

DESIGN AND DEVELOPMENT OF PUF FOR HARDWARE SECURITY UPDATE

PROJECT REPORT

Submitted by

NAVABHARATHI T	2022504526
DHARSHAN R	2022504517
LAKSHMANAN S	2022504310

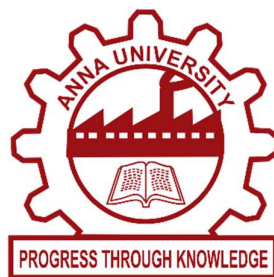
In partial fulfilment for the award of the degree

of

BACHELOR OF ENGINEERING

IN

ELECTRONICS AND COMMUNICATION ENGINEERING



DEPARTMENT OF ELECTRONICS ENGINEERING

MADRAS INSTITUTE OF TECHNOLOGY

ANNA UNIVERSITY: CHENNAI – 600 044

NOVEMBER 2025

BONAFIDE CERTIFICATE

Certified that this project report title “**DESIGN AND DEVELOPMENT OF PUF FOR HARDWARE SECURITY UPDATE**”, is the bonafide work of

NAVABHARATHI T 2022504526

DHARSHAN R 2022504517

LAKSHMANAN S 2022504310

who carried out the project under my Supervision.

SIGNATURE

Dr. P.T.V. BHUVANESHWARI

HEAD OF THE DEPARTMENT

Professor

Department of Electronics Engineering,

Madras Institute of Technology,

Anna University,

Chennai – 600 044.

SIGNATURE

Dr. V. R. VIJAYKUMAR

SUPERVISOR

Professor

Department of Electronics Engineering,

Madras Institute of Technology,

Anna University,

Chennai – 600 044.

ACKNOWLEDEMENT

We consider it as our privilege and our primary duty to express our gratitude and respect to all those who guided, helped and inspired us in the successful completion of the project.

We owe solemn gratitude to **Dr. P. JAYASHREE**, Dean, Madras Institute of Technology, for having given consent to carry out the project work.

We owe sincere gratitude to **Dr. P.T.V. BHUVANESHWARI**, Professor and Head of the Department of Electronics Engineering, who has encouraged and motivated us in our endeavours.

We are extremely grateful to our project guide **Dr. V. R. VIJAYKUMAR**, Professor, Department of Electronics Engineering for his timely and thoughtful guidance and encouragement for the completion of the project.

We wish to extend our sincere thanks to the review panel faculty members **Dr. D. MEGANATHAN** and **Dr. O. VIGNESH** for their valuable suggestions during all the reviews which took our project to greater heights.

We would like to thank our project coordinators **Dr. M. VASIM BABU** and **Mrs. B. MEHAR NISHA BEGAM** and all the teaching and non-teaching staff members of Department of Electronics Engineering, for their support in all the aspects.

Place: CHENNAI	NAVABHARATHI T	2022504526
Date:	DHARSHAN R	2022504517
	LAKSHMANAN S	2022504310

ABSTRACT

This project focuses on the development, simulation, and realization of an Arbiter-based Physical Unclonable Function (PUF) architecture aimed at providing secure hardware authentication in resource-limited electronic systems. The proposed work focuses on developing novel hybrid PUF structures that enhance key performance parameters such as uniqueness, reliability, and uniformity while maintaining low hardware complexity. The system leverages inherent manufacturing variations within integrated circuits to produce distinct and non-replicable digital fingerprints, thereby providing device-level security without relying on stored cryptographic keys. The proposed hybrid architecture combines XOR and separate configurations to achieve improved entropy, randomness, and balance between performance metrics.

A comparative analysis of the proposed architecture with existing Arbiter PUF variants is carried out to evaluate improvements in security and efficiency. The hardware-based authentication protocol implemented in this work validates the practical usability of the design for secure identity verification. The system demonstrates consistent and reproducible challenge–response behavior, even under moderate environmental variations. The proposed design is simulated in Xilinx Vivado and implemented on the Basys-3 FPGA board to evaluate its real-time hardware performance and effectiveness.

The results show that proposed hybrid PUF achieves higher reliability and uniqueness compared to conventional designs. By integrating lightweight design, robustness, and adaptability, this work contributes to energy-efficient, tamper-resistant, and scalable hardware security solutions for next-generation IoT and embedded systems.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	iv
1.	INTRODUCTION	1-7
	1.1 OVERVIEW	1
	1.2 WHAT IS PUF	1
	1.3 HOW PUFs WORK	2
	1.4 TYPES OF PUFs	2
	1.5 PROPERTIES OF PUFs	3
	1.6 APPLICATIONS OF PUFs	4
	1.7 CHALLENGES AND CONSIDERATIONS	5
2	LITERATURE SURVEY	8-11
3	DESIGN OF ARBITER PUF	12-18
	3.1 ARBITER PUF	12
	3.2 XOR PUF	14
	3.2.1 WORKING PRINCIPLE	14
	3.3 MUX and DEMUX ARBITER PUF	16
	3.3.1 WORKING PRINCIPLE	17
4	PROPOSED ARCHITECTURES	19-24

	4.1 16-BIT XOR PUF	20
	4.2 PROPOSED ARCHITECTURE-2	22
	4.2.3 OUTPUT	23
5	PERFORMANCE ANALYSIS	25-34
	5.1 UNIFORMITY	25
	5.2 UNIQUENESS	26
	5.3 RANDOMNESS	26
	5.4 RELIABILITY	26
	5.5 RESULTS	27
	5.6 AUTHENTICATION	29
	5.6.1 SYSTEM ARCHITECTURE	29
	5.6.2 ENROLLMENT PHASE	30
	5.6.3 AUTHENTICATION PHASE	31
	5.6.4 CHOICE OF ARBITER PUF	32
6	ASIC SYNTHESIS AND IMPLEMENTATION METHODOLOGY	32-38
	6.1 INTRODUCTION TO APPLICATION- SPECIFIC INTEGRATED CIRCUITS (ASICS)	32
	6.1.1 FROM CONCEPT TO SILICON	32
	6.1.2 A COMPARATIVE ANALYSIS: ASIC VS. FPGA FOR PUF IMPLEMENTATION	33
	6.2 SYNTHESIS ENVIRONMENT AND TECHNOLOGY LIBRARIES	35

	6.2.1 SYNTHESIS TOOLCHAIN: CADENCE GENUS SYNTHESIS SOLUTION	35
	6.2.2 TARGET TECHNOLOGY NODE: A 180NM CMOS PROCESS	35
	6.2.3 THE FOUNDATION: STANDARD CELL AND I/O PAD LIBRARIES	36
	6.3 SYNTHESIS SCRIPTING AND AUTOMATION	37
7	CONSTRAINT-DRIVEN LOGIC SYNTHESIS OF THE 8-BIT ARBITER PUF	39-46
	7.1 RTL DESIGN FOR SYNTHESIS: A STRUCTURAL REVIEW	39
	7.2 DECONSTRUCTING THE SYNTHESIS PROCESS:	42
	7.3 TIMING CONSTRAINTS	45
	7.3.1 INTRODUCTION TO STATIC TIMING ANALYSIS (STA)	45
	7.3.2 LINE-BY-LINE BREAKDOWN OF PUF_TIME.SDC	46
8	POST-SYNTHESIS VERIFICATION AND RESULT ANALYSIS	47-54
	8.1 THE PRODUCTS OF SYNTHESIS: NETLIST AND DELAY INFORMATION	47
	8.1.1 THE GATE-LEVEL NETLIST	47
	8.1.2 THE STANDARD DELAY FORMAT FILE	48

	8.2 IN-DEPTH ANALYSIS OF SYNTHESIS REPORTS	48
	8.2.1 QUALITY OF SERVICE (QOS) SUMMARY (FINAL.RPT)	48
	8.2.2 AREA REPORT ANALYSIS	50
	8.2.3 POWER REPORT ANALYSIS	52
	8.3 POST-SYNTHESIS VERIFICATION	53
	8.3.1 FUNCTIONAL AND TIMING VERIFICATION	53
	8.3.2 NEXT STEPS: HAND-OFF TO PHYSICAL DESIGN	54
9	CONCLUSION AND FUTURE WORKS	55-58
	9.1 SUMMARY OF RESEARCH AND CONTRIBUTIONS	55
	9.2 LIMITATIONS OF THE CURRENT WORK	56
	9.3 SCOPE FOR FUTURE WORK	57
10	REFERENCES	59-61

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

A Physical Unclonable Function (PUF) is a security element implemented in hardware that generates a unique identity for each device by leveraging minute, inherent variations that occur during the fabrication of semiconductors. Since these manufacturing imperfections cannot be intentionally replicated, every circuit possesses a distinctive and unpredictable character. When the PUF receives an input, known as a challenge, it generates a corresponding response that functions as the device's unique digital signature. This capability allows for secure authentication, identification, and the dynamic creation of cryptographic keys, eliminating the risk associated with storing secrets in non-volatile memory.

1.2 What is PUF?

A **Physically Unclonable Function (PUF)** is a core physical component built into a hardware system. Its purpose is to create a specific and randomized response for every input (challenge). This output is directly defined by the chip's particular physical attributes, which originate from variations introduced during the manufacturing process. These characteristics originate from uncontrollable manufacturing variations (such as microscopic imperfections in the silicon structure) introduced during the fabrication process, rendering each PUF instance uniquely individual and virtually impossible to duplicate or clone. This intrinsic, device-specific, and highly stable uniqueness allows PUFs to reliably serve as robust hardware "fingerprints," establishing a crucial, trustable foundation for secure identification, authentication, and secure key storage within modern

electronic systems, especially for Internet of Things (IoT) devices that operate under limited hardware and power resources.

1.3 How PUFs Work?

Physical Unclonable Functions (PUFs) operate by capitalizing on the minute, spontaneous variations that arise during the production of Integrated Circuits (ICs). These fundamental manufacturing differences impact key physical attributes within the silicon, such as signal propagation delays, capacitances, and transistor threshold voltages. When a specific input, called a challenge, is fed into the PUF circuit, these unique internal characteristics become the decisive factor influencing the circuit's behavior. This process, in turn, yields a unique corresponding output, or response. This reliable challenge-response protocol forms the basis for using PUFs in security. Specifically, while a single PUF device consistently delivers the same response to a repeated challenge, a different PUF receiving the identical challenge will produce a distinct response due to its singular physical properties.

1.4 Types of PUFs

Physical Unclonable Functions (PUFs) are generally classified according to their architecture and how they implement the challenge-response process:

Silicon PUFs: These are implemented within semiconductor devices and exploit manufacturing variations in silicon ICs. Common types include:

- **Arbiter PUFs:** Utilize race conditions between signal paths to generate unique responses.
- **Ring Oscillator PUFs:** Rely on frequency variations in ring oscillators caused by manufacturing differences.

- **SRAM PUFs:** This category capitalizes on the power-up condition of SRAM cells that haven't been initialized, a state that is directly affected by fluctuations in threshold voltage.
- **Optical PUFs:** Use the unique scattering properties of a physical medium, such as a transparent material with randomly distributed particles, to produce a distinctive optical response when illuminated.

1.5 Properties of PUFs

Several key properties define the effectiveness and applicability of PUFs in security systems:

Uniqueness: Each PUF should produce a distinct set of responses, ensuring that no two PUFs are identical.

Reproducibility: A PUF needs to consistently generate the identical response when given a specific challenge, provided the operational environment remains unchanged.

Unclonability: Due to the inherent randomness introduced during manufacturing, replicating a PUF's exact physical characteristics is virtually impossible, even with identical design specifications.

Tamper Resistance: PUFs are resistant to invasive attacks, as any attempt to physically probe or alter the PUF can change its physical properties, thereby altering its responses and rendering it unusable.

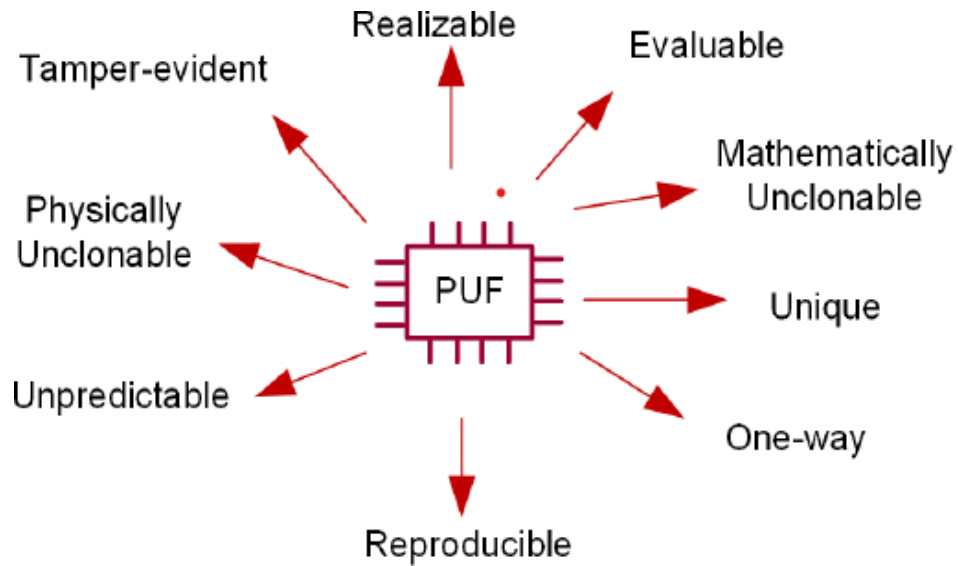


Figure 1.1 Properties of PUF

1.6 Applications of PUFs

PUFs are widely used in various areas of hardware security, including:

Device Authentication: PUFs can verify the identity of hardware devices by generating unique responses that serve as device fingerprints.

