7-7-2020

Hybrid PUF Design using Bistable Ring PUF and Chaotic Network

Madhan Kumar Thirumoorthi
University of Windsor

Hybrid PUF Design using Bistable Ring PUF and Chaotic Network

By

Madhan Thirumoorthi

A Thesis

Submitted to the Faculty of Graduate Studies

through the Department of Electrical and Computer Engineering

in Partial Fulfillment of the Requirements for

the Degree of Master of Applied Science

at the University of Windsor

Windsor, Ontario, Canada

2020

Hybrid PUF Design using Bistable Ring PUF and Chaotic Network

By

Madhan Thirumoorthi

APPROVED BY:

E. Tam
Department of Civil & Environmental Engineering

M. Azzouz
Department of Electrical & Computer Engineering

M. Khalid
Department of Electrical & Computer Engineering

M. Mirhassani, Advisor
Department of Electrical & Computer Engineering

April 27^{th}, 2020

Declaration of Originality

I hereby certify that I am the sole author of this thesis and that no part of this thesis has been published or submitted for publication.

I certify that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

Physical Unclonable Function(PUF) is lightweight hardware that provides affordable security for electronic devices and systems which can eliminate the use of the conventional cryptographic system which uses large area and storage. Among the several models, Bi-stable Ring PUF(BR-PUF) is considered as a secure and efficient PUF model since it has no mathematical model still found. In this thesis, we proposed a modified design called a hybrid model of BR-PUF and a Chaotic network to improve the BR-PUF resilience against machine learning attacks. We experimented with the current modification XOR technique to analyze the uniqueness, reliability and resource consumption. The proposed PUF was implemented on Xilinx Artix 7 FPGA and the PUF metrics were captured and compared with the results of XOR-ed based PUF integration techniques. The lightweight PUF model was achieved with 16% resource reduction when compared to XOR-ed BR PUF with no compromise in PUF quality.

I dedicate my thesis to my father.

Acknowledgments

I wish to express my most sincere gratitude to my advisor Dr. Mitra Mirhassani who has been more than an advisor to me. she was the source of motivation and constant support throughout the course of this work.

I would like to extend my gratitude to my Co-Advisor Dr. Mohammed A. S. Khalid for helping to acquire better knowledge of FPGA CAD Tools.

I would like to thank my committee members; Dr. Maher Azzouz and Dr. Edwin Tam for their motivation and guidelines.

I would also like to thank my colleagues and friends Bharani, Nishanth, Bharath and Hari Krishnan for their help and support.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

PUF	Physical Unclonable Functions
VLSI	Very-Large Scale Integration
FPGA	Field Programmable Gate Array
CLB	Configurable Logic Blocks
HT	Hardware Trojan
NVM	Non-Volatile Memory
RFID	Radio-Frequency Identification
RO	Ring Oscillator
BR	Bi-stable Ring
SRAM	Static Random Access Memory
LUT	Look-Up Tables
MUX	Multiplexor
DEMUX	Demultiplexor
FF	Flip-Flop
HDL	Hardware Description Language
HLS	High Level Synthesis
RTL	Register Transfer Level
IP	Intellectual Property
IoT	Internet of Things
CRP	Challenge-Response Pairs
ML	Machine Learning

Chapter 1

Introduction

1.1 Motivation

With the rapid development of the Internet of Things (IoT), security has attracted considerable interest. The embedded system and mobile devices encourage interconnection stages, which empowers various applications, for example, the Internet of Things, Banking and online networking.

This interconnected platform handles secret information that should be ensured with security. The present best practice for this sort of security application is to make a secret key by utilizing the cryptographic algorithm and store it in non-volatile memory, for example, EEPROM/battery-powered RAM. It further uses cryptographic functions, for example, digital signature and encryption systems to guarantee the security of the devices[1].

Secret keys are commonly stored in non-volatile memory, which usually is challenging to store and verify. This approach is expensive both in terms of design area and power consumption. They may not be relevant to resource-constrained applications, for example, RFID Tags. These storage units are physically exposed

and open to attack. These attacks include obtrusive, semi intrusive and side-channel attacks, which can lead to crucial exposure and full security breaks.

Compared with software security, hardware security can increase the expense and difficulties of attacks. Simultaneously, the high computational requirements of encryption algorithms requires more utilization of hardware resources. Additionally, classic encryption algorithms perform complex mathematical operations with keys. Subsequently, the security of the cryptographic key decides the security of the entire system[1].

Conventionally, the key is stored in Non-Volatile Memory (NVM), which can be exposed by non-invasive or invasive attacks. Moreover, a secure cryptographic algorithm requires a substantial amount of resources and time, which are typically not accessible in IoT devices. Hardware-based security primitives can give faster and safer authentication schemes than conventional cryptographic systems.

1.2 Introduction to Hardware Security

The limited hardware resources of IoT nodes prompted the advancement of Physical Unclonable Function(PUF). A PUF can extract the unique value from Integrated Circuits (ICs) from process variations that happen during the manufacturing process and evaluate internal mismatches using binary sequences[2]. Regardless of whether the structure of any two chips is the equivalent, there will be minor contrasts because of manufacturing variations.

A PUF can extract these differences and provide each chip with a unique identification, the same as a digital fingerprint for the chip. As this IC fingerprint is acquired from the random variations in the circuit manufacturing process, even the chip manufacturer cannot duplicate a PUF within a series of manufactured chips.

Moreover, the response of a PUF is electrical and naturally generated and vanishes when the power is turned off. A PUF can be utilized for online key generation or authentication. For instance, the key is created from the PUF module only when needed. This is more secure than storing the key in NVM as utilized in the conventional encryption schemes[2]. The use of smart cards, RFID tags, credit and debit cards are a common application of hardware security. However, they are also susceptible to many attacks, such as side-channel attacks, to trace the key stored in digital forms[3].

The described situation was one motivation that led to the development of Physical Unclonable Function.

1.3 Physically Unclonable Function

PUF technology is considered to be particularly appropriate for IoT security due to its unique features.

The main advantages of the PUF are that it is low in cost, and offers fast authentication. PUF can provide integrated and lightweight security primitives for secure communication in IoT[4]. Although the cost of PUF structures should be as low as possible for IoT devices, many PUF proposals in the literature suffer from consuming significant hardware resources. A Physical Unclonable Function or a PUF is a die–specific random function that is unique for every instance of the die.

PUFs derive their randomness from the uncontrolled random variations in the IC manufacturing process to create practically unclonable functions[5]. PUFs are recently produced by many multinational companies and are widely being used for their security applications because of their randomness nature. Some of the security applications are IP Piracy, Device Authentication, injecting a Hardware

Trojan (HT) in the IC or an IP or an FPGA such that they can be made to use at evaluation time and then we can disrupt the functionality after the specified time[4].

1.4 Classification of PUF

The two most important subtypes of PUFs, which should be distinguished explicitly in any sound treatment of the topic, are called "Strong PUFs" and "Weak PUFs" [1]. They are discussed in this and the upcoming section.

1.4.1 Strong PUFs

Strong PUFs derive a more complex challenge-response behavior from the physical disorder present in the PUF. Typically, many physical components are involved in the generation of a response, and there is a very large number of possible challenges that can be applied to the PUF. Strong PUFs are usually employed with a publicly available CRP interface,i.e., anyone holding the PUF or the PUF embedding hardware can observe challenges and read out responses. The lack of access limit mechanisms on strong PUFs is, therefore, a key difference from weak PUFs[1]. In recent years, strong PUFs have turned out to be a very versatile cryptographic and safety primitive: First of all, by using a fixed set of challenges, they may be employed for inner key derivation, just like weak PUFs. They can also implement several advanced cryptographic protocols, ranging from identification to key exchange to oblivious transfer. They can be subsumed as follows:

1.4.1.1 Many CRPs

Strong PUFs have a very large number of possible challenges, ideally (but not necessarily) exponentially many challenges in some system parameters. This prevents a full read-out of all CRPs, even if an adversary holds physical possession of the PUF for a considerable time.

1.4.1.2 Unpredictability

Even if an adversary knows a large subset of CRPs, he cannot extrapolate or predict the other, yet unknown CRPs.

1.4.1.3 Unprotected Challenge-Response Interface

In all but very few applications of Strong PUFs, it is assumed that it has a free, publicly accessible challenge-response interface (or a freely accessible challenge-response mechanism, respectively). Anyone holding physical possession of the PUF or the PUF-carrying hardware can apply arbitrary challenges to the Strong PUF and read out the corresponding responses.

1.4.2 Weak PUFs

Weak PUFs essentially are a new form of storing secret keys in vulnerable hardware, offering an alternative to ROM, Flash or other non-volatile memories (NVMs)[1]. As all PUFs, Weak PUFs exhibit some internal, unclonable physical disorder, and possess some form of challenge-response mechanism that exploits this disorder.

1.4.2.1 Few Challenge

A Weak PUF has got very few, fixed challenges, commonly only one challenge per PUF instance.

1.4.2.2 Access Restricted Responses

In all but very few applications, the challenge-response interface (or the challenge-response mechanism, respectively) of a Weak PUF needs to be access-restricted. It is assumed that adversaries cannot access to the Weak PUF's responses, even if they hold physical possession of the PUF-carrying hardware.

1.5 Applications of Physical Unclonable Function

PUF technology is considered to be particularly appropriate for IoT security due to its unique features, which are unclonable, low in cost and offer fast authentication. The important feature of the PUF exploits from the circuit's internal delay and manufacturing process variations.

1.5.1 Authentication

This is one of the main applications for the PUF, which is widely used with less hardware overhead by a challenge-response protocol. A secure database that stores all the set of CRPs from each PUF pair to use[6]. When we need to check the authenticity of the circuit, the set of CRPs are queried randomly from the database and are applied to the PUF circuit. The response which is obtained is stored in the database and checked whether it matches the response from the

database for the IC or FPGA. If it matches, then we can authenticate the FPGA or the IC.

The important feature of the PUF is we can use different CRPs, and they all are random because of the manufacturing process of the IC or the FPGA since all the circuits are not doped in the same concentration to behave similarly and give the same time delay to produce the output[6]. We use this property of the PUF to build authentication for the device. It is needed to have a strong PUF for authentication, as we can have a lot of CRPs for authentication.

1.5.2 Secret key and Random Number Generation

Modern Cryptographic Primitives need the fundamentals for generating the Random Number Generator(RNG) and secret key for the message authentication and secret key generation[6]. The utilization of PUF response as secret keys was proposed, and it should be guaranteed that each bit of the response is reliable.

The PUFs cannot completely produce the exact response due to noise and environmental conditions, so we need some error correction processes. During this process, an error syndrome is evaluated when the challenge is applied, and this syndrome is utilized when it is being utilized for reconstruction of the PUF response, which may have a few mistakes because of noise and environmental conditions. Subsequent to computing and evaluating the PUF response, we hash the response, so we get the secret key that can be utilized officially for authentication. The most significant part is we have to generate random numbers based on this process with a low area overhead.

1.6 Chaos Based Cryptography

Chaos is one of the potential practices related to the development of a nonlinear physical framework and happens for explicit estimations of system parameters. The disclosure of this evidently irregular conduct following out of deterministic system ended up being very progressive, prompting numerous issues interconnecting security hypothesis, new geometrical highlights, and new marks portraying dynamical characteristics. Given an initial condition of a deterministic system, it is notable that the future conditions of the system can be anticipated. In any case, for the confusing system, long term expectation is inconceivable. For values of parameters, two directions, which are at the first close, wander exponentially in a short timeframe. This is the main reason for the chaos-based cryptography being used in security applications such as encryption and authentication. We used this chaos principle in the PUF technology to eliminate the drawbacks of existing system.

