that we can skip recording the output of the first round due to its greater susceptibility to attack, as discussed in section 5.4.4. The *bitwise* machine learning accuracy results are presented in Figure 5.7. We see that the normalized mean and standard deviation of the accuracy distribution for the designs with varying entropy sizes remains close to the ideal of 50 % and and 0, respectively. Hence, our design performs well regardless of the size of the input challenge or the entropy source.



(a) Source = 32 bits



(b) Source = 64 bits



(c) Source = 128 bits

Figure 5.7: Machine Learning Accuracy for PUF with 32-bit Challenge and varying Entropy Source sizes

5.4.4.3 Neural Network (NN) Attacks

Multiple recent works have focused on using Multi-layer Perceptron (MLP) based neural networks to successfully attack PUFs, especially n-XOR PUFs [20, 22, 58, 90, 110]. The work by Mursi et al.[90] was able to reduce both the number of CRPs required and the training times by more than an order of magnitude each while

achieving $\geq 98\,\%$ accuracy for upto 9-XOR PUF. Furthermore, the authors proposed an architecture where the number of neurons varies based on the number of PUFs being XOR'ed (n-XOR).

In this work, we utilize the 5-layer (3 hidden layers) neural network proposed by Mursi et al.[90] to attack the 64-bit *Consec_DynTap* design variant utilizing a 64-bit entropy source. For this purpose, we simulated 10 such PUFs and collected 4 *million* CRPs each. In particular, we focused on the responses from Round 2 as our primary outputs for analysis. 64 separate models are generated to attack each PUF response bit. We utilize the parameters recommended for 9-XOR PUFs [90] while setting the number of epochs to 200. The total CRP dataset size is varied by 1, 2 and 4 million and we set training to test dataset ratio to $9:1$.

Figure 5.8 illustrates the average mean and standard deviation of the accuracy across all 64 PUF response bits, across the population and for the various dataset sizes. The results show that our PUF design is still resilient with 4 million CRPs being exposed. In practice, since we do not need to store the CRPs in a database server, the attacker needs to spend time to collect such a large population of CRPs and hence, expend considerable resources for a single device under attack.

5.4.5 Hardware Implementation

The 64-bit design variants were synthesized using 45 nm Nangate open cell library [5] to obtain the area. *Consec* variant resulted in the smallest implementation with $\approx 750\,\mu m^2$ as it does not have the complex byte selection sequence generation requirement like the *AESshift* variant. Adding the necessary sequence generation and extra control circuitry results in an area $\approx 850\,\mu m^2$ for the *AESshift* variant. Furthermore, the core circuitry involving the LFSR, tap selection and Hamming Distance calculation only takes up $\approx 30\,\%$ of the total area, with the rest needed for control and data

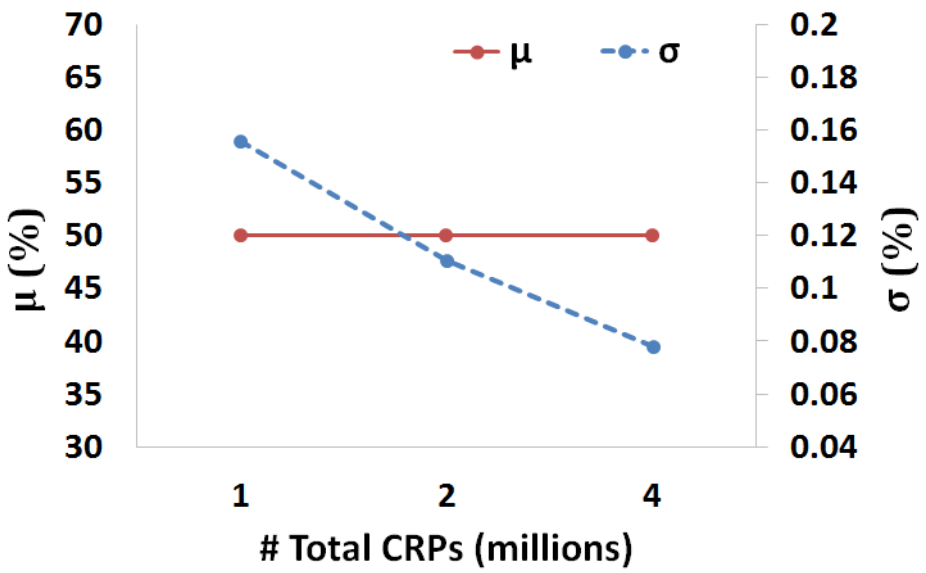


Figure 5.8: Learning Accuracy metrics for Neural Network Attack on 64-bit *Consec_DynTap* PUF design variant

movement/storage. Hence, these areas represent the overhead needed to implement a Strong PUF given a set of stable Weak PUF bits.

For comparison with alternate key generation schemes, a keyed hash, such as Keccak [48], would require 2280 gate-equivalents or $\approx 1900\mu m^2$. Additionally, the area overhead for the 64-bit entropy source variant of the weightless neural network (WNN) based PUF design by Santiago et al.[109] is quoted as $\approx 2060\,\mu m^2$. Hence, we see that our system is comparably *lightweight* and can be easily implemented in resource-constrained devices.

The quoted area numbers do not include the area needed to generate the initial stable Weak PUF bits, as this area is dependent on the mechanisms implemented to increase PUF reliability. However, as an example, the lowest area for a robust 128-bit Weak PUF key quoted by Patil et al.is $\approx 824\,\mu m^2$ [96]. For such a system, our design overheads would be $\approx 950\,\mu m^2$ and $\approx 1050\,\mu m^2$ for *Consec* and *AESshift* variants, respectively.

5.4.6 Latency

Once the input challenge is available, it can be divided into 8 substrings for processing. For each substring, we need 255 iterations to fully cover all possible states of the maximal-length LFSR and find the index with the minimum HD, as described in Section 5.3.1. We utilize an additional cycle for selecting the next pair of challenge input/PUF substrings to process and to perform the tap selection. In this cycle, we also write the obtained 8-bit index output back to the correct location in the input register for use in later rounds as we reuse the input register. Hence, we need $8 * 256 = 2048$ cycles to process one round. From our results, we see that we need to perform a minimum of 2 rounds and so, the latency for obtaining a response from our system is 4096 cycles. From performing a timing analysis on our implementations in 45 nm Nangate open cell library [5], we find that the fastest clock we can use is 1 GHz. Including the time to shift in the 64-bit challenge and shift out the 64-bit response serially, we can obtain a response every $\approx 4.25\mu s$. Using more rounds increase the time linearly. This translates to a throughput of ≈ 15 Mbps.

It is possible to reduce the latency further by utilizing two 8-bit LFSRs where each LFSR processed a set of 4 substrings in a round. This reduces the number of cycles/round by half. However, we note that we trade latency for greater area overhead from the LFSRs and additional control circuitry required.

For the 32-bit input challenge PUF design with a multi-word entropy source, it is possible to output the responses of each round (except the first) for the case of $k > 1$ as the entropy source is different in each round. We need $4 * 256 = 1024$ cycles to process one round. Including the time required for serially shifting in the 32-bit challenge and shifting out a 32-bit response from each round, we get a throughput of ≈ 26 Mbps. For generating larger keys, we need to wait to record outputs from k rounds to obtain the final key. However, if we transmit the output of each round (except first) as a partial key, we can maintain the higher throughput.

5.5 Conclusion

Strong PUFs have shown potential for providing a low cost alternative to cryptography-based authentication for application in resource-constrained devices. However, unreliability in Strong PUFs is of great concern as this can increase the number of CRPs needed for proper authentication, especially in the presence of millions of devices. Also, low modeling attack resistance can pose a significant security risk due to the large attack surface in applications such as IoT systems. Hence, it is desirable to explore Strong PUF architectures that provide a high level of robustness by leveraging the extensively researched reliability enhancements for Weak PUFs. In this work, we propose a Strong PUF system that utilizes an initial, robust Weak PUF entropy source along with a pattern-matching scheme based on Hamming distance comparisons between Weak PUF bits and the states of a linear-feedback shift register (LFSR). We also explore variants of the basic design to increase security and achieve close to ideal metrics in terms of uniqueness and modeling attack resistance. Our designs are shown to be lightweight and hence, can be incorporated into a large array of devices with varying resource constraints.

CHAPTER 6

META-OBFUSCATION OF PHYSICAL LAYOUTS

6.1 Introduction

One area of hardware intellectual property (IP) security research, to combat IC counterfeiting, that has received much attention is *hardware obfuscation*. The goal of obfuscation is to hide the true functionality of a design. Techniques have been proposed at various levels of design abstraction and can vary from introduction of additional gates to lock a circuit [106] to system level techniques [31]. Obfuscation can also involve the creation of camouflaged cells [33, 101] whose function can be hard to determine.

However, most of the design is driven by automatic place and route tools whose algorithms can result in structural information about a circuit that can aid a reverse engineer in forming a hypothesis about the function of the circuit. For example, flip flops and latches can easily be identified by following clock signals and SRAM is apparent by its regular structure. Hierarchy in chip integration can also reveal information. Hard IP is typically confined to lower metal layers, and signals in upper metal layers can be identified as inter-block signals. Buses or on-chip networks can be relatively easy to identify as wide collections of wires running together between a common set of agents.

So along with standard obfuscation certain additional processing is required to reduce such information leakage. With this as motivation, we propose meta-obfuscation techniques. These are *complimentary* to standard obfuscation and aim to prevent leakage of visual information to an attacker. The purpose of these techniques is to

break any identifying features of a circuit and allow for the creation of a generic structure. Along with the obfuscation mechanisms already employed this will enhance the security of the circuit against reverse engineering.

The salient contributions of this work are:

- We introduce a taxonomy of various visual information leakage avenues for circuits based on standard cell design.
- We discuss custom techniques, termed *meta-obfuscation*, for reducing visual information leakage.
- We explore iterative design algorithms based on the custom techniques to improve the quality of meta-obfuscation.

In this chapter, we concentrate on studying the feasibility of meta-obfuscation using a benchmark circuit as an example and do not explore the performance impact. More complex techniques that do take timing efficiency and additional overheads into account will be explored in future works.

6.2 Background

In this section, we will discuss research into methods for securing design IP in a supply chain using split manufacturing. Later, hardware obfuscation techniques that have been proposed in previous literature are discussed. We will explore hardware obfuscation at various levels of design abstraction. Lastly, we explore relevant works on reverse engineering of ICs.

6.2.1 Split Manufacturing

Karri *et al.* explore the security offered by split manufacturing and list the challenges that accompany such an approach [102]. Split manufacturing requires that the wafers be transported from the FEOL foundry to the BEOL foundry. These wafers

are thin and can crack or delaminate during transportation. Also, BEOL foundry will face alignment issues that need to be addressed properly. It is also shown that the well known heuristics used in typical floorplanning, placement and routing tools can be used by the attacker to predict the missing BEOL connections.

6.2.2 Hardware Obfuscation

Extensive surveys on protecting hardware intellectual property (IP) have been performed. A robust understanding of various aspects of hardware security and trust can be obtained via the work by Tehranipoor and Wang [118]. Colombier and Bossuet [35] provide an exhaustive survey of previous research into hardware protection. Guin et al. [52] perform a detailed study of counterfeiting and appropriate security measures including obfuscation.

6.2.2.1 System-level obfuscation

Many reverse engineering techniques engage in *component recognition*. Techniques have been proposed to increase the complexity of delineating components through various techniques such as combining two modules or replacing entire circuit with obfuscated equivalent [87, 88, 95].

Alkabani et al. [18, 19] have proposed using FSM modification to lock a chip by obfuscating its power-up state and only the correct key unlocks the IC. Chakraborty and Bhunia [31] propose a methodology to perform simultaneous obfuscation and authentication of an SoC design netlist. Rajendran *et al.* [100] propose a solution to secure against untrusted supply chain and trojan insertion by malicious insider at a fabless company. The various methods of securing the design include instruction set randomization and additional security modules in the processor pipeline. A related application is logic encryption, where additional gates are added to the design such that only the correct input values allow the circuit to work properly [69, 106].

Hardware-Software co-design based security: The work by Schrittwieser *et al.* focuses on protecting software vendor from piracy by the creation of a strong hardware-software binding [113]. Zheng *et al.* propose a scheme where they utilize PUFs for instruction obfuscation [133]. In their scheme, each instruction is stored in memory in two parts: the obfuscated instruction and a challenge word to the PUF device. The PUF responses produce the actual opcode to decode the instruction within the instruction pipeline. Our work introduces more flexibility into the use of PUF responses in the instruction pipeline and describes additional security considerations.