Key Generation and Storage: PUFs are capable of dynamically generating cryptographic keys, removing the need to store them in non-volatile memory, a location that is often susceptible to extraction attacks.

Secure Communications: PUFs enable the establishment of secure channels between devices by providing unique keys used in the processes of encryption and decryption.

Internet of Things (IoT) Security: In IoT networks, PUFs help secure communication between low-power devices without heavy cryptographic storage. Each IoT node has a unique PUF identity for secure key exchange and authentication.

1.7 Challenges and Considerations

While PUFs offer promising security benefits, several challenges must be addressed:

Environmental Sensitivity: The responses generated by PUFs can fluctuate because of environmental factors, such as shifts in supply voltage and temperature. Consequently, the integration of error correction methods is essential for sustaining both consistent and reliable output signals.

Modeling Attacks: Advanced machine learning techniques can potentially model PUF characteristics inferred from the analysis of collected challenge-response pairs. Designing PUFs with a large and complex challenge-response space can mitigate this risk.

Limited Uniqueness: Some PUFs may generate similar responses for different devices, reducing their ability to act as a unique fingerprint. This low uniqueness can cause confusion in authentication and lower the overall security strength of the system.

Aging and Reliability: Over time, the physical properties of a PUF may change due to aging effects, potentially impacting its reliability. Continuous monitoring and recalibration may be necessary to maintain performance.

In conclusion, Physically Unclonable Functions represent a significant advancement in hardware security, providing unique, tamper-resistant identifiers for electronic devices.

The primary goal of this project is multi-faceted, beginning with the examination and analysis of fundamental arbiter PUF architectures with respect to core parameters like uniqueness, uniformity, and reliability. Following this, the project aims to identify the specific limitations of existing PUF designs, particularly focusing on managing the trade-offs between response randomness and reliability in the presence of varying environmental conditions. Based on these findings, the project seeks to design and implement a novel hybrid PUF architecture that combines separate and XOR configurations to enhance output uniqueness and uniformity. A further key objective is to propose a modified MUX-DEMUX reconfigurable arbiter PUF that incorporates dynamic path configuration to achieve superior delay control and increased reliability, which will then be integrated within a heterogeneous XOR framework. Finally, the most reliable resulting architecture will be implemented within an authentication use-case to practically demonstrate its applicability in secure hardware systems.

The organization of the rest of the report is as follows: Chapter 2 is dedicated to a review of the relevant literature and existing research, with the aim of highlighting potential areas where the Arbiter PUF design can be further enhanced and improved. Chapter 3 presents the implementation of some existing Arbiter PUF architectures and their variants from the literature survey. In Chapter 4, two novel architectures based on the implemented existing architectures to improve its performance are developed and analyzed. In Chapter 5, the performance is evaluated by its metrics for all the implemented designs and a simple authentication protocol is implemented to assess its performance.

CHAPTER 2

LITERATURE SURVEY

Hemavathy and Bhaaskaran (2023) presented an extensive study on Arbiter Physical Unclonable Functions (APUFs), focusing on their design, composition, and security aspects. Arbiter Physical Unclonable Functions (APUFs) are hardware security components that exploit intrinsic manufacturing variations within integrated circuits (ICs) to generate responses that are both unique and unpredictable, which makes them highly useful for both cryptographic and authentication purposes. The authors discuss various APUF architectures, including feed-forward and XOR-based designs, and evaluate their performance on ZedBoard hardware for 16, 32, and 64-bit responses. They also examine the vulnerabilities of APUFs to machine learning attacks and propose design enhancements to improve resistance against such threats. The study emphasizes the usefulness of APUFs in protecting resource-limited devices within the Internet of Things (IoT) environment, such as RFID tags and smart cards, while also recognizing the necessity for continued research to overcome current security limitations.

Farha et al. (2021) introduced a reliable and lightweight authentication scheme. This method utilizes Static Random Access Memory (SRAM) Physical Unclonable Functions (PUFs) and was designed specifically to improve the security of resource-constrained Internet of Things (IoT) devices. Recognizing that traditional cryptographic methods impose significant computational overhead and are vulnerable to physical attacks that expose secrets stored in nonvolatile memory, the authors propose exploiting the SRAM cells' startup values (SVRCs) as a tamper-resistant hardware fingerprint. They employ challenge-response pairs (CRPs) in which the challenges are represented by rearranged memory addresses. To overcome the low entropy problem commonly found in weak SRAM-PUFs, a hashing function is applied to the reordered PUF

outputs, allowing the generation of a large number of CRPs with minimal overhead and limited helper data storage. In addition, the authors analyze the PUF's performance under various environmental factors such as temperature and magnetic fields and integrate an Error Correction Coding (ECC) algorithm to maintain the reliability of the generated fingerprint. Through their experiments, they demonstrate that the proposed method achieves efficient authentication with low computational effort and memory usage, confirming its practical effectiveness for protecting resource-constrained IoT devices.

In 2023, Plusquellic et al. introduced two mutual authentication protocols designed to preserve privacy for resource-limited Internet of Things (IoT) devices. These protocols rely on the shift-register reconvergent-fanout (SiRF) PUF, recognized as a strong Physical Unclonable Function. Acknowledging that traditional methods like public-private key infrastructure are impractical for low-power IoT, and symmetric cryptography is vulnerable to physical attacks in unsupervised environments, the authors propose rooting security in the SiRF PUF hardware. The SiRF PUF removes the need to store secrets and provides very large sets of authentication credentials. The protocols, COBRA and PARCE, inherit the SiRF PUF's security properties. COBRA prevents Model-Building (MB) attacks by using only helper data bitstrings, whereas PARCE employs an innovative PUF-based encryption approach known as Secure-Key-Encoding (SKE). The two authentication protocols underwent testing and verification via hardware experiments. These were carried out on SoC-based FPGA platforms while operating under typical industrial conditions.

Manivannan et al. (2024) propose an efficient, low-complexity and forward-secure Authentication and Key Agreement (AKA) protocol designed to establish secure communication within a cluster of resource-constrained Internet of Things (IoT) devices. The protocol incorporates a Physical Unclonable Function (PUF) within each device, serving as a hardware-based root of trust. A

Fuzzy Extractor (FE) corrects the PUF's noisy responses, ensuring the reliable generation and reproduction of private keys and public helper data. Forward security in the AKA protocol is achieved through the Elliptic Curve Diffie-Hellman (ECDH) method, which deliberately avoids complex pairing operations to minimize computational and resource overhead. Notably, the confidential PUF response is never stored on the server. The proposed scheme is formally validated against both active and passive attacks using the Verifpal tool, and its practical efficiency is demonstrated through a prototype implementation on an FPGA employing the Arbiter PUF (APUF) design.

Hemavathy and Bhaaskaran (2021) presented a secure quasi-adiabatic tristate Physical Unclonable Function (PUF). This design was specifically engineered for energy-efficient hardware security applications, such as those found in RFID and IoT systems. This PUF leverages inherent threshold voltage variations in PMOS transistors, arising from manufacturing inconsistencies, to generate unique device identifiers. The quasi-adiabatic logic design significantly reduces power usage that is lower than that of traditional CMOS-based PUFs, maintaining performance across varying temperatures and supply voltages. Experimental results demonstrate enhanced uniqueness (50.27%), uniformity (49%), and reliability (99.82%), indicating robust resistance to differential power analysis attacks. The 128-bit response output makes it particularly suitable for low-power, secure identification in resource-constrained environments.

Zhou (2020) introduces an FPGA-oriented design methodology for an adjustable Arbiter Physical Unclonable Function (APUF), aiming to enhance hardware security in embedded systems. The proposed design allows for configurability in the arbiter circuit, enabling adjustments to delay paths to improve uniqueness and reliability. This adaptability addresses challenges associated with environmental variations and manufacturing inconsistencies. By implementing the design on FPGA platforms, the study demonstrates its

practicality for real-world applications, particularly in resource-constrained IoT devices. The adjustable APUF exhibits improved resistance to modeling attacks, thereby making it a potential solution for secure device authentication and cryptographic key generation. This research supports the advancement of reliable, adaptable, and efficient PUF architectures that meet the demands of contemporary hardware security applications.

Zhang and Shen (2021) propose a set-based obfuscation method to enhance the security of strong Physical Unclonable Functions (PUFs) against machine learning (ML) attacks. Their approach, called Random Set-based Obfuscation (RSO), it works by pre-storing a group of stable responses generated from the PUF. During the authentication process, a true random number generator picks two keys from this set to conceal both the challenge and the response through XOR operations. This dynamic obfuscation strategy significantly increases the difficulty for attackers attempting to model the PUF behavior using ML techniques. Results from experiments using a 64×64 Arbiter PUF indicated that the prediction accuracy of various machine learning models—including logistic regression, support vector machines, artificial neural networks, convolutional neural networks, and covariance matrix adaptive evolutionary strategies—decreases to approximately 50%, equivalent to a random guess. This outcome holds true even when one million challenge-response pairs (CRPs) are gathered. The use of the Random Set-based Obfuscation (RSO) technique offers this enhanced security while requiring only a minimal hardware investment, thus proving it to be a practical and effective method for securing resource-constrained devices in Internet of Things (IoT) environments.

He et al. (2020) proposed a highly reliable Arbiter Physical Unclonable Function (A-PUF) architecture optimized for FPGA implementation, which employs a Bit-Self-Test (BST) technique to improve performance while minimizing hardware resource usage. In this approach, a delay monitoring circuit

is embedded within the A-PUF to track real-time delay variations during response generation. If the detected deviation surpasses a specified threshold, the corresponding response is considered unreliable and discarded, ensuring that only stable outputs are retained. This technique enhances the reliability of the A-PUF without relying on conventional error correction codes (ECCs) that typically need non-volatile memory for storing helper data. When deployed on a Xilinx Artix-7 FPGA, the Bit-Self-Test Arbiter PUF (BST-A-PUF) demonstrated high effectiveness for lightweight cryptographic uses within resource-limited systems. Its performance metrics included a bit error rate below 10^{-9} , a bias of 50.3%, and a uniqueness value of 49.1%.

Wisiol et al. (2021) conducted a re-examination and enhancement of neural network-based modeling attacks targeting XOR Arbiter Physical Unclonable Functions (PUFs). They successfully demonstrated the high efficacy of these attacks against several architectures, including XOR, Feed-Forward Arbiter, and Interpose PUFs. By leveraging the Keras machine learning framework, the investigators were able to achieve predictions that were both quicker and more precise, utilizing a smaller dataset of challenge-response pairs compared to conventional methods like logistic regression. This research underscores the necessity of making neural network-based attack models a mandatory component of any PUF security evaluation.

This literature survey explores various architectures and enhancements of Physical Unclonable Functions (PUFs), particularly Arbiter PUFs, highlighting their effectiveness for secure, lightweight authentication in IoT systems. It emphasizes design innovations—like feed-forward, XOR, tristate, and configurable PUFs—to improve resistance against modeling and machine learning attacks. The survey also addresses reliability, energy efficiency, and environmental robustness, underscoring ongoing challenges and the possibilities for further research and development in the field of secure hardware design.

CHAPTER 3

DESIGN OF ARBITER PUF

A Physical Unclonable Function (PUF) acts as a hardware-level security tool, leveraging the spontaneous and unavoidable variations introduced during the fabrication of integrated circuits to produce responses that are both unique and unpredictable when provided with specific inputs, known as challenges. Because these variations are virtually impossible to replicate, they result in a "digital fingerprint" specific to each device, making PUFs essential for secure authentication and identification. Unlike conventional cryptographic techniques that depend on keys kept in storage, PUFs create keys dynamically, which increases their resilience against physical attacks and reverse engineering. This inherent, on-demand generation of unique keys makes PUFs an ideal solution for applications demanding robust hardware security, such as those found in secure communication links and Internet of Things (IoT) systems.

3.1 ARBITER PUF

The Arbiter Physical Unclonable Function (Arbiter PUF) represents one of the most basic and extensively studied forms of PUFs, known for its simple architecture and significant role in hardware-based security systems.