1.7 Thesis objective

In this research, we proposed, designed and evaluated a new hybrid PUF model which is more lightweight compared to existing XOR-ed Bistable Ring PUF. The proposed PUF model was tested on Xilinx Artix 7 FPGA. The experimental study was conducted and evaluation metrics such as uniqueness, reliability and resource utilization were computed for existing XOR based Bi-stable PUF model. After that the hybrid PUF was proposed with one dimensional chaotic network and experimental evaluation was conducted. Finally, both the experimental results for existing and proposed PUF model were compared. A good lightweight PUF model was achieved for the FPGA security applications.

1.8 Thesis Outline

In this thesis, We begin with introduction to PUF technology followed by literature review in chapter II. Chapter III explores the shortcoming of an existing XOR-ed BR-PUF with experimental study and suggests ways to improve. Chapter IV presents the design and implementation of Hybrid BR-PUF using chaotic networks. Chapter V explains the experimental results and analysis of the proposed PUF model on FPGA and concludes the thesis with a summary and recommendations for future work.

Chapter 2

Literature Review

2.1 Physical Unclonable Functions

Physical Unclonable Functions (PUF) are primitives that generate precise chip-specific signatures dynamically by the use of the process variations inside the silicon chip due to fabrication variation [1]. PUFs are classified into two types, namely strong and weak PUFs. The strong PUFs can produce multiple random responses by accepting challenges as input and mapping every challenge to a corresponding response such that we will have a report of the specific challenge-response pair and using those for authentication. On the alternative hand, Weak PUFs generate only a limited number of responses.

A PUF makes use of random differences inside the IC manufacturing process to provide precise identity tags for chips. It can take in a challenge and then return a response as its output signal. The values of the undertaking-response pairs (CRPs) are unique for each chip.

CRPs are used for authentication applications primarily based on PUFs [7]. The relationship between challenge and response is decided through the circuit itself and must be specific and preferably unpredictable. Therefore, a PUF is like

a random process P with an input C that produces an output R, as defined in Equation 2.1. The traits of the random method are determined while the PUF is established, and the characteristics of every PUF are different.

$$R = PUF(C) \tag{2.1}$$

where PUF is the internal characteristic of IC, C is the input challenge and R is the output response

2.2 PUF Metrics

To compare the overall performance of a PUF circuit, several metrics had been proposed,which includes *uniqueness* and *reliability*. These two metrics are the most usually used to evaluate PUFs and are also used on this work [8]. Furthermore, we endorse resource utilization metric to examine the resource usage of a PUF applied on FPGA.

2.2.1 Uniqueness

Uniqueness measures the difference between the responses of a PUF, implemented on different chips, under the same challenge. The intra-Hamming distance, i.e., uniqueness is given as follows:

$$U = \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{(HD(P_i(C), P_j(C))}{n} \times 100\% \tag{2.2}$$

where k is the total number of chips,$P_i(C)$ is the response of the i^{th} chip and $P_j(C)$ is the response of the j^{th} chip under the same challenge C.

The ideal value of uniqueness should be close to 50%, indicating the maximum difference in the responses of two PUFs. The Uniqueness denotes the randomness of the responses, and therefore, it is also related to the security of a PUF.

2.2.2 Reliability

Reliability is an important metric of PUF performance, which is characterized by comparing the Hamming distances of output responses generated by a same challenge, however, under different environmental conditions such as aging, temperature variations[9]. Ideally, the response of a PUF ought to remain identical with the same challenge. However, as PUFs extract small differences in chips, the output of a PUF is unavoidably sensitive to a change inside the running environment. Reliability displays the PUF's balance and its ideal value should be near to 100%, indicating that the PUF response value is stable and no bit flips occur inside the response sequence below different test conditions. The equation for reliability is as follows:

$$R = \left(1 - \frac{1}{m}\sum_{t=1}^{m}\frac{HD(P(C_i,t_0),P(C_i,t))}{n}\right) \times 100\% \tag{2.3}$$

where n is the sequence length of the response and m is the number of tests. The $P(C_i,t)$ denotes the t_th sample of $P(C_i)$ among m repeated responses. For a PUF device, the reliability is established as 1 minus the average value of the inter-Hamming distance of the response samples under different operating conditions.

2.3 Properties of PUF

The PUFs employ two properties i.e intra-chip variations and inter-chip variations.

2.3.1 Intra-Chip Variation

It is defined by the hamming distance between the responses of a PUF for the same given challenge. Each time a challenge is given to a PUF, due to environmental parameters some bits of its final response can be different from those that are expected by the verifier. To achieve a high level of robustness we want the intra-chip variation to be as less as possible, so the verifier can accept the slightly different response as a valid answer

2.3.2 Inter-Chip Variation

It is defined by the hamming distance between the responses of different PUFs for the same given challenge. If the construction process of the PUFs is biased, the sequence of bits of the final response will not be equally distributed. This will compromise the security as an adversary could have more chances to predict the response of the PUF. The best possible inter-chip variation is 50% since this means that the response bits have equal probability to have logic value "0" or "1".

2.4 Existing PUF Models

2.4.1 Arbiter PUF

Arbiter PUF was among the first set of Silicon PUF circuits to be proposed [10]. This PUF circuit leverages the delay variations across chips to generate unique challenge-response pairs. The Arbiter PUF circuit consists of a set of delay stages followed by an Arbiter circuit as shown in Figure 2.1. Each delay stage consists of two multiplexers with the inputs connected. The challenge bits form the select inputs to the multiplexers that decide the path taken by the top and bottom signals [10]. To evaluate the response bit for a particular challenge,

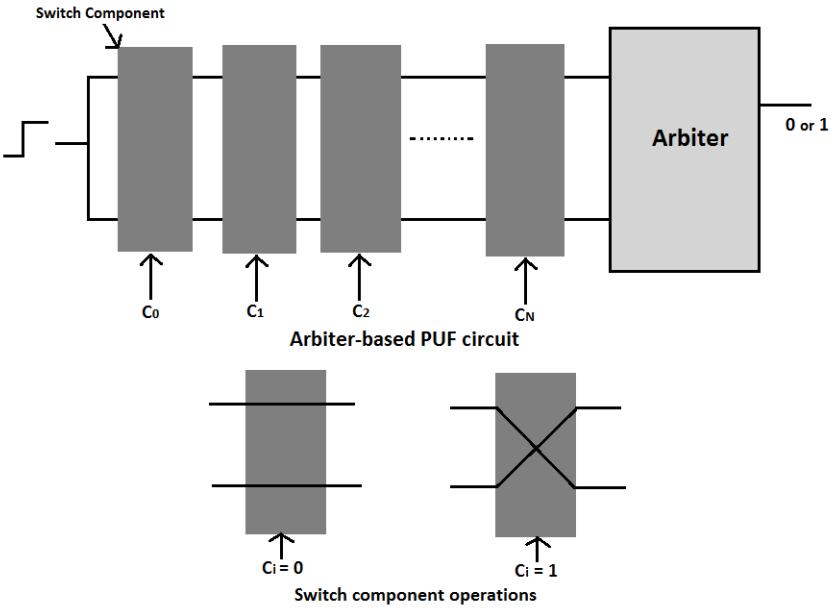

FIGURE 2.1: Schematic of Arbiter PUF

an input rising edge is propagated through the delay stages. The response bit is determined to be a 1 or 0 based on the top and bottom signal arrival times. In this PUF circuit, a D latch is used as the arbiter to determine which signal arrived first.

The arbiter PUF circuit implementation is a robust construction that supports an exponential number of challenge-response pairs. For instance, a 64-bit Arbiter PUF can support 2^{64} CRPs. Another key property of this PUF circuit is that it relies on relative comparison to generate CRPs. This improves the reliability of the PUF circuit in the presence of environmental variations significantly. The exponential number of delay paths available makes this circuit hard to model. However, the Arbiter PUF is a linear structure in which the cumulative path delay can be assumed to be a sum of the individual stage delays [10]. By assuming an additive delay model, a software model can be created through Support Vector Machines (SVM). SVM uses a set of challenge-response pairs as training samples to construct the model. This model can be used to predict other challenge-response pairs with a high degree of accuracy. Prediction rates greater than 90% can be

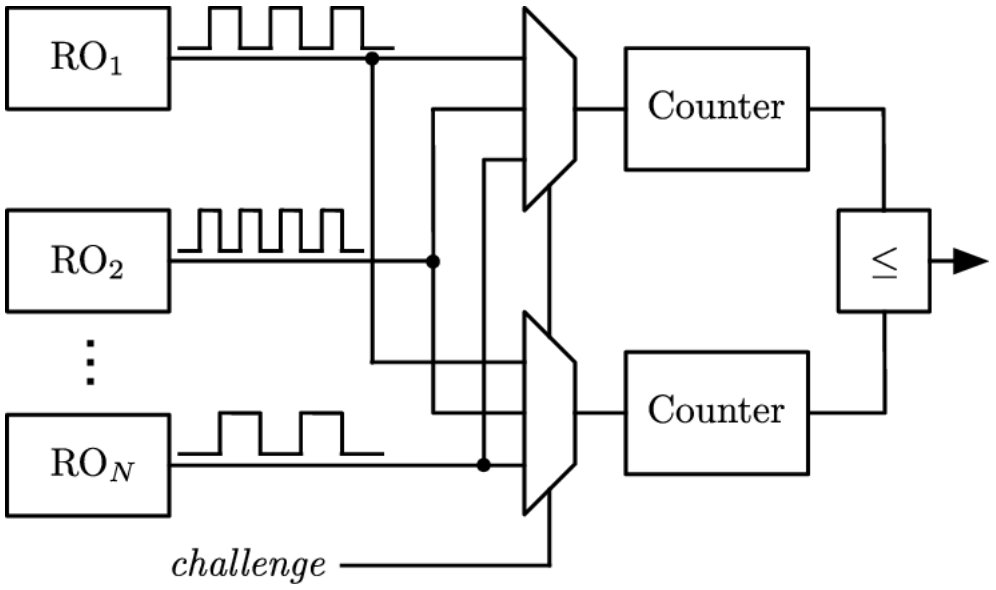

FIGURE 2.2: Schematic of Ring Oscillator PUF model with N ring oscillators

achieved through SVM attacks [11].

2.4.2 Ring Oscillator PUF

The manufacturing variability intrinsic to circuit gate delay can also be used to instantiate a "Ring Oscillator PUF"[12]. This PUF architecture contains N identically designed ring oscillators synthesized onto a field-programmable gate array(FPGA) or an application-specific integrated circuit(ASIC).

Due to the variation in delay of the inverters in the ring oscillator, each will have a slightly different frequency. The frequencies of two oscillators are measured and compared to reveal one of the PUF output bits. If there are N oscillators, there are N(N-1)/2 possible pairings [12]. However, the number of output bits is limited due to correlations (if ring oscillator A is faster than B, and B is faster than C, then clearly A is faster than C). For N oscillators, there is a specific ordering of fastest to slowest. If the oscillators are truly identical and manufacturing 10 variations dominate, then each of these N! orderings is equally likely. Therefore, there are a maximum of log(N!) bits that can be extracted from the PUF.

Note that the ring-oscillator PUF is weak since there are a limited number of "challenge bits" that can configure the PUF's operation. Once fabricated, the ring oscillator's frequency is set, so the output bits of the PUF will always remain constant. Because the ring-oscillator PUF measures differences in gate delay like the arbiter PUF, the ring-oscillator PUF is susceptible to the same set of environmental variations and noise sources. Therefore, error correction will be equally important in this application. In this architecture, several oscillator PUF banks are instantiated, with each oscillator bank comprising 2K ring oscillators [12]. A K-bit challenge is applied to each bank, to determine which oscillators correspond to the top delays, and which oscillators correspond to the bottom delays. The top and bottom rows are summed to produce x and y, respectively. These values are used to produce a single bit PUF output and associated "soft-decision" information corresponding to a PUF challenge. Specifically, the output bit is the sign of x–y.

