A Systematic Approach to Design an Efficient Physical Unclonable Function

Abhranil Maiti

Dissertation submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy
in
Computer Engineering

Patrick Schaumont, Chair
Joseph Tront
Sandeep Shukla
Leyla Nazhandali
Inyoung Kim

April 30, 2012
Blacksburg, Virginia

Keywords: Physical Unclonable Function, Security, Process variation

A Systematic Approach to Design an Efficient Physical Unclonable Function

Abhranil Maiti

(ABSTRACT)

A Physical Unclonable Function (PUF) has shown a lot of promise to solve many security issues due to its ability to generate a random yet chip-unique secret in the form of an identifier or a key while resisting cloning attempts as well as physical tampering. It is a hardware-based challenge-response function which maps its responses to its challenges exploiting complex statistical variation in the logic and interconnect inside integrated circuits (ICs). An efficient PUF should generate a key that varies randomly from one chip to another. At the same time, it should reliably reproduce a key from a chip every time the key is requested from that chip. Moreover, a PUF should be robust to thwart any attack that aims to reveal its key. Designing an efficient PUF having all these qualities with a low cost is challenging. Furthermore, the efficiency of a PUF needs to be validated by characterizing it over a group of chips. This is because a PUF circuit is supposed to be instantiated in several chips, and whether it can produce a chip-unique identifier/key or not cannot be validated using a single chip. The main goal of this research is to propose a systematic approach to build a random, reliable, and robust PUF incurring minimal cost.

With this objective, we first formulate a novel PUF system model that uncouples PUF measurement from PUF identifier formation. The proposed model divides PUF operation into three separate but related components. We show that the three PUF quality factors, randomness, reliability, and robustness, can be improved at each component of the system model resulting in an overall improvement of a PUF. We proposed three PUF enhancement techniques using the system model in this research. The proposed techniques showed significant improvements in a PUF.

Second, we present a large-scale PUF characterization method to validate the efficiency of a PUF as a secure primitive. A compact and portable method measured a sizable set

of around 200 chips. We also performed experiments to test a PUF against variations in operating conditions (temperature, supply voltage) and circuit aging.

Third, we propose a method that can evaluate and compare the performance of different PUFs irrespective of their underlying working principles. This method can help a designer to select a PUF that is the most suitable one for a particular application.

Finally, a novel PUF that exploits the variability in the pipeline of a microprocessor is presented. This PUF has a very low area cost while it can be easily integrated using software programs in an application having a microprocessor.

Dedication

Dedicated to my late grandfather, Dr. Syamsundar Maiti, my mother, Kabyalakshmi, my two sisters, Rupabani and Banabani, and my brother, Indranil.

Acknowledgments

It is my duty to acknowledge multiple individuals who helped me directly or indirectly in reaching this far.

First of all, I must mention my advisor, Dr. Patrick Schaumont, for his precious guidance, support, encouragement, and knowledge that helped me accomplish my goals. His accurate thinking, discipline, and commitment to work have been a truly learning experience for me. He has been a great mentor as well as a source of inspiration throughout my PhD study.

My PhD committee members, Dr. Joseph Tront, Dr. Sandeep Shukla, Dr. Leyla Nazhandali, and Dr. Inyoung Kim have been really helpful with their guidance. I specially thank Dr. Kim for being a collaborator in my research. I must acknowledge the ECE Department, Virginia Tech for offering me an admission with a GTA. I thank National Science Foundation and ICTAS, Virginia Tech for funding my research. I also thank Cynthia Hopkins and Becky Simones for their help.

I had the privilege to work with a wonderful group of people in the lab. Eric Xu Guo, Zhimin Chen, Michael Gora, Sergey Morozov, Raghunandan Nagesh, Anand Reddy, Ambuj Sinha, Sri Krishna Iyer, Suvarna Mane, Lyndon Judge, and Mostafa Taha have been great friends. My research included collaboration with Raghu, Anand, Mike, and Sergey. I specially thank the undergraduate research collaborators: Jeff Casarona, Luke McHale, Logan McDougall, Vikash Gunreddy, and Michael Cantrell. You all have been really outstanding and made the research effort successful.

I am also thankful to Dr. Braja Krishna Mandal, Dr. Dimitrios Velenis, Dr. Alexander Flueck, and Dr. Erdal Oruklu at Illinois Institute of Technology for their support during my MS.

In Blacksburg, I had a wonderful group of friends without whom life would not have been easy. Their support and love kept me going all these years. I must mention Soumik, Sayan, Ganesh, Sandesh, Suvojit, Souvick, Souvik, Surya, Saikat, Arnab Roy, Arnab Gupta, Bikram, Shibabrat, Udit, Debamoy, Sanghamitra, Bireswar, Poulami, Deena, Syed, Sara, Mainak, Sumit, and Sachin. I also had a wonderful group of friends in Association for India's Development (AID): Vyas, Chaitanya, Sachi, Sunny, Naresh, Prathyusha, Tila, Abhijit, Sharmistha, Manjushree, Tannistha, Balaji, Sindhuja, Kapil, Shreya, Neeta, and Gaurav.

Finally, I am extremely indebted to my family. My mother, Kabyalakshmi has been the main source of support and inspiration always. My sisters, Rupabani and Banabani, and my brother, Indranil have been an integral part of my life. I am grateful to my aunts, Bijoylakshmi, Dipalakshmi, Gitalakshmi, and Ritulakshmi for their support and love. A special thanks goes to Gitalakshmi for her care and support to pursue my goals. I am also grateful to my grandmothers, Sovana and Late Nirmalasundari, my father, Kamal Krishna, my uncles, Kalyan Krishna and Manoranjan, my brother-in-laws, Arindam and Shantanu, my sister-in-law, Moumita, my cousins, Neilanjan and Manosita, my nephews, Riku and Pramit, and my niece, Riddhi for their love and support.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Why is Electronic Security Important?

Use of electronic devices is pervasive in every aspect of day-to-day life. They exist in a wide range of applications starting from hand-held devices such as mobile phones to servers in high-end data centers. With this wide range of applications, they process sensitive, user-specific data which, if disclosed, may lead to loss of privacy and many other unwanted implications. Additionally, intellectual property theft, software piracy, and counterfeit hardware are serious issues affecting the electronic industry significantly. 10% of all high-technology products sold globally are counterfeit [13]. As a result, electronic security has become a critical area of concern. With rapidly growing usage of electronic devices, it is highly likely that security will remain a practical concern for a long time to come.

Traditional security mechanism exploits cryptographic techniques to implement security measures such as authentication, integrity, confidentiality, and non-repudiation. The strength of such security measures depends on the secrecy of the key used for encryption or the device identifier used for authentication. Attackers aim to reveal this key in order to steal private data or to break an authentication protocol. Hence, both generation and

storage of the key/identifier need to be robust against attacks. Software-generated keys are deterministic in nature and thus easily predictable. On the other hand, storing a key in a non-volatile or battery-driven memory is not only expensive but also vulnerable to attacks. Moreover, adversaries are using more advanced techniques and sophisticated equipments for mounting attacks. This requires novel security solution with higher ability than before to protect a key.

1.2 What can a Physical Unclonable Function do?

A Physical Unclonable Function (PUF) offers promising solution to this problem. An on-chip Physical Unclonable Function is a chip-unique, hardware challenge-response function. Its challenge-response relationship is determined by deep sub-micron-level variations in the logic and interconnect of an integrated circuit(IC) chip. This variation, known as manufacturing process variation, is caused by uncontrollable deviations in the chip manufacturing process. The following properties of a PUF makes it a potential security solution.

- **Random identifier/key generation** - The imprint of process variation remains static in the post-fabrication phase of a chip, but it varies randomly from one chip to another manufactured from the same mask. This can be exploited to generate random yet unique device identifiers or random keys.
- **Secure and low-cost key storage** - A PUF key is generated on demand and does not exist in the powered-off state of a chip, leading to better security. Additionally, standard circuit design techniques can implement a PUF without special fabrication technique as required by non-volatile memory such as flash memory.
- **Unclonability** - The complex and random nature of process variation makes a PUF very hard to be cloned even if one has the original mask of the PUF design.
- **Tamper Resistance** - Physically invasive attacks aimed at revealing the PUF secret

are most likely to destroy the sensitive process variation imprint thus destroying the PUF itself. This makes a PUF a tamper-resistant solution.

Many useful applications for PUFs have been proposed so far. For example, they can be used in device authentication and secret key generation [45]. Guajardo et al. discussed the use of PUFs for Intellectual Property (IP) protection, remote service activation, and secret-key storage [13]. A PUF-based RFID tag has also been proposed to prevent product counterfeiting [2, 10].

1.3 Problem Statement

Even though a PUF shows great potential to solve several security-related problems, the following questions are critical to be answered for a PUF to become an efficient security primitive.

- First, the cryptographic strength of an encryption key depends on its entropy or randomness. How much randomness does a PUF-generated key contain? What are the factors that affect its randomness? How can we maximize the randomness of a PUF?
- Second, are the PUF keys reproduced with a consistent value every time they are generated? Do they remain the same when operating conditions change (such as temperature, supply voltage) so that it can be reliably used in an application? How can we improve the reliability of PUF-generated keys?
- How easy or difficult is it to attack a PUF and reveal its key? How can we make a PUF more robust?

These three quality factors, randomness, reliability, and robustness, determine the efficiency of a PUF as a secure primitive. There are unwanted effects that degrade these three quality factors. For example, randomness decreases in the presence of systematic process variation.

Circuit-level noise and metastability affect the reliability of a PUF. It is a challenging task to build an efficient PUF that can tolerate these effects and can exhibit strong PUF qualities.

Moreover, while enhancing the quality factors, one has to keep in mind that area, performance, and power consumption in a PUF also need to be optimized. For example, in a resource-constrained embedded application, area and power are among the main design constraints for a designer. Finding a sweet spot both in terms of improved PUF qualities as well as optimized resource is not trivial.

Besides the quality factors and the resource optimization, another important aspect that needs to be addressed is the PUF characterization. A PUF is supposed to be instantiated across a population of chips. Hence, the efficiency of a PUF is required to be validated over a sizable group of chips. As a part of the characterization, a PUF also needs to be tested against variations in operating conditions such as varying temperature and fluctuating supply voltage. The effect of circuit aging on PUF should also be studied.

Additionally, several different PUFs have been proposed so far in the literature. Since the first PUF proposal by Pappu et al. in 2001 [38], there has been at least a new PUF proposal every year on an average. However, there is no commonly accepted method that can fairly compare these PUFs in terms of their performance in order to select the most suitable one for an application. Therefore, a method is needed for comparing different PUFs.

These are some of the major problems existing in the area of PUF research. In this research, we have focused to solve these problems.

1.4 Contributions of this Research

The main contributions of this research can be summarized as follows.

- We propose a systematic approach to PUF design by introducing a PUF system model. The system model is composed of three components: sample measurement, identity

mapping, and quantization. We show that the three PUF quality factors can be improved by tuning the individual component of the system model in order to achieve an overall improvement of a PUF. We present three techniques to improve randomness, reliability, and robustness of a PUF at different components of the system model.

First, we show how systematic process variation adversely affects PUF randomness and robustness and present a solution to mitigate the effect. Second, we present a configurable ring oscillator technique that enhances the reliability of a PUF while using minimum area. Finally, we solve the problem of challenge-response pairs (CRPs) vs area of a PUF by proposing an identity mapping function along with its hardware implementation. Our proposed techniques are supported by experimental results from on-chip implementations.

- We formulate a large-scale experiment to characterize a PUF and validate its efficiency by estimating its performance over a large group of chips. A compact and portable experiment measures a sizable sample set having nearly 200 chips. We also tested the PUF against varying temperature as well as varying supply voltage. Moreover, a thorough study of the aging effect on the PUF has been done with the help of on-chip experiments as well as simulations.

- We propose a systematic method to evaluate and compare the performance of different PUFs. We first propose three generic dimensions of PUF measurements. We then define several parameters to quantify the performance of a PUF along these dimensions. We also analyze existing parameters proposed by other researchers. Based on our analysis, we propose a compact set of parameters that will be used as a tool to evaluate as well as compare the performance of different PUFs.

- Finally, we present a novel PUF exploiting the variability existing in a microprocessor pipeline to uniquely identify the microprocessor chip. We call it a microprocessor-intrinsic PUF. The PUF accepts a microprocessor instruction as a challenge and produces the delay in a data path or a control path in the microprocessor as the response.

We demonstrate our proposed idea using an instruction set implemented in a 32-bit microprocessor. This PUF has a very low area cost and can be easily integrated using software programs in an application having a microprocessor.

The remainder of this dissertation is organized as follows. Chapter 2 presents background material on PUFs. We also discuss the quality factors of PUFs in detail and define few parameters to quantify the quality factors. These parameters will be used to explain the experimental results subsequently. In Chapter 3, we introduce the PUF system model that will be used as a basis to introduce several PUF enhancement techniques that we proposed in this research. We also present a brief overview of the ring oscillator PUF (RO-PUF) that will be used to demonstrate the idea of the system model as well as the proposed PUF enhancement techniques. In Chapter 4, we describe three techniques that enhance the quality factors of a PUF. In Chapter 5, we discuss the characterization of a PUF using a large group of chips. The study on the effect of aging on a PUF is also presented in this chapter. The method to compare different types of PUFs is presented in Chapter 6. Our new PUF proposal, the microprocessor-intrinsic PUF along with its implementation results, is presented in Chapter 7. Finally, we make conclusions about this research work and present an outline of the future work in Chapter 8. We discuss several possible directions in which the current research work can be extended.

Chapter 2

Background

In this chapter, we first define a PUF and then discuss different types of PUFs that have been proposed so far. Since this research work is based on on-chip PUFs, we discuss variability in integrated circuits (ICs). Finally, several parameters measuring the quality factors of a PUF are discussed in detail. This chapter will help in understanding several concepts presented in later chapters.

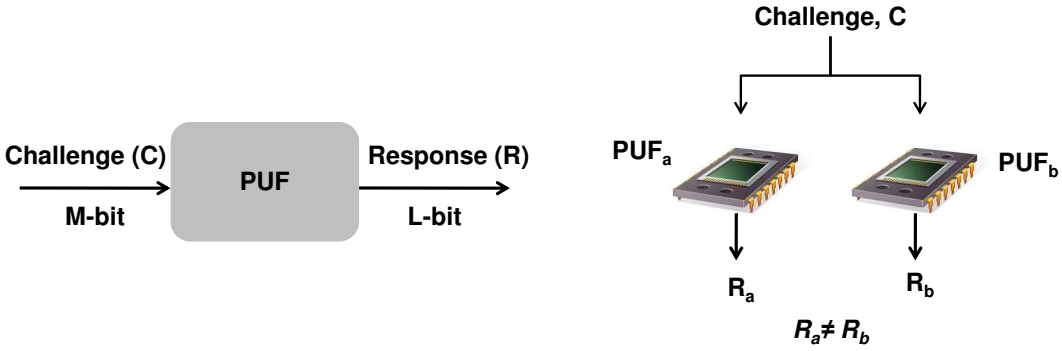

Figure 2.1: Basic functionality of a PUF

2.1 Definition of a PUF

The basic functionality of a Physical Unclonable Function (PUF) is to generate an L-bit response (R) upon receiving an M-bit challenge (C) (Figure 2.1). The challenge-response mapping is determined by complex structural disorder of a physical material. Though in the introduction, we defined a PUF based on deep sub-micron variability in a silicon chip, a PUF can be constructed using non-silicon material as well. We will describe different types of PUF in the next section.

Although a PUF is named as a function, it differs from the definition of a conventional mathematical function. A mathematical function is deterministic in nature and always generates a fixed output if the input remains the same. On the contrary, in a PUF, the challenge-response mapping changes from one instance of a PUF to another stochastically. This means that a fixed challenge C, when applied to two PUF instances PUF_a and PUF_b, produces responses R_a and R_b respectively when $R_\text{a} \neq R_\text{b}$ with high probability (Figure 2.1). This is because the challenge-response relationship is unique to a physical instantiation of the PUF, and two PUF instances are hardly alike.

Since the challenge-response relationship is determined by the structural disorder of a physical material (such as silicon in a chip), it is a *physical function*. It is *unclonable* because it is very difficult, if not impossible, to build an exactly identical copy of a PUF instance. Though no successful physical cloning has been reported thus far, modeling it mathematically has been shown [39].

Functionality of a PUF can be broadly divided into two parts: enrollment and evaluation. During the *enrollment*, a PUF is characterized to extract a reference response R_ref corresponding to a challenge C. The challenge-response pair $\{C, R_\text{ref}\}$ is stored in a secure database by an authorized entity. After the enrollment, the PUF is deployed in an application where, for example, it may serve the purpose of device authentication or may be used as a key in a cryptographic operation. During the *evaluation*, the PUF is provided with the

challenge C, and generates a response R. If $R_{\text{ref}} = R$, a device is authenticated successfully or an encryption/decryption occurs.

2.2 Types of PUF

One of the seminal works in the area of PUF is that of Lofstrom et al. in 2000 exploiting mismatch in silicon devices for identification of ICs [27]. Though the authors did not call it a PUF, the objective of their work was very similar to that of a PUF. Around the same time in 2001, Pappu et al. presented the concept of physical one-way function which led to the idea of a PUF [38]. After that, several PUF techniques have been proposed. One may refer to the work by Maes et al. that presents a comprehensive discussion on different PUFs proposed so far [31]. PUFs can be broadly classified into two types depending on the physical material used to construct it: a) non-silicon PUFs b) silicon PUFs.

2.2.1 Non-silicon PUF

In this type, the structural disorder of non-silicon material is exploited to build a PUF. For example, in the physical one-way function, random speckle pattern of an optical medium due to incident laser light is used to build challenge-response pairs (CRPs) [38]. In a coating PUF, a protective coating consisting of material doped with randomly distributed dielectric material is added on the top of the passivation layer of a chip. An array of capacitive sensors is embedded in the top metal layer to measure the capacitance of the coating. When voltage is applied between the plates of the sensors as a challenge, the capacitance due to the coating layer measured by the sensors is obtained as a response [47]. In this category, there are several other PUFs such as acoustical PUF [50], Magnetic PUF [20], paper PUF [6, 7], RF PUF [8], and CD PUF [15]. Though many of them are not termed as PUF by their inventors, their functional behavior resembles closely with that of a PUF.

2.2.2 Silicon PUF

On the other hand, in a silicon PUF, manufacturing process variation in the logic and interconnect inside a chip is exploited to derive the PUF CRPs. We have already introduced the term on-chip PUF. Though a non-silicon PUF, like the coating PUF, can be implemented in a chip, in this research we mean on-chip PUF as silicon PUF and use these two terms interchangeably.

In an SRAM PUF, random power-up values of SRAM cells are used to construct response bits [14]. A similar idea has been presented by Holcomb et al [17]. Su et al. implemented a custom-built circuit to generate identifier of a chip [43]. Maes et al. used the power-up values of D flip-flops (D-FFs) in FPGAs to construct a PUF [29]. The Butterfly PUF has been proposed to emulate the behavior of the SRAM PUF on an FPGA platform by cross-coupling two latches and creating a race condition using the set-reset inputs of the latch pair [22]. In an Arbiter PUF, delay mismatch between a pair of identical interconnecting wires due to process variation is used to generate PUF responses [24]. All these PUFs readily produce binary response bits.

On the other hand, there are PUFs that produce real-valued quantities as their responses and a quantization method converts them into binary response bits. In a ring oscillator PUF (RO-PUF), frequencies of an array of identically laid-out ring oscillators are measured as a set of real-valued data. Due to process variation, they vary randomly from each other and are subsequently converted to binary responses [45]. Helinski et al. introduced the idea of a PUF that exploits the variation in the equivalent resistance of the on-chip power distribution grid [16].

In this research, we work only on silicon PUFs. The next section discusses on-chip variability that is exploited to build the PUF CRPs in a silicon PUF.

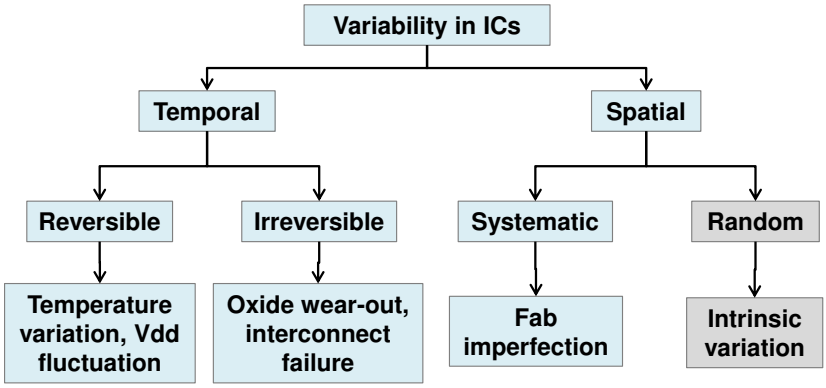

Figure 2.2: Variability in ICs

2.3 Variability in Integrated Circuits

An on-chip PUF exploits variability in the logic and interconnect inside an IC. Figure 2.2 shows different types of variability in an IC. It is mainly of two types: temporal and spatial. Temporal variability is caused during the operating phase of a chip. It can be reversible or irreversible. The spatial variability is introduced in a chip during the chip manufacturing process. This variability could be either systematic or random.

A PUF exploits the spatial random variability to form its CRPs. It is caused by sources such as random concentration of dopants in transistors and variation in gate oxide thickness [40]. These are intrinsic to the silicon material and cannot be controlled. This type of variability is the main cause for producing random responses in a PUF.

Other types of variability adversely affect the quality of a PUF. Systematic spatial variability which is caused by imperfections in the fabrication process reduces the randomness of a PUF. Temporal reversible variability caused by thermal effect, temperature variation makes PUF responses noisy and unreliable. Temporal, irreversible variability or aging also affects the reliability of a PUF.

In this research, we focus on maximizing the effect of spatial random variability while reducing the effect of other types of variability.

2.4 PUF Quality Factors

As a secure hardware primitive, a PUF is required to generate random yet chip-unique responses. At the same time, reliability of the PUF responses under varying operating conditions is crucial. Additionally, it must be robust against attacks that aim to disclose its secret. In these section, we discuss these three quality factors in detail. We introduce several parameters to quantify the quality factors of a PUF.

In Chapter 6, we will introduce additional parameters proposed by other researchers to evaluate and compare the performance of different PUFs.

2.4.1 PUF Randomness

We estimate the randomness of a PUF by three parameters: uniqueness, uniformity, and bit-aliasing.

1. **Uniqueness** - With a pair of chips, i and j ($i \neq j$), each having an n-bit response, R_i and R_j respectively for a challenge C, the average uniqueness among a group of k chips is defined as:

$$Uniqueness = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{HD(R_i, R_j)}{n} \times 100\% \tag{2.1}$$

   HD stands for Hamming distance. This is an average of all possible pairwise HDs among k chips. This expression is an estimate of the inter-chip variation in terms of PUF responses and not the actual probability of the inter-chip process variation.

2. **Uniformity** - Uniformity of a PUF estimates how uniform the proportion of '0's and '1's is in the response bits of a PUF. We define uniformity of a PUF as the percentage

Hamming Weight(HW) of its n-bit response:

$$(Uniformity)_i = \frac{1}{n}\sum_{l=1}^{n} r_{i,l} \times 100\%$$

where $r_{i,l}$ is the $l-$th binary bit of an $n-$bit response from a chip i. (2.2)

3. **Bit-aliasing** - In bit-aliasing, different chips produce nearly identical PUF responses for the same challenge which is an undesirable effect. It reduces uniqueness resulting in false positives in chip authentication. We estimate bit-aliasing of the l-th bit in the PUF response as the percentage Hamming Weight(HW) of the l-th bit of the response across k devices:

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

where $r_{i,l}$ is the $l-$th binary bit of an $n-$bit response from a chip i. (2.3)

For truly random PUF responses, all three of these parameters should converge to a value of 50%. However, in practice, a PUF may not produce truly random response bits. There are several factors that may contribute to make a PUF response biased. For example, systematic process variation tends to reduce the uniqueness of a PUF.

2.4.2 PUF Reliability

Though the PUF responses are expected to be static, factors such as temperature variation, supply voltage fluctuation, thermal noise makes them unsteady, and thus affect the reproducibility of PUF responses. Reliability quantifies the reproducibility of PUF responses over varying operating conditions.

A PUF can also be affected by temporal degradation i.e. aging of a chip. Effects such as hot carrier injection (HCI), electro-migration (EM), and negative bias temperature instability (NBTI) may cause irreversible changes in the chip structure over time. It is important to

assess how PUF behaves in the face of aging. A detailed analysis of the aging effect on PUF is discussed in Chapter 5.3.2.

To estimate the PUF reliability, we evaluate two parameters.

1. **Reliability using average intra-chip Hamming Distance(HD)** - We extract an n-bit reference response (R_i) from the chip i at the normal operating condition (at room temperature and with regular V_{dd} of the chip). The same n-bit response is extracted at a different operating condition (different ambient temperature or different supply voltage) with a value R'_i. m samples of R'_i is taken for each of the operating conditions. For the chip i, the intra-chip HD is estimated as follows.

$$HD_{\text{intra}} = \frac{1}{m}\sum_{t=1}^{m}\frac{HD(R_i, R'_{i,t})}{n} \times 100\% \tag{2.4}$$

   where $R'_{i,t}$ is the t-th sample of R'_i. HD_{intra} indicates the average number of unreliable/noisy PUF response bits. Hence, a lower value of it results in higher PUF reliability. The reliability of a PUF is be defined as:

$$Reliability \;=\; 100\% \;- HD_{\text{intra}} \tag{2.5}$$

2. **Distinct Unreliable Bits** - To estimate the worst-case reliability, we also estimate the total number of distinct unreliable bits over the whole range of varying operating condition. This is the total number of distinct bit positions in a PUF response that flipped at least once out of the total number of times a PUF response has been sampled.

   For example, suppose in one sample measurement of a PUF, bit a, b and c out of a 100-bit long response flipped. In another sample measurement of the same PUF, bit b, c and d flipped. In both the cases, the intra-chip HD is 3%. However, the total percentage of distinct unreliable bit is 4% (includes a, b, c and d).

2.4.3 PUF Robustness

Robustness is the ability of a PUF to prevent an adversary from revealing its secret. No parameter is defined to estimate this quality factor. Instead, we assess it qualitatively. We discuss a set of possible attacks that can be attempted against a PUF.

1. **Active attack** - This attack can be physically invasive or non-invasive. In an invasive attack, a chip is delayered or probed using advanced devices such as focused ion beam or using specialized chemicals. It is widely believed that this type of attacks tend to change the internal configuration of a chip permanently and hence destroy the PUF. As a result, the PUF secret is not revealed making it tamper-resistant. However, in practice, no such attack has been carried out on a PUF thus far to confirm this property.

   On the other hand, non-invasive attacks do not cause permanent change in PUF structure, but they make the PUF mechanism unstable by disturbing the operating condition of a PUF. For example, change in ambient temperature, supply voltage or noise injection in the power line can make a PUF produce a wrong key.

2. **Passive attack** - A PUF may be attacked passively by using side-channel information such as power consumption or electromagnetic radiation emanated from a chip containing a PUF. It is interesting to study how a PUF can withstand a side-channel attack.

3. **Replay attack** - An attacker can copy a challenge-response pair of a PUF during an authentication process and use it later on to fake the original PUF. It can be avoided by using a particular CRP only once, but it requires a large number of CRPs to be generated from a PUF so that a chip can be authenticated significantly many times before it runs out of CRPs. However, generating a large number of CRPs using limited circuit resources is a challenge.

4. **Cloning attack** - In a cloning attack, an adversary may attempt to build a physical copy of the original PUF with identical challenge-response behavior. However, in practice, no successful cloning attacks on PUF has been reported so far. One of the reasons being one has to manufacture a large number of chips using the same layout mask, and has to measure the complex delay characteristics of each of them to find out any close matching. Given the fact that time and money are limited, and that we have no control over the manufacturing variation, it is reasonable to believe that it is extremely difficult to successfully clone a PUF.

5. **Modeling attack** - Creating a model of a PUF mathematically is another security threat. A successful modeling attack can replace a legitimate PUF and steal sensitive information or get access to restricted resources. Successful modeling attack on delay-based PUFs has been reported by Ruhmair et al [39]. Producing truly random PUF responses is important to thwart a modeling attack. Therefore, the PUF randomness has a close relation with the robustness a PUF.

2.5 Summary

In this section, we presented a definition of a PUF and discussed different types of PUF proposed so far. We also defined several quality factors of a PUF. Different types of on-chip variability and their sources have also been discussed.

In the next section, we propose a PUF system model that enables a designer to split PUF operation into different components and show that improving each of the components of the system model has an overall positive impact on the PUF quality.

Chapter 3

PUF System Model

In an on-chip PUF, capturing the process variation information in terms of parameters such as delay and threshold voltage, and the subsequent quantization to produce binary responses are two separate components. This may allow a designer to tune each component separately to optimize the overall PUF design. However, it requires a well-defined partitioning of the PUF. We propose a generic PUF system model which clearly divides a PUF into several key components. The proposed system model can be used as a generic template for designing a PUF as it is defined independent of any particular PUF technique. Figure 3.1(a) shows the proposed PUF system model. It divides a PUF into three different components: sample measurement, identity mapping, and quantization.

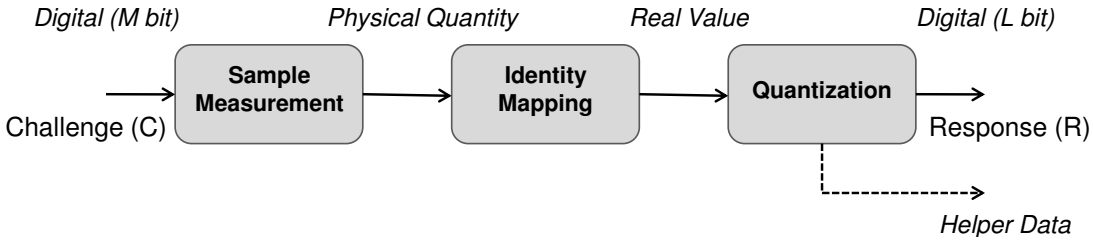

Figure 3.1: PUF system model

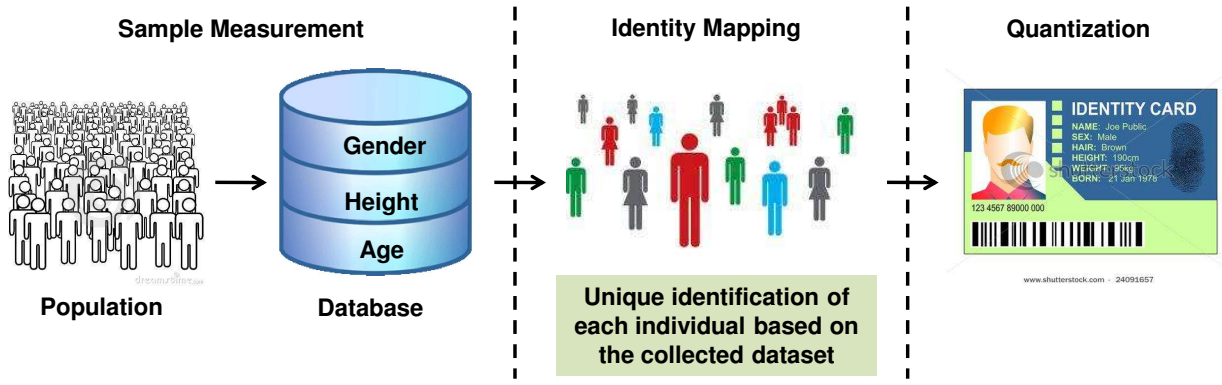

Figure 3.2: Analogous example of PUF system model

3.1 An Analogy to the PUF System Model

We first present an example analogous to the idea of the PUF system model. Suppose we are building an identification database for the population in a city to assign a unique identifier to each citizen. This process has close similarities with the PUF system model (refer to Figure 3.2).

During the sample measurement, for the citizen database, we collect several types of data about an individual such as gender, age, height, complexion, hair color and store them in a database (Figure 3.2). Similarly, using a PUF circuit, we collect process variation information from each chip in a population in terms of a physical measurement such as delay.

In identity mapping, analysis is done to map the collected dataset to each individual in such a way that every individual in the database can be distinguished from each other clearly. For example, it is important to distinguish two individuals if their records are alike. In the PUF context, the collected variation information is analyzed to uniquely identify a group of chips. We derive a suitable decision variable that distinctly captures the identity of each chip using the data collected in the sample measurement.

Finally in quantization, a unique identifier code is assigned to each of the citizen. Different methods can be applied for this purpose such as random identifier assignment or constructing

an identification code based on the collected data. For example, in a biometric application, an individual is assigned an identification code based on her biometric properties such as voice, fingerprint, and iris pattern. Protecting the privacy of an individual is of paramount importance in this phase. Similarly, in a PUF, the decision variable generated in identity mapping is converted into a binary string using a transformation function. The resultant string is used as a device identifier or a key. The transformation function needs to produce uniformly distributed response bits such that reverse engineering of a PUF response or prediction of unknown responses is not possible.

3.2 Definition of the PUF System Model

We discuss each component of the system model in detail in this section.

- **Sample Measurement:** In the sample measurement, an M-bit challenge (C) is applied to the PUF, and a set of physical measurements are obtained. This set of measurements contain the process variation imprint of a chip, and hence its distribution is expected to vary from one chip to another. In other words, this set represents the identity of a chip. Let us define the set of measurements as:

$$X = \{x_1, x_2, ..., x_I\} \tag{3.1}$$

  As an example, the set X can be a collection of frequencies from an RO PUF [45] or a set of variable resistances in a power PUF [16]. We split an individual measurement x_i into three parts.

$$x_i = x_{\text{avg}} + \Delta x_{i,\text{pv}} + \Delta x_{i,\text{noise}} \tag{3.2}$$

x_{avg} is the average value of x, and it is constant for all $x_i, 1 \leq i \leq I$ in a chip. $\Delta x_{i,\text{pv}}$ is the deviation due to the process variation (pv) and varies across $x_i, 1 \leq i \leq I$. $\Delta x_{i,\text{noise}}$ is the variation due to noise. We assume that it is a zero-mean random variable.

Ambient temperature variation, supply voltage fluctuation, and thermal noise negatively affect the sample measurement resulting in noisy PUF responses. Additionally, factors like metastability, systematic process variation can also degrade X. For example, an Arbiter PUF suffers from metastability in its arbiter flip-flop when the setup-hold time is violated [24]. Moreover, bit-aliasing in the PUF responses can occur due to systematic variation in ICs [17, 43].

Many of these issues can be addressed at the sample measurement itself to enhance the qualities of a PUF. For example, using averaging over multiple samples, the noise factor $x_{i,\text{noise}}$ can be reduced. This eventually simplifies a post-processing/error-correction method applied at any subsequent component of the system model. Several circuit-level as well as architecture-level optimization can be applied at the sample measurement [49, 57].

- **Identity Mapping:** The main objective of identity mapping is to produce a decision variable DV using X. An identity mapping function f_{IM} is defined that takes X as input and produces the DV. This decision variable is eventually used in the quantization component of the PUF to generate binary response bits.

$$DV = f_{\text{IM}}(X) \tag{3.3}$$

The set of DV represent the unique identity of a chip. Depending on the nature of f_{IM}, the size of the DV set varies, and as a result, the PUF response space also varies. This has an effect on the robustness of a PUF. An identity mapping function that is able to generate a large DV set leads to a large PUF response space. As a result, the robustness of the PUF against replay attacks (as discussed in Section 2.4.3) improves. Moreover, to prevent reverse engineering, the f_{IM} can be made non-linear in nature. This way, it becomes difficult for an attacker to reveal the raw value of the

sample measurements by analyzing the DVs. In brief, a careful design of the identity mapping is important to achieve a PUF with high quality factors.

- **Quantization:** The quantization is another key component of a PUF. In quantization, DVs are converted to binary response bits. There are different ways of implementing the PUF quantization. Suh et al. proposed a simple method of comparing a pair of frequencies in an RO PUF and deciding a binary bit depending on the outcome of the comparison [45]. Unlike the RO PUF, the SRAM PUF [14], the Arbiter PUF [24], and the Butterfly PUF [22] perform a direct time-to-digital conversion instead of a separate quantization step.

  Reconstruction of the PUF responses in the presence of noise and randomizing the responses to prevent reverse engineering attack can be implemented in the quantization. In the domain of information theory, they are known as information reconciliation and privacy amplification respectively. There are methods of quantization that have been proposed for this purpose. These methods are widely used in the field of biometrics. Dodis et al. proposed a fuzzy extractor to reproduce uniformly distributed keys from a noisy source of data [11]. Linnartz et al. proposed the Shielding function to create keys from biometric data such that the privacy of an individual is protected [26]. Most of these methods use a helper data that is used to reconstruct the original key. Hence, they are called helper data algorithms (HDAs). A helper data is public and leaks partial information about the secret key.

3.3 Ring-Oscillator PUF and the PUF System Model

In this research, we show the application of the proposed system model on a ring-oscillator based PUF (RO PUF). In the next chapter, we will present three enhancements of the RO PUF at different components of the system model. They are listed as follows.

- Improving the randomness and robustness of the RO PUF at the sample measurement

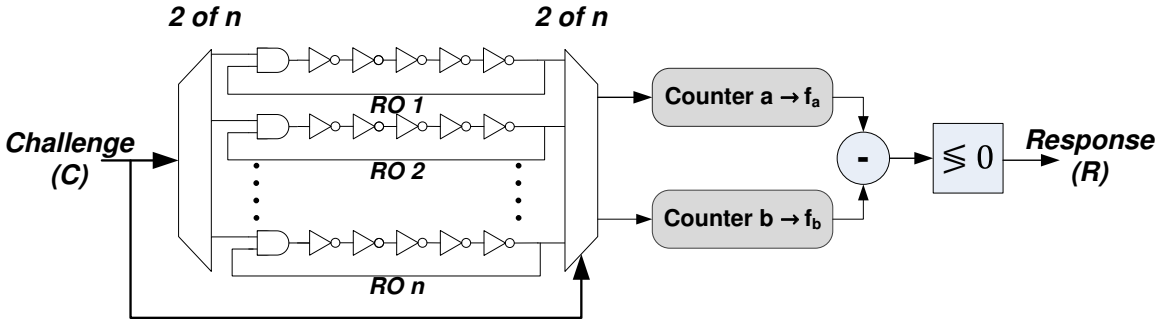

Figure 3.3: A Ring Oscillator PUF circuit

using a compact placement of the ROs and an adjacent pairing technique.

- Improving the reliability of the RO PUF at the sample measurement by replacing a regular RO structure with a compact configurable ring oscillator (CRO) structure.
- Enhancing the CRP set of the RO PUF using a new identity mapping function thus enhancing its robustness.

Before we continue to the next chapter, we introduce the RO PUF.