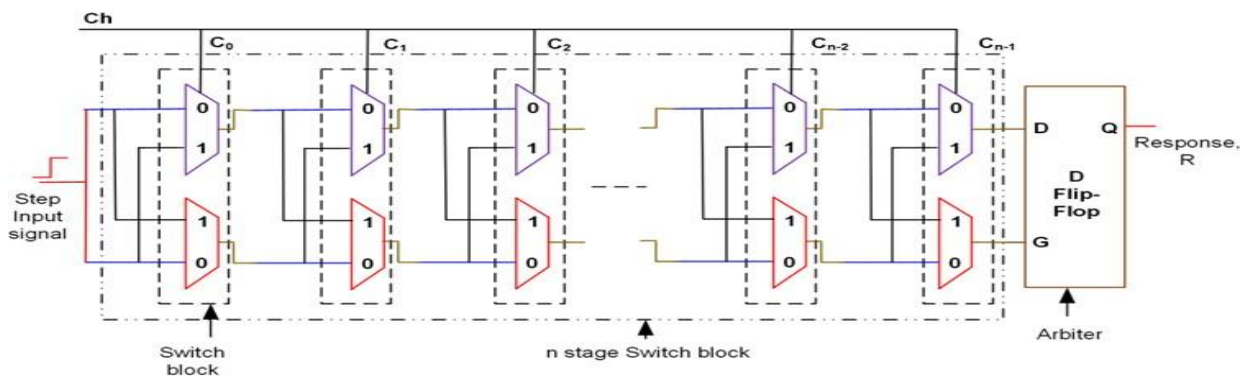
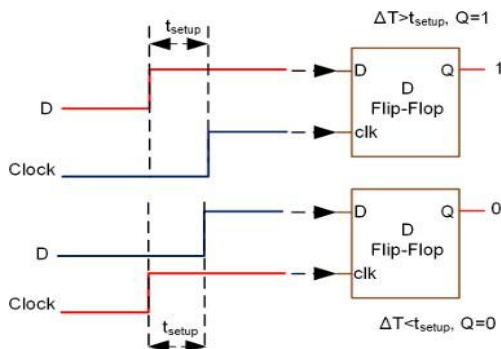


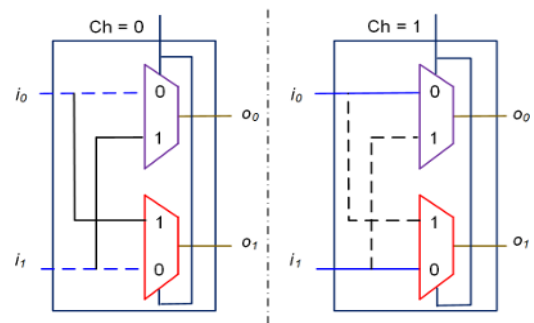
Figure 3.1 Arbiter PUF

Arbiter PUFs are counted among the first silicon-based Physical Unclonable Function (PUF) architectures and are classified as strong PUFs because they are capable of producing an enormous quantity of distinct challenge-response pairs (CRPs). The fundamental principle of their operation is to leverage the random and inherent variations that happen during the production of integrated circuits (ICs) to establish a unique digital identifier for each specific chip. These minute differences in manufacturing lead to fractional discrepancies in the propagation delays of signals traveling through otherwise identical circuits on separate chips. The Arbiter PUF is designed to detect, capture, and convert these timing variations into a digital output.

Typically, an Arbiter PUF is structured with a cascade of n switch blocks. Within each switch block, there is a pair of 2-to-1 multiplexers, which create two distinct pathways for two signals to race against one another. A binary challenge vector of length n determines how the signals are routed through the switch blocks. The paths influenced by this challenge vector create two signal paths with distinct delays. These signals then arrive at a final circuit element, called the arbiter, which decides which signal came first. The arbiter produces a binary output (0 or 1), forming the response to the challenge. The randomness in IC delay characteristics ensures the output is unique and difficult to replicate.



**Figure 3.2 Switch Block for
CH=0 & CH=1**



**Figure 3.3 Response Determined
by the Signal Transition in D-FF**

3.2 XOR PUF

The XOR Arbiter PUF is a more advanced version of the basic Arbiter Physical Unclonable Function (PUF), designed to improve security and resistance to modeling attacks. It builds on the same core concept of delay-based signal comparison but adds complexity and non-linearity by combining multiple Arbiter PUFs through XOR operations. This approach makes it significantly harder for adversaries to reverse-engineer or clone the PUF using machine learning techniques.

While the standard Arbiter PUF is both straightforward and efficient, its inherent linear design makes it susceptible to machine learning attacks. Attackers can frequently predict its responses by creating a model of the signal delay paths, especially when they have acquired a large dataset of challenge-response pairs (CRPs). To mitigate this vulnerability, researchers created the XOR Arbiter PUF. This advanced architecture introduces non-linearity into the response generation process by combining the individual outputs of multiple, separate arbiter chains using XOR operations. This resulting complexity enhances the PUF's resistance to modeling attacks and substantially boosts the randomness and unpredictability of its final outputs.

3.2.1 WORKING PRINCIPLE

XOR Arbiter PUF consists of multiple arbiter chains, each receiving the same n -bit challenge vector. Each chain processes the challenge through its own delay path composed of multiplexers and outputs a 1-bit response based on which signal (upper or lower path) arrives first at the arbiter. These individual 1-bit responses from k arbiter PUFs are then XOR-ed together to produce a single output bit. This final response is used for authentication or cryptographic key generation.

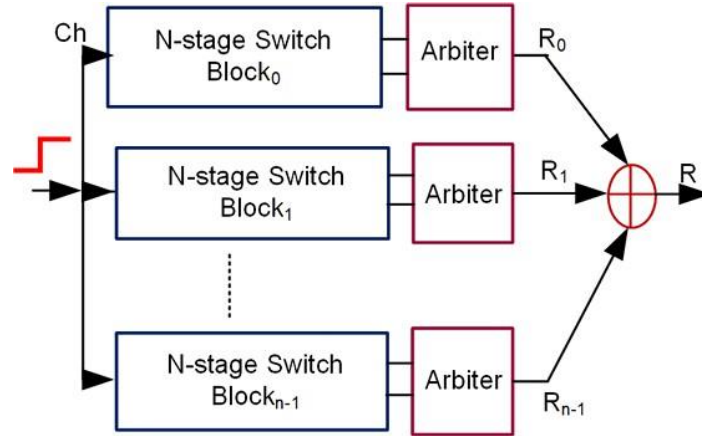


Figure 3.6 XOR PUF

The XOR operation ensures that even if some of the individual PUFs are biased or predictable, the final output becomes much more difficult to model, as the output now represents a non-linear combination of multiple delay behaviors.

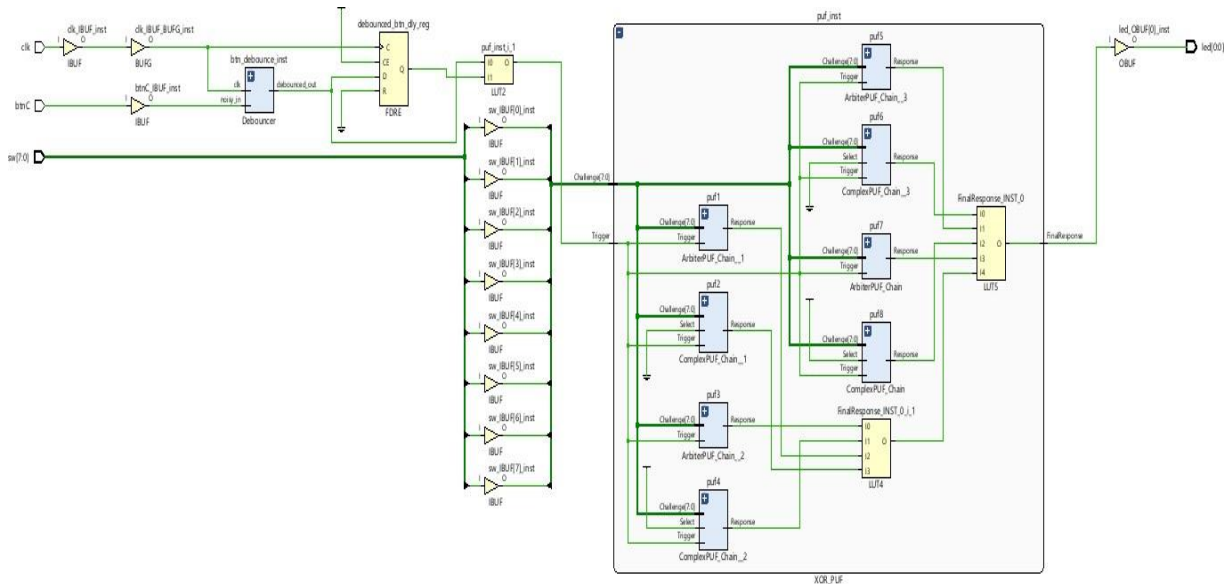


Figure 3.7 Schematic for XOR PUF

3.3 MUX and DEMUX ARBITER PUF

A Physical Unclonable Function (PUF) that relies on a MUX-DEMUX architecture serves as a hardware security feature. It uses the inherent variations arising during manufacturing to generate outputs that are both unique and unpredictable. These PUFs consist of a chain of multiplexers (MUXes) and demultiplexers (DEMUXes), which form a signal path network controlled by an input known as the challenge. The challenge bits determine how the signal propagates through the MUX-DEMUX network, selecting specific paths from multiple possible ones. Each chip exhibits slight differences in delay across the signal paths due to unavoidable variations during fabrication, and this randomness forms the core of the PUF's uniqueness.

In operation, the same input signal is split and races through two paths defined by the challenge bits. These paths are built using cascaded MUX-DEMUX stages, and due to the physical differences in each chip, even identical challenge bits will result in different delays from one chip to another. At the end of the network, a D flip-flop (or a latch) captures which signal arrives first. The winner of this "race" determines the output bit (0 or 1), known as the response.

The strength of MUX-DEMUX based PUFs lies in their simplicity, compact design, and low power requirements. Because the unique response is generated from the natural delay variations within the chip's signal paths, there is no requirement to store any secret information in memory, which makes it naturally resistant to many forms of physical attacks. Additionally, these PUFs can be employed to create cryptographic keys or verify hardware authenticity without depending on a trusted third party.

In applications like device authentication and secure key generation, MUX-DEMUX PUFs provide a lightweight and reliable solution. Their ability to

produce repeatable yet unique responses for a given challenge ensures high security while maintaining efficiency. As such, they are increasingly being adopted in embedded systems, IoT devices, and secure hardware design.

3.3.1 WORKING PRINCIPLE

The working principle of a MUX-DEMUX based Physical Unclonable Function (PUF) relies on exploiting the inherent delay variations in signal paths caused by manufacturing differences in integrated circuits. When a binary input challenge is provided to the PUF, it configures a chain of multiplexers (MUXes) and demultiplexers (DEMUXes) to create two parallel paths for an input signal. These paths are selected based on the challenge bits, which determine how the signal will travel through the network of logic gates. Even though the paths are logically identical, the actual physical delays differ slightly from chip to chip due to uncontrollable variations in fabrication.

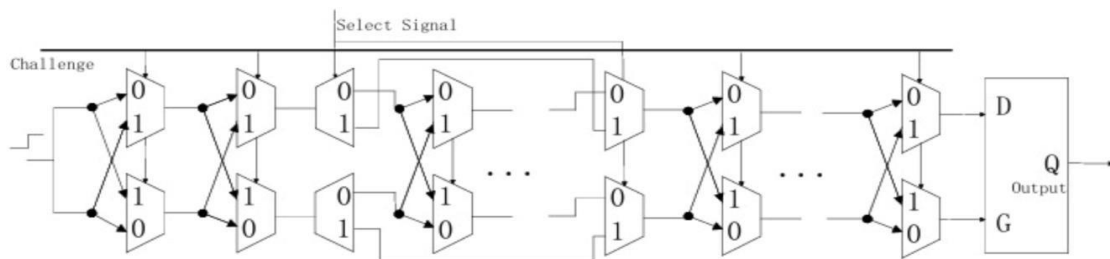


Figure 3.8 MUX and DEMUX Arbiter PUF

An input signal, typically a clock pulse, is split into two identical copies that race through the selected paths simultaneously. As the signals travel, the physical delay in each path causes one signal to arrive at the output slightly earlier than the other. At the end of the circuit, a latch or D flip-flop captures the first arriving signal, and the output is set based on which path was faster. The result is

a single-bit response that depends on the unique internal delay characteristics of the chip for the given challenge. Repeating this process with different challenges generates a set of unique and reproducible response bits that serve as a digital fingerprint for the device.

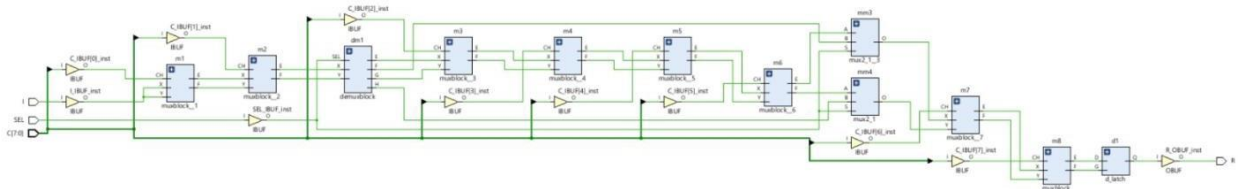


Figure 3.9 Schematic for MUX and DEMUX Arbiter PUF

CHAPTER 4

PROPOSED ARCHITECTURES

To boost the reliability, security, and practicality of systems built on Physical Unclonable Functions (PUFs), the proposed design presents a robust yet streamlined architecture that addresses critical shortcomings in previous methods. Capitalizing on the intrinsic unpredictability and uniqueness of PUFs, this design integrates optimized control logic, lightweight authentication protocols, and efficient communication interfaces to facilitate real-time, secure device authentication.