2.4.3 Bistable Ring PUF

Bi-stable Ring is a ring involving a even number of inverting gates such gate configuration will act like uninitialized SRAM cells and will fall into any of these two states either "101010..." or "010101..."[13].

As depicted in Figure 2.3, an N stage BR PUF is made out of n phases, where each stage has two inverting delay gates (NOR gate for instance). A challenge vector C = C_1,C_2, ... ,C_n} chooses the NOR gates utilized in each Bi-stable ring configurations by giving values to the Multiplexer(MUX) and Demultiplexer (DEMUX) gates of the stages. Since each NOR gate has unique propagation time delay due to the manufacturing process variation and each input challenge bits makes a unique Bi-stable ring and altogether 2^N unique setups can be made. A typical "RESET" signal is added to each phase to set up a known "0" state before letting the ring settle to deliver its response. When "RESET" is set low or pulled

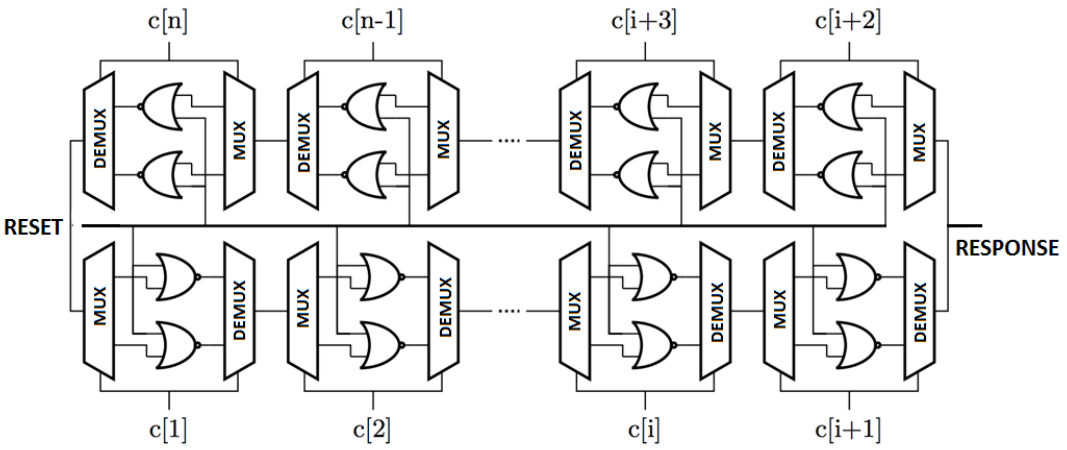

FIGURE 2.3: Schematic of a Bi-stable Ring PUF with n stage

off and the ring begins to oscillate through all the selected NOR gate [13]. When the ring achieved a stable state, the output of the selected NOR will be either "101..." or "010...".

The decision among the two stable conditions of the ring relies upon manufacturing and process variations of the NOR gates utilized in the ring configuration [13]. Any interconnection nodes between the two phases can be utilized as a response port.

2.4.4 SRAM PUF

Both the arbiter PUF and the ring-oscillator PUF ultimately depend on variations in the propagation delay of gates. However, this is not the only physical property on which a PUF can be built [14]. A popular weak PUF structure exploits the positive feedback loop in a SRAM or SRAM like structure shown in Figure 2.4. A SRAM cell has two stable states (used to store a 1 or a 0), and positive feedback to force the cell into one of these two states and, once it is there, prevent the cell from transitioning out of this state accidentally.

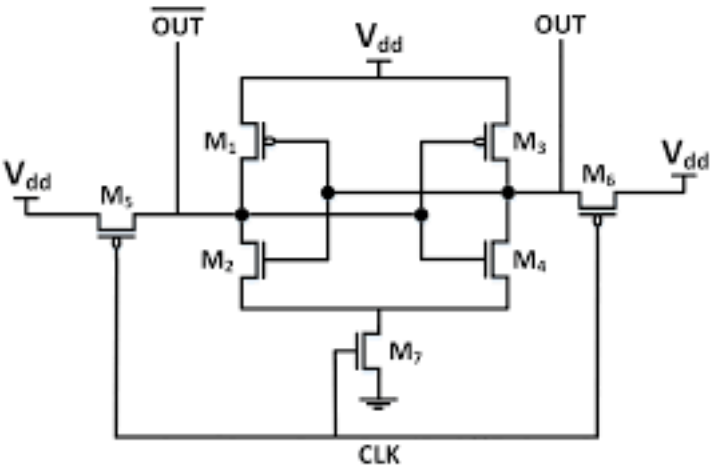

FIGURE 2.4: Schematic of Static-RAM (SRAM) PUF

A write operation forces the SRAM cell to transition toward one of the two states. However, if the device powers up and no write operation has occurred, the SRAM cell exists in a metastable state where theoretically, the feedback pushing the cell toward the '1' state equals the feedback pushing the cell toward the '0' state, thereby keeping the cell in this metastable state indefinitely [14]. In actual implementations, however, one feedback loop is always slightly stronger than the other due to small transistor threshold mismatches resulting from process variation. Natural thermal and shot noise trigger the positive feedback loop, and the cell relaxes into either the '1' or '0' state depending on this process variation. Note that since the final state depends on the difference between two feedback loops, the measurement is differential.

Therefore, common mode noise such as die temperature, power supply fluctuations, and common mode process variations should not strongly impact the transition. Like other strong and weak PUF implementations, the SRAM PUF is also sensitive to noise. If the two feedback loops of the SRAM cell are sufficiently close, then random noise or other small environmental fluctuations can result in an output bit flip [14]. Therefore, error correction of this output will be necessary. Like the ring-oscillator PUF, the architecture of the SRAM PUF can be used to make intelligent decisions regarding error coding. The key recognition is

that the relative strengths of the two feedback loops in a SRAM cell are relatively static. A cell strongly biased toward '1' or '0' will remain strongly biased toward '1' or '0' respectively. Therefore, by using repeated measurements, one can assess the stability of a SRAM PUF output bit and selectively use the most stable bits as the PUF output. This process is used in conjunction with traditional coding techniques to mitigate the noise inherent to SRAM PUFs [1].

2.5 Attacks on Physical Unclonable Functions

Recently, it has been exhibited that Machine Learning based modeling attacks utilizing logistic regression techniques can break all current PUF developments[11]. Utilizing a polynomial measure of resources and CRPs, it is conceivable to break the security of these PUF circuits. The authors present an attack model for various order of PUF circuits to be specific strong PUFs, controlled PUFs and weak PUFs in [11],[15]. In all machine learning, it is basic to approach a lot of challenge-response pairs that structure the training samples. In the event of strong PUFs, access to CRPs is unhindered and an attacker can acquire CRPs either through listening stealthily or by direct access of the PUF circuit.

Most delay-based PUFs, for example, Arbiter PUFs, Feedforward Arbiter PUFs, Bi-stable Ring PUFs, and Lightweight Secure PUFs and in any event, Ring oscillator PUFs are viewed as strong PUFs. Controlled PUFs utilize a strong PUF and obfuscate the challenge inputs and responses produced through one way hash functions. Right now, attackers can't acquire the CRPs legitimately [15]. Anyway, it is conceivable to probe the challenge and response of the strong PUF by figuring out techniques to acquire computerized CRP data. This procedure is over the top expensive and lumbering. Given that CRPs are acquired right now, the Controlled PUF convention can be broken if the fundamental solid PUF can be effectively demonstrated. Weak PUFs regularly offer barely any CRPs that are not let out to

the outside world. These are vulnerable to figuring out and side-channel attacks. Some weak PUFs are built by incorporating solid PUFs and this usage again ends up being helpless to machine learning attacks.

2.6 Modification and Performance Improvements of PUFs

To eliminate this machine learning vulnerability and resist the PUF against modeling attacks, some modified designs have proposed by researchers. These include XOR-based PUF fusion design and hybrid PUF integration techniques. These ideas will combine the security behavior of two or more PUFs and act as a resilient model for the authentication applications as proposed in [16],[17] and [18]. In this section, we are going to concentrate on these two types of PUF modification techniques.

2.6.1 XOR-Based PUF Integration

The design of XOR PUFs is based on placing multiple linear PUFs in parallel. As illustrated in Figure 2.5, the XOR PUF uses n PUFs as components, where

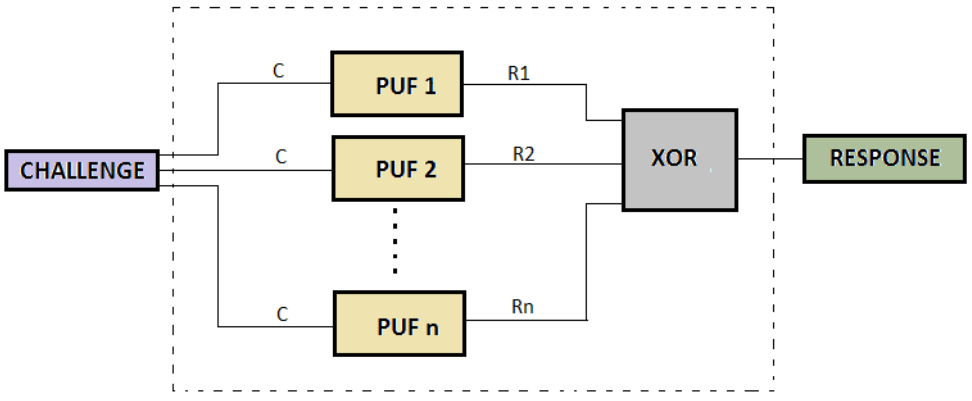

FIGURE 2.5: Block Diagram of XOR based PUF fusion model

all of the PUFs use the same challenge C as input challenge. The responses from all individual PUFs are XORed together to produce the final response for the corresponding input challenge. Thus, the response of an n-stage PUF as presented in Figure 2.5 can be expressed as:

$$Response = \oplus_{j=1...n} Rj$$

The employment of XOR operation in this design is in fact important. It increases the non-linearity of the relationship between the response r and the transformed challenges. Although adding PUFs increase the chip area and silicon implementation cost of the PUFs, it considerably increases the dimensionality of the parameter space needed to be machine-learned by attackers, leading to higher resistance against machine learning attacks.

2.6.2 Hybrid PUF Integration

Due to increased complexity of XOR-Based PUFs, it is harder for machine learning approaches to learn the patterns of these type of PUFS. Therefore, it has been suggested to combine two (or more) PUFs each with its own source of randomness in order to improve the uniqueness while maintaining or improving the FoM of the combined system.

Integration of two different PUFs is a challenging task as the designer needs to get a outline of how the control signals will be combined. In this case, each PUF has a unique process control which requires control signal to be in time to get a actual randomness.

There are many models proposed such as hybrid design of Ring oscillator and Anderson PUF [17], Arbiter and Butterfly PUF [16], and Ring Oscillator and Arbiter PUFs [18].

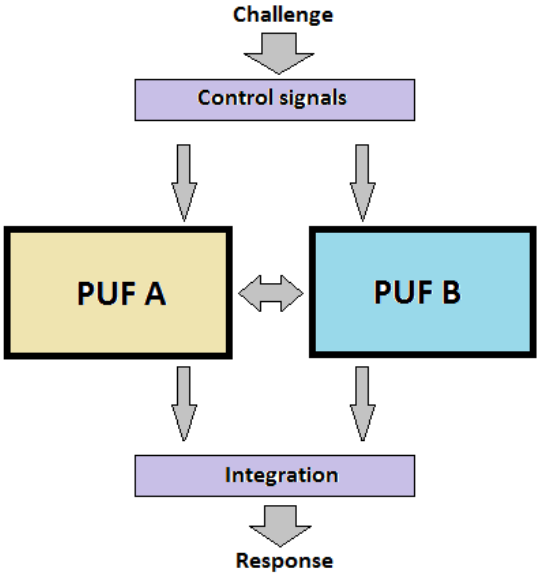

FIGURE 2.6: Block diagram of Hybrid PUF integration technique with PUF A and B

The idea of integration is to reduce the disadvantages associated with a particular model of PUF, when it is used in fusion with another strong PUF. However, the main drawback is that this concept of hybrid PUF fusion utilizes large amount of resources only for drafting the control signal integration as mentioned in the Figure 2.6.