An RO PUF is composed of n identically laid-out ROs, RO_1 to RO_n (Figure 3.3). A pair of ROs out of the array of n ROs, are selected using multiplexers with the PUF challenge as the select bits of the multiplexers. Their frequencies, f_a and f_b (a $\neq$ b), are measured using a pair of counters. Due to process variation, f_a and f_b tend to differ randomly from each other. A response bit r_ab is produced by the quantization of two real-valued quantity, f_a and f_b, using a simple comparison method as follows -

$$r_\mathrm{ab} = \begin{cases} 1 & \text{if} \quad f_\mathrm{a} > f_\mathrm{b} \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

In Figure 3.4, an RO PUF is represented using the proposed system model. Measurement of RO frequencies belongs to the sample measurement. In the identity mapping phase, the difference of a pair of frequencies, f_a and f_b, is calculated as the DV. The sign and magnitude of the quantity $\Delta f = f_\mathrm{a} - f_\mathrm{b}$ differ from one chip to another due to process variation. Finally,

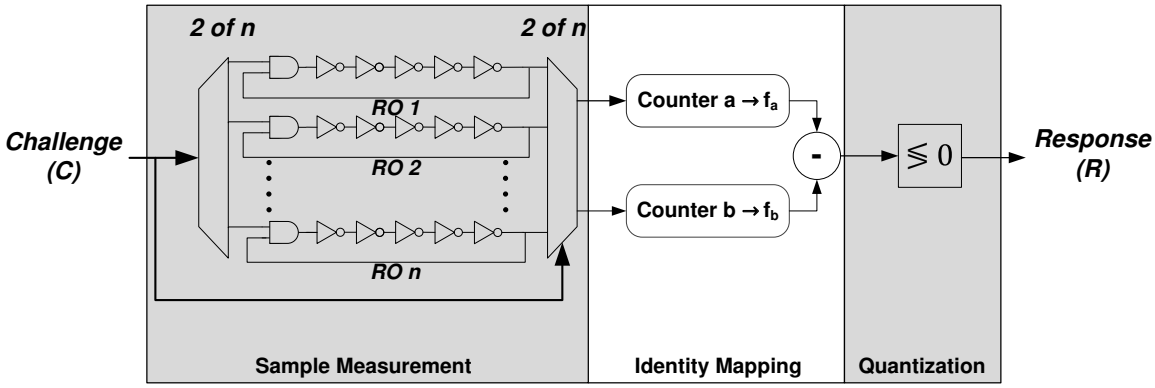

Figure 3.4: A system model representation of an RO PUF

in the quantization, a binary response bit is generated based on the sign of Δf following Equation 3.4.

3.4 Summary

In this chapter, we introduced the PUF system model. The model divides a PUF into three different components based on three distinct components of a PUF in general: measurement of circuit variability, mapping of variability information to an individual chip, and identifier formation. We also presented an analogous example to the idea of the proposed system model. We briefly described a ring oscillator PUF which will be used to demonstrate our proposed PUF enhancement techniques in the next section.

Chapter 4

PUF Enhancements

In this chapter, we present three enhancements in the RO PUF. First, we show how systematic process variation affects the randomness and the robustness of the RO PUF. A compensation technique is proposed at the sample measurement. Second, we propose a compact technique that enhances the reliability of the RO PUF at the sample measurement. Finally, we propose a technique that expands the challenge-response set of the RO PUF by post-processing RO frequencies at the identity mapping, thus enhancing the robustness of the PUF.

4.1 Effect of the Systematic Process Variation on PUF and its Compensation

On-chip spatial variation can be random as well as systematic. The random or stochastic variation is caused by atomic-level variations in semiconductor film thickness or line-edge roughness of layout structures. On the other hand, the systematic variation is caused by errors in the fabrication mechanism such as mask errors from process model inaccuracies and reticle stepper alignment errors.

PUF responses are truly random only if they are generated due to the random variation. However, with continuous scaling down of feature size of silicon devices, the systematic variation is becoming more prominent than before. We show that the systematic variation can cause bit-aliasing in PUF responses across different chips. This reduces the PUF uniqueness, resulting in false positives or ID-collision in device authentication. Su et al. showed an estimate of the probability of ID-collision that increases linearly with the size of the chip population for a fixed length of PUF response [43]. The systematic process variation can also help an adversary to predict an unknown PUF response.

We first establish the relation between the bit-aliasing and the uniqueness parameter in general. Next, we will show how the systematic variation affects the bit-aliasing in an RO PUF and thus affects the uniqueness of the PUF. Let us consider an arbitrary l-th bit in an L-bit response of a PUF ($1 \leq l \leq L$). If there are N chips implementing the PUF, there are N such l-th bits. Suppose, a fraction x ($0 \leq x \leq 1$) of these N bits are '1'. According to Equation (2.3), the bit aliasing for the l-th bit is $(x.N)/N = x$. Since, the number of bits with value '0' is $(1-x).N$, the uniqueness of the PUF based on the l-th bit is $x.N.(1-x).N/(N(N-1)/2) = 2.(N/(N-1)).(x(1-x))$ (Equation (2.1)). For a large value of N, we assume $N/(N-1) \approx 1$. Hence, the uniqueness becomes $2.x.(1-x)$. Therefore, the relation is:

$$\text{For the } l-\text{th response bit}, \text{Uniqueness} = 2.\text{Bit}-\text{aliasing}.(1-\text{Bit}-\text{aliasing}) \tag{4.1}$$

Figure 4.1 shows how the uniqueness varies with the bit-aliasing. The uniqueness reaches a maximum value of 50% when the bit-aliasing parameter is 50% i.e. when the number of '0's and '1's are equal at a bit position across a population of chips. On the contrary, when a bit produces identical values across the population, the uniqueness becomes zero.

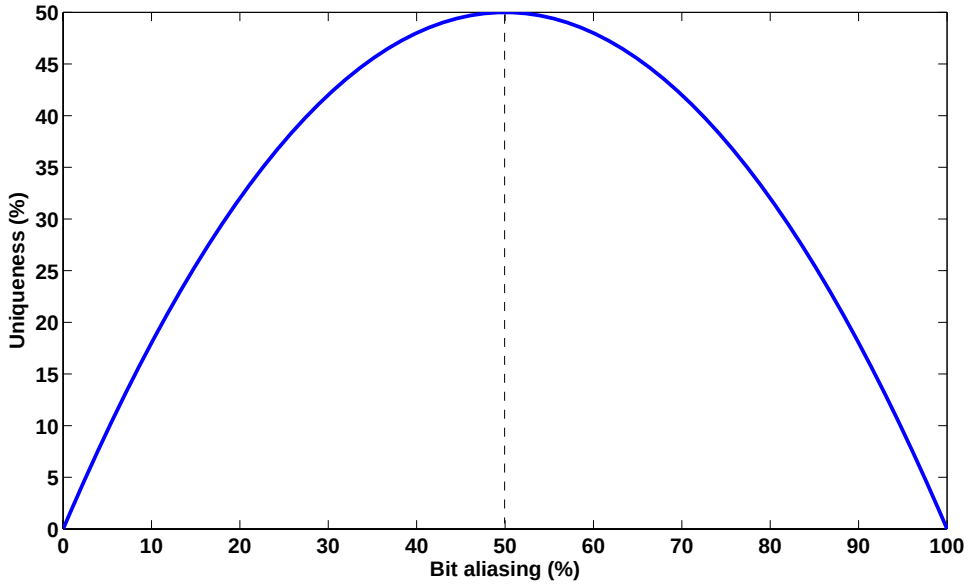

Figure 4.1: Uniqueness vs Bit aliasing

4.1.1 Effect of the Systematic Variation on PUF Uniqueness

In the RO PUF, the total delay in a ring oscillator loop can be represented as follows.

$$d_{\text{loop}} = d_{\text{avg}} + \Delta d_{\text{rand}} + \Delta d_{\text{syst}} \tag{4.2}$$

where d_{avg} = the nominal delay that is same for all identical ROs, d_{rand} = the delay variation due to the random process variation, d_{syst} = the delay variation due to the systematic variation. We ignore the noise component in delay for this analysis. The difference in d_{loop} between a pair of ROs, a and b, is determined as follows.

$$\Delta d_{\text{loop\_ab}} = (d_{\text{avg}} + \Delta d_{\text{rand\_a}} + \Delta d_{\text{syst\_a}}) - (d_{\text{avg}} + \Delta d_{\text{rand\_b}} + \Delta d_{\text{syst\_b}}) = \Delta d_{\text{rand\_ab}} + \Delta d_{\text{syst\_ab}} \tag{4.3}$$

Equation (3.4) shows how a single response bit r_{ab} using a pair of RO frequencies, f_{a} and f_{b}, is generated. Equation (3.4) can be expressed in terms of the loop delay as:

$$r_{\text{ab}} = \begin{cases} 1 & \text{if} \quad \Delta d_{\text{loop\_ab}} < 0 \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

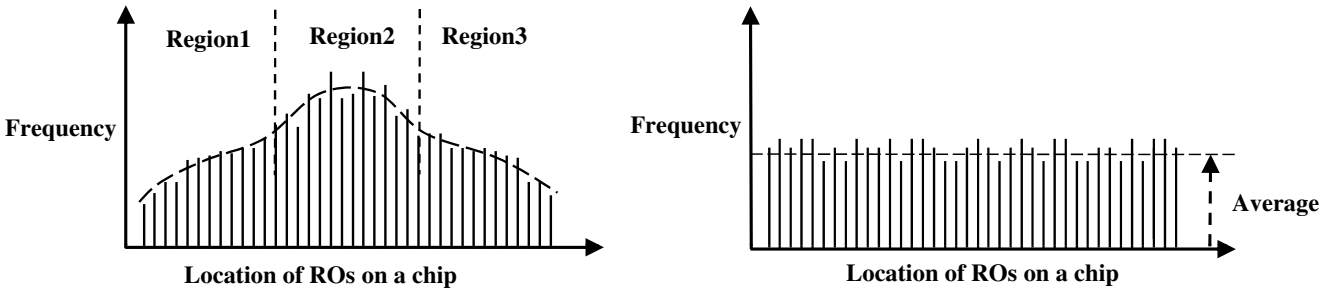

Figure 4.2: (a)Case with the systematic variation. (b)Case without the systematic variation.

Generation of r_{ab} can be modeled as a Bernoulli trial with the probability of success p as $Prob(r_{\text{ab}} = 1)$, and the probability of failure q as $Prob(r_{\text{ab}} = 0), (p + q = 1)$. For an L-bit PUF response, we have L such Bernoulli trials. From Equation (4.3) and (4.4), it is evident that the value of p is determined by $\Delta d_{\text{rand\_ab}}$ and $\Delta d_{\text{syst\_ab}}$. The outcome of the Bernoulli trial will be equally likely between '0' and '1' ($p = 0.5$) if $\Delta d_{\text{syst\_ab}} = 0$. Therefore, the relation between p and the systematic variation can be summarized as follows.

$$p = \begin{cases} 0.5 \ \text{ if } \Delta d_{\text{syst\_ab}} = 0 \\ 0.5 \pm \Delta p \ \text{ if } \Delta d_{\text{syst\_ab}} \neq 0 \end{cases} \tag{4.5}$$

where Δp is the bias introduced in the value of p due to the systematic variation ($0 < \Delta p \leq 0.5$). The systematic process variation can create a regular pattern in the delay/frequency values of ROs. This type of pattern shows a gradual change in the frequency as a function of the physical locations of ROs in a chip. Sedcole et al. mentioned such a pattern in the RO frequencies in a 90nm FPGA [40]. We use an arbitrary hypothetical example of such a pattern in RO frequencies as shown in Figure 4.2(a)[1] to explain how Δp can have a non-zero value in Equation (4.5).

In Figure 4.2(a), the frequencies follow a pattern caused by the systematic variation with respect to the locations of the ROs. A frequency comparison between an RO from the region 1 and that from the region 2 is more likely to produce a '0' PUF response (-ve Δp). The PUF is more likely to produce a '1' as response when a frequency comparison takes place

[1]These figures are representation of a conceptual idea, and not based on any real data.

	Bit-aliasing	Uniqueness
$u = N, v = 0$	100%	0%
$u = 0, v = N$	0%	0%
$u = v$	50%	50%

Table 4.1: The limits of the bit-aliasing and the uniqueness of a PUF

between an RO from the region 2 and the region 3 respectively (+ve Δp). In the example of Figure 4.2(b), no systematic-variation-induced pattern is found. As a result, the value of p for any pair of ROs is close to 0.5 in this case.

Suppose, there are N chips, each producing an L-bit PUF response. We consider any arbitrary bit position l out of the possible L positions. For N chips, there are N such l-th response bits. For pair-wise comparisons among N chips, there will be a total of $\binom{N}{2}$ $= N(N-1)/2$ comparisons for the l-th response bit. Out of the N bits, let us assume an arbitrary composition of bits biased due to the systematic variation:

$$\begin{aligned} u &= \text{number of response bits with } p = 0.5 + \Delta p, \quad (0 \leq u \leq N) \\ v &= \text{number of response bits with } p = 0.5 - \Delta p, \quad (0 \leq v \leq N) \\ k - u - v &= \text{number of unbiased response bits i.e. } p = 0.5, \quad (0 \leq u + v \leq N) \end{aligned} \tag{4.6}$$

We assume that half of the unbiased bits are likely to be '1', and the other half are likely to be '0'. Therefore, the total number of response bits with a likely value of '1' is $u+(N-u-v)/2 = N/2 + (u-v)/2$. Hence, the bit-aliasing for the l-th bit is :

$$\text{Bit} - \text{aliasing} = (N/2 + (u-v)/2)/N = 0.5 + (u-v)/2N \tag{4.7}$$

Combining Equation (4.1) and (4.7), we estimated the PUF uniqueness in terms of u and v. Table 4.1 shows the limits of the bit-aliasing and the uniqueness based on the values of u and v.

Now we discuss how u and v in Equation (4.6) are dependent on the systematic variation. The systematic variation in a chip may occur for several reasons. One of the main causes is

the die pattern or the layout dependency [37, 3]. A wafer scale variation such as variation in wafer thickness can also produce it. Though this variation is systematic inside a chip, it may be randomly distributed over a group of chips [3]. This case is more likely to create an even proportion of u and v for any bit position across N chips. This will result in a higher value of uniqueness according to Table 4.1. However, bit-aliasing will occur if the systematic variation creates a similar trend across a group of chips resulting in higher proportion of either u or v. Table 4.1 shows that when $u = N$ or $v = N$, the uniqueness becomes zero. Experimental results on $0.18\mu m$ technology reported by Onodera shows that similar pattern of systematic variation does exist across different chips [37]. In our experimental result section, we show that similar pattern of systematic variation is visible in the sample 90nm FPGAs used.

From a security point of view, if an adversary has the information about the pattern of RO frequency variation (as shown in Figure 4.2(a)), and she knows the on-chip locations of ROs, she may be able to predict unknown PUF responses with high probability. For example, if she can determine that a pair of ROs are located in different regions as shown in Figure 4.2(a), she will be able to predict the response generated out of the RO pair with high probability.

4.1.2 Proposed Solution to Compensate for Systematic Variation

Since modifying a chip is not possible, we can only minimize the effect of systematic variation. The logic and interconnect in close proximity are affected by systematic process variations in a similar way. Therefore, two closely located ROs will have similar Δd_{syst} in Equation (4.2) resulting in a very low value of $\Delta d_{\text{syst\_ab}}$ in Equation (4.3). As a result, $\Delta d_{\text{rand\_ab}}$ will be more likely the dominant factor. The end result is that the value of p, as described in Equation (4.5), will converge towards 0.5 i.e. the individual response bits will be more likely to be unbiased. This will reduce the value of u and v in Equation (4.6) thus increasing the uniqueness. To implement this strategy, we propose two steps.

- The group of ring oscillators are placed as close as possible to each other e.g. in a 2-dimensional array formation.
- While evaluating a responses bit, a physically adjacent pair of ROs are selected.

These two steps ensure maximum proximity of a pair of ROs used in creating a PUF CRP. Ideally, if one knows the distribution of the RO frequencies in a PUF, then it is possible to avoid the effect of systematic variation by analyzing the distribution. However, in a PUF with large number of ROs, this is a time-consuming and costly process. Moreover, it is impractical for fabrication, since each PUF would need to be measured and calibrated. The proposed method is very easy to implement, and it works always even if no information about the nature of the systematic variation is available. An additional advantage is that the scheme helps in reducing the effect of environmental noise in the frequency difference of an RO pair assuming closely located ROs will be subjected to similar environmental noise. A disadvantage of this method is that it restricts the maximum number of independent response bits that can be extracted from a PUF with L ring oscillators to $(L-1)$. This is, however, a pessimistic estimate. In practice, additional independent response bits can be extracted using other techniques.

4.1.3 Experimental Results

Experimental results, based on five Xilinx Spartan XC3S500E FPGAs, is presented in this section. In order to demonstrate the proposed method, we implemented the PUF circuit in two ways. In the *first* design, the placements of the ROs were decided by the place and route tool without any user constraint. After the placement and routing process, it was confirmed that the ROs are spread randomly over the FPGA. In the *second* design, with a placements constraint the ROs were aligned closely in a 2-D array fashion as shown in Figure 4.3. We follow three methods of extracting PUF responses to incrementally show the validity of the compensation method.

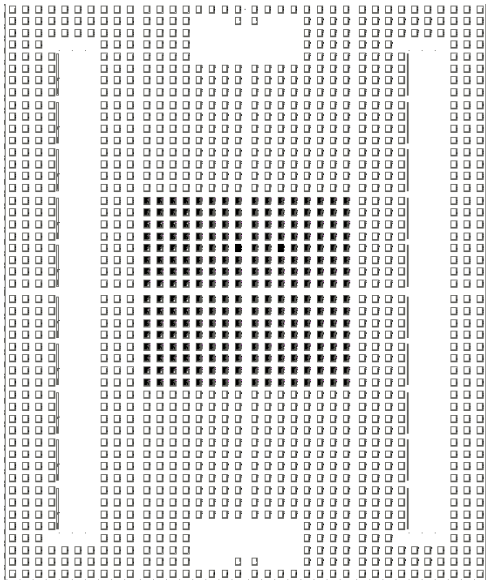

Figure 4.3: Sample array configuration of a PUF with 256 ROs on Spartan XC3S500E FPGA. Black boxes represent ROs.

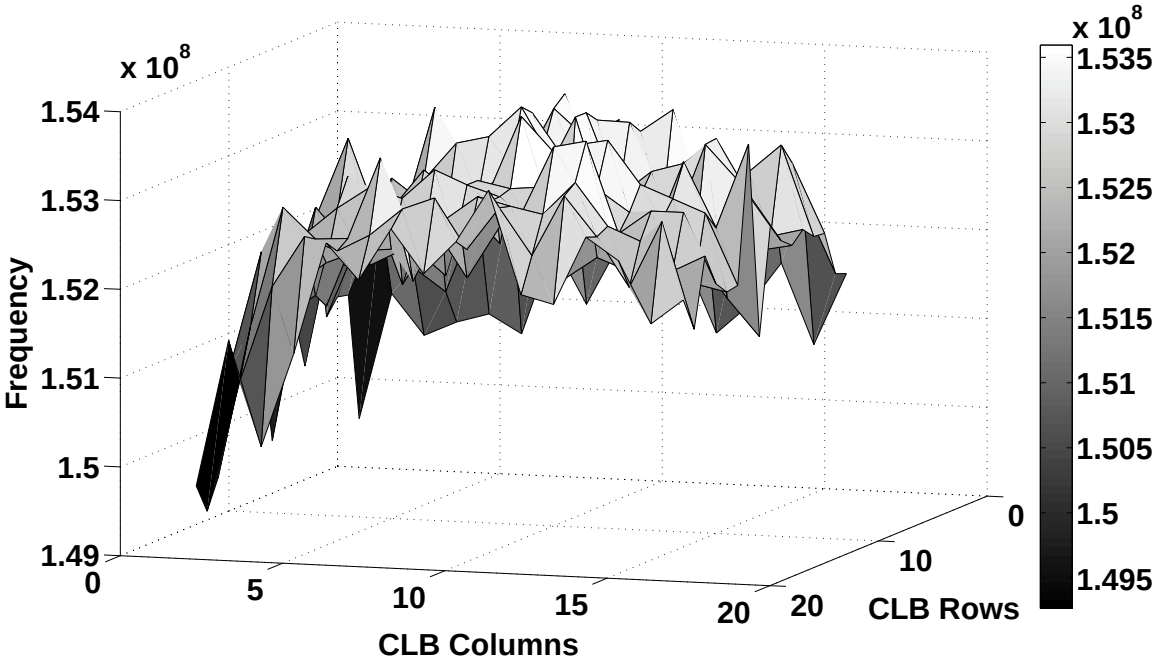

Figure 4.4: Distribution of the average frequency of ROs in a PUF with 256 ROs with controlled placement.

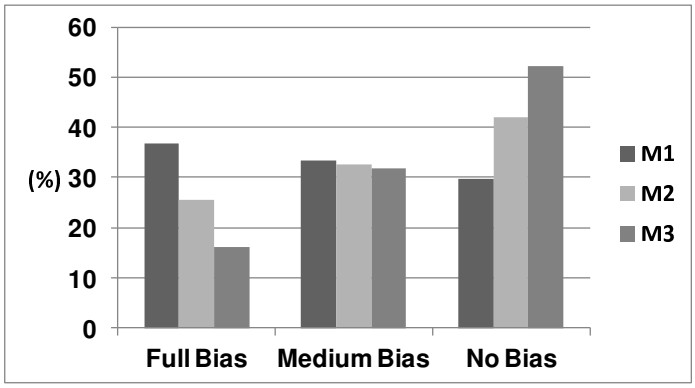

Figure 4.5: Bit-aliasing comparison

Method 1($M1$) - The PUF responses were generated from the first implementation that has the ROs spread across the chip.

Method 2($M2$) - The responses were generated from the second implementation. A pair of ROs are selected in such a way that they were not located adjacent to each other. This implements only the first step of the compensation method.

Method 3($M3$) - Physically adjacent RO pairs are selected for generating responses from the second implementation. This implements both the steps of our proposed method.

We validate our proposed technique by measuring the following parameters defined in Section 2.4.1.

1. Bit-aliasing - This is an estimate of the bias Δp (Equation (4.5)) introduced in the response bits due to the systematic variation. In our experiment, for 5 FPGAs ($N = 5$), we classified the bit positions in three groups:

   fully biased : all 5 bits are either 1 or 0

   medium biased : 4 bits are 1 and 1 bit is 0 or vice versa

   unbiased : 3 bits are 1 and 2 bits are 0 or vice versa

   Figure 4.5 show the distribution of fully biased, medium biased, and unbiased bit positions for a PUF with 256 ROs for each of the experimental method. It is evident

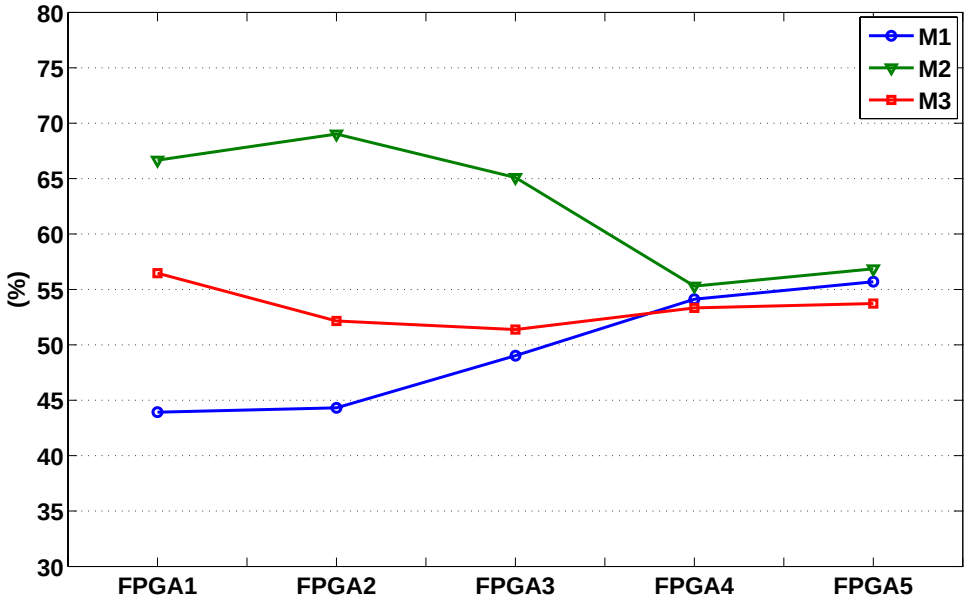

Figure 4.6: Uniformity of responses from a PUF with 256 ROs

that our proposed method enhances the proportion of the unbiased bit positions while reducing the proportion of the fully biased bit positions. For a PUF with 256 ROs, there are 255 bit positions. M2 yielded an increase of 11% in the unbiased bit positions compared to M1 while it yielded a decrease of 12% in the fully biased bit positions compared to M1. Similarly, M3 produced an increase of 22% in the unbiased bit positions compared to M1 while it yielded a decrease of 20% in the fully biased bit position compared to M1.

2. Uniformity - The results in Figure 4.6 show that M3 yields fairly consistent value of uniformity that is close to 50%. The other two methods produce fluctuating results deviating from 50%.

3. Uniqueness - We evaluated the uniqueness for three different sizes of PUF: 64 ROs, 128 ROs, and 256 ROs for each of the above methods $M1, M2$, and $M3$. For all the three PUFs, M1 produces the least uniqueness whereas M3 produces the highest value of the uniqueness. The net improvements between M1 and M3 are 18.09%, 9.6%, and 12.78% for 64, 128, and 256 RO settings respectively.

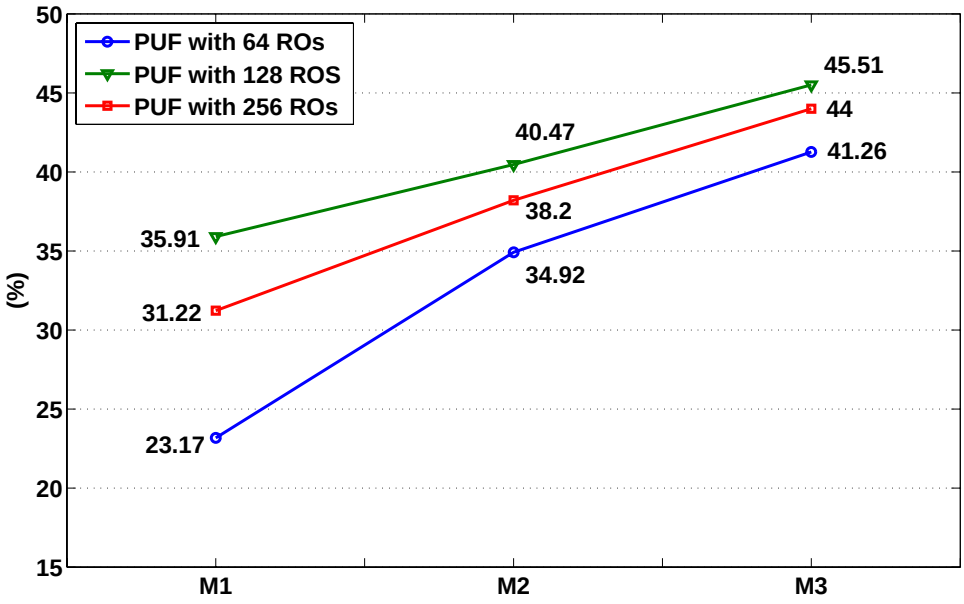

Figure 4.7: Average inter-chip HD improvement

4.2 Configurable Ring Oscillator (CRO) for Enhanced PUF Reliability

In an RO PUF, the reliability of a response bit, generated out of a pair of frequencies f_a and f_b, is dependent on the quantity $\Delta f = f_a - f_b$. A large value of Δf will lead to a highly reliable response bit. On the other hand, a pair of frequencies having a small Δf can lead to unreliable response bits in the presence of environmental variation such as temperature variation or supply voltage fluctuation. In this work, we propose a modified architecture of a ring oscillator loop that can maximize the value of Δf resulting in higher reliability in the RO PUF. This PUF enhancement technique is implemented at the sample measurement component of the system model.

4.2.1 Configurable Ring Oscillator (CRO) Technique

A redundancy scheme such as the majority voting or 1-out-of-k method is suitable for improving the reliability of an RO PUF. In a 1-out-of-k method, a response bit is generated by selecting a pair of ROs from a group of k ROs such that the frequency difference between the selected pair is maximum [45]. However, implementing such a scheme comes with the price of high area footprint. We show that redundancy can still be achieved by modifying the RO structure while avoiding the excess usage of area. We propose a new ring oscillator structure called the configurable ring oscillator (CRO). In a CRO structure, several 2:1 multiplexers is inserted in the RO loop to implement several loop configurations as shown in Figure 4.8. In a CRO with m muxes, 2^m distinct oscillator loops can be configured. In the circuit in Figure 4.8, we can configure eight different ROs using the control inputs C1, C2, and C3 of the three 2:1 multiplexers respectively. Applying the same control inputs to two different CROs, we can configure eight ring oscillators in each of the CROs. Hence, it is possible to evaluate eight frequency comparisons between two CROs instead of just a single comparison using a pair of regular ROs. Due to process variation, these eight pairs of RO frequencies are expected to have varying frequency differences. To achieve the maximum reliability, we can select the pair which has the maximum difference in frequency. We call it the most reliable pair. The selection scheme is illustrated in Figure 4.9.

The comparison is done between equal configurations among different CROs: only those ring oscillators will be structurally identical. The RO pair for which Δf is maximum is stored as the challenge during the PUF enrollment. For a pair of configurable ROs, CRO_{a} and CRO_{b}, creating a response bit r_{ab} with the multiplexer select input s (a 3-bit binary number with the MSB as C1,the 2nd bit as C2, and the LSB as C3) for configuring the most reliable pair ($0 \leq s \leq 7$), the challenge-response pair will be stored as : $r_{\mathrm{ab}} \longleftarrow (CRO_a, CRO_b, s)$.

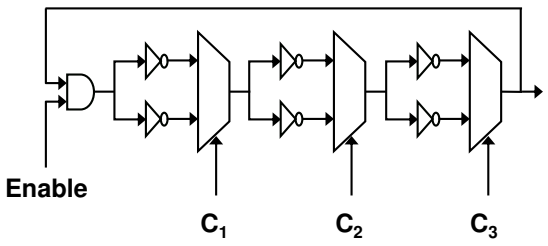

Figure 4.8: Configurable Ring Oscillator (CRO)

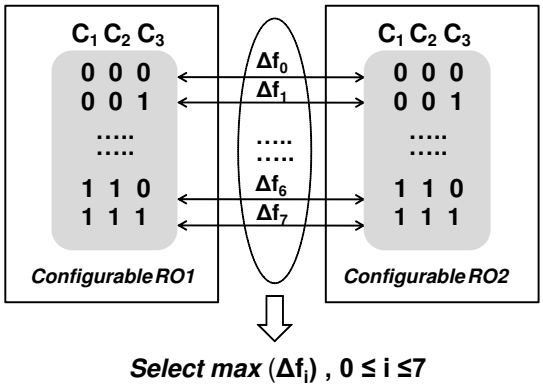

Figure 4.9: CRO pair with the maximum difference in frequency

4.2.2 Area Savings

We show savings in area using the proposed technique on both FPGA and ASIC paltforms. Figure 4.10 shows how the CRO can be mapped into a single CLB [2] of a Xilinx FPGA. This design consumes 7 look-up tables (6 for inverters, 1 for AND gate) and three dedicated multiplexers. An individual loop in the CRO contains seven delay stages (3 inverters, 3 MUXes, 1 AND gate). A regular RO loop with seven delay stages would require seven LUTs. Since a CLB on the used platform has eight LUTs, both the regular RO and the CRO consumes one CLB. Hence, the proposed CRO has no extra area cost. This technique can be applied to other FPGA platforms with similar logic resources.

In an ASIC context, we estimate the area in terms of gates to compare the CRO with the regular RO. A 7-stage regular RO requires six inverters and a NAND gate. The CRO consumes 6 inverters, 1 AND gate, and 3 MUXes (Figure 4.8). If we assume a standard construction of a 2:1 multiplexer using transmission gates, it is composed of 4 transistors that is equivalent to a NAND gate. Hence, it requires 6 inverters, 1 AND gate and 3 NAND gate equivalents.

To implement a 1-out-of-8 scheme using 8 separate regular ROs, the cost is 48 inverters

[2] A CLB is a unit on Xilinx FPGAs containing the resource for configuring logic implemented by users.

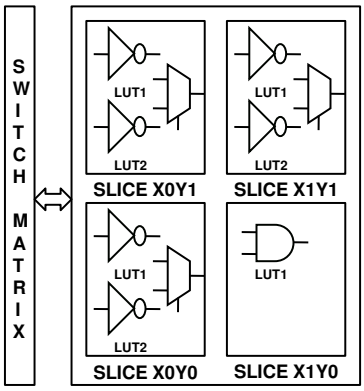

Figure 4.10: The mapping of a CRO in a single CLB on a Xilinx Spartan 3E platform

and 8 NANDs. To achieve the same reliability, a CRO only consumes 6 inverters, 1 AND gate and 3 NAND gates. Hence, the CRO is significantly more cost-effective than the regular RO on the ASIC platform.

4.2.3 Environment-adaptive Reliability Enhancement Technique

The CRO technique can be utilized to calibrate the PUF responses according to different operating conditions. The entire operating range of temperature and voltage can be divided into few regions. During the enrollment, the PUF will be characterized in each of these regions separately to find out the most reliable RO pair for a response bit. This environment-specific information can be stored in the CRP database. During a subsequent CRP evaluation, based on the current operating condition, the appropriate challenge will be used. If a pair of configurable ROs, CRO_a and CRO_b, create a response bit r_{ab} at temperature T_{REF} and supply voltage V_{REF} with the index s for the most stable pair ($0 \leq s \leq 7$), then the challenge-response pair will be stored as : $r_{ab} \longleftarrow (CRO_a, CRO_b, s, T_{REF}, V_{REF})$.

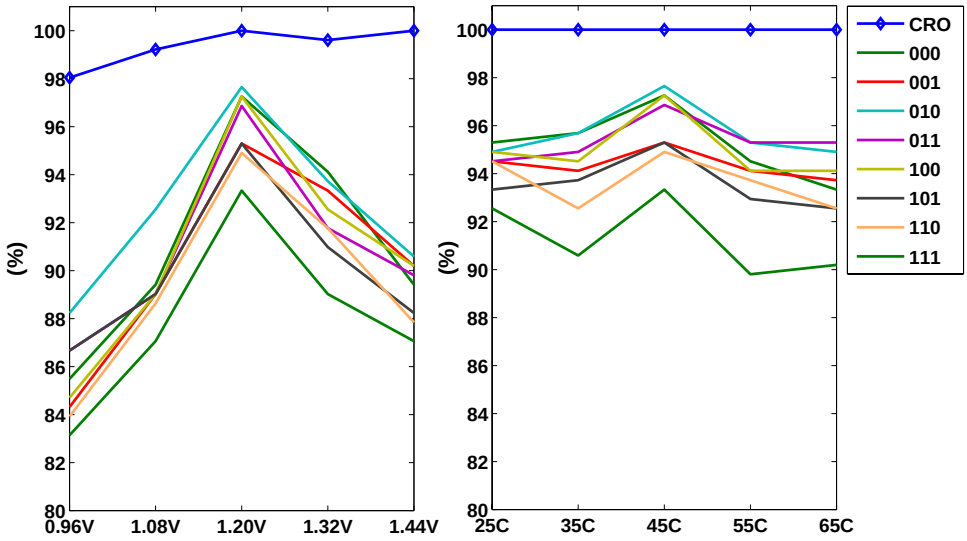

Figure 4.11: Reliability with varying voltage and temperature for a PUF with 256 ROs

4.2.4 Experimental Results

We present experimental results for the CRO technique based on an implementation using a Spartan XC3S500E FPGA. Experiments were carried out for temperature variation from 25^0C to 65^0C using a temperature controlled chamber. The core voltage of a Spartan XC3S500E FPGA was varied $\pm 20\%$ using a controllable DC power supply. We compare our proposed method against all eight individual RO configurations. This shows how much improvement we can achieve using the CRO technique compared to the situation with no configurable RO. Figure 4.11 shows the reliability for both voltage and temperature variation for a PUF with 256 ROs. We show nine cases. The 'CRO' shows the reliability using the proposed CRO technique. Each of the other results (shown by 3-bit indexes) show the reliability corresponding to a fixed configuration for all ROs specified by the 3-bit index. For example, the line corresponding to '001' shows the reliability of the PUF when all the 256 CROs are configured with C1=0, C2=0, and C3=1. C1, C2, and C3 are the select inputs of the three 2:1 multiplexers in the CRO loop as shown in Figure 4.10(a). With varying voltage, our proposed method has the highest reliability in all the cases with no unreliable bits at the normal operating voltage at 1.2V. For varying temperature, the CRO technique

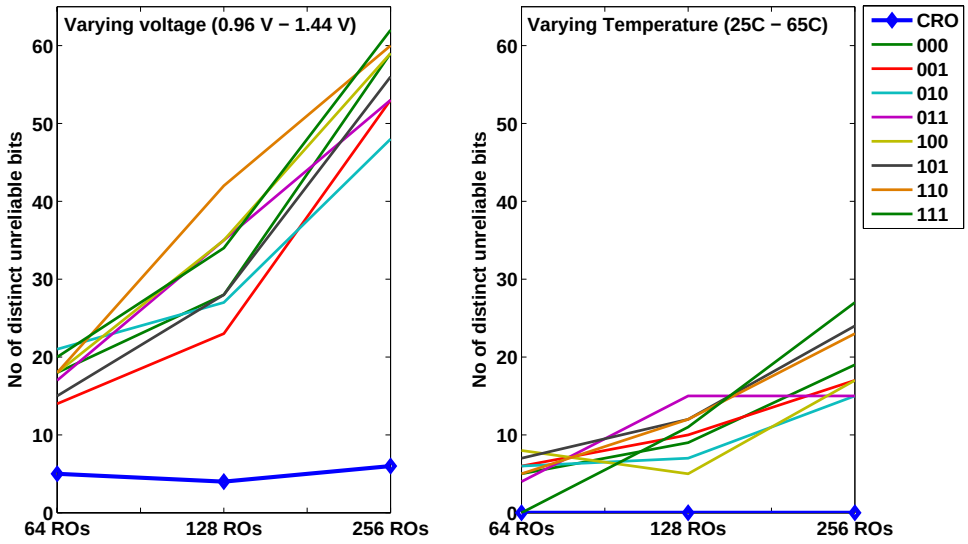

Figure 4.12: Distinct unreliable bits for different PUF settings

produces 100% reliable PUF responses in all the cases.

Figure 4.12 shows the number of distinct unreliable bits for three different sizes of PUFs: 64 ROs, 128 ROs, and 256 ROs. It is clear that the proposed method yields a result that is a distinct outlier with a consistently lower number of unreliable bits.