In contrast to standard implementations that can be hampered by environmental sensitivity or require cumbersome post-processing, this new architecture prioritizes stability, low-latency response generation, and easy integration with external validation systems, such as host computers or microcontrollers, typically utilizing UART communication. This modular and scalable approach not only simplifies deployment across diverse platforms but also guarantees stable response behavior, positioning it as an excellent choice for embedded systems, particularly in Internet of Things (IoT) and edge computing environments.

By expertly linking a meticulously planned challenge-response interface with minimal hardware resource consumption and improved resilience against errors, this architecture achieves an optimal combination of security, performance, and power efficiency. This balance establishes it as a compelling solution for the next generation of hardware security, which requires adaptability and trustworthiness in dynamic operating settings.

4.1 16 BIT XOR PUF

In the pursuit of improving the reliability, scalability, and practical usability of Arbiter-based Physical Unclonable Functions (PUFs), the proposed design introduces a 16-bit XOR-based Arbiter PUF integrated with a UART communication interface. This architecture not only strengthens the randomness and uniqueness properties of the PUF responses but also enhances real-time data communication between the FPGA and external devices, such as a personal computer or microcontroller, for efficient challenge–response verification.

The proposed system leverages a 16-bit challenge input that propagates through multiple *ArbiterStage* modules. Each stage introduces controlled routing delays based on the input challenge bits, thereby generating distinct signal race conditions. The final signal pair is evaluated by an *ArbiterCell* to determine the output response bit. This modular structure allows precise analysis and tuning of each stage’s delay characteristics, ensuring improved design flexibility and scalability.

To further enhance entropy, the architecture employs eight parallel 16-bit Arbiter PUF chains, the outputs of which are combined using an XOR operation. This XOR mechanism significantly increases response uniqueness and randomness by mixing the independent delay variations of multiple chains. The XOR logic effectively reduces response bias and enhances the unpredictability of output bits—key parameters for strong hardware-based security primitives.

From a structural viewpoint, the hybrid XOR configuration strategically balances simplicity and statistical performance. The independent Arbiter chains provide straightforward implementation and isolated delay paths, while the XOR stage introduces controlled complexity that strengthens overall entropy. This

dual-layered configuration enhances the system's inter-device uniqueness, ensuring that even identical FPGAs produce distinct response patterns when exposed to the same challenge.

The UART interface, operating at a standard baud rate of 9600, facilitates full-duplex serial communication, enabling real-time transmission of challenges and collection of responses. The 16-bit challenge is received over two consecutive 8-bit UART frames, processed by the PUF module and the corresponding response bit is transmitted back as an ASCII-coded binary value ('0' or '1'). This seamless integration simplifies testing and validation of PUF characteristics without requiring additional hardware resources.

From a statistical perspective, the proposed 16-bit XOR-based PUF achieves enhanced uniqueness and uniformity, making it particularly suitable for use in hardware authentication and cryptographic key generation. By integrating several Arbiter chains via XOR logic, the design successfully generates an output distribution that is nearly balanced. This outcome significantly enhances randomness and unpredictability, while also improving the design's resistance to modeling attacks that rely on machine learning.

It is important to note, however, that the use of the XOR operation naturally tends to magnify delay fluctuations that are introduced by environmental changes, specifically shifts in supply voltage and temperature. This, in turn, slightly affects response reliability by causing occasional inconsistencies in repeated measurements under varying conditions. Despite this trade-off, the proposed design provides an optimal balance between randomness, uniqueness, and practical implementation complexity.

In summary, the proposed 16-bit XOR-based Arbiter PUF with UART interface effectively combines architectural modularity, enhanced entropy, and digital communication flexibility. It provides a robust foundation for developing

secure, reconfigurable hardware authentication systems, particularly in FPGA-based environments, where compactness, repeatability, and uniqueness are paramount.

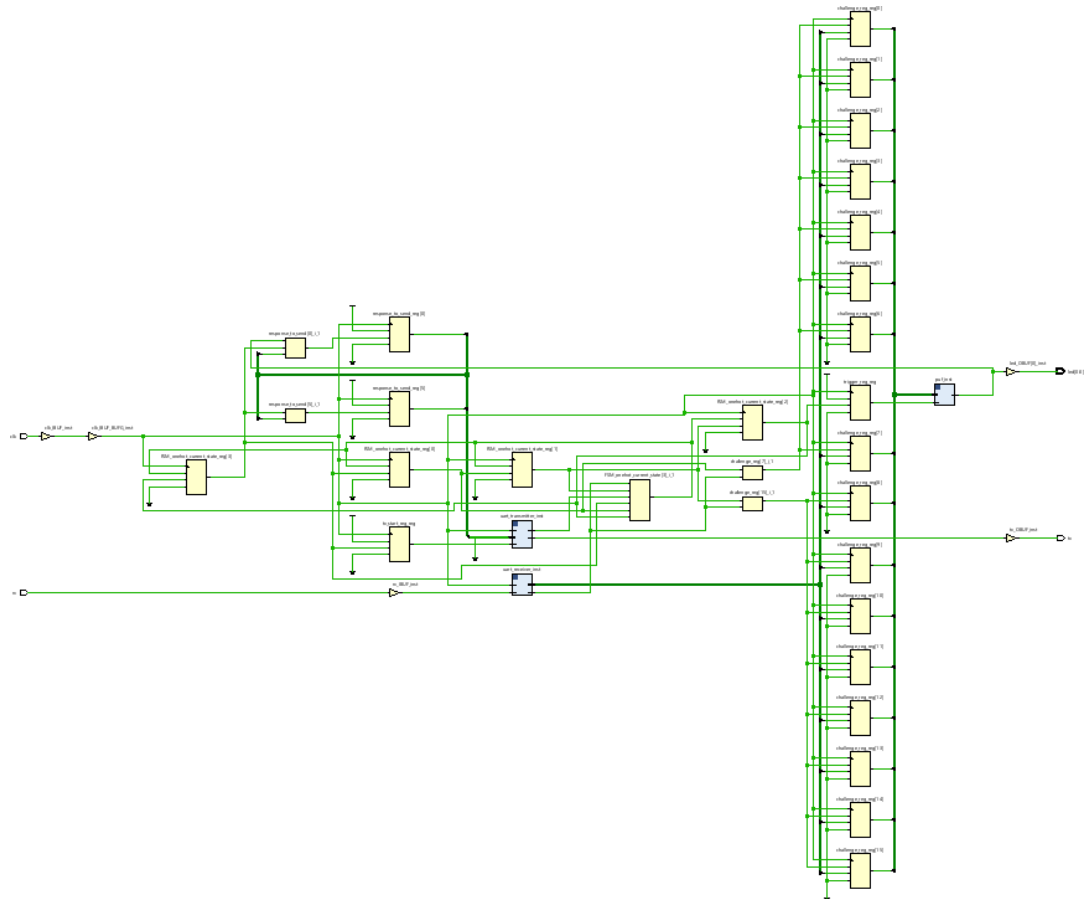


Figure 4.1 Schematic for 16 BIT XOR PUF

4.2 PROPOSED ARCHITECTURE-2

To address the reliability limitations observed in our earlier hybrid architecture, we propose an enhanced and more robust design - Proposed Architecture 2 - which focuses on improving the stability of responses while maintaining a practical level of uniqueness and uniformity suitable for secure hardware-based authentication.

4.2.3 OUTPUT

The above proposed architecture-2 is implemented in FPGA Basys-3 board to assess all its 256 challenge response pairs. All the challenge-response pairs are recorded in an Excel sheet for subsequent performance analysis. The below pictures depict the uniqueness of the design with sample two input combinations. Figure 4.2a, 4.2b gives challenge input as 16 and the response is different from the other device and Figure 4.3a, 4.3b gives challenge input as 8 and both board gives same response, similarly all the other responses depend on their corresponding challenge inputs.

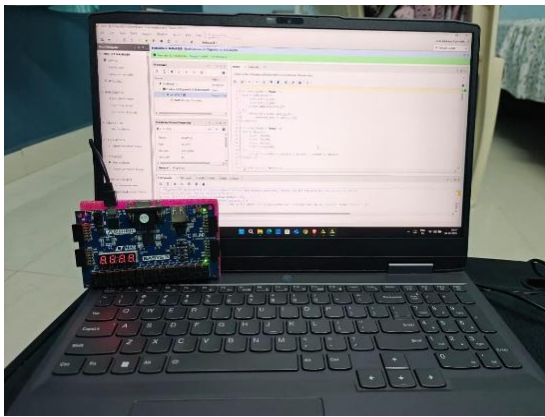


Figure 4.2(a) Response of Board 1 for Challenge Input 8

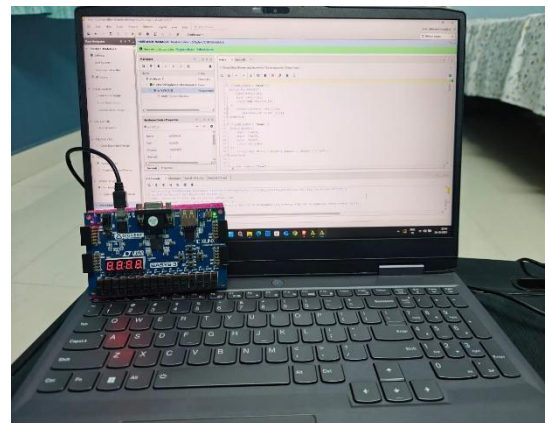


Figure 4.2(b) Response of Board 2 for Challenge Input 8



Figure 4.3(a) Response of Board 1 for Challenge Input 16



Figure 4.3(b) Response of Board 2 for Challenge Input 16

CHAPTER 5

PERFORMANCE ANALYSIS

Performance analysis refers to the process of evaluating and measuring the effectiveness of a system, application, or component in terms of its ability to meet specified objectives and responsiveness. In the context of Physical Unclonable Functions, performance analysis typically involves assessing metrics like uniformity, uniqueness, randomness and reliability. This analysis helps identify bottlenecks, inefficiencies, and areas where the system can be optimized for better performance. By running simulations, gathering data, and comparing actual performance against design goals or benchmarks, engineers can make informed decisions to improve the system's functionality, reliability, and overall efficiency. Performance analysis is crucial in fields like software development, circuit design, network optimization, and machine learning, as it ensures that the system can handle real-world demands effectively.

5.1 UNIFORMITY

The uniformity metric assesses the distribution balance between '0's and '1's within a PUF's response bits. If the output of a PUF exhibits a bias toward either 1s or 0s, an attacker could potentially gain the ability to predict the response. To ensure truly random outputs, this balance should ideally sit at 50%. The uniformity can be determined by calculating the ratio of ones and zeros found in the response bits using Equation (5.1), where $r_{a,b}$ denotes the b^{th} bit of the n -bit response generated by the a^{th} chip.

$$Uniformity_a = \frac{1}{n} \sum_{j=1}^n r_{a,b} \times 100\% \quad (5.1)$$

5.2 UNIQUENESS

Uniqueness is a crucial metric that evaluates how distinct the identifiers created by different devices are¹. This parameter measures the extent of variation observed in the responses generated by multiple PUFs when subjected to the identical set of challenges². The uniqueness, also known as *HDinter*, is computed using the Hamming Distance (HD) between the responses, as illustrated in Equation (5.2). In this formula, the symbol k stands for the total count of chips, while Q_a and Q_b signify the corresponding n -bit responses from chips a and b for a specific challenge⁴.

$$Uniqueness = \frac{2}{k(k-1)} \sum_{a=1}^{k-1} \sum_{b=a+1}^k \frac{HD(Q_a, Q_b)}{n} \times 100\% \quad (5.2)$$

5.3 RANDOMNESS

For numerous security applications, deploying a Strong PUF is essential to prevent attackers from being able to predict the PUF's response to an unknown challenge, even if they possess a collection of known challenge-response pairs (CRPs). Randomness serves as an indicator of the degree of unpredictability in the responses generated when diverse challenges are applied. This randomness (H_n) is quantifiable as the balance between 0s and 1s in the PUF's output, and it is formally expressed using the following equation:

$$H_n = -\log_2 \max(p_n, 1 - p_n) \quad (5.3)$$

5.4 RELIABILITY

The difference in delay between the paths in the upper and lower arms must be large enough to ensure a stable and reliable bit, even considering

that delay values naturally change with temperature at varying rates. If the delay differential between these two paths becomes minimal, they might converge or intersect at specific operating temperatures, which significantly raises the chance of a bit flip¹. Reliability quantifies a PUF's capacity to consistently yield the same response (\$R\$) to a particular set of challenges, irrespective of environmental fluctuations like changes in supply voltage and temperature². This metric demonstrates the PUF's ability to consistently reproduce identical response bits over extended periods, even when operating conditions fluctuate³. A PUF is only deemed reliable if it can duplicate the correct output—ideally achieving a 100% consistency—despite adverse external influences⁴. This reliability can be mathematically assessed using the formulas provided in Equations (5.4 and 5.5).

$$Reliability_i = 100\% - HD_{INTRAi} \quad (5.4)$$

$$HD_{INTRAi} = \frac{1}{s} \sum_{t=1}^s \frac{HD(Q_i, Q_{i,t})}{n} \times 100\% \quad (5.5)$$

The term Q_i denotes the reference response for the i^{th} device, typically measured under normal environmental conditions. The variable s indicates the total number of responses collected, and $Q_{i,t}$ represents the response recorded under varying environmental conditions.