Chapter 3

Experimental Study of XOR-Based BR-PUF Model

In this chapter, the implementation results of the XOR-based PUF is presented. Experimental results of the design, as well as its evaluation metrics, are provided.

3.1 Design Concepts

Since the Bistable Ring PUF model does not have any valid mathematical model. Therefore, it is considered a more secure and reliable strong PUF. For this reason, we conducted an experimental study of XOR-based Bi-stable Ring PUF fusion to understand the behaviour of this PUF in terms of its evaluation metrics such as Uniqueness, Reliability, and Resource utilization.

The basic principle of XOR-based Bi-stable Ring PUF is as follows: a group of PUFs each is presented with the same challenge, generates n unique output responses, These responses then will be XOR-ed together to produce 1-bit final response. The general building block of an XOR-based PUF is presented in Figure 3.1.

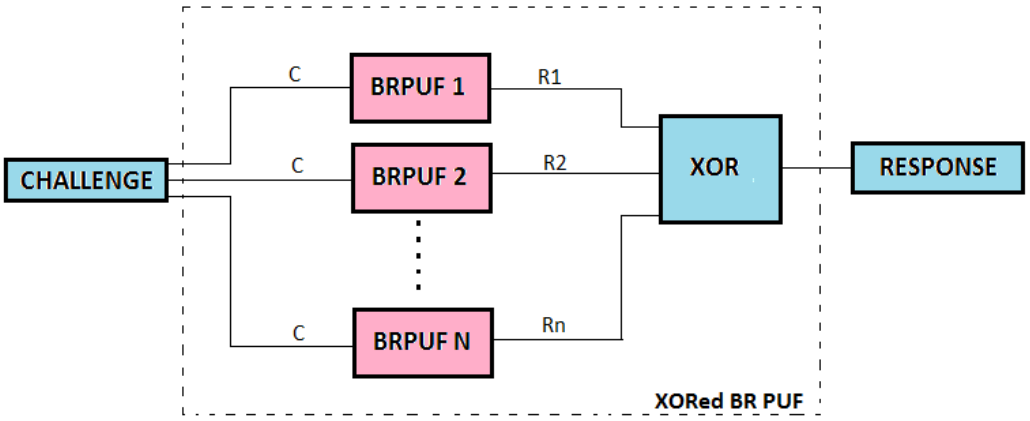

FIGURE 3.1: Block diagram of XOR-ed Bi-stable Ring PUF

The reason for XOR-based PUFs is that assessing the response of each individual PUF dependents on the last XOR response is extremely complex and tedious. This is the reason XOR PUFs are accepted to be more secure than standard Hybrid PUF models[19].

As shown in [19], the security and stability of an XOR PUF emphatically rely upon the number of PUFs utilized for the XOR activity. The training time of a Hybrid PUF model increases exponentially with the number of parallel PUFs, which is exceptionally alluring from a security angle. In any case, the level of unstable response likewise increments exponentially, which implies that the higher security of XOR PUFs includes some significant pitfalls of lower stability.

3.2 Experimental Evaluation

To analyze the working principle of XOR-based Bistable Ring PUF, we started creating verilog HDL (Hardware Description Language) for each component of desired PUF model. By using the Xilinx Vivado design suite[20], we synthesized the Verilog model of the PUF for implementation and evaluation in Xilinx Artix 7 FPGA[21].

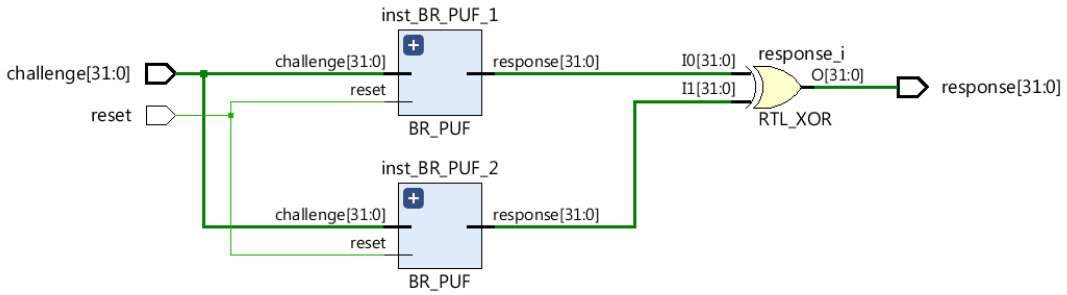

FIGURE 3.2: RTL schematics of XOR-ed Bi-Stable Ring PUF

We employed two BR-PUFs for this XOR operation, as presented in Figure 3.2. For this experiment, The Look-Up Tables (LUTs) are considered the main resource of the FPGA device. For our purpose, using only 2 BR-PUF can produce sufficient results for evaluation of XOR-based BR-PUF. We implemented PUF components such as MUX, DEMUX and 2 NOR gates in the desired location using manual place and route feature in Xilinx Vivado.

3.2.1 IP Integration

To provide the control signal for the PUF, we designed a MicroBlaze-based processing unit to control the PUF's input challenge and acquire the output response. With the help of Vivado's IP integrator tool, we utilized the Xilinx IPs such as AXI interconnection bridges, AXI GPIO, Memory units, and Xilinx UART IP.

The IP integration plays a major role for the PUF evaluation because practically, we cannot handle the real-time 32-bit input and output acquisition of Bi-stable Ring PUF manually. Hence it is easier to create a processing unit that can provide 32 challenge bits and capture all the 32-bit output response as shown in Figure 3.3.

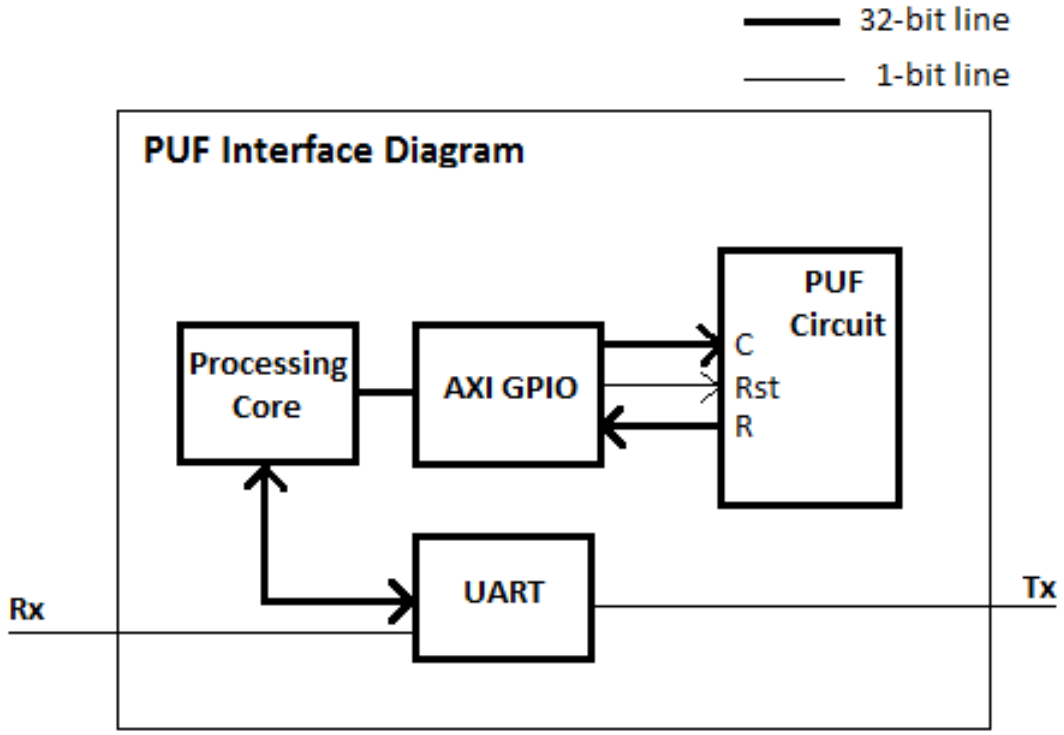

FIGURE 3.3: Interface diagram of the top module of the system designed

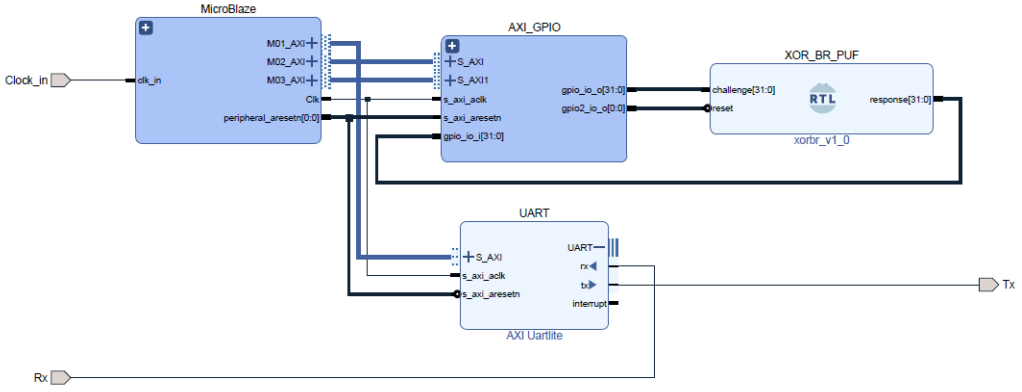

FIGURE 3.4: Block diagram of the Xilinx IP integrator

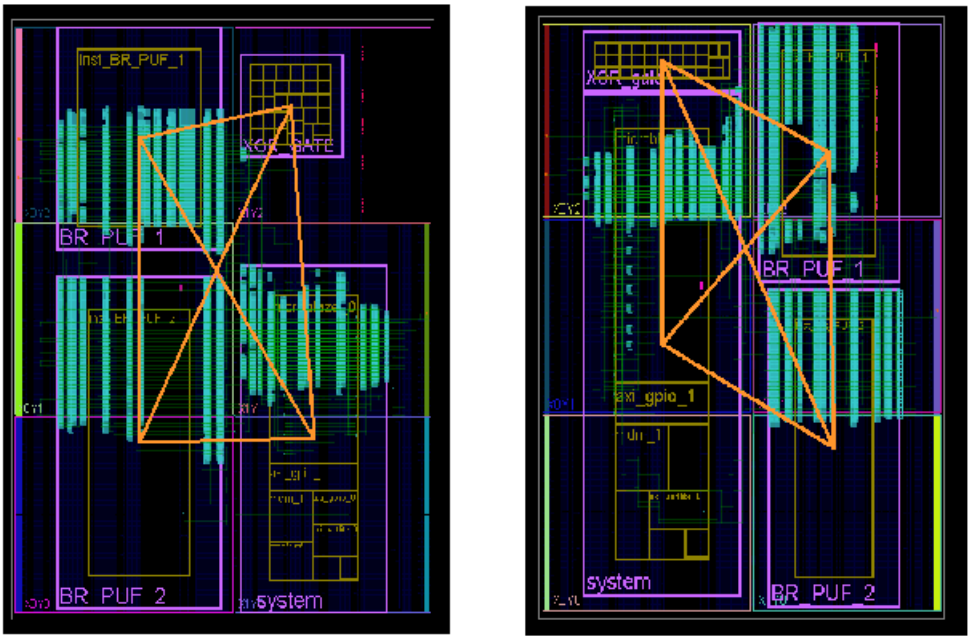

FIGURE 3.5: Floor-plan of the PUF for two locations on FPGA IC

As presented in Figure 3.4, we integrated the above-mentioned IPs to control the PUF using the AXI GPIO, which can be controlled by the Xilinx SDK[22]. This function helps us to collect the CRPs in text files through the UART serial port. We used the LabVIEW VISA[23] tool to monitor the CRPs and collect that in text files.