For the environment-adaptive technique, we enrolled the PUF at three different reference voltages of 0.96V, 1.2V, and 1.44 V. Since the CRO method yielded 100% reliable outputs for temperature variation, we do not enroll for different temperatures. Figure 4.13 shows that when the PUF is enrolled at a reference voltage of .96V, it produces no unreliable response bits at 0.96V and 1.08V. Enrollment at 1.2 V produces no unreliable response bits at 1.2 V and 1.32V while the same PUF enrolled at 1.44V produces no unreliable response bits at 1.32V and a single unreliable bit at 1.44V. This indicates that characterizing the PUF at different operating condition can significantly enhance the reliability.

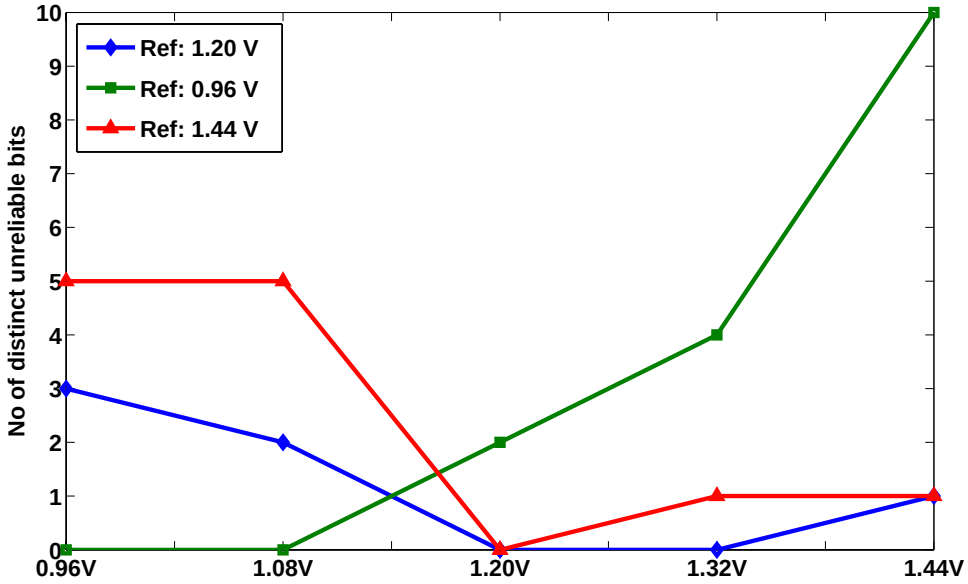

Figure 4.13: Environment adaptive reliability method using the CRO technique

4.3 A Compact Technique to Enhance CRPs of a PUF

In the previous two sections, we presented two techniques for enhancing the uniqueness, reliability, and robustness at the sample measurement component of the proposed system model. In this section, we propose a technique that mainly enhances the robustness of a PUF by expanding the size of its challenge-response set at the identity mapping component of the system model. A large set of CRPs from a PUF is essential for the following reasons.

1. A large set of PUF response bits lead to longer cryptographic keys, which may offer stronger cryptographic strength.

2. In a device authentication process, PUF CRPs are exchanged between the authentication authority and the device to be authenticated. Since the CRPs are public information, and since they are transferred over insecure channels, an adversary can capture and re-use this information. To prevent such a replay attack, a CRP should not be used more than once. However, this requires a large set of CRPs in order to authenticate a device a significant number of times before the CRP set is exhausted.

Moreover, a longer device identifier composed of many response bits can authenticate a bigger population of devices with less error.

3. A number of PUF response bits cannot be used reliably as they are erroneous or unstable in nature due to noise effects and environmental variations. To compensate for the error, additional response bits need to be generated.

The number of CRPs that can be extracted from a PUF is directly related with the amount of entropy present in it. In an information-theoretic sense, the amount of entropy in a PUF can be quantified as the total number of independent challenge-response pairs that can be derived from a PUF. By independent CRPs, we mean that by knowing a CRP i, one cannot guess or predict another CRP j even with a low probability i.e. the mutual information, $I(j;i)$ is zero, $i \neq j$. However, the amount of entropy in a PUF circuit is limited by the number of circuit components used to construct the PUF. For example, in an RO PUF, the source of entropy is a group of RO frequencies [45]. Hence, to produce a large number of independent CRPs, one may have to pay a price in terms of increased area cost.

In this work, we show that the source of entropy in a PUF can be processed through a new identity mapping function to increase the number of CRPs generated from a PUF. These CRPs, though not independent with each other in information-theoretic sense, exhibit strong PUF qualities in terms of high uniqueness and reliability of the response bits. We emphasize that the proposed function does not increase the entropy extracted from a PUF. However, the distinct advantage that we get out of this function is that using a smaller area, a chip can produce a significantly large set of CRPs that are useful for security applications even if the entropy is limited by the circuit resource. These extended set of CRPs could be termed as pseudo-independent CRPs of a PUF. This is similar to the idea of a pseudo-random number generator which is used to produce a stream of bits that are statistically indistinguishable from a truly random sequence of number. With a detailed security analysis, we show that the proposed PUF can prevent relevant attacks against it.

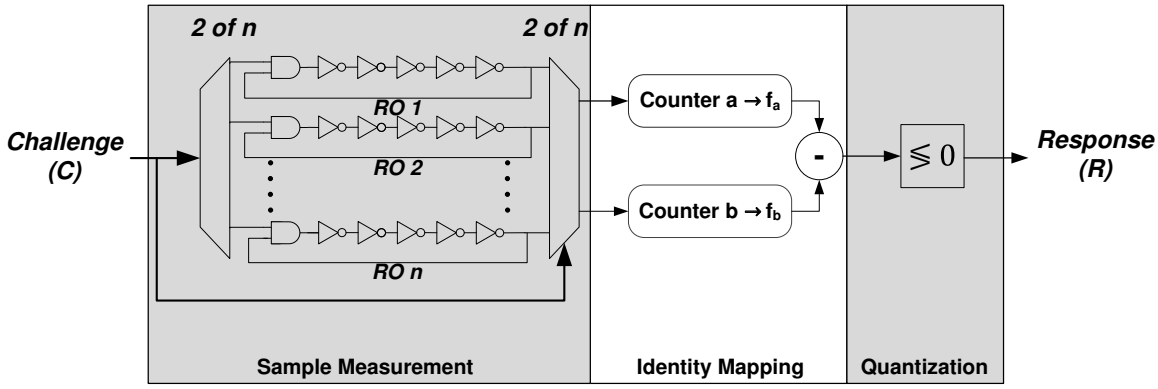

Figure 4.14: Traditional RO PUF

Though we demonstrate the complete idea based on the RO PUF, it can be applied to any PUF that produces real-valued responses. We analyzed the identity mapping function using statistical hypothesis test to demonstrate the enhanced CRP generation capability of the proposed function. The detail of the hypothesis testing can be found in [33]. We show a prototype hardware implementation of the enhanced PUF on a low cost commercial off-the-shelf FPGA. We also characterize the enhanced PUF over a group of 125 FPGAs. Moreover, we test it over varying temperature and supply voltage and provide PUF reliability data.

4.3.1 New Identity Mapping Function

In this section, we present the detailed construction of the proposed identity mapping function. The new identity mapping function (Q) replaces the frequency difference-based identity mapping used in the traditional RO PUF. Figure 4.14 and Figure 4.15 show the traditional RO PUF and the modified RO PUF respectively using the system model. We also employ a different quantization method (sh) called Shielding function [26] instead of the simple comparison method.

In order to describe the construction of the identity mapping function, we first define few notations and introduce a model to explain our approach in detail. Let f_{ijl} be the frequency value for the l-th measurement of the j-th RO in chip i, where $i = 1, ..., n$, $j = 1, \ldots, m$, and

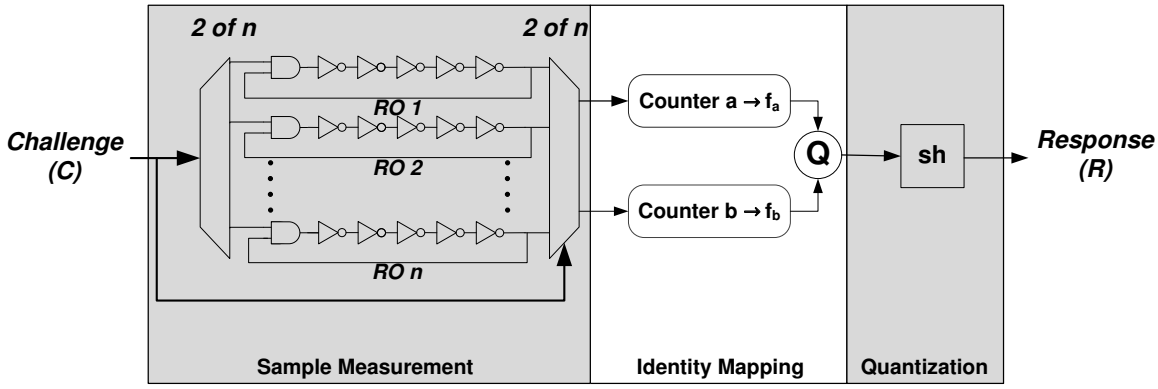

Figure 4.15: Modified RO PUF

$l = 1, \ldots, r$. Using these frequency data, we consider the following model which is expressed as a function of unknown parameters and an additive error term,

$$f_{ijl} = f_{ij} + \epsilon_{ijl} \tag{4.8}$$

where f_{ij} is the fixed unknown mean frequency of the j-th RO in the i-th chip, and ϵ_{ijl} is a random measurement error. f_{ij} is estimated by averaging f_{ijl} over a sample size of 100 ($l = 1, 2, ..., 100$) assuming a normal distribution of error ϵ_{ijl} with zero mean. Using f_{ij} from a chip i, we define a sample space S_t which is the set of possible outcomes of an experiment in which t RO frequencies are selected from m RO frequencies where $2 \leq t \leq m$. (For the simplicity of discussion, we omit the subscript i assuming the subsequent analysis is done for the chip, i). For example, S_2 contains all possible pairs of RO frequencies as shown below.

$$S_2 = \{(f_1, f_2), (f_1, f_3), ..., (f_1, f_m), (f_2, f_3), (f_2, f_4), ..., (f_2, f_m), ...(f_{m-1}, f_m)\}$$

Therefore, $|S_2| = \binom{m}{2}$. Similarly, S_3 contains all possible triplets of RO frequencies ($|S_3| = \binom{m}{3}$). Likewise, $|S_t| = \binom{m}{t}$. We now define the random variable Q_t that assigns a real

number, X, to each outcome of S_t.

$$
\begin{aligned}
&Q_t : S_t \rightarrow X \text{ such that} \\
&Q_t(f_{x1}, f_{x2}, f_{x3}, \ldots, f_{xt}) = \sum_{u=1}^{t-1} \sum_{v=u+1}^{t} w_{(xu)(xv)} \cdot ||f_{xu} - f_{xv}||^e. \\
&where\ 1 \leq x1, x2, x3, ..., xt \leq m \\
&and\ \ x1 \neq x2 \neq x3 \neq .... \neq xt\ \ and\ \ 2 \leq t \leq m
\end{aligned} \tag{4.9}
$$

The weight factor $w_{(xu)(xv)}$ can include additional information about the design. In our case, we assign it the Euclidean distance between the ROs xu and xv in the chip. In the implementation section, we will describe how the ROs are assigned spatial co-ordinates. $||.||$ is the absolute difference between the pair of RO frequencies f_{xu} and f_{xv}. To make the function Q a non-linear one, e may be any real number except 1. For example, we may assign $e = 0.5$ or 2. In our case we assign it as 0.5.

Corresponding to each of the $\binom{m}{t}$ combinations in S_t, there is one Q value. Therefore, for all possible values of t ($2 \leq t \leq m$), the total number of Q values is $|Q| = \binom{m}{2} + \binom{m}{3} + \binom{m}{4} + ... + \binom{m}{m} = 2^m - m - 1$. $2^m - m - 1$ response bits can be generated from these Q-values using a quantization method. In contrast, the traditional method defines the upper bound on the number of CRPs as $log_2(m!)$ [45]. Moreover, the bit-extraction method proposed in [45] can produce only $\binom{m}{2} = m(m-1)/2$ response bits using all possible pairwise comparison among m RO frequencies. Both of these quantities are smaller than our proposed approach. Whether the distributions of Q values are the same among chips or not has been validated using statistical hypothesis test. The detailed result of the hypothesis test can be found in [33].

4.3.2 Quantization

For the quantization of the Q values, we evaluated several standard quantization methods that are used in the biometric applications including the Shielding function[26] and the reli-

able bit extraction method [46]. Based on the comparison results, we selected the Shielding function. It has the property to shield any knowledge about the real-valued quantities that it quantizes from being revealed through the quantized binary output, leading to its name. We briefly introduce the method of Shielding function.

In the operation of a PUF, there are two phases: a) In the *enrollment* phase of the PUF, a reference response bit r is generated at normal operating condition by applying a challenge c and is stored in a secure database for subsequent operation of the PUF. b) In the *evaluation* phase, the PUF is supplied with the challenge c to generate a noisy version of r called r'. In the Shielding function, the range of a Q value is divided in several equal intervals with a width q during the enrollment phase. The intervals are alternatively assigned '0' and '1'. Any Q_{ti} of a chip i falling within an interval is assigned the corresponding binary digit to create r_{ti}. A helper data W_{ti}, based on q, is generated so that a noisy Q'_{ti} is correctly placed in the same interval as Q_{ti} during the evaluation phase. The helper data during the enrollment is derived as follows -

$$W_{ti} = \begin{cases} (2n + 0.5)q - \mu_{ti}, & \text{if } r_{ti} = 1 \\ (2n - 0.5)q - \mu_{ti}, & \text{if } r_{ti} = 0 \end{cases} \tag{4.10}$$

where $n \in Z$ such that $W_{ti} < q$. μ_{ti} is the average of Q_{ti}. r_{ti} is the binary digit assigned to the interval in which Q_{ti} resides. During the response evaluation, a binary output r'_{ti} is derived as follows -

$$r'_{ti} = \begin{cases} 1 & \text{if } \quad 2nq \leq Q'_{ti} + W_{ti} < (2n + 1)q \\ 0 & \text{if } \quad (2n - 1)q \leq Q'_{ti} + W_{ti} < 2nq \end{cases} \tag{4.11}$$

r'_{ti} is a noisy version of r_{ti} and not the complement of r_{ti}.

4.3.3 Robust PUF Implementation

In this section, we describe our prototyping effort for the proposed solution. The prototype includes a PC and an FPGA configured with the modified RO PUF (shown by the dotted

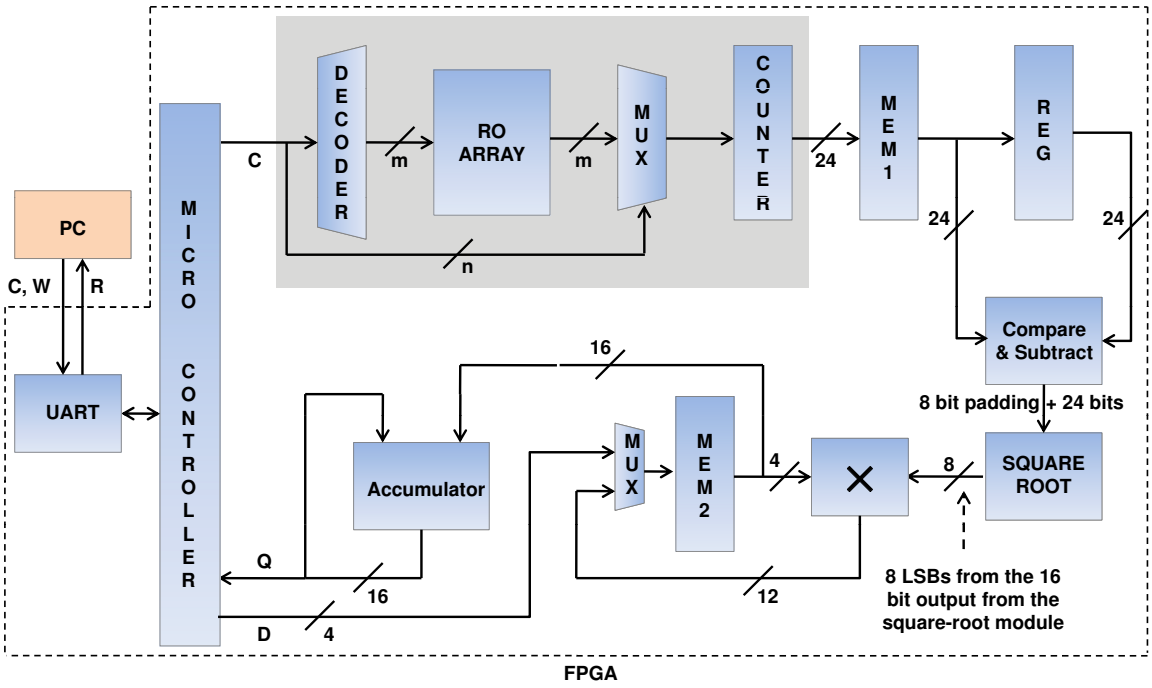

Figure 4.16: Architecture of our robust PUF

box in Figure 4.16). During the enrollment of the PUF, several samples of the Q values are taken using a set of challenge inputs (C) (refer to Figure 4.16) and a set of reference PUF responses (R) are derived and stored in the PC along with the corresponding helper data (W) for the Shielding function (refer to Equation(4.10)). During the PUF response evaluation, the PC sends C and W to the PUF to derive the response R.

The PUF has an 8-bit micro-controller with a data-path to implement the sample measurement of the RO frequencies, the generation of the Q values for the identity mapping, and the quantization of the Q values. Detail of the implementation is shown in Figure 4.16. All the control signals and memory addresses are supplied by the 8-bit micro-controller unit (the control signals, the address bus, clock inputs are not shown for simplicity). The statistical hypothesis test is not a part of the implementation and has been separately done. We now discuss the implementation in detail.

Sample Measurement

The part of the circuit shown in the gray box measures the RO frequencies from an array of m ROs. The ROs are placed in a 2D array. The (x,y) position of an RO in the array is used to calculate the Euclidean distance between an RO pair. The RO at the leftmost location of the bottom row of the array is assigned the co-ordinate (0,0), and the rests are assigned co-ordinates with respect to it. An n-bit challenge input C enables one of the m ring oscillators at a time (usually, $m = 2^n$), and the frequency of the enabled RO is measured with a 24-bit counter. The individual RO frequencies are stored in the memory $MEM1$. This is a one-time operation.

Identity Mapping

We used $e = 0.5$ and the $w_{(xu)(xv)}$ as the Euclidean distance in Equation(4.9) defined in Section 4.3.1. It requires an integer square-root and a multiplication operation. The square-root module is implemented using a non-restoring algorithm with the iterative method [23]. This implementation results in lower throughput but smaller area. An 4x8 bit pipelined multiplier is used for the multiplication operation. The compare-subtract module is used to compute the absolute difference of two RO frequencies. One of the operands for the compare-subtract module comes directly from the memory $MEM1$ and, the other operand is provided through the memory element REG. The Q values are generated using three steps.

Step 1 - Since the factor ($w_{(xu)(xv)}$) is constant, the memory $MEM2$ is initialized with all possible pairwise distances using the input D from the micro-controller. It is subsequently overwritten by the corresponding Q_2 values in the step 2.

Step 2 - According to the definition of Q, all the Q values are the combination of the Q_2 values (refer to Equation(4.9)). To minimize the number of multiplication and square-root operations, all the required Q_2 values are calculated at a time and stored in memory $MEM2$.

This a one-time operation during the PUF operation. All the responses can be generated by re-using these values as explained in the step 3.

Step 3 - During the generation of a PUF response, the required Q value is generated by summing the appropriate Q_2 values (Equation(4.9)) stored in $MEM2$. This summation operation takes place in the accumulator.

Quantization

The micro-controller calculates the sum $W + Q$ (Eqn 4.11) and compares it with the appropriate interval of the Shielding function to generate the binary response that is sent to the PC.

4.3.4 Results

A prototype circuit with 16 ROs has been built on a Xilinx Spartan 3E S500 FPGA. An RO with five inverters is created as a hard macro and instantiated to create a 2x8 array. A Picoblaze micro-controller core is used with an UART interface to communicate with the PC. The total FPGA resource used after place and route is 456 slices along with 2 Block RAMs. A Spartan 3E S500 FPGA has 4656 slices in total. Hence, less than 10% ($(456/4656) \times 100\% = 9.8\%$) of the available resource is used by the implementation making enough resources available to implement other circuits on the FPGA besides the PUF. Table 4.2 shows a summary of the implementation including the total number of challenges available, total area consumed, total dynamic power dissiption, and the clock cycles required for different operations. The cycle counts showing the delay overhead does not include the communication with the PC using UART. To evaluate one single response bit using t ROs, the required number of cycles is $66000 + \binom{16}{2} \times 78 + 20 \times \binom{t}{2} + 100 = 73920 + 20 \times \binom{t}{2} + 100$. For example, a challenge consisting 4 ROs will take approximately 1.5 msec to derive the response using a 50MHz clock. We note that this includes the one-time overhead of RO frequency

Table 4.2: Detail of the implemented design

No of ROs	16
No of challenges	65519
Total area of ROs	64 slices
Area of the rest of the components	392 slices
Total Dynamic power consumption	2.73 mW
Sample Measurement	66,000 cycles for each RO
Identity Mapping	78 cycles for each Q_2 $20 \times \binom{t}{2}$ cycles for each Q_t
Quantization	Approx. 100 cycles for each Q_t
Clock frequency	50MHz

measurement and the Q_2 calculation consuming 73920 cylces. In a particular session of PUF CRP extraction, this overhead is incurred only once and is not required to be calculated for each response bits extracted.

Evaluation of the PUF Responses

We evaluate the PUF responses using the parameter defined in Section 2.4 namely uniqueness, bit-aliasing, uniformity and reliability. These parameters are estimated using 65519 ($= 2^{16} - 16 - 1$) PUF response bits generated from each of a group of 125 FPGAs. Figure 4.17 shows the uniqueness based on $\binom{125}{2}$=7750 comparisons. It is clear from the figure that the distribution is centered around 50% which is the ideal value for truly random PUF responses. Also in Table 4.3, the minimum value of 49.21% of the uniqueness shows that any pair of FPGAs in the experiment has at least 32241 (49.21% of 65519) different response bits. Thus, the PUF with the identity-mapping function is able to distinguish the FPGAs clearly. Figure 4.18 and 4.19 show the uniformity of PUF responses from 125 FPGAs, and

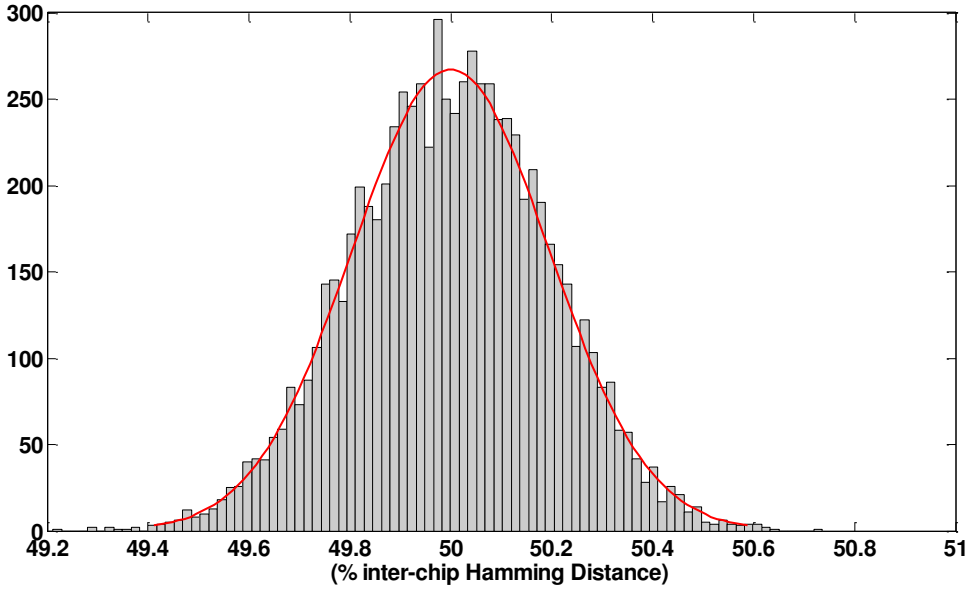

Figure 4.17: Distribution of the PUF uniqueness using identity mapping.

the bit aliasing for each of the 65519 bit positions respectively. The average value of 50.02% for uniformity (Table 4.3) shows that the PUF with the identity mapping function can produce response bits that are fairly uniformly distributed between '0's and '1's. The plot in Figure 4.18 show that the deviation from the average is small.

On the other hand, bit-aliasing is also found to be around 50% (Figure 4.19). However, more importantly, the minimum value of 31.2% and the maximum value of 68.8% in Table 4.3 show that all the bit positions are fairly different from a complete bit-aliasing. A value

Table 4.3: Performance of PUF using identity mapping

	Average	Minimum	Maximum
Uniqueness	49.99%	49.21%	50.73%
Uniformity	50.02%	49.41%	50.52%
Bit-aliasing	50.02%	31.2%	68.8%

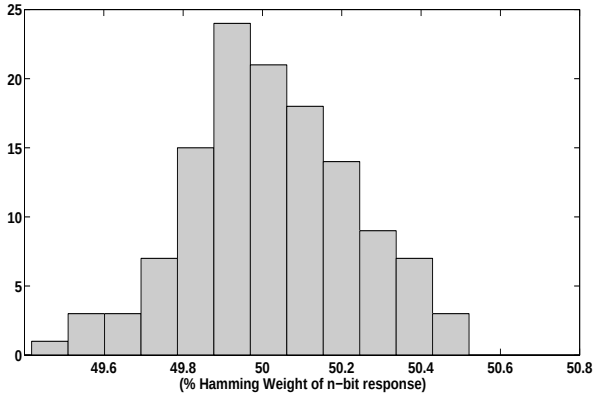

Figure 4.18: Uniformity of PUF responses using identity mapping.

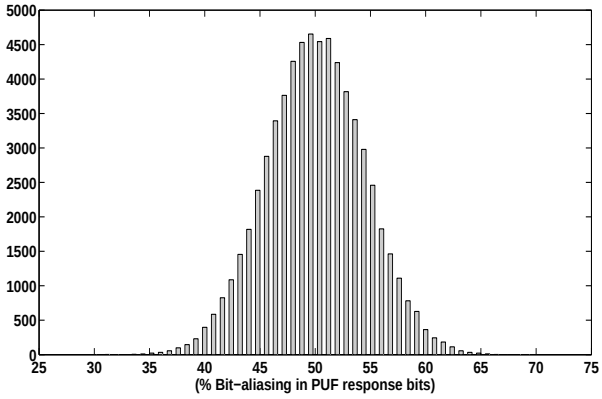

Figure 4.19: Bit aliasing of PUF responses using identity mapping.

of 0% at the l-th bit would mean all the chips produce 0 at that bit position while 100% would mean all the chips produce 1. Both these cases are complete bit-aliasing.

Now we present a uniqueness comparison analysis between the PUF with the identity-mapping function and the one without it. Both the designs implement the quantization using the Shielding function. This is done in order to show the advantage of introducing the identity-mapping function while keeping the other two components of the PUF system model, the sample measurement and the quantization, the same. For the PUF without the identity mapping, we directly quantized the raw frequencies from 16 ROs using the Shielding function to produce 16 response bits. It is done for all 125 FPGAs.

Table 4.4 shows the PUF uniqueness without the identity-mapping function. Though both have an average around 50%, it is evident that the PUF with the identity-mapping function has a smaller deviation from the average. On the contrary, the PUF without the identity mapping has a large deviation with the minimum and the maximum being 6.25% and 93.75% respectively (Table 4.4). The minimum value shows that there is a high probability that some pairs of chips will map to the same identifier value, in particular under environmental variations. However, it is not the case for the PUF with the identity-mapping function as the minimum is 49.21%. Since both of them use 16 ROs, it is clear that a more efficient PUF

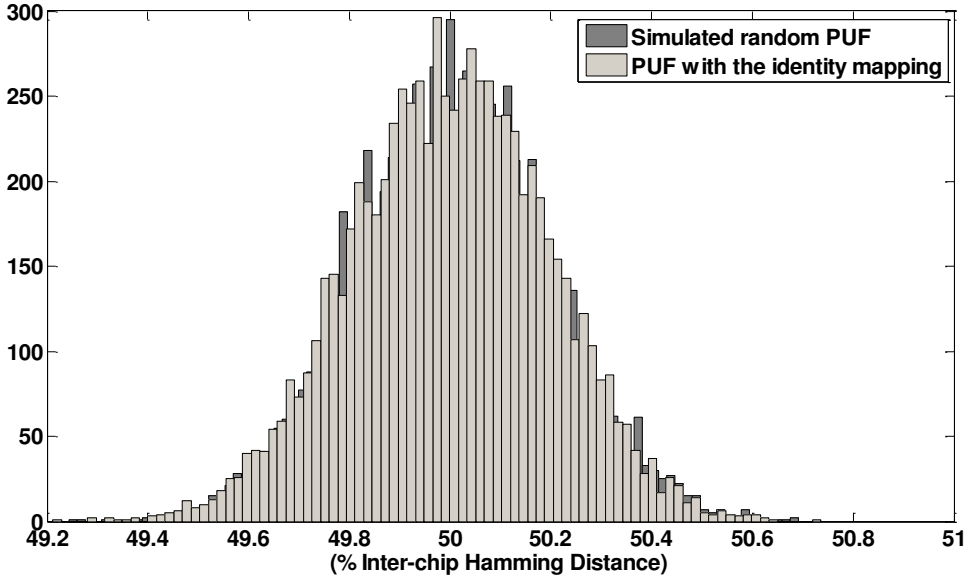

Figure 4.20: Comparison of the uniqueness between the PUF with identity mapping and a random source of response.

can be designed using the identity-mapping function compared with a PUF without identity mapping. We note here that the minimum and maximum value of bit-aliasing in the PUF without the identity mapping are respectively higher and lower than those in the proposed PUF. However, the average of bit-aliasing in the proposed PUF is around 50%, and the majority are centered around the average as shown in Figure 4.19. We further compared our proposed technique with a perfectly random source. Using Matlab simulation, we generated 125 uniformly distributed, 65519-bit long binary responses and calculated the uniqueness.

Table 4.4: Performance of PUF without identity mapping

	Average	Minimum	Maximum
Uniqueness	50.07%	6.25%	93.75%
Uniformity	49.4%	18.75%	81.25%
Bit-aliasing	49.4%	41.6%	59.2%

Figure 4.20 shows the uniqueness distribution of the proposed PUF overlapped on that of the random source. It is clear from the figure that the proposed PUF behaves very similar to a random source.

Regarding the area cost, our implementation of the PUF with the identity mapping uses 456 slices including measurement and communication circuitry. Though our implementation uses 2 Block RAM units, they are used for Picoblaze instruction memory and to store 120 Q_2 values (MEM2 in Figure 4.16). It is reasonable to assume that the PUF without the identity mapping would also need a Picoblaze-like unit for the control and communication and a memory space to store RO frequencies. Though the PUF without the identity mapping does not require the circuit for generating Q values, scaling up the number of CRPs involves significant increase in area as one CRP requires an RO consuming 4 slices on the used platform.

In the traditional RO-PUF, 128 response bits have been produced using 1024 ROs with the help of a 1-out-of-8 scheme to enhance the reliability [45]. Without the reliability scheme, it needs two ROs for 1 bit response, leading to a total of 256 ROs for 128 response bits. Hence, the required number of slices just for implementing the ROs is $256 \times 4 = 1024$. Therefore, only the ROs in the traditional method uses $1024/456 = 2.25$ times more area compared to the total area (with all the required components) consumed by the proposed method while the proposed method can produce 65519 bits compared to 128 bits produced by the traditional method.

The reliability of the PUF responses is measured over varying environmental conditions. We tested the PUF responses for 9 different ambient temperatures from 0^oC to 70^oC using a convection heat chamber to cover the allowed operating range of the used FPGA. We also tested them for $\pm 20\%$ of the supply voltage of the FPGA core using a DC regulated power supply. All the measurements are taken with 100 samples for a *single* FPGA chip. Figure 4.21 shows a comparison of the reliability for the PUF with the identity-mapping function and the one without the identity-mapping function. The trend shows that the PUF with

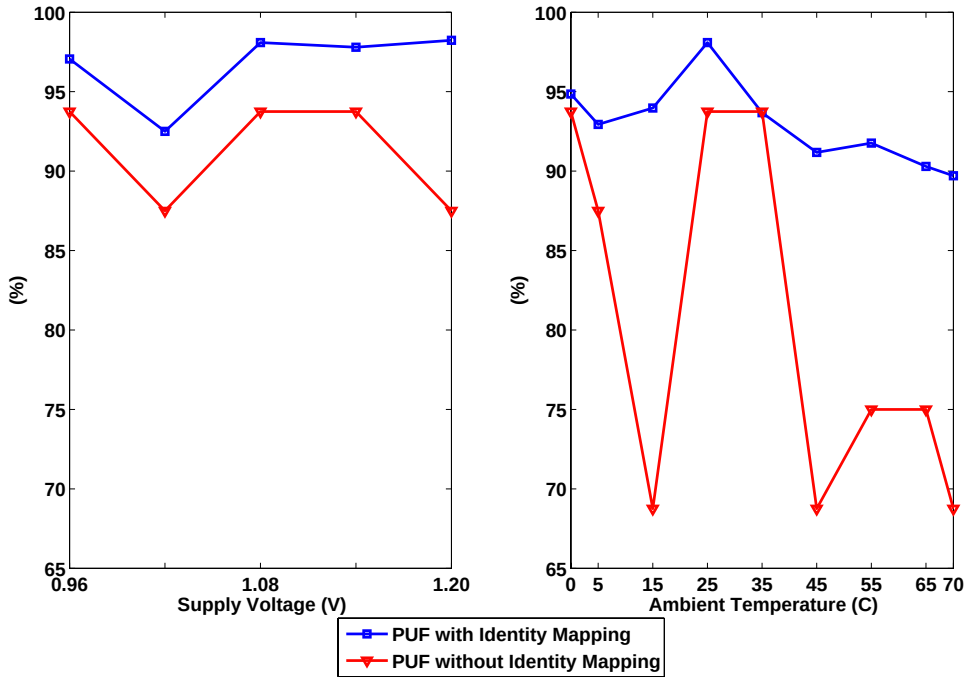

Figure 4.21: Comparison of the reliability between the PUF with the identity mapping and without the identity mapping

the identity-mapping function is more reliable than the PUF without the identity mapping for both varying voltage and temperature. Though the exact reason for this observation cannot be confirmed without a detailed analysis, one of the possible reasons might be as follows. Since a Q value is a summation of several frequency differences, the variations in the individual frequency differences are averaged out. We plan to investigate the reliability factor in more detail as part of our future work.

Security Analysis

The proposed PUF generates CRPs that are not fully independent in information-theoretic sense. This may give rise to security threats such as prediction attack in which the adversary attempts to predict the value of an unknown response bit using an already-known response bit. In this section, with the help of a security analysis of the implementation of

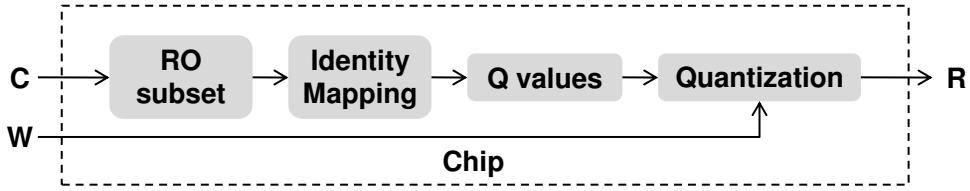

Figure 4.22: Operational set-up of the proposed PUF

our technique, we show that this threat can be averted.

We assume that the PUF security is not violated during the enrollment phase, and no information about the PUF is leaked during this period. We analyze the security vulnerability when the PUF is in use in a hostile environment. We assume that an adversary can observe challenges(C), responses(R), and helper data (W) of the PUF. He can also adaptively apply C and W to observe corresponding responses. The input challenge C creates the RO frequency subset, the identity-mapping function generates the Q-values, and the quantization method converts the Q-values to binary response bits R using W as shown in Figure 4.22. By $R(RO_a, RO_b)$, we denote the binary response produced through the Q_2 value between a pair of ring oscillators, RO_a and RO_b. Similarly, $R(RO_a, RO_b, RO_c)$ denotes the binary response produced through the Q_3 value between RO_a,RO_b, and RO_c. The security analysis is based on two sub-groups of 2 and 3 ROs out of the 16 ROs. However, the security analysis can be extended to the sub-groups with higher number of ROs. We consider six different scenarios to assess the security risk of the proposed PUF.

1. Uniformity of response - To begin with, we evaluate the proportion of '0'/'1' in the derived response bits. A set of uniformly distributed response bits are essential to prevent an attacker from guessing if a response of a particular chip is biased towards a particular binary value. A similar method has been used by Majzoobi et al. as single bit probability in evaluating security of Arbiter PUF [35]. The value of the uniformity in Table 4.3 shows that the proposed PUF produces response bits with a proportion of '0'/'1' that is very close to 50%, thus excluding any bias.

2. Response conditioned by challenge - In this scenario, we check if there is any

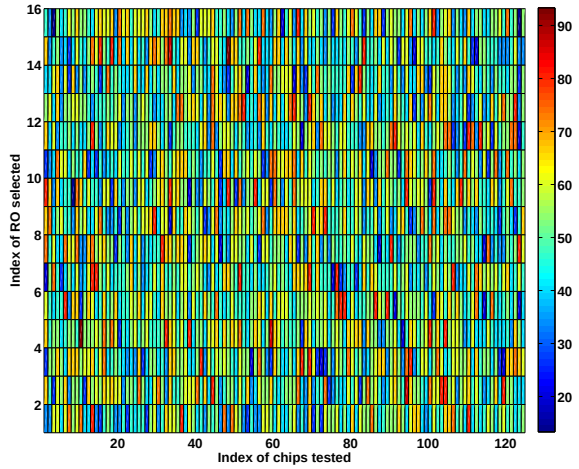

Figure 4.23: Response conditioned by challenge for groups of 2 ROs. Average=51.7%.