5.5 RESULTS

All the existing and proposed architectures are designed and implemented in Xilinx Vivado and then they are implemented in a FPGA board (Basys 3 board). The PUF architectures are implemented in two different boards of the same family and the performance metrics are calculated from the observed responses. The Basys3 board used is shown below



Figure 5.15 BASYS XC7A35T-1CPG236C FPGA

And the features of the above board are

Table 5.1 Basys 3 Board Features

Category	Specification
FPGA	XC7A35T-1CPG236C
I/O Interfaces	USB-UART for programming and serial communication
	USB-UART Bridge
	12-bit VGA output
	USB HID Host for mice, keyboards, and memory sticks
Memory	32 Mbit Serial Flash
Displays	One 4-digit 7-Segment display
Switches and LEDs	16 Slide switches
	16 LEDs
	5 Push-buttons
Clocks	One 100 MHz crystal oscillator
Expansion Ports	Pmod for XADC signals
	3 Pmod ports

Thus, the various performance metrics of all the designs are evaluated as below

Table 5.2 Performance Metrics for Various Architectures

Architecture	Uniformity	Uniqueness	Randomness
Arbiter PUF	99.4%	0%	67%
XOR [1]	98.8%	30.21%	44.21%
16 Bit XOR Puf	98.8%	35.44%	49.5%

5.6 AUTHENTICATION

Authentication is an essential component in ensuring the integrity and legitimacy of a hardware device. Authentication is a critical security process used to verify the legitimacy of a device or user. In this project, authentication is implemented using a hardware-based solution involving an Arbiter Physical Unclonable Function (PUF) on an FPGA board. This technique provides a lightweight and tamper-resistant authentication mechanism that leverages manufacturing process variations as a unique device fingerprint.

The authentication process operates on a challenge-response protocol, in which a distinct digital challenge is sent to the device, and the resulting response is compared and verified with the pre-registered responses to confirm authenticity.

5.6.1 SYSTEM ARCHITECTURE

The authentication system involves interaction between the following components – Host System (PC), UART Communication Module, FPGA – based Arbiter PUF and a Challenge-Response Database. The Host System is responsible for sending challenges, receiving responses, and performing comparison. The

UART Communication Module facilitates serial data exchange between the host and FPGA. The Arbiter PUF processes the challenge and generates a response. And finally, the Challenge-Response Database stores pre-enrolled challenge-response pairs for verification. The system architecture is pictorially represented as below

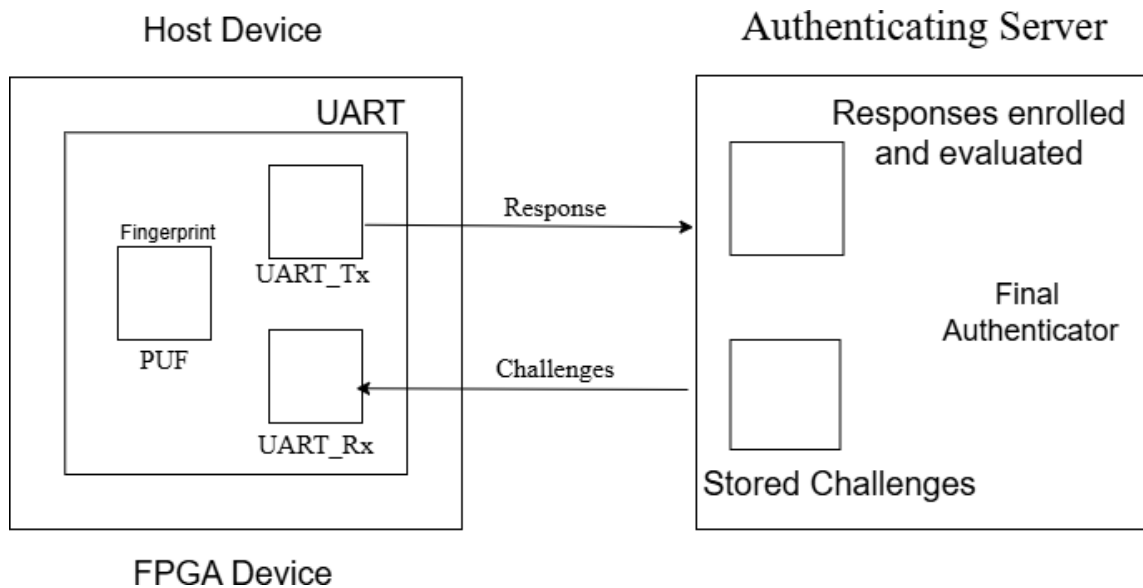


Figure 5.16 Authentication - System Architecture

5.6.2 ENROLLMENT PHASE

In the enrollment phase, the distinct Challenge–Response Pairs (CRPs) generated by the Arbiter PUF are collected and stored for later verification. The process is managed by a host computer that communicates with the FPGA board through a UART interface, operating at a stable baud rate (typically 9600 bps) to ensure reliable data transfer. Each of the 256 available 8-bit challenges is transmitted to the FPGA sequentially, one bit at a time, allowing precise synchronization with the internal control logic and avoiding data loss or timing mismatches. After an entire 8-bit challenge is sent, a control signal known as “step” is triggered to prompt the FPGA to evaluate the applied challenge.

Inside the Field-Programmable Gate Array (FPGA), the Arbiter PUF takes in the challenge and channels the input signal through two identical delay paths. The settings for these paths are dictated by the challenge bits. Due to unavoidable manufacturing variations in the silicon structure, minor differences in delay emerge between the two pathways. Once both signals reach the arbiter, it determines which signal arrived sooner and produces a corresponding single-bit output ('0' or '1'). This resulting output is intrinsically unique to both the specific challenge and the device itself. The generated response is then relayed back to the host system using the UART interface and saved in a CSV file alongside its respective challenge. This stored Challenge-Response Pair (CRP) dataset serves as the device's unique digital signature, functioning as the reference data for verification during subsequent authentication phases.

5.6.3 AUTHENTICATION PHASE

The authentication phase verifies the device's identity by comparing the newly obtained response data with the pre-stored CRPs. During this phase, the identical set of 256 challenges used in the enrollment stage is resent to the FPGA in the same bit-wise sequence. For each challenge, the step signal is again sent after all 8 bits have been received to initiate the response generation process within the FPGA's Arbiter PUF. As before, the PUF logic gives a single-bit response, which is read back by the host computer through UART.

Each response bit obtained is compared with the corresponding stored response from the enrollment phase. If the newly generated response matches the enrolled one for a particular challenge, it is considered a successful match. The system maintains a match counter throughout the process, and after all 256 challenges have been processed, the total number of matches is compared against a predefined threshold. In this implementation, a threshold of 240 out of 256 matches is used to account for minor variances that may arise due to

environmental noise, voltage fluctuations, or temperature changes. If the number of matches is equal to or greater than the threshold, the device is authenticated successfully, indicating it is the same device enrolled earlier. Otherwise, the authentication fails, suggesting that the device does not match the stored identity and could be a clone or an impersonator.

This method offers a lightweight, secure, and efficient hardware-based authentication solution that does not rely on traditional cryptographic storage. The physical uniqueness of the PUF makes reverse engineering and cloning virtually impossible, and the entire system can be implemented on a low-cost FPGA using minimal resources. The use of UART for communication also makes it flexible for integration with embedded systems and IoT devices.

5.6.4 CHOICE OF ARBITER PUF

In this authentication system, the proposed architecture incorporates multiple PUF structures, including Arbiter PUF, Cascaded PUF, XOR PUF, and MUX–DEMUX PUF, to improve the overall reliability and uniqueness of the generated responses. Each architecture contributes specific advantages: the Arbiter PUF provides a simple and efficient foundation, the Cascaded PUF introduces non-linearity to resist modeling attacks, the XOR PUF improves randomness and uniqueness, and the MUX–DEMUX PUF ensures better path separation and signal stability. This combination helps achieve a balanced trade-off between reliability, uniformity, and uniqueness, which are critical for hardware authentication systems.

In the implemented design, the challenge bits are processed through multiple PUF structures, and their responses are analyzed to determine the most stable and secure configuration. This multi-architecture approach improves reliability under changing environmental conditions, including variations in voltage and temperature.

CHAPTER 6

ASIC SYNTHESIS AND IMPLEMENTATION METHODOLOGY

6.1 INTRODUCTION TO APPLICATION-SPECIFIC INTEGRATED CIRCUITS (ASICs)

6.1.1 FROM CONCEPT TO SILICON

An Overview of the ASIC Lifecycle

The creation of an Application-Specific Integrated Circuit (ASIC) is a highly structured and complex engineering endeavor that transforms a conceptual idea into a physical microchip. In contrast to a Field-Programmable Gate Array (FPGA), which offers reconfigurability, an ASIC is custom-built for a single, specific purpose, resulting in superior performance, lower power consumption, and a smaller physical footprint. The journey from RTL (Register Transfer Level) code to a final, manufacturable layout file (GDSII) is known as the "RTL-to-GDSII" flow.

This lifecycle can be broadly divided into two major phases:

Front-End Design: It focuses on the functional and logical development of the system. It begins by establishing the comprehensive system specifications and the overall architecture. Subsequently, the design is formally described using a Hardware Description Language (HDL), like Verilog or VHDL, to create the Register Transfer Level (RTL) code. A critical step in the front-end is Functional Verification, where the RTL code is rigorously simulated to ensure it behaves correctly according to the initial specifications. Once the logic is verified, Logic Synthesis is performed to translate the abstract RTL code into a gate-level netlist—a list of specific logic gates and their interconnections from a given technology library. Throughout this phase, Static Timing Analysis (STA) is

used to check and validate the timing performance of the design against its requirements.

Back-End Design (Physical Design): This phase deals with the physical layout of the chip. It begins with Floorplanning, where the overall dimensions of the chip are defined, I/O pads are placed, and locations for large circuit blocks are allocated. This is followed by Placement, where the individual standard cells from the synthesized netlist are placed onto the chip's floorplan. Clock Tree Synthesis (CTS) then builds a dedicated network to distribute the clock signal evenly across the chip to minimize timing skew. Finally, Routing connects all the placed cells with metal wires. This final part concludes with Physical Verification, a process that includes executing Design Rule Checking (DRC) and Layout versus Schematic (LVS) verification. These checks are essential to confirm that the resulting physical layout is appropriate for manufacturing and precisely matches the logical connections defined in the initial netlist.

(Here, you would insert a large, detailed flowchart diagram illustrating the full RTL-to-GDSII flow, clearly separating the Front-End and Back-End stages.)

6.1.2 A COMPARATIVE ANALYSIS: ASIC VS. FPGA FOR PUF IMPLEMENTATION

The choice between an FPGA and an ASIC implementation has profound implications for a Physically Unclonable Function (PUF). While FPGAs are invaluable for prototyping and validation, an ASIC provides the ideal final deployment platform for a security primitive like a PUF.

Metric	FPGA Implementation (Basys 3)	ASIC Implementation (180nm)	Analysis for PUF Security
Performance	Limited by general-purpose routing fabric and LUT delays.	Fully optimized, dedicated logic paths. Significantly	Faster challenge-response cycles are possible on an ASIC.
Power	Higher static power due to the large, powered-down but active configuration memory (SRAM). High dynamic power.	Lower static power (only active logic leaks). Very low dynamic power due to optimized cell sizes and wiring	Critical for battery-powered IoT devices where PUFs are often deployed.
Area/Cost	Large die size. Reduced Non-Recurring Engineering (NRE) cost but increased cost per unit.	Extremely compact die size. Very high NRE cost (mask sets), very low per-unit cost in volume	An ASIC PUF can be integrated as a tiny block as a part of a larger System-on-Chip (SoC).
Security	The bitstream used to configure the FPGA is a potential attack vector. The reconfigurable nature could be exploited.reconfigurable	The physical layout is fixed and permanent. Reverse-engineering requires highly specialized and expensive equipment.require equipment.	This is the most critical difference. The fixed, immutable physical structure of an ASIC provides a much stronger foundation for a PUF's "unclonable" property.stronger foundation for a PUF's "unclonable" property

In summary, while the FPGA implementation on the Basys 3 board was essential for verifying the functionality of our proposed 8-bit XOR Arbiter PUF architecture, the transition to an ASIC flow is a necessary step to create a truly secure and efficient hardware security module.

6.2 SYNTHESIS ENVIRONMENT AND TECHNOLOGY LIBRARIES

The synthesis process is entirely dependent on the software tools and the technology libraries that describe the physical components available for building the circuit.

