Instituto de Microelectrónica de Sevilla - IMSE (CSIC/US)

Design of a hardware Root-of-Trust on embedded systems

Eros Camacho Ruiz

Supervisors
Dr. Piedad Brox Jiménez
Prof. Francisco Vidal Fernández Fernández

This dissertation is submitted for the degree of
Doctor of Philosophy

February 2024

A mis abuelos, Juani y Pepe
A mi madre, Virginia.

Acknowledgements

This dissertation has been developed at the Instituto de Microelectrónica de Sevilla - IMSE (CSIC/US). It was supported also by TOGETHER (TEC2016-75151-C3-3-R) and VIGILANT (PID2019-103869RB-C31) projects from the Spanish Government, and SPIRS (GA: 952622) and QUBIP (GA: 101119746) from the European Commission. It was partially supported by FPU (Formación del Profesorado Universitario) fellowship program for PhD Student from Spanish Government (December 2021 - September 2023). During 2023, a stay of 3 months has been accomplished at the "Network and Information Security Group (NISEC)" of the Tampere University in Finland, supported by SPIRS project.

I would also like to express my deepest appreciation to my thesis directors, whose expertise, understanding, and patience, added considerably during this stage. Firstly, I would like to thank to Paco for contributing with his enormous knowledge capacity and his critical vision. Also, it is not among in the supervisors list, I don't want to miss the opportunity to thank Rafa for his invaluable and selfless help in the completion of this thesis. Your knowledge in the analog design field and your perseverance.

I cannot forget to thank Santi, who although he has not been the supervisor of the thesis, his help has been a key pillar in all the work done. It is impossible to forget the "Thursday meeting" that kept us alive during the pandemic. And finally, I don't want to forget Piedad. If I am something called a scientist or researcher, it is because of you and because you believed in me when nobody else did. You have given me the opportunity to grow as a person and as a professional for which I am eternally grateful.

I would like to thank anyone who wasted their time on helping me during this stage of my life, particularly Macarena and Felipe. For our talks at Sapiens. All the colleges that now are more than just colleges from PB-12: Andrés, Franpi, Chema, Virginia, Manu. Even if we are worlds apart, we will always find time to have breakfast at La Manuela. I would also like to thank all the IMSE staff who have made it possible for me to have a place to work every day: Technical Unit, janitors and cleaners.

Por último, dar las gracias a mis abuelos, porque si soy algo, hoy día, es gracias a ellos. Y por supuesto a mi madre, por nunca perder la esperanza y mantenerme a tu lado todo lo que has podido, por muchos kilómetros que tuviéramos que recorrer. Fíjate hasta donde hemos llegado.

Preface

Nowadays, cybersecurity is essential for world's economic security, protecting sensitive information and ensuring safe online services. It is critical for several infrastructures, data privacy, national security or global communications. To this end, several organizations around the globe (i.e., National Institute of Standards and Technology) are in charged to develop new strategies to prevent cyber-attacks or information leakages. To this end, the CIA Triad, is a fundamental cybersecurity model that forms the basis for security systems development, representing Confidentiality, Integrity, and Availability.

The Root-of-Trust in cybersecurity, which is closely tied to the principles of the CIA Triad, serves as the foundation for creating a secure computing environment. Root-of-Trusts, which include hardware, firmware, and software components, provide essential functions dependent on trust. In terms of Confidentiality, the Root-of-Trusts securely store cryptographic keys and manages them, ensuring sensitive data remains confidential. For Integrity, they can verify the integrity of essential components during the boot process, maintaining system integrity and preventing unauthorized modifications. Physical Unclonable Functions enhance security by providing unique identifiers and contributing to various security functions. Regarding Availability, they can reduce the risk of compromise and unauthorized access, mitigating security incidents that could disrupt service. Hardware Root-of-Trust are preferred over software ones due to their immutability, reduced susceptibility to attacks, and dependable performance, despite the trade-off in deployment speed.

On the other hand, the advent of quantum computing poses a challenge to traditional cybersecurity paradigms, as it could render widely-used cryptographic algorithms obsolete by performing complex computations at unprecedented speeds. To counter the challenges posed by quantum computing, the field of Post-Quantum Cryptography is emerging with the goal of developing cryptographic algorithms that remain secure in the quantum computing era. In 2017, National Institute of Standards and Technology initiated a Post-Quantum Cryptography algorithm standardization contest to define the first standard.

Internet of Things, a network of interconnected devices embedded with sensors, offers opportunities for efficiency and convenience but also presents unique security challenges. The hypothesis of this dissertation is that the design and implementations of hardware cryptographic

modules are the most suitable to secure the so-called Internet of Things devices. The combination of these hardware components gives birth to a Root-of-Trust that offers the most efficient performance results, together with an extra security level of protection derived from the physical implementations that avoid software attacks (malware). The proposed solution presented in this dissertation encompasses a set of cryptographic primitives that cover a full suite of security features. Hash functions (SHA-2 and SHA-3), as well as Post-Quantum Cryptography accelerations are incorporated for Confidentiality, and a Physical Unclonable Function to ensure Integrity. The Availability aspect is also considered during the design and implementation phases of each Root-of-Trust component. Each module is designed separately and after their individual validation they are integrated into the Root-of-Trust, demonstrating its utility in different use cases.

In summary, the goals address with this dissertation are the following:

- Design of a novel Physical Unclonable Function based