10-4-2023

A New Resource Efficient Multi PUF Design

Sai Preetham Bonagiri
University of Windsor

A NEW RESOURCE EFFICIENT MULTI PUF DESIGN

By

SAI PREETHAM BONAGIRI

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

2023

A NEW RESOURCE EFFICIENT MULTI PUF DESIGN

by

SAI PREETHAM BONAGIRI

APPROVED BY:

X. Yuan
School of Computer Science

A. H. Sakr
Department of Electrical and Computer Engineering

M. Khalid, Advisor
Department of Electrical and Computer Engineering

September 13, 2023

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 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 hardware-based security for electronic devices and systems which can eliminate the use of the conventional cryptographic system which uses large area and storage. The current imminent concern is the vulnerability of PUFs to popular machine learning attacks such as Covariance Matrix Adaptability and Evaluation Strategy (CMA-ES) attacks and Linear Regression (LR) attacks. To address this issue, many PUF models have been proposed to minimize the vulnerability of PUFs to machine learning attacks. Multi PUFs (MPUFs) are one of the popular models used in this domain and have proven to be successful in providing better security. These models demand large resources and possess comparatively inferior PUF metric values. In this thesis, we propose two new MPUF designs, which also incorporate the XOR technique, to provide improved PUF metric values and also decreased resource usage.The proposed MPUF designs were implemented in a Xilinx Artix 7 FPGA. Experimental evaluation results demonstrate that, compared to existing MPUFs, the proposed MPUF designs provide better uniqueness, reliability, and reduced resource consumption.

DEDICATION

I would like to dedicate this thesis to my mother for her incredible love and support. I really believe she is the one who worked selflessly and supported me unconditionally in every moment of my life.

Furthermore, I dedicate it to my father for believing in me and giving me wings to fly. To my grandfather, who has been a real inspiration to me since childhood and for always cheering me and supporting me in my hard time.

ACKNOWLEDGEMENTS

I would like to sincerely express my most profound gratitude towards my supervisor Dr. Khalid, whose input helped me immensely. With his input and support, I was able to look at my research with a different perspective and a more critical eye

Secondly, I would like to express my gratitude to my thesis committee members Dr. Ahmed Hamdi Sakr and Dr. Xiaobu Yuan for their beneficial advice and suggestions for my thesis

I would like to thank Madhan Kumar Thirumoorthi for giving me valuable advice at crucial times in my research

I would also like to thank my friends Vivek Ladhe and Bharath Chandra for always encouraging and supporting me

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

PUF	Physical Unclonable Function
CRP	Challenge Response Pair
CMAES	Covariance Matrix Adaption and Evolution Strategy
LR	Linear Regression
FPGA	Field Programmable Gate Array
ASIC	Application Specific Integrated Circuit
IoT	Internet of Things
RNG	Random Number Generator
AI	Artificial Intelligence
APUF	Arbiter PUF
ROPUF	Ring Oscillator PUF
SRAM	Static Random Access Memory
ML	Machine Learning
MPUF	Multi-PUF
FFAPUF	FlipFlop based Arbiter PUF
FOXFFAPUF	Feedback-oriented XORed Flipflop-based arbiter PUF
AXI	Advanced Extensible Interface
GPIO	General Purpose Input/Output
UART	Universal Asynchronous Receiver/Transmitter
IP	Intellectual Property
IC	Integrated Circuit

RTL	Register Transfer Logic
LUT	Look Up table

CHAPTER 1

Introduction

Physical Unclonable Functions, or PUFs, are physical entities embedded in integrated circuits and used for providing hardware-based security for real-world digital designs[7]. A wide range of applications are enabled by PUFs such as random number generation, device authentication, memory protection, secret key generation, etc[7].

1.1 Motivation

In the current rapidly advancing technological era, the IoT (Internet of Things) has made significant growth. A wide range of devices are interconnected through the internet in our current IoT world, because of which several challenges and security concerns have emerged. The IoT encompasses a vast array of devices, ranging from sensors and actuators to smart appliances and industrial machines. Many of these devices have limited computational resources, making it challenging to implement complex security measures[13]. They often have limited processing power, memory, and energy resources. This can hinder the implementation of strong software-based security measures, making hardware-based security a more suitable option.

The sheer number of IoT devices increases the attack surface for cybercriminals. Vulnerabilities in one device could potentially compromise an entire network or system, making it crucial to have reliable security measures in place. IoT devices can be connected to networks with varying levels of security. Weaknesses in network security could expose devices to unauthorized access and data breaches, emphasizing the importance of securing devices themselves. IoT devices collect and transmit sen-

sitive data, often related to user behavior, location, health, or personal preferences. Ensuring the privacy and integrity of this data is essential to maintain user trust. All of these devices are frequently managed remotely, which introduces potential vulnerabilities if attackers can exploit remote management interfaces or gain unauthorized access[7].

Insecure firmware can open up devices to various attacks, including those that take advantage of unintended vulnerabilities. Hardware-based security can help ensure the integrity of firmware updates and prevent unauthorized modifications. Also, many IoT devices are deployed in environments where physical access can't be fully controlled. This exposes devices to potential tampering, which hardware-based security measures can help mitigate[13]. Additionally, IoT devices are susceptible to counterfeiting and cloning, where attackers produce unauthorized copies of devices to gain access to networks or data. Hardware-based security measures like unique identifiers can help verify the authenticity of devices.

Given these challenges, hardware-based security becomes crucial in the IoT era for several reasons:

- Device Authentication: Hardware-based security can provide unique identifiers and cryptographic keys for device authentication, ensuring that only legitimate devices can access a network[7].

- Data Protection: Hardware-based security measures can encrypt and protect sensitive data stored on or transmitted by IoT devices, safeguarding user privacy and preventing data breaches[21].

- Tamper Resistance: Hardware-based security can protect devices against physical tampering and attacks aimed at extracting sensitive information[22].

- Long-Term Security: Hardware-based security solutions can provide long-lasting

protection, even in situations where software vulnerabilities are discovered over time[21].

In summary, the IoT era introduces a wide range of security challenges due to the diversity of devices, resource constraints, network vulnerabilities, and data privacy concerns. Hardware-based security offers a way to address these challenges and build a more secure foundation for the interconnected devices that make up the IoT ecosystem.

1.2 Introduction to Hardware Security

Hardware security involves implementing security measures at the physical level of electronic devices to protect against various threats, including unauthorized access, tampering, counterfeiting, and data breaches. It focuses on utilizing the unique properties of hardware components to provide stronger security than can be achieved through software alone. Hardware security solutions are important in contexts where software-based approaches might be susceptible to attacks or vulnerabilities. PUFs are a subset of hardware security mechanisms that exploit inherent variations in the physical properties of electronic components, such as transistors, to create unique and unpredictable responses[8].

The motivation for hardware security arises from the limitations and vulnerabilities of software-based security solutions. Software can be vulnerable to attacks, malware, and vulnerabilities that can compromise the integrity, confidentiality, and availability of systems and data. Hardware security provides a complementary layer of protection by leveraging the following key principles.

1.3 General Functionality of PUF

PUFs take advantage of innate manufacturing variations, which in turn produces process variations in IC's, such that each PUF fabricated from the same design entities gives a different response or otherwise called fingerprints[7]

A chip (FPGA or ASIC) in its core, is made of innumerable transistors (which are combined to form logic gates, switches, CLBs etc.). The existence of very minute differences in doping concentration or other manufacturing variations between these transistors is the driving principle of a PUF architecture

Consider a 32-bit PUF circuit. This circuitry takes a 32-bit challenge and outputs a 32-bit response. The PUF circuitry contains 32 identical lines, each resulting in a 1-bit response with the 32-bit input[7]. A PUF line takes a 32-bit challenge as an input, has a fixed number of components, and adheres to specific placement and routing constraints, giving a 1-bit output as a response. A PUF line is replicated 32 times to produce a 32-bit response, which changes according to the input challenge due to innate manufacturing variations

1.4 Classification

PUF circuitry is easy to implement, but difficult to replicate. It is a function that accepts a challenge 'C' and provides a response 'R', thus giving us numerous Challenge-Response Pairs (CRPs). This function/circuitry is embedded in each device during the fabrication process and a subset of CRPs are recorded after the fabrication. These CRPs can be used in various applications according to the PUF adaptability and compatibility.

PUFs are characterized into two categories based on the number of CRPs available: Strong PUFs and Weak PUFs

1.4.1 Strong PUF

Strong PUFs are those with a large set of CRPs. They possess a large number of challenges and complete determination of all these CRPs within a limited time frame is nearly impossible[18]. Typically, many physical components are involved in the generation of a response, and it possesses a large possibility of challenges that can be applied to the PUF. For a decade, many strong PUF models have been proposed and are in use for crucial cryptographic applications that demand highly secured PUF mechanism

1.4.2 Weak PUF

Weak PUFs are those with few challenges and in some cases, just one challenge. Weak PUFs also exhibit unclonable physical behavior to some extent, but not as much as strong PUFs[18]. They are typically used for key storage applications. Access to the challenge-response interface or the challenge-response mechanism is usually restricted. The predictability of a weak PUF response is much higher than a strong PUF. In the case of a strong PUF, even if an adversary is aware of a large subset of CRPs, extrapolation of remaining unknown CRPs is very difficult

1.5 PUF Evaluation Metrics

Among many, uniqueness and reliability are the two most widely used metrics to evaluate a PUF. We could also include other important metrics such as resource usage and power consumption in this work.

1.5.1 Uniqueness

Uniqueness refers to the difference between the responses of a PUF implemented on different chips, considering the same challenge is provided[6]. The Uniqueness (or) Intra hamming distance is given by:

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

where 'k' refers to the total number of chips and HD stands for hamming distance

An ideal PUF is considered to produce a uniqueness value of approximately 50% which indicates the maximum difference in the responses of two PUFs.

1.5.2 Reliability

Reliability is the most important factor in PUF performance. For a PUF implemented on a chip, reliability refers to the ability to produce the same response 'R' for a particular Challenge 'C', under different conditions such as time, aging, temperature variations, etc.,

$$U = (1 - \frac{1}{n} \sum_{t=1}^{n} \frac{(HD(P(C_i, t_0), P(C_i, t))}{m}) * 100\%$$

where n is the number of tests and m is the length of the response. The $P(C_i, t)$ denotes the t_{th} sample of $P(C_i)$ among n repeated responses[6]

An ideal PUF is considered to produce a reliability value of 100 percent, which in turn states that PUF will produce the same exact response after a certain time or at a different temperature etc.,

1.6 Applications of PUF

PUFs are widely used in high-security requirement scenarios, more specifically cryptography, Internet of Things (IoT) devices, and privacy protection.