6.2.2.2 Circuit-level obfuscation

Circuit-level obfuscation methods create or modify cell libraries to hide gate functions and adding non-essential structures to the design. These approaches add extra complexity to reverse-engineering the logic.

Camouflaging of cells can be achieved through custom design to either mimic other cells or to allow for post-manufacturing programmability to customize gate functions. Certain companies have specialized in creating camouflaged cell libraries [117]. Rajendran et al.[101] have studied how to efficiently deploy a small number of camouflaged cells to maximize the hardness of reverse engineering at minimal overhead. Filler cells may also be used for the purpose of obfuscation [32, 33]. Some of the cells can connect to functional logic without hampering operation. This leads to an increase in the amount of data that an attacker will need to sort through during reverse engineering in order to extract the underlying logic.

6.2.3 Reverse Engineering

Numerous reverse engineering attacks have been showcased over the last few years [66, 91, 114]. Moreover, support for reverse engineering by CAD tools such as Chipworks' ICWorks and the open-source tool Degate [112] has resulted in attackers being able to steal IP with relative ease.

6.3 Taxonomy of Visual Information Leakage

In this section, we will classify the various avenues for visual information leakage, with relevant examples, that aid an attacker in reverse engineering the design. Later, we will concentrate on the specific issues in the presence of sequential circuits.

6.3.1 Standard cell types and sizes

Fabless designers depend on using commercially available cell libraries, often from the foundries like TSMC where the designs will be manufactured. A vast library of standard cells with different types and sizes allow a design to be optimized for performance, power and area. However, these libraries are also available to attackers to assist with their reverse engineering efforts. In case a designer chooses to use custom obfuscated cells, these can stand out allowing attacker to focus more on the locations of these cells.

The types and sizes of cells used allow an attacker to figure out the nature of a circuit [91]. For example, a 8×8 multiplier instantiated with a full Nangate 45nm library [5] has recognizable full adder cells, as highlighted (in green) in Figure 6.1. The number of such cells also gives an attacker clues to this circuit being a multiplier of a particular size.

6.3.2 Size of the module

An attacker can distinguish the boundaries of a design module through various methods like observing the interconnects in higher metal layers to identify connections between modules; the power grid, where some modules are surrounded by ring structures; and whitespace between modules. Different design modules can have distinct sizes. This, along with other visual information, allows the attacker to formulate hypotheses on the nature of a particular large module.

For example, comparing the sizes of an 8×8 adder ($37.2\mu m^2$) and a 8×8 multiplier ($379.5\mu m^2$) instantiated using Nangate 45nm cell library [5], we see that the multiplier

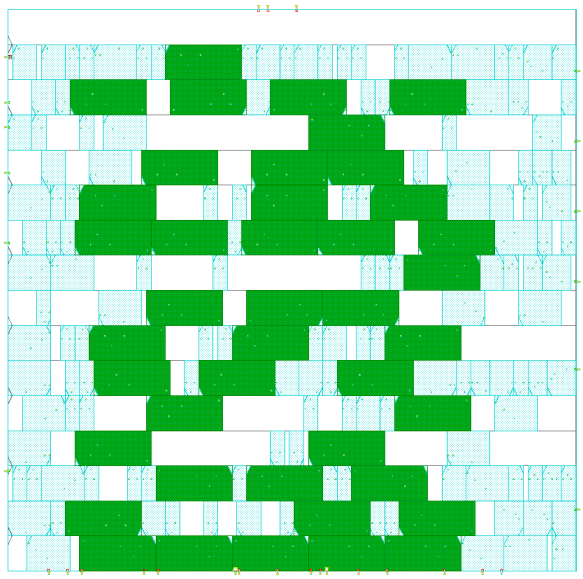


Figure 6.1: Floorplan of 8×8 multiplier with highlighted full adders (green)

is $\sim 10\times$ larger. This size variance can be used by an attacker to clearly recognize a multiplier. The attacker can then focus on the particular module to infer its function.

6.3.3 Structural information

Certain designs can leak information due to the observable structure of placed components. This includes location of certain large cells, identifiable repeating unit blocks (SRAMs), placement of pins and so on. Also, placement algorithms used in commercial tools can become a crucial method for extracting information about a design. An attacker can separate the design into sub-blocks and focus effort on understanding the function of each sub-block instead of trying to tackle the whole design at once.

6.3.4 Routing and metal density

Automated reverse engineering tools allow an attacker to retrieve the routing and metal density information of a design with relative ease [112]. The use of metal layers in a design is quite well-defined, for example the topmost metal layers used for routing

power lines. Even if the design was affected to hide placement information, routing can be used by an attacker to identify clusters and logical neighbors in a design and delineate the circuit into sub-blocks. Hence, it becomes crucial to address this for the protection of a particular design.

6.3.5 Leakage in sequential circuits

6.3.5.1 Clock Paths

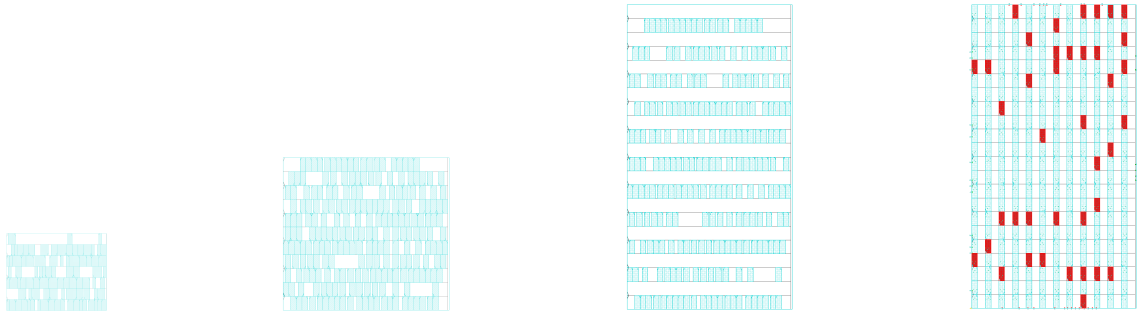
Typically, clock is distributed in a design via H-trees [42]. Tracing these clock paths is one of the primary goals of an attacker as these readily identify the flip-flops, located at the leaf nodes of the tree, in a design. Using this information, the attacker will be able to divide the design into manageable parts for analysis. The attacker can then apply SAT techniques [62] along with knowledge of other visual information discussed in this section to identify the functions implemented between flip-flops.

6.3.5.2 Design for Test logic

Tracing of scan chains provides another avenue for an attacker to identify flip-flops. Since the flip-flops on a scan chain may be accessible through the scan pins of an IC, the attacker can use this knowledge to aid in the deciphering of a circuit's functionality.

6.4 Meta-obfuscation techniques

In this section, we discuss the potential solutions to address the problems discussed in the previous section. Concentrating on a single circuit (ISCAS C432 [53]), we illustrate the solutions to highlight the changes made for each case. We explore how these techniques will compliment existing obfuscation mechanisms in confusing the attacker.



(a) C432 with full cell library

(b) C432 with only universal gates

(c) C432 expanded in y-direciton

(d) C432 with dummy cells (red)

Figure 6.2: Meta-obfuscation techniques applied to ISCAS C432 [53]

6.4.1 Standard cell types and sizes

One possible way to complement obfuscation techniques is to affect the standard cells used in the design. It is vital to restrict the circuit to use only universal gates (NAND, NOR and Inverter) as much as possible to make sure that the standard cells themselves do not reveal any information. Also, constraining their sizes allows us to visually make the cells look the same and forces an increase in reverse engineering effort to distinguish the cells.

For the purpose of illustration, we instantiate the ISCAS C432 using the full Nangate 45nm standard cell library [5] using Synopsys Design Compiler and carried out placement in Cadence Encounter [1], shown in Figure 6.2a. Next, we re-compiled the design and restricted the standard cells to universal gates and of comparable sizes (especially for the inverter). The design placement was carried out again, as shown in Figure 6.2b.

However, just the use of universal gates may not be enough to protect the design. We still need to address the presence of structural information. Also, in case a designer needs to utilize more cell types, we need to address this through the usage of dummy cells. These issues will be dealt with in detail below.

6.4.2 Size of the module

Certain obfuscation techniques allow us to affect the size of a module by combining it with other neighboring modules [95]. Most other techniques can also incur an increase the design area. Scaling the design further may prove beneficial as it allows us the flexibility to institute further changes. Additionally, depending on the nature of scaling we can affect the module's shape. This can prove useful by making the current design look similar to its larger neighbors. Affecting the size of the module incurs the most expense in terms of design resources.

For our purpose, we expanded C432 by a 100% in the y-direction changing its shape. We need to account for the standard cell height during expansion for proper placement of cells by the tools. The new module, with the cell locations also scaled, is shown in Figure 6.2c.

While affecting the size of the module may prove useful, we still need to account for the placement tool behavior which may introduce certain structural information that can still aid the attacker.

6.4.3 Structural information

To account for the issues discussed in section 6.3.3, we systematically modify a design placement to make it look more generic. This can involve the use of dummy cells to introduce a uniformity to the cell placement. Also, the addition of dummy cells gives us the option of utilizing more of the available standard cells in a cell library than just the universal gates.

For C432, while scaling the module in the y-direction changed its size, the cells were placed in alternate rows, as shown in Figure 6.2c. To fill up all the rows, we shifted every other cell in each row to the next empty row. We also introduce dummy cells (NAND gates) with the express purpose of equalizing the number of cells in each row. Lastly, we spaced the cells in each row equally resulting in a uniform layout,

as shown in Figure 6.2d. It should be noted that we are not concerned about the logical proximity of the cells in a design as further steps will target such proximity to enhance meta-obfuscation achieved.

Such a layout effectively breaks up clustering of cells reducing the attacker's ability to form hypotheses as to the nature of such clusters in a larger design. A point to note is that if the designer does choose to incorporate more cells types, including custom obfuscated gates, then it becomes imperative to utilize more dummy cells of the new types to shift focus away from the cells used for the circuit logic. This may incur a cost in terms of area. Also, hiding structural information becomes a more complex problem as we have to account for more cell sizes in the layout.

6.4.4 Routing and metal density

Intricately connected to structural regularity is the net routing and metal density information. To hide the leakage of cell proximity and module boundary information via the routing of a design, we need to affect the cell placement and its corresponding routing. The designer needs to be aware that the methods used to increase complexity in this step will affect the performance of the design, increase the metal resources needed, and adapt accordingly.

A possible solution is to change the cell positions in a design and increase the distance between logical neighbors. This will force the place and route tools to utilize more of the upper metal layers for the new longer connections. Also, the presence of dummy cells introduced previously affords us the option of controlling their input and output connections which will affect routing.

A metric is needed to provide a quantifiable measure of quality of meta-obfuscation for a design. Wire lengths in various metal layers and their density can be considered a good starting point. However, swapping of cells greatly affects the metal layers

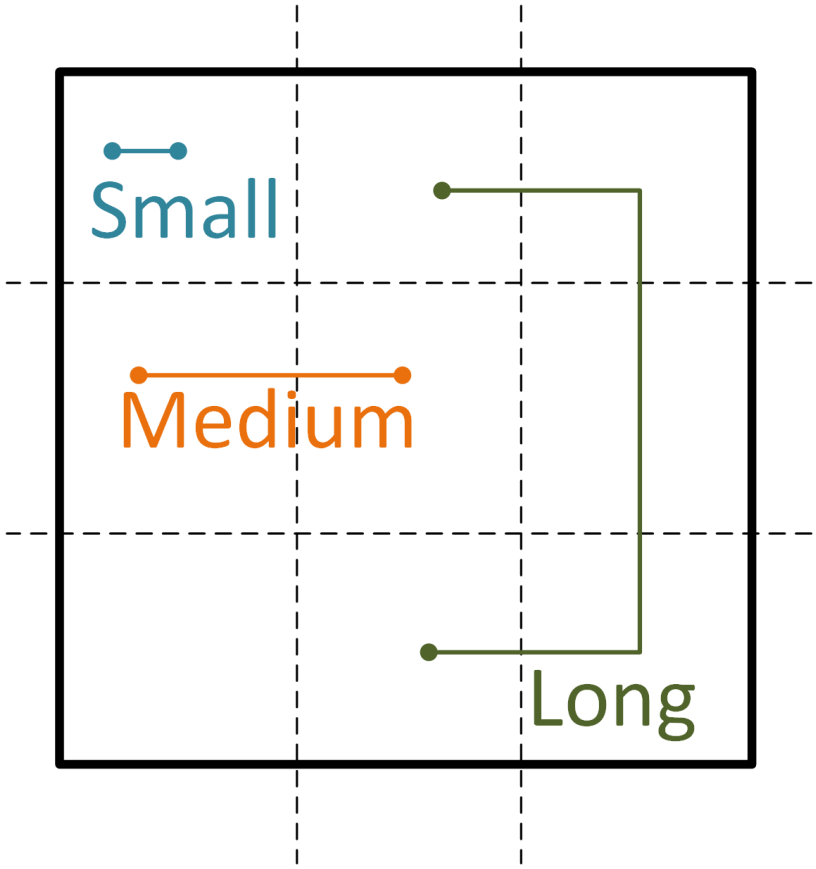


Figure 6.3: Representation for classification of nets as *small*, *medium* and *long*

used and the lengths of wires in each layer. Hence, for the purpose of driving an optimization algorithm we consider a different approach.