3.2.2 Manual Place and Route

To study the resource utilization, we needed to separate the actual PUF circuit and the processing unit we designed. Generally, the Vivado tool will synthesize the entire circuit model into a whole schematic which will bring the confusion on studying the actual resource utilized by the PUF. By using the P-block allocation, we were able to set up the dedicated place for the PUF circuit in the FPGA. This helps us to obtain the resource utilization for the PUF.

For this experiment, we analyzed the intra-chip Hamming distance of the response produced by the PUF in two different locations. This allow us to characterize the uniqueness metric. By using the TCL command, the constraint files were created to place the individual components of the PUF, such as MUX, DEMUX and NOR gates in desired LUTs.

The Figure 3.5 shows the floor plan of the PUF circuit on two different locations by creating the P-blocks.

3.3 Experimental Results

As mentioned in the previous section, the quality of PUF can be evaluated mainly by three metrics such as uniqueness, reliability and resource consumption.

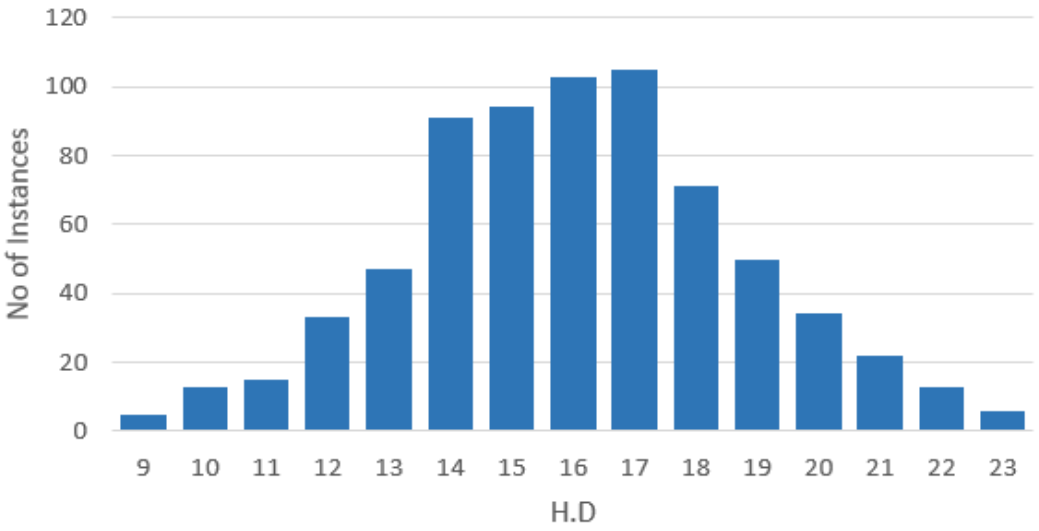

FIGURE 3.6: Graph representing the Hamming distance between the PUF responses on two locations

For demonstrating the uniqueness analysis on XOR-based Bi-stable Ring PUF, we collected Challenge-Response Pairs (CRPs) on two different locations of the FPGA to evaluate the intra-chip Hamming Distance (HD). The calculated

HD values were used in the Equation 2.1 to find the uniqueness between these two PUFs.

The graph in Figure 3.6 shows the Hamming distance of the two PUF responses on two different locations and for the same challenges. By analyzing the uniqueness as described by Equation 2.2, we can state that the uniqueness between these two PUFs is **50.1%**, which is a very good result.

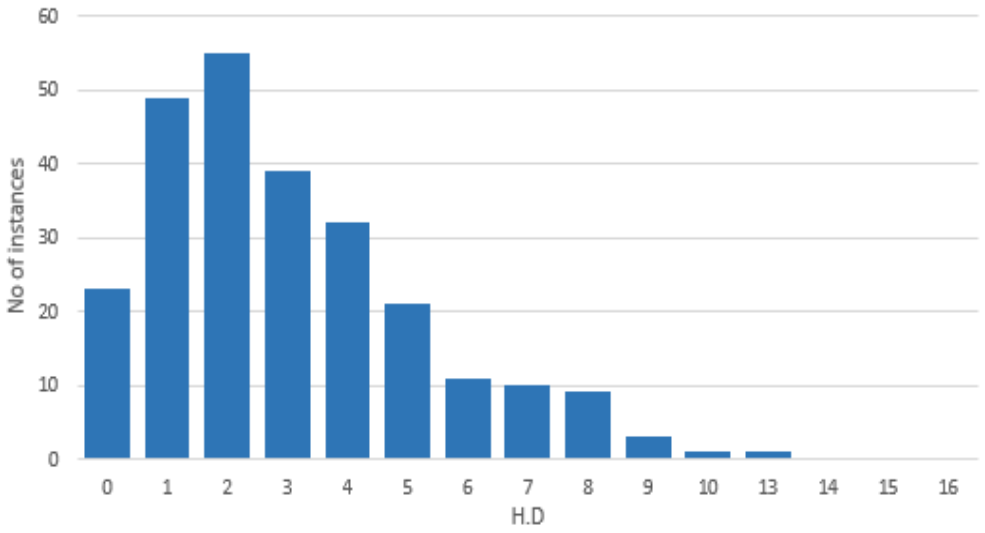

FIGURE 3.7: A graph represents Hamming Distance between the PUF responses reproduced in a PUF

To analyze the reliability metrics of the PUF model, we collected CRPs on a PUF with a time interval. This will gives an idea of how the PUF can reproduce the responses over time. We evaluated the two sets of CRPs to find the Hamming distance, and the graph (Figure 3.7) represents the hamming distance between the collected responses over the time interval.

This calculated Hamming distance values were used in Equation 2.3 to formulate the reliability of the PUF circuit. This equation gives the value of **96%**, which is also a good result. The total number of CRPs used for this calculation was 1000 CRP sets.

As a part of the evaluation, resource utilization for the PUF circuit was recorded using Xilinx Vivado synthesis tool.

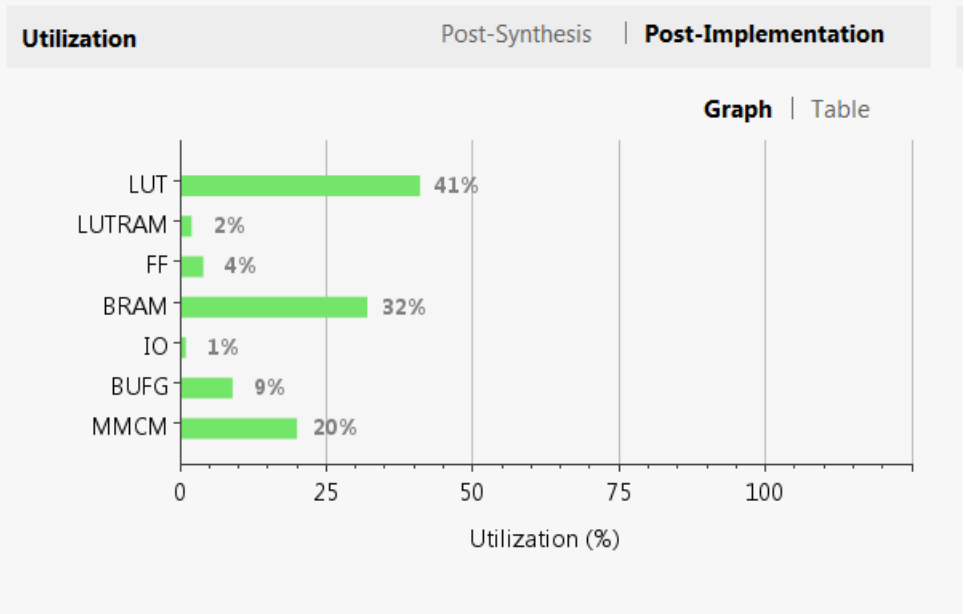

FIGURE 3.8: Chart representing resource utilization using Xilinx Vivado

As mentioned in Figure 3.8, the XOR based Bistable Ring PUF uses **44%** of total LUTs presented in the Artix 7 FPGA (this value excludes the resource consumed by MicroBlaze processing unit). This particular model uses 2 Bi-stable Ring PUF to perform XOR operation.

3.4 Limitation of XOR-ed PUF model

In this section, we elaborate on the limitation of the XOR-ed Bistable Ring PUF model in the FPGAs.

The primary motivation of the Physically Unclonable Function is to deliver the security with less resource usage as lightweight circuits [24]. As mentioned in this section, XOR-based PUF operation involves two or more PUF integration, which will give rise to the utilization of hardware resources greatly. It further

increases the cost and time of the implementation, making it less efficient. As the experimental results shows, the XOR operation will not be suitable for many resource-constrained applications. So it requires some modification to reduce the high amount of resource usage.

Chapter 4

Proposed PUF Design

In this chapter, we provide the implementation results for the proposed design in terms of uniqueness, reliability and resource utilization. First, design and implementation of proposed design was explained, followed by comparison of the results of the proposed design and XOR-based PUF model.

4.1 Design Procedure of the Proposed Hybrid PUF

In this section, we propose the novel hybrid PUF model. This Hybrid PUF combines randomness of Bistable Ring PUF and chaotic network function in order to make the design more lightweight while keeping the performance metrics competitive.

This structure is based on the basic Bi-stable Ring PUF, which is combined with the one-dimensional chaotic mapping function that produces shuffled response as shown in Figure 4.1.

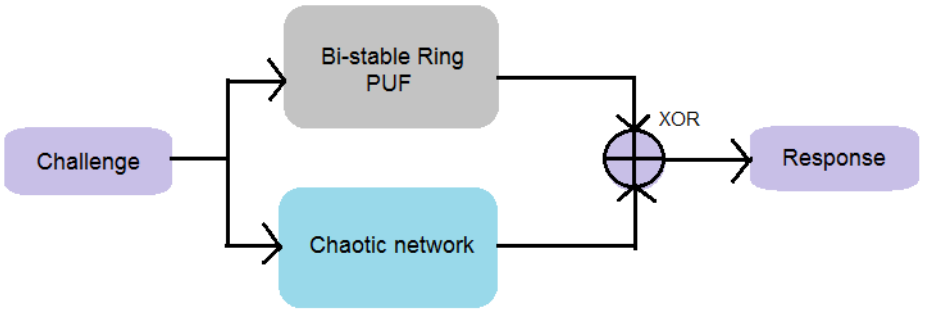

FIGURE 4.1: Block diagram of the proposed design

4.2 One-Dimensional Chaotic Mapping

The Chaotic mapping can be defined as a polynomial mapping of degree two, often cited as an archetypal example of how complex, chaotic behaviour can arise from very simple non-linear dynamical equations as follows.

$$x_{n+1} = r \cdot x_n(1 - x_n) \tag{4.1}$$

where x_n is a number between zero and one that represents the ratio of the existing population to the maximum possible population and the values of interest for the parameter r are those in the interval [0,4], the value of r beyond 4 will give rise to negative population sizes[25] which is not useful for our purpose.

When r is greater than or equal to 3.57, then this function will give rise to chaotic behaviour in terms of stretching and folding operation on the interval (0,1)[26]. These stretching and folding values can be used as a chaotic factor for the design to shuffle the response of the BR PUF.