1.6.1 Authentication

This is one of the primary uses for the PUF, which is commonly employed with reduced hardware overhead via a challenge-response protocol. A secure database is maintained that stores a whole set of CRPs from each PUF pair to be used. During the time of validation, a random set of CRPs are extracted from the database and applied to the PUF circuit. The acquired response is saved in the database and compared to the existing response from the database for the IC or FPGA. If the data matches, the PUF device (IC or FPGA) will be authenticated [7].

The main aspect of the PUF is that we may utilize different CRPs, and they are all random due to the manufacturing process of the IC or the FPGA, since all the circuits are not doped in the same concentration to function similarly and create the output with the same time delay. We implement authentication for the device using this PUF feature. We need a strong PUF for authentication because we will have a lot of CRPs for authentication.

1.6.2 Random Number Generation

Random number generators are used in various fields such as statistical sampling, gambling, computer simulation, cryptography, entirely randomized design, and other fields where an unpredictable output is desired. In general, hardware generators are chosen over pseudo-random techniques in situations where unpredictability is a key characteristic, such as security applications[7]

Strong PUFs have a large set of CRPs, which makes them highly desirable random number generators(RNG).In the last decade, many PUFs have been proposed with increased complexity in the algorithm/functionality, which in turn decreases its vulnerability. These PUFs make a perfect match for highly secure RNGs.

1.7 Multi PUF technique

Multi PUF models [MPUF] is a type of PUF architecture in which both strong PUF and weak PUF are used. Whether it is strong or weak PUF[22], both of them has its own set of advantages and disadvantages. Through this technique, the drawbacks of each other are mitigated.

One of the imminent concerns in the current PUF scenarios is vulnerability to Machine Learning [ML] attacks. Due to the rapid advancement in the field of Artificial Intelligence [AI] and ML, many advanced sophisticated algorithms have been introduced to model a PUF[5]. Modeling refers to the prediction of a large set of unknown CRPs on the basis of a small subset of known CRPs. Popular attacks among them are CMAES (Covariance Matrix Adaptation and Evolution Strategy) and LR (Linear Regression) attacks. These attacks have successfully modeled many strong PUFs with a nearly 100 percent prediction rate.

The MPUF techniques are proven to be successful in increasing security towards these ML attacks[5]. In some models, a huge decrease of more than 50 percent prediction rate has been displayed. But these techniques either come with a compromise in uniqueness values or with a requirement of substantial resource overhead

1.8 Thesis Objectives

In this thesis, we proposed, designed, and evaluated new resource-efficient MPUF techniques compared to the existing traditional MPUF model. The proposed PUFs were tested on Xilinx Artix-7 FPGA. The experimental study was conducted and evaluation metrics such as uniqueness, reliability, and resource utilization were computed for the existing PUF models such as FFAPUF, FFAPUF-based Multi PUF, and XOR-based Multi PUF. Subsequently, the proposed Multi PUF techniques are implemented and the corresponding evaluation metrics are computed. Finally, both the experimental results for existing models and the proposed PUF model were compared. A resource-efficient PUF model was achieved which uses the MPUF technique.

1.9 Thesis Outline

The remainder of this thesis is organized as follows: we present a review of related work in Chapter 2. Chapter 3 describes the traditional MPUF model and suggests ways to improve resource usage. Chapter 4 presents the design concepts and experimental evaluation strategy of the proposed MPUF techniques. Chapter 5 presents the implementation and experimental results, evaluation, and comparison with related research, of the proposed PUF models on FPGA. Chapter 6 concludes the thesis with a summary and recommendations for future work.

CHAPTER 2

Related Work

Numerous PUF models have been proposed since the late 1990s. Meanwhile, most of them are based on these 3 basic principal PUFs: Arbiter PUF(APUF), Ring Oscillator PUF(ROPUF), and SRAM PUF.

Many strong PUFs with high reliability and desired uniqueness have been proposed considering the Arbiter PUF (APUF) and Ring Oscillator PUF (ROPUF) as their base. Some examples based on arbiter PUF are flip-flop-based APUF[17], XOR-ed APUF[3], Feedback Oriented XOR-ed FFAPUF[4], etc. Some examples based on ring oscillator PUF are bistable RO-PUF[19], hybrid PUF[9], and programmable ROPUF based on a switch Matrix.

The advancement in the field of Artificial Intelligence (AI) led to a few machine learning algorithm attacks, which successfully modeled these above-mentioned PUFs. Popular machine learning attacks in this domain are LR attacks (linear regression) and CMA-ES attacks (Covariance matrix adaptation evolution strategy)[3][5]. Since then, the primary goal in the domain of PUF design has shifted to resilience toward ML attacks instead of uniqueness and reliability. MPUF model is one of the obfuscation techniques that has shown good results in terms of resistance towards attacks but as a trade-off with comparatively a bit less uniqueness and reliability

2.1 Arbiter PUF

The Arbiter PUF Design is shown in Fig. 2.1.1 [1]. It comprises two multi-stage multiplexer chains running in parallel to each other. These chains are interconnected

in a cross or straight manner to the adjacent multiplexers, and the input signal is shared between the first two multiplexers. Depending on the challenge bits, this shared signal is further propagated along cross or straight paths.

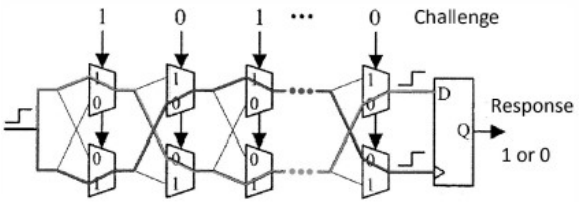

Fig. 2.1.1: Arbiter PUF [1]

At the final stage, a d-flip flop is utilized as an arbiter to compare the arrival times [1] of the two signals. Due to inherent manufacturing variations, the signals arriving at the end of the two parallel paths will experience different delays. The arbiter, in this case, compares these delays between the two signals and generates the corresponding response bit. This mechanism exploits the timing differences induced by manufacturing variations to produce unique and unclonable response bits in the arbiter PUF.

However, this architecture suffers from low uniqueness, meaning that it may not provide the desired level of distinctiveness in response patterns. Consequently, there was a pressing need to explore other PUF designs that can achieve higher uniqueness. Artix-7 FPGA has been used in this work to validate the design. It has produced a uniqueness of 33.9 percent and a reliability of 96.09 percent.

2.1.1 Novel Lightweight Flip Flop based Arbiter PUF Design

The Flip-Flop-based Arbiter PUF design (FFAPUF) presents a compact architecture with a focus on strong uniqueness and high reliability [2]. Similar to the arbiter PUF, the FFAPUF also consists of two parallel lines through which electrical signals race simultaneously. Each line contains a fixed number of slices, serving as delay entities.

Each slice comprises 4 flip-flops and 3 multiplexers. Within each slice, 3 challenge bits are employed, with one assigned to each multiplexer. As a result, the number of challenge bits used must be a multiple of 3. The slice of an FFAPUF is shown in Fig. 2.1.2.

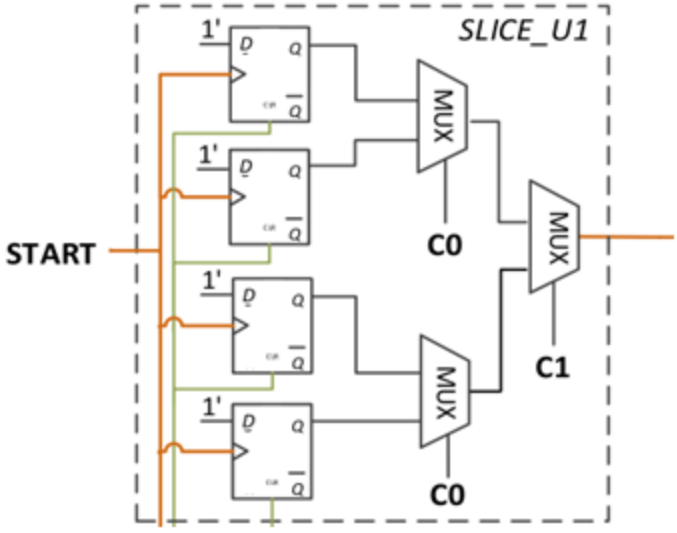

Fig. 2.1.2: Slice of FFAPUF[2]

In contrast to the Arbiter PUF, the FFAPUF utilizes an SR-latch instead of an arbiter to determine the fastest path between the two lines. An SR-Latch made with NAND gates is shown in Fig 2.1.3. This design achieved a uniqueness of 40% and a reliability of 96.10%. Additionally, the FFAPUF was subjected to Modelling Attacks, and the obtained prediction rates are also documented in this study. Two popular Modelling attacks, namely the LR attack and CMA-ES attack, were considered.

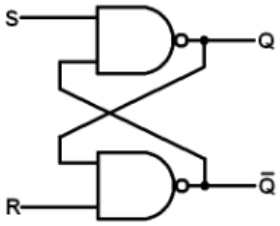

Fig. 2.1.3: SR latch

This FF-APUF requires 44x64 slices to generate a 64-bit response on an Artix-7

FPGA. This design showcases its potential for providing a balance between compactness, uniqueness, and reliability, making it a promising choice for hardware-based authentication and cryptographic applications. However, further optimizations are needed to enhance its performance against potential attacks and improve its resource efficiency on different FPGA platforms. The design of FFAPUF is given below in Fig. 2.1.4.

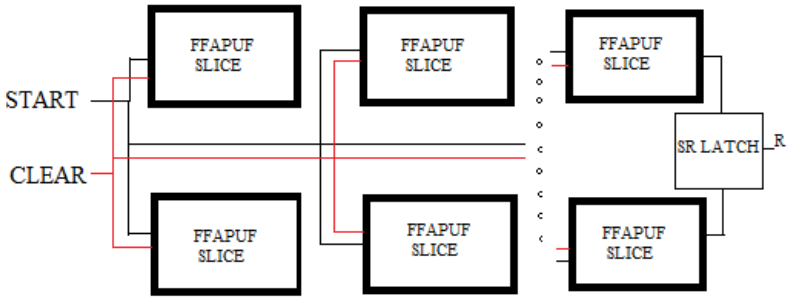

Fig. 2.1.4: FAPUF Design

2.1.2 XOR based Arbiter PUF

The XOR-Arbiter PUF[3] is shown in Fig. 2.1.5. The principle behind this circuit is based on performing an XOR operation between the response outputs of parallel Arbiter PUFs, which leads to a comparatively higher level of security. However, a drawback of this model is the generation of various unstable responses for a particular set of challenges, resulting in lower reliability and uniqueness. The value 'n' represents the number of PUF lines being XORed, and the security of the PUF increases with a higher value of 'n'. The research suggests that in order to achieve a secure design against machine learning attacks, a minimum of 10 PUF lines should be XORed, assuming a very large training set size. However, the disadvantage of this PUF lies in its consumption of a large number of resources, as it requires 10 PUF lines to generate just a 1-bit response. To strike a balance between security and resource usage, further

optimizations are necessary to enhance the efficiency of the design.