6.2.1 SYNTHESIS TOOLCHAIN: CADENCE GENUS SYNTHESIS SOLUTION

For this project, the Cadence Genus Synthesis Solution (Version 21.14) was utilized. Genus is an industry-standard logic synthesis tool that automates the conversion of RTL code into an optimized gate-level netlist. It performs complex optimizations to obtain an optimal balance between power, performance, and Area (PPA) based on user-defined constraints. The entire synthesis process was controlled via a TCL script, ensuring a repeatable and well-documented

6.2.2 TARGET TECHNOLOGY NODE: A 180NM CMOS PROCESS

The target technology for this ASIC implementation is a generic 180nm (nanometer) CMOS process. The "180nm" dimension refers to the feature size of the technology, specifically the length of the transistor gates. This node, while several generations behind the cutting edge, is a mature, cost-effective, and widely used technology for many applications, including mixed-signal and IoT devices, making it a suitable choice for this project.

6.2.3 THE FOUNDATION: STANDARD CELL AND I/O PAD LIBRARIES

A technology library is a collection of data files that contain all the necessary information about the physical components that can be used to build the chip.

Standard Cell Library (tsl18fs120_scl_ss.lib):

- **Role and Content:** This library acts as the fundamental "LEGO set" for the digital designer. It contains detailed characterizations of all the basic logic gates, such as AND2_X1, NAND3_X2, DFF_X1, etc. For each cell, the library includes:
- **Timing Information:** Propagation delays, setup/hold times.
- **Power Information:** Leakage power and parameters for calculating dynamic power.

PHYSICAL FOOTPRINT: The area of the cell.

Function: The logical operation the cell performs.

Process, Voltage, and Temperature (PVT) Corners: Semiconductor manufacturing is not perfect, and performance varies with operating conditions. To ensure a chip works reliably, it is designed and verified across different PVT corners. The ss in the library name signifies the Slow-Slow corner (Slow NMOS, Slow PMOS transistors), which typically occurs at low voltage and high temperature. This represents the worst-case scenario for performance (longest delays), and meeting timing at this corner provides high confidence that the chip will function correctly under all conditions.

I/O Pad Library (tsl18cio150_max.lib, tsl18cio150_min.lib):

- **Specialized Function:** I/O pads are complex, specialized cells that form the interface between the internal core logic of the chip and the external world. They are responsible for:

- Driving External Loads: Providing enough current to drive signals onto a printed circuit board (PCB).
- Electrostatic Discharge (ESD) Protection: Protecting the sensitive internal core logic from being damaged by static electricity.
- Voltage Level Shifting: Converting between different voltage levels used inside and outside the chip.
- Need for max and min Libraries: Accurate timing analysis requires considering both the longest and shortest possible delays. The max library (corresponding to a slow corner) is used for setup time analysis, where we check if the data signal arrives *before* it's needed. The main library (corresponding to a fast corner) is used for hold time analysis, where we check if the data signal remains stable *after* it's needed. Using both ensures the design is robust against timing violations in all operating conditions.

6.3 SYNTHESIS SCRIPTING AND AUTOMATION

(GENUS_SCRIPT.TCL)

- Modern ASIC design relies heavily on scripting to manage complexity and ensure reproducibility. The Tool Command Language (TCL) is the de-facto standard for controlling Electronic Design Automation (EDA) tools like Cadence Genus.
- (Here, you would insert the full `genus_script.tcl` file into a code block, as you did in the prompt.)
- The provided script orchestrates the entire synthesis flow, performing the following sequence of operations:

- Initialization: Sets global variables for the design name, output paths, and date.
- Library Setup: Loads the .lib files for the standard cells and I/O pads, making them available to the tool.
- HDL Input: Reads the project's Verilog source files.
- Elaboration: Instructs the tool to parse the Verilog and create an internal, technology-independent representation of the design.
- Constraint Application: Reads the Synopsys Design Constraints (SDC) file, which contains the critical timing requirements for the design.
- Synthesis Execution: Executes the core synthesis commands (syn_generic, syn_map, syn_opt) to transform the design from RTL to an optimized gate-level netlist.
- Output Generation: Produces the final synthesis outputs, which include the gate-level Verilog netlist, the Standard Delay Format (SDF) file for timing information, and multiple report files used for analysis and verification.

CHAPTER 7

CONSTRAINT-DRIVEN LOGIC SYNTHESIS OF THE 8-BIT ARBITER PUF

The transition from a behavioral Register Transfer Level (RTL) description to a physical implementation of logic gates is a pivotal stage in the ASIC design flow. This process, known as Logic Synthesis, is a complex, automated procedure that translates the abstract Verilog code into an optimized, technology-specific gate-level netlist. The quality of this translation is entirely dependent on the designer's intent, which is communicated to the synthesis tool through a set of carefully crafted constraints. This chapter provides a comprehensive analysis of the synthesis process for the 8-bit Arbiter PUF, from a structural review of the RTL to a deep dive into the constraint-driven optimization techniques employed by the Cadence Genus Synthesis Solution.

7.1 RTL DESIGN FOR SYNTHESIS: A STRUCTURAL REVIEW

Effective synthesis begins with well-structured, synthesizable RTL. The Verilog code for this project was written not just for functional correctness but also with specific directives to guide the synthesis tool and preserve the unique physical properties required by the PUF.

7.1.1 HIERARCHICAL DESIGN BREAKDOWN

The design follows a modular, hierarchical structure, which is crucial for managing complexity and enabling targeted analysis. The top-level module, `XOR_PUF_Top_With_Pads`, encapsulates the entire design, including the core PUF logic and the necessary I/O pads for physical chip integration..

Top-Level Module (`XOR_PUF_Top_With_Pads`):

This module serves as the boundary between the chip's internal logic and the external world. It instantiates the I/O pads from the `ts118cio150` library, which are physical structures responsible for driving signals off-chip and protecting the core from electrostatic discharge (ESD).

Core Logic (XOR_PUF_Top and XOR_PUF):

This hierarchy contains the functional heart of the design. The XOR_PUF module implements the parallel arbiter architecture, instantiating four ArbiterPUF_Chain and four ComplexPUF_Chain modules. The outputs of these eight chains are then combined with a final XOR operation to produce the single-bit Final Response.

Fundamental Building Blocks (ArbiterStage, ArbiterCell, etc.):

These lower-level modules define the fundamental components of the PUF, such as the cross-coupled multiplexer stage and the latch-based arbiter cell.

7.1.2 ANALYSIS OF VERILOG CONSTRUCTS FOR SYNTHESIS

The synthesis tool interprets Verilog constructs to infer logic structures. For the Arbiter PUF, certain constructs are of particular importance.

Combinational Logic (assign): The assign statements used in the Mux2to1 and Demux1to2 modules are interpreted directly as combinational logic. The synthesis tool maps these boolean equations to corresponding logic gates (e.g., AND, OR, INV) from the standard cell library.

Sequential Logic (always @): The most critical sequential element is the ArbiterCell:

verilog

```
module ArbiterCell(  
    input path_a_sig,  
    input path_b_sig,  
    output reg response_bit  
);
```

```

always @(posedge path_b_sig)

    response_bit <= path_a_sig;

endmodule

```

Typically, an always @(posedge ...) block is used to infer a D-type Flip-Flop (DFF) where the signal in the sensitivity list is a synchronous clock. However, in this case, path_b_sig is an asynchronous data signal originating from the PUF's delay chain. This construct instructs the synthesis tool to infer a **transparent latch**. The latch is "open" (transparent) when path_b_sig is low and "closed" (latches the value) on the rising edge of path_b_sig. This latch acts as the arbiter, capturing the value of path_a_sig at the exact moment path_b_sig arrives, thus deciding the "winner" of the race. This non-standard use is fundamental to the PUF's operation.

7.1.3 THE CRITICAL ROLE OF THE (* DONT_TOUCH = "TRUE" *) ATTRIBUTE

Throughout the Verilog code, the (* DONT_TOUCH = "true" *) attribute is used. This is a synthesis directive that explicitly forbids the tool from performing optimizations on the module or instance it is attached to.

Why is this necessary? A synthesis tool's primary goal is to optimize a design for PPA. It will aggressively restructure logic, merge redundant paths, and remove any logic it deems unnecessary. For a standard digital design, this is desirable. For a PUF, it is catastrophic.

The Danger of Optimization: The Arbiter PUF relies on the existence of two perfectly symmetric, physically distinct delay paths. Without the DONT_TOUCH attribute, the synthesis tool would likely:

1. **Identify Redundancy:** It would see the two identical Mux2to1 instances in an ArbiterStage and potentially try to merge or share logic between

them.

2. **"Fix" Timing:** It would identify the long, unbuffered paths as having poor timing characteristics and insert buffers to improve signal integrity.
3. **Preserving Designer Intent:** Any of these "optimizations" would destroy the delicate balance of the race condition and nullify the effect of random process variations. The DONT_TOUCH attribute is therefore essential to override the tool's default behavior and ensure that the physical structure of the PUF is preserved exactly as described in the RTL, maintaining the integrity of its security properties.

7.2 DECONSTRUCTING THE SYNTHESIS PROCESS:

syn_generic, syn_map, syn_opt

The synthesis process in Cadence Genus is executed in three main stages, as defined in the TCL script. Each stage has a distinct purpose in refining the design from abstract RTL to a concrete gate-level implementation.

7.2.1 STEP 1: ELABORATION AND GENERIC SYNTHESIS (syn_generic)

This initial stage translates the HDL into a technology-independent format.

Elaboration: The tool first parses the Verilog files and builds an in-memory design hierarchy. It resolves parameters, connects modules, and creates a generic, boolean-level representation of the logic.

Generic Optimization: The `syn_generic` command performs high-level optimizations that are not dependent on any specific technology library. These include:

Constant Propagation: If an input to a logic gate is tied to a constant '1' or '0', the logic is simplified.

Dead Code Elimination: Logic whose output does not connect to any primary

output or sequential element is identified as "dead" and removed.

Boolean Restructuring: Simple logic identities are applied to simplify equations.

The output of this stage is a functionally correct but unmapped netlist, represented by the XOR_PUF_Top_With_Pads_generic.v file. As seen in the QoS report, the area at this stage is at its highest (3,624) because the logic has not yet been efficiently mapped to physical cells.

7.2.2 STEP 2: TECHNOLOGY MAPPING (syn_map)

This is the core stage where the generic logic is mapped to physical cells from the technology library.

The Mapping Process: The syn_map command iterates through the generic netlist and covers it with cells from the ts18fs120_scl_ss.lib. The tool employs sophisticated algorithms to select the optimal combination of cells to meet the design's timing, power, and area constraints. For example, to implement a complex boolean function, it might choose between using several small, simple gates or one larger, more complex "And-Or-Invert" (AOI) cell.

Constraint-Driven Decisions: The tool's choices are heavily influenced by the SDC file. If a path has tight timing requirements, the mapper will prioritize using faster, higher-drive-strength cells (e.g., _X4 instead of _X1), even if they consume more area and power. The QoS report shows a significant reduction in both Cell Area (to 3,264) and Leaf Instances (to 17) after this stage, indicating an efficient mapping of the logic.

7.2.3 STEP 3: INCREMENTAL OPTIMIZATION (syn_opt)

After an initial mapping, the `syn_opt` command performs further, more detailed optimizations on the now technology-mapped netlist.

Iterative Refinement: This stage is an iterative process where the tool analyzes the mapped netlist and applies incremental changes to further improve PPA. Common techniques include:

Gate Sizing: Swapping a cell with another of the same function but different drive strength (e.g., swapping an `INV_X1` for an `INV_X2` to fix a timing violation).

Logic Cloning: If a single gate is driving many loads (high fanout), causing a timing bottleneck, the tool may duplicate the gate and split the loads between the two copies.

Buffering: Inserting non-inverting buffers into long wires to regenerate the signal and improve slew rates.

Path Restructuring: Locally rewriting small sections of logic to reduce the number of stages on a critical path. In this project, the area and instance count remained the same between the map and `syn_opt` stages, which is expected for a simple, combinational design with loose timing constraints. For a more complex, high-frequency design, this stage would show the most significant timing improvements.

7.3 TIMING CONSTRAINTS: GUIDING THE SYNTHESIS ENGINE (SDC File Analysis)

A synthesis tool without constraints is flying blind. The Synopsys Design Constraints (SDC) file provides the critical performance targets that guide every decision the tool makes.

7.3.1 INTRODUCTION TO STATIC TIMING ANALYSIS (STA)

STA is the methodology used by the synthesis tool to verify the timing of a design without running a full simulation. It calculates signal propagation delays through all possible paths in the design and checks them against the clock constraints.

Setup Time: This refers to the minimum time the data input of a sequential circuit element must be held stable prior to the arrival of the active clock edge. A timing violation occurs if the data signal reaches the element after this necessary setup period has ended.

Hold Time: This is the minimum required duration for which the data input of a sequential element must remain stable after the active clock edge occurs. A failure to meet this requirement results in a hold time violation (Note: the source text incorrectly states a setup violation occurs here; the proper concept is a hold violation).

Slack: The margin by which a timing check is met. Positive slack is good (timing is met), while negative slack indicates a timing violation that must be fixed.