We divide the design into various grids and classify the nets, i.e. connection between cells irrespective of metal layers used, as either *small*, *medium* or *long* nets, as illustrated in Figure 6.3. For each grid, we find the number of these nets present in the grid or passing through it and classify them into the 3 categories. This results in a distribution of nets of varying lengths for each grid. Our goal is to increase the number of *long* and *medium* nets for each grid with greater emphasis on longer nets.

Section 6.5 will explore the use of the above metric for C432 circuit in greater detail and quantify the results obtained. The results will highlight the effectiveness of this approach.

6.4.5 Sequential circuits

One possible mechanism to hide the clock distribution network is to utilize clock grid instead of identifiable H-trees. Obfuscation techniques like secret vias [101] would allow us to hide the locations where the clock in connected to a design. Addition of dummy flip-flops also can enhance security by confusing attackers while not affecting the primary circuit. This will be explored in greater detail in the future.

6.5 Methodology for meta-obfuscation

In this section, we detail the various methods considered to improve the meta-obfuscation of a circuit. We first start with a general description of the proposed method and then, objective function used to drive the optimization is detailed. Then, we focus on the various models used for obtaining a solution. As with section 6.4, we illustrate the results with ISCAS C432 circuit.

6.5.1 Proposed Method

We randomly chose one of the inputs to each dummy cell from the primary and intermediate nets present in the design and the output of a previously processed dummy cell as the other input. The first dummy cell's inputs were assigned using the primary inputs and the last dummy's output was turned into a dummy primary output. This increases the number of pins for the design but, does not affect any of the valid logic. The relevant files were modified accordingly.

We take the design floorplan from Figure 6.2d and output a Design Exchange Format (DEF) file. This is modified by the various algorithms discussed below which select the cells to swap and how many swaps to perform. The new intermediate DEF file is then forwarded to Encounter for full routing. The final routed design is again output in DEF format and is processed.

The designer can choose the number of grids for analysis and their sizes. For our purpose, we chose 3×3 grids of equal size. Grid $(0, 0)$ contained the floorplan origin and $(2, 2)$ was the farthest grid. The nets were classified as explained in section 6.4.4. The distribution of nets obtained was then used to generate the *score*, explained below, which drove the algorithms described subsequently.

All approaches used were only limited by the routing time of Cadence Encounter tool [1] and not the methods themselves.

6.5.2 Objective function

In section 6.4.4, we chose to divide the design into grids and classify nets into *small*, *medium* and *long* (Figure 6.3). Using the distribution of such nets in each grid, we can derive an objective function for use in optimization. The initial distribution of *medium* and *long* nets for each grid for C432 is shown in Figure 6.5a. Our target is to increase the amount of *long* and *medium* nets designated to each grid. It is entirely possible that an algorithm may end up increasing the number of *long* nets for a particular grid while reducing it for another if we just consider maximizing the mean for all *long* (μ_{long}) net values across all grids. So, we need to consider the standard deviation across the grids (σ_{long}) and seek to reduce it. To achieve both an increase for mean and smallest standard deviation we express the objective as a fraction as shown in (6.1). Similar objective is defined for medium nets.

$$Obj_{long} = \frac{\mu_{long}}{\sigma_{long}} \tag{6.1}$$

The final *score*, as given by (6.2), is the sum of the two objectives with *long* nets given 70% weight and 30% weight for the *medium* nets. These values can be changed depending on the design requirements. Longer nets may degrade performance and

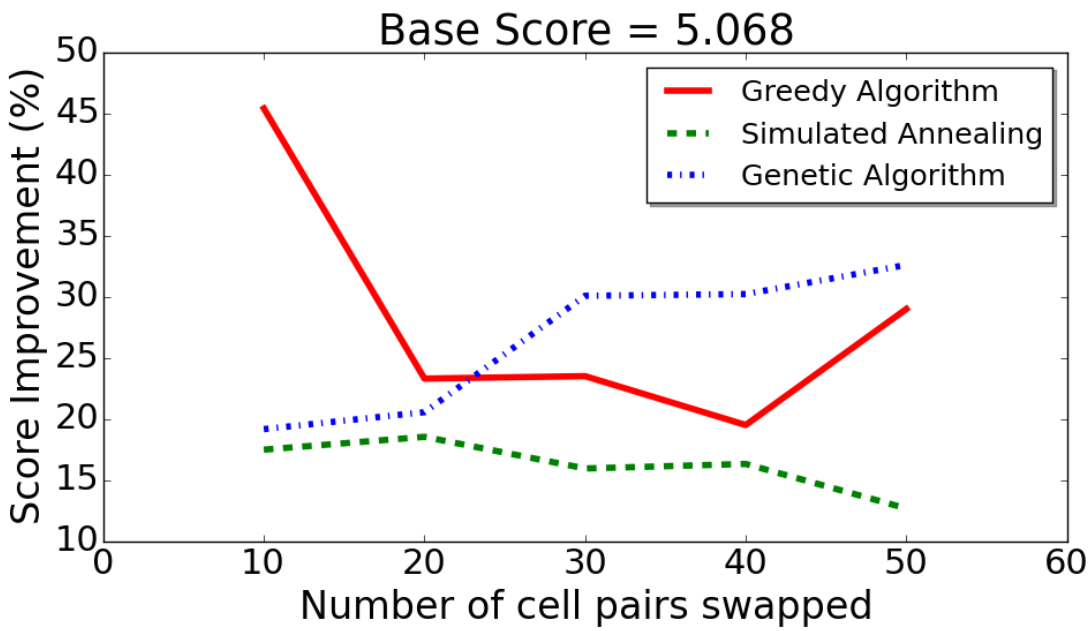


Figure 6.4: Comparison of performance for Greedy, Simulated Annealing and Genetic algorithms for varying number of swaps

hence, the designer may choose to give *medium* nets greater priority. The initial design is processed and its score is used as a starting point.

$$Score = 0.7 * Obj_{long} + 0.3 * Obj_{medium} \tag{6.2}$$

6.5.3 Greedy algorithm

First, we tried a greedy approach by randomly swapping pairs of cells and checking the score. The swaps are only accepted if the new score is greater than the previous maximum. An iteration limit is set to terminate the program and return the final obtained score.

We set the program limit to 20 iterations and varied the number of swaps performed between 10 $\rightarrow$ 50. For each case, the experiment was conducted multiple times and the best improvement obtained over base score was plotted, as shown in

Figure 6.4. Due to the random nature of the swaps, greedy algorithm is not able to provide a consistent results. There is no guarantee that increasing the number of swaps will have any beneficial effect on the solution. Hence, we next try simulated annealing.

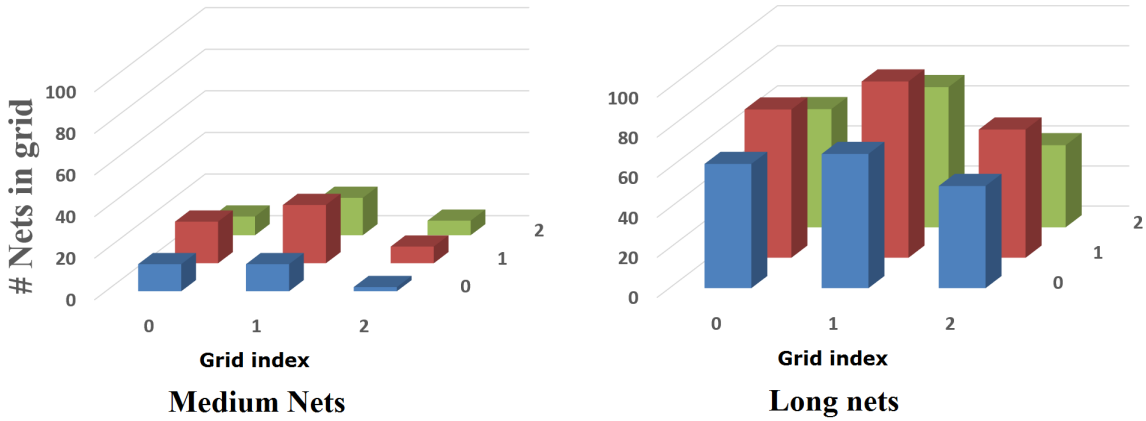
6.5.4 Simulated Annealing

Next, simulated annealing [11] was utilized to study its performance. Initially, bad results were accepted with a probability of 0.5 reducing acceptance probability (to 0) as the simulation iteration limit was reached. A quartic function was used to calculate the probability. Similar to the greedy approach, we plot the results, in Figure 6.4, for swaps varying between $10 \rightarrow 50$ with a 20 iterations limit. We see that simulated annealing performed more consistently than greedy algorithm. However, we observed that increasing the iteration limit made the acceptance probability curve decline more steeply and negatively affected the results.

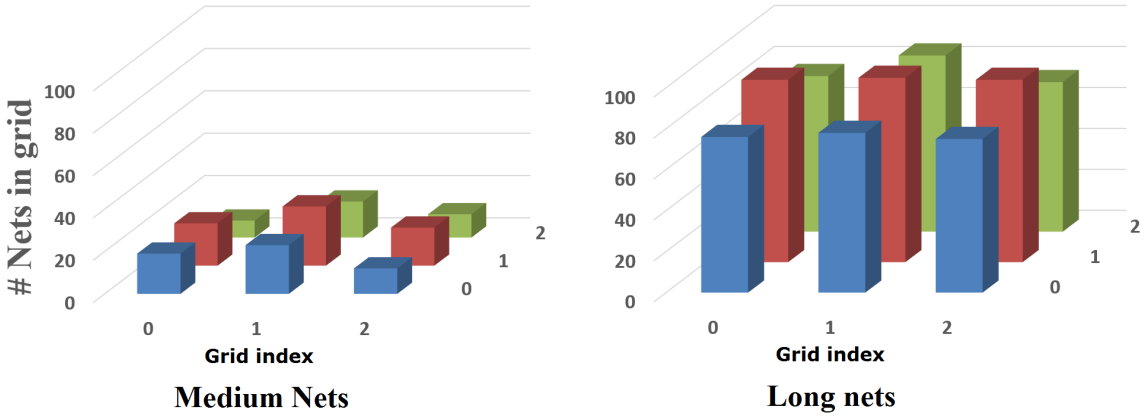
6.5.5 Genetic Algorithm

We consider the use of genetic algorithms (GA), a subset of evolutionary algorithms (EA). They imitate natural evolution utilizing analogous concepts like mutation, reproduction, crossover and selection. The genetic algorithm produces a population of candidates for whom a score/fitness is calculated and children are selected for the next generation based on some internal heuristics. We make use of the open-source toolkit, Pyevolve [8] to implement the genetic algorithm.

Initially, we set a population size of 20 and observe the best candidate for a single generation while varying the number of swaps between $10 \rightarrow 50$, similar to the previous approaches. The objective function that calculates the fitness is the same as before, given by (6.2). The results for *medium* and *long* nets, from Figure 6.4 show that, GA performs quite well and gives significant improvement over greedy and simulated annealing approaches.