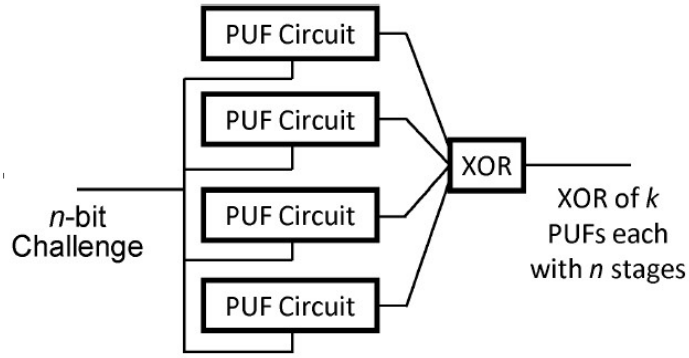

Fig. 2.1.5: XOR PUF Circuit

2.1.3 Feedback Oriented XORed Flip Flop based Arbiter PUF

R. Sushma [4] introduces a novel Feedback-oriented XORed FFAPUF (FOXFFAPUF) as its main contribution. This PUF design combines the fundamental concepts of FFAPUF and XORed APUF. Initially, the architecture resembles FFAPUF, with the subsequent step involving the XOR combination of responses. The multiplexers in each slice are linked to an added delay component, for which a feedback-embedded d-flipflop is employed. This extension results in the creation of a new slice termed Type-1 slice, shown in Fig 2.1.6.

Given that FFAPUF utilization limits to a challenge bit count of a multiple of 3, the FOXFFAPUF introduces a new slice known as the Type-2 slice to accommodate this scenario. In the Type-2 slice shown in Fig 2.1.7, 3 flip-flops are employed along with one challenge bit, in contrast to the 7 flip-flop configurations seen in the previous slice.

The novel PUF method suggested in this study exhibited superior levels of uniqueness compared to FFAPUF. However, a notable drawback is its substantial resource

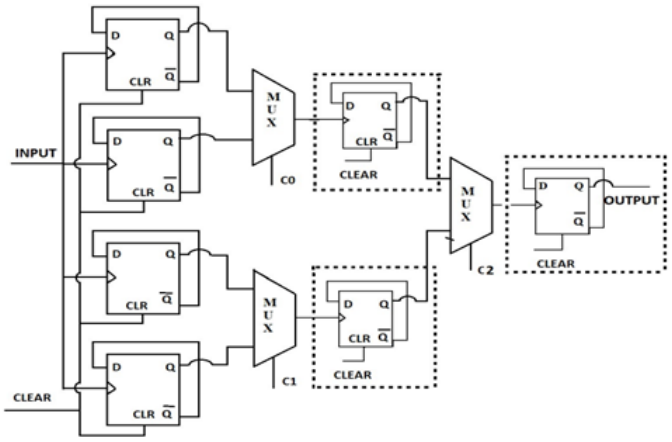

Fig. 2.1.6: Slice 1 of FOXFFAPUF[4]

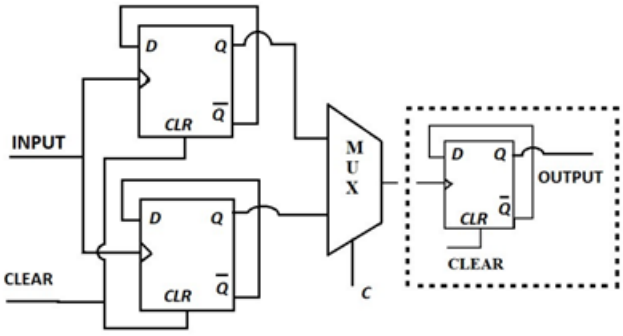

Fig. 2.1.7: Slice 2 of FOXFFAPUF[4]

consumption on an FPGA platform. Specifically, when generating an 8-bit challenge response on a Virtex-6 FPGA, this method utilized 960 slices. In contrast, the FFA-PUF achieved the same outcome using 192 slice registers, highlighting a significant difference in resource utilization between the two approaches.

CHAPTER 3

Experimental Study of the Existing MPUF Design

In this chapter, the design concepts and the implementation results of the existing MPUF-based FFAPUF[5] are presented. Experimental results of the design, as well as its evaluation metrics, are provided.

3.1 Design Concepts

The MPUF design strategy aims to create a highly resilient PUF architecture, particularly in the face of machine learning attacks [5]. The MPUF approach involves using a weak PUF to obfuscate the challenge bits, and then these modified challenge bits are fed into a strong PUF. The basic MPUF principle is depicted in Fig. 3.1.1. The weak PUF can be any weak PUF, such as a basic conventional arbiter PUF or otherwise called PICO PUF shown in Fig. 3.1.2.

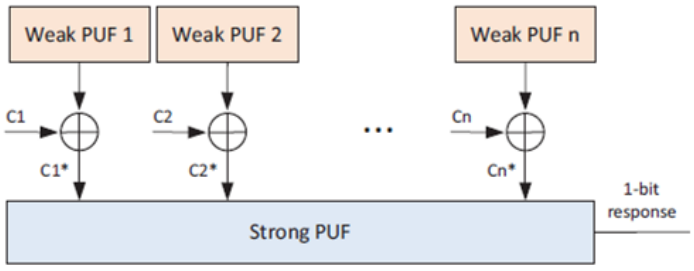

Fig. 3.1.1: Basic MPUF Design

In terms of resilience against Machine Learning (ML) Attacks, the MPUF demonstrates significant improvement, with a drastic decrease of 50%[5] in the prediction

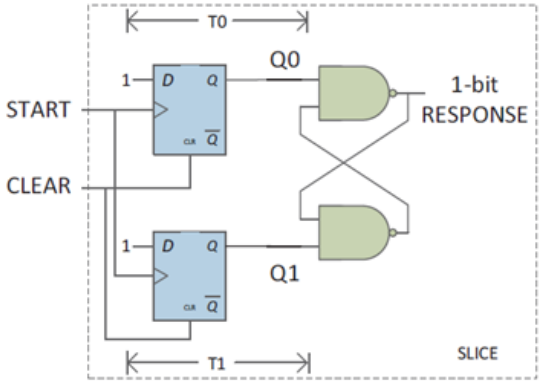

Fig. 3.1.2: PICO PUF

rate compared to other PUF designs. This means that attackers' ability to predict responses accurately is significantly hindered when using the MPUF. Specifically, CMA-ES attacks, which are known to be highly effective, achieve a 100%[5] prediction rate for conventional Arbiter PUFs. However, when applied to an MPUF-based Arbiter PUF, the prediction rate drops to less than 80%[15], even with large training sets. This substantial reduction in prediction rate indicates the superior robustness of the MPUF design against advanced attacks. Fig. 3.1.3 [5] and Fig 3.1.4[5] depict the prediction rates of APUF versus MPUF.

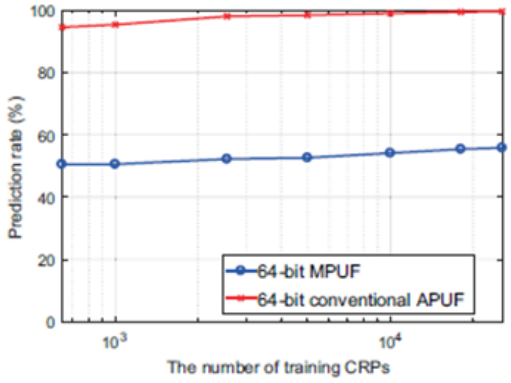

Fig. 3.1.3: Predcition rates for conventional APUF and MPUF designs by LR attack[5]

Overall, the MPUF strategy combined with the XOR technique stands as a promising solution for enhancing PUF security by combining the strengths of weak and strong PUFs, thereby thwarting the effectiveness of ML Attacks and making it a

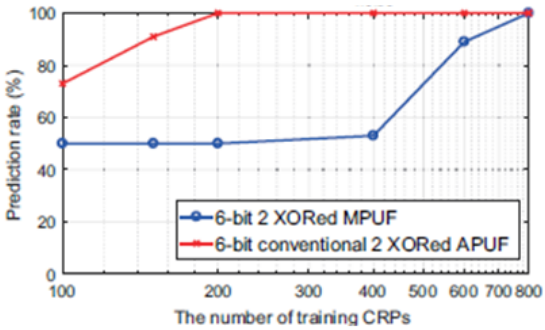

Fig. 3.1.4: Predcition rates for XORed APUF and XORed MPUF designs by CMA-ES attack[5]

viable option for applications that demand high-security measures.

3.2 Experimental Evaluation

To analyze the working principle of this Multi-PUF design, we started creating Verilog HDL (Hardware Description Language) for each component of the desired PUF model. By using the Xilinx Vivado Design Suite, we synthesized the Verilog model of the PUF for implementation and evaluation in Xilinx Artix-7 FPGA.

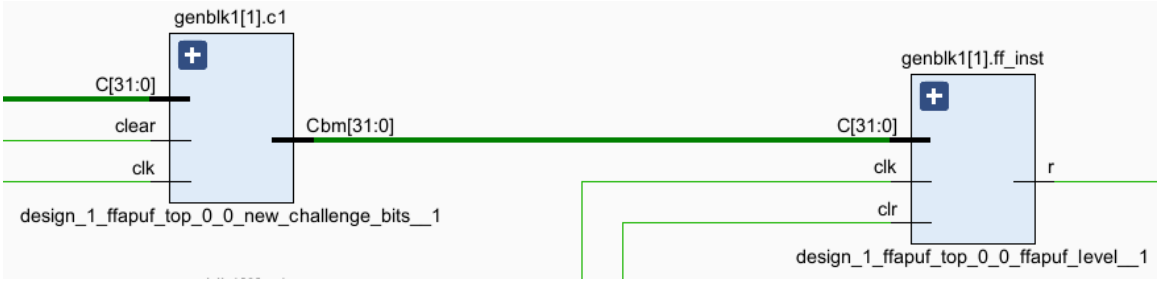

Fig. 3.2.1: RTL Schematics of a 1-bit response MPUF

In the above Fig. 3.2.1, the C[31:0] are the original challenge bits and the Cbm[31:0] are the new obfuscated challenge bits that act as the input to the strong PUF and we receive a 1-bit output. Such PUF lines are repeated 32 times to receive a 32-bit output.