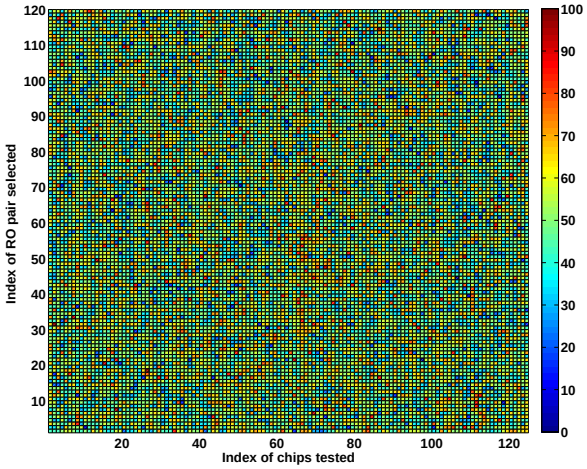

Figure 4.24: Response conditioned by challenge for groups of 3 ROs. Average=50.23%.

correlation between a challenge and a response when we keep a part of a particular challenge fixed while changing the other part of the challenge. A similar technique has been employed in [35] as conditional probability. For a challenge with 2 ROs, we keep one of the ROs fixed while changing the other. The fixed ring oscillator, say RO_a, can form 15 response bits given by $R(RO_a, RO_i), 0 \leq i \leq 15, a \neq i$. We calculate the quantity $\frac{1}{15}\sum_{i=0}^{15} HW(R(RO_a, RO_i)) \times 100\%, a \neq i$. ($HW$ denotes Hamming weight.) For an unbiased PUF, it will generate a value of 50% on an average meaning that fixing an RO as part of the challenge produces '0's and '1's with equal probability. Figure 4.23 shows the distribution of the above quantity for all 16 ROs for 125 FPGA chips tested. It shows that the sample set is averaged around 50% with no obvious trend to suggest any type of bias. We carried out the similar test for groups of 3 ROs. We kept a pair of ROs out of the 3 ROs fixed while changing the remaining one. The fixed pair of ROs, say (RO_a, RO_b), can form 14 response bits given by $R(RO_a, RO_b, RO_i), 0 \leq i \leq 15, a \neq b \neq i$. We calculate the quantity $\frac{1}{14}\sum_{i=0}^{15} HW(R(RO_a, RO_b, RO_i)) \times 100\%, a \neq b \neq i$. For an unbiased PUF, this quantity should also produce a value 50%. Figure 4.24 shows the distribution for 120 possible pairs (R_a, R_b). It also shows that there is no prominent trend in the result to suggest a bias.

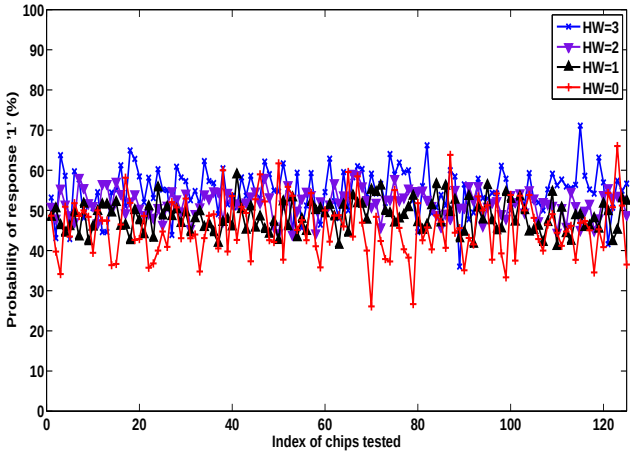

Figure 4.25: Inter-response dependency

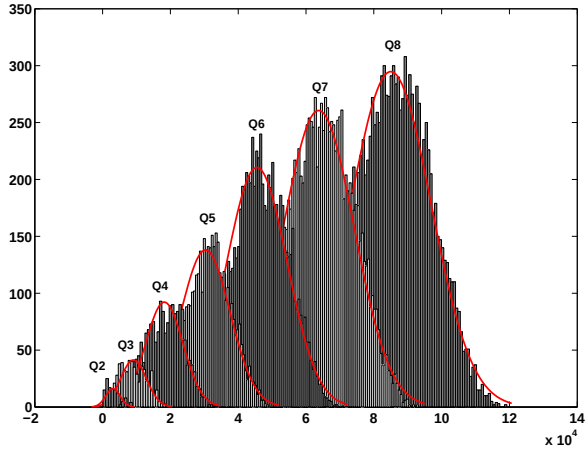

Figure 4.26: Distribution of Q values

3. Inter-response dependency test - In the proposed PUF, the building blocks for all Q values are the Q_2 values (Eqn (4.9)). We want to see if we can predict responses from Q_3 values by observing responses from corresponding Q_2 values. In particular, we test if it is possible to predict $R(RO_a, RO_b, RO_c)$ by knowing $R(RO_a, RO_b)$, $R(RO_b, RO_c)$ and $R(RO_a, RO_c)$. We first observe all possible values of $R(RO_a, RO_b, RO_c)$. There are a total of 560 such responses. Now, corresponding to each of these 560 responses, we calculate the quantity $HW(R(RO_a, RO_b), R(RO_b, RO_c), R(RO_a, RO_c))$. It has four possible values: 3,2,1 and 0. We divide 560 responses in four groups: HW=3, HW=2, HW=1 and HW=0. Now, we count how many '1' bits have been produced by $R(RO_a, RO_b, RO_c)$ in total for each of those four groups. The HW information of the responses from Q_2 values should not have any correlation with the corresponding response from Q_3 value i.e. $R(RO_a, RO_b, RO_c)$. For example, if the group with HW=3 has x response bits out of 560 $R(RO_a, RO_b, RO_c)$ bits, and y bits out of x are '1', then $(y/x) * 100\%$ (shown as 'probability of response 1' on the y-axis in Figure 4.25) should be around 50% for an unbiased PUF. This will indicate that it is difficult to construct $R(RO_a, RO_b, RO_c)$ by knowing $R(RO_a, RO_b)$, $R(RO_b, RO_c)$ and $R(RO_a, RO_c)$. Figure 4.25 shows that all the four cases are centered around 50% over a set of 125 chips. Therefore, this result shows that inter-response dependency is not present in the proposed PUF.

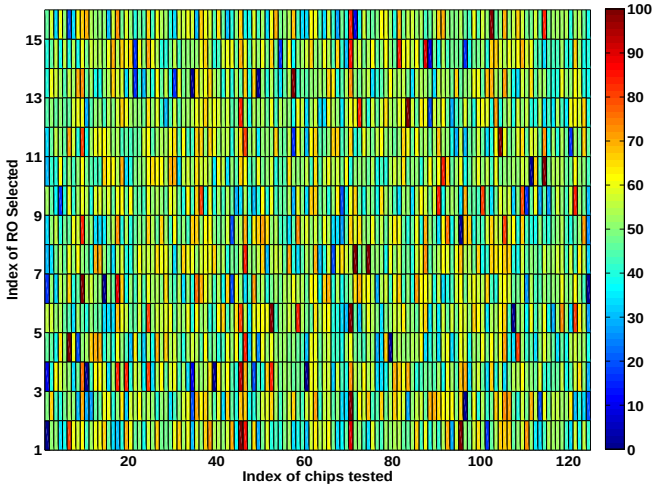

Figure 4.27: Differential attack scenario with HW=0. Average=51.27%.

In this context, we analyze the dependencies among Q values using their distributions. Figure 4.26 shows distributions of different Q values (taken from an arbitrary chip from the pool of 125 chips). Each of the distributions has different mean and variance. This shows that they do not have same range. Additionally, even if there exist overlaps among different Q values, response bits from the overlapped Q values do not have correlation as each of the Q values have their own quantization width q based on their individual variances. This is supported by the experimental results. Table 4.3 shows that the response bits are fairly equally distributed between '0' and '1'.

4. Differential attack - In this scenario, we observe $R(RO_a, RO_b)$ and then replace RO_b with RO_c and observe $R(RO_a, RO_c)$. The objective is to predict $R(RO_b, RO_c)$ from these two observed responses. For each $RO_a, 0 \leq a \leq 15$, there are $\binom{15}{2} = 105$ possible pairs $(RO_b, RO_c), a \neq b \neq c, 0 \leq b \leq 15, 0 \leq c \leq 15$. We now calculate $HW(R(RO_a, RO_b), R(RO_a, RO_c))$ for each 105 pairs. The possible values of the HW are 0,1,2. We divide 105 responses in three groups: HW=2, HW=1 and HW=0. We then count how many '1' bits have been produced by $R(RO_b, RO_c)$ in total by each of those three group. For example, if the group with HW=2 has x response bits out of 105 $R(RO_b, RO_c)$ bits, and y bits out of x are '1', then $(y/x) * 100\%$ should be around 50% for an unbiased PUF. Figure 4.27, 4.28 and 4.29

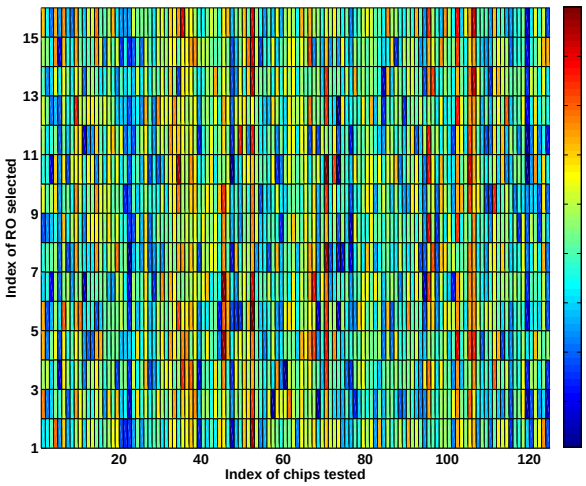

Figure 4.28: Differential attack scenario with HW=1. Average=51.40%.

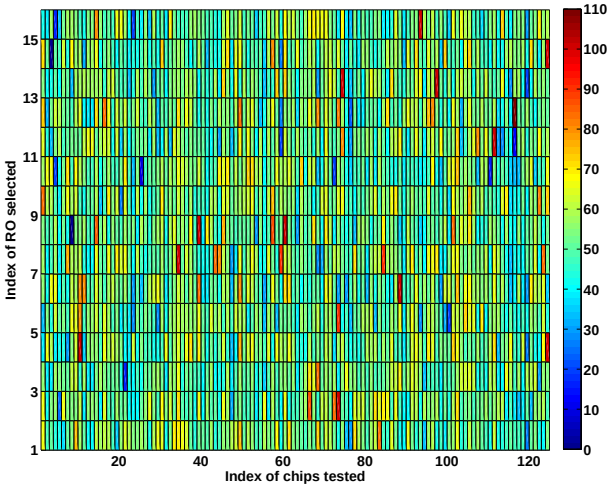

Figure 4.29: Differential attack scenario with HW=2. Average=52.54%.

show the scenarios for HW=0, HW=1 and HW=2 respectively. Each of the scenarios include data for all individual 16 ROs (x-axis) for all 125 chips tested (y-axis). The results show that none of the scenarios suggest any bias that can lead to a successful differential attack.

5. Model building through machine learning - Ruhmair et al. proposed model building attacks on several PUF architectures using machine learning techniques [39, 41]. The technique that has been used to attack the RO PUF is a quick sorting method to determine the relative ranking of the RO frequencies. This technique is particularly suitable for attacking the traditional RO PUF as its responses can be constructed purely based on frequency ranking without the need of knowing the absolute frequency values. However, in the proposed PUF, the final response is derived after the transformation of RO frequencies to Q values and then a random assignment of binary values by the quantization method. Therefore, the sorting technique will not work against the proposed PUF as it does not depend on the ranks of RO frequencies. The similar reason is valid for the reverse engineering attack as will be discussed as the next scenario.

6. Reverse engineering attack - An attacker may try to gain knowledge of the RO frequencies or the Q values to construct R. At the input, even if an adversary knows the

value of C i.e. the RO subset, the corresponding Q-value is calculated within the FPGA chip and never comes out of the chip. This gives two advantages:
1) Even though Q_j and Q_i are not completely independent, predicting Q_j by knowing Q_i $(i \neq j)$ is not possible because no information about any Q values is accessible.
2) Since Q values are not accessible, reverse engineering of the original RO frequencies from Q values becomes impossible.

On the other hand, at the output, there are two possible points of vulnerability: the helper data W of the Shielding function and the response R. An attacker may try to predict an unknown response bit R_i from its public helper data W_i or from a known response bit $R_j, (i \neq j)$. However, an attacker faces difficulties due to the following reasons.
1) The probability that information about R leaked by W is of the order of 10^{-5} for the Shielding function [25] when $q/\sigma = 1$, q = width of the quantization interval, σ is the standard deviation of Q. Therefore, predicting R using W with high probability is difficult.
2) In a traditional RO-PUF, R is evaluated by pair-wise comparison of RO frequencies. As a result, only the knowledge of the ranking of the RO frequencies can reveal R. Unlike direct comparison, the Shielding function maps arbitrary binary bits to Q-values. Therefore, to predict an unknown response R_i, one has to estimate the absolute value of corresponding Q which is difficult.
Finally, it should be mentioned that there is a possibility of side-channel attack on the proposed PUF. This is because the proposed implementation uses arithmetic units such as a square-root unit and a multiplier. It also makes use of intermediate storage (MEM1 and MEM2 in Figure 4.16). Using the side-channel power and timing measurements, one can try to read the frequency values. Though we did not consider side-channel attack in this research, we consider it as an important security issue and keep it as a part of future work.

	Sample Measurement	Identity Mapping	Quantization
Uniqueness	**Compensation for systematic process variation**	**CRP Enhancement technique**	-
Reliability	**Configurable Ring Oscillator**	**CRP Enhancement technique**	-
Robustness	**Compensation for systematic process variation**	**CRP Enhancement technique**	-

Figure 4.30: Summary of PUF enhancements using the system model

4.4 Summary

In this chapter, we presented three techniques to enhance the quality factors of a PUF. The first technique to compensate for the systematic process variation enhanced both the uniqueness and robustness at the sample measurement. The configurable ring oscillator (CRO) technique improved the reliability of PUF at the sample measurement too. Finally, we proposed a new identity mapping function to significantly expands the CRP set of RO-PUF using low area. We found from the experimental results that it improves the uniqueness and reliability, too.

In Figure 4.30, a summary representation of the proposed enhancements has been shown. The three proposed enhancements have been marked in different colors to distinguish them clearly. The locations of the enhancement techniques in the figure shows the component of the system model (column headings) at which they were applied as well as the quality factor of the PUF (row headings) they enhanced.

Chapter 5

PUF Characterization

Being an emerging security solution, PUFs are being studied extensively by the research community. Besides inventing new PUFs, researchers have also proposed many applications of PUFs and several different error correction techniques to make PUFs noise-free. Another important aspect of this research is the characterization of PUFs. A thorough characterization of PUFs is important because of the following reasons.

- A PUF circuit can only be completely characterized over a group of chips. Whether it can produce a chip-unique identifier or not cannot be validated using a single chip.
- A PUF exploits spatial random variability to extract an identity of a chip while it is adversely affected by several other on-chip variabilities due to thermal effect, ambient temperature variation, supply voltage fluctuation, and aging. A detailed understanding about manufacturing process variation, variability of circuit parameters (delay, threshold voltage) over changing operating conditions, and aging is essential to design an efficient PUF. Moreover, the study of circuit variability is not only helpful for the PUF research, it can also benefit circuit design in general.

We focused on a detailed characterization of PUFs as one of the major components of this research. We first briefly summarize the key contributions.

- We formulated a compact and portable experiment to carry out the characterization based on the RO PUF. Variability was measured in terms of ring oscillator frequencies. This experiment was able to measure a large group of chips in a fairly short period of time.

- We measured 193 chips. The measured chips were commercially available off-the-shelf FPGAs. They were manufactured out of a 90nm technology node. We also measured chips under variations in temperature and supply voltage. A detailed analysis of the collected dataset was also done.

- We studied the effect of circuit aging on the functionality of a PUF. On-chip accelerated aging tests were performed to observe the effect of aging on a PUF. Additionally, we extrapolated the effect of aging on a large group of chips using simulation. Also, a mitigation solution against aging has been proposed.

The usefulness of this experiment and the collected dataset is manifold.

- Ring oscillator data are widely used in modeling process variation [40, 37]. However, unavailability of large sample size is a problem. Sedcole et. al. mentioned the limitation of a small sample size in their variability study [40]. Our dataset can be used for studying several on-chip variability issues such as systematic variation and layout-dependent circuit behavior.

- We have collected several samples for individual RO frequencies. This will help in studying noise effects and to investigate new PUF error correction methods.

- We note that many of the existing experimental PUF research tend to use a small sample size [45, 55, 57], or an older generation of chips [55]. Our dataset based on a group of 90-nm FPGAs will help the researchers significantly.

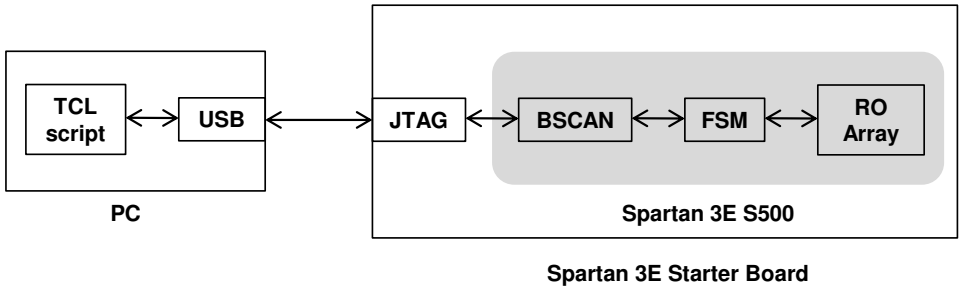

Figure 5.1: Experimental setup for the PUF characterization

5.1 Experimental Setup

This section briefly describes the experimental setup used for the characterization. While designing the experimental setup, the main emphasis was to collect as much useful data as possible keeping the runtime of the experiment within a fairly low limit. This was required because we measured the FPGAs belonging to a large group of graduate and undergraduate students in the ECE Department of Virginia Tech. The complete experimental setup is shown in Figure 5.1. The hardware part is implemented in a Spartan 3E FPGA board, and the software part is implemented in a PC. In the PC, a TCL script runs to interact with the hardware. The script configures the JTAG channel using a USB port and sends instructions to a finite state machine (FSM) to read the RO frequencies.

An array of 512 ROs were implemented in the Spartan3E S500 FPGA. The ROs were placed in a 32×16 array in the middle of the FPGA fabric. Individual ROs have five inverting stages implemented using Look Up Tables (LUTs). For identical layout, the ROs are created as hard macros and instantiated multiple times during synthesis. More information regarding the experimental setup can be found in [32].

5.2 A Large Scale Characterization of the RO PUF

In this section, we analyze the data collected from the experiment involving 193 FPGAs. 100 frequency samples for each of the 512 ROs per FPGAs have been used in the analysis. These samples were collected at normal operating condition. The accuracy of the measured RO frequencies was $\approx \pm 205$ppm. The average cycle counts in the counter that measured the RO frequencies was 256080. Hence, the resolution of the measurement was around $log_2(256080) \approx 18$ bits.

5.2.1 RO Loop Delay Model for Data Analysis

We define a simple RO loop delay model for analyzing the collected data set. It is similar to Equation (4.3) except the fact that in this model, we include the noise factor. For a pair of ROs, a and b, the loop delays are defined below.

$$d_{\mathrm{a}} = d_{\mathrm{avg}} + \Delta d_{\mathrm{pv\_a}} + \Delta d_{\mathrm{noise\_a}} \tag{5.1}$$

$$d_{\mathrm{b}} = d_{\mathrm{avg}} + \Delta d_{\mathrm{pv\_b}} + \Delta d_{\mathrm{noise\_b}} \tag{5.2}$$

where d_{avg} is the average delay of the RO. This quantity is assumed to be same across all the ROs in a chip. d_{pv} is the delay component due to the process variation (pv). This is constant for an RO and may vary from one RO to another. We assume it to be constant over time (we neglect aging effect here). d_{noise} is the delay component due to noise. It is a dynamic component and changes over time.

In a simple comparison method of quantization in the RO PUF, two RO frequencies, f_{a} and f_{b} are compared. Since $f = 1/d$, the sign of the quantity $d_{\mathrm{a}} - d_{\mathrm{b}}$ determines the PUF response.

$$d_{\mathrm{a}} - d_{\mathrm{b}} = (\Delta d_{\mathrm{pv\_a}} - \Delta d_{\mathrm{pv\_b}}) + (\Delta d_{\mathrm{noise\_a}} - \Delta d_{\mathrm{noise\_b}}) = \Delta d_{\mathrm{pv\_ab}} + \Delta d_{\mathrm{noise\_ab}} \tag{5.3}$$

During the PUF enrollment, average of an RO frequency is calculated using many samples to negate the noise factor. Hence, we assume $\Delta d_{\text{noise\_ab}} = 0$ during the enrollment. Therefore, when we generate the reference response during the enrollment, the sign of $d_{\text{a}} - d_{\text{b}}$ is nothing but the sign of $\Delta d_{\text{pv\_ab}}$. Hence, $\Delta d_{\text{pv\_ab}}$ determines the reference response.

During the evaluation, $\Delta d_{\text{noise\_ab}}$ comes into effect. If it cancels $\Delta d_{\text{pv\_ab}}$, the generated response bit becomes different from the reference. This results in an unreliable response bit. For a reliable operation of the PUF, $\Delta d_{\text{noise\_ab}}$ should be minimized.

5.2.2 Variability Statistics in terms of RO Frequency

First, we show how the average RO frequency is distributed over the entire sample set. The average frequency of the i-th chip ($1 \leq i \leq 193$), is measured as

$$F_i = \frac{1}{512} \sum_{j=1}^{512} f_{i,j} \tag{5.4}$$

where individual RO frequencies, $f_{i,j}$, are measured as

$$f_{i,j} = \frac{1}{100} \sum_{k=1}^{100} f'_{i,j,k} \tag{5.5}$$

where $f'_{i,j,k}$ is the k-th frequency sample of the j-th RO in the i-th chip. F is the estimate of d_{avg} in Equations (5.1) and (5.2) whereas f is the estimate of the quantity $d_{\text{avg}} + d_{\text{pv}}$. Figure 5.2 shows the distribution of the average frequencies of the individual chips across the entire sample set. The average of the distribution is 205.7 MHz which we call the global average. The slowest chip has a frequency of 171.6 MHz while the fastest has a frequency of 230.2 MHz. There is no chip having a frequency beyond 230 MHz. This is due to the speed binning of the chips. On the other hand, those chips having frequencies around 170 MHz may have defects due to reasons such as short-circuit and aging. The standard deviation of F_i across all 193 chips is 12.67 MHz which is 6.16% of the global average. This represents the inter-chip variation in terms of the average RO frequencies of the measured chips.

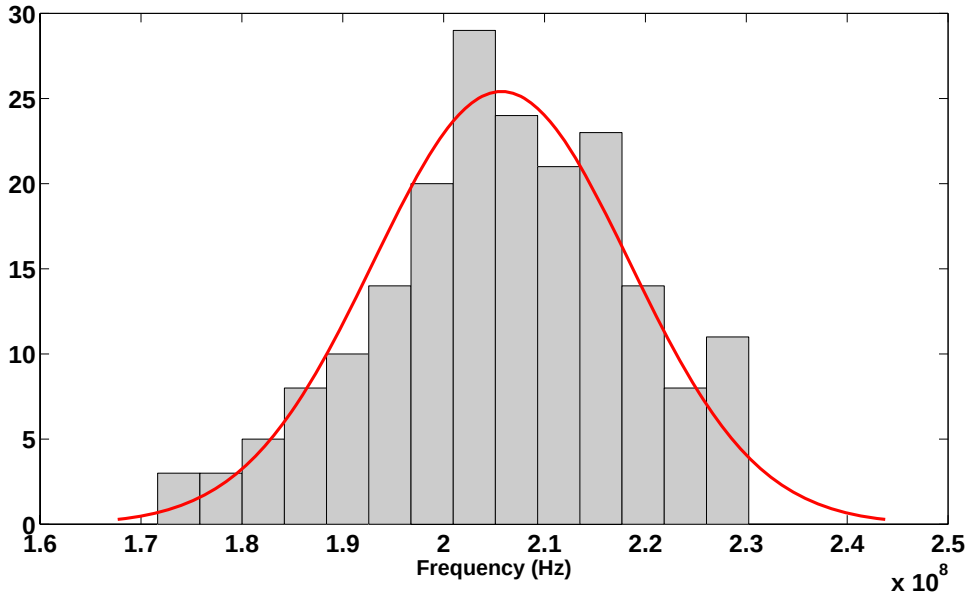

Figure 5.2: Distribution of the average RO frequency of individual chips

We evaluate the intra-chip variation of the RO frequencies. We calculate the standard deviation of the 512 RO frequencies in a chip. For the i-th chip, it is derived as follows.

$$\sigma_{\text{pv\_}i} = \sqrt{\frac{1}{511}\sum_{j=1}^{512}(f_{i,j} - F_i)^2} \tag{5.6}$$

Figure 5.3 shows the distribution of the normalized quantity $(\sigma_{\text{pv\_}i}/F_i) \times 100\%$. This is an estimate of the process variation component, Δd_{pv}, in the loop delay, and we term it as $\sigma_{\text{pv\_}i}$. Since it is supposed to remain static in a chip in the post-fabrication phase, we call it static intra-chip variation. The average static intra-chip variation is 0.75% with the maximum and the minimum of 1% and 0.58% respectively. For an RO PUF, this quantity should be greater than the noise component in order to have high reliability of PUF response bits. This is because high static intra-chip variation will lead to a high value of $\Delta d_{\text{pv\_ab}}$ in Equation (5.3), and $\Delta d_{\text{noise\_ab}}$ needs to be large enough to flip a response bit.

To estimate the value of d_{noise}, we first separately calculate the standard deviation of each

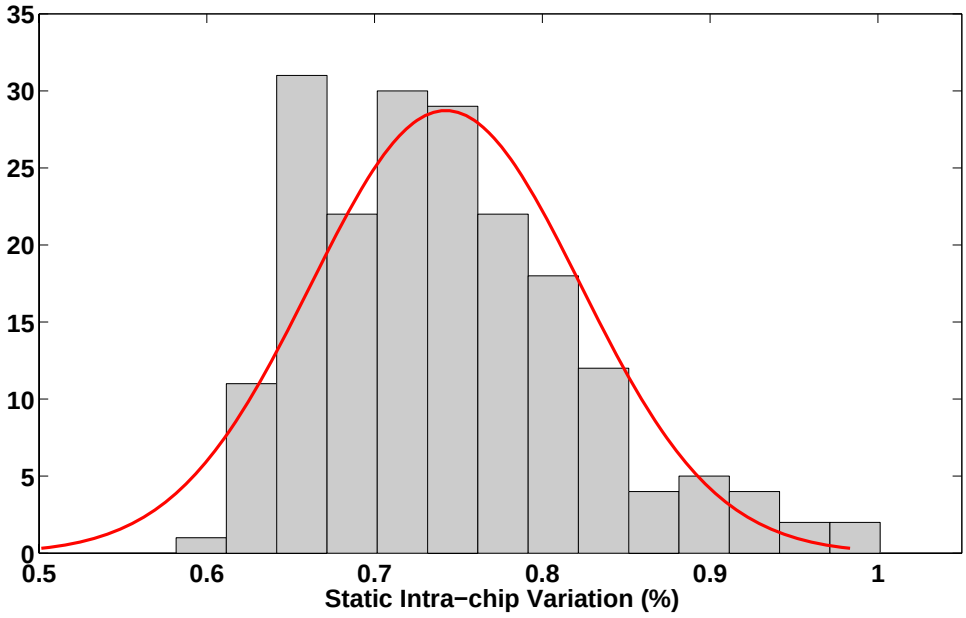

Figure 5.3: Distribution of static intra-chip variation

RO frequency over 100 samples as follows.

$$\sigma_{i,j} = \sqrt{\frac{1}{99}\sum_{k=1}^{100}(f'_{i,j,k} - f_{i,j})^2} \tag{5.7}$$

Then we define $\sigma_{\text{noise\_}i}$ for each chip to estimate d_{noise} in a normalized form as shown below. Since this quantity changes with time, we call it dynamic variation.

$$\sigma_{\text{noise\_}i} = \frac{1}{512}\sum_{j=1}^{512}(\frac{\sigma_{i,j}}{f_{i,j}} \times 100\%) \tag{5.8}$$

Figure 5.4 shows the distribution of $\sigma_{\text{noise\_}i}$. It can be noticed that most of the chips are centered around a value of 0.025% with a few outliers. These outliers are expected to generate relatively higher number of unreliable responses. However, the average static intra-chip variation of 0.75% is significantly higher than the average dynamic variation of 0.025%. This shows that the RO PUF is expected to be highly reliable at the normal operating condition.

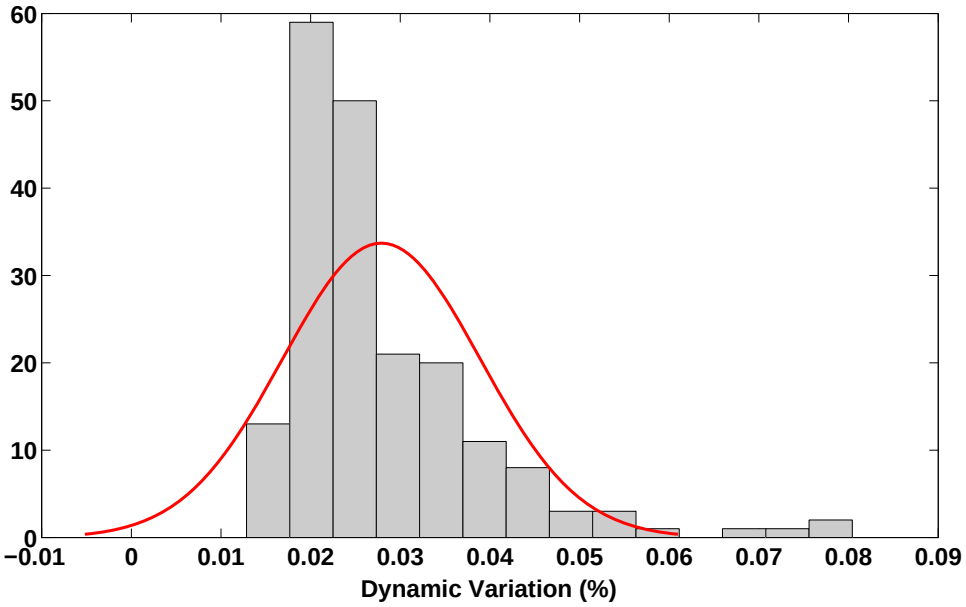

Figure 5.4: Distribution of dynamic variation

5.2.3 Why is a Large Scale Experiment Better?

We have so far presented few parameters representing the on-chip variability statistics in terms of RO frequency. By a simple analysis, we show that the size of the sample set has a large impact on this statistics. As a test case, we calculated the average RO frequency using a group of 16 chips. Using a sliding window method, we formed several groups of 16 chips (1 to 16, 2 to 17 and so on) in an arbitrary order. Figure 5.5 shows the plot of the average of each of those groups. The global average computed over 193 chips is approximately 205 MHz. We note that the variation of the smaller groups is as high as 6% compared to the global average. In other words, the variation due to a smaller group is comparable in magnitude with the inter-chip variation of the larger group. Clearly, a larger dataset is required to minimize this variation.

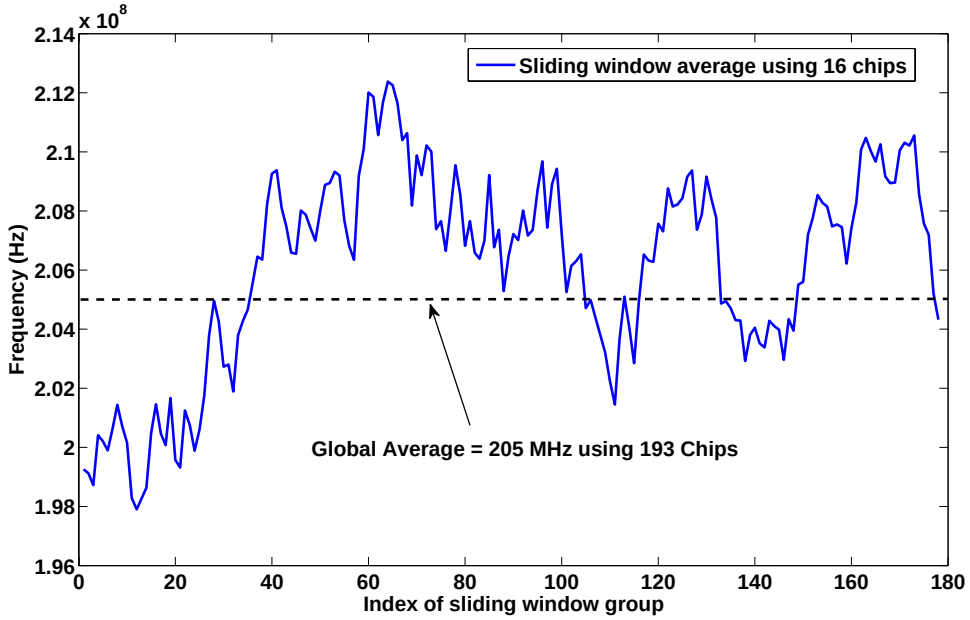

Figure 5.5: Comparison of measurement using large and small sample set

5.2.4 Evaluation of PUF Responses

We evaluate the PUF responses generated from each of the 193 chips. We use three randomness parameters: uniqueness, uniformity, and bit-aliasing as defined in Section 2.4.1. We also evaluate the reliability of the PUF and distinct unreliable bits defined in Section 2.4.2. 511 response bits were extracted from the array of 512 ROs. These 511 bits are extracted by comparing adjacent pair of ROs in the array. We used $f_{i,j}$ introduced in Equation (5.5) for this purpose. Figure 5.6 shows the distribution of the uniqueness among the sample chips. The average is 47.2% with a maximum value of 57.7% and a minimum of 36.9%. The minimum value of 36.9% shows that any two chips will have at least 188 (36.9% of 511) different response bits between them. Hence, the PUF is able to distinguish each chip clearly.

Figure 5.7 shows the uniformity of PUF responses across 193 chips. The average value is 50.5%. The maximum value is 56.9% whereas the minimum value is 45.9%. This shows that the response bits are fairly evenly distributed among '0' and '1'.

Figure 5.8 shows the bit-aliasing for all 511 bit positions across the sample set. The

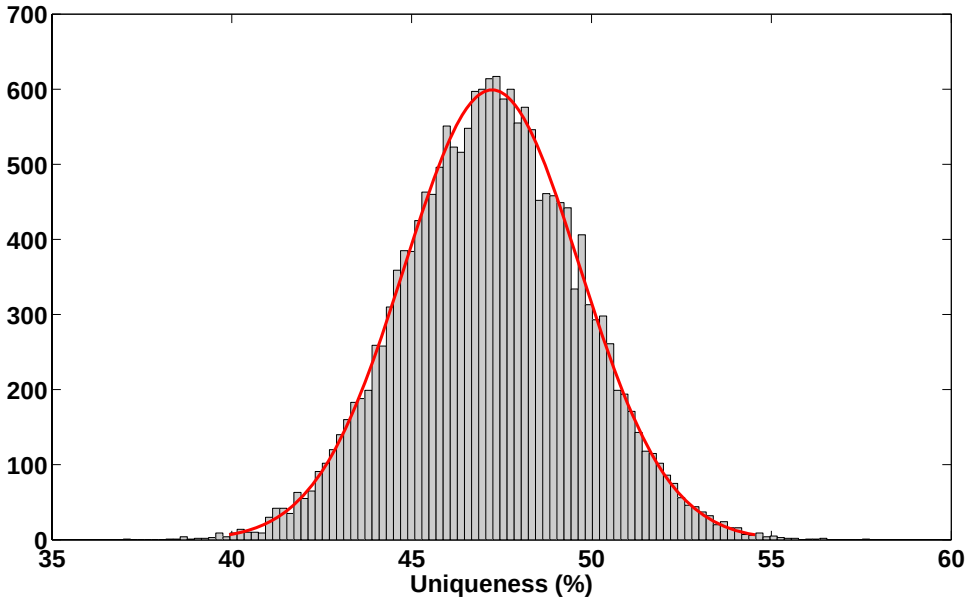

Figure 5.6: Distribution of the uniqueness

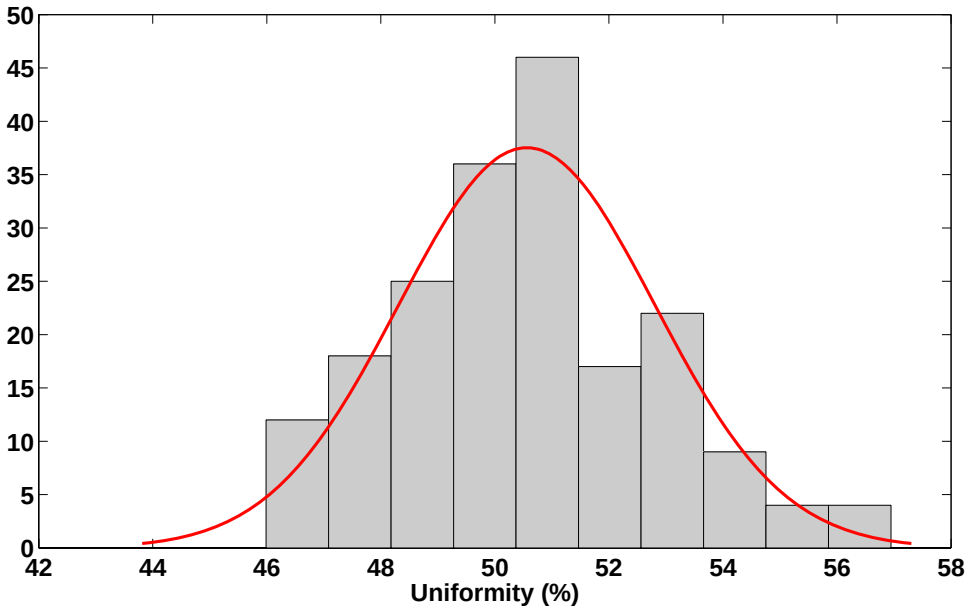

Figure 5.7: Distribution of the uniformity

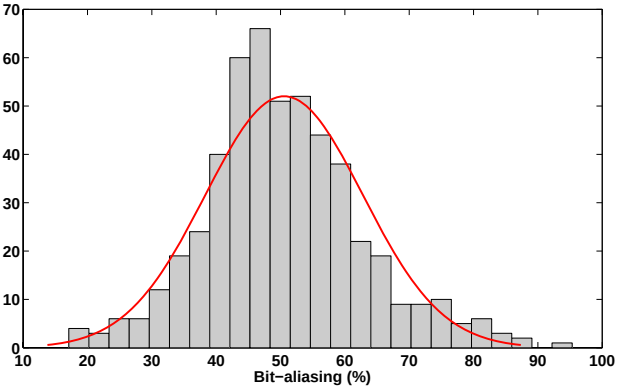

Figure 5.8: Distribution of the bit-aliasing

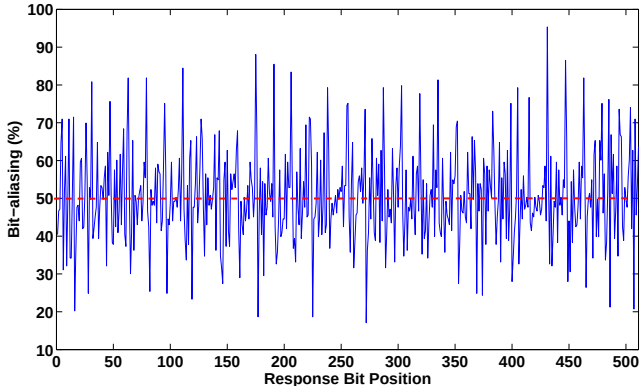

Figure 5.9: Bit-aliasing at different responses

average value of bit-aliasing is 50.5% which is close to the ideal value of 50%. However, there are occurrences of extreme bit-aliasing at a regular interval of the bit positions (the spikes in Figure 5.9). A bit-aliasing value close to 0% at a bit position indicates that almost all the chips produce a '0' bit at that position. On the other hand, a value close to 100% at a bit position indicates that nearly all the chips produce a '1' bit at that position. In both the cases, the corresponding bits do not contain much information that can be used for a security-related application. The maximum value is 95.3% (found at the 431st bit position) which means approximately 184 chips out of 193 produced '1' for the particular bit position. Since the pair of ROs that are compared in the response evaluation are selected from physically adjacent locations in the FPGA, systematic process variation may be ruled out as the cause behind this regular bit-aliasing pattern. Upon closer examination of the RO array, we found that these bits are located at two opposite boundaries of the 2-D RO array. These boundaries are closer to the internal memory of the chip. One of the possible reasons might be the variation in power distribution near the memory cells. However, it requires further investigation to reach a conclusion.