4.2.1 Simulation of Chaotic Mapping

To study the behaviour of the chaotic mapping function, we created a systematic model of this network using the LabVIEW software. This model was then

tested for some random initial values as 0.1, 0.125, 0.15 and 0.1525 and simulation results are shown in Figures 4.2, 4.3, 4.4 and 4.5.

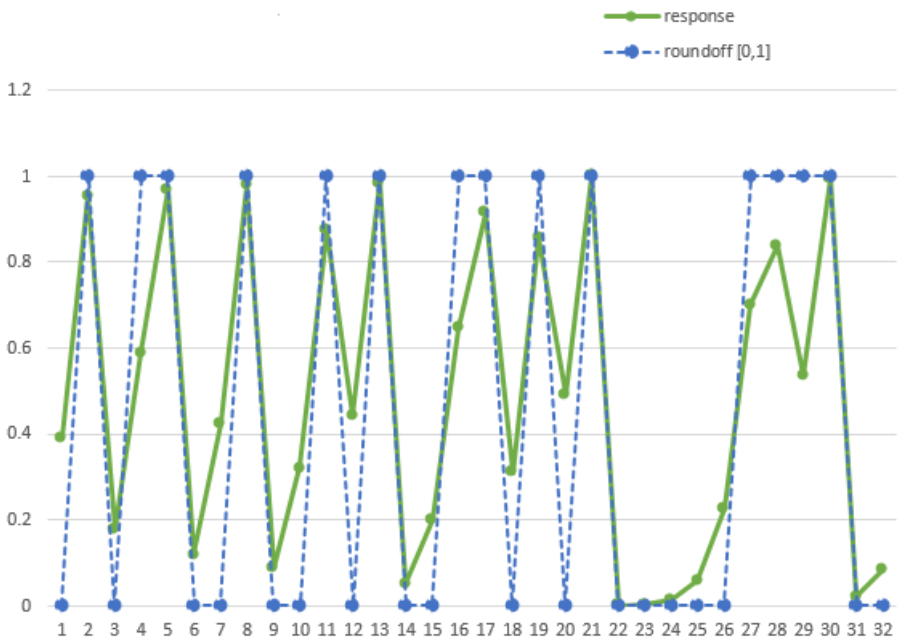

FIGURE 4.2: Chaotic response for initial value 0.1

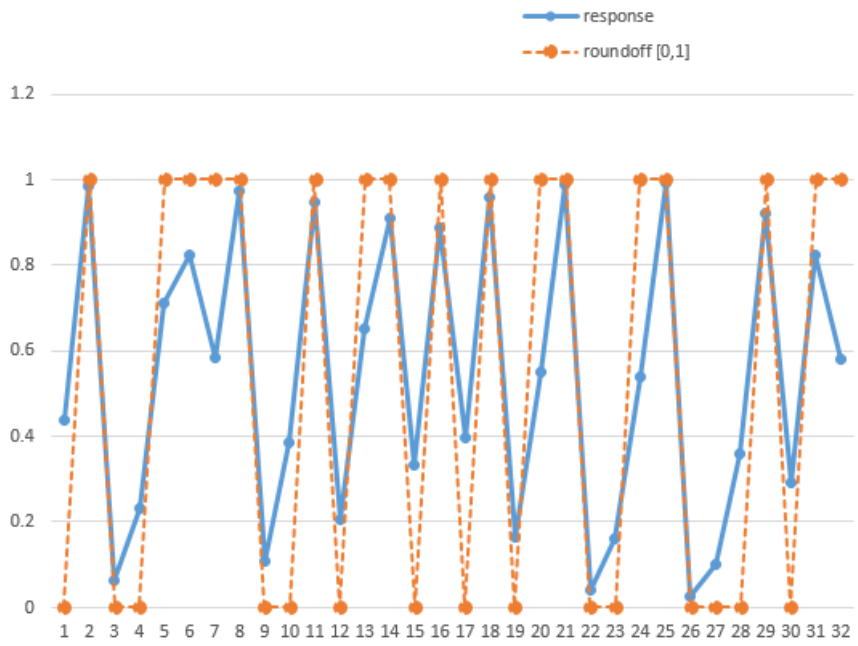

FIGURE 4.3: Chaotic response for initial value 0.125

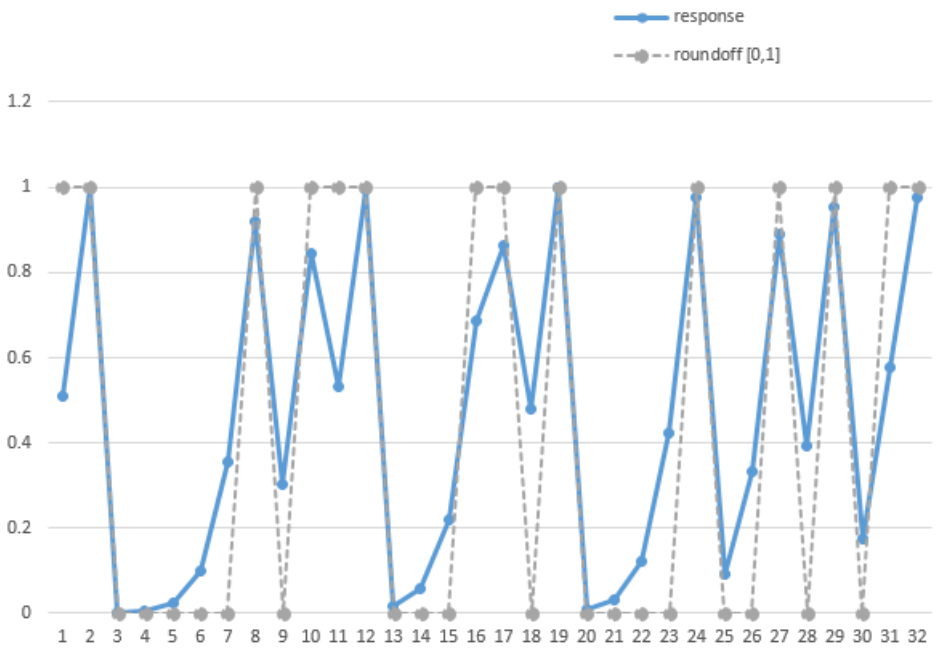

FIGURE 4.4: Chaotic response for initial value 0.15

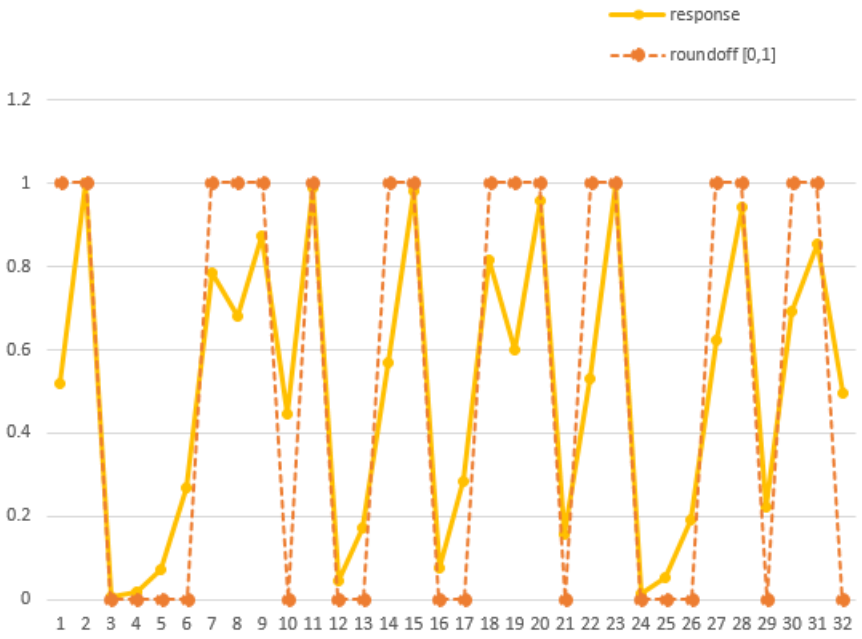

FIGURE 4.5: Chaotic response for initial value 0.1525

These graphs explain how the chaotic network behaves in a non-linear way for given initial values. These output responses from the network can be used for our purpose.

Hence we used this chaotic function for our method to shuffle the response further to enhance the security of the Bi-stable Ring PUF.

4.3 Development of the Chaotic Network

We synthesized the popular chaotic network called "One-dimensional logistic mapping function"[27] into the digital circuit using the Xilinx High-Level Synthesis[28] tool. This function includes complex floating-point arithmetic which was synthesis by the Xilinx HLS tools.

4.4 Hardware Implementation of the Proposed Hybrid PUF

The basic function blocks of the proposed Hybrid PUF are illustrated in Figure 4.6. This figure explains the integration part of the BR-PUF and chaotic response generator. Here this chaotic part was used to eliminate the bit repetition, which occurs in the Bi-stable Ring PUF, and it gives more non-linearity to the model.

We used the Xilinx HLS tool to design the hardware to implement the one-dimensional chaotic mapping function. Initially, we modelled this design using the C++ program and then synthesized the code to Register Transfer Level (RTL) design using the HLS tool.

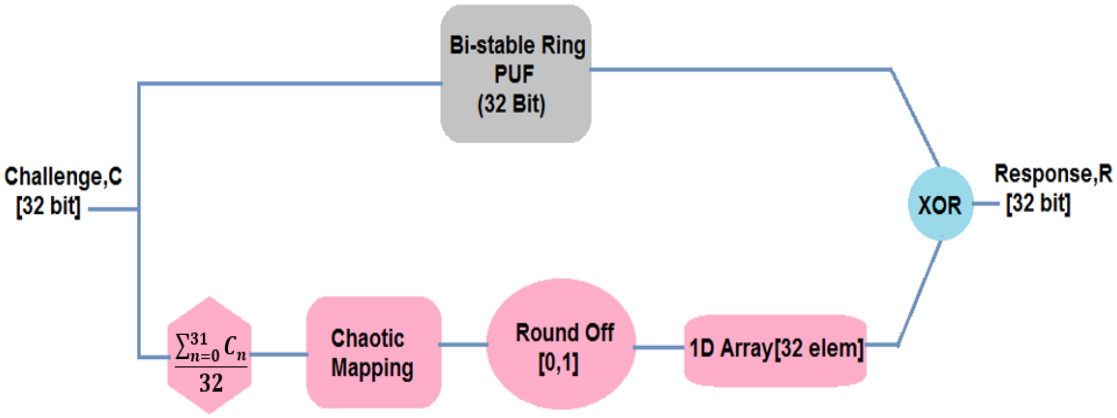

FIGURE 4.6: Functional block diagram of Proposed Hybrid PUF

We included many optimizing techniques such as HLS Pipeline and Unroll to get an optimized design. After this process, we then created a custom IP block, which includes all the functional blocks of chaotic mapping function. As shown in the Figure 4.7, the RTL interface ports of custom IP are *Challenge*, *Response*, *Response Valid* and Control Signals such as *ap_start*, *ap_clock* and *ap_done*.

4.4.1 Chaotic Response IP

The IP block for chaotic mapping function was created as shown in the Figure 4.7. This IP can be used for creating 32-bit chaotic response bit for our proposed design.

FIGURE 4.7: Custom IP generated using the Xilinx HLS

4.4.2 Computing Response from the Chaotic IP

By analyzing the Equation 4.1, we designed a list of computational steps to compute the response from the IP created using Xilinx HLS as shown in the Algorithm 1 below.