We also performed the implementation of the XOR-based MPUF technique[5].

Here, we utilized 3 PUF levels for the XOR operation. For these experiments, both Look-Up Tables (LUTs) and Slice Registers of the FPGA are considered resources. We also implemented PUF components such as MUX, D-flipflop, NAND gates and XOR gates for the experimental evaluation.

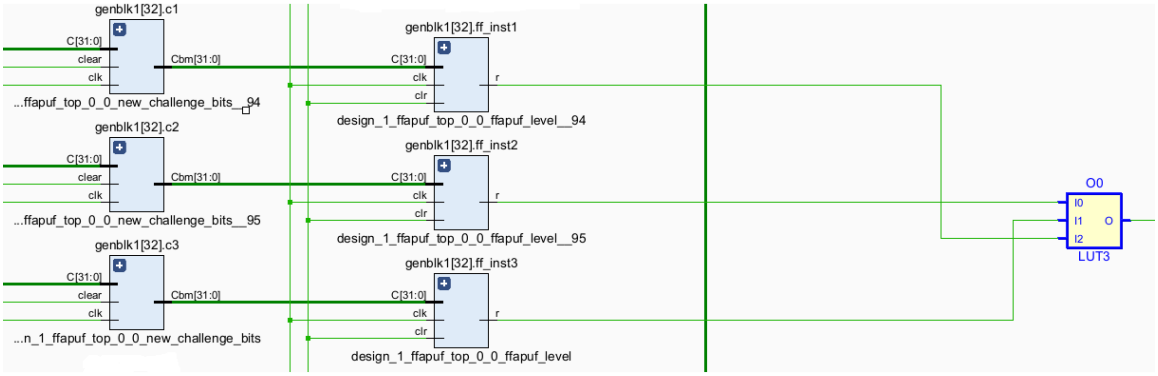

Fig. 3.2.2: RTL Schematics of a 1-bit response XORed MPUF

3.2.1 IP Integration

In order to manage the control signals for the PUF, we devised a MicroBlaze-based processing unit. This unit was responsible for both controlling the input challenge given to the PUF and collecting the resulting output response.

Incorporating Xilinx IPs facilitated this process, utilizing tools like Vivado's IP integrator. We employed various Xilinx IPs including AXI interconnection bridges, AXI GPIO, Memory units, and the Xilinx UART IP.

The integration of these IPs was crucial for the evaluation of the PUF. The manual handling of real-time 32-bit input and output acquisition from the Bi-stable Ring PUF is practically challenging. Therefore, opting for the creation of a processing unit was more practical. This unit was capable of supplying the 32 challenge bits required and capturing the entire 32-bit output response, as illustrated in Figure 3.2.3.

As presented in Fig. 3.2.4, we integrated the above-mentioned IPs to control the PUF using the AXI GPIO, which can be controlled by the vitis IDE. This function

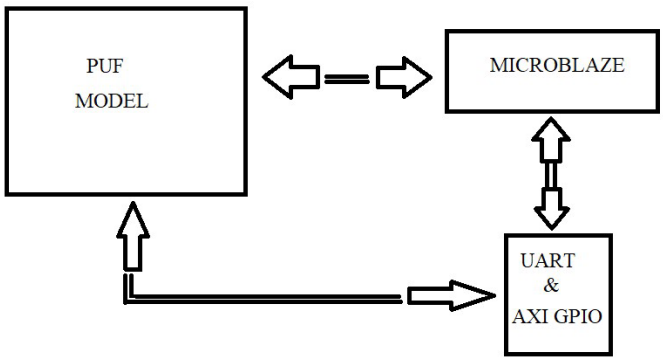

Fig. 3.2.3: Interface diagram of the top module of the system

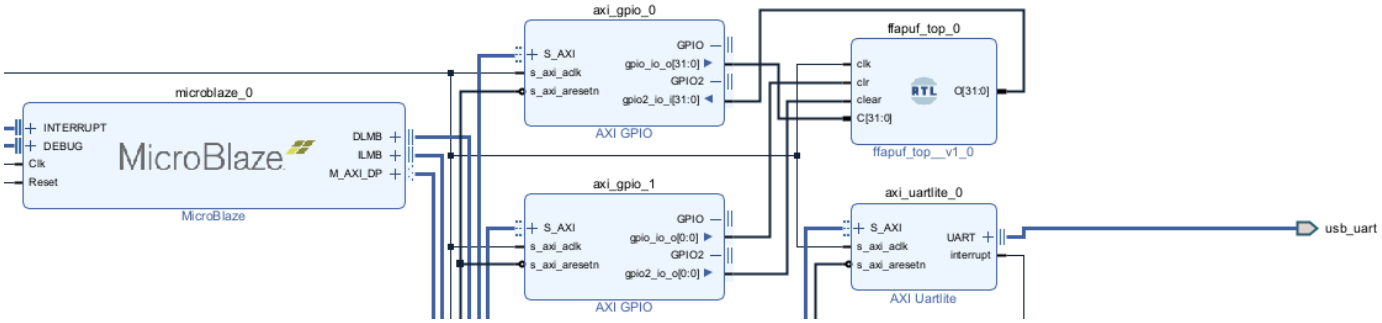

Fig. 3.2.4: Block diagram of the Xilinx IP Integrator

helps us to collect the CRPs in text files through the UART serial port. We used a serial terminal software to monitor the CRPs and to collect them in text files.

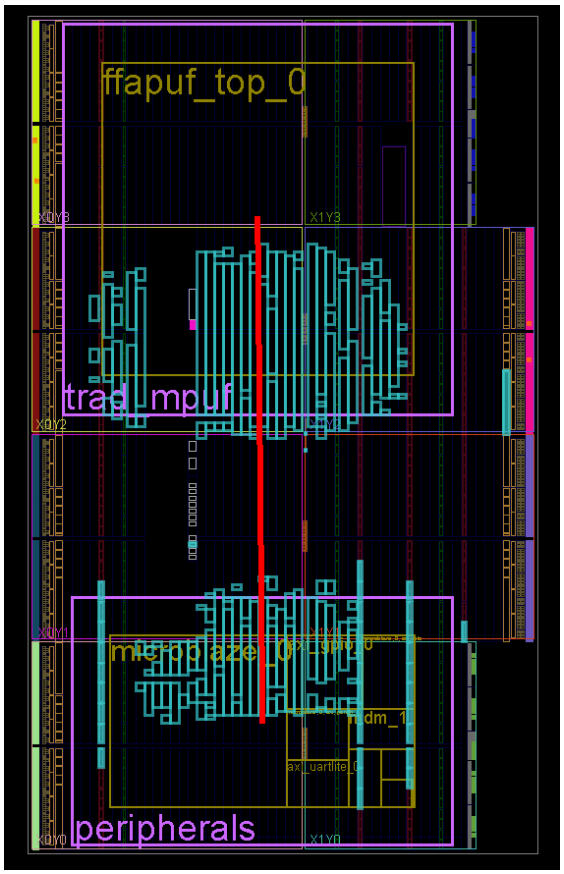

Fig. 3.2.5: Floor plan for the PUF on FPGA

3.2.2 Manual Place and Route

To analyze the resource consumption accurately, it was essential to isolate the PUF circuit from the processing unit we developed. Typically, when using the vivado tool, the entire circuit model is synthesized as a unified schematic. This can make it challenging to precisely assess the specific resources utilized by the PUF.To overcome this issue, we employed P-block allocation[24]. This technique enabled us to allocate a dedicated area within the FPGA specifically for the PUF circuit. As a result, we could obtain precise resource utilization data solely for the PUF, without the interference

or complexity of the larger circuit's synthesis[23][24]

In this particular experiment, we focused on examining the intra-chip Hamming distance between the responses generated by the PUF in two distinct locations, which can be done by the P-block allocation method. This analysis enabled us to establish the uniqueness metric of the PUF. To facilitate this process, we utilized TCL commands to generate constraint files. These files were instrumental in arranging the specific components of the PUF, including MUX, DEMUX, and NOR gates, within the desired Look-Up Tables (LUTs).

Figure 3.2.5 provides a visual representation of the floor plan of the PUF circuit on an FPGA. This was achieved through the creation of dedicated P-blocks, allowing us to designate specific areas for the PUF circuit within the overall layout.

3.3 Experimental Results

The corresponding Resource usage for the MPUF and XOR-based MPUF are recorded and tabulated. The number and percentage of Slices, LUTs, and register flip-flops used in the FPGA by the PUF design, without including the peripherals such as Microblaze, AXI GPIOs, and UART module are considered. Here we are considering 3 PUFs for the XOR operation.

Table 3.3.1: Resource usage of FFAPUF based Multi PUF

Type of Resource	***MPUF***	***XORed MPUF***
LUTs	5184	15584
Register flipflops	4864	14592

3.4 Limitations of MPUF model

In this section, we elaborate on the limitation of the traditional MPUF model in the FPGAs.

The primary motivation of the Physically Unclonable function is to deliver security with less resource usage as lightweight circuits [2]. According to this paper, it is concluded that the incorporation of the XOR technique with the MPUF produces magnificent results with respect to resilience toward ML attacks. As previously indicated, the XOR-based PUF operation necessitates the integration of two or more PUFs, significantly increasing the consumption of hardware resources. This, in turn, results in escalated implementation costs and time, rendering the approach less efficient. The experimental outcomes states that this specific XOR operation might not be well-suited for numerous resource-constrained applications. Therefore, there's a need for adjustments or modifications to mitigate the substantial resource overhead and enhance resource efficiency.

CHAPTER 4

Proposed MPUF - Design and Evaluation

In this chapter, we provide the implementation and evaluation results for the proposed design in terms of uniqueness, reliability, and resource utilization. First, the design and implementation of the proposed design were explained, followed by a comparison of the results of the proposed design and the related PUF models.

4.1 Design Procedure of the Proposed MPUF techniques

In this section, we propose the novel MPUF techniques MPUF-t1 and MPUF-t2. These techniques incorporate the principles of FOXFFAPUF, FFAPUF, and traditional Multi-PUF. The resulting proposed MPUF technique proves to be resource-efficient while maintaining the existing uniqueness and reliability values.