To estimate the reliability, all the 511 bits of the reference response R were derived using the average RO frequencies, $f_{i,j}$ $(1 \leq j \leq 512)$. 100 sample responses (R's) are derived using $f'_{i,j,k}$ $(1 \leq k \leq 100)$. Figure 5.10 shows the distribution of the reliability across the tested chips. The average is 99.13% with a maximum value of 99.61% and a minimum of

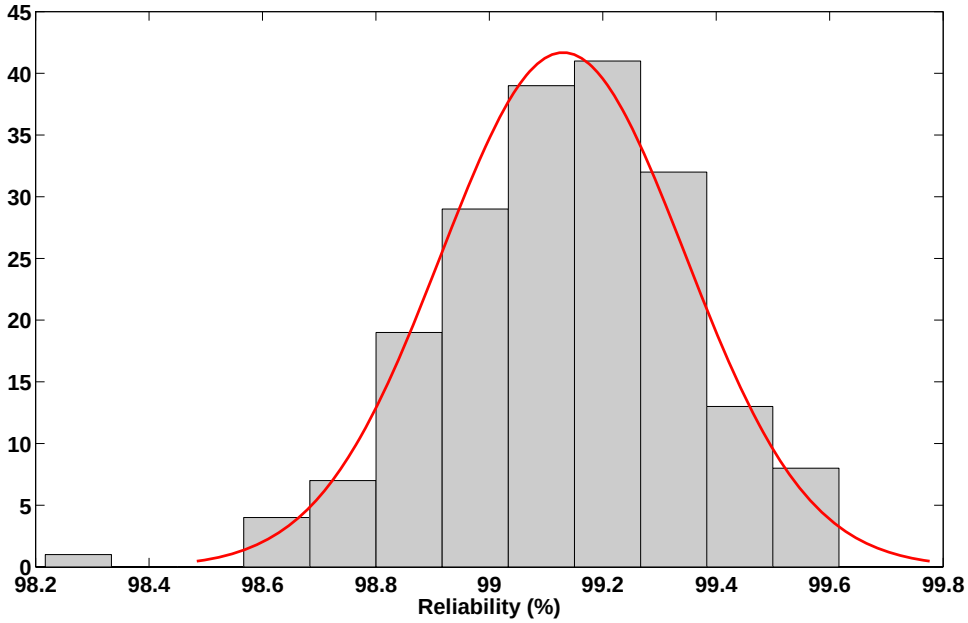

Figure 5.10: Distribution of the reliability

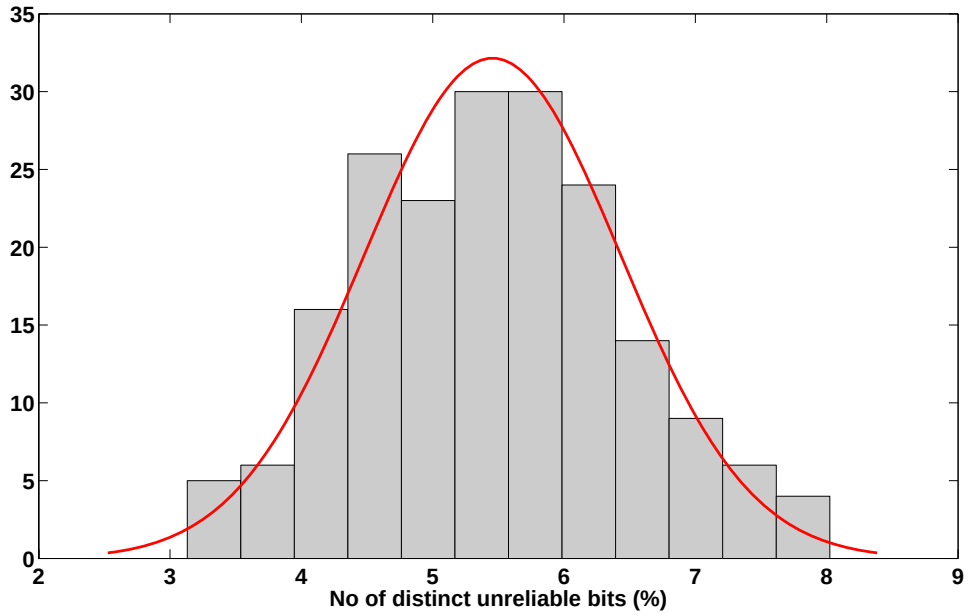

Figure 5.11: Distribution of the distinct unreliable bits

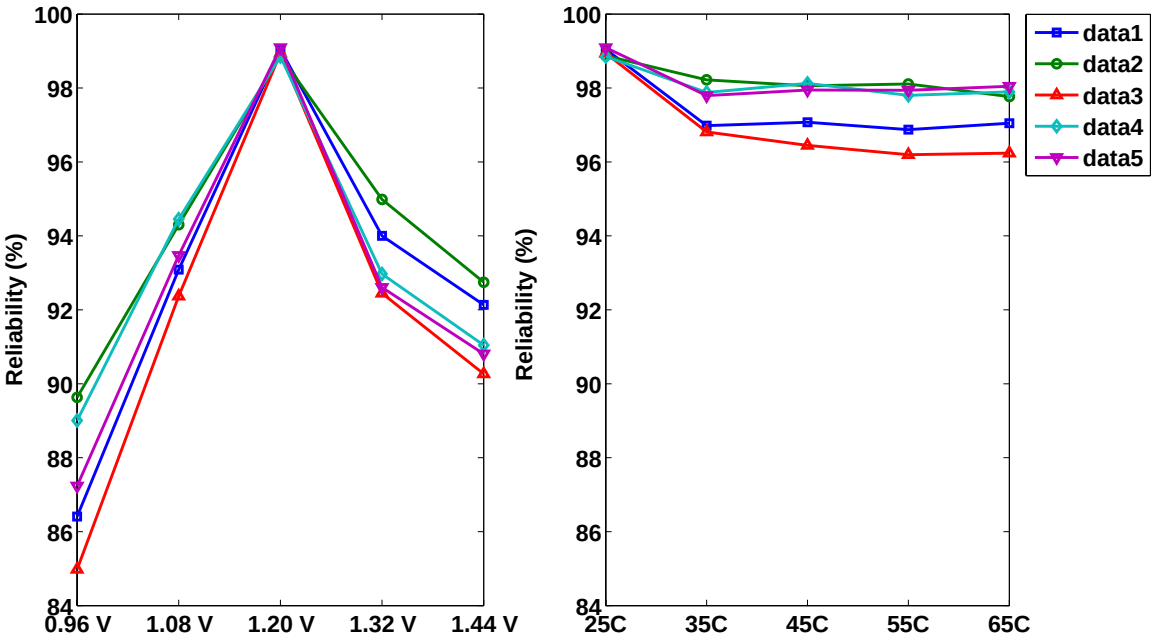

Figure 5.12: Reliability at different values of temperature and core supply voltage

98.21%. This shows that the RO PUF is highly reliable at normal operating condition. This observation is consistent with the result in the previous section where it was observed that the average static intra-chip variation in RO freqiencies is significantly larger than the average dynamic variation or noise, indicating high reliability in the PUF responses. We also evaluated the number of distinct unreliable response bits. Figure 5.11 shows the distribution of the percentage distinct unreliable bits for the whole sample set at normal operating condition. The average is 5.45% that is approximately 28 bits out of 511 bits. The maximum value is 8.02% while the minimum is 3.13%.

We evaluated the reliability at different temperatures and supply voltages for a group of five chips. The normal supply voltage to the core of the used chips is 1.2V. The range of voltage variation was 1.2V±20% with a step of 10%. The ambient temperature was varied from 25°C to 65°C with a step of 10°C. Figure 5.12 shows the result. The voltage variation results in noticeably lower reliability compared to the temperature variation. For the voltage variation, the lowest reliability between the five chips was 84.9%, and it was observed at a

Table 5.1: Distinct unreliable bits (%) due to temperature and voltage variation

	Voltage (0.96 V - 1.44 V)	Temperature (25°C - 65°C)
Chip1	27.9%	10.1%
Chip2	25%	7.6%
Chip3	32.6%	9.9%
Chip4	27.5%	9.1%
Chip5	29.3%	8.4%

supply voltage of 0.96V. For the temperature variation, the lowest reliability was observed to be 96.2% at 55°C and 65°C. Overall, the reliability remains fairly consistent for the entire range of temperature variation, but it varies significantly due to the variation in the supply voltage. In Table 5.1, we show the total number of distinct unreliable bits in the PUF response for both temperature and supply voltage variation for five chips. The value of 27.9% in the second row and the second column in Table 5.1 shows that the 511-bit response bit generated from chip1 contains 27.9% of 511 $\approx$ 143 bits that flipped at least once during the entire range of voltage variation from 0.96V to 1.44V. The overall observation is that the voltage variation causes more unreliable bits in the PUF response than the temperature variation.

5.3 The Effect of Aging on PUFs

In most of the PUF proposals in literature, the main focus has been to maximize the extraction of the PUF entropy (randomness) and to reduce errors (reliability) in the PUF responses. Additionally, error correction in PUF responses has been separately investigated in several works. Maes et al. proposed a soft decision helper data algorithm to rectify errors in the responses from SRAM PUFs [30]. Meng Day et al. used BCH coding and Reed

Muller coding to produce error-free responses from the RO PUF [58]. Yin et al. proposed two methods of correcting errors in the responses from the RO PUF. In the first work, they introduced the idea of temperature-aware cooperative (TAC) method to make the RO PUF robust against temperature variation [55]. Later, they proposed a method called the longest increasing subsequence-based grouping algorithm (LISA) to extract response bits from the RO PUF while improving their reliability [56]. An implementation of an efficient helper data key extraction has been demonstrated by Bosch et al [4].

All these works, including both the new PUF proposals as well as the dedicated error correction efforts, have one thing in common. While solving the error correction, almost all of them focused on the errors introduced by the reversible temporal variations in ICs. Reversible temporal variations in circuit behavior disappear once the cause of the variation is withdrawn. For example, at an elevated temperature, an IC might become slower than normal, but it can restore its normal frequency once it is cooled down to its normal operating temperature. This type of variation is fast and can be observed over a reasonably short period of time. On the other hand, aging is irreversible variations in circuit components over time, leading to permanent shift in the circuit behavior. VLSI phenomena such as negative bias temperature instability (NBTI), temperature-dependent dielectric breakdown (TDDB), hot carrier injection (HCI), and electro-migration are some of the causes of aging [51]. With continuous shrinking of silicon devices, aging is becoming more prominent [28]. Therefore, it is essential to study the effect of aging on PUFs. Kirkpatrik et al. proposed software-based techniques to prevent drift in PUF responses due to aging with the assumption that aging alters the PUF responses; no analysis of the aging effect on the PUF functionality was provided [21]. Though a preventive solution is required in case a PUF is affected by aging, it is equally important to study how aging actually affects the functionality of a PUF. We present a study of the aging effect on a PUF based on on-chip aging experiments as well as simulations. The experimental data is based on the RO PUF implemented on commercially available FPGAs. We have four distinct contributions.

1. We present a detailed analysis of the effect of circuit aging on the functionality of a PUF. The analysis is done in the context of two main PUF application scenarios: device authentication and cryptographic key generation.

2. We performed on-chip accelerated aging experiments to observe the effect of aging on PUFs. This shows how the frequencies of the ROs in an RO PUF change with aging and how that change eventually influences the PUF challenge-response mechanism.

3. We extrapolated the aging effect on a large group of FPGA chips using simulation to estimate how the ability of a PUF to distinguish several chips is influenced by it.

4. Finally, we study how the aging effect on a PUF can be mitigated. We show that redundancy can prevent aging effect by using a configurable ring oscillator architecture.

The experimental results, based on a group of 90-nm Xilinx FPGAs, show that aging makes the PUF responses unreliable. However, the randomness of the PUF responses remains unaffected.

5.3.1 Background on Aging

In this section, we discuss an overview of circuit aging and few related works on the aging effect on PUFs.

Circuit Aging

With continuous usage of ICs, circuit components gradually undergo structural degradation, resulting in hard faults. These faults cannot be rectified and make a chip unreliable to use. Two main types of degradation are: a) oxide wear-out and b) interconnect failure [51].

a) Oxide wear-out - The gate oxide of a transistor wears out due to the following phenomena.

1) Negative Bias Temperature Instability (NBTI) - Due to the applied electric field across the gate oxide in a transistor, dangling bonds are developed at the interface of the channel and the oxide layer. This affects the transistor by increasing the threshold voltage thus making switching difficult. NBTI is enhanced by high temperature and high supply voltage.

2) Hot Carrier Injection (HCI) - When carriers with high energy collide with the gate oxide layer and remain trapped there, the oxide layer is damaged, resulting in alteration of the transistor characteristics. High switching rate of a circuit as well as excess supply voltage enhance this effect.

3) Temperature-Dependent Dielectric Breakdown (TDDB) - Due to the voltage applied across the gate oxide, conduction starts through it using trapped charges, resulting in gradual break-down of the oxide layer. A high operating voltage as well as higher temperature accelerate TDDB.

b) Interconnect failure- Besides the transistors, interconnecting wires too are prone to aging. Electro-migration causes failure in interconnects. Due to high current flow, metal atoms in the wires shift, resulting in faulty connections. This is enhanced by high temperature.

Assessment and prevention of failures in ICs are among the major concerns for the designers. Research has been done to predict the failure of chips as well as to prevent it. Since aging is time dependent and often takes a long period of time to introduce noticeable errors, accelerated aging test has been one of the ways pursued by the researchers to estimate aging. Stott et al. performed accelerated aging tests to estimate how FPGA components such as LUTs, routing wires degrade with time [42]. In this research, we run accelerated aging tests on an FPGA-based RO PUF to observe how the PUF is influenced by aging.

Previous work on the Aging Effect on PUFs

Kirpatrik et al. proposed software-based techniques to prevent the effect of aging on PUFs [21]. They proposed two solutions: a) detection of drift in PUF responses due to aging and updating the affected challenge-response pairs (CRPs) b) prevention of drift in PUF responses by making the lifespan of a CRP short. The proposed solutions employed protocol-level techniques such as Feige-Fiat-Shamir zero-knowledge protocol and Markel Hash tree and did not involve any implementation results. In our work, we mainly focus on estimating the effect of aging on the PUF functionality using on-chip experiments as well as simulations.

Lim et al. briefly mentioned about aging in their work on the Arbiter PUF [24]. They performed a one-month-long aging test on the Arbiter PUF under normal condition without applying increased temperature or excess supply voltage. On the contrary, in this research, we performed an aging test not only under normal condition but also under elevated temperature and excess voltage on the PUF. Additionally, we analyze the effect of aging on PUFs with significant elaboration.

Guajardo et al. also discussed about the robustness of the SRAM PUF against aging [14]. However, the PUF robustness was evaluated against a specific case of continuous writing of ones and zeros in SRAM cells. Aging of the SRAM PUF was not done under severe operating conditions such as increased temperature and excess supply voltage. We instead evaluated the aging effect on PUFs caused by the traditional VLSI degradation factors using voltage and temperature stress on the RO PUF.

5.3.2 PUF Functionality and Aging

In this section, we analyze how the functionality of a PUF could be possibly affected by aging. We propose a hypothesis that is based on the functional characteristics of a PUF. We consider two primary application scenarios of a PUF and point out the implications they will have if aging has any influence on a PUF. The scenarios are: a) device authentication

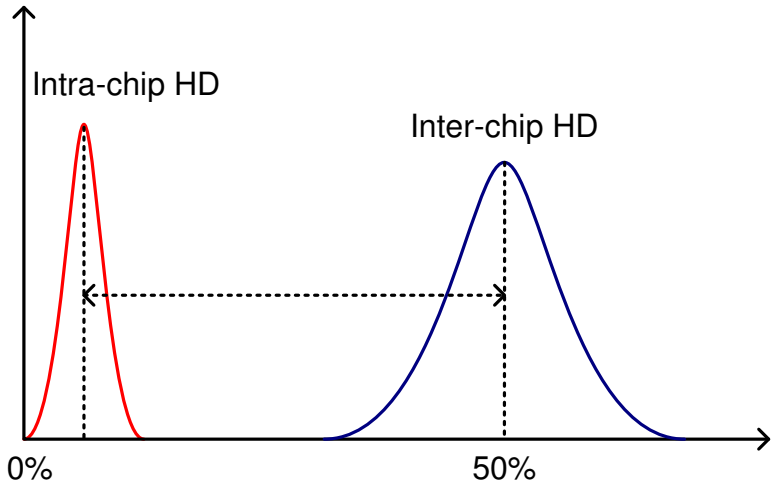

Figure 5.13: Basic PUF functionality

and b)cryptographic key generation.

The uniqueness of a PUF is estimated as the average inter-chip Hamming distance (HD) as we defined it in Equation (2.1). The reliability of a PUF is a function of its average intra-chip HD; it has been introduced in Equation (2.5). We discuss the effect of aging using the inter-chip HD and the intra-chip HD of a PUF.

Figure 5.13 shows an example of the distributions of the intra-chip HD and the inter-chip HD of a PUF [1]. Ideally, for a truly random PUF, the inter-chip HD is centered around 50%. For minimal error, the intra-chip HD distribution should be centered near 0%. We now discuss how the two applications mentioned earlier could be affected by aging.

a) Device Authentication - For an error-free authentication, the two distributions shown in Figure 5.13 should not overlap with each other. For high reliability, the locations of these two distributions should be as far as possible. If they shift closer to each other (shown by the dashed curves in Figure 5.14) and eventually have an overlap, there will be errors in the authentication. There are two types of error:

1)False negative - If the responses of a chip deviate significantly from its reference eval-

[1]The figure represents a hypothetical case. It is not based on real data.

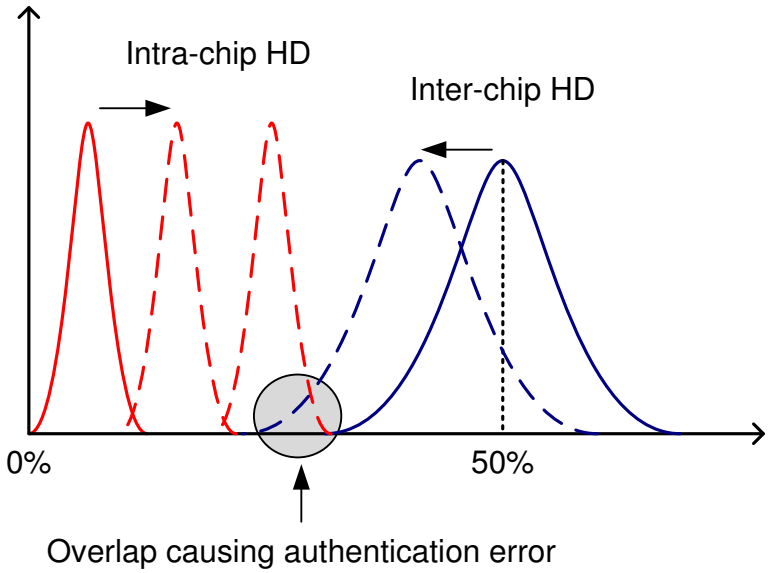

Figure 5.14: Possible effect of aging on the functionality of PUF

uated during the enrollment, it might be deemed as a different chip and rejected during authentication, resulting in a false negative.

2)False positive - On the other hand, if a chip's responses drift from its reference and become nearly equal to some other chip's responses, it might be accepted with the other chip's identity. This is called false positive. It is useful to mention that when multiple chips produce similar responses, bit-aliasing occurs, and the randomness of the PUF goes down. This is because if a particular response bit produces the same binary value across multiple chips, it becomes deterministic and cannot be used any more as a source of randomness/entropy.

Our hypothesis is that if the circuit aging affects the functionality of a PUF, it will result in a time-dependent shift of the intra-chip as well as the inter-chip distribution. For a sizable population of chips, the average inter-chip HD cannot exceed 50% [14, 45]. Therefore, the inter-chip HD distribution cannot move towards its right on the x-axis in Figure 5.14 and can only shift towards the left. On the other hand, the intra-chip HD distribution being non-negative can only move towards the right. Both the constraints lead to a possible error in authentication. With our experiments, we intend to observe whether a shift takes place

or not. It is important to emphasize that this shift due to aging is a one-way process and cannot be reverted if takes place.

b) Cryptographic Key Generation - In cryptographic applications, the keys formed by the response bits of a PUF must be completely error-free. This is because a cryptographic algorithm is supposed to generate drastically different outputs from an input for a pair of keys that differ even by a single bit. However, PUF-generated responses are noisy by nature and cannot be reproduced exactly over multiple samplings. This is the reason why the intra-chip HD may not be zero in most of the cases in practice. To minimize the noise, researchers have employed several error correction methods to produce error-free PUF keys. Implementing an error-correction scheme does not come for free, and the cost depends on how many bits need to be corrected by it. This is aggravated if the intra-chip HD distribution shifts towards its right (refer to Figure 5.14) due to aging. Therefore, estimating errors in the PUF responses due to aging as well as mitigating it are important.

As a summary, our hypothesis is that the characteristic functionality of a PUF may not remain static over time as circuit aging may introduce a time-dependent shift in its behavior. Using an accelerated aging test as well as simulation, we want to validate the hypothesis and estimate how severe the effect of aging can be.

5.3.3 Experimental Results

In this section, we present the result of the accelerated aging test carried out on the RO PUF. We also analyze our simulation method and its results.

Experimental setup for the Accelerated Aging Test

An RO PUF with a group of 512 ring oscillators in a 32×16 array form was implemented on a Xilinx Spartan 3E FPGA (XC3S500E). As mentioned in Section 5.3.1, the aging phenomena are primarily caused by elevated temperature and excess supply voltage. Hence, we enhanced

the process of aging by applying stress in terms of elevated temperature and excess supply voltage to the core of the FPGA chips. The FPGAs were heated using a convection-based heating oven, and the supply to the FPGA core was varied using a regulated DC power supply.

We applied four different types of stresses on the FPGAs:

- Oscillation-only stress - Each of the ROs in the array were activated simultaneously for an extended period of time at normal operating condition i.e. without any temperature or voltage stress.
- Voltage only stress (V) - The FPGA under test was operated at 1.5 V, 1.8 V, and 2 V in three phases. The normal supply voltage for the FPGA used is 1.2 V.
- Temperature only stress (T) - The FPGA under test was operated at 70°C and 80°C in two phases. The maximum allowed temperature for the FPGA device used is 85°C. We kept a safety margin of 5°C.
- Temperature + Voltage stress (T+V) - In this stress condition, the FPGA underwent three different phases: 70°C and 1.5 V, 80°C and 1.8 V and finally, 80°C and 2V.

Four separate FPGAs with the same model was used for these four stress conditions as shown in Table 5.2. For each of the T, V, and T+V stress conditions, each of the 512 ROs were activated simultaneously.

The Effect of Aging on PUFs

First, we show the change in RO frequencies over the course of aging. We define the average change in the RO frequencies w.r.t to the pre-aging condition as:

$$\Delta f_{\text{age},i} = \frac{1}{512} \sum_{j=1}^{512} \frac{f_{\text{avg},i,j}|_{t=x} - f_{\text{avg},i,j}|_{t=0}}{f_{\text{avg},i,j}|_{t=0}} \times 100\% \tag{5.9}$$

Table 5.2: Different cases of accelerated aging

	Room temp	70°C	80°C
1.2 V	FPGA1	FPGA2	FPGA2
1.5 V	FPGA3	FPGA4	-
1.8 V	FPGA3	-	FPGA4
2.0 V	FPGA3	-	FPGA4

$f_{\text{avg},i,j}|_{t=0}$ is the average frequency of the j-th RO of the i-th FPGA before the start of the aging experiment, and $f_{\text{avg},i,j}|_{t=x}$ is that at time $t = x, x > 0$ and $f_{\text{avg},i,j} = \frac{1}{100}\sum_{k=1}^{100} f_{i,j,k}$ where $f_{i,j,k}$ is the k-th sample frequency of the j-th RO of the i-th FPGA. Figure 5.15 shows $\Delta f_{\text{age},i}$ for the four FPGAs stressed under different conditions at two different point of aging: 200 hours and 400 hours. We observe that the reduction (-ve y-axis) in frequency for the oscillation-only stress and the T stress are negligible compared to the other two cases. Therefore, we analyze the cases of V stress and T+V stress only. Figure 5.16 and 5.17 shows the variation in the frequencies of 512 ROs for the V stress and T+V stress respectively. The total period of aging was 1100 hours including three recovery intervals when the chips were kept out of stress. On the x-axis, the 0th hour represents the condition before the accelerated aging started. By comparing these two figures, we see that the reduction in frequency is more in case of T+V stress compared to the case of V stress. At the end of 1100 hours of aging, the RO frequencies reduced by nearly 9.5% under T+V stress, while the amount of reduction is 6.6% under V stress. We deliberately allowed the FPGAs to remain under stress-free condition intermittently to see if the chip recovers. An increase in frequency, though small, at the end of the recovery period implies that the V and T stress induce temporary changes besides permanent shifts. Overall, we see that there is a significant reduction in RO frequencies due to aging. All the FPGAs were found to be operational at the end of the aging test.

Reliability of PUF due to aging - 511 response bits were generated from the RO

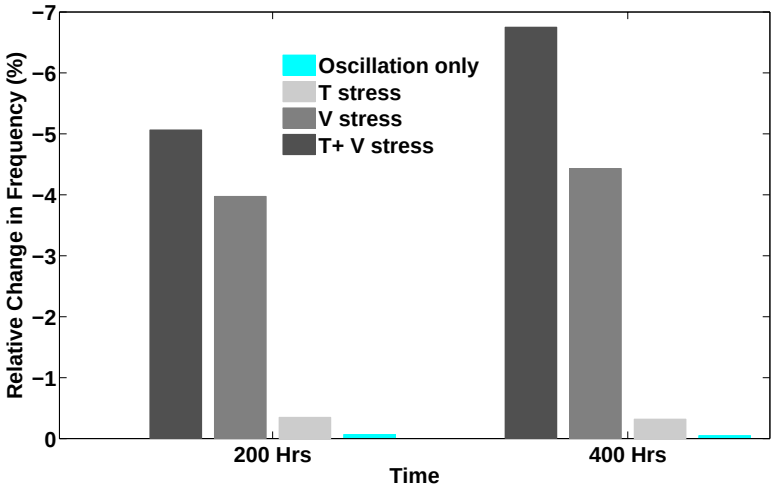

Figure 5.15: Change in RO Frequencies for different stresses

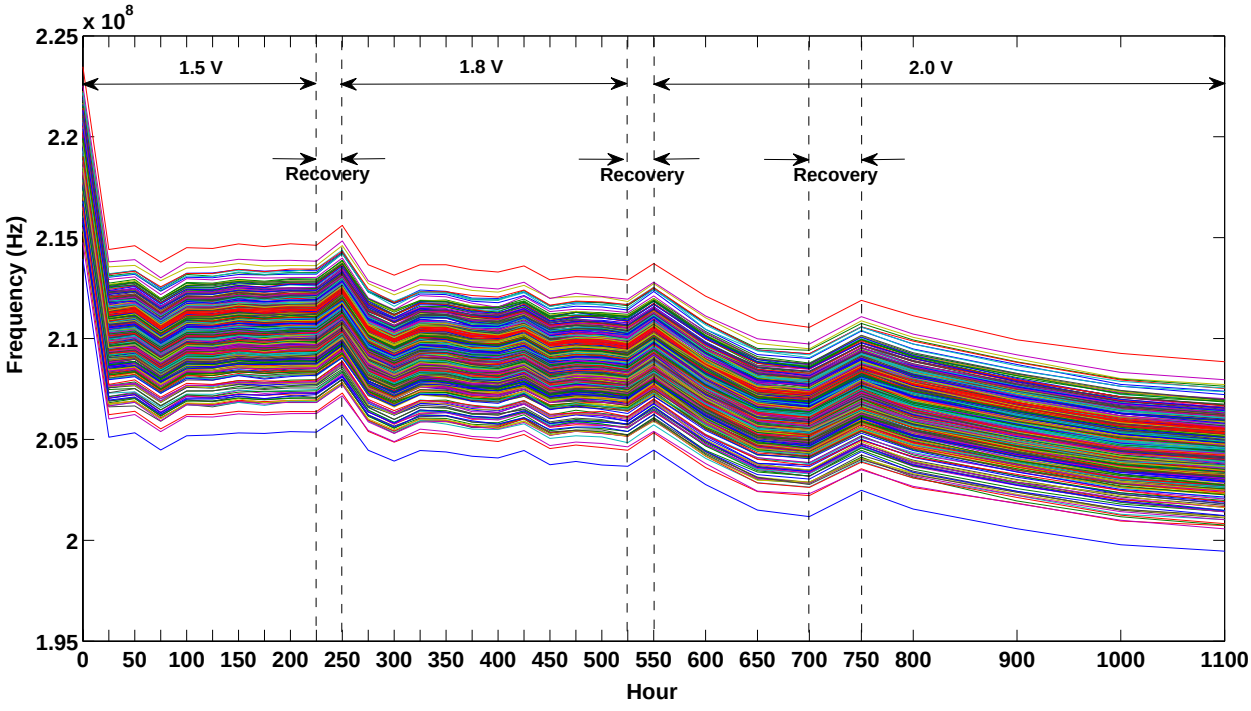

Figure 5.16: RO Frequency variation with aging under V stress

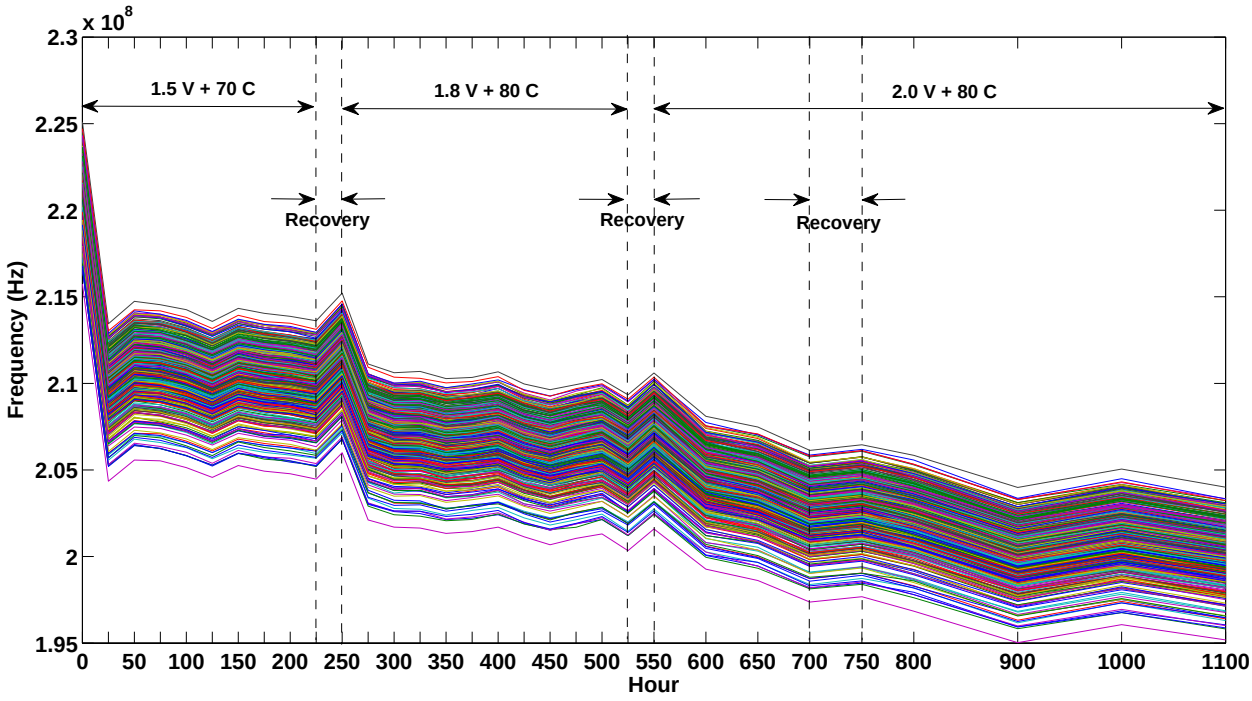

Figure 5.17: RO Frequency variation with aging under T+V stress

PUF using the simple comparison method. These responses were used to evaluate the PUF parameters.

We calculated the reliability using Equation (2.5) for the V and T+V cases as shown in Figure 5.18. It is clear that the reliability for both the cases has decreased noticably compared to the pre-aging condition. This means aging causes more unreliable bits in PUF, resulting in the shift of the intra-chip HD distribution towards the inter-chip HD distribution.

Under T+V stress, the number of unreliable bits increased from ≈ 5 ($\approx 1\%$) to ≈ 44 ($\approx 8\%$). This is a significant reduction in the reliability of the PUF from 99% to 92%. It can be noticed that the reliability improved slightly on the first two instances of recovery interval. However, it degraded at the last instance of recovery between 700th hour and 750th hour. For the V stress, the the reliability reduced to 95% at the end of 1100 hours of aging.

Uniqueness of PUF based on simulated aging - The uniqueness or average inter-chip HD of a PUF is a population-based parameter and requires a group of chips for estimation.

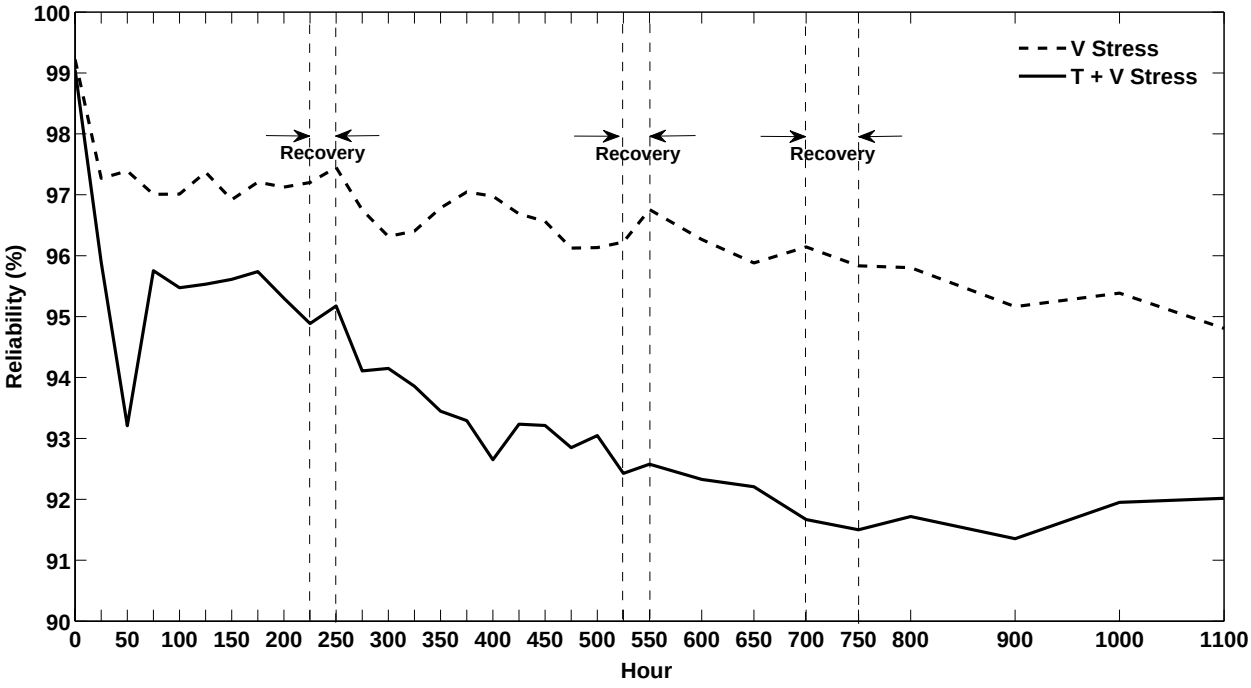

Figure 5.18: Change in reliability of PUF with aging

Practically, it is not an easy task to age a significant number of chips as it would require a long period of time as well as money. Therefore, we employed a simulation technique to extrapolate the aging effect on a large group of FPGAs that were measured using the RO PUF under normal condition. We decompose the RO frequency as:

$$f_{i,j} = f_{\text{nom},i,j} + \Delta f_{\text{pv},i,j} + \Delta f_{\text{age},i,j} + \Delta f_{\text{noise},i,j} \tag{5.10}$$

$f_{\text{nom},i,j}$ is the nominal characteristic frequency of the j-th RO on the i-th FPGA and does not change with aging. $\Delta f_{\text{pv},i,j}$ is the deviation due to manufacturing process variation and also remains static over time. $\Delta f_{\text{age},i,j}$ is the aging factor and is time-dependent. $\Delta f_{\text{noise},i,j}$ is the variation induced by the noise factor. Experimental data shows that this factor also changes with time.