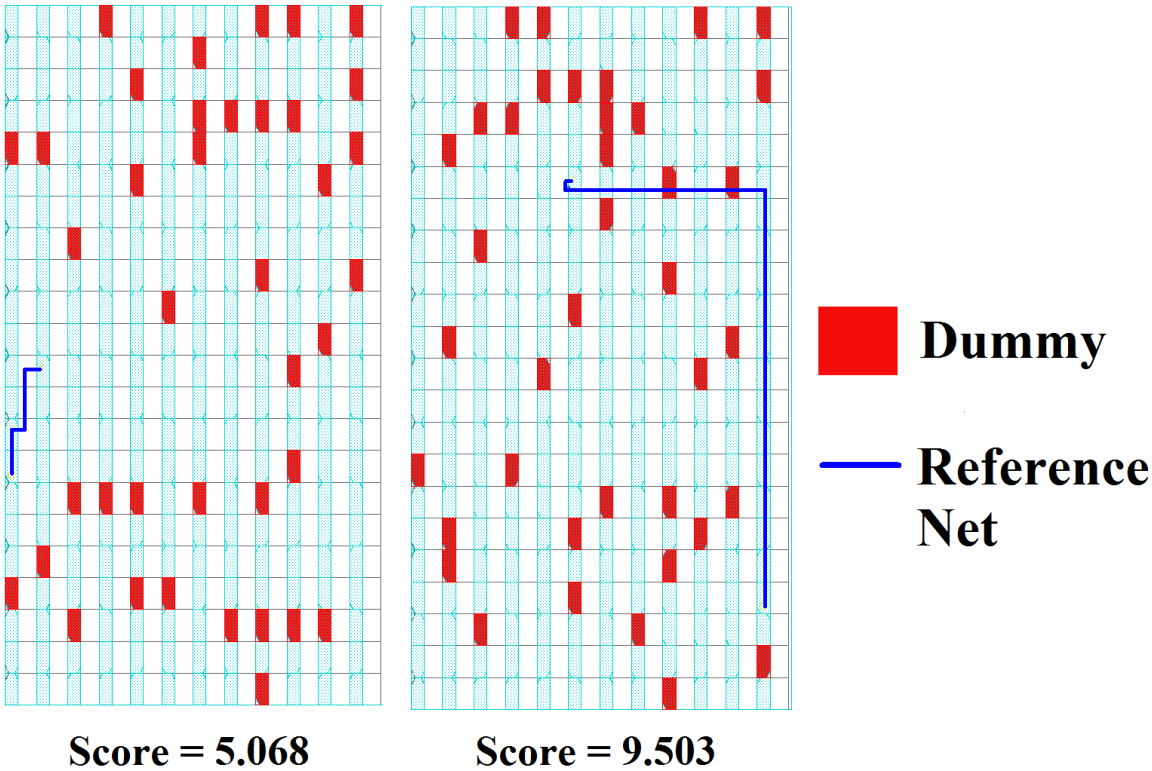
(a) Grid Metrics before optimization



(b) Grid Metrics after optimization

Figure 6.5: Genetic Algorithm Results ISCAS C432 [53]

We reset the population size to the default, 80 in Pyevolve, and increased the number of generations to 10. This gave us an improvement of 87.5% in the metric. We plot the number of *small*, *medium* and *long* nets in each grid of the design before and after applying the genetic algorithm in Figure 6.5. The results show an increase in the *medium* and *long* nets while also keeping the variation across the grids low. The initial and final floorplans with dummies (red) and select net (blue) highlighted for comparison is shown in Figure 6.6. This shows that some shorter nets become long during optimization and improve the score. Also, the metal lines in metal layer $M3$ have been plotted in Figure 6.7 before and after optimization. We see that the metal lines are more spread out after processing compared to the original.



(a) Floorplan before optimization

(b) Floorplan after optimization

Figure 6.6: Genetic Algorithm Results for ISCAS C432 [53]

(a) M3 before optimization (b) M3 after optimization

Figure 6.7: Comparison of metal lines in M3 layer for ISCAS C432 [53]

6.6 Conclusion

Obfuscation techniques have been proposed to counter reverse engineering with an intent to steal IP from legal parties by malicious actors. However, the physical layout of a design can still leak visual information. In this work, we explore the various ways in which such information can be leaked and provide mechanisms, termed as *meta-obfuscation*, to address them. We also discuss a metric that can be used to quantify the quality of meta-obfuscation. A methodology to maximize the metric is also discussed. The results show that a significant improvement can be achieved via the use of evolutionary algorithms.

CHAPTER 7

ON LEVERAGING MULTI-THRESHOLD FINFETS FOR DESIGN OBFUSCATION

7.1 Introduction

Robust hardware obfuscation, on a circuit-level, can be achieved by leveraging the intrinsic characteristics, such as threshold voltage, of transistors instead of just relying on physical structures for creating camouflaged gates. Such characteristics are not susceptible to imaging techniques and analyzing the chemical composition of individual transistors in advanced nodes is exceedingly difficult as the characteristic is determined by just a few hundred doping atoms. Specifically, we leverage the large number of threshold voltages supported by FinFETs, which have been conventionally targeted towards enabling increased optimization for power and performance [2]. For example, commercial FinFET process development kits (PDKs) from TSMC offer four transistor threshold options in 16 nm technology node [12]. In contrast, MOSFETs could support one or two threshold voltages. Furthermore, FinFETs permit larger transistor stacking height in a cell design [3] and hence, can support greater number of inputs and hence, implement more functions efficiently compared to MOSFETs.

In this chapter, we propose a simple n-input gate design that utilizes 4 threshold voltage transistor options available from a commercial vendor's 16 nm process development kit (PDK). We analyze 2-, 3- and 4- input variants of the design and show that it is possible to implement a large number of functions using the same structure and different threshold voltage assignments. Our designs are shown to be stable across different process, voltage and temperature (PVT) ranges. Further, we

explore SAT-based attacks [77] that can reverse engineer the functionality of a circuit by observing the outputs during operation and possessing basic layout details, such as the number of gates and interconnection information. Our designs are shown to increase the effort required to de-obfuscate the circuit by an order of magnitude compared to designs synthesized with commercial cell libraries for ISCAS-85 benchmarks [53]. Lastly, we explore certain special functions whose outputs are only dependent on the state of a subset of the inputs and study the effect of their incorporation on de-obfuscation.

7.2 Background

Previously, we briefly explored obfuscation techniques that have been proposed in previous research at various levels of design abstraction in Section 6.2. In this section, we further explore previous research on circuit-level obfuscation techniques and also, discuss works on reverse engineering obfuscated designs, with focus on SAT-based attacks.

7.2.1 Hardware Obfuscation

Circuit-level obfuscation techniques involve creation/modification of cell designs with the goal of hiding the true functionality of the camouflaged gate. Rajendran et al.[101] have proposed a cell design that can implement a variety of Boolean functions by utilizing programmable dummy contacts. However, their design realizes a limited number of functions while incurring large area and power overheads ($4-5\times$).

While previous techniques require layout modifications to achieve their goal, it is possible to use the intrinsic properties of transistors to realize hardware obfuscation. Transistor doping-based camouflaging was explored by Malik et al.[84] to create an obfuscated cell. Multiple such cells are combined to obtain obfuscated gates. Threshold voltage-based camouflaging techniques include leveraging pass transistors [34, 60]

or employing sense amplifier-based (SABL) logic and changing the threshold of transistors post-fabrication [15]. These techniques also do not realize a large array of functions with SABL logic-based technique incurring even greater overheads ($> 6\times$).

7.2.2 Reverse Engineering Attacks

Extensive surveys of state-of-the-art hardware reverse engineering techniques at varying levels of attacker capabilities have been conducted [43, 99]. In this work, we assume that the attacker is capable of extracting most layout information and so, we focus on SAT-based attacks.

Many of the previously proposed obfuscation techniques are susceptible to SAT solver-based attacks where the attacker is capable of querying the IP and obtaining its outputs for a set of given inputs. The camouflaged gates can be represented as abstractions, such as a MUX or switch network with programming inputs, and the SAT algorithms utilized to guess the function of camouflaged gates by finding the programming inputs [131, 132]. In this work, we utilize the SAT-based de-obfuscation scheme proposed by Yu et al.[132] as it is shown to perform significantly better compared to other SAT approaches.

7.3 Multi-threshold FinFET Camouflaged Cell

In this section, we first describe the basic design of the FinFET camouflaged cell and analyze the performance of the camouflaged cells under environmental variations. Next, we explore the requirements to realize an exclusive-or function with our design. Lastly, we create a custom cell library and synthesize various ISCAS-85 benchmark circuits and compare them against the same circuits synthesized using a commercial cell library.

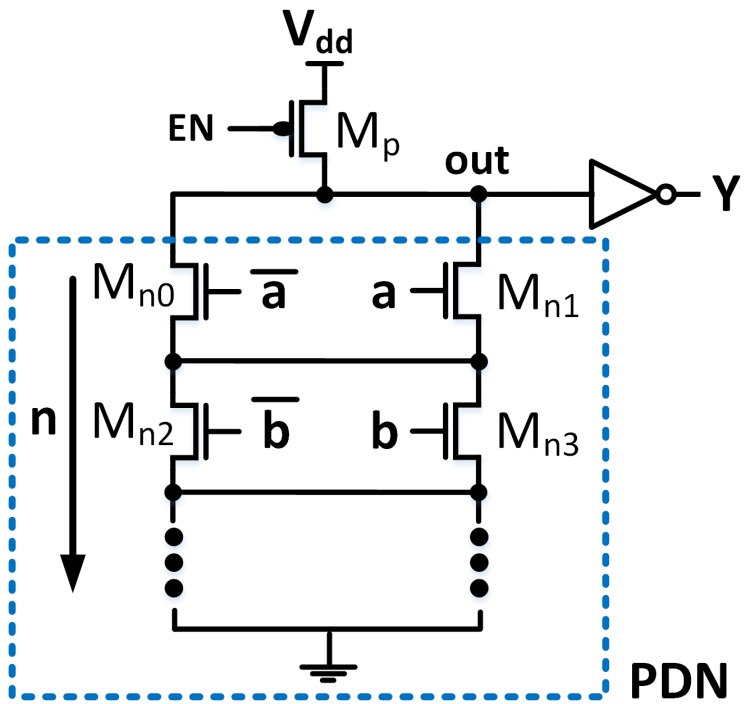


Figure 7.1: General structure of Multi-Vt FinFET camouflaged cell

7.3.1 Basic Cell Design

Our design goal was to be able to realize as many functions as possible for an n-input gate without changing its physical structure. Ideally, 2^{2^n} logic functions are possible with an n-input gate design. Additionally, we sought a scalable cell design that could implement an $n+1$-input gate with simple, limited modification to an n-input gate. Towards achieving these ends, we propose the cell design as shown in Figure 7.1.

The n-type FinFET pull-down network (PDN), in Figure 7.1, consists of transistors (M_{nx}) comprising two stacks of height n. Therefore, the total number of transistors in the PDN is $2n$. For example, a 2-input cell will have 4 transistors ($\{M_{n0} \ldots M_{n3}\}$) in the PDN with M_{n2} and M_{n3} connected to ground. The source/drain nodes of any two transistors at the same level, like $\{M_{n0}, M_{n1}\}$ or $\{M_{n2}, M_{n3}\}$, in each stack are shorted. The inputs to one stack are the original gate inputs ($\{a, b, \ldots\}$) while the other stack receives the complimented inputs ($\{\overline{a}, \overline{b}, \ldots\}$).

As can be seen from Figure 7.1, such a ladder structure allows all *possible* paths between the output node, *out*, and the ground for various input combinations. The pull-up network consists of a p-type FinFET, M_p, connected to an enable input, *EN*. An inverter at the output of the PDN is used to restore the signal voltage levels and generate the final gate output, *Y*.

When *EN* is *logic-1*, the circuit is gated and saves power. Switching *EN* to *logic-0* allows the current to flow into the PDN network. Based on the threshold voltage (V_t) parameters of the transistors, certain paths to the ground will be dominant. The PDN circuit acts as a voltage-controlled resistive divider. Combined with the input signal combinations to the cell, we can realize different functions. To implement various logic functions, we permutate through the available threshold voltage options for the PDN transistors and all possible 2^n input combinations and perform SPICE simulations to generate the truth tables for each set of transistor permutations. The transistors sizes, in terms of the number of fins ($nFin$), are chosen accordingly. We note that multiple sets of PDN transistor V_t assignments may implement the same logic function. This provides us flexibility in selecting the most stable assignment after analyzing the effect of environmental variations.

In our work, we consider a commercial 16 nm PDK that supports 4 threshold voltage options, having a nominal voltage of 0.8 V and analyze 2-, 3- and 4-input gate designs. The available threshold voltage options will be designated as *ulvt*, *lvt*, *svt* (standard) and *hvt* in increasing order of threshold voltage, respectively. For an n-input cell, the total number of possible multi-threshold assignments and hence, the number of SPICE circuits analyzed is 4^{2n}. This gives us 256, 4096 and 65536 distinct circuits to analyze for 2-, 3- and 4-input gates, respectively. The p-type FinFET, M_p, has width, $nFin = 2$. For the 2-input and 3-input cells, the PDN transistors have a width of $nFin = 3$, while the 4-input PDN transistors are sized $nFin = 4$. In the physical layout, we note that each transistor with a different threshold voltage

Table 7.1: Number of Stable Logic Functions for Camouflaged gate with varying fan-in

Fan-in (n)	**# Unique Logic Combinations**	
	Possible	**Realized**
2	16	12
3	256	96
4	65536	1096

requires a minimum area, while two transistors of the same threshold voltage type could share the same area. For our gate area calculations, we assume each transistor assignment is different to get the upper bound.