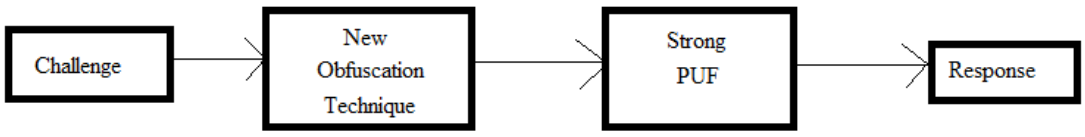

Fig. 4.1.1: Block Diagram of the PUF with Proposed MPUF technique

As previously stated, an MPUF is a combination of weak PUF and strong PUF[5]. Here, we introduce two new MPUF techniques, denoted as "MPUF-t1" and "MPUF-t2" to obfuscate the challenge bits. Both techniques employ PICO PUFs as weak PUFs

for the challenge obfuscation process. Subsequently, the new challenge bits generated by the weak PUFs are fed into the flip-flop-based arbiter PUF (FFAPUF)[2], which functions as the strong PUF in this scenario. For an N-bit challenge, it is observed that MPUF-t2 requires a smaller area than MPUF-t1.

4.1.1 Challenge Obfuscation

As discussed in Chapter 3, improvement in resource efficiency is required in the traditional MPUF model in order to make it suitable for resource-constrained applications. In our proposed MPUF technique, we employ a different strategy in the challenge obfuscation process. We propose two obfuscation techniques resulting in MPUF-t1 and MPUF-t2 respectively. Particularly, the MPUF-t2 results in providing a high resource efficiency. These two techniques are discussed thoroughly in the subsequent sections.

4.1.1.1 MPUF-t1

In MPUF-t1, the design consists of 3 XOR gates, 2 2:1 multiplexers, and 2 PICO PUFs. The output of one PICO PUF serves as the select line for the two multiplexers. The inputs of the two multiplexers are the outputs of the respective PICO PUFs and are also interchanged. Fig. 4.1.2 illustrates the basic building unit of t1. Such units are repeated for all the existing challenge bits.

The following steps are then considered in t1 to obtain new challenge bits for the strong PUF:

- The output from MUX1 is XORed with the challenge bits Cn-2 and Cn-1 separately.
- The output from MUX2 is XORed with the challenge bit Cn.
- The results obtained from the XOR operations in steps 1 and 2 are considered

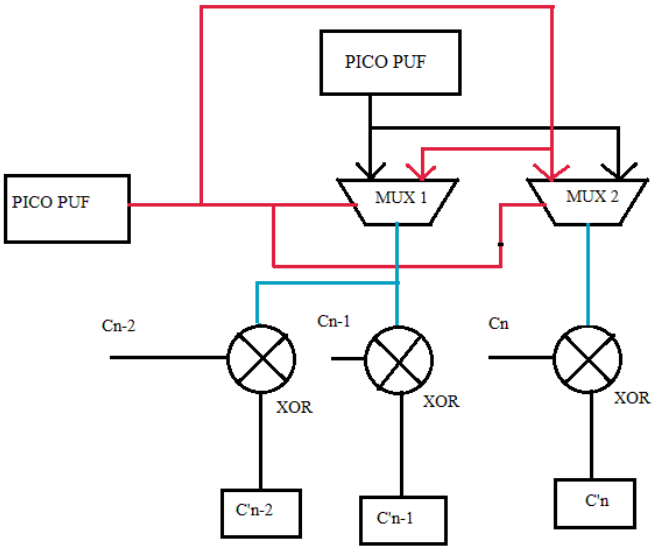

Fig. 4.1.2: Basic Unit of MPUF-t1

as new challenge bits for the strong PUF, denoted as C'n-2, C'n-1, and C'n, respectively.

4.1.1.2 MPUF Technique-2

This strategy is basically an extension of MPUF-t1. The new challenge bits obtained after obfuscation in t1 are then XORed again to obtain 1 new bit shown as CB_m, and also the value of m is $m = \lfloor n/3 \rfloor$. Fig. 4.1.3 illustrates the obfuscation technique MPUF-t2.

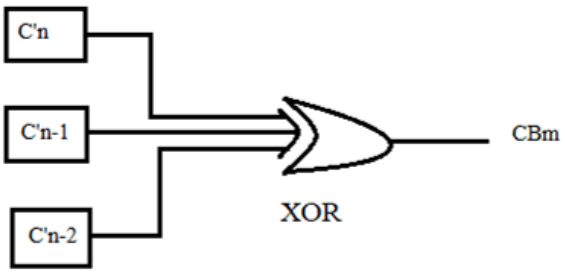

Fig. 4.1.3: MPUF-t2

4.1.2 Strong PUF models

For our research, we are considering the FFAPUF as our strong PUF. The new obfuscated challenge bits obtained through the proposed MPUF techniques act as input to the FFAPUF[2]. The architecture of the FFAPUF is shown in Fig 2.1.2 of Chapter 2. The components required for designing an FFAPUF are D-flipflops, 2:1 Multiplexers, and NAND SR Latch.

4.1.3 XOR-ed PUF model

Additional to the MPUF procedure, we also employed the XOR technique in our proposed MPUF models[8]. We have considered 3 PUF levels for the XOR operation. Both MPUF-t1 and MPUF-t2 will undergo the 3-input XOR operation respectively.

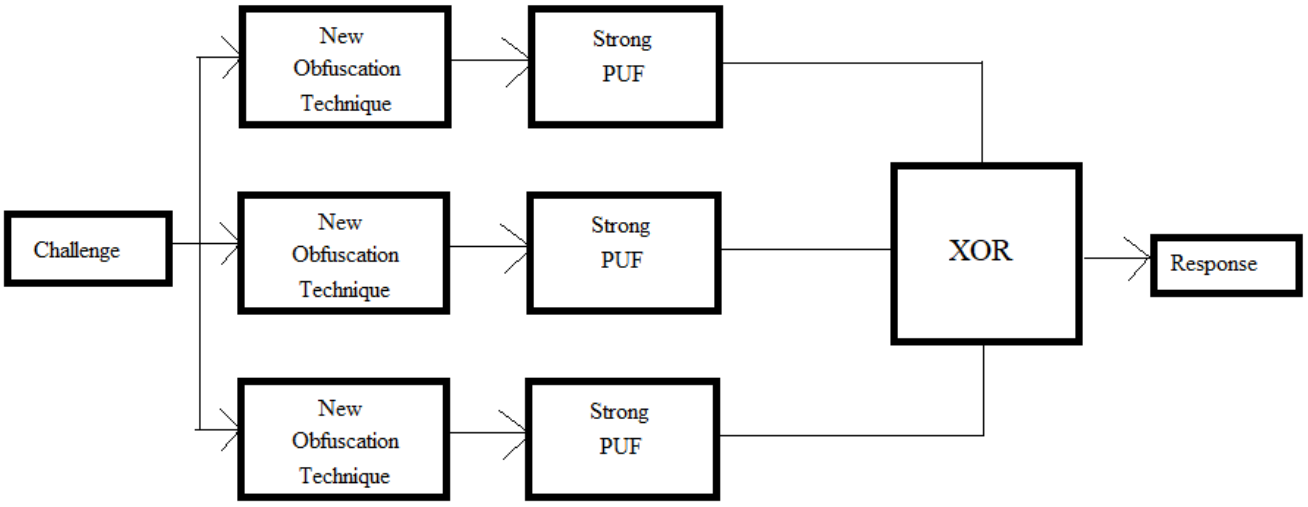

Fig. 4.1.4: Block Diagram of the XORed PUF with Proposed MPUF Design

4.2 Implementation of the Proposed Multi PUF Designs

In this work, nine different PUF models have been implemented and analyzed on the Arty A7 FPGA using Vivado. These models are as follows:

- FFAPUF: The strong PUF, FFAPUF, serves as the reference design in the study. It is implemented and analyzed for 32-bit challenges.
- XORed FFAPUF: The FFAPUF is modified with 3 XOR lines, and corresponding Challenge-Response Pairs (CRPs) are recorded and analyzed.
- FOXFFAPUF: The modified strong PUF, FFAPUF, serves as the reference design in the study. It is implemented and analyzed for 32-bit challenges.
- Traditional MultiPUF: The traditional Multi PUF design, is implemented and is considered as a comparative reference for our proposed PUF.
- Traditional XORed MultiPUF: The traditional Multi PUF design combined with the XOR technique is implemented and is considered as a comparative reference for our proposed PUF.
- Proposed MPUF-t1: The first proposed Multi PUF technique MPUF-t1, is implemented and incorporated with the strong PUF, FFAPUF.
- Proposed MPUF-t2: The second proposed Multi PUF technique MPUF-t2, is implemented and incorporated with the strong PUF, FFAPUF.
- XOR-based MPUF-t1: This design combines the architectures of XORed FFAPUF and MPUF-t1.
- XOR-based MPUF-t2: This design combines the architectures of XORed FFAPUF and MPUF-t2.

For the implementation, the larger 100t variant of the Arty A7 FPGA is used, as it is capable of accurately identifying smaller delay variations between the two parallel paths for the considered number of challenges. The XOR strategy is applied to enhance the overall security and unpredictability of the Multi PUF techniques, providing valuable insights into their effectiveness for hardware-based authentication and cryptographic applications.

The implementation of the PUF is done using the Xilinx Vivado tool. Firstly, respective verilog codes for the PUF components such as D-flipflop, 2:1 Multiplexer, 3-input XOR gate, and NAND SR Latch are written. This serves as the base for designing proposed MPUF techniques and FFAPUF.

Using the principles of FFAPUF[2] and FOXFFAPUF[4], we design two types of FFAPUF slices called type-1 slice and type-2 slice. We are considering 32-bit challenges for the evaluation procedure. The type-1 slice requires 3 bits of challenges and the type-2 requires 2 bits of challenges. Using these two types of slices we can accommodate 32 bits of challenges. These are illustrated in Fig 4.2.1 and 4.2.2 respectively.

4.2.1 IP Integration

A large number of challenge set is considered for the evaluation of PUF devices, which in turn leads to a large response set. Also, the system clock, and reset serve as input to the system. So in order to manage all these control signals for the PUF, we devised a MicroBlaze-based processing unit. This unit will be responsible for both controlling the input challenge given to the PUF along with collecting the resulting output response, and it also provides timely reset signals for the flipflops.