Algorithm 1: Computation of output response through chaotic mapping

Result: Response[32] = [R_0,R_1,....,R_{31}];

Input: Challenge[32] = [C_0,C_1,...,C_{31}];

Challengesum = 0 ; // Initialization

$r = 3.57$;

X_0= 0;

X_n=[X_0,X_1,....,X_{31}];

for $i \leftarrow 0$ **to** 31 **do**

  challengesum = $\sum_{i=0}^{31}$ challenge[i]; ; // Loop 1

end

$X_0 = \frac{challengesum}{32}$

for $n \leftarrow 0$ **to** 31*;* // Loop 2

do

  $X_{n+1} = r.X_n(1 - X_n)$; ; // Chaotic mapping equation

end

for $n \leftarrow 0$ **to** 31*;* // Loop 3

do

  if $X_n < 0.5$ **then**

    $R_n = 0$

  else

    $R_n = 1$

  end

end

Return Response[32];

Here, the RTL ports of the IP are Challenge[32] and Response[32]. The HLS C++ code for performing this operation is given in Appendix 2.

4.4.3 Integration of IPs

As we mentioned in the previous chapter, the acquisition of Challenge and Response Pairs (CRPs) can be carried easily by developing a processing unit. We developed a MicroBlaze processing unit similar to the experimental study to ease that process. The Figure 4.8 shows the schematic of the PUF and MicroBlaze processor integration.

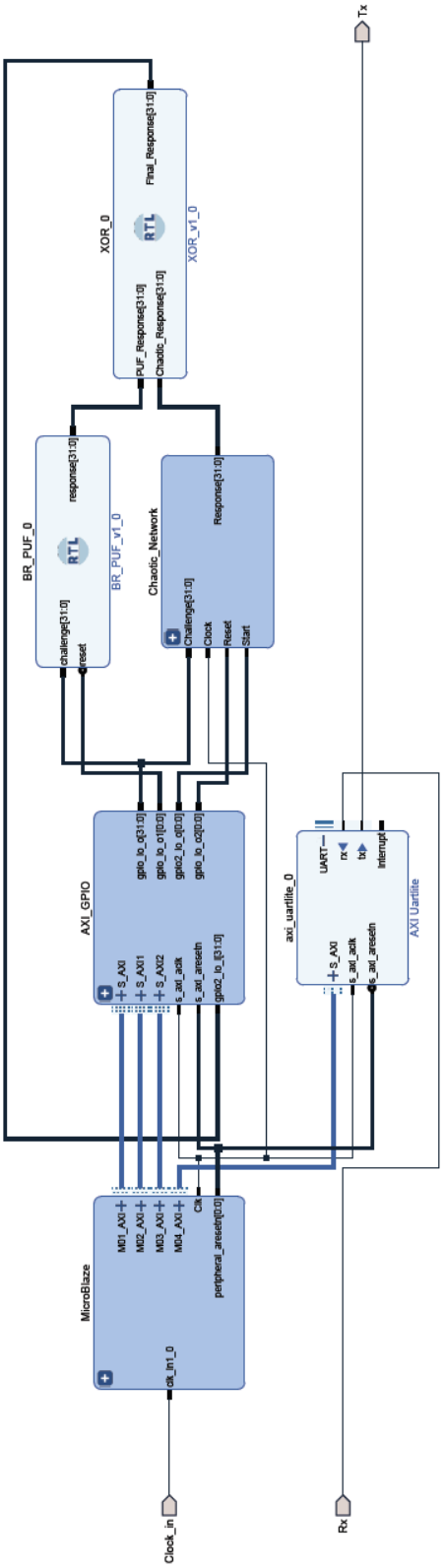

FIGURE 4.8: IP Integration of Hybrid PUF with MicroBlaze processor

4.4.4 Implementation of proposed PUF on FPGA

By using the Xilinx Vivado tool, we synthesized the whole model (shown in the Figure 4.8) and implemented in the Xilinx Artix 7 FPGA. To measure the quality of the PUF, we used two different locations of the chip to analyze the intra-chip Hamming distance. We used P-block allocation feature to fix placement as mentioned in Chapter 3. We differentiated these two locations as Location A and Location B, as shown in Figure 4.9. The orange line shown in the Figure 4.9 represents the communication signal between the component blocks.

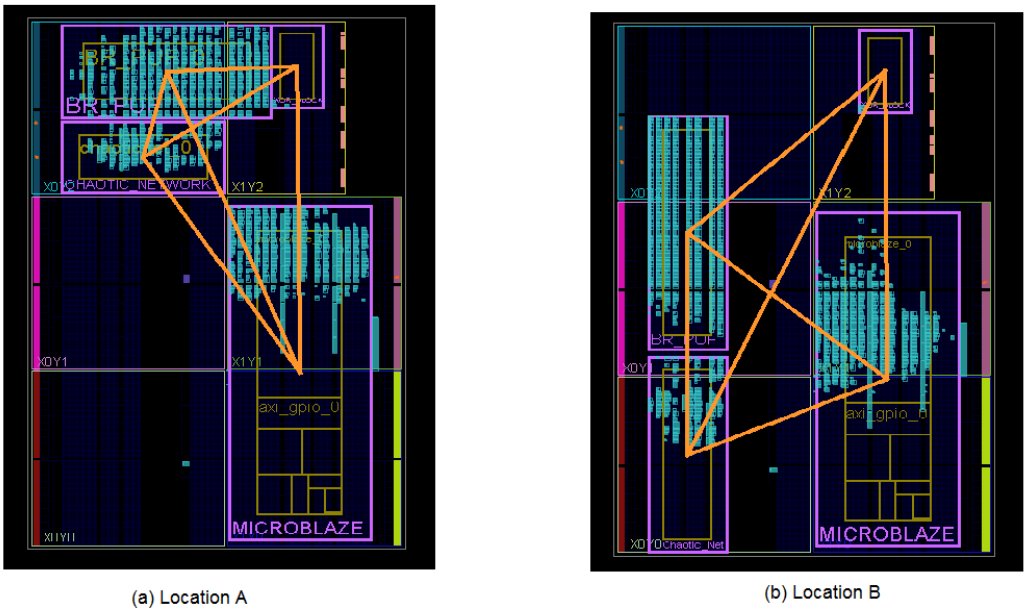

FIGURE 4.9: Floor plan of implemented PUF on two different locations

Here the figure represents four P-blocks which include BR-PUF, Chaotic Network, XOR gates, and MicroBlaze System.

4.5 Implementation Results

In this section, we analysed the quality of proposed PUF in terms of Uniqueness, Reliability and Resource utilization.

4.5.1 Uniqueness Analysis

For demonstrating the uniqueness analysis for the proposed hybrid PUF, we collected 1000 Challenge-Response Pairs (CRPs) on two different locations of the FPGA to evaluate the intra-chip Hamming Distance (HD).

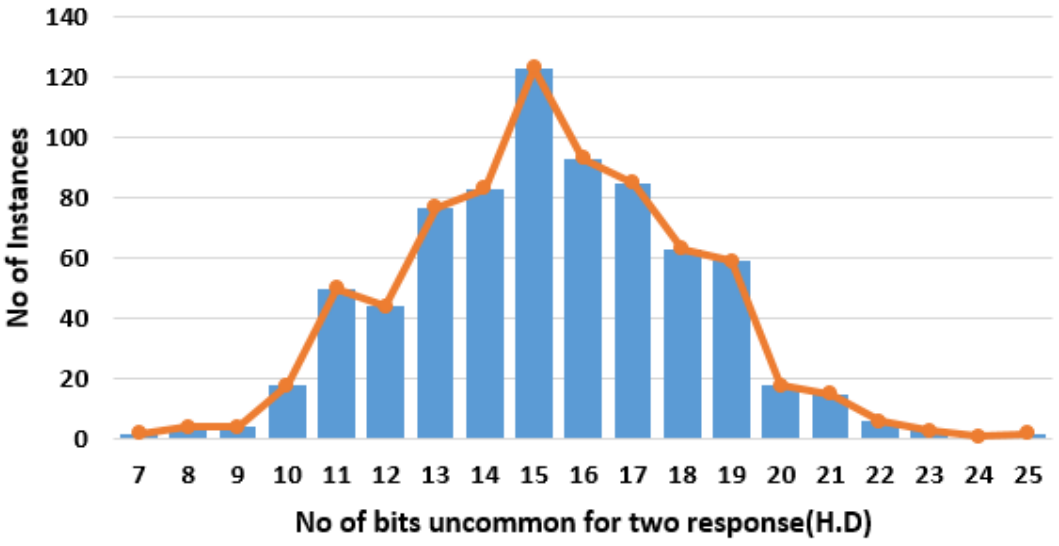

FIGURE 4.10: Hamming distance between two responses for two locations

The Figure 4.10 shows the Hamming distance of the two PUF responses on two different locations for the same challenges. These calculated HD values were used in the Equation 2.1 to find the uniqueness between these two PUFs. The proposed hybrid model produces the uniqueness value of **48%**, which can be considered as good value.

4.5.2 Reliability Analysis

As mentioned in the Chapter 3, to analyze the reliability metrics of the PUF model, we collected 1000 CRPs on a PUF with a random time interval.

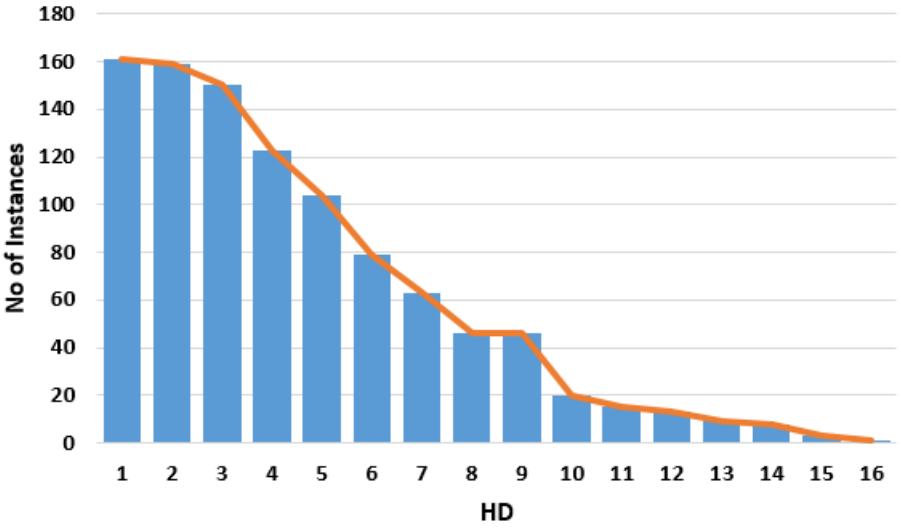

FIGURE 4.11: Hamming Distance between two responses reproduced at different time

We evaluated the two sets of CRPs to find the hamming distance, and Figure 4.11 shows the hamming distance between the collected responses over a time interval. The proposed hybrid model produces reliability of **91%**, which is a good result as explained in Chapter II.

4.5.3 Resource Utilization

This section provides resource consumption of the proposed hybrid PUF design. Here we used LUT utilization as the main parameter to determine the resource utilization of the PUF circuit.

As shown in Figure 4.12, our proposed model uses **28%** of total LUTs available in the Artix 7 FPGA (this value excludes the resource consumed by MicroBlaze

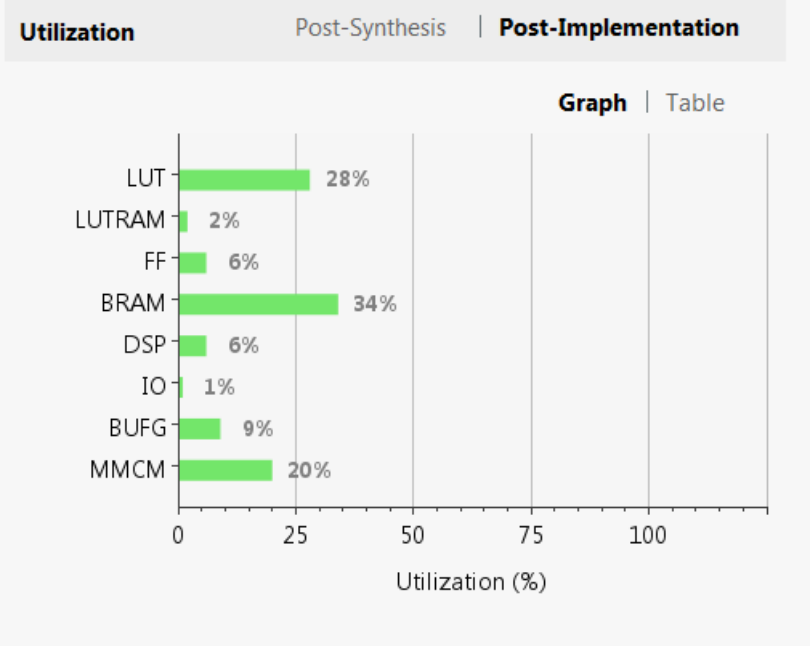

FIGURE 4.12: Resource utilization of proposed PUF in Xilinx Artix 7 FPGA

processing unit). By analysing the obtained result, there is a significant reduction in resource usage which is 16% in LUT reduction.