We note that a second inverter can be used to generate the compliment, $\overline{Y}$, using the input Y. This allows us to generate both the necessary input signals needed by any subsequent fan-out gate for the current cell's output. An added advantage is the reduction in the total cell size. For example, a 4-input camouflaged cell would need to include 4 inverters to generate the complementary signals for all its inputs if the previous fan-in camouflaged cells did not generate a pair of signals, adding significant area overhead. Only the primary inputs (PIs) to the netlist would need an explicit set of inverters to generate their complements. However, this approach does increase the routing density for the entire design.

7.3.1.1 Environmental variations

Given the usage of transistors with differing threshold voltages, it is critical to ensure that the particular transistor permutation selected to implement a function is stable in the presence of supply voltage noise and temperature variations. Due to the undoped channel in a FinFET, its threshold voltage is not susceptible to Random Dopant Fluctuations (RDF) like conventional MOSFETs. However, irregularities in

the *fin* surface introduced due to manufacturing process variation can have an impact in the form of metal-gate work function variation (WFV) [86]. Hence, we need to consider the effect of process variations while making the final selection of threshold voltage assignments.

First, we seek to find the maximum number of unique logic functions that can be realized in the presence of supply voltage variations. Towards this end, we analyze each design with a specific set of PDN transistor V_t assignments for supply voltage variations of $nominal \pm 10\%$. The nominal voltage is set at 0.8 V for the commercial 16 nm PDK used in this work. Synopsys HSPICE was used for all simulations. Table 7.1 lists the number of observed stable logic functions across all supply voltages and the total possible logic functions for 2-, 3- and 4-input camouflaged gate designs. Since a logic function may be realized with more than one set of PDN V_t assignments and some assignments may change the circuit behavior under supply voltage noise, we need to ensure that we record only the assignments that result in a stable output function.

Once we obtain a database of possible output functions for an n-input gate and the corresponding PDN V_t assignments, we run further simulations on these designs only. We analyzed their performance across a temperature range of 273 K to 373 K and applied the PDK vendor recommended process variations. For process variations, 1000 Monte Carlo simulations were performed for each design and we check which PDN assignments were able to maintain their functionality the most. This further whittles down the number of PDN V_t assignment sets for a given output function. Hence, we see that the total number of unique functions reduces significantly as n increases, as indicated in Table 7.1.

In Table 7.2, we compare the area, delay and power performance of 2-, 3- and 4-input *NAND* gates implemented using complimentary logic and our camouflaged design. For area and delay, our camouflaged version of the NAND gate see a maximum

Table 7.2: Comparison of Area, Delay and Power characteristics between Regular (Reg) and Camouflaged (Camo) NAND cells

	NAND2		**NAND3**		**NAND4**	
	Reg	**Camo**	**Reg**	**Camo**	**Reg**	**Camo**
Area (μm^2)	0.17	0.35	0.22	0.5	0.28	0.65
Delay (ps)	6.10	15.56	8.39	24.62	11.83	26.78
Power (μW)	294.7	69.55	391.77	69.30	482.98	69.31

overhead of 2.32× and 2.93×, respectively. However, with regards to power, we see a minimum reduction of 4.24×. Furthermore, all n-input camouflaged gates consumed similar amounts of power. This is due to the fact that our circuit does not see a full-rail voltage swing due to the fact that p-type FinFET, M_p, is active when the PDN evaluates the output. Hence, there is a reduction in the power consumed at the expense of reduction in the output noise margin at node *out*, with the minimum noise margin observed being 200 mV. However, the output inverter ensures that the voltages are restored at node Y.

7.3.1.2 Implementing XOR/XNOR with via manipulation

We note that our camouflaged cells are not able to realize XOR/XNOR logic functions for any n-input gate considered. This is due to the shorting of the internal nodes to create the ladder structure, as shown in Figure 7.1. For implementing a regular 2-input XOR/XNOR function, there needs to be an *exclusive* path to the ground only when both inputs are the same (XOR) or both are complementary (XNOR). This is not possible with our camouflaged structure. Furthermore, for gates with $n > 2$ inputs, the XOR/XNOR functions are even more complex and cannot be implemented.

One possible method to implement XNOR function for 2-input camouflaged cell by disconnecting the drain nodes M_{n2} and M_{n3} in the PDN shown in Figure 7.1 via manipulation techniques to introduce dummy via contacts in the physical layout to make it appear as if a connection exists [101, 127]. Furthermore, we can utilize dummy vias and create a layout that can implement the XOR function. We can route the nets from an input (say b) to both the transistors (M_{n2}, M_{n3}). We can add a legitimate via between the input and the desired gate (M_{n2}) and adding a dummy via for the other gate (M_{n3}). Repeating the same for the complementary input ($\bar{b}$) we can confuse an attacker and increase reverse-engineering effort. However, such a layout modification would need to be made for all the 2-input gate functions implemented to prevent differentiation between XOR and other gates. Hence, XNOR may be easier to implement in practical use. If such techniques are available, then we only need to use *svt* option for the PDN transistors $\{M_{n0} \ldots M_{n3}\}$ to achieve XOR/XNOR functionality.

7.3.2 Camouflaged Cell Library

We perused the stable logic functions for 2-, 3- and 4-input camouflaged gates and constructed a camouflaged cell library with a total of 62 logic gates. We included a basic inverter to allow proper design synthesis and assume that we have 2-input XOR/XNOR available, using modifications described in Section 7.3.1.2. We selected one cell design with a particular PDN V_t assignment set for each logic function. The selection process was driven with the aim of matching as many available functions as possible in a commercial 16 nm standard cell library. We note that our cell library is not exhaustive with respect to possible 3- and 4-input logic functions and other designers may choose to implement an even larger cell library, including more than one cell design for a logic function.

Table 7.3: Area estimates for ISCAS-85 Benchmark circuits

Circuit	**Commercial Cell Library**		**Camouflaged Cell Library**		**Area Over-head (%)**
	# Gates	**Area (μm^2)**	**# Gates**	**Area (μm^2)**	
c17	3	0.88	4	1.7	92.15
c432	88	20.68	78	37.2	79.88
c499	154	54.19	178	62.7	15.70
c1908	176	55.52	191	75.7	36.35
c2670	354	90.52	471	149.2	64.83
c3540	454	124.58	551	239.8	92.48
c6288	1492	356.60	1505	660.6	85.25
c7552	768	242.64	999	387.6	59.74

We compare our camouflaged cell library and the commercial cell library by synthesizing a set of ISCAS-85 [53] benchmark circuits using Synopsys Design Compiler. It should be noted that in practical scenarios, a designer may choose to camouflage only part of the design and hence, the area overhead will be much lower. Table 7.3 lists the number of gates in the synthesized netlist for each cell library and the area estimates. The area overhead depends on the logic functions utilized in a particular benchmark circuit. However, our camouflaged netlist is generally larger due to the basic 2- and 3-input cells being larger than the standard cell versions, while 4-input function cells were comparable in size. Also, there is no variation in size among camouflaged cells with a particular number of inputs due to the common physical layout while the standard cell library would have different areas. Lastly, certain functions were not available for the camouflaged cell library due to their instability in the presence environmental variations. Also, the unavailability of XOR/XNOR functionality for gates with more than 2 inputs adversely affects the netlist area.

7.4 SAT Solver based De-obfuscation

In this section, we explore the resilience of our camouflaged cell design against SAT solver attacks. We, first, describe the capabilities of an attacker seeking to reverse engineer our design. Then, we compare the performance of our camouflaging against utilizing a limited gate camouflaging technique [101] with respect to the de-obfuscation time for ISCAS-85 benchmark circuits. Lastly, our camouflaged cells can realize certain functions where the output is decided by only a subset of the inputs while the rest of the inputs can be considered as dummies. We explore the utilization of such cells in a design and their effect on de-obfuscation performance.

7.4.1 Attacker Capabilities

Due to the chip manufacturer (foundry) having perfect information about the layout, we assume the manufacturer is a *trusted* vendor. Our attacker is a third-party who wishes to reverse engineer our protected IP and is able to obtain multiple chips containing our design. The attacker is capable of reverse-engineering the layout of the IP using imaging and de-layering techniques and extract basic physical information such as the number of primary inputs/outputs, gates and their interconnections. The attacker can identify non-camouflaged gates readily. The attacker can differentiate between different sized camouflaged gates and can easily identify inverters. However, we make the assumption that the attacker cannot distinguish between legitimate and dummy vias. Also, the attacker does not have information about the logic functions implemented in our camouflaged cell library. The attacker is also able to query our IP block without restrictions and obtain the outputs for usage in the Oracle-guided Incremental SAT solver technique [132].

7.4.2 SAT Attack

The main advantage of our proposed camouflaging is the large number of possible logic functions that can be realized with the same physical layout for a given n-

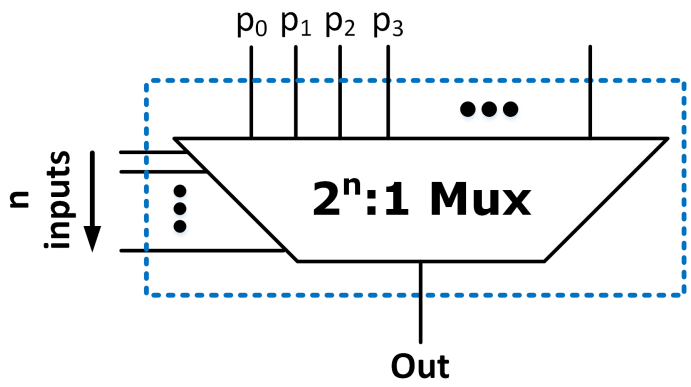


Figure 7.2: General structure of Multi-Vt FinFET camouflaged cell for SAT Solver [132]

input gate. To illustrate the increased de-obfuscation effort, we compare against a camouflaging technique with a cell design implements a limited set of functions like NAND/NOR/XOR [101].

An attacker using the powerful Oracle-guided SAT based de-obfuscation scheme [132] can model an n-input camouflaged cell as a $2^n : 1$ Mux with n select inputs, as shown in Figure 7.2. The programming vector bits, $\{p_0, p_1, \ldots, p_{2^n-1}\}$ are utilized by the SAT solver to guess the output functions based on querying the circuit multiple times and making observations about the internal functionality. For example, solving a 2-input NAND gate with inputs $\{AB\}$ taking the values $\{00, 01, 10, 11\}$ would output $\{p_0, p_1, p_2, p_3\} = \{1, 1, 1, 0\}$. For the limited NAND/NOR/XOR scheme, the attacker would have to guess only 3 sets of values to distinguish between the possible gates. There is no increased effort when using camouflaged gate with greater number of inputs as this information is readily available in the layout and the number of logic functions implemented is still the same. However, since our camouflaged cell design can implement a large array of functions (2^{2^n} possible), the SAT solver would have to consider all possible combinations for the programming vector bits.

For the limited camouflaging technique, we synthesized the ISCAS-85 benchmark circuits using the commercial 16 nm standard cell library while allowing the design

to only utilize 2-, 3- and 4-input versions of NAND, NOR and XOR gates. The total number of gates in the final netlist is tabulated in Table 7.4 and is noted to be greater than the number of gates needed by our camouflage technique (listed in Table 7.3). However, the maximum increase in the number of gates is less than 24%.

To obtain the de-obfuscation effort for both techniques, we assume that 25% of the gates in a the synthesized netlist are obfuscated to the attacker and need to be solved for using SAT. We select the gates to be camouflaged randomly and for each benchmark circuit/camouflaging technique, 10 iterations are performed with a specified Timeout of 24 hours (86400 s). Such a scenario can be considered practical, as a designer may only choose to obfuscate part of the IP to keep the area overhead to a minimum.

Table 7.4 lists the highest de-obfuscation time observed for each benchmark circuit and camouflaging technique. The results indicate a minimum of 10× increase in the de-obfuscation effort for the attacker when using our camouflaged cells with 2 orders of magnitude increase being more likely.

7.4.3 Dummy-input Gates

Among the available logic functions from our camouflaged cells, there are certain functions whose output is determined only by a subset of inputs. For example, in a function $Y = a \cdot (b + \bar{b})$, the input b is redundant and can be considered as a dummy input while the function implements a buffer for a. For gates with $n > 2$ inputs, more complex functions with more redundant inputs can exist.