Incorporating Xilinx IPs facilitated this process, utilizing tools like Vivado's IP integrator. We employed various Xilinx IPs including AXI interconnection bridges,

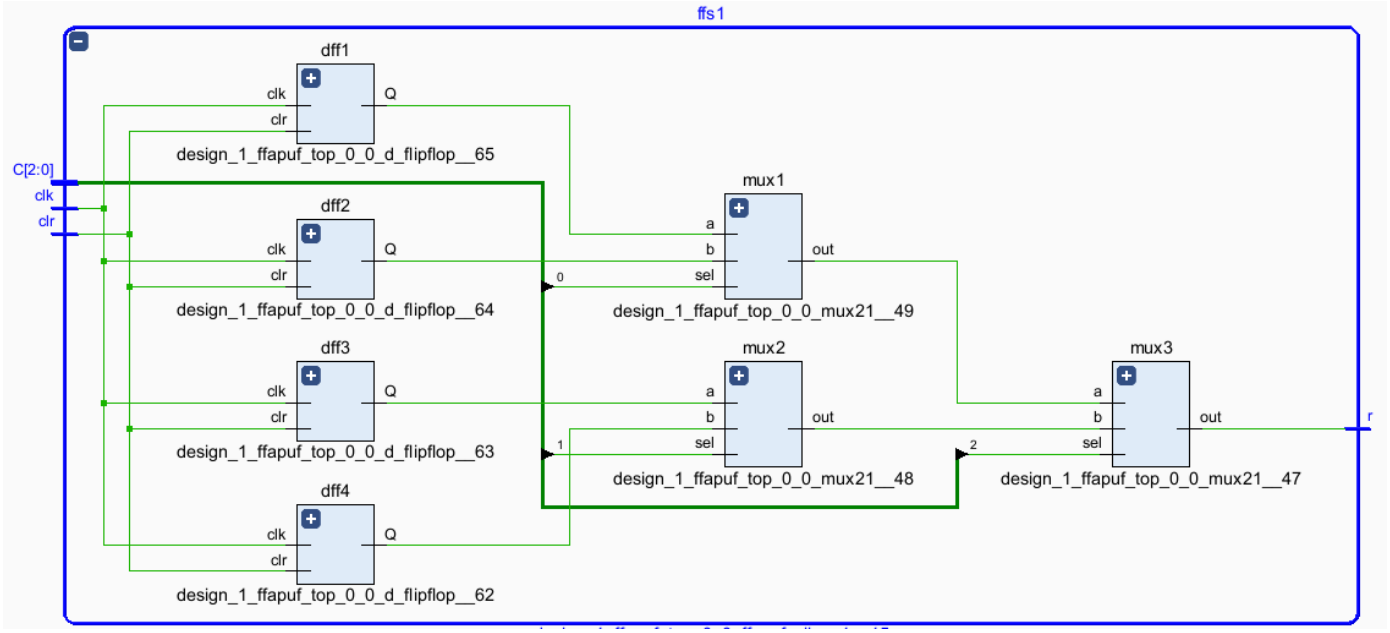

Fig. 4.2.1: RTL of FFAPUF Type-1 slice

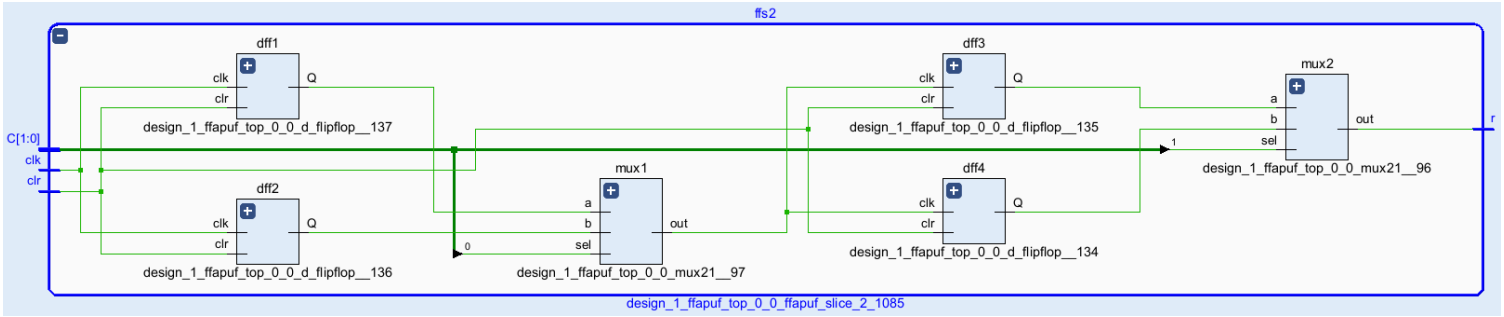

Fig. 4.2.2: RTL of FFAPUF Type-2 slice

AXI GPIO, Memory units, and the Xilinx UART IP.

The integration of these IPs was crucial for the evaluation of the PUF. The manual handling of real-time 32-bit input and output acquisition from the MPUF is practically challenging. Therefore, opting for the creation of a processing unit was more practical. This unit was capable of supplying the 32 challenge bits required and capturing the entire 32-bit output response. A similar unit is used in the implementation of the traditional MPUF and illustrated in Figure 3.2.3 of Chapter 3.

4.2.2 Manual Place and Route

We used the P-block allocation method to isolate the PUF circuit from peripherals such as Microblaze, AXI GPIOs, etc[23][24]. The PUF circuit is placed in two different locations, and this is used to find the inter-chip hamming distance values, which in turn represents the important PUF metric known as Uniqueness.

This also helps in evaluating the resource usage such as LUTs and Register FFs, utilized by the PUF design on an FPGA.

4.2.3 Vitis and Serial terminal software

Once the design is implemented in Vivado, the corresponding bit stream is generated. This bitstream file is then used in Vitis to program a PUF on an FPGA. Here, corresponding Challenges along with control signals are also defined. The responses are obtained through USB UART protocol and can be recorded using serial terminal emulators. These recorded CRPs are saved into files and are then used to evaluate the PUF metrics.

As our PUF models utilize 32-bit challenges, the total number of challenges that can be used is 2^{32}. But the number of challenges used for this experiment is 16384 which is 2^{14}.

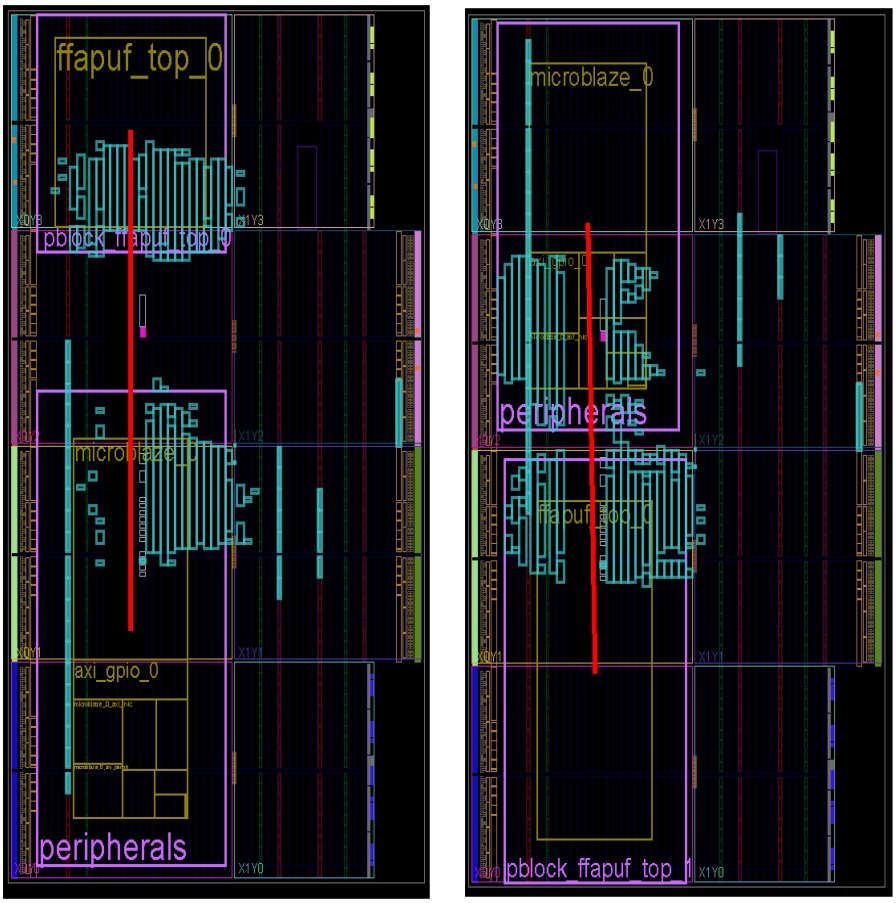

Fig. 4.2.3: Floor plan of MPUF-t2 in two different locations

4.3 Experimental Results and Analysis

In this section, we present the overall comparison of the existing MPUF models along with our proposed MPUF techniques.

All nine PUF designs are implemented and respective CRP sets have been recorded and stored for Experimental analysis. The first table depicts the PUF metrics of different PUF models along with our proposed PUF MPUF-t1 and MPUF-t2.

Tables 4.3.2 and 4.3.3 compares the resource consumption of different PUF models. The resource types considered are LUTs and Register flipflops

Table 4.3.1 includes the uniqueness and reliability values of the respective PUFs. In terms of these parameters, we can conclude that our proposed MPUFs have main-

Table 4.3.1: Uniqueness and Reliability

PUF Model	***Uniqueness***	***Reliability***
FFAPUF	47.3%	99.56%
XORed FFAPUF	46.37%	99.0317%
Traditional MPUF	47.71%	98.54%
MPUF_t1	45.6%	99.06%
MPUF_t2	46.2%	98.14%
XORed Traditional MPUF	43.4%	98.02%
XORed MPUF_t1	44.57%	98.78%
XORed MPUF_t2	44.94%	98.92%

Table 4.3.2: Resource Consumption

PUF Model	***LUTs***	***Register FFs***
FFAPUF	2112	2816
XORed FFAPUF	6368	8448
FOXFFAPUF	6368	14208

Table 4.3.3: Resource Consumption of MPUF models

PUF Model	***LUTs***	***Register FFs***
Traditional MPUF (reference)	5184 (100%)	4864 (100%)
MPUF_t1	4992 (96.3%)	4096(84.3%)
MPUF_t2	3072 (59.3%)	2304(47.4%)
Traditional XORed MPUF (reference)	15584 (100%)	14592 (100%)
Trad MPUF-based FOXFFAPUF	15616 (100.2%)	20352 (139.47%)
XORed MPUF_t1	15008(96.30%)	12288 (84.21%)
XORed MPUF_t2	9248 (59.4%)	6912 (47.4%)

tained good uniqueness and reliability values, which sequentially reflect the quality of a PUF.