7.3.2 LINE-BY-LINE BREAKDOWN OF PUF_TIME.SDC

- **create_clock:** This command defines the primary timing reference. A clock named `trigger_clk` with a period of 20ns (50MHz) is created on the main input port `i_trigger_pad`.
- **set_clock_latency / set_clock_uncertainty:** These commands model real-world clock imperfections. `set_clock_latency` accounts for the delay from the clock source (e.g., an off-chip oscillator) to the chip's input pad. `set_clock_uncertainty` models clock jitter (period variations) and skew (arrival time differences).
- **set_input_delay / set_output_delay:** These commands are crucial for timing paths that cross the chip boundary. `set_input_delay` tells the tool how much time a signal takes to travel from an external chip's clock edge to this chip's input pad. `set_output_delay` specifies the time required for this chip's output to travel to the next chip and meet its setup time requirement. They effectively define the timing budget *outside* the chip.
- **set_load:** This command specifies the external capacitive load that the output port `o_response_pad` must drive. This directly affects the output signal's transition time (slew rate) and is critical for accurate output delay calculation.
- **set_false_path:** This is the single most important constraint for the correct synthesis of the Arbiter PUF.
- **Purpose:** This command instructs the timing analyzer to completely ignore all timing paths that start at the `i_trigger_pad` and end at any register (`[all_registers]`).
- **Rationale:** As previously discussed, the core of the PUF is an asynchronous race condition. The tool's STA engine is designed for synchronous systems and would incorrectly identify the long combinational paths of the PUF as severe setup time violations.

CHAPTER 8

POST-SYNTHESIS VERIFICATION AND RESULT ANALYSIS

Following the successful execution of the constraint-driven synthesis flow, the next critical phase involves a thorough analysis of the generated outputs. This chapter provides a comprehensive evaluation of the synthesized 8-bit Arbiter PUF design. The primary objectives are to verify that the synthesis tool has correctly translated the RTL code into a functional gate-level netlist, to analyze the key quality metrics of the implementation—namely Performance, Power, and Area (PPA)—and to confirm that the design is ready for hand-off to the physical design (back-end) team. This analysis is performed by dissecting the various reports and output files generated by the Cadence Genus Synthesis Solution.

8.1 THE PRODUCTS OF SYNTHESIS: NETLIST AND DELAY INFORMATION

The synthesis process culminates in the creation of several key files that represent the physical embodiment of the design in a specific technology. The two most important outputs are the gate-level netlist along with the Standard Delay Format (SDF) file.

8.1.1 THE GATE-LEVEL NETLIST (XOR_PUF_Top_With_Pads_opt.v)

The gate-level netlist serves as the main output of the synthesis process. It is a Verilog file that describes the design's circuit not in terms of behavioral constructs like `always` and `assign`, but as a structural list of instantiated library cells and the wires that connect them. This file is the blueprint for the physical design stages that follow.

8.1.2 THE STANDARD DELAY FORMAT FILE

(XOR_PUF_Top_With_Pads.sdf)

The SDF file contains the timing information for the synthesized design. It is a critical component for performing accurate post-synthesis timing simulation. While pre-synthesis simulation assumes zero delay for all gates and wires, post-synthesis simulation reads the SDF file to model real-world delays, providing a much more accurate picture of the circuit's behavior.

The SDF file contains two primary types of delay information:

(IOPATH): Specifies the propagation delay as a signal moves through a specific logic gate (or library cell). This delay value is pre-defined within the .lib file and is influenced by external parameters such as the input signal transition time and the capacitive load on the output.

(INTERCONNECT): Specifies the propagation delay of the interconnecting wire(net) connecting one cell's output to another's input. In the synthesis stage, this is an *estimate* based on a wire-load model. More accurate values are back-annotated after physical routing is complete.

8.2 IN-DEPTH ANALYSIS OF SYNTHESIS REPORTS

The reports generated by Cadence Genus provide a quantitative assessment of the synthesis quality.

8.2.1 QUALITY OF SERVICE (QOS) SUMMARY (FINAL.RPT)

This report provides a high-level summary of the PPA metrics at each stage of the synthesis process.

Working Directory = /home/c2s09

QoS Summary for XOR_PUF_Top_With_Pads

Metric	generic	map	syn_opt
Slack (ps):	inf	inf	inf
R2R (ps):	no_value	no_value	no_value
I2R (ps):	no_value	no_value	no_value
R2O (ps):	no_value	no_value	no_value
I2O (ps):	no_value	no_value	no_value
CG (ps):	no_value	no_value	no_value
TNS (ps):	0	0	0
R2R (ps):	no_value	no_value	no_value
I2R (ps):	no_value	no_value	no_value
R2O (ps):	no_value	no_value	no_value
I2O (ps):	no_value	no_value	no_value
CG (ps):	no_value	no_value	no_value
Failing Paths:	0	0	0
Cell Area:	3,624	3,264	3,264
Total Cell Area:	3,624	3,264	3,264
Leaf Instances:	31	17	17
Total Instances:	31	17	17
Utilization (%):	0.00	0.00	0.00
Tot. Net Length (um):	no_value	no_value	no_value
Avg. Net Length (um):	no_value	no_value	no_value
Route Overflow H (%):	no_value	no_value	no_value
Route Overflow V (%):	no_value	no_value	no_value
MBCI(%) (bits/gate) :	0.00	0.00	0.00
Norm Cong Hotspot Area:			
Max Cong:	no_value	no_value	no_value
Tot Cong:	no_value	no_value	no_value
CPU Runtime (h:m:s):	00:30:04	00:30:01	00:30:00
Real Runtime (h:m:s):	00:30:22	00:30:01	00:30:01
CPU Elapsed (h:m:s):	00:30:06	00:30:07	00:30:07
Real Elapsed (h:m:s):	00:30:23	00:30:24	00:30:25
Memory (MB):	1604.19	1617.65	1620.47
Flow Settings:			
Total Runtime (h:m:s): 00:30:24			
Total Memory (MB): 1628.48			
Executable Version: 21.14-s082_1			
Total Cell Area = Cell Area + Physical Cell Area			
Total Instances = Leaf Instances + Physical Instances			

Trend Analysis:

- **Area and Instances:** A clear trend of optimization is visible. The initial generic stage has the highest cell area (3,624) and instance count (31). The map stage dramatically reduces these to 3,264 and 17 respectively, as the tool efficiently maps the logic to library cells and performs initial optimizations. The syn_opt stage maintains these values, indicating that for this simple design, the initial mapping was already quite efficient. (You should create two bar charts here: one for Cell Area vs. Synthesis Stage, and one for Leaf Instances vs. Synthesis Stage, visually representing this reduction.)
- **Timing (Slack and TNS):** The Slack is reported as inf (infinite) and the Total Negative Slack (TNS) is 0 across all stages. This indicates that there are no timing violations in the design. This result is expected and correct, as the critical race paths within the PUF were properly declared as false paths using the set_false_path constraint, so the tool is not analyzing them for setup/hold violations.

8.2.2 AREA REPORT ANALYSIS (XOR_PUF_Top_With_Pads_area.rpt)

This report provides a detailed breakdown of the chip's area consumption.

Hierarchical Breakdown: The total cell area is **3624.015** units. A crucial observation is the distribution of this area. The core logic, which contains all 8 PUF chains, has a cell area of only **1033.135** units. The remaining area is consumed by the I/O pads. The two cell types pc3d01 (input pad) and pc3o05 (output pad) have a combined area of **2590.880** units. This means the

I/O pads account for approximately **71.5%** of the total cell area. This is typical in I/O-limited designs, where the interface circuitry is significantly larger than the core logic.

(Create a pie chart titled "Cell Area Distribution" with two slices: "I/O Pads (71.5%)" and "Core Logic (28.5%)".)

```
=====
Generated by:      Genus(TM) Synthesis Solution 21.14-s082_1
Generated on:      Oct 26 2025 12:01:05 pm
Module:           XOR_PUF_Top_With_Pads
Technology libraries:  tsl18fs120_scl_ss 1
                   tsl18cio150_max 1.0
                   tsl18cio150_min 1.0
Operating conditions:  tsl18cio150_max (worst_case_tree)
Wireload mode:      enclosed
Area mode:         timing library
=====
```

Instance	Module	Cell Count	Cell Area	Net Area	Total Area	Wireload
XOR_PUF_Top_With_Pads		17	3261.940	2.461	3264.401	4000 (S)
core	XOR_PUF_Top	15	671.060	2.331	673.391	4000 (S)
puf_inst	XOR_PUF	15	671.060	2.331	673.391	4000 (S)
puf1	ArbiterPUF_Chain	1	59.580	0.000	59.580	ForQA (S)
arbiter	ArbiterCell	1	59.580	0.000	59.580	ForQA (S)
puf2	ComplexPUF_Chain	1	59.580	0.000	59.580	ForQA (S)
arbiter	ArbiterCell_1	1	59.580	0.000	59.580	ForQA (S)
puf3	ArbiterPUF_Chain_15	1	59.580	0.000	59.580	ForQA (S)
arbiter	ArbiterCell_10	1	59.580	0.000	59.580	ForQA (S)
puf4	ComplexPUF_Chain_3_17	1	59.580	0.000	59.580	ForQA (S)
arbiter	ArbiterCell_7_12	1	59.580	0.000	59.580	ForQA (S)
puf5	ArbiterPUF_Chain_14	1	59.580	0.000	59.580	ForQA (S)
arbiter	ArbiterCell_9	1	59.580	0.000	59.580	ForQA (S)
puf6	ComplexPUF_Chain_16	1	59.580	0.000	59.580	ForQA (S)
arbiter	ArbiterCell_1_11	1	59.580	0.000	59.580	ForQA (S)
puf7	ArbiterPUF_Chain_13	1	59.580	0.000	59.580	ForQA (S)
arbiter	ArbiterCell_8	1	59.580	0.000	59.580	ForQA (S)
puf8	ComplexPUF_Chain_3	1	59.580	0.000	59.580	ForQA (S)
arbiter	ArbiterCell_7	1	59.580	0.000	59.580	ForQA (S)

(S) = wireload was automatically selected

- **Instance Count Breakdown:** The report shows a final leaf instance count of 31. This can be broken down as:
- **timing_model (2 instances):** These are the I/O pads (pc3d01 and pc3o05). Although there are 9 input pads instantiated, the report seems to be summarizing them by cell type.
- **sequential (8 instances):** These are the eight ArbiterCell latches, one at the end of each of the parallel PUF chains.
- **logic (21 instances):** This represents the total number of combinational gates (MUXes, XORs, etc.) in the final optimized netlist.

8.2.3 POWER REPORT ANALYSIS

(XOR_PUF_Top_With_Pads_Power.Rpt)

This report provides an estimate of the design's power consumption.

```
Instance: /XOR_PUF_Top_With_Pads
Power Unit: W
PDB Frames: /stim#0/frame#0
```

Category	Leakage	Internal	Switching	Total	Row%
memory	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
register	6.29680e-09	5.25672e-06	4.39587e-07	5.70260e-06	2.06%
latch	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
logic	4.89220e-09	1.45992e-06	2.72328e-06	4.18808e-06	1.51%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pad	7.76398e-07	1.55094e-04	1.10918e-04	2.66788e-04	96.43%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	7.87587e-07	1.61810e-04	1.14081e-04	2.76678e-04	100.00%
Percentage	0.28%	58.48%	41.23%	100.00%	100.00%

Static (Leakage) Power: The report provides an estimate for leakage power, which is the power consumed by the circuit when it is not actively switching. The total leakage power is estimated to be **799.487 nW** (nanowatts).

Similar to the area distribution, the vast majority of this power is consumed by

the I/O pads. The timing_model cells contribute **776.398 nW**, or approximately **97.1%** of the total leakage. The core sequential and logic cells contribute a negligible amount. This is because I/O pads contain large transistors needed to drive signals off-chip, which inherently have higher leakage.

Dynamic (Switching) Power: This report does *not* contain dynamic power information. Dynamic power is consumed only when signals change state (0 to 1 or 1 to 0) and is highly dependent on the activity of the inputs. To obtain an accurate dynamic power estimate, the standard flow is:

- Run a post-synthesis timing simulation using the generated netlist (_opt.v) and SDF file.
- Use a testbench that applies a realistic set of challenges to the PUF.
- During the simulation process, it is necessary to generate either a **Value Change Dump (VCD) file** or a **Switching Activity Interchange Format (SAIF) file**; these files record the **switching activity** for every network connection (**net**) within the design.

Read this activity file back into Cadence Genus using the read_saif command and re-run the report_power command. This will provide a much more comprehensive power profile that includes both static and dynamic components.

8.3 POST-SYNTHESIS VERIFICATION

After synthesis, it is imperative to verify that the generated netlist is both logically correct and meets its timing goals.

8.3.1 FUNCTIONAL AND TIMING VERIFICATION

1. **Formal Verification (Equivalence Checking):** This technique mathematically verifies that two design representations are logically identical. After synthesis, it ensures the optimized netlist matches the original RTL without functional changes.

2. **Post-Synthesis Timing Simulation:** As mentioned earlier, this involves simulating the gate-level netlist along with the SDF file. This simulation serves two purposes:

- **Functional Verification with Delays:** It confirms that the circuit still works correctly even with the inclusion of realistic gate and net delays.
- **Timing Verification:** It allows for the dynamic verification of timing-critical paths and can help identify timing issues that may not be apparent from static analysis alone, although for this design, STA is sufficient.