For our work, we explore the effect of leveraging such functions to further increase the reverse-engineering effort for SAT attacks. Specifically, we will consider a 2-input buffer with one dummy input and 3-input buffer with two dummies. This allows us to add such gates anywhere in the design without changing the functionality of the downstream logic. Once inserted, we can connect a dummy input to an *arbitrary*

Table 7.4: De-obfuscation effort for limited and Multi-Vt FinFET Camouflaging with [Timeout] of 24 hours

Circuit	**Limited Camouflaging**		**Multi-Vt FinFET Camouflaging**
	NAND/NOR/XOR		
	# Gates	**Attack Time (s)**	**Attack Time (s)**
c432	120	53	14088
c499	150	41	30169
c1908	225	201	[Timeout]
c2670	536	2990	33843
c3540	749	259	12385
c7552	1233	1376	[Timeout]

output generated in a previous *logic level* or to a primary input. This prevents the possibility of creating logic loops.

We chose to analyze 3 designs - *c432*, *c499* and *2670*. We consider the full design to be de-obfuscated and only add the camouflaged dummy-input gates. This allows us to only observe the effect of adding such gates. We perform the gate insertion in stages to find the minimum number of dummy-input gates that can provide the maximum increase in SAT attack effort. The number of dummy-input gates added in each subsequent stage increases by 5% of the total number of gates in the original netlist and we choose either the 2- or 3-input version with equal probability. We generate 10 netlists with randomly inserted dummy-input gates and run the SAT attack for each netlist with a timeout of 24 hours.

From the results in Table 7.5, we can see that it is possible to significantly increase the reverse-engineering effort by multiple orders of magnitude with just inserting

Table 7.5: Resilience of Dummy-input gate insertion to SAT attacks (Timeout = 24 hrs)

Circuit	**Dummy inserted (% of #Gates)**	**Attack Time (s)**
c432	15	[Timeout]
c499	10	25310
c2670	5	[Timeout]

dummy-input gates in a design. In practical usage, such gates would be included in the cell library and the dummy connections would be made after design synthesis.

7.5 Conclusion

Hardware obfuscation is a promising approach for protecting intellectual property against reverse engineering attacks. In this chapter, we proposed an n-input circuit design that can realize a vast number of unique output functions just by changing the threshold voltage options used for certain FinFET transistors, while keeping the physical layout the same. We compared our camouflaged gates to regular gate designs and while our designs due incur an overhead in terms of area and delay, they performed better in terms of power. However, we show that the structurally identical camouflaged cells are able to increase the reverse engineering effort by an order of magnitude for a SAT-based attack. Furthermore, we demonstrated that the reverse engineering effort can further be increased by utilizing dummy-input gates which do not change the logic functionality, but appear to be legitimate to an attacker.

CHAPTER 8

CONCLUSION

Physically Unclonable Functions (PUFs) have shown promise as lightweight hardware security primitives with various applications ranging from key-generation and authentication to IP protection. The salient properties of PUFs are – Uniqueness, Reliability and Unpredictability/Security. Hence, improving these core properties of PUFs is essential for their deployment in practical applications. The first half of this dissertation focused on solutions to improve reliability and unpredictability of PUFs.

Weak PUFs have been primarily utilized for cryptographic key generation applications and thus, are required to have very high reliability (¡ 1 failure in 10^6 key generations). Typically, Weak PUF bits undergo post-processing to generate error-free keys. However, such an approach can incur significant area and power overheads, reducing viability for usage in resource-constrained environments. In this dissertation, we explored new circuit designs to better harness inherent process variations to create a more robust Weak PUF bit. This reduces the post-processing resources needed to generate highly reliable keys. Furthermore, we proposed an intelligent burn-in/accelerated aging system to further improve the reliability of the Weak PUF cells. Our system provides relevant feedback, in an automated fashion, to allow manufacturers to reduce the burn-in time required and hence, increase manufacturing throughput.

Strong PUFs can provide a low cost alternative to cryptography-based authentication. However, unreliability in Strong PUFs can increase the cost of proper authentication and low resistance against model building attacks using machine learning tech-

niques creates a security risk. In this dissertation, we address the problem of machine learning resistance by proposing new Strong PUF architectures utilizing WiSARD, a simple Weightless Neural Network (WNN) model. Furthermore, the WNN Strong PUF architectures were extended to utilize a reliable Weak PUF entropy source to realize a robust Strong PUF. The concept of utilizing reliable entropy source was further explored to generate Strong PUF system based on pattern-matching schemes that compare the Weak PUF bits and states of a linear-feedback shift register (LFSR) to generate robust and secure keys. All of our proposed Strong PUF designs are lightweight and hence, can be incorporated into a large array of devices with varying resource constraints. Further research includes fine-tuning the proposed designs for fabrication and testing for real-world applications.

Obfuscation techniques have been proposed, at various levels of design abstraction, to counter reverse engineering with an intent to steal IP from legal parties by malicious actors. In this dissertation, we focused on system-level meta-obfuscation techniques to prevent information leakage from the physical layout and developed metrics to measure the quality of obfuscation. Our methodology maximized the obfuscation metrics utilizing evolutionary algorithms. This dissertation also explored circuit-level obfuscation by leveraging multi-threshold FinFETs to create n-input circuit design that can realize a vast number of unique output functions by just changing the assigned FinFET, without any changes in physical layout. These camouflaged gates were compared against regular gate designs in terms of area, power and latency overheads and performed favorably. Furthermore, their resilience against reverse-engineering attacks were demonstrated utilizing standard benchmark circuits.

With Internet-of-Things (IoTs) being touted as a major future technology revolution, securing these systems against various forms of attack is of paramount importance. Low cost security solutions will be of great necessity to fully realize the potential of IoTs. PUFs show promise in implementing lightweight security features

and the solutions proposed in this dissertation will make them viable for widespread deployment. Also, intellectual property theft is a universal problem and this dissertation provides hardware-based solutions to prevent reverse-engineering of valuable design IPs.

BIBLIOGRAPHY

[1] Cadence Encounter Digital Implementation System. http://www.cadence.com/products/di/edi-system/pages/default.aspx.

[2] Iedm 2017 – controlling threshold voltage with work function metals. https://semiwiki.com/semiconductor-services/ic-knowledge/7259-iedm-2017-controlling-threshold-voltage-with-work-function-metals/. (Visited on 05/10/2019).

[3] Imec advancing state-of-the-art in finfets. https://phys.org/news/2007-06-imec-advancing-state-of-the-art-finfets.html. (Visited on 05/10/2019).

[4] International Technology Roadmap for Semiconductor (ITRS). http://www.itrs.net/Links/2006Update/2006UpdateFinal.htm.

[5] Nangate Open Cell Library. http://www.si2.org/openeda.si2.org/projects/nangatelib.

[6] NCSU FreePDK 45nm. http://www.eda.ncsu.edu/wiki/FreePDK45:Contents.

[7] Predictive Technology Model (PTM). http://ptm.asu.edu/.

[8] Pyevolve Genetic Algorithms. http://pyevolve.sourceforge.net/.

[9] scikit-learn: Machine Learning in Python. http://scikit-learn.org/stable/.

[10] SEMI. Innovation is at risk as semiconductor equipment and materials industry loses up to $4 billion annually due to IP infringement. www.semi.org/en/Press/P043775.

[11] Simulated annealing. https://en.wikipedia.org/wiki/Simulated-annealing.

[12] Six ways to exploit the advantages of finfets. https://www.techdesignforums.com/practice/technique/six-ways-to-exploit-the-advantages-of-finfets/. (Visited on 01/10/2019).

[13] Trusted Integrated Chips (TIC) Program, Intelligence Advanced Research Projects Activity. http://www.iarpa.gov/index.php/research-programs/tic.

[14] Welcome to LightGBM's documentation!

[15] Akkaya, N. E. C., Erbagci, B., and Mai, K. A secure camouflaged logic family using post-manufacturing programming with a 3.6GHz adder prototype in 65nm CMOS at 1V nominal VDD. In *2018 IEEE International Solid - State Circuits Conference - (ISSCC)* (2018), pp. 128–130.

[16] Aleksander, I., Thomas, W.V., and Bowden, P.A. WISARD·a radical step forward in image recognition. *Sensor Review 4*, 3 (1984), 120–124.

[17] Aleksander, Igor, Gregorio, Massimo De, França, Felipe Maia Galvão, Lima, Priscila Machado Vieira, and Morton, Helen. A brief introduction to Weightless Neural Systems. In *ESANN 2009, 17th European Symposium on Artificial Neural Networks, Bruges, Belgium, April 22-24, 2009, Proceedings* (2009).

[18] Alkabani, Y., Koushanfar, F., and Potkonjak, M. Remote Activation of ICs for Piracy Prevention and Digital Right Management. In *Computer-Aided Design, 2007. ICCAD 2007. IEEE/ACM International Conference on* (Nov 2007), pp. 674–677.

[19] Alkabani, Yousra M., and Koushanfar, Farinaz. Active Hardware Metering for Intellectual Property Protection and Security. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium* (Berkeley, CA, USA, 2007), SS'07, USENIX Association, pp. 20:1–20:16.

[20] Alkatheiri, Mohammed Saeed, and Zhuang, Yu. Towards fast and accurate machine learning attacks of feed-forward arbiter pufs. In *2017 IEEE Conference on Dependable and Secure Computing* (2017), pp. 181–187.

[21] Anderson, Ross. *Security engineering: A guide to building dependable distributed systems. 2001.* Wiley.

[22] Aseeri, Ahmad O., Zhuang, Yu, and Alkatheiri, Mohammed Saeed. A Machine Learning-Based Security Vulnerability Study on XOR PUFs for Resource-Constraint Internet of Things. In *2018 IEEE International Congress on Internet of Things (ICIOT)* (2018), pp. 49–56.

[23] Bhardwaj, Sarvesh, Wang, Wenping, Vattikonda, Rakesh, Cao, Yu, and Vrudhula, Sarma. Predictive modeling of the NBTI effect for reliable design. In *IEEE Custom Integrated Circuits Conference 2006* (2006), IEEE, pp. 189–192.

[24] Bhargava, M., and Mai, K. An efficient reliable PUF-based cryptographic key generator in 65nm CMOS. In *2014 Design, Automation Test in Europe Conference Exhibition (DATE)* (March 2014), pp. 1–6.

[25] Bhargava, Mudit, Cakir, Cagla, and MAI, Khanh. Attack resistant sense amplifier based PUFs (SA-PUF) with deterministic and controllable reliability of PUF responses. In *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on* (2010), IEEE, pp. 106–111.

[26] Bhargava, Mudit, and Mai, Ken. A high reliability PUF using hot carrier injection based response reinforcement. In *Cryptographic Hardware and Embedded Systems-CHES 2013*. Springer, 2013, pp. 90–106.

[27] Bledsoe, W. W., and Browning, I. Pattern Recognition and Reading by Machine. In *Papers Presented at the December 1-3, 1959, Eastern Joint IRE-AIEE-ACM Computer Conference* (New York, NY, USA, 1959), IRE-AIEE-ACM '59 (Eastern), ACM, pp. 225–232.

[28] Bösch, Christoph. Efficient Fuzzy Extractors for Reconfigurable Hardware. Master's thesis, Dept. EECS, Massachusetts Institute of Technology, 2008.

[29] Bösch, Christoph, Guajardo, Jorge, Sadeghi, Ahmad-Reza, Shokrollahi, Jamshid, and Tuyls, Pim. *Efficient Helper Data Key Extractor on FPGAs*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 181–197.

[30] Bucci, M., and Luzzi, R. Identification circuit and method for generating an identification bit using physical unclonable functions, Nov. 12 2013. US Patent 8,583,710.

[31] Chakraborty, R.S., and Bhunia, S. HARPOON: An Obfuscation-based SoC Design Methodology for Hardware Protection. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on 28*, 10 (Oct 2009), 1493–1502.

[32] Chow, L.W., Baukus, J.P., Wang, B.J., and Cocchi, R.P. Camouflaging a standard cell based integrated circuit, Apr. 3 2012. US Patent 8,151,235.

[33] Cocchi, R.P., Baukus, J.P., Chow, Lap Wai, and Wang, B.J. Circuit Camouflage Integration for Hardware IP Protection. In *Design Automation Conference (DAC), 2014 51st ACM/EDAC/IEEE* (June 2014), pp. 1–5.

[34] Collantes, M. I. M., Massad, M. E., and Garg, S. Threshold-Dependent Camouflaged Cells to Secure Circuits Against Reverse Engineering Attacks. In *2016 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)* (2016), pp. 443–448.

[35] Colombier, B., and Bossuet, L. Survey of hardware protection of design data for integrated circuits and intellectual properties. *IET Computers Digital Techniques 8*, 6 (2014), 274–287.