The Resources consumed by the proposed models are compared and tabulated below in Tables 4.3.2 and 4.3.3. The resources occupied on Arty a7 FPGA are shown. It provides a comprehensive overview of resource consumption across various PUF models. For the purpose of resource consumption analysis, the specific resource types taken into consideration are Look-Up Tables (LUTs) and Register Flip-Flops. The traditional MultiPUF technique is utilized as the benchmark for comparison in this context.

In comparison to the reference MPUF model, the MPUF-t1 model showcases a similar but noteworthy resource optimization. Specifically, it demonstrates a reduction of 3.7 percent in LUT consumption and a substantial decrease of 15.7 percent in register flip-flop usage. On the other hand, the MPUF-t2 model exhibits even more pronounced improvements in resource efficiency. It registers a significant decline of 40 percent in LUT consumption and an impressive 53 percent reduction in the utilization

of register flip-flops.

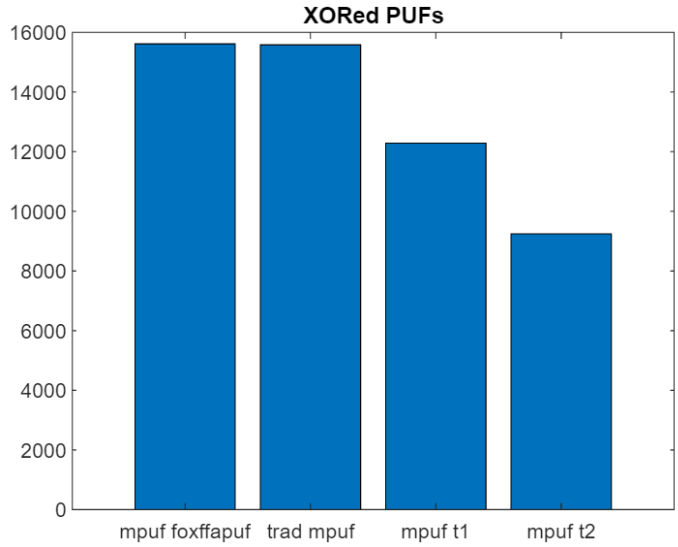

Fig. 4.3.1: LUT usage by various XORed MPUF models

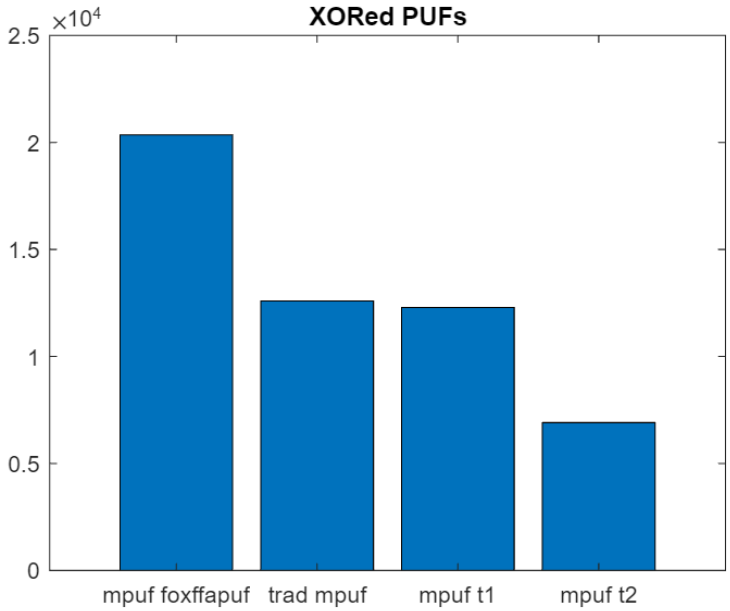

Fig. 4.3.2: Register flipflops usage by various XORed MPUF models

Fig 4.3.2 and Fig 4.3.1 portray the same for XORed PUF models by using bar graphs. These resource consumption figures reflect the enhanced efficiency achieved through the introduced MPUF techniques, particularly MPUF-t1 and MPUF-t2.

CHAPTER 5

Conclusion and Future Work

In this Chapter, we summarize the main contributions of this thesis and propose future work in related areas.

5.1 Summary of Contributions

In this thesis, we have implemented the following PUF models: FFAPUF, Traditional MPUF, and proposed MPUF models MPUF-t1 and MPUF-t2. We also incorporated XOR technique into all these models. The Arty A7 FPGA board was employed for the implementation and evaluation of all models including proposed Multi PUF techniques. Both techniques demonstrated commendable PUF quality, maintaining high levels of uniqueness and reliability similar to the traditional MPUF model. Notably, the resource consumption analysis highlighted the enhanced efficiency, particularly evident in the case of MPUF-t2.

The conducted analysis revealed a significant 9 percent average reduction in resource consumption for MPUF-t1, while MPUF-t2 exhibited an even more remarkable average decrease of 46.5 percent in resource consumption when compared to the conventional MPUF model. These efficiency gains can potentially contribute to more optimized and streamlined hardware designs, addressing critical considerations in modern hardware development.

The proposed Multi-PUF (MPUF) models hold significant potential for Multi-PUF applications, particularly those grappling with resource constraints. The outcomes showcased in this study are derived from the thorough implementation of these schemes on the Arty A7 FPGA board, considering multiple instances for robust evaluation. Furthermore, as previously addressed, the introduced MPUF technique holds broader applicability. Its characteristics can be explored more comprehensively by

synergizing it with other robust PUF models.

5.2 Future Work

For future endeavors, it is plausible to extend the research by subjecting the proposed MPUF techniques to potential attacks. This exploration would provide insights into the resilience of these techniques against machine learning (ML) attacks, enhancing the overall understanding of their security characteristics. Moreover, considering the vulnerability aspect, future work might involve experimenting with altering delay elements within the PUF models to adapt to varying levels of vulnerability, further enhancing the robustness and adaptability of the proposed techniques.

By combining the strengths of the proposed MPUF technique with other established strong PUF models, there exists the prospect of creating even more sophisticated and secure architectures. Such collaborations could lead to the development of innovative and hybridized PUF approaches that cater to various security and efficiency demands across diverse hardware applications.

REFERENCES

[1] Mohammed Saeed Alkatheiri, Yu Zhuang, Mikhail Korobkov, and Abdur Rashid Sangi. An experimental study of the state-of-the-art pufs implemented on fpgas. In *2017 IEEE Conference on Dependable and Secure Computing*, pages 174–180, 2017. doi: 10.1109/DESEC.2017.8073844.

[2] Chongyan Gu, Yijun Cui, Neil Hanley, and Máire O'Neill. Novel lightweight ff-apuf design for fpga. In *2016 29th IEEE International System-on-Chip Conference (SOCC)*, pages 75–80, 2016. doi: 10.1109/SOCC.2016.7905439.

[3] Chen Zhou, Keshab Parhi, and C.H. Kim. Secure and reliable xor arbiter puf design: An experimental study based on 1 trillion challenge response pair measurements. pages 1–6, 06 2017. doi: 10.1145/3061639.3062315.

[4] R. Sushma and N.S. Murty. Feedback oriented xored flip-flop based arbiter puf. In *2018 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT)*, pages 1444–1448, 2018. doi: 10.1109/ICEECCOT43722.2018.9001605.

[5] Qingqing Ma, Chongyan Gu, Neil Hanley, Chenghua Wang, Weiqiang Liu, and Maire O'Neill. A machine learning attack resistant multi-puf design on fpga. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 97–104, 2018. doi: 10.1109/ASPDAC.2018.8297289.

[6] Urbi Chatterjee, Rajat Chakraborty, and Debdeep Mukhopadhyay. A puf-based secure communication protocol for iot. *ACM Transactions on Embedded Computing Systems*, 16:1–25, 04 2017. doi: 10.1145/3005715.

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

[8] G. Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *2007 44th ACM/IEEE Design Automation Conference*, pages 9–14, 2007.

[9] Madhan Thirumoorthi, Marko Jovanovic, Mitra Mirhassani, and Mohammed Khalid. Design and evaluation of a hybrid chaotic-bistable ring puf. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 29(11):1912–1921, 2021. doi: 10.1109/TVLSI.2021.3111588.

[10] Mohammad Ebrahimabadi, Mohamed Younis, Wassila Lalouani, and Naghmeh Karimi. A novel modeling-attack resilient arbiter-puf design. In *2021 34th International Conference on VLSI Design and 2021 20th International Conference on Embedded Systems (VLSID)*, pages 123–128, 2021. doi: 10.1109/VLSID51830.2021.00026.

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

[12] Akshay Gireesh, Ramesh Bhakthavatchalu, and K N Devika. Performance analysis of different types of delay based pufs. In *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, pages 180–184, 2022. doi: 10.1109/ICOEI53556.2022.9776755.

[13] Takanori Machida, Dai Yamamoto, Mitsugu Iwamoto, and Kazuo Sakiyama. A new arbiter puf for enhancing unpredictability on fpga. *The Scientific World Journal*, 2015, 09 2015. doi: 10.1155/2015/864812.

[14] Husam Kareem and Dmitriy Dunaev. Physical unclonable functions based hardware obfuscation techniques: A state of the art. In *2021 16th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–6, 2021. doi: 10.23919/CISTI52073.2021.9476669.

[15] Swati K Kulkarni, R. M. Vani, and P. V. Hunagund. Implementation of arbiter physical unclonable function on the xilinx system on chip fpga. In *2022 9th International Conference on Computing for Sustainable Global Development (INDIACom)*, pages 648–653, 2022. doi: 10.23919/INDIACom54597.2022.9763207.

[16] Sk. Shariffuddin, N.M Sivamangai, A Napolean, R Naveenkumar, S Kamalnath, and G Saranya. Review on arbiter physical unclonable function and its implementation in fpga for iot security applications. In *2022 6th International Conference on Devices, Circuits and Systems (ICDCS)*, pages 369–374, 2022. doi: 10.1109/ICDCS54290.2022.9780766.

[17] Chongyan Gu, Weiqiang Liu, Yijun Cui, Neil Hanley, MÁire O'Neill, and Fabrizio Lombardi. A flip-flop based arbiter physical unclonable function (apuf) design with high entropy and uniqueness for fpga implementation. *IEEE Transactions on Emerging Topics in Computing*, 9(4):1853–1866, 2021. doi: 10.1109/TETC.2019.2935465.

[18] Zhangqing He, Wanbo Chen, Lingchao Zhang, Gaojun Chi, Qi Gao, and Lein Harn. A highly reliable arbiter puf with improved uniqueness in fpga implementation using bit-self-test. *IEEE Access*, 8:181751–181762, 2020. doi: 10.1109/ACCESS.2020.3028514.