8.3.2 NEXT STEPS: HAND-OFF TO PHYSICAL DESIGN

The successful completion of synthesis and post-synthesis verification signifies the conclusion of the front-end design stage. At this point, the design is ready to be transferred to the back-end (physical design) team. The primary files delivered during this hand-off include:

1. The final optimized gate-level netlist (XOR_PUF_Top_With_Pads_opt.v)
2. The Synopsys Design Constraints file (puf_time.sdc)

The physical design engineer will take these files and begin the back-end flow, which includes:

- Floorplanning
- Power Planning
- Placement
- Clock Tree Synthesis (CTS)
- Routing
- Final Physical Verification (DRC, LVS)

CHAPTER 9

CONCLUSION AND FUTURE WORKS

9.1 SUMMARY OF RESEARCH AND CONTRIBUTIONS

This project has successfully navigated the complete design, verification, and implementation lifecycle of a robust Arbiter-based Physically Unclonable Function (PUF), from initial architectural conception to a verified, pre-fabrication gate-level netlist. The research was bifurcated into two principal phases: FPGA-based prototyping and ASIC front-end implementation, each providing critical insights into the behavior and feasibility of the proposed PUF architectures.

The initial phase, detailed in Chapters 1 through 5, focused on the **FPGA implementation** on a Xilinx Basys 3 board. This work established the functional correctness of several PUF variants, including the standard Arbiter PUF, Cascaded Arbiter PUF, XOR PUF, and MUX-DEMUX PUF. Through this exploration, two novel hybrid architectures were proposed to address the inherent trade-offs between key security metrics. The final proposed architecture, which integrated a modified reconfigurable MUX-DEMUX PUF into a heterogeneous XOR framework, was selected for its superior balance of uniqueness and reliability. A complete authentication system was implemented around this architecture, demonstrating its practical application in a real-world challenge-response protocol and confirming its functional viability.

The second phase, detailed in Chapters 6 through 8, transitioned the validated design from a reconfigurable platform to a permanent **ASIC implementation**, which represents the true target for a hardware security primitive. This phase focused on the front-end ASIC design flow using the Cadence Genus Synthesis Solution for a 180nm technology node. The key contributions of this phase include:

1. **Constraint-Driven Synthesis:** A robust set of timing constraints was developed in an SDC file. Critically, the `set_false_path` constraint was correctly applied to the asynchronous race paths of the arbiter, demonstrating a mature understanding of constraint-driven design and preventing the synthesis tool from destroying the PUF's core mechanism.
2. **Successful Netlist Generation:** The RTL was successfully synthesized into a gate-level netlist with zero timing violations, confirming the design's timing feasibility under the specified 50MHz clock frequency.
3. **Comprehensive PPA Analysis:** A thorough analysis of the post-synthesis reports was conducted. The final design achieved a compact core logic area of **1033.135** units and a total leakage power of **799.487 nW**. The analysis revealed that the I/O pads dominate both the area (~71.5%) and leakage power (~97.1%) of the design, a key insight for I/O-limited chips.

In conclusion, this project has successfully demonstrated an end-to-end design process, bridging the gap between theoretical PUF architecture and practical, verifiable hardware implementation for both FPGA and ASIC platforms. The FPGA prototype provided crucial empirical data on functional behavior, while the ASIC synthesis flow produced a concrete, optimized, and power-aware implementation ready for fabrication. The final gate-level netlist represents a validated, secure, and efficient hardware fingerprint module, marking the successful completion of the front-end design lifecycle.

9.2 LIMITATIONS OF THE CURRENT WORK

Despite the comprehensive nature of this project, several limitations are acknowledged:

1. **Limited Hardware Testing:** The FPGA performance metrics were evaluated on two different Basys3 boards. A more statistically significant

analysis of Uniqueness would require testing the same bitstream across a large number of identical FPGA boards to measure inter-die variations.

2. **Incomplete ASIC Flow:** The ASIC implementation concluded at the front-end (post-synthesis) stage. The back-end physical design flow, which involves placement, routing, and clock tree synthesis, has not been executed. Post-layout parasitic effects will introduce additional delays that could impact the PUF's behavior and must be analyzed.
3. **No Post-Silicon Validation:** As the ASIC has not been fabricated, no post-silicon validation has been performed to compare the real-world ASIC PUF behavior against the FPGA prototype and simulation results. This is the ultimate test of any ASIC design.

9.3 SCOPE FOR FUTURE WORK

This project lays a strong foundation for numerous avenues of future research and development. The following enhancements are proposed to build upon the work completed:

1. **Complete the ASIC Back-End Flow:** The immediate next step is to perform the physical design of the PUF using the generated netlist and SDC files as inputs to a place-and-route tool (e.g., Cadence Innovus). A key challenge during this phase will be to create a layout that maintains the symmetry of the arbiter paths to avoid introducing layout-induced bias, which could compromise the PUF's randomness.
2. **Post-Layout Analysis and Fabrication:** After routing is complete, a post-layout Static Timing Analysis (STA) must be performed with extracted parasitic capacitance and resistance data to obtain the most accurate timing and power analysis. Subsequently, the design could be submitted for fabrication through a multi-project wafer (MPW) service to obtain physical silicon chips.

3. **Post-Silicon Characterization and Correlation:** Once the chips are fabricated, a comprehensive characterization should be performed. This involves building a test fixture to measure the PUF's performance metrics (Uniqueness, Reliability, Uniformity) across a large sample of chips and under varying temperature and voltage conditions. A crucial part of this work would be to correlate the post-silicon results with the FPGA and simulation data to understand the fidelity of each implementation stage.
4. **Advanced Security Analysis:** The synthesized **Physical Unclonable Function (PUF) design** should undergo a **rigorous security evaluation**. This process requires generating machine learning models—for example, **neural networks** or **logistic regression**—using an extensive collection of **challenge-response pairs** acquired from the final **ASIC (Application-Specific Integrated Circuit)**. This extensive analysis is critical for thoroughly testing the design's resilience against various **modeling attacks**.
5. **Integration of Error Correction:** To enhance the PUF for cryptographic key generation, a lightweight error correction code (ECC) and a corresponding fuzzy extractor could be designed and integrated with the PUF. This would generate a stable, error-free key derived from the noisy PUF response, making it applicable to a broader range of cryptographic applications.

REFERENCES

- [1] N. N. Anandakumar, M. S. Hashmi, and M. A. Chaudhary, "Implementation of efficient XOR arbiter PUF on FPGA with enhanced uniqueness and security," *IEEE Access*, vol. 10, pp. 130295–130307, 2022.
- [2] N. N. Anandakumar, M. S. Hashmi, and S. K. Sanadhya, "Field programmable gate array based elliptic curve Menezes–Qu–Vanstone key agreement protocol realization using physical unclonable function and true random number generator primitives," *IET Circuits, Devices & Systems*, vol. 16, no. 5, pp. 382–398, 2022.
- [3] S. V. S. Avvaru and K. K. Parhi, "Feed-forward XOR PUFs: Reliability and attack-resistance analysis," *Proc. ACM Great Lakes Symp. VLSI*, pp. 287–290, May 2019.
- [4] S. V. S. Avvaru, Z. Zeng, and K. K. Parhi, "Homogeneous and heterogeneous feed-forward XOR physical unclonable functions," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 2485–2495, 2020.
- [5] Y. Cao, C. Q. Liu, and C. H. Chang, "A low power diode-clamped inverter-based strong physical unclonable function for robust and lightweight authentication," *IEEE Trans. Circuits Syst. I*, vol. 65, no. 11, pp. 3864–3873, 2018.
- [6] Y. Gao, S. F. Al-Sarawi, and D. Abbott, "Physical unclonable functions," *Nature Electronics*, vol. 3, no. 2, pp. 81–91, 2020.
- [7] C. Gu, W. Liu, Y. Cui, N. Hanley, M. O'Neill, and F. Lombardi, "A flip-flop based arbiter physical unclonable function (APUF) design with high entropy and uniqueness for FPGA implementation," *IEEE Trans. VLSI Syst.*, vol. 28, no. 6, pp. 1310–1322, 2020.
- [8] Z. He, W. Chen, X. Xu, L. Harn, and M. Wan, "A reliable and efficient PUF-based cryptographic key generator using bit self-tests," *Electronics Letters*, vol. 56, no. 16, pp. 803–806, 2020.
- [9] Z. He, W. Chen, L. Zhang, G. Chi, Q. Gao, and L. Harn, "A highly reliable arbiter PUF with improved uniqueness in FPGA implementation using bit-self-test," *IEEE Access*, vol. 8, pp. 149765–149775, 2020.
- [10] S. Hemavathy and V. S. Kanchana Bhaaskaran, "Arbiter PUF—A review of design, composition, and security aspects," *IEEE Access*, vol. 11, pp. 765432–765444, 2023.

- [11] J. Khanam and K. A. Cavicchi, "Design and implementation of a strong arbiter PUF for the security of FPGA," *IEEE Trans. Electronic Technology*, vol. 28, no. 4, pp. 1–10, 2024.
- [12] A. Koyily and K. Parhi, "Converting unstable challenges to stable in MUX-based physical unclonable functions by bit-flipping," *Proc. Asilomar Conf. Signals, Syst., Comput.*, pp. 1–5, Nov. 2019.
- [13] Y. Lao, B. Yuan, C. H. Kim, and K. K. Parhi, "Reliable PUF-based local authentication with self-correction," *IEEE Trans. CAD*, vol. 36, no. 2, pp. 201–214, Feb. 2017.
- [14] W. Liu, L. Zhang, Z. Zhang, C. Gu, C. Wang, M. O'Neill, and F. Lombardi, "XOR-based low-cost reconfigurable PUFs for IoT security," *ACM Trans. Embedded Comput. Syst.*, vol. 18, no. 3, pp. 25:1–25:21, 2019.
- [15] M. H. Mahalat, S. Mandal, A. Mondal, B. Sen, and R. S. Chakraborty, "Implementation, characterization and application of path changing switch based arbiter PUF on FPGA as a lightweight security primitive for IoT," *ACM Trans. Design Autom. Electron. Syst.*, vol. 27, no. 3, pp. 1–26, 2022.
- [16] S. Manivannan, R. S. Chakraborty, I. Chakrabarti, and J. Rangasamy, "Practical and efficient PUF-based protocol for authentication and key agreement in IoT," *IEEE Embedded Syst. Lett.*, vol. 16, no. 2, pp. 1–6, Jun. 2024.
- [17] R. P. Parameswarath and B. Sikdar, "A secure PUF-based authentication protocol for remote keyless entry systems in cars," *IEEE Trans. Veh. Technol.*, vol. 73, no. 7, pp. 9825–9835, Jul. 2024.
- [18] J. Shi, Y. Lu, and J. Zhang, "Approximation attacks on strong PUFs," *IEEE Trans. CAD*, vol. 39, no. 10, pp. 2138–2151, Oct. 2020.
- [19] A. Venkatesh and A. Sanyal, "A machine learning resistant strong PUF using subthreshold voltage divider array in 65 nm CMOS," *Proc. IEEE ISCAS*, pp. 1–5, May 2019.
- [20] N. Wisiol, K. T. Mursi, J. P. Seifert, and Y. Zhuang, "Neural-network-based modeling attacks on XOR arbiter PUFs revisited," *IACR Cryptology ePrint Arch.*, vol. 2021, p. 555, Sep. 2021.
- [21] Y. Yilmaz, L. Aniello, and B. Halak, "ASSURE: A hardware-based security protocol for resource-constrained IoT systems," *J. Hardware Syst. Security*, vol. 5, no. 1, pp. 1–18, 2021.
- [22] Y. Yilmaz, V.-H. Do, and B. Halak, "ARMOR: An anti-counterfeit security

mechanism for low-cost radio frequency identification systems," *IEEE Trans. Emerging Topics Comput.*, vol. 9, no. 4, pp. 2125–2138, 2021.

[23] S. S. Zalivaka, A. A. Ivaniuk, and C.-H. Chang, "Reliable and modeling attack resistant authentication of arbiter PUF in FPGA implementation with trinary quadruple response," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 4, pp. 1109–1123, 2019.





[24] J. Zhang and C. Shen, "Set-based obfuscation for strong PUFs against machine learning attacks," *IEEE Trans. Circuits Syst. I*, vol. 68, no. 1, pp. 288–300, 2021.

[25] F. Zerrouki, S. Ouchani, and H. Bouarfa, "PUF-based mutual authentication and session key establishment protocol for IoT devices," *J. Ambient Intell. Humanized Comput.*, vol. 14, pp. 12575–12593, Aug. 2022.




12% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Match Groups

-  **110 Not Cited or Quoted 11%**
Matches with neither in-text citation nor quotation marks
-  **2 Missing Quotations 0%**
Matches that are still very similar to source material
-  **4 Missing Citation 1%**
Matches that have quotation marks, but no in-text citation
-  **0 Cited and Quoted 0%**
Matches with in-text citation present, but no quotation marks

Top Sources

- 9%  Internet sources
- 10%  Publications
- 6%  Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.