[36] Cortez, Mafalda, Hamdioui, Said, van der Leest, Vincent, Maes, Roel, and Schrijen, Geert-Jan. Adapting voltage ramp-up time for temperature noise reduction on memory-based PUFs. In *Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on* (2013), IEEE, pp. 35–40.

[37] Costello, Katie, and Omale, Gloria. Gartner says global it spending to reach $3.8 trillion in 2019. https://www.gartner.com/en/newsroom/press-releases/2019-01-28-gartner-says-global-it-spending-to-reach–3-8-trillio. (Visited on 01/30/2019).

[38] DeJean, Gerald, and Kirovski, Darko. *RF-DNA: Radio-frequency certificates of authenticity*. Springer, 2007.

[39] Delvaux, J., Gu, D., Schellekens, D., and Verbauwhede, I. Helper Data Algorithms for PUF-Based Key Generation: Overview and Analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 34*, 6 (June 2015), 889–902.

[40] Dodis, Yevgeniy, Ostrovsky, Rafail, Reyzin, Leonid, and Smith, Adam. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM J. Comput. 38*, 1 (Mar. 2008), 97–139.

[41] Dollfus, Philippe, Bournel, Arnaud, Galdin, Sylvie, Barraud, Sylvain, and Hesto, Patrice. Effect of discrete impurities on electron transport in ultrashort MOSFET using 3D MC simulation. *IEEE Transactions on Electron Devices 51*, 5 (2004), 749–756.

[42] Friedman, E. G. Clock distribution networks in synchronous digital integrated circuits. *Proceedings of the IEEE 89*, 5 (May 2001), 665–692.

[43] Fyrbiak, M., Strauß, S., Kison, C., Wallat, S., Elson, M., Rummel, N., and Paar, C. Hardware reverse engineering: Overview and open challenges. In *2017 IEEE 2nd International Verification and Security Workshop (IVSW)* (2017), pp. 88–94.

[44] Ganji, Fatemeh. *On the learnability of physically unclonable functions*. PhD thesis, Technische Universität Berlin, 2017.

[45] Ganta, D., and Nazhandali, L. Circuit-level approach to improve the temperature reliability of Bi-stable PUFs. In *Quality Electronic Design (ISQED), 2014 15th International Symposium on* (March 2014), pp. 467–472.

[46] Garg, A., and Kim, T.T. Design of SRAM PUF with improved uniformity and reliability utilizing device aging effect. In *Circuits and Systems (ISCAS), 2014 IEEE International Symposium on* (June 2014), pp. 1941–1944.

[47] Gassend, Blaise, Clarke, Dwaine, van Dijk, Marten, and Devadas, Srinivas. Silicon Physical Random Functions. In *Proceedings of the 9th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2002), CCS '02, ACM, pp. 148–160.

[48] Ghoreishizadeh, S. S., Yalçın, T., Pullini, A., Micheli, G. De, Burleson, W., and Carrara, S. A Lightweight Cryptographic System for Implantable Biosensors. In *2014 IEEE Biomedical Circuits and Systems Conference (BioCAS) Proceedings* (Oct 2014), pp. 472–475.

[49] Goo, Jung-Suk, Choi, Chang-Hoon, Abramo, A., Ahn, Jae-Gyung, Yu, Zhiping, Lee, T. H., and Dutton, R. W. Physical origin of the excess thermal noise in short channel MOSFETs. *IEEE Electron Device Letters 22*, 2 (Feb 2001), 101–103.

[50] Guajardo, Jorge, Kumar, Sandeep S., Schrijen, Geert-Jan, and Tuyls, Pim. *FPGA Intrinsic PUFs and Their Use for IP Protection.* Springer Berlin Heidelberg, Berlin, Heidelberg, 2007, pp. 63–80.

[51] Guajardo, Jorge, Škorić, Boris, Tuyls, Pim, Kumar, Sandeep S, Bel, Thijs, Blom, Antoon HM, and Schrijen, Geert-Jan. Anti-counterfeiting, key distribution, and key storage in an ambient world via physical unclonable functions. *Information Systems Frontiers 11*, 1 (2009), 19–41.

[52] Guin, U., Huang, K., DiMase, D., Carulli, J. M., Tehranipoor, M., and Makris, Y. Counterfeit Integrated Circuits: A Rising Threat in the Global Semiconductor Supply Chain. *Proceedings of the IEEE 102*, 8 (Aug 2014), 1207–1228.

[53] Hansen, M. C., Yalcin, H., and Hayes, J. P. Unveiling the iscas-85 benchmarks: a case study in reverse engineering. *IEEE Design Test of Computers 16*, 3 (1999), 72–80.

[54] Hofer, Maximilian, and Boehm, Christoph. An alternative to error correction for SRAM-like PUFs. In *International Workshop on Cryptographic Hardware and Embedded Systems* (2010), Springer, pp. 335–350.

[55] Holcomb, Daniel E, Burleson, Wayne P, and Fu, Kevin. Initial sram state as a fingerprint and source of true random numbers for rfid tags. In *Proceedings of the Conference on RFID Security* (2007), vol. 7.

[56] Holcomb, Daniel E, and Fu, Kevin. Bitline PUF: building native challenge-response PUF capability into any SRAM. In *International Workshop on Cryptographic Hardware and Embedded Systems* (2014), Springer, pp. 510–526.

[57] Holcomb, D.E., Burleson, W.P., and Fu, K. Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *Computers, IEEE Transactions on 58*, 9 (Sept 2009), 1198–1210.

[58] Hospodar, Gabriel, Maes, Roel, and Verbauwhede, Ingrid. Machine learning attacks on 65nm Arbiter PUFs: Accurate modeling poses strict bounds on usability. In *2012 IEEE International Workshop on Information Forensics and Security (WIFS)* (2012), pp. 37–42.

[59] Islam, M. N., Patil, V. C., and Kundu, S. A guide to graceful aging: How not to overindulge in post-silicon burn-in for enhancing reliability of weak PUF. In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)* (May 2017), pp. 1–4.

[60] Iyengar, Anirudh S., Vontela, Deepak, Reddy, Ithihasa, Ghosh, Swaroop, Motaman, Syedhamidreza, and Jang, Jae-won. Threshold defined camouflaged gates in 65nm technology for reverse engineering protection. In *Proceedings of the International Symposium on Low Power Electronics and Design* (New York, NY, USA, 2018), ISLPED '18, Association for Computing Machinery.

[61] Jang, J. W., and Ghosh, S. Design and analysis of novel SRAM PUFs with embedded latch for robustness. In *Sixteenth International Symposium on Quality Electronic Design* (March 2015), pp. 298–302.

[62] Jha, Susmit, Gulwani, Sumit, Seshia, Sanjit A, and Tiwari, Ashish. Oracle-guided component-based program synthesis. In *Software Engineering, 2010 ACM/IEEE 32nd International Conference on* (2010), vol. 1, IEEE, pp. 215–224.

[63] Jiang, Dan, and Chong, Cheun Ngen. Anti-counterfeiting using phosphor puf. In *Anti-counterfeiting, Security and Identification, 2008. ASID 2008. 2nd International Conference on* (2008), IEEE, pp. 59–62.

[64] Johnson, J. B. Thermal Agitation of Electricity in Conductors. *Phys. Rev. 32* (Jul 1928), 97–109.

[65] Kalyanaraman, M., and Orshansky, M. Novel strong PUF based on nonlinearity of MOSFET subthreshold operation. In *Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on* (June 2013), pp. 13–18.

[66] Kammerstetter, Markus, Muellner, Markus, Burian, Daniel, Platzer, Christian, and Kastner, Wolfgang. Breaking Integrated Circuit Device Security through Test Mode Silicon Reverse Engineering. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security* (2014), ACM, pp. 549–557.

[67] Kang, Kunhyuk, Park, Sang Phill, Roy, Kaushik, and Alam, Muhammad A. Estimation of statistical variation in temporal nbti degradation and its impact on lifetime circuit performance. In *Proceedings of the 2007 IEEE/ACM international conference on Computer-aided design* (2007), IEEE Press, pp. 730–734.

[68] Komano, Yuichi, Ohta, Kazuo, Sakiyama, Kazuo, Iwamoto, Mitsugu, and Verbauwhede, Ingrid. Single-Round Pattern Matching Key Generation Using Physically Unclonable Function. *Security and Communication Networks 2019* (2019).

[69] Koushanfar, F. Provably Secure Active IC Metering Techniques for Piracy Avoidance and Digital Rights Management. *Information Forensics and Security, IEEE Transactions on 7*, 1 (Feb 2012), 51–63.

[70] Kumar, R., and Burleson, W. Hybrid modeling attacks on current-based PUFs. In *Computer Design (ICCD), 2014 32nd IEEE International Conference on* (Oct 2014), pp. 493–496.

[71] Kumar, R., and Burleson, W. On design of a highly secure PUF based on non-linear current mirrors. In *Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on* (May 2014), pp. 38–43.

[72] Kumar, Raghavan, and Burleson, Wayne. Side-Channel Assisted Modeling Attacks on Feed-Forward Arbiter PUFs Using Silicon Data. In *Radio Frequency Identification. Security and Privacy Issues*, Stefan Mangard and Patrick Schaumont, Eds., vol. 9440 of *Lecture Notes in Computer Science*. 2015, pp. 53–67.

[73] Kumar, Sanjay V, Kim, Chris H, and Sapatnekar, Sachin S. An analytical model for negative bias temperature instability. In *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design* (2006), ACM, pp. 493–496.

[74] LAM, Suk Wah Louisa. Theory and application of majority vote: From condorcet jury theorem to pattern recognition. *2nd Int. Conf. mathematics education into the 21st century: Mathematics for Living* (2000).

[75] Lee, J.W., Lim, D., Gassend, B., Suh, G.E., van Dijk, M., and Devadas, S. A technique to build a secret key in integrated circuits for identification and authentication applications. In *VLSI Circuits, 2004. Digest of Technical Papers. 2004 Symposium on* (June 2004), pp. 176–179.

[76] Linnartz, Jean-Paul, and Tuyls, Pim. *New Shielding Functions to Enhance Privacy and Prevent Misuse of Biometric Templates*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003, pp. 393–402.

[77] Liu, D., Yu, C., Zhang, X., and Holcomb, D. Oracle-guided incremental sat solving to reverse engineer camouflaged logic circuits. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)* (March 2016), pp. 433–438.

[78] Maes, Roel. An Accurate Probabilistic Reliability Model for Silicon PUFs. In *Proceedings of the 15th International Conference on Cryptographic Hardware and Embedded Systems* (2013), CHES'13, Springer-Verlag, pp. 73–89.

[79] Maes, Roel. An Accurate Probabilistic Reliability Model for Silicon PUFs. Cryptology ePrint Archive, Report 2013/376, 2013.

[80] Maes, Roel, Tuyls, Pim, and Verbauwhede, Ingrid. A soft decision helper data algorithm for SRAM PUFs. In *Information Theory, 2009. ISIT 2009. IEEE International Symposium on* (2009), IEEE, pp. 2101–2105.

[81] Maes, Roel, Tuyls, Pim, and Verbauwhede, Ingrid. Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs. In *Cryptographic Hardware and Embedded Systems-CHES 2009*. Springer, 2009, pp. 332–347.

[82] Maes, Roel, and van der Leest, Vincent. Countering the effects of silicon aging on SRAM PUFs. In *Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on* (2014), IEEE, pp. 148–153.

[83] Maes, Roel, Van Herrewege, Anthony, and Verbauwhede, Ingrid. Pufky: A fully functional puf-based cryptographic key generator. In *Cryptographic Hardware and Embedded Systems–CHES 2012*. Springer, 2012, pp. 302–319.

[84] Malik, S., Becker, G. T., Paar, C., and Burleson, W. P. Development of a Layout-Level Hardware Obfuscation Tool. In *2015 IEEE Computer Society Annual Symposium on VLSI* (2015), pp. 204–209.

[85] Mathew, Sanu K, Satpathy, Sudhir K, Anders, Mark A, Kaul, Himanshu, Hsu, Steven K, Agarwal, Amit, Chen, Gregory K, Parker, Rachael J, Krishnamurthy, Ram K, and De, Vivek. A 0.19 pJ/b PVT-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22nm CMOS. In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)* (2014), IEEE, pp. 278–279.

[86] Matsukawa, T., Liu, Y., Endo, K., i. O'uchi, S., and Masahara, M. Variability origins of FinFETs and perspective beyond 20nm node. In *IEEE 2011 International SOI Conference* (Oct 2011), pp. 1–28.

[87] McDonald, J Todd, Kim, Yong C, and Grimaila, Michael R. Protecting reprogrammable hardware with polymorphic circuit variation. In *Proceedings of the 2nd Cyberspace Research Workshop* (2009), pp. 63–78.