As described in Section 5.2, we have measured a group of 193 ($1 \leq i \leq 193$) Xilinx Spartan3E S500 FPGA chips using the RO PUF with 512 ROs ($1 \leq j \leq 512$) at normal

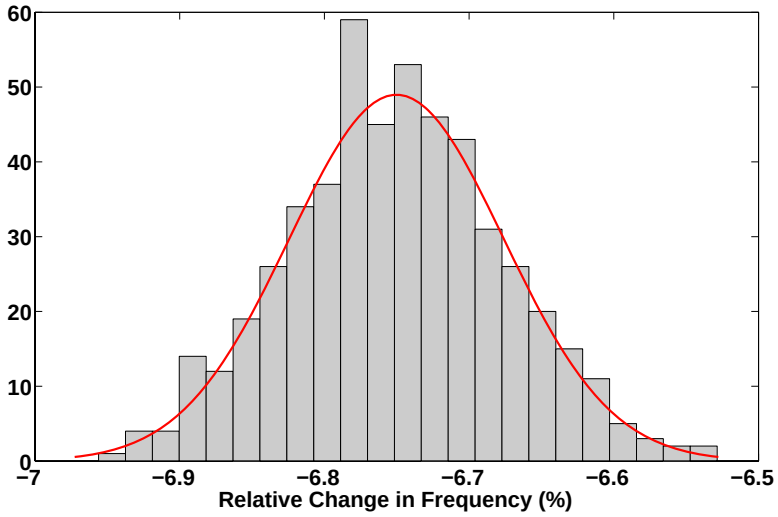

Figure 5.19: Aging distribution after 400 Hrs with T+V stress

condition without any stress.[2] We have collected 100 samples for each of the RO frequencies and calculated an average to remove the noise to get $f_{\text{nom},i,j} + \Delta f_{\text{pv},i,j}$. Now, we simulate $\Delta f_{\text{age},i,j}$ and $\Delta f_{\text{noise},i,j}$ using the aging as well as the noise distribution of the FPGAs that went through aging. Here, we assume that all the chips undergo similar changes due to aging under the same stress condition.

First, at $t = x$ $(x > 0)$, we calculated the parameter $\frac{f_{\text{avg},i,j}|_{t=x} - f_{\text{avg},i,j}|_{t=0}}{f_{\text{avg},i,j}|_{t=0}}$ for each of the 512 ROs for the FPGA under stress. By plotting these values, we observed that they follow a normally distributed pattern. One example case for the stress condition T=80°C + V=1.8V is shown in Figure 5.19. We also verified that these values do not have any correlation with the location of the ROs on the FPGA. Using the mean and the standard deviation of these values, we randomly generated a set of 512 normally distributed values as Δf_{age} for each of the 193 FPGAs using Matlab and added them to the pre-aging average frequency values i.e. $f_{\text{nom},i,j} + \Delta f_{\text{pv},i,j}$, resulting in $f_{\text{nom},i,j} + \Delta f_{\text{pv},i,j} + \Delta f_{\text{age},i,j}$.

For $\Delta f_{\text{noise},i,j}$, we first derived the noise factor of an RO at $t = x$ as the standard deviation of the RO frequency over 100 samples: $\sqrt{\frac{1}{99}\sum_{k=1}^{100}(f_{i,j,k}|_{t=x} - f_{\text{avg},i,j}|_{t=x})^2}$. We calculated it

[2]This data is from on-chip measurements and not from simulation.

Table 5.3: Uniqueness of PUF due to aging

	Avg	Min	Max	Std
Pre-Aging	47.24%	36.99%	57.72%	2.44%
V:200 Hrs	47.24%	36.20%	58.31%	2.45%
V:400 Hrs	47.30%	37.57%	56.94%	2.43%
V:600 Hrs	47.24%	38.36%	56.75%	2.46%
V:800 Hrs	47.27%	37.77%	56.95%	2.44%
V:1000 Hrs	47.31%	38.34%	56.56%	2.46%
T+V:200 Hrs	47.28%	37.96%	57.53%	2.45%
T+V:400 Hrs	47.25%	37.57%	59.30%	2.45%
T+V:600 Hrs	47.28%	36.98%	56.56%	2.44%
T+V:800 Hrs	47.27%	37.77%	60.27%	2.44%
T+V:1000 Hrs	47.32%	37.96%	57.14%	2.46%

for each of the 512 ROs of the stressed FPGA. We observed that this factor also shows normally distributed pattern when plotted. Using the mean and standard deviation of these noise factors, we generated a set of 512 normally distributed $\Delta f_{\text{noise},i,j}$ values for each of the 193 FPGAs using Matlab. We further created 100 samples of $\Delta f_{\text{noise},i,j}$ using $\Delta f_{\text{noise},i,j}$ as the standard deviation for a zero-mean normally distributed variable. These simulated $\Delta f_{\text{noise},i,j}$ values are finally added to the previously computed $f_{\text{nom},i,j} + \Delta f_{\text{pv},i,j} + \Delta f_{\text{age},i,j}$ values to complete the simulation of $f_{i,j}$ in Equation (5.10). We derived response bits using the simulated frequency values and then estimated the uniqueness for five time instances: at t=200 hours, t=400 hours, t=600 hours, t=800 hours and t=1000 hrs of aging. It is done for each of the T and T+V cases. Table 6.1 shows the average, minimum, maximum, and standard deviation values of the uniqueness distributions for all the cases. It is very consistent across different instances of aging. One of the possible reasons might be that the bit-flips in the PUF responses due to aging is random. In order to verify it, we evaluated the

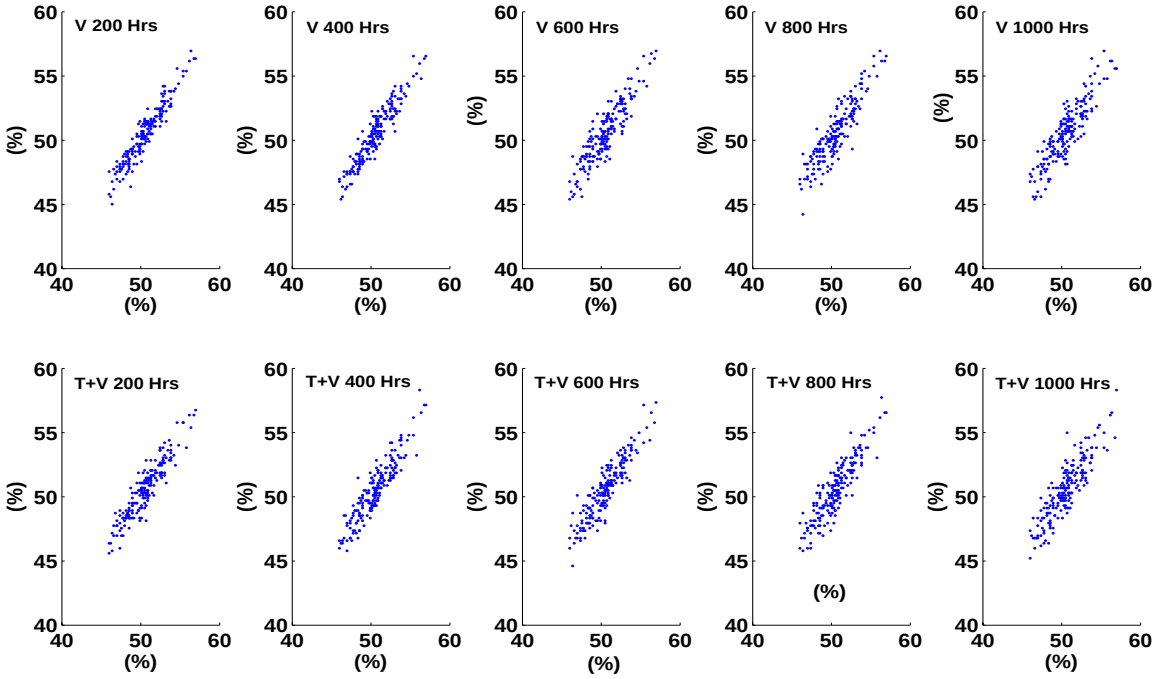

Figure 5.20: Uniformity of PUF due to aging (aged vs original)

uniformity as well as the bit-aliasing of the PUF. Figure 5.20 shows the scatter plot for the uniformity for each of the cases of aging simulation with respect to the original (pre-aging) condition. A 1:1 trend shows that aging does not cause noticeable change in uniformity. This shows that, on an average, the number of response bits that flipped from '0' to '1' during aging is equal to those that flipped from '1' to '0'. In other words, the change in PUF responses due to aging is random. A similar trend in Figure 5.21 for the bit-aliasing factor indicates that PUF randomness is not affected by aging.

5.3.4 Security Risks and Countermeasures

We observed that by stressing a PUF using excess voltage and temperature, a significant change in the reliability can be made. This gives the adversaries an opportunity to attack a PUF using temperature and voltage stress. Suppose, in an application, a chip collects critical data from the field and sends them to a server. Now if the chip is authenticated

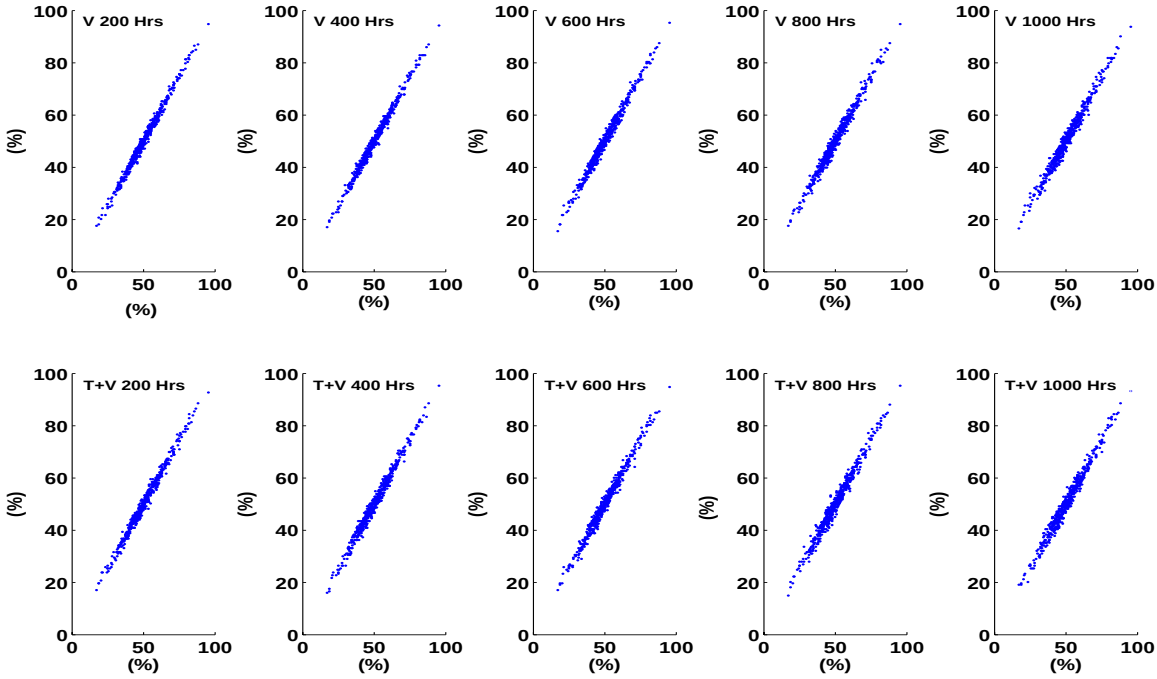

Figure 5.21: Bit-aliasing of PUF due to aging (aged vs original)

using a PUF, an adversary might attempt to permanently change the property of the PUF by applying heat or excess supply voltage. This way the chip may fail to authenticate itself preventing the critical data to be supplied to the server. Moreover, replacing a damaged chip may not be straightforward and may incur significant cost depending on the nature of the application.

In fact, the use of excess temperature and voltage as an active means of attack against embedded systems has been discussed previously [12]. This type of attack against the RO PUF can be thwarted with two types of countermeasures: a) active and b)passive. As an active method, a detection mechanism needs to be implemented and necessary preventive actions need to be taken as proposed in [21]. As a passive countermeasure, if we use a simple comparison method to create a response bit, the redundancy technique (proposed by Suh et al.[45]) that selects an RO pair with maximum frequency difference is a possible solution. Another passive technique could be to employ a different quantization method to produce a PUF response instead of the simple comparison method to break the dependency of a PUF

response on the frequency difference between a pair of ROs.

In the next section, we show a mitigation technique against aging with the help of configurable ring oscillator.

5.3.5 Aging Mitigation using Configurable Ring Oscillator

Suh et al. mentioned that the rate at which the frequency of a ring oscillator A changes with temperature may not be same as that of another ring oscillator B [45]. This is also true for the aging scenario. Had the rate of frequency change of the oscillators been the same, there would not have been any bit flips in the PUF responses. However, Figure 5.18 clearly shows that is not the case. A significant number of PUF response bits have flipped during the aging testing indicating the fact that many RO pairs have different rate of frequency change due to aging.

The reliability of a response bit depends on the magnitude of the difference between a pair of ring oscillators. The higher the difference the lower is the probability that the response bit produced by them will change its value during the evaluation phase. One way of solving the bit flipping is to use redundancy in terms of ring oscillators. In this scheme, a pair of ROs out of a group of k is chosen in such a way that the difference in frequency between them is maximum. Remaining $k-2$ ROs are not used. We proposed configurable ring oscillator(CRO) in section 4.2 to implement this redundancy scheme to improve the reliability of PUF against variation in operating conditions. A CRO loop can configure 2^m oscillating loops using m configuration inputs. Figure 4.8 shows a construction of a CRO with 3 inputs. It can configure eight oscillating loops.

We replaced the regular RO loop with a CRO and tested it under the same stress condition as the regular one. We evaluated the reliability of the PUF with CRO. Figure 5.22 shows a comparison of the PUF with regular ROs vs the one with CROs in term of the reliability for V and T+V stress. It shows that the PUF with CRO improves the PUF reliability

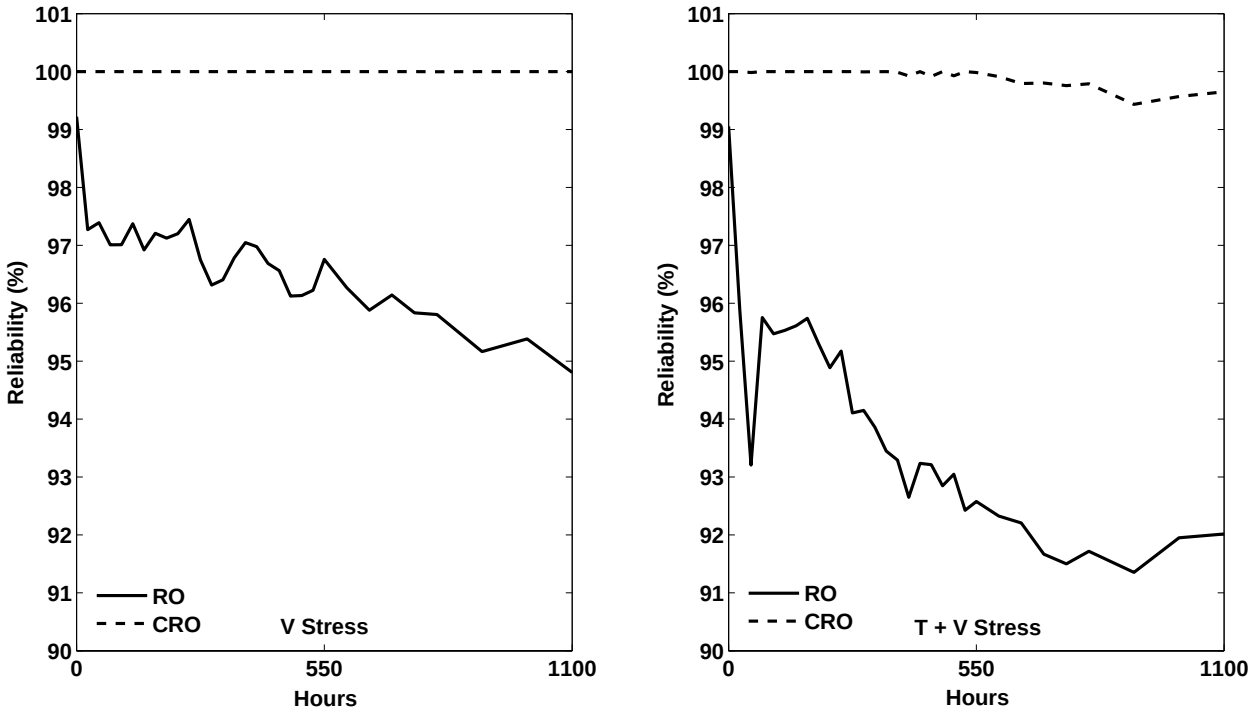

Figure 5.22: Reliability comparison between RO and CRO against aging

significantly. For the V stress, the reliability is 100%. For the T+V stress, the reliability degrades by less than 1% after 1100 hours of aging.

5.4 Summary

We characterized the RO PUF over a significantly large group of chips. The results show that PUF responses are fairly uniformly distributed with high uniqueness. The high ratio of static variation to the dynamic variation conforms to high reliability of PUF responses. Through this experiment, we are able to do a feasibility study of a PUF over a large group of chips. We also presented characterization results under temperature and voltage variation.

Furthermore, we studied the effect of aging on PUFs based on an accelerated aging experiment on an FPGA-based RO PUF implementation. Experimental results show that the reliability of PUF goes down with aging. We extrapolated the aging effect on a large group

of FPGA chips using simulation to estimate how the uniqueness of PUF behaves. Results based on a group of 90-nm FPGAs show that the uniqueness of PUF remains unaffected by aging. It also indicates that aging does not cause any loss in the randomness of the PUF. A mitigation technique using configurable ring oscillator shows significant improvement in reliability.

Chapter 6

PUF Evaluation and Comparison

Since the inception of the idea of PUFs, different types of PUFs have been proposed so far. The availability of several different PUFs gives us choices to select a particular one suitable for an application. However, it also raises a few practical questions: how do we know if a PUF is efficient or not? How do we compare one PUF with another? Currently, there is no commonly accepted method to fairly compare different PUFs. A concrete as well as easy-to-use evaluation-comparison method will be useful for a designer who may want to employ a PUF in her design. Armknecht et al. expressed the same view in their work on formalizing the security features of PUFs [1]. In this research, we propose a systematic method to evaluate the performance of PUFs and to make a fair comparison among them. As part of this work, we have identified the following goals:

First, we need to clearly define the parameters that will quantify the performance of a PUF in a concrete manner. In order to do that, we not only propose our own PUF parameters but also explore several other parameters defined by other researchers. No analysis has been done so far to compare these parameters in order to estimate how effective they are in evaluating performance of a PUF. It might be possible that many of these parameters are similar in nature or redundant. We aim to find any such cases to define a compact set of PUF parameters while removing redundancy.

Second, our goal is to make the comparison method independent of the underlying PUF technique. For example, it should be able to compare a delay-based PUF like the RO PUF [45] with a memory-based PUF such as the SRAM PUF [14]. Therefore, we focus on the statistical properties of the binary PUF responses. This is possible because every PUF produces binary responses (or responses can be converted to binary form) irrespective of the underlying technique. With these goals in mind, we make the following contributions in this research.

- We first propose three measurement dimensions of a PUF. They are : device, time, and space. The PUF performance parameters will be defined along these dimensions. We explain the significance of these dimensions in detail and show how several PUF parameters can be defined based on them.

- We propose that a set of m parameters be used to evaluate and compare the performance of different PUFs while each PUF may have different number of challenge and response bits. This is possible as the parameters rely only on the statistical properties of the binary PUF responses. A simple view of the idea is presented in Figure 6.1. As a preliminary effort, we propose seven parameters: uniqueness, reliability, randomness, steadiness, bit-aliasing, diffuseness, and probability of misidentification. We have defined some of these parameters while others have been selected from the works done by other researchers. We explain in detail why these parameters are useful in evaluating the performance of PUFs.

- Finally, we compare two different PUFs: the RO PUF and the Arbiter PUF using the above mentioned parameters. We used measured PUF data for this comparison. A detailed comparison result is presented in this paper to validate the proposed method.

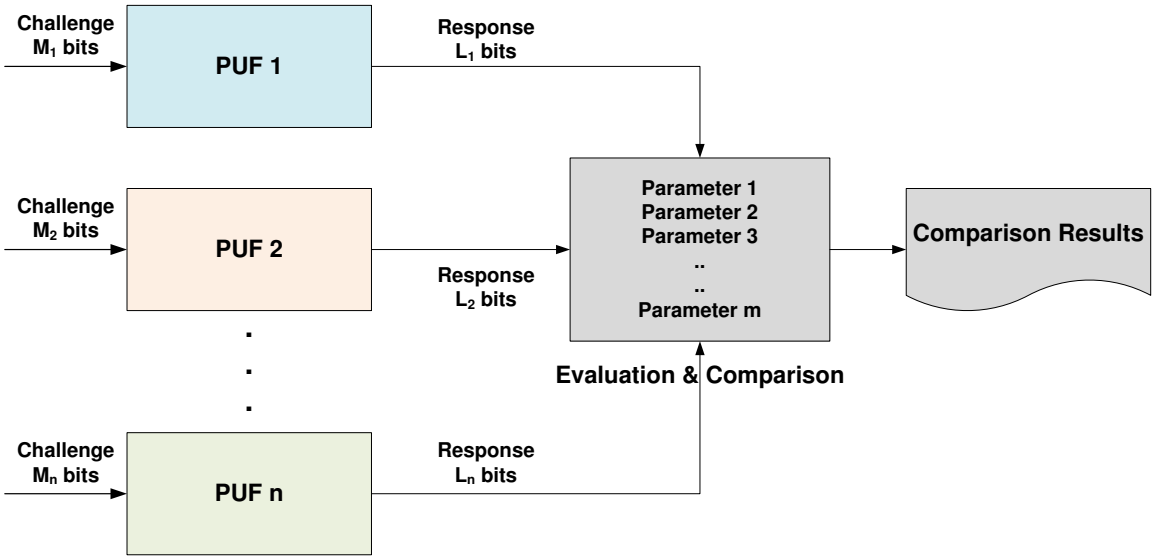

Figure 6.1: The basic idea of a PUF evaluation and comparison method

6.1 Related Work

We discuss related research on the performance measurement of PUFs. Majzoobi et al. proposed several parameters to test the security of PUFs [35]. They tested three security properties of a PUF: predictability, sensitivity to component accuracy, and susceptibility to reverse engineering. Two variants of the Arbiter PUF, linear and feed-forward, were tested. However, no comparison of different PUFs were made.

In another work, Armknecht et al. formalized three properties of a PUF: robustness, unclonability, and unpredictability [1]. The analysis result presented in this work is based on the SRAM-based PUF proposed by Guajardo et al. [14]. No comparison between different PUFs was presented in this work.

Van der Leest et al. tested the performance of a D flip-flop-based PUF (DFF PUF) implemented on ASIC using several parameters [48]. This PUF was originally proposed by Maes et al. using flip-flops in FPGAs [29]. This work applied Hamming weight test, inter-chip uniqueness test, and NIST randomness test on the PUF responses. This work presented a comparison between the SRAM-based PUF, the Butterfly PUF [22], and the DFF PUF.

However, this comparison did not use the statistical properties of the PUF responses except the entropy. Apart from the entropy, the comparison was done based on the number of gates used to implement the PUF, the platform used (ASIC/FPGA), and the spread on a die.

A comprehensive performance evaluation of the Arbiter PUF has been done by Hori et al [18]. In this work, several PUF parameters were defined systematically and were validated based on a large set of measured PUF data. The PUF parameters proposed by this work are uniqueness, randomness, correctness, steadiness, and diffuseness. We have studied this work as a part of this research. We will discuss this work in detail in the subsequent sections. However, this work also did not present any comparison analysis on different types of PUFs.

The work by Maes et al. presented a detailed comparative analysis between several PUFs [31]. One of the comparisons presented in this work was made using several parameters of PUFs such as evaluability, uniqueness, reproducibility, physical unclonability, mathematical unclonability, unpredictability, one-way-ness, and tamper evidence. Another comparison analysis, presented in this work, includes randomness, challenge-response mechanism, CRP space, implementation platform, average inter-chip and average intra-chip variation in PUF responses, entropy, tamper-evidence, and model building as the comparison metrics. The results presented for the CRP space, the entropy, the inter- and intra-chip variation were quantitative while the others were qualitative.

Our contribution in this research is different in many ways from the related work discussed. First, we define a basis for the PUF performance measurement/evaluation. We propose three measurement dimensions for this purpose. The PUF measurement dimensions have not been formally defined before. We explain how these dimensions capture useful information in order to evaluate the performance of PUFs. We then define several parameters using the proposed dimensions to evaluate the statistical property of the PUF responses. Hori et al. defined their PUF parameters using a similar approach, but they did not explicitly define the dimensions [18]. Second, we analyze several PUF evaluation parameters to identify any redundancy that may exist among them. Based on our analysis, we propose a compact set of parameters

for PUF evaluation as well as comparison. We did not find any work in the literature that has made a similar effort. Finally, we compare two different PUFs: the RO PUF and the Arbiter PUF. Unlike previous efforts, our comparison is done entirely quantitatively using the parameters we proposed.

6.2 PUF Evaluation and Comparison Method

In this section, we first introduce the PUF measurement dimensions. Then we discuss the four PUF parameters: uniqueness, uniformity, bit-aliasing, and reliability. They have already been introduced in the background section. In this section, they are mapped along the proposed measurement dimensions of a PUF. Next, we analyze few parameters defined by other researchers. We compare them with our proposed parameters. Finally, we propose a set of parameters as the components of the evaluation-comparison method.

6.2.1 PUF Measurement Dimensions

Figure 6.2 shows three different dimensions of PUF measurement along three axes: device, space, and time. The inter-chip variation in PUF responses is captured using the device axis. The two other axes are used to capture the intra-chip variation.

The device axis represents the population dimension of PUF measurements. A PUF not only needs to generate random responses for a chip, the generated responses also need to uniquely identify the chip among several other chips of the same type. To estimate this property, one needs to measure a set of PUF instantiations on several chips/devices. Hence, we included the device axis. A group of k devices have been shown in Figure 6.2 to represent a population. The space axis stands for the location of a single-bit response r in an m-bit response string R. The rationale behind naming it the space axis is that the PUF response bits are generated at different physical locations on a chip. For example, in the SRAM PUF,

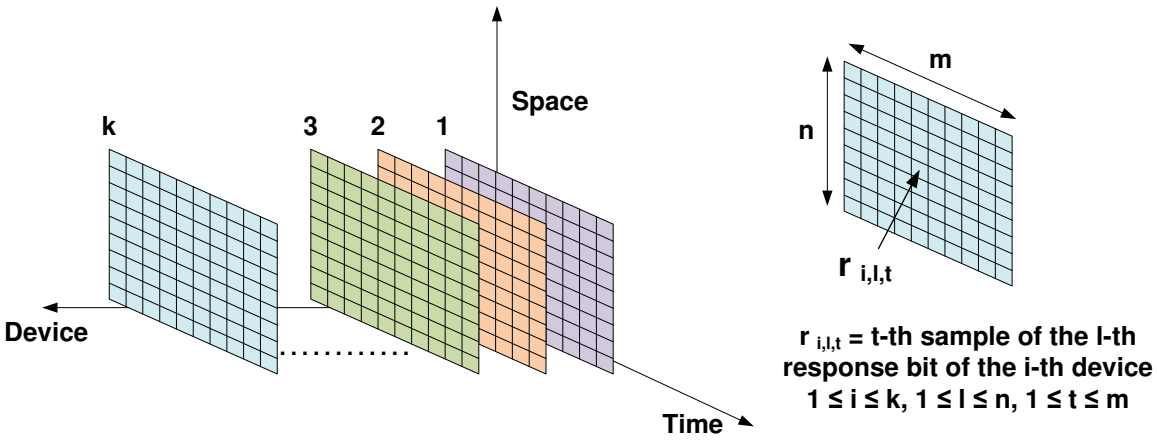

Figure 6.2: Dimensions of PUF measurement

the SRAM cell that produces a response i has an on-chip location that is different from that of another SRAM cell that produces a response $j (i \neq j)$. In the Arbiter PUF, the location of the Arbiter that produces the response is fixed. However, the locations of the stimulated delay paths change depending on the challenge (whether straight connections or criss-cross connections through a switch). To estimate the randomness of a PUF, we examine multiple response bits from a PUF. Hence, the space dimension is useful.

Finally, the time-dependent properties of a PUF are captured along the time axis. A critical attribute of a PUF is the reliability of the responses. It estimates how consistently the responses can be generated against varying operating conditions such as variable ambient temperature and fluctuating supply voltage. To estimate the reliability, we take multiple samples of the responses at different instances of time under different operating conditions. Samples of PUF responses are also useful in estimating the circuit aging effect on PUFs.

6.2.2 Defining PUF Parameters

Now, we will introduce four PUF parameters based on the measurement dimensions introduced. These parameters are: uniqueness, reliability, uniformity, and bit-aliasing.

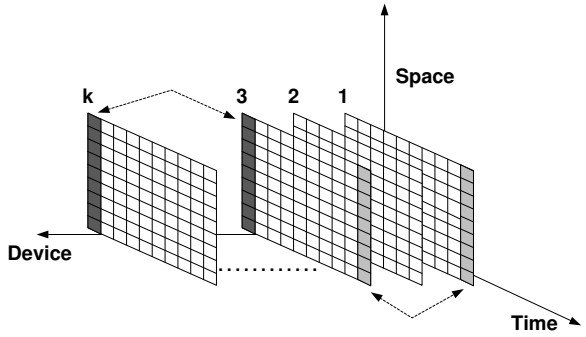

Figure 6.3: PUF uniqueness evaluation

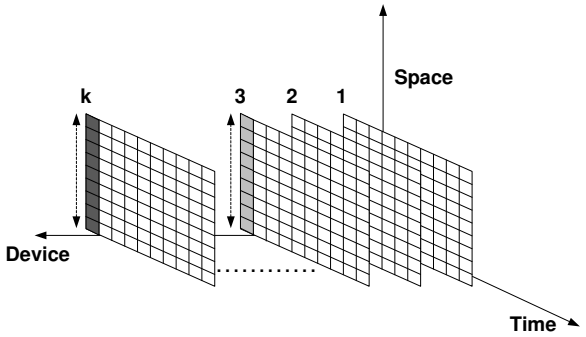

Figure 6.4: PUF uniformity evaluation

1) Uniqueness:

According to Equation (2.1), the uniqueness of a PUF is defined as:

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

It is the average inter-chip Hamming distance (HD) computed across a group of chips. In Figure 6.3, it is shown that the uniqueness is estimated along the device axis. One comparison is shown in dark gray between the device 3 and the device k. Another comparison is shown in light gray between the device 1 and the device 3.

2) Uniformity

We define the uniformity of an n-bit PUF identifier as the percentage Hamming Weight(HW) of the n-bit identifier (Equation (2.2)):

$$(Uniformity)_i \ = \frac{1}{n} \sum_{l=1}^{n} r_{i,l} \times 100\%$$

where $r_{i,l}$ is the $l-$th binary bit of an $n-$bit response from a chip i.

In Figure 6.4, the uniformity of the device k and the device 3 are evaluated along the space axis (marked in dark gray and light gray respectively).

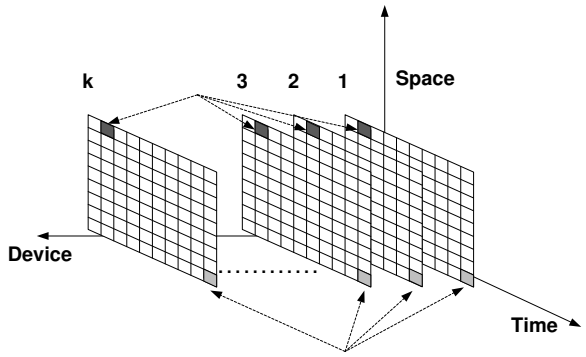

Figure 6.5: PUF bit aliasing evaluation

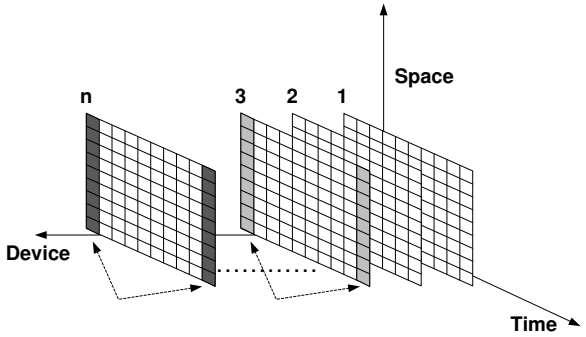

Figure 6.6: PUF reliability evaluation

3) Bit-aliasing

We estimate the bit-aliasing of the l-th bit in the PUF identifier as the percentage Hamming Weight(HW) of the l-th bit of the identifier across k devices (Equation (2.3)):

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

where $r_{i,l}$ is the $l-$th binary bit of an $n-$bit response from a chip i.

In Figure 6.5, the bit-aliasing is evaluated along the device axis for two different bit locations (marked in dark gray and light gray).

4) Reliability:

In equation (2.5), we defined the PUF reliability as:

$$Reliability = 100\% - HD_{\text{intra}}$$

HD_{intra} is estimated as follows.

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

Figure 6.6 shows how the reliability of a PUF is evaluated using the time dimension of PUF measurement. The two intra-chip Hamming distance measurements are shown along the time axis for the device 3 and the device k with light gray and dark gray respectively.

Table 6.1: Different PUF parameters

Hori et al. [18]	Randomness Steadiness Correctness Diffuseness Uniqueness
Maiti et al. [32]	Uniformity Bit-aliasing Uniqueness Reliability
Su et al. [43]	Probability of Misidentification
Majzoobi et al. [35]	Single-bit Probability Conditional Probability
Yamamoto et al. [54]	Variety

6.2.3 Analysis of PUF Parameters Defined by Other Researchers

In this section, we explore different PUF parameters defined by other researchers in the research community. We also try to find out if there is any redundancy among these parameters. Table 6.1 shows few examples of different PUF parameters proposed by several research groups. In this research, we will be presenting an analysis of the parameters proposed by Hori et al. in [18] with a comparison of the parameters introduced by us in Section 6.2.1. The reason we chose this work for analysis is that it has a similar effort to define PUF parameters like ours. Moreover, the authors of this work made a large PUF dataset based on the Arbiter PUF available for analysis. This motivated us to use the Arbiter PUF dataset in carrying out a detailed comparison analysis with a similarly large dataset based on the RO PUF that we generated. We also analyze the parameter "probability of misiden-

tification" proposed by Su et al [43]. This parameter estimates the rate of false positives in chip identification for a given number of noisy bits in the PUF responses.

There are several other parameters existing in the literature. For example, Majzoobi et al. defined parameters such as single-bit probability and conditional probability [35]. A parameter called variety has been proposed by Yamamoto et al [54] [1]. As part of the future effort, we plan to analyze many of these parameters to enhance the evaluation-comparison method.

At first, we introduce few notations that will be used to describe the parameters. The notations are:

$$
\begin{aligned}
N &= \text{ total number of chips} \\
n &= \text{ index of a chip } (1 \leq n \leq N) \\
K &= \text{ total number of identifiers(IDs) generated per chip} \\
k &= \text{ index of an ID in a chip } (1 \leq k \leq K) \\
T &= \text{ total number of samples measured per ID} \\
t &= \text{ index of a sample } (1 \leq t \leq T) \\
L &= \text{ total number of response bit in an ID} \\
l &= \text{ index of a response bit } (1 \leq l \leq L) \\
M &= \text{ total number of ring oscillators} \\
m &= \text{ index of an oscillator } (1 \leq m \leq M)
\end{aligned}
$$

The above notations, apart from m and M for the ring oscillators, have been proposed by Hori et al. [18]. We decide to keep these notations to define any PUF parameters except the fact that we use r in place of b (used in [18]) to denote a single response bit from a PUF. We notice that the parameters in Section 6.2.1 used k as the total number of devices, n as the

[1] This term was introduced in the slides for the presentation of the paper [54] by the authors in CHES,2011. [53]

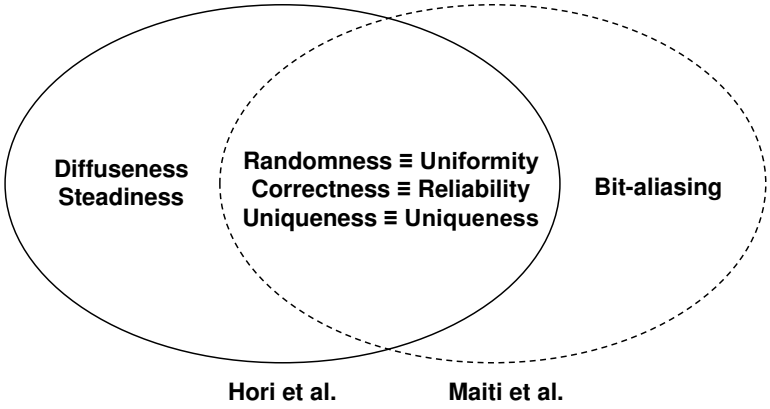

Figure 6.7: Relation between parameters defined by Hori et al. and this work

total number of response bits from a PUF, and m as the number of samples of the response bits. While we compare the parameters defined by the two groups, we will express all the PUF parameters using the notations above.

We have proposed four parameters as described in Section 6.2.1. They are uniqueness, reliability, bit-aliasing, and uniformity. The five parameters proposed by Hori et al. are uniqueness, randomness, correctness, steadiness, and diffuseness. Upon analyzing these parameters, we have found that there are similarities among these parameters. Figure 6.7 shows the relation between these parameters. There are three parameters from each group that are similar in definition while others are different. We explain all these parameters in detail.

Randomness vs Uniformity:

The randomness by Hori et al. is defined below.

$$- log_2 max(p_n, 1 - p_n) \tag{6.1}$$

$$\text{where } p_n = \frac{1}{K.T.L} \sum_{k=1}^{K} \sum_{t=1}^{T} \sum_{l=1}^{L} r_{n,k,t,l} \tag{6.2}$$

On the other hand, the uniformity parameter by us has been defined as follows.

$$\frac{1}{K.L}\sum_{k=1}^{K}\sum_{l=1}^{L} r_{n,k,l} \tag{6.3}$$

It can be noticed that the randomness includes T samples of the response bits while the uniformity does not include the samples and is based on the reference/correct bits only. Also, the randomness is expressed in the min-entropy form. Otherwise, the two definitions are very similar in nature. Both of them estimate the ratio of '1' vs '0' in all the response bits generated by a PUF.

Correctness vs Reliability:

The correctness parameter is defined as follows.

$$1 - \frac{2}{K.T.L}\sum_{k=1}^{K}\sum_{t=1}^{T}\sum_{l=1}^{L}(r_{n,k,l} \oplus r_{n,k,t,l}) \tag{6.4}$$

On the other hand, the reliability is defined as below.

$$1 - \frac{1}{K.T.L}\sum_{k=1}^{K}\sum_{t=1}^{T}\sum_{l=1}^{L}(r_{n,k,l} \oplus r_{n,k,t,l}) \tag{6.5}$$

The reliability parameter is different from the correctness only in terms of the factor by which it is normalized. The reliability is calculated based on the average value of the intra-chip HD, whereas the correctness is normalized by the maximum value of the fractional Hamming distance between the correct ID (the ID which is considered as the reference) and the sample IDs. There are few points that are not clearly mentioned in the definition of the correctness parameter.