[19] Qingqing Chen, György Csaba, Paolo Lugli, Ulf Schlichtmann, and Ulrich Rührmair. The bistable ring puf: A new architecture for strong physical unclonable functions. In *2011 IEEE International Symposium on Hardware-Oriented Security and Trust*, pages 134–141, 2011. doi: 10.1109/HST.2011.5955011.

[20] M. Tehranipoor and C. Wang. *Introduction to Hardware Security and Trust*. 10 2012. ISBN 978-1-4419-8079-3. doi: 10.1007/978-1-4419-8080-9.

[21] Wei Hu, Chip-Hong Chang, Anirban Sengupta, Swarup Bhunia, Ryan Kastner, and Hai Li. An overview of hardware security and trust: Threats, countermeasures, and design tools. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 40(6):1010–1038, 2021. doi: 10.1109/TCAD.2020.3047976.

[22] David Lie, John Mitchell, Chandramohan A. Thekkath, and Mark A. Horowitz. Specifying and verifying hardware for tamper-resistant software. In *OAKLAND03*, May 2003. URL `https://security.csl.toronto.edu/wp-content/uploads/2018/06/lie-oakland2003.pdf`. (Acceptance: 19/131, 19%).

[23] Advanced Micro Devices. Vivado design suite user guide: Implementation (ug904). `https://docs.xilinx.com/r/en-US/ug904-vivado-implementation`, 2023. Accessed: 14/04/2023.

[24] Advanced Micro Devices. Ultrafast design methodology guide for xilinx fpgas and socs (ug949). `https://docs.xilinx.com/r/2021.1-English/ug949-vivado-design-methodology`, 2023. Accessed: 14/04/2023.

APPENDICES

APPENDIX A

Verilog Codes

A.1 D-flipflop

```
‘timescale 1ns / 1ps
 module d_flipflop(
      input clk,
      input clr,
      output reg Q);
 always @(posedge clk or posedge clr)
 begin
 if (clr==1’b1)
 Q<=1’b0;
 else
 Q<=1’b1;
 end
 endmodule
```

A.2 Multiplexer

```
‘timescale 1ns / 1ps
 module mux21(
      input a,
      input b,
      input sel,
```

```
     output out
);
assign out = (sel)?  a:b;
endmodule
```

A.3 SR-Latch

```
‘timescale 1ns / 1ps
module SR(
     input a,
     input b,
     output r
      );
wire temp1,temp2;
nand N1(temp1,a,temp2);
nand N2(temp2,b,temp1);
assign r=temp1;
endmodule
```

A.4 Pico-Puf

```
‘timescale 1ns / 1ps
     module picopuf(
     input clk,
     input clear,
     output o);
wire temp1,temp2;
wire w1;
(* dont_touch= "yes" *)d_flipflop d1(clk,clear,temp1);
(* dont_touch= "yes" *)d_flipflop d2(clk,clear,temp2);
```

```
SR SR1(temp1,temp2,o);
endmodule
```

A.5 FFAPUF slice1

```
‘timescale 1ns / 1ps
module ffapuf_slice_1(
          input clk,
          input clr,
          input [2:0]C,
          output r
          );
wire Y1,Y2,Y3,Y4;
wire Z1,Z2;
(* dont_touch= "yes" *)d_flipflop dff7(Z3,clr,r);
(* dont_touch= "yes" *)d_flipflop dff1(clk,clr,Y1);
(* dont_touch= "yes" *)d_flipflop dff2(clk,clr,Y2);
(* dont_touch= "yes" *)d_flipflop dff3(clk,clr,Y3);
(* dont_touch= "yes" *)d_flipflop dff4(clk,clr,Y4);
(* dont_touch= "yes" *)mux21 mux1(Y1,Y2,C[0],Z1);
(* dont_touch= "yes" *)mux21 mux2(Y3,Y4,C[1],Z2);
(* dont_touch= "yes" *)mux21 mux3(Z1,Z2,C[2],r);
endmodule
```

A.6 FFAPUF slice2

```
‘timescale 1ns / 1ps
module ffapuf_slice_2(
          input clk,
          input clr,
```

```
            input [1:0]C,
            output r
);
wire Y1,Y2,Y3,Y4;
wire Z1;
(* dont_touch= "yes" *)d_flipflop dff1(clk,clr,Y1);
(* dont_touch= "yes" *)d_flipflop dff2(clk,clr,Y2);
(* dont_touch= "yes" *)mux21 mux1(Y1,Y2,C[0],Z1);
(* dont_touch= "yes" *)d_flipflop dff3(Z1,clr,Y3);
(* dont_touch= "yes" *)d_flipflop dff4(Z1,clr,Y4);
(* dont_touch= "yes" *)mux21 mux2(Y3,Y4,C[1],r);
endmodule
```

A.7 FFAPUF line

```
‘timescale 1ns / 1ps
 module ffapuf_line(
            input clk,
            input clr,
            input [31:0]C,
            output r
           );
 wire clock[9:0];
 (* dont_touch= "yes" *)ffapuf_slice_1 ffs1_init(clk,clr,C[2:0],clock[0]);
 generate
 genvar i;
 for (i=2;i<11;i=i+1)
 begin
 (* dont_touch= "yes" *)ffapuf_slice_1 ffs1(clock[i-2],clr,C[3*i-1:3*i-3],
 clock[i-1]);
```

```
end
endgenerate
ffapuf_slice_2 ffs2(clock[9],clr,C[31:30],r);
endmodule
```

A.8 FFAPUF level

```
‘timescale 1ns / 1ps
module ffapuf_level(
          input clk,
          input clr,
          input [31:0]C,
          output r
          );
wire X1,X2;
(* dont_touch= "yes" *)ffapuf_line ffl_1(clk,clr,C[31:0],X1);
(* dont_touch= "yes" *)ffapuf_line ffl_2(clk,clr,C[31:0],X2);
SR SR1(X1,X2,r);
endmodule
```

A.9 FFAPUF

```
‘timescale 1ns / 1ps
module ffapuf(
          input clk,
          input clr,
          input [31:0]C,
          output [31:0]O
          );
generate
```

```
genvar i;
for (i=1;i<33;i=i+1)
begin
(* dont_touch = "yes" *)ffapuf_level ff_inst(clk,clr,C,O[i-1]);
end
endgenerate
endmodule
```

A.10 Basic unit of MPUF-t1

```
‘timescale 1ns / 1ps
module challenge_bits_t1(
              input clk,
              input clear,
              input [2:0]C,
              output [2:0]C_bar

              );
wire w1,w2; wire z1,z2;
(* dont_touch= "yes" *)pico_puf p1(clk,clear,w1);
(* dont_touch= "yes" *)pico_puf p2(clk,clear,w2);
(* dont_touch= "yes" *)mux21 m1(w1,w2,w1,z1);
(* dont_touch= "yes" *)mux21 m2(w2,w1,w1,z2);
(* dont_touch= "yes" *)assign C_bar[0]=z1Ĉ[0];
(* dont_touch= "yes" *)assign C_bar[1]=z1Ĉ[1];
(* dont_touch= "yes" *)assign C_bar[2]=z2Ĉ[2];
endmodule
```

A.11 MPUF-t1

```
‘timescale 1ns / 1ps
 module technique_t1(
              input clk,
              input clear,
              input [29:0]C,
              output [29:0]Cbar
              );
 wire a,b;
 (* dont_touch= "yes" *)challenge_bits_t1 cb_t1(clk,clear,C[2:0],Cbar[2:0]);
 generate
 genvar i;
 for (i=2;i<11;i=i+1)
 begin
 (* dont_touch= "yes" *)challenge_bits_t1 cb_t1_inst(clk,clear,C[3*i-1:3*i-3],
                                                    Cbar[3*i-1:3*i-3]);
 end
 endgenerate
 endmodule


 module mpuf_t1(
          input clk,
          input clear,
          input clr,
          input [31:0]C,
          output [31:0]O
 );
 generate
```

```
genvar i;
for (i=1;i<33;i=i+1)
begin
wire [31:0]C_bar;
assign C_bar[30]=C[30];
assign C_bar[31]=C[31];
(* dont_touch = "yes" *)technique_t1 t1_inst(clk,clear,C[29:0],C_bar[29:0]);
(* dont_touch = "yes" *)ffapuf_level ff_inst(clk,clr,C_bar[31:0],O[i-1]);
end
endgenerate
endmodule
```

A.12 Basic unit of MPUF-t2

```
‘timescale 1ns / 1ps
module challenge_bits_t2(
              input clk,
              input clear,
              input [2:0]C,
              output Cbm
              );
wire w1,w2;
wire z1,z2;
(* dont_touch= "yes" *)pico_puf p1(clk,clear,w1);
(* dont_touch= "yes" *)pico_puf p2(clk,clear,w2);
(* dont_touch= "yes" *)mux21 m1(w1,w2,w1,z1);
(* dont_touch= "yes" *)mux21 m2(w2,w1,w1,z2);
assign Cbm=z1ẑ2Ĉ[0]Ĉ[1]Ĉ[2];
endmodule
```

A.13 MPUF-t2

```
‘timescale 1ns / 1ps
 module technique_t2(
              input clk,
              input clear,
              input [29:0]C,
              output [9:0]Cbm
              );
 wire a,b;
 (* dont_touch= "yes" *)challenge_bits_t2 cb_t2(clk,clear,C[2:0],Cbm[0]);
 generate
 genvar i;
 for (i=2;i<11;i=i+1)
 begin
 (* dont_touch= "yes" *)challenge_bits_t2 cb_t2_inst(clk,clear,C[3*i-1:3*i-3],
                                                    Cbm[i-1]);
 end
 endgenerate
 endmodule


 module mpuf_t2(
          input clk,
          input clear,
          input clr,
          input [31:0]C,
          output [31:0]O
          );
 generate
```

```
genvar i;
for (i=1;i<33;i=i+1)
begin
wire [11:0]C_bar;
assign C_bar[10]=C[30];
assign C_bar[11]=C[31];
(* dont_touch = "yes" *)technique_t2 t2_inst(clk,clear,C[29:0],C_bar[9:0]);
(* dont_touch = "yes" *)ffapuf_level ff_inst(clk,clr,C_bar[11:0],O[i-1]);
end
endgenerate
endmodule
```

VITA AUCTORIS

NAME:	Sai Preetham Bonagiri
PLACE OF BIRTH:	Nizamabad, Telangana, India
YEAR OF BIRTH:	1999
EDUCATION:	International Institute of Information Technology (IIIT) Bhubaneswar, India, 2016-2020
	Bachelor of Technology (Electronics and Telecommunications)