[88] McDonald, J Todd, and Yasinsac, Alec. Program Intent Protection Using Circuit Encryption. In *Proceedings of the 8th International Symposium on System and Information Security* (2006).

[89] Mistry, K., Allen, C., Auth, C., Beattie, B., Bergstrom, D., Bost, M., Brazier, M., Buehler, M., Cappellani, A., Chau, R., Choi, C. H., Ding, G., Fischer, K., Ghani, T., Grover, R., Han, W., Hanken, D., Hattendorf, M., He, J., Hicks, J., Huessner, R., Ingerly, D., Jain, P., James, R., Jong, L., Joshi, S., Kenyon, C., Kuhn, K., Lee, K., Liu, H., Maiz, J., McIntyre, B., Moon, P., Neirynck, J., Pae, S., Parker, C., Parsons, D., Prasad, C., Pipes, L., Prince, M., Ranade, P., Reynolds, T., Sandford, J., Shifren, L., Sebastian, J., Seiple, J., Simon, D., Sivakumar, S., Smith, P., Thomas, C., Troeger, T., Vandervoorn, P., Williams, S., and Zawadzki, K. A 45nm Logic Technology with High-k+Metal Gate Transistors, Strained Silicon, 9 Cu Interconnect Layers, 193nm Dry Patterning, and 100% Pb-free Packaging. In *2007 IEEE International Electron Devices Meeting* (Dec 2007), pp. 247–250.

[90] Mursi, Khalid T., Thapaliya, Bipana, Zhuang, Yu, Aseeri, Ahmad O., and Alkatheiri, Mohammed Saeed. A Fast Deep Learning Method for Security Vulnerability Study of XOR PUFs. *Electronics 9*, 10 (2020).

[91] Nohl, Karsten, Evans, David, Starbug, Starbug, and Plötz, Henryk. Reverse-engineering a Cryptographic RFID Tag. In *Proceedings of the 17th Conference on Security Symposium* (Berkeley, CA, USA, 2008), SS'08, USENIX Association, pp. 185–193.

[92] Nyquist, H. Thermal Agitation of Electric Charge in Conductors. *Phys. Rev. 32* (Jul 1928), 110–113.

[93] Pappu, Ravikanth, Recht, Ben, Taylor, Jason, and Gershenfeld, Neil. Physical one-way functions. *Science 297*, 5589 (2002), 2026–2030.

[94] Paral, Z., and Devadas, S. Reliable and efficient puf-based key generation using pattern matching. In *2011 IEEE International Symposium on Hardware-Oriented Security and Trust* (June 2011), pp. 128–133.

[95] Parham, James D., McDonald, J. Todd, Kim, Yong C., and Grimaila, Michael R. Hiding Circuit Components Using Boundary Blurring Techniques. In *Proc. of IEEE Annual Symposium on VLSI* (2010).

[96] Patil, Vinay C., Vijayakumar, Arunkumar, Holcomb, Daniel E., and Kundu, Sandip. Improving Reliability of Weak PUFs via Circuit Techniques to Enhance Mismatch. In *2017 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)* (May 2017), IEEE, pp. 146–150.

[97] Paul, Bipul C, Kang, Kunhyuk, Kufluoglu, Haldun, Alam, Muhammad A, and Roy, Kaushik. Impact of nbti on the temporal performance degradation of digital circuits. *IEEE Electron Device Letters 26*, 8 (2005), 560–562.

[98] Pille, J., Adams, C., Christensen, T., Cottier, S. R., Ehrenreich, S., Kono, F., Nelson, D., Takahashi, O., Tokito, S., Torreiter, O., Wagner, O., and Wendel, D. Implementation of the Cell Broadband Engine™; in 65 nm SOI Technology Featuring Dual Power Supply SRAM Arrays Supporting 6 GHz at 1.3 V. *IEEE Journal of Solid-State Circuits 43*, 1 (Jan 2008), 163–171.

[99] Quadir, Shahed E., Chen, Junlin, Forte, Domenic, Asadizanjani, Navid, Shahbazmohamadi, Sina, Wang, Lei, Chandy, John, and Tehranipoor, Mark. A Survey on Chip to System Reverse Engineering. *J. Emerg. Technol. Comput. Syst. 13*, 1 (Apr. 2016).

[100] Rajendran, J., Kanuparthi, A.K., Zahran, M., Addepalli, S.K., Ormazabal, G., and Karri, R. Securing Processors Against Insider Attacks: A Circuit-Microarchitecture Co-Design Approach. *Design Test, IEEE 30*, 2 (April 2013), 35–44.

[101] Rajendran, Jeyavijayan, Sam, Michael, Sinanoglu, Ozgur, and Karri, Ramesh. Security Analysis of Integrated Circuit Camouflaging. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security* (New York, NY, USA, 2013), CCS '13, ACM, pp. 709–720.

[102] Rajendran, Jeyavijayan, Sinanoglu, Ozgur, and Karri, Ramesh. Is split manufacturing secure? In *Design, Automation Test in Europe Conference Exhibition (DATE), 2013* (March 2013), pp. 1259–1264.

[103] Ramesh, P., Patil, V. C., and Kundu, S. Peer pressure on identity: On requirements for disambiguating PUFs in noisy environment. In *2017 IEEE North Atlantic Test Workshop (NATW)* (May 2017), pp. 1–4.

[104] Risch, L. Pushing CMOS beyond the roadmap. *Solid-State Electronics 50*, 4 (2006), 527–535.

[105] Roy, Gareth, Brown, Andrew R, Adamu-Lema, Fikru, Roy, Scott, and Asenov, Asen. Simulation study of individual and combined sources of intrinsic parameter fluctuations in conventional nano-MOSFETs. *IEEE Transactions on Electron Devices 53*, 12 (2006), 3063–3070.

[106] Roy, J.A., Koushanfar, F., and Markov, I.L. EPIC: Ending Piracy of Integrated Circuits. In *Design, Automation and Test in Europe, 2008. DATE '08* (March 2008), pp. 1069–1074.

[107] Rührmair, Ulrich, Sehnke, Frank, Sölter, Jan, Dror, Gideon, Devadas, Srinivas, and Schmidhuber, Jürgen. Modeling Attacks on Physical Unclonable Functions. In *Proceedings of the 17th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2010), CCS '10, ACM, pp. 237–249.

[108] Santiago, L., Patil, V. C., Prado, C. B., Alves, T. A. O., Marzulo, L. A. J., França, F. M. G., and Kundu, S. Realizing strong PUF from weak PUF via neural computing. In *2017 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT)* (Oct 2017), pp. 1–6.

[109] Santiago de Araújo, Leandro, C. Patil, Vinay, B. Prado, Charles, A. O. Alves, Tiago, A. J. Marzulo, Leandro, M. G. França, Felipe, and Kundu, Sandip. Design of robust, high-entropy strong pufs via weightless neural network. *Journal of Hardware and Systems Security 3*, 3 (Sep 2019), 235–249.

[110] Santikellur, Pranesh, Bhattacharyay, Aritra, and Chakraborty, Rajat Subhra. Deep Learning based Model Building Attacks on Arbiter PUF Compositions. *IACR Cryptol. ePrint Arch. 2019* (2019), 566.

[111] Sarpeshkar, Rahul, Delbruck, Tobias, and Mead, Carver A. White Noise in MOS Transistors and Resistors. *IEEE Circuits and Devices 9*, 6 (1993), 23–29.

[112] Schobert, Martin. *Softwaregestütztes Reverse-Engineering Von Logik-Gattern in Integrierten Schhaltkreisen*. PhD thesis, Humboldt-Universität zu Berlin, June 2011.

[113] Schrittwieser, S., Katzenbeisser, S., Merzdovnik, G., Kieseberg, P., and Weippl, E. AES-SEC: Improving Software Obfuscation through Hardware-Assistance. In *Availability, Reliability and Security (ARES), 2014 Ninth International Conference on* (Sept 2014), pp. 184–191.

[114] Skorobogatov, Sergei. Physical Attacks and Tamper Resistance. In *Introduction to Hardware Security and Trust*. Springer, 2012, pp. 143–173.

[115] Stolk, P. A., Widdershoven, F. P., and Klaassen, D. B. M. Modeling statistical dopant fluctuations in MOS transistors. *IEEE Transactions on Electron Devices 45*, 9 (Sep 1998), 1960–1971.

[116] Su, Ying, Holleman, Jeremy, and Otis, Brian P. A digital 1.6 pj/bit chip identification circuit using process variations. *IEEE Journal of Solid-State Circuits 43*, 1 (2008), 69–77.

[117] SypherMedia. SypherMedia Library - Circuit Camouflage Technology. `http://www.smi.tv/SMI_SypherMedia_Library_Intro.pdf`, 2012. Online; accessed 21-April-2015.

[118] Tehranipoor, Mohammad, and Wang, Cliff. *Introduction to Hardware Security and Trust*. Springer Publishing Company, Incorporated, 2011.

[119] Triantis, Dimitris P, Birbas, Alexios N, and Kondis, D. Thermal noise modeling for short-channel MOSFETs. *IEEE Transactions on Electron Devices 43*, 11 (1996), 1950–1955.

[120] Tsutsui, Gen, Saitoh, Masumi, Nagumo, Toshiharu, and Hiramoto, Toshiro. Impact of SOI thickness fluctuation on threshold voltage variation in ultra-thin body SOI MOSFETs. *IEEE Transactions on nanotechnology 4*, 3 (2005), 369–373.

[121] Tuyls, Pim, Schrijen, Geert-Jan, Škorić, Boris, Van Geloven, Jan, Verhaegh, Nynke, and Wolters, Rob. Read-proof hardware from protective coatings. In *Cryptographic Hardware and Embedded Systems-CHES 2006*. Springer, 2006, pp. 369–383.

[122] Vassighi, Arman, and Sachdev, Manoj. Thermal runaway in integrated circuits. *IEEE Transactions on Device and Materials Reliability 6*, 2 (2006), 300–305.

[123] Vattikonda, Rakesh, Wang, Wenping, and Cao, Yu. Modeling and minimization of pmos nbti effect for robust nanometer design. In *Proceedings of the 43rd annual Design Automation Conference* (2006), ACM, pp. 1047–1052.

[124] Vijayakumar, A., and Kundu, S. A novel modeling attack resistant PUF design based on non-linear voltage transfer characteristics. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2015* (March 2015), pp. 653–658.

[125] Vijayakumar, A., Patil, V. C., Prado, C. B., and Kundu, S. Machine learning resistant strong PUF: Possible or a pipe dream? In *2016 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)* (May 2016), pp. 19–24.

[126] Vijayakumar, Arunkumar, Patil, Vinay, and Kundu, Sandip. On Improving Reliability of SRAM-Based Physically Unclonable Functions. *Journal of Low Power Electronics and Applications 7*, 1 (Jan 2017), 2.

[127] Vijayakumar, Arunkumar, Patil, Vinay C., Holcomb, Daniel E., Paar, Christof, and Kundu, Sandip. Physical design obfuscation of hardware: A comprehensive investigation of device and logic-level techniques. *Trans. Info. For. Sec. 12*, 1 (Jan. 2017), 64–77.

[128] Wang, Yao, Cotofana, Sorin D, and Fang, Liang. Statistical reliability analysis of NBTI impact on FinFET SRAMs and mitigation technique using independent-gate devices. In *Nanoscale Architectures (NANOARCH), 2012 IEEE/ACM International Symposium on* (2012), IEEE, pp. 109–115.

[129] Xiao, Kan, Rahman, M.T., Forte, D., Huang, Yu, Su, Mei, and Tehranipoor, M. Bit selection algorithm suitable for high-volume production of SRAM-PUF. In *Hardware-Oriented Security and Trust (HOST), 2014 IEEE International Symposium on* (May 2014), pp. 101–106.

[130] Xu, Xiaolin, and Burleson, W. Hybrid side-channel/machine-learning attacks on PUFs: A new threat? In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014* (March 2014), pp. 1–6.

[131] Yasin, M., Mazumdar, B., Sinanoglu, O., and Rajendran, J. Removal attacks on logic locking and camouflaging techniques. *IEEE Transactions on Emerging Topics in Computing* (2017), 1–1.

[132] Yu, C., Zhang, X., Liu, D., Ciesielski, M., and Holcomb, D. Incremental sat-based reverse engineering of camouflaged logic circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 36*, 10 (2017), 1647–1659.

[133] Zheng, Jinxing, Li, Dongfang, and Potkonjak, Miodrag. A Secure and Unclonable Embedded System using Instruction-level PUF Authentication. In *Field Programmable Logic and Applications (FPL), 2014 24th International Conference on* (2014), IEEE, pp. 1–4.