4.6 Result and Discussion

In this section, we present the overall comparison of the existing XOR based PUF model with our proposed hybrid PUF model.

As we mentioned in our experimental study, XOR-ed BR PUF requires 44% of LUT in Artix 7 FPGA and it has uniqueness of **50%** and reliability of **96%**. To reduce resource consumption without compromising the strength of the PUF model, we proposed a Hybrid PUF model that will consume 28% of FPGA LUT and produce similar uniqueness value and reliability.

The Table 1 shows the comparison of Bistable Ring PUF, XOR-ed Bistable ring PUF and our hybrid model in terms of resource usage, uniqueness, reliability and challenge length.

Content	BRPUF	XOR-ed BRPUF	Our proposed PUF
Resources(LUT)	24%	44%	28%
on-chip power(W)	0.193	0.193	0.193
Uniqueness	34%	50%	48%
Reliability	98%	96%	91%
Challenge length	32	32	32

TABLE 1: **BR** vs **XOR-based BR PUF** vs **Our proposed model**

From the Table 1, our proposed PUF has less design complexity in terms of resource utilization, while maintaining the uniqueness and reliability competitive. So our hybrid model can be considered as a good replacement for XOR-ed based PUF integration model.

Chapter 5

Conclusion

In this chapter, we summarize the main contributions in this thesis and propose future work in related area.

5.1 Summary of Contribution

In this thesis, we have implemented following PUF structures, Bi-stable Ring PUF, XOR-ed Bi-stable Ring PUF and a Hybrid model PUF, on Xilinx Artix 7 FPGA and have investigated their performance in terms of reliability, uniqueness and resource utilization. We have proposed a Hybrid PUF scheme in which a PUF and Chaotic network are combined to improve the randomness of the response. In the traditional approach such as XOR PUFs, large resource utilization is needed since it combines two or more PUF models to increase the non-linearity. This significantly increases the area and power consumption of design.

The proposed hybrid model can be considered as a good replacement of XOR-ed based PUF fusion and make the system much more lightweight. Extensive experimental analyses were done by collecting 1000 CRPs for every instance for all the three PUFs. Implementation results show that the proposed Hybrid

PUF scheme provides good values in uniqueness and reliability while at the same time,there is significant reduction in resource utilization.

The results presented in this work are all obtained by implementing the schemes on the Xilinx FPGA board with many instances. Additionally, as discussed previously, the proposed Hybrid PUF is a general method and its characteristics can be further investigated using other strong PUF models.

5.2 Future Work

We would recommend that this proposed method can be used for other strong PUF models to enhance the security. It can be considered as a good replacement for the two or more PUF integration methods used. This research can be extended for the high generation FPGAs such as Intel Arria, Stratix and Xilinx Zynq. We can explore further optimization of the chaotic network for application such as authentication and secure key generation systems.

References

[1] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *2007 44th ACM/IEEE Design Automation Conference*, pp. 9–14, IEEE, 2007.

[2] M. Tehranipoor and C. Wang, *Introduction to hardware security and trust*. Springer Science & Business Media, 2011.

[3] J. A. Roy, F. Koushanfar, and I. L. Markov, "Epic: Ending piracy of integrated circuits," in *Proceedings of the conference on Design, automation and test in Europe*, pp. 1069–1074, 2008.

[4] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[5] R. Pappu, B. Recht, J. Taylor, and N. Gershenfeld, "Physical one-way functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002.

[6] C. Herder, M.-D. Yu, F. Koushanfar, and S. Devadas, "Physical unclonable functions and applications: A tutorial," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1126–1141, 2014.

[7] D. Lim, J. W. Lee, B. Gassend, G. E. Suh, M. Van Dijk, and S. Devadas, "Extracting secret keys from integrated circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 10, pp. 1200–1205, 2005.

[8] M. Soybali, B. Ors, and G. Saldamli, "Implementation of a puf circuit on a fpga," in *2011 4th IFIP International Conference on New Technologies, Mobility and Security*, pp. 1–5, IEEE, 2011.

[9] Y. Wen and Y. Lao, "Enhancing puf reliability by machine learning," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, 2017.

[10] S. Tajik, E. Dietz, S. Frohmann, J.-P. Seifert, D. Nedospasov, C. Helfmeier, C. Boit, and H. Dittrich, "Physical characterization of arbiter pufs," in *International Workshop on Cryptographic Hardware and Embedded Systems*, pp. 493–509, Springer, 2014.

[11] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proceedings of the 17th ACM conference on Computer and communications security*, pp. 237–249, 2010.

[12] X. Xin, J.-P. Kaps, and K. Gaj, "A configurable ring-oscillator-based puf for xilinx fpgas," in *2011 14th Euromicro Conference on Digital System Design*, pp. 651–657, IEEE, 2011.

[13] Q. Chen, G. Csaba, P. Lugli, U. Schlichtmann, and U. Rührmair, "The bistable ring puf: A new architecture for strong physical unclonable functions," in *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*, pp. 134–141, IEEE, 2011.

[14] A. Garg and T. T. Kim, "Design of sram puf with improved uniformity and reliability utilizing device aging effect," in *2014 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1941–1944, IEEE, 2014.

[15] A. Mahmoud, U. Rührmair, M. Majzoobi, and F. Koushanfar, "Combined modeling and side channel attacks on strong pufs.," *IACR Cryptology ePrint Archive*, vol. 2013, p. 632, 2013.

[16] J. Kokila and N. Ramasubramanian, "Enhanced authentication using hybrid puf with fsm for protecting ips of soc fpgas," *Journal of Electronic Testing*, vol. 35, no. 4, pp. 543–558, 2019.

[17] S. Khoshroo, "Design and evaluation of fpga-based hybrid physically unclonable functions," in *Electronic Thesis and Dissertation Repository - https://ir.lib.uwo.ca/etd/1281*, Western University, 2013.

[18] S. Sankaran, S. Shivshankar, and K. Nimmy, "Lhpuf: Lightweight hybrid puf for enhanced security in internet of things," in *2018 IEEE International Symposium on Smart Electronic Systems (iSES)(Formerly iNiS)*, pp. 275–278, IEEE, 2018.

[19] C. Zhou, K. K. Parhi, and C. H. Kim, "Secure and reliable xor arbiter puf design: An experimental study based on 1 trillion challenge response pair measurements," in *Proceedings of the 54th Annual Design Automation Conference 2017*, pp. 1–6, 2017.

[20] "Xilinx vivado design suite datasheet, https://www.xilinx.com/products/design-tools/vivado.html."

[21] "Xilinx artix 7 fpga datasheet, https://www.xilinx.com/products/silicon-devices/fpga/artix-7.html."

[22] "Xilinx software development kit(sdk) datasheet, https://www.xilinx.com-/products/design-tools/embeddedsoftware-/sdk.html."

[23] "National instrument's labview datasheet, https://www.ni.com/en-ca/lab-view.html."

[24] U. Chatterjee, R. S. Chakraborty, and D. Mukhopadhyay, "A puf-based secure communication protocol for iot," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 16, no. 3, pp. 1–25, 2017.

[25] P. Dabal and R. Pelka, "Fpga implementation of chaotic pseudo-random bit generators," in *Proceedings of the 19th International Conference Mixed Design of Integrated Circuits and Systems-MIXDES 2012*, pp. 260–264, IEEE, 2012.

[26] N. Pareek, V. Patidar, and K. Sud, "Cryptography using multiple one-dimensional chaotic maps," *Communications in Nonlinear Science and Numerical Simulation*, vol. 10, no. 7, pp. 715–723, 2005.

[27] H. Thunberg, "Periodicity versus chaos in one-dimensional dynamics," *SIAM review*, vol. 43, no. 1, pp. 3–30, 2001.

[28] "Xilinx high level synthesis datasheet, https://www.xilinx.com/products/design-tools/vivado/integration/esl-design.html."

Appendices

Appendix A

Verilog code for BR-PUF

A.1 Multiplexor

```
\‘timescale 1ns / 1ps
(* keep_hierarchy = "true"*)module mux(
            input ia,
            input ib,
            input isel,
            output oout
            );
wire oout;
assign oout = (isel) ? ia : ib;
endmodule
```

A.2 Demultiplexor

```
‘timescale 1ns / 1ps
(* keep_hierarchy = "true"*)module demux(
```

```
            input i,
            input isel,
            output outa,
            output outb);
wire outa;
wire outb;
assign outa = i & isel;
assign outb = i & ~(isel);
endmodule
```

A.3 Bistable Ring

```
‘timescale 1ns / 1ps
module ring(
            input [31:0]challenge,
            input reset,
            output response
            );
(*keep = "true"*) wire [96 : 0]net;
assign net[0] = net[96];
generate
genvar i;
for (i = 1; i <= 32; i = i + 1)
begin
   demux inst_demux(
                     .i(net[i * 3 - 3]),
                     .isel(challenge[i - 1]),
                     .outa(net[i * 3 - 2]),
```

```
                    .outb(net[i * 3 - 1])
                    );
    mux inst_mux(
                    .ia(~(net[i * 3 - 2]|reset)),
                    .ib(~(net[i * 3 - 1]|reset)),
                    .isel(challenge[i - 1]),
                    .oout(net[i * 3])
                    );
end
endgenerate
assign response = net[48];
endmodule
```

A.4 BR-PUF

```
‘timescale 1ns / 1ps
module BR_PUF(
              input [31:0] challenge,
              input reset,
              output [31:0] response
              );

(*keep = "true"*)wire [31:0] net;
generate
genvar i;
for (i = 1; i <= 32; i = i + 1)
begin
ring inst_ring(.challenge(challenge),
```

```
                .reset(reset),
                .response(net[i-1])
               );
end
endgenerate
assign response = net;
endmodule
```

Appendix B

C++ code for Xilinx HLS

```
void chaoticnet(bool challenge[32],bool response[32])
{

#pragma HLS ARRAY_PARTITION variable=response complete dim=0
#pragma HLS ARRAY_PARTITION variable=challenge complete dim=0

int challengesum=0;
float xn[32];
float x0;

for (int h = 0; h < 32; h++)
{
challengesum=challengesum+challenge[h];
}

x0=(float)challengesum/32;
xn[0]=x0;
```

```
for (int i=0;i<32;i++)
{
xn[i+1]=xn[i]*4*(1-(xn[i]));
}

for (int j = 0; j < 32; j++)
{
#pragma HLS UNROLL
if (xn[j]<0.5)
{
response[j] = 0;
}
else
{
response[j] = 1;
}
}
}
```

Vita Auctoris

NAME	Madhan Thirumoorthi
PLACE OF BIRTH	Erode, Tamilnadu, India
YEAR OF BIRTH	1997
EDUCATION	Anna University, India, 2014 - 2018
	Bachelor of Eng.(Electrical and Electronics)