The correctness is defined using the sum of Hamming Distances (SHD) normalized by T, K, and L. A parameter $c_{n,k,l}$ has been defined as the SHD between the correct bit $r_{n,k,l}$ and the generated bit $r_{n,k,t,l}$ through T tests.

$$c_{n,k,l} = \sum_{t=1}^{T} r_{n,k,l} \oplus r_{n,k,t,l} \tag{6.6}$$

However, it is not clear from the definition what the time instances of the measurements are. Since it is used to capture the effect of device defect or aging, it can be assumed that the correct bit $r_{n,k,l}$ is measured before the aging or defect whereas $r_{n,k,t,l}$ is measured after the aging effect. If that is the case then the following inequality (mentioned in [18])does not necessarily hold good.

$$0 \leq c_{n,k,l} = \sum_{t=1}^{T} r_{n,k,l} \oplus r_{n,k,t,l} \leq \frac{T}{2} \tag{6.7}$$

This is because there might be a case when the aging or device defect might flip a correct bit in such a way that the subsequent T samples produce a complementary value for more than $T/2$ samples. This inequality holds good *always* only if both $r_{n,k,l}$ and $r_{n,k,t,l}$ are measured during the same sampling instance which contradicts the definition of correctness.

Uniqueness by Hori et al. vs Uniqueness by Maiti et al.:

The uniqueness is defined by Hori et al. as:

$$\frac{1}{K.L}\frac{4}{N^2}\sum_{k=1}^{K}\sum_{l=1}^{L}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(r_{i,k,l} \oplus r_{j,k,l}) \tag{6.8}$$

The uniqueness is defined by us as:

$$\frac{1}{K.L}\frac{2}{N(N-1)}\sum_{k=1}^{K}\sum_{l=1}^{L}\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(r_{i,k,l} \oplus r_{j,k,l}) \tag{6.9}$$

In the case of the uniqueness, two definitions differ from each other with respect to the normalization factor. Hori et al. used the sum of Hamming distance (SHD) of all the possible combinations of the PUF identifiers as the normalization factor. For a group of N chips, the value of that factor is $K.L.N^2/4$. On the other hand, we used the total number of all possible pairwise combinations of response bits as the normalizing factor the value of which is $K.L.N.(N-1)/2$. For a large value of N, the normalization factor used by us. is approximately two times bigger than that used by Hori et al.

Parameters uniquely defined by both the groups:

The term bit-aliasing is uniquely defined by us. On the other hand, the steadiness and the diffuseness are uniquely defined by Hori et al. The diffuseness is same as the uniqueness except the fact that the diffuseness is defined inside a single chip among several different IDs while the uniqueness is measured across several chips.

Steadiness: The steadiness measures the degree of bias of a response bit towards '0' or '1' over T samples. It is defined as:

$$S_n = 1 + \frac{1}{K.L}\sum_{k=1}^{K}\sum_{l=1}^{L} log_2 max(p_{n,k,l}, 1 - p_{n,k,l}) \tag{6.10}$$

$$\text{where } p_{n,k,l} = \frac{1}{T}\sum_{t=1}^{T} r_{n,k,t,l} \tag{6.11}$$

This parameter is somewhat similar to the correctness parameter. A lower value of steadiness will produce a lower correctness. In this case also, the time stamps of the sample measurements are important. This is because the steadiness of a PUF may change when operating conditions change. However, Hori et al. did not discuss the time factor while defining the steadiness parameter [18].

Diffuseness: The diffuseness is defined as:

$$\frac{1}{L}\frac{4}{K^2}\sum_{l=1}^{L}\sum_{i=1}^{K-1}\sum_{j=i+1}^{K}(r_{n,i,l} \oplus r_{n,j,l}) \tag{6.12}$$

One question arises regarding the diffuseness parameter. Since the diffuseness is estimated among K signatures (L bits each) generated in a chip, its value might change depending on how we create a group of bits to produce an ID. For example, there are a total of $K \times L$ binary bits in K L-bit signatures. These $K \times L$ bits can be divided in many possible K groups with L bits each. Therefore, the same set of $K \times L$ bits can lead to different values of diffuseness based on the combination we select. Since, the PUF challenges are selected by a software program [18], the diffuseness can be controlled deterministically.

Bit-aliasing: The bit-aliasing parameter is defined as:

$$(\text{Bit} - \text{aliasing})_{k,l} = \frac{1}{N}\sum_{n=1}^{N} r_{n,k,l} \tag{6.13}$$

Probability of misidentification (PMSID):

We introduce another parameter probability of misidentification defined by Su et al [43]. It is a useful parameter to estimate the probability of a chip being falsely identified as another chip due to noise in the response bits.

Suppose a chip X has a reference PUF identifier R_X with L bits. At some other point in time, it produces an identifier R'_X. Therefore, the number of unreliable bits in that PUF is $HD(R_X, R'_X)$. Now, if there exists another chip Y such that $HD(R_Y, R'_X) \leq HD(R_X, R'_X)$, there will be a misidentification. For a PUF identifier with L response bits, if p is the fraction of the unreliable bits (fractional intra-chip HD), and h is the value of HD between the L-bit identifier of the chip and that of another chip ($h \leq L$), the probability of misidentification is defined as [43]:

$$\sum_{h=0}^{L}\left[\binom{L}{h}0.5^h(1-0.5)^{L-h}.\sum_{h/2}^{h}\binom{h}{h/2}p^{h/2}(1-p)^{h-h/2}\right] \tag{6.14}$$

6.2.4 Final Set of PUF Parameters

We have discussed several parameters that quantify the performance of a PUF. As a conclusion, we suggest a set of parameters with some modifications as the components of the PUF evaluation-comparison method. It includes seven parameters: uniformity, reliability, steadiness, uniqueness, diffuseness, bit-aliasing, and probability of misidentification. Basically, we excluded the redundant parameters while analyzing the parameters proposed by Hori et al., Su et al., and us. This set of parameters will serve as the starting point of the evaluation-comparison method. However, we believe that this set is subject to further modification and enhancement. Figure 6.8 shows these seven parameters mapped along the proposed

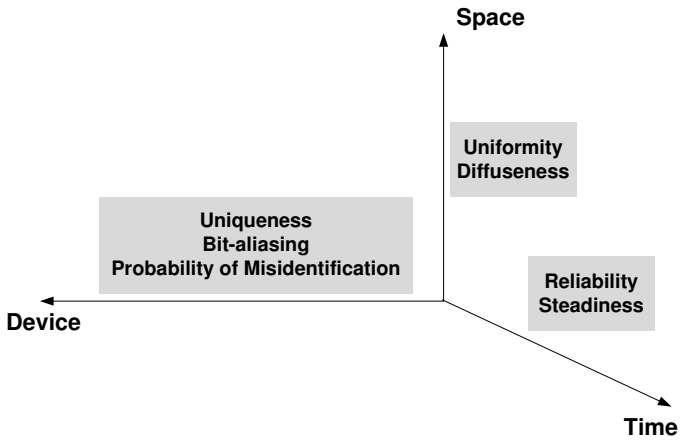

Figure 6.8: Final parameters mapped on the PUF measurement dimension

PUF measurement dimensions. We briefly discuss why we inlcuded these parameters as the components of the evaluation-comparison method.

1) Uniformity: As the analysis shows, uniformity does not include samples of response bits unlike randomness does. If samples are included, it becomes dependent on time. Since we want to evaluate the ratio of '1's and '0's on an average i.e. based on the reference response bits, uniformity is a good fit. Moreover, to estimate the time dependency of a PUF response, we have other parameters such as reliability and steadiness.

2) Reliability: Our analysis showed that both the correctness and the reliability are defined in a similar way. However, the time reference is not defined well in case of the correctness. Moreover, the inequality (Eqn(6.7)) that determines the normalization factor for the correctness has been shown not to be valid always. Hence, we propose to include the reliability parameter defined in Equation (6.5).

3) Steadiness: Even though the steadiness parameters seems to be closely related to the reliability/correctness parameter, it represents the bias of individual response bits on an average. Therefore, we suggest to include this parameter as well. However, this parameter needs to be defined based on time stamps i.e. a chip may have different steadiness values depending on the time when it is measured.

4) Uniqueness: For the uniqueness parameter, both Hori et al. and us employ average inter-chip HD of the PUF identifier except the normalization factor. The normalization factor used by Hori et al. represents the upper bound of the SHDs among all possible IDs which is an useful information about a PUF. Hence, we suggest to include the uniqueness defined by Hori et al.

5) Diffuseness: The diffuseness, as discussed earlier, is very similar to the uniqueness. This parameter becomes useful when a PUF has a large CRP space like the Arbiter PUF, and several identifiers can be produced from a single chip. Therefore, we suggest to use the diffuseness parameter when the PUF being evaluated has a large CRP space.

6) Bit-aliasing: The bit-aliasing parameter is very useful in estimating the bias of a particular response bit across several chips. It may also give us information about any systematic, spatial effect across several devices.

7) Probability of misidentification: Finally, we also propose to include the probability of misidentification to estimate the rate of error in identification by a PUF. This parameter shows how chip identification may be affected by noise in the response bits.

6.3 Comparison of the RO PUF with the Arbiter PUF: A Test Case

In this section, we compare the RO PUF with the Arbiter PUF using the parameters defined by Hori et al. and us as well as the probability of misidentification. We used two datasets for this purpose: a) one dataset consists of the RO PUF responses from 193 FPGAs measured by us. b) the other dataset consists of the Arbiter PUF responses from 45 FPGAs measured by Hori et al. We have already introduced the RO PUF in Section 3.3. We briefly describe the Arbiter PUF.

An Arbiter PUF, proposed by Lim et al., exploits the delay mismatch between a pair of

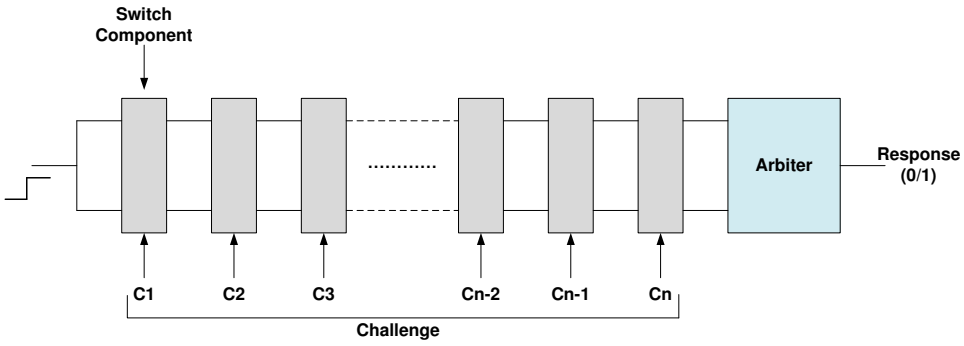

Figure 6.9: Arbiter PUF

identically routed paths to generate a response bit [24]. Depending on which of the delay path is faster, the arbiter flip-flop produces a '0' or '1' as a PUF response. Due to random variation in delay paths, this response bit is random. Several pairs of delay paths can be configured by setting the inputs of the switch components (shown in Figure 6.9) used as challenge inputs.

Table 6.2 describes both the datasets that have been used for the analysis. It also includes the device technology of the FPGAs used. We measured the RO PUF at normal operating conditions. Since Hori et al. did not mention any variation in the operating condition during their measurement, we assume the Arbiter PUF was also measured under normal operating condition.

We first evaluate the parameters defined by Hori et al. using the RO PUF dataset and compare them with the results from the Arbiter PUF reported in [18]. Since the RO PUF we implemented produces only *one* 511-bit identifier (K=1), the diffuseness is not calculated for the RO PUF dataset. After that, we evaluate the parameters defined by us using the Arbiter PUF dataset and compare them with the results based on the RO PUF dataset. We also compare both the datasets using the probability of misidentification. Finally, we summarize the comparison based on the set of seven parameters we selected.

Table 6.2: Detail of the datasets used

	Arbiter PUF	**RO PUF**
N	45	193
T	1024	100
K	1024	1
L	128	511
M	-	512
Device technology	65nm (Virtex 5)	90nm (Spartan 3E)

Table 6.3: Comparison of the Arbiter PUF and the RO PUF using the parameters defined by Hori et al.

	Arbiter PUF	**RO PUF**	**Ideal Value**
Average Randomness	84.69%	96.81%	100%
Average Probability	55.61%	49.82%	50%
Average Steadiness	98.48%	98.51%	100%
Average Correctness	98.28%	98.29%	100%
Average Uniqueness	36.75%	94.07%	100%

6.3.1 Comparison using Parameters defined by Hori et al.:

Table 6.3 shows the parameters defined by Hori et al. evaluated on both the datasets. The parameter values based on the Arbiter PUF dataset have been taken from [18]. It can be noticed that the RO PUF shows better randomness compared to the Arbiter PUF. This is supported by the fact that the average bit probability of the RO PUF is close to 0.5 i.e. the RO PUF responses are more equally likely between '0' and '1'. Normally it is expected that a Virtex 5 FPGA on a 65-nm technology will have more variability (hence

Table 6.4: Confidence Interval comparison results with 95% confidence level

	Arbiter PUF		**RO PUF**	
	Confidence Interval	Width	Confidence Interval	Width
Randomness	[0.8388, 0.8546]	0.01586	[0.9892, 0.9990]	0.00986
Bit Probability	[0.5530, 0.5591]	0.00611	[0.4962, 0.5003]	0.00407
Steadiness	[0.9626, 1.0000]	0.04134	[0.9846, 0.9857]	0.00110
Correctness	[0.9579, 1.0000]	0.04206	[0.9822, 0.9834]	0.00121
Uniqueness	[0.2127, 0.5222]	0.30950	[0.9334, 0.9481]	0.02940

more randomness in PUF responses) than a Spartan 3E FPGA on a 90-nm technology. However, this result shows that the RO PUF has better randomness than the Arbiter PUF. This indicates that individual PUF technique may have significant influence on extracting the variability information. Another significant difference can be observed in the value of the uniqueness. The uniqueness of the RO PUF is distinctly much higher than that of the Arbiter PUF. A lower value of the randomness in the Arbiter PUF is one of the reasons why the uniqueness is lower in it. In the next section, we will evaluate the bit-aliasing parameter that may explain this contrast in the uniqueness more. Apart from that, both the PUFs show similar values of the steadiness and the correctness. It implies that the two PUFs are in general highly tolerant to noise at normal operating condition.

Table 6.4 shows the confidence interval(CI) proposed by Hori et al. for a confidence level of 95% for both the datasets. It can be noticed that the RO PUF dataset shows significantly narrower CI compared to the Arbiter PUF dataset. This indicates that the size of the PUF dataset (the RO PUF having a larger dataset compared to the Arbiter PUF) has substantial impact on determining the confidence interval of the parameters.

Table 6.5: Comparison of the Arbiter PUF and the RO PUF using the parameters defined by Maiti et al.

	Arbiter PUF	**RO PUF**	**Ideal Value**
Uniformity	55.69%	50.56%	50%
Bit-aliasing	19.57%	50.56%	50%
Uniqueness	7.20%	47.24%	50%
Reliability	99.76%	99.14%	100%

6.3.2 Comparison using Parameters defined by Maiti et al.:

Table 6.5 shows the average values of the parameters defined by us for both the datasets. We considered 511 response bits for the RO PUF whereas $1024 \times 128 = 131072$ response bits were considered for the Arbiter PUF.

The uniformity result is consistent with the average probability reported in Table 6.3. For the bit-aliasing, the average in the RO PUF is close to the ideal value of 50% while the average in the Arbiter PUF deviates significantly from 50%. Moreover, we found that the minimum value of bit-aliasing is 0% in case of the Arbiter PUF. This shows that there are bit positions that produce a value of '0' for all the 45 chips. In fact, we found 21314 out of 131072 bit positions (nearly 16%) produced a value of 0%. These bits do not contain any useful information. This is consistent with a very low value of the uniqueness in the Arbiter PUF reported in Table 6.3. One reason for this may be the difficulty in ensuring routing symmetry in an Arbiter PUF implemented on an FPGA [36]. The sharp difference in the value of the uniqueness is visible in this case also. The reliability values are comparable for both the datasets indicating both the RO PUF and the Arbiter PUF are equally reliable.

Table 6.6: Comparison of probability of misidentification

	Minimum	Maximum	Average
RO PUF	2.81×10^{-71}	4.42×10^{-39}	1.18×10^{-40}
Arbiter PUF	3.03×10^{-13}	3.91×10^{-12}	1.50×10^{-12}

6.3.3 Comparison using the Probability of Misidentification:

Table 6.6 shows the value of probability of misidentification for the two datasets. The RO PUF dataset shows a much lower value compared to the Arbiter PUF dataset. However, this is due to the fact that the length of the identifier for the RO PUF is 511 whereas it is 128 for the Arbiter PUF. In any case, even the Arbiter PUF shows a very low probability of misidentification. This is consistent with the fact that both the PUF showed a very high value of reliability indicating that the proportion of the noisy bits in both the PUFs are low.

6.3.4 Summary of Comparison between the RO PUF and the Arbiter PUF

Table 6.7 shows the summary of the comparison between the Arbiter PUF and the RO PUF in terms of the seven parameters we selected as part of the proposed evaluation-comparison method. The two PUFs exhibit comparable performance in terms of the uniformity, the reliability, and the steadiness. However, the RO PUF shows much better performance compared to the Arbiter PUF in terms of the uniqueness, the bit-aliasing, and the PMSID. The diffuseness parameter could not be evaluated for the RO PUF.

Table 6.7: Summary of comparison between the Arbiter PUF and the RO PUF

	Arbiter PUF	**RO PUF**	**Ideal Value**
Uniformity	55.69%	50.56%	50%
Reliability	99.76%	99.14%	100%
Steadiness	98.48%	98.51%	100%
Uniqueness	36.75%	94.07%	100%
Diffuseness	98.39%	-	100%
Bit-aliasing	19.57%	50.56%	50%
PMSID	1.50×10^{-12}	1.18×10^{-40}	0%

6.4 Summary

In this work, we aimed at defining a method to evaluate as well as compare the performance of several PUFs irrespective of the underlying PUF techniques. Our approach relies on the statistical properties of the binary response bits of a PUF. We first proposed three dimensions of PUF measurement. Based on our analysis on parameters defined by us as well as by others, we proposed a set of seven PUF parameters as the primary building block of the evaluation-comparison method. Subsequently, we compared two different PUFs, namely the RO PUF and the Arbiter PUF using these parameters based on measured PUF data. The RO PUF shows better performance than the Arbiter PUF in terms of the uniqueness, the bit-aliasing, and the probability of misidentification while other parameters yielded comparable results from both the PUFs.

As part of the future work, we plan to improve this method by studying other parameters that have not been covered in this work. We also plan to include implementation-related aspects of PUFs such as area, performance, and power as part of the evaluation criteria.

Chapter 7

Microprocessor-intrinsic PUF

In this chapter, we introduce a novel PUF that extracts the variability existing in a microprocessor pipeline to uniquely identify the microprocessor chip. This PUF accepts a microprocessor instruction as a challenge and produces the delay in a data path or a control path in the microprocessor as the response. The delay value is captured by operating the microprocessor at an elevated clock frequency also known as over-clocking. The entire PUF mechanism can be controlled by the microprocessor itself using software programs. Moreover, the proposed PUF does not require any dedicated hardware or any type of modification in the existing hardware except a circuit that can vary the clock frequency of the microprocessor. We assume the availability of a digital clock manager (DCM) or a phase locked loop (PLL) that can be used for clock generation/variation in general, not just for the PUF. For example, on an FPGA platform, a DCM or a PLL is available as a standard component and hence, can be used for this PUF with no extra cost.

Why do we need a new PUF? This is an important question, given the number of PUF proposals already available. Most of the proposed PUFs require a significant amount of hardware resources. For example, a ring oscillator-based PUF requires a pair of ring oscillators to generate one bit of PUF response [45]. A butterfly PUF consumes two latches to produce the same [22]. However, there are PUFs that can be implemented without the need

of dedicated hardware resource; they are called *intrinsic* PUFs. Guajardo et al. exploited random power-up values of SRAM cells to build the SRAM PUF [14]. Similar work has been done by Holcomb et al. [17]. The main drawback of this type of PUFs is that they require power cycling. Hence, the random information is not easily accessible. Maes et al. showed that the power-up states of the D flip-flops (DFFs) inside an FPGA can be captured by modifying the programming bitfile [29]. Later on, Leest et al. showed a similar DFF-based PUF on an ASIC platform [48]. In the DFF-based PUFs also, in order to extract the PUF CRPs, one has to power cycle the chip which is not the best practical solution.

Our proposed PUF is also intrinsic in nature. The key difference between the existing intrinsic PUFs and the proposed one is that our PUF does not depend on the power-up state of the circuit. As a result, the proposed PUF can evaluate the CRPs anytime during the operation of a circuit. We identify several benefits out of the proposed PUF. First, a microprocessor is a ubiquitous circuit element, present in almost every embedded application. This makes it an ideal candidate for an intrinsic PUF. Second, we use microprocessor instructions to build the PUF challenge-response mechanism. This provides an easy way of integrating low-level hardware information with the high-level software applications. Finally, any post-processing of the PUF data can be flexibly done in software obviating any need of error-correction hardware. The main contributions of this research are :

- We first propose how variability in the data path and the control path of a microprocessor pipeline can be captured using microprocessor instructions. We introduce a challenge-response mechanism based on the variability information.
- We demonstrate the idea on a SPARC instruction set implemented in a 32-bit LEON3 processor in a Spartan 3E FPGA. We analyzed different types of instructions such as arithmetic instructions, logical instructions, and control instructions. Our results show good amount of uniqueness among the chips we tested in terms of the PUF responses. It produced 37 secure response bits for authentication based on a subset of five SPARC instructions. Furthermore, they also exhibit a high value of reliability.

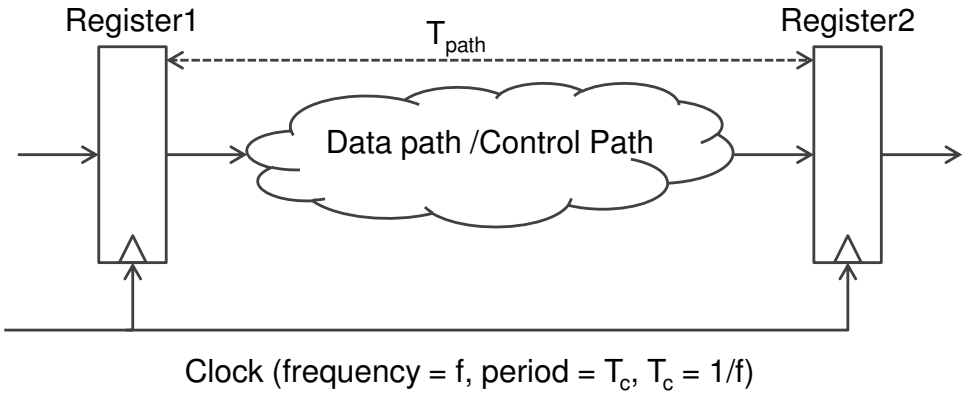

Figure 7.1: A pipelined delay path

7.1 Detail of the microprocessor-intrinsic PUF

In this PUF, we aim to exploit the delay variability that exists in several data paths and control paths in a microprocessor pipeline. Depending on the instruction that is run on the microprocessor, a particular set of data paths or control paths are activated. Due to manufacturing process variation, these paths have variable delay from one chip to another. We extract this variable delay to authenticate a chip. Figure 7.1 shows a typical pipelined delay path. In a microprocessor pipeline, similar paths exist. The clock period T_{c} is constrained by the following relationship:

$$T_{\mathrm{c}} \geq t_{\mathrm{c\_q}} + t_{\mathrm{comb}} + t_{\mathrm{setup}} \tag{7.1}$$

where $t_{\mathrm{c\_q}}$ is clock to output delay of the register 1, t_{comb} is the combinational path delay, and t_{setup} is the set up time of the register 2. We represent $t_{\mathrm{c\_q}} + t_{\mathrm{comb}} + t_{\mathrm{setup}}$ as T_{path}. The value of T_{path} for a particular path varies from one chip to another due to process variation. In addition, the value of T_{path} also depends on the type of instruction being executed, as well as its operands. If we increase the clock frequency i.e. reduce the clock period, an instruction will fail once the value of the reduced T_{c} reaches the value of T_{path}. In other words, the instruction will reach a failure point once the slack (which is the difference between T_{path} and the original T_{c}) becomes negative. We call the frequency at which an instruction fails due to over-clocking a frequency failure point (FFP). This has been demonstrated in Figure

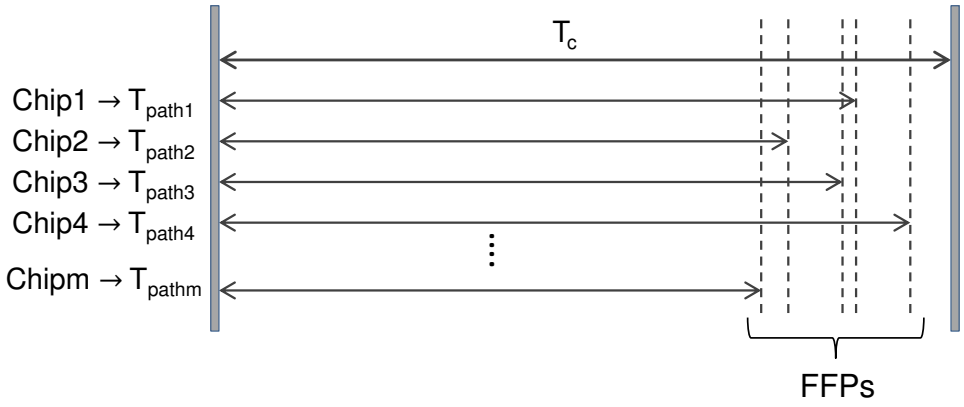

Figure 7.2: Variable FFPs based on variable slacks in data path / control path

7.2; the value of T_{path} for a particular data path/control path varies randomly over a set of m chips. The instruction set of a microprocessor leads to a set of FFPs. The objective of the proposed PUF is to authenticate a microprocessor through the FFPs of its instruction set, or a suitable subset of it.

7.1.1 How do we generate a PUF Response Bit?

We first explain a simple concept that shows how a PUF response bit can be generated using the FFPs corresponding to an instruction. We observed that an instruction does not stop executing abruptly at a particular frequency while the microprocessor is being over-clocked. It goes through a transition phase in which there are several frequencies at which an instruction executes with partial failure. This means when run several times at a particular frequency, it fails a fraction of the total number of times it is run at that frequency. With further increase in the clock frequency, it fails completely beyond a certain frequency. Essentially, the failure of an instruction due to over-clocking occurs through a region and not at a point. Figure 7.3 shows such a scenario[1]. It shows how a particular instruction fails due to over-clocking for three different chips. The failure behavior is estimated by running the instruction $k(k > 0)$ times at several frequencies, and counting the number of times

[1]It is a hypothetical example and not based on real data.

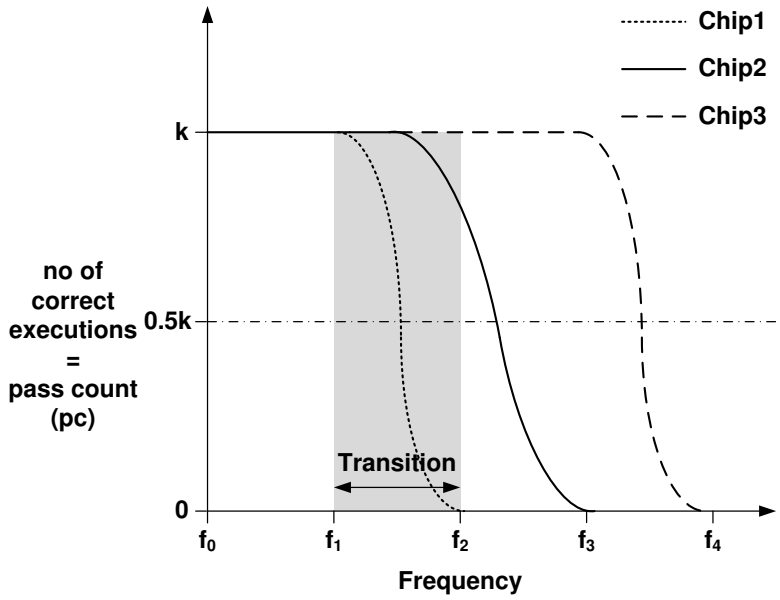

Figure 7.3: Comparison of failure behavior of an instruction across a set of chips

it is correctly executed; we call it the pass count (pc). It is represented along the Y-axis in Figure 7.3. Before the frequency f_1, the pc of chip1 is k. It starts failing partially at frequency f_1. This means that the pc is a fraction of k at f_1. Finally, the pc becomes zero at f_2. The region between f_1 and f_2 (colored in gray) is the transition region for chip1 for the instruction executed. Due to process variation, chip2 and chip3 have their transition regions for the same instruction at different frequencies as illustrated in Figure 7.3. The transition region for chip2 and chip3 are not highlighted for simplicity. Now, to generate a response bit r at an arbitrary frequency f for a particular instruction, let us apply a simple quantization rule as follows:

$$r = \begin{cases} 1 & \text{if } pc/k > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{7.2}$$

Following the above rule, three chips in Figure 7.3 produce different response bits at different frequencies. For example, at frequency f_2, chip1, chip2, and chip3 produce '0','1', and '1' respectively. At f_3, they produce '0', '0', and '1' respectively. We can create a string of response bits this way by sampling an instruction at different frequencies. This is the basic

working principle of the proposed PUF.

7.1.2 Formalizing the PUF Mechanism

We follow the standard PUF mechanism consisting of two phases: enrollment and evaluation. In the enrollment phase, we characterize several different instructions at different levels of over-clocking to measure the transition regions. We slightly modify the basic quantization rule proposed in the previous section. We divide the transition region in four parts based on the *pc* value and assign a binary value (used as PUF response, r) to those sub-regions using the follow rule:

$$r = \begin{cases} 00 & \text{if } \ pc/k \leq 0.1 \\ 01 & \text{if } 0.1 < pc/k \leq 0.5 \\ 10 & \text{if } 0.5 < pc/k \leq 0.9 \\ 11 & \text{if } pc/k > 0.9 \end{cases} \tag{7.3}$$

The idea is depicted in Figure 7.4. Suppose a chip produces the following *pc* values for an instruction at five consecutive sampling frequencies in the transition region as 95, 87, 46, 17, 5. Assuming k=100, the corresponding responses would be 11, 10, 01, 01, 00 according to Equation (7.3). The resultant response is the concatenation of these bit-strings i.e. 1110010100.

The region between $0.5k$ and $0.9k$ is more likely to execute an instruction correctly, but it is distinct from the region where the processor has 100% *pc*. On the other hand, the region between $0.1k$ and $0.5k$ is more likely to execute an instruction incorrectly, but it is distinct from the region where the *pc* is 0. This is the reason why we divided the transition region in four parts. The reason behind choosing the threshold of $0.1k$ and $0.9k$ is to reduce errors in the response bits. Due to noise in the circuit, the *pc* around the frequency f_{start} (refer to Figure 7.4) in the transition region may fluctuate between 100% and values slightly lower than 100%. Similarly, near the frequency f_{end} (refer to Figure 7.4), it may fluctuate

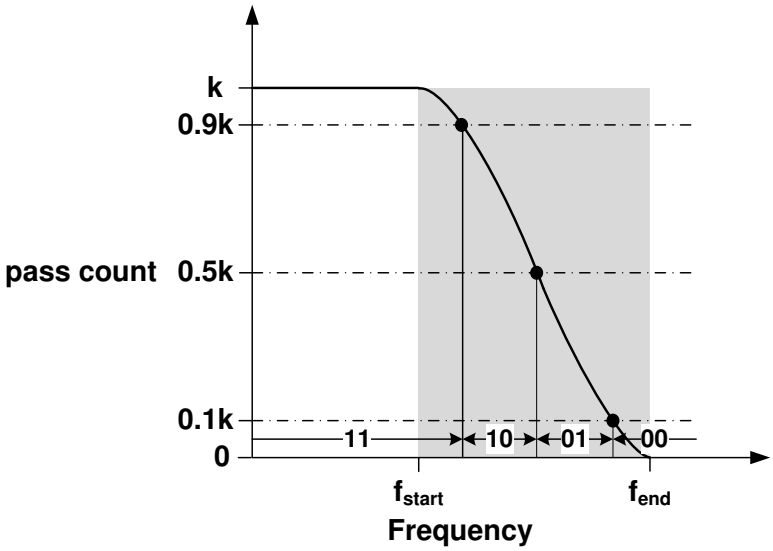

Figure 7.4: Response generation in the proposed PUF

between 0 and values slightly higher than 0. During the enrollment, we collect three types of information: a) the frequency at which an instruction starts failing partially. (the transition start point f_{start}) b) the frequency at which it completely stops executing. (the transition end point f_{end}) c) the sampling frequencies in the transition region and the *pc* values at the sampling frequencies including f_{start} and f_{end}. These three types of information are captured and stored in a secure database as failure transition information (FTI). Each of these three types of information are utilized for authentication as they vary from one chip to another.

During the evaluation phase, a verifier can check the chip's claimed identity as follows. First, the verifier defines a challenge by selecting one or more instructions and a set of frequencies at which the instructions will be run. The claiming chip then executes the selected instructions at those frequencies and returns the binary responses for each of the selected instructions following Equation (7.3). Finally, the verifier calculates the binary responses using the FTI stored in his secure database and compares them with those returned by the chip. A match implies an authentic chip. To clone a chip, an adversary needs to know the precise detail of the FTI of a chip for all of the characterized instructions.

7.2 Experimental Results

We characterized three types of instructions: arithmetic, logical, and control transfer. We present results for addition, unsigned multiplication, unsigned division, AND operation, and a branch instruction. We implemented our proposed PUF using the SPARC instruction set in a 32-bit LEON3 processor. It is tested on five Xilinx Spartan3E 500 FPGAs. The processor is configured with an internal RAM of 32 Kbytes along with a multiplier and a divider unit. It does not have any external memory, instruction cache or data cache. All the instructions are written in assembly code and loaded to the memory using the JTAG-based debug interface of the LEON3 processor.

It is important to mention that the microprocessor can be restored to its fully functional state from the FFPs of the characterized instructions by small, step-by-step variation of the clock frequency. We used a step of 0.5 MHz. This way the microprocessor can run the PUF fully autonomously. We controlled the processor frequency by two different methods. Initially, we used an external, high-speed pulse generator. In a further refinement of the design, we replaced the external pulse generator by the on-board FPGA frequency generators (DCMs), to demonstrate that the proposed PUF can also run fully autonomously. The step of 0.5 MHz is easily available using the pulse generator. Even a DCM in a Virtex V FPGA can achieve this step as our implementation shows. Moreover, a frequency step as low as 0.038 MHz using a DCM on an FPGA has been reported in [52]. The results presented in this research are generated using the frequency generator.

To save the value of *pc* in memory, we first characterized the FTI of the store instruction and ensured that the clock frequency never exceeds the transition region of the store instruction. This characterization also showed that the FTIs of the arithmetic instructions, logical instruction, and the control instructions are located at a much lower frequency than that of the store instruction. This ensures that the collected data represents the variability of the instruction we want to characterize (such as addition and multiplication) and not the variability of the store instruction.

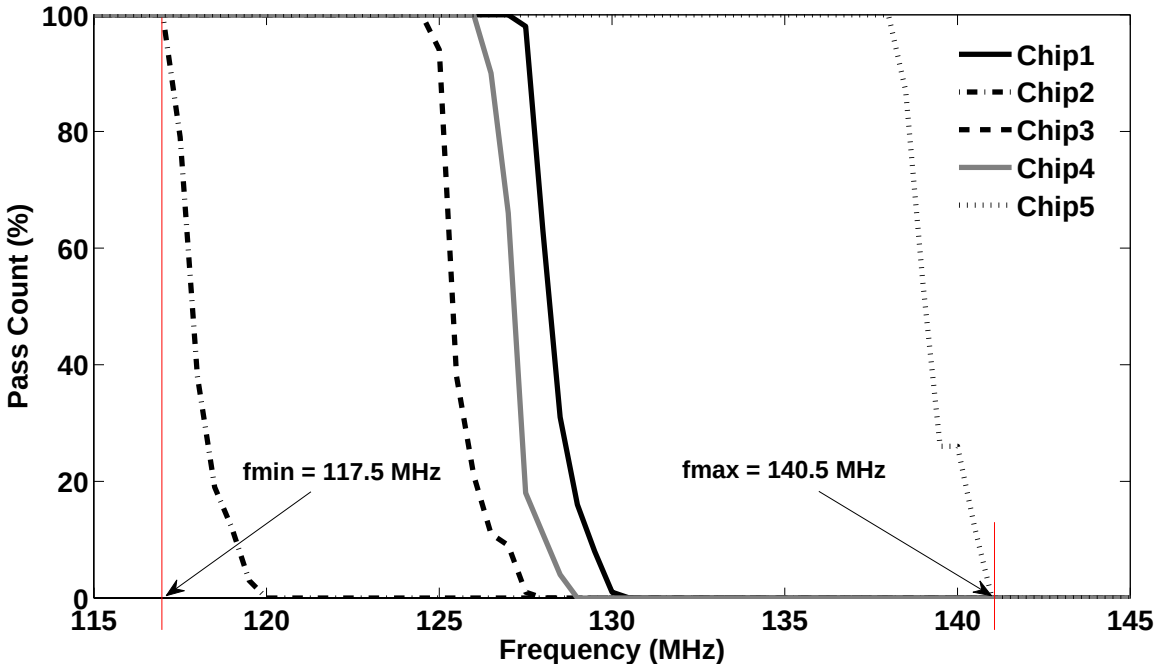

Figure 7.5: Result for ADD 0x7FFFFFFF, 0x1

7.2.1 Performance Evaluation of the PUF

Two parameters are used to evaluate the performance of the PUF: uniqueness and reliability. They have been defined in Equation (2.1) and (2.5) respectively. In order to evaluate these parameters, we need to form an n-bit response string using the PUF. For the proposed PUF, not only the transition regions for an instruction are located at different frequencies for different chips, the length of the transition region also varies from one chip to another. We used 90nm technology FPGAs, and the length of the transition region is 2 MHz on an average. This requires careful calibration of the PUF measurements. We use a simple method to construct the response string in order to evaluate the PUF performance. Let us explain with an example of the addition instruction.

Figure 7.5 shows the result in terms of *pc* values for the operation adding 0x7FFFFFFF and 0x1. The *pc* values are shown as percentage of k with $k = 100$ (refer to Equation (7.3)). Among the five chips, chip2 has the lowest f_{start} (shown as f_{min} in Figure 7.5), and chip5

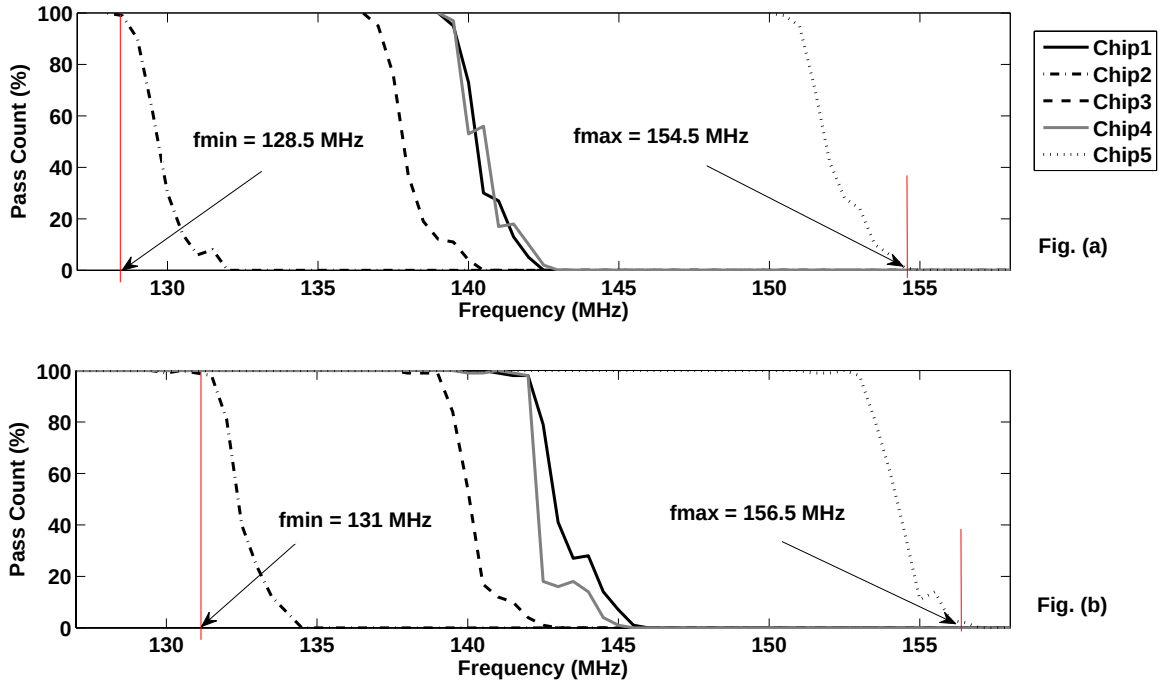

Figure 7.6: (a)Result for MUL 0xFFFFFFFF, 0xFFFFFFFF (b)Result for MUL 0xFFFFFFFF, 0x80000001

has the highest f_{end} (shown as f_{max} in Figure 7.5). Before f_{min}, each of the chips has 100% *pc* value, and hence all of them produce '11' in this region. On the other hand, after f_{max}, all the chips have a *pc* value of 0 leading to '00' response by all. Therefore, there is no information available outside the range from f_{min} to f_{max}. Hence, we derive response bits using Equation (7.3) starting at f_{min} until f_{max} with a step of 0.5 MHz. With f_{min}= 117.5 MHz and f_{max} = 140.5 MHz, there are 47 sampling points in total each producing 2 response bits according to Equation (7.3). Therefore, there are 47 × 2 = 94 responses produced by each of the chips. The average uniqueness among five chips based on a 94-bit response string is 38.7% (refer to Table 7.1). We also tested different input operands such as different length of carry propagation. However, our observation is that no significant variability is found based on carry length variation for the addition instruction.

For the unsigned multiplication operation, we present results for two pairs of operands: (0xFFFFFFFF, 0xFFFFFFFF) and (0xFFFFFFFF, 0x80000001). Figure 7.6 shows that

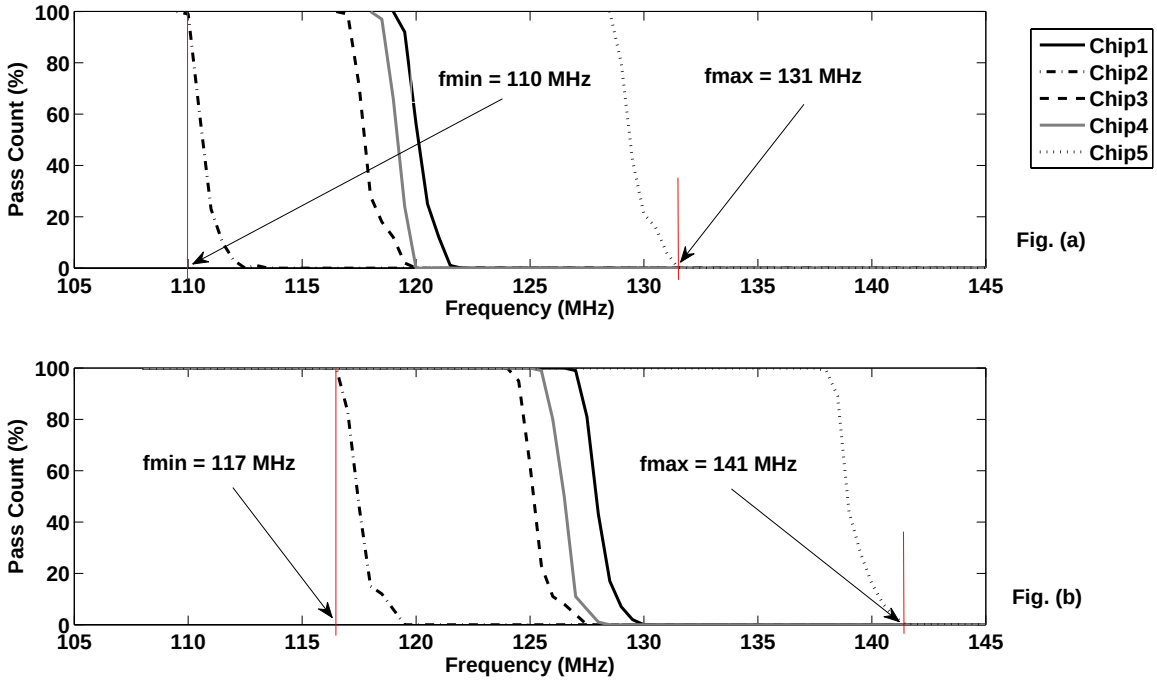

Figure 7.7: (a)Result for DIV 0xFFFFFFFE00000001,0xFFFFFFFF (b)Result for DIV 0xFA0, 0x14

this instruction has an FTI that is dependent on the input operands. For each of the tested chips, the transition region for the two multiplications are located at different frequencies. For example, in Chip2, f_{start}=128.5 MHz for the first pair of operands (shown as f_{min} in Figure 7.6(a)) while f_{start}=131 MHz for the second pair of operands (shown as f_{min} in Figure 7.6(b)). Using the same method as described for the addition instruction, the multiplication instruction produced 106 response bits for 53 sampling frequencies for the first pair of operands. The second pair of operands produced 104 response bits for 52 sampling frequencies. The average uniqueness values are 36% and 36.1% respectively.

Similar to the unsigned multiplication operation, we observed operand dependency in case of the unsigned division instruction also. We present two cases: (0xFFFFFFFE00000001, 0xFFFFFFFF) and (0xFA0, 0x14). Figure 7.7 shows that the two pairs of operands led to FTIs that are located at distinct frequency locations. The uniqueness values are 38.1% and 37.3% respectively for the two cases respectively.

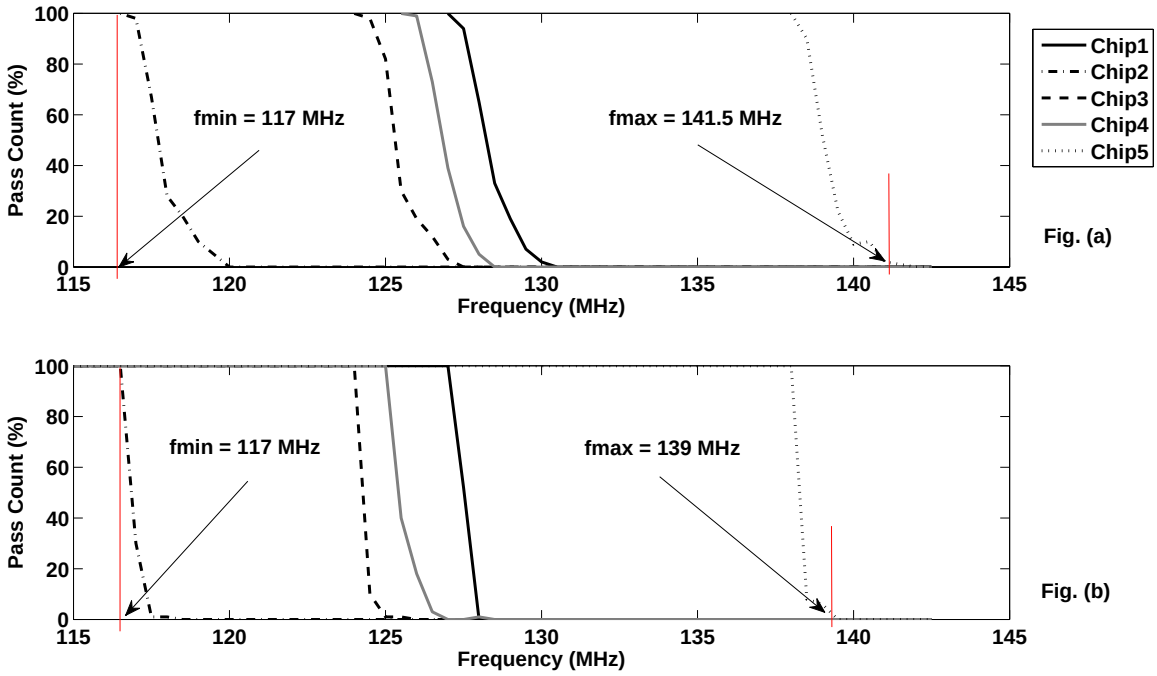

Figure 7.8: (a)Result for AND 0xFFFFFFFF, 0xAAAAAAAA (b)Result for BGE instruction

There are nine different logic instructions supported by the SPARC instruction set. We characterized all of them. The results show that all of the instruction have a very similar FTI. We have presented the result for the AND operation with the operand pair (0xFFFFFFFF, 0xAAAAAAAA) in Figure 7.8(a). The measured uniqueness is 36%.

Finally, we present results for the control transfer instruction. There are sixteen branch instructions supported by the SPARC instruction set. Like the logical operations, they also show similar FTIs. Figure 7.8(b) shows the result for the instruction BGE (branch on greater or equal). It can be noticed that unlike other types of instruction, the branch instruction has a very short transition region for all the chips and reaches a *pc* value of 0 abruptly. The average uniqueness produced by this instruction is 40.6%.

Table 7.1 shows a summary for different instructions used as challenge for the proposed PUF. Though the average uniqueness value shows good result, it deviates from the ideal value of 50% for fully random responses. This is because the proportion of '0's and '1's in

Table 7.1: Summary of PUF performance for different operations

Operation (no of response bits)	**Uniqueness**	**Reliability**
ADD 0x7FFFFFFF, 0x1 (94)	38.7%	97.4%
MULT 0xFFFFFFFF, 0xFFFFFFFF (106)	36%	98.1%
MULT 0xFFFFFFFF, 0x80000001 (104)	36.1%	98%
DIV 0xFFFFFFFE00000001, 0xFFFFFFFF (86)	38.1%	99%
DIV 0x0000000000000FA0, 0x14 (98)	37.3%	95.6%
AND 0xFFFFFFFF, 0xAAAAAAAA (100)	36%	99%
BGE (90)	40.6%	98.3%

the produced responses is partly biased. A small group of five chips partially contributes to a low value of uniqueness. Moreover, we present the uniqueness value without any post-processing. By using hashing or other techniques as has been used by several other PUFs the uniqueness can be improved. Moreover, the pre-transition region produces only '11' as response while the post-transition region produces only '00'. However, it still helps to distinguish one chip from another if two chips have different FTIs. The experimental results support that. For example, in Figure 7.5, from the frequency 120 MHz to 127.5 MHz, chip1 is in pre-transition region producing only '11' as responses. In the same frequency range, chip2 is in post-transition region producing only '00' as responses, and hence they are distinguished. The PUF shows high reliability. The reliability parameter is estimated based on 100 measurement samples. A value close to 100% indicates that the PUF responses are highly reliable at the normal operating condition. We plan the evaluation of environmental reliability (temperature, voltage) as future work.

Table 7.2: Secure response bit estimation chart

r at f_{i-1}	r at f_{i+1}	Possible r at f_i	No of secure bits at f_i
11	10	11, 10	1
11	01	11, 10, 01	1.5
11	00	11, 10, 01, 00	2
10	01	10, 01	1
10	00	10, 01, 00	1.5
01	00	01, 00	1

7.2.2 Security Analysis of the Proposed PUF

We estimate the number of secure response bits for authentication. By secure bits, we mean those response bits that cannot be predicted given the information about other response bits. The number of secure response bits is a function of the number of instructions we characterize, as well as the frequency spacing with which we sample the transition region (0.5 MHz in our experiment). These are the design parameters of the proposed PUF.

Let us assume that an adversary is attempting to predict the response r at a frequency f_i. We assume that she knows the value of r at the previous sampling frequency f_{i-1} and the next sampling frequency f_{i+1} ($f_{i-1} < f_i < f_{i+1}$). Then she can guess r at f_i to some extent. If r at f_{i-1} and r at f_{i+1} are same, it is obvious that r at f_i has the same value too. This is because with increase in frequency, *pc* can either stay the same or decrease. On the other hand if r at f_{i-1} and r at f_{i+1} are not same, different possibilities may arise. Table 7.2 shows the possibilities depending on all possible combination of r at f_{i-1} and r at f_{i+1} when they are different. Assuming the possible values of r at f_i are equally likely, we can estimate the number of secure bits at f_i using Shannon's formula. For example, the first row in the third column in Table 7.2 shows that the possible values of r at f_i is '11' or '10' i.e two possibilities. Hence the number of secure bits is $2 \times (1/2) \times (-log_2(1/2)) = 1$. In this

Table 7.3: Estimated Number of secure response bits

Chip1	Chip2	Chip3	Chip4	Chip5	Average
38	38	37	38	34	37

case, r cannot have values '01' and '00' as pc can either stay constant or can decrease with increase in frequency. Similarly, for the second row, it is $3 \times (1/3) \times (-log_3(1/3)) \approx 1.5$.

This analysis shows that secure bits are available at the transition regions. There are seven transition regions for five instructions with multiplication and division having two operations each. With each transition region producing around five secure bits on an average, each chip has 37 secure bits on an average as shown in table Table 7.3.

7.3 Related Work

Besides the SRAM PUF and the DFF-based PUF that have been already discussed, there are several other works that are relevant to this research. Suh et al. proposed a secure processor architecture using PUF as a source of secret [44]. The proposed architecture provided secure environment for trusted computing by preventing physical attacks on memory and tampering of program. This work did not propose any new PUF architecture; they used the Arbiter PUF in their architecture. In our work, we exploit the variability in a processor pipeline to build a new PUF. In another work, hardware authentication is achieved by calculating a checksum using micro-architectural complexity of a processor model [9]. This technique can distinguish one processor model from another; distinguishing one chip from the other is not possible by it. We instead focus on chip-level authentication.

Wong et al. used variable clock to characterize the delay variability in several components of an FPGA such as LUTs, flip-flops [52]. In our proposed PUF, we also employ variable clock frequency to a soft-core microprocessor in an FPGA in order to measure variability.

However, the work by Wong et al. only focused on characterizing variability in generic FPGA components. We instead built a PUF mechanism using the variability in a microprocessor pipeline. Majzoobi et al. used a very similar technique as proposed in [52] to build a PUF [34]. However, their PUF exploits the delay variability of FPGA building blocks and not of a microprocessor pipeline like ours. Brown et al. measured variability in FPGAs to build a variability model for general-purpose processors (GPPs) [5]. In their work, they measured the variability in GPPs by over-clocking them. We also use a similar mechanism of over-clocking to characterize variability in a processor pipeline. However, our purpose is to identify a chip using the variability information, whereas they proposed a low-cost method to model processor variability.

7.4 Summary

We introduced a microprocessor-intrinsic PUF. We introduced the concept supported by an on-chip implementation and experimental results. The experimental results shows a moderate value of uniqueness in the response bits, whereas the reliability is high at normal operating condition. As part of the future work, we plan to characterize those instructions that have not been tested. We would also like to test the reliability of the PUF under varying temperature and supply voltage. A detailed security analysis is also a part of the future plan.

Chapter 8

Conclusion and Future Work

In this research, we made an effort to answer those critical questions that are pivotal for a physical unclonable function to become an efficient security solution. We identified three main quality factors that determine the efficiency of a PUF. These factors are often degraded by several unwanted effects. To find out a systematic way to optimize these quality factors, a PUF system model has been proposed. The research effort made so far confirms the effectiveness of the model. Three PUF enhancement techniques have been proposed using the model: a compensation technique for the systematic process variation to improve the PUF randomness and robustness, a compact configurable ring oscillator technique to improve the reliability of a PUF, and a method to expand the number of CRPs of a PUF for better robustness. All of these techniques show significant improvement in the PUF quality factors and have been validated using on-chip implementations. Two of the proposed techniques achieved significant reduction in area footprint.

We also validated the efficiency of a PUF by characterizing it over a large group of chips. The characterization is not only done over a sizable group, it also captures the behavior of a PUF against different types of environmental variations as well as circuit aging. A combination of accelerated aging experiments as well as simulations was used to study the effect of aging on a PUF. We also proposed a mitigation technique against the aging effect.

In a summary, our characterization effort is comprehensive in nature as it covers a wide range of variability factors that affect a PUF.

The proposed PUF evaluation-comparison method is a step forward towards formalizing the performance evaluation of different PUFs and comparison among them. We not only proposed the method, we also presented a test case that showed interesting results. For example, we observed that the RO PUF demonstrated higher PUF qualities compared to the Arbiter PUF in spite of the fact that the RO PUF was implemented on a 90-nm chip while the Arbiter PUF was implemented on a 65-nm chip; one would expect that a smaller technology node will show higher process variation and hence better PUF quality. Furthermore, we made the RO PUF experimental data as well as the analysis programs available on the internet so that the research community can get an easy access to them. This way, other researchers can add their comments, leading to further refinement of the method. By monitoring the access log of the website, we found that the repository is regularly accessed across various locations in the world.

Finally, the new microprocessor-intrinsic PUF showed promising results in terms of identifying a chip exploiting the variability in the pipeline of a microprocessor. We believe this technique will lead to an easy and low-cost integration of PUFs into several systems and facilitate several different applications of PUFs.

A list of publications related to this research are mentioned below.

[1] A. Maiti and P. Schaumont, "A Novel Microprocessor-intrinsic Physical Unclonable Function", the International Conference on Field Programmable Logic and Applications (FPL), IEEE, 2012.**(under review)**

[2] A. Maiti, V. Gunreddy and P. Schaumont, "A Systematic Method to Evaluate and Compare the Performance of Physical Unclonable Functions", IACR ePrint, 2011.

[3] A. Maiti, L. McDougall and P. Schaumont, "The Impact of Aging on an FPGA-Based Physical Unclonable Function", Proceedings of the International Conference on Field Pro-

grammable Logic and Applications (FPL), IEEE, 2011. (Received the best paper award).

[4] A. Maiti, I. Kim and P. Schaumont, "A Robust Physical Unclonable Function with Enhanced Challenge-Response Set", Transactions on Information Forensics and Security, IEEE, 2011.

[5] A. Maiti and P. Schaumont, "Improved Ring Oscillator PUF: An FPGA-Friendly Secure Primitive, IACR Journal of Cryptology, Springer, 2010.

[6] A. Maiti, J. Casarona, L. McHale, and P. Schaumont, "A Large Scale Characterization of RO-PUF, Proceedings of the International Workshop on Hardware-Oriented Security and Trust (HOST)", IEEE, 2010.

[7] A. Maiti and P. Schaumont, "Improving the Quality of a Physical Unclonable Function using Configurable Ring Oscillators"', Proceedings of the International Conference on Field Programmable Logic and Applications (FPL), IEEE, 2009.

[8] A. Maiti, R. Nagesh, A. Reddy and P. Schaumont, Physical Unclonable Function and True Random Number Generator : a Compact and Scalable Implementation, Proceedings of the Great Lake Symposium on VLSI (GLSVLSI), ACM, 2009.

[9] Online repository of PUF data analysis. http://rijndael.ece.vt.edu/puf/

[10] I. Kim, A. Maiti, L. Nazhandali, P. Schaumont, V. Vivekraja, H. Zhang, "From Statistics to Circuits: Foundations for Future Physical Unclonable Functions," chapter in Hardware Intrinsic Security," eds. A. Sadeghi, Springer Information Security and Cryptography Series, 2011.

[11] M. Gora, A. Maiti and P. Schaumont, A Flexible Design Flow for Software IP Binding in FPGA, Transactions on Industrial Informatics (TII), IEEE, 2010.

[12] S. Morozov, A. Maiti and P. Schaumont, An Analysis of Delay Based PUF Implementations on FPGA, 6th International Symposium on Applied Reconfigurable Computing (ARC), LNCS, 2010.

[13] M. Gora, A. Maiti and P. Schaumont, A Flexible Design Flow for Software IP Binding in Commodity FPGA,Proceedings of the International Symposium on Industrial Embedded Systems (SIES), IEEE, 2009.

8.1 Future Work

We discuss possible directions to extend this research. Three directions are suggested. They are listed below.

- Novel quantization techniques for PUFs
- Extension of the PUF evaluation-comparison method
- Further study on the microprocessor-intrinsic PUF

8.1.1 Novel Quantization Technique for PUFs

In the proposed PUF system model, quantization is an essential component. A quantization method should have the ability to reproduce consistent binary responses from a PUF in the presence of noise. At the same time, producing uniformly distributed response bits is another requirement for the robustness of a PUF.

In literature, a simple comparison-based method has been employed for the RO PUF [45]. The equivalent resistant-based PUF employs a similar method to compare output voltages from the PUF circuit using an operational amplifier [16]. In the area of biometrics, a number of quantization method have been proposed to produce binary signature out of noisy biometric data. Dodis et al. proposed the concept of fuzzy extractor that can reconstruct keys from noisy biometric data using helper data [11]. Similar methods have been proposed in [26, 46].

There are few limitations in the existing quantization techniques that motivate new effort in the PUF quantization area. Simple comparison-based quantization is very easy to be attacked. For example, in an RO PUF, the ranking of the RO frequencies is sufficient to predict the challenge-response pairs. On the other hand, biometric-related quantization methods employ helper data which is publicly stored for reconstructing a key. As a result, it partially leaks the entropy of the key. Moreover, there are quantization methods, such as the one proposed in [46], that require global statistics e.g. global average of frequencies in an RO PUF. This makes them difficult to be implemented on a chip as updating global parameters dynamically across many chips is not easy. Hence, one of the suggested future work would be to propose a PUF quantization method that will be:

1. Robust against attack
2. Less dependent on helper data to reduce entropy leakage.
3. Easy to be implemented in hardware.

In this research, we so far proposed enhancements at the level of sample measurement and identity mapping. An effort along this direction will complete the exploration of the entire system model.

8.1.2 Extension of the PUF evaluation-comparison method

In Chapter 6, we introduced the idea of an evaluation-comparison method for PUFs. We proposed a set of seven parameters for this method. This method can be extended further, and additional parameters can be added. For example, Majzoobi et al. defined a set of PUF parameters while testing PUFs [35]. These parameters are useful to evaluate the robustness of a PUF. Armknecht also discussed several parameters to evaluate the performance of PUFs [1]. Furthermore, standard techniques such as NIST randomness test may also be applied on PUF responses.

Another important point that requires attention is the entropy estimation of PUFs. Entropy of a PUF is very critical for evaluating its robustness. However, there is no commonly used method to estimate the entropy of several PUFs and compare them. Ignatenko et al. proposed the use of context-tree weighting (CTW) method to estimate the entropy of a PUF [19]. They applied this method to estimate the secrecy rate of an optical PUF [38]. Their work did not discuss if the CTW method is applicable for PUFs in general or not.

Apart from the statistical analysis of the PUF responses, another approach to compare different PUFs would be to use implementation-related aspects of PUFs such as area, performance, and power.

8.1.3 Further study on the microprocessor-intrinsic PUF

We proposed the microprocessor-intrinsic PUF in Chapter 7. We demonstrated a preliminary idea based on a subset of the instruction set of a LEON3 microprocessor. Possible extensions of this work are listed below.

1. A characterization of the whole instruction set of the LEON3 microprocessor needs to be done along with variations in the input operands to the instructions. This will create a complete picture of the variability inherent in a general-purpose microprocessor pipeline.

2. The proposed PUF needs to be tested against variation in operating conditions such as temperature variation and supply voltage fluctuation. This will give us an idea about the reliability of the PUF. This is a critical part for the effectiveness of the proposed PUF.

3. We have tested only five chips so far with the proposed PUF. Characterizing the PUF using a larger set of devices should be done in order to obtain more reliable statistical evidence regarding the performance of the PUF.

4. To validate the portability of this technique, this PUF should be implemented on other microprocessor architecture such as Microblaze and PowerPC. Implementing this PUF in a device with a different technology node may also give us an idea about its portability.

5. We have done an analysis to estimate the number of response bits that are secure in terms of preventing prediction attack. However, a detailed threat analysis should be done in order to determine its robustness against several other attacks.

We have discussed possible directions of this research in future based on the effort we have made. However, there are many other areas in PUF research that can be looked at besides the suggested ones. For example, apart from model building attack, active attacks or side-channel attacks on PUFs have not been studied in detail. Mounting different types of attacks as well as finding out countermeasures against them is an area that might be interesting.

Bibliography

[1] F. Armknecht, R. Maes, A.-R. Sadeghi, F.-X. Standaert, and C. Wachsmann. A Formal Foundation for the Security Features of Physical Functions. *IEEE Security and Privacy 2011*, 2011(1):16, 2011.

[2] L. Bolotnyy and G. Robins. Physically unclonable function-based security and privacy in rfid systems. In *Pervasive Computing and Communications, 2007. PerCom '07. Fifth Annual IEEE International Conference on*, pages 211 –220, march 2007.

[3] D. S. Boning and S. Nassif. Models of process variations in device and interconnect. In *Design of High Performance Microprocessor Circuits, chapter 6*. IEEE Press, 1999.

[4] C. Bösch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls. Efficient helper data key extractor on fpgas. In *Proceeding sof the 10th international workshop on Cryptographic Hardware and Embedded Systems*, CHES '08, pages 181–197, Berlin, Heidelberg, 2008. Springer-Verlag.

[5] M. Brown, C. Bazeghi, M. Guthaus, and J. Renau. Measuring and modeling variabilityusing low-cost fpgas. In *Proceedings of the ACM/SIGDA international symposium on Field programmable gate arrays*, FPGA '09, pages 286–286, New York, NY, USA, 2009. ACM.

[6] J. D. R. Buchanan, R. P. Cowburn, A.-V. Jausovec, D. Petit, P. Seem, G. Xiong, D. Atkinson, K. Fenton, D. A. Allwood, and M. T. Bryan. Forgery: Fingerprinting documents and packaging. *Nature*, pages 436–475, July 2005.

[7] P. Bulens, F.-X. Standaert, and J.-J. Quisquater. How to strongly link data and its medium: the paper case. *Information Security, IET*, 4(3):125 –136, september 2010.

[8] G. Dejean and D. Kirovski. Rf-dna: Radio-frequency certificates of authenticity. In *Proceedings of the 9th international workshop on Cryptographic Hardware and Embedded Systems*, CHES '07, pages 346–363, Berlin, Heidelberg, 2007. Springer-Verlag.

[9] D. Y. Deng, A. H. Chan, and G. E. Suh. Hardware authentication leveraging performance limits in detailed simulations and emulations. In *Proceedings of the 46th Annual Design Automation Conference*, DAC '09, pages 682–687, New York, NY, USA, 2009. ACM.

[10] S. Devadas, E. Suh, S. Paral, R. Sowell, T. Ziola, and V. Khandelwal. Design and implementation of puf-based "unclonable" rfid ics for anti-counterfeiting and security applications. In *RFID, 2008 IEEE International Conference on*, pages 58–64, april 2008.

[11] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, Mar. 2008.

[12] G. Gogniat, T. Wolf, W. Burleson, J.-P. Diguet, L. Bossuet, and R. Vaslin. Reconfigurable hardware for high-security/ high-performance embedded systems: The safes perspective. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 16(2):144 –155, Feb. 2008.

[13] J. Guajardo, S. Kumar, G.-J. Schrijen, and P. Tuyls. Brand and ip protection with physical unclonable functions. In *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pages 3186 –3189, may 2008.

[14] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls. Fpga intrinsic pufs and their use for ip protection. In *Proceedings of the 9th international workshop on Cryptographic*

Hardware and Embedded Systems, CHES '07, pages 63–80, Berlin, Heidelberg, 2007. Springer-Verlag.

[15] G. Hammouri, A. Dana, and B. Sunar. Cds have fingerprints too. In *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '09, pages 348–362, Berlin, Heidelberg, 2009. Springer-Verlag.

[16] R. Helinski, D. Acharyya, and J. Plusquellic. A physical unclonable function defined using power distribution system equivalent resistance variations. In *Proceedings of the 46th Annual Design Automation Conference*, DAC, pages 676–681, New York, NY, USA, 2009. ACM.

[17] D. Holcomb, W. Burleson, and K. Fu. Power-up sram state as an identifying fingerprint and source of true random numbers. *Computers, IEEE Transactions on*, 58(9):1198 –1210, sept. 2009.

[18] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh. Quantitative and statistical performance evaluation of arbiter physical unclonable functions on fpgas. In *Reconfigurable Computing and FPGAs (ReConFig), 2010 International Conference on*, pages 298 –303, dec. 2010.

[19] T. Ignatenko, G.-J. Schrijen, B. Skoric, P. Tuyls, and F. Willems. Estimating the secrecy-rate of physical unclonable functions with the context-tree weighting method. In *Information Theory, 2006 IEEE International Symposium on*, pages 499 –503, july 2006.

[20] R. S. Indeck and M. Muller. Method and apparatus for fingerprinting magnetic media. *US Patent No. 5365586*, November 1994.

[21] M. S. Kirkpatrick and E. Bertino. Software techniques to combat drift in puf-based authentication systems. In *Workshop on Secure Component and System Identification (SECSI 2010)*, page 9, 2010.

[22] S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls. Extended abstract: The butterfly puf protecting ip on every fpga. In *Hardware-Oriented Security and Trust, 2008. HOST 2008. IEEE International Workshop on*, pages 67 –70, 2008.

[23] Y. Li and W. Chu. Implementation of single precision floating point square root on fpgas. In *FPGAs for Custom Computing Machines. Proceedings., The 5th Annual IEEE Symposium on*, pages 226 –232, Apr. 1997.

[24] D. Lim, J. Lee, B. Gassend, G. Suh, M. van Dijk, and S. Devadas. Extracting secret keys from integrated circuits. *Very Large Scale Integration Systems, IEEE Transactions on*, 13(10):1200 – 1205, 2005.

[25] J.-P. Linnartz and P. Tuyls. New shielding functions to enhance privacy and prevent misuse of biometric templates. In *Proceedings of the 4th international conference on Audio- and video-based biometric person authentication*, AVBPA'03, pages 393–402, Berlin, Heidelberg, 2003. Springer-Verlag.

[26] J.-P. M. G. Linnartz and P. Tuyls. New shielding functions to enhance privacy and prevent misuse of biometric templates. In *AVBPA*, pages 393–402, 2003.

[27] K. Lofstrom, W. Daasch, and D. Taylor. Ic identification circuit using device mismatch. In *Solid-State Circuits Conference, 2000. Digest of Technical Papers. ISSCC. 2000 IEEE International*, pages 372 –373, 2000.

[28] D. Lorenz, G. Georgakos, and U. Schlichtmann. Aging analysis of circuit timing considering nbti and hci. In *On-Line Testing Symposium, (IOLTS). 15th IEEE International*, pages 3 –8, June 2009.

[29] R. Maes, P. Tuyls, and I. Verbauwhede. Intrinsic pufs from flip-flops on reconfigurable devices. In *3rd Benelux Workshop on Information and System Security (WISSec 2008)*, page 17, Eindhoven,NL, 2008.

[30] R. Maes, P. Tuyls, and I. Verbauwhede. Low-overhead implementation of a soft decision helper data algorithm for sram pufs. In *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*, CHES '09, pages 332–347, Berlin, Heidelberg, 2009. Springer-Verlag.

[31] R. Maes and I. Verbauwhede. Physically unclonable functions: A study on the state of the art andfuture research directions. In *Towards Hardware-Intrinsic Security*. Springer, 2010.

[32] A. Maiti, J. Casarona, L. McHale, and P. Schaumont. A large scale characterization of ro-puf. In *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, pages 94 –99, 2010.

[33] A. Maiti, I. Kim, and P. Schaumont. A robust physical unclonable function with enhanced challenge-response set. *Information Forensics and Security, IEEE Transactions on*, 7(1):333 –345, feb. 2012.

[34] M. Majzoobi, A. Elnably, and F. Koushanfar. Fpga time-bounded unclonable authentication. In *Proceedings of the 12th international conference on Information hiding*, IH'10, pages 1–16, Berlin, Heidelberg, 2010. Springer-Verlag.

[35] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Testing techniques for hardware security. In *Test Conference, 2008. ITC 2008. IEEE International*, pages 1 –10, Oct. 2008.

[36] S. Morozov, A. Maiti, and P. Schaumont. An analysis of delay based puf implementations on fpga. In P. Sirisuk, F. Morgan, T. El-Ghazawi, and H. Amano, editors, *Reconfigurable Computing: Architectures, Tools and Applications*, volume 5992 of *Lecture Notes in Computer Science*, pages 382–387. Springer Berlin / Heidelberg, 2010.

[37] H. Onodera. Variability modeling and impact on design. In *Electron Devices Meeting, 2008. IEDM 2008. IEEE International*, pages 1 –4, dec. 2008.

[38] R. S. Pappu, B. Recht, J. Taylor, and N. Gershenfeld. Physical one-way functions. *Science*, 297:2026–2030, 2002.

[39] U. Rührmair, F. Sehnke, J. S ölter, G. Dror, S. Devadas, and J. u. Schmidhuber. Modeling attacks on physical unclonable functions. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, pages 237–249, New York, NY, USA, 2010. ACM.

[40] P. Sedcole and P. Y. K. Cheung. Within-die delay variability in 90nm fpgas and beyond. In *Field Programmable Technology, 2006. FPT 2006. IEEE International Conference on*, pages 97 –104, dec. 2006.

[41] F. Sehnke, C. Osendorfer, J. Sölter, J. Schmidhuber, and U. Rührmair. Policy gradients for cryptanalysis. In *Proceedings of the 20th international conference on Artificial neural networks: Part III*, ICANN'10, pages 168–177, Berlin, Heidelberg, 2010. Springer-Verlag.

[42] E. A. Stott, J. S. Wong, P. Sedcole, and P. Y. Cheung. Degradation in fpgas: measurement and modelling. In *Proceedings of the 18th annual ACM/SIGDA international symposium on Field programmable gate arrays*, pages 229–238, New York, NY, USA, 2010. ACM.

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

[44] G. Suh, C. O'Donnell, I. Sachdev, and S. Devadas. Design and implementation of the aegis single-chip secure processor using physical random functions. In *Computer Architecture, 2005. ISCA '05. Proceedings. 32nd International Symposium on*, pages 25 – 36, june 2005.

[45] G. E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th annual Design Automation Conference*, DAC '07, pages 9–14, New York, NY, USA, 2007. ACM.

[46] P. Tuyls, A. Akkermans, T. Kevenaar, G.-J. Schrijen, A. Bazen, and R. Veldhuis. Practical biometric authentication with template protection. In T. Kanade, A. Jain, and N. Ratha, editors, *Audio- and Video-Based Biometric Person Authentication*, volume 3546 of *Lecture Notes in Computer Science*, pages 436–446. Springer Berlin / Heidelberg, 2005. 10.1007/1152792345.

[47] P. Tuyls, G.-J. Schrijen, B. Škorić, J. van Geloven, N. Verhaegh, and R. Wolters. Read-proof hardware from protective coatings. In *Cryptographic Hardware and Embedded Systems Workshop*, volume 4249 of *LNCS*, pages 369–383. Springer, October 2006.

[48] V. van der Leest, G.-J. Schrijen, H. Handschuh, and P. Tuyls. Hardware intrinsic security from d flip-flops. In *Proceedings of the fifth ACM workshop on Scalable trusted computing*, STC '10, pages 53–62, New York, NY, USA, 2010. ACM.

[49] V. Vivekraja and L. Nazhandali. Circuit-level techniques for reliable physically uncloneable functions. In *Proceedings of the 2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, HST '09, pages 30–35, Washington, DC, USA, 2009. IEEE Computer Society.

[50] S. Vrijaldenhoven. Acoustical physical uncloneable functions. *Masters thesis, Technische Universiteit Eindhoven, the Netherlands*, October 2005.

[51] N. Weste and D. Harris. *CMOS VLSI Design: A Circuits and Systems Perspective, (4th edition)*.

[52] J. S. J. Wong, P. Sedcole, and P. Y. K. Cheung. Self-measurement of combinatorial circuit delays in fpgas. *ACM Trans. Reconfigurable Technol. Syst.*, 2:10:1–10:22, June 2009.

[53] D. Yamamoto, K. Sakiyama, M. Iwamoto, K. Ohta, T. Ochiai, M. Takenaka, and K. Itoh. Variety enhancement of puf responses based on the locations of random outputting rs latches. In *Proceedings of the 13th international conference on Cryptographic hardware and embedded systems*.

[54] D. Yamamoto, K. Sakiyama, M. Iwamoto, K. Ohta, T. Ochiai, M. Takenaka, and K. Itoh. Uniqueness enhancement of puf responses based on the locations of random outputting rs latches. In *Proceedings of the 13th international conference on Cryptographic hardware and embedded systems*, CHES'11, pages 390–406, Berlin, Heidelberg, 2011. Springer-Verlag.

[55] C.-E. Yin and G. Qu. Temperature-aware cooperative ring oscillator puf. In *Proceedings of the 2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, HST '09, pages 36–42, Washington, DC, USA, 2009. IEEE Computer Society.

[56] C.-E. Yin and G. Qu. Lisa: Maximizing ro puf's secret extraction. In *Hardware-Oriented Security and Trust (HOST), 2010 IEEE International Symposium on*, pages 100 –105, June 2010.

[57] H. Yu, P. Leong, H. Hinkelmann, L. Moller, M. Glesner, and P. Zipf. Towards a unique fpga-based identification circuit using process variations. In *Field Programmable Logic and Applications, 2009. FPL 2009. International Conference on*, pages 397 –402, 2009.

[58] M.-D. M. Yu and S. Devadas. Secure and robust error correction for physical unclonable functions. *IEEE Des. Test*, 27:48–65, January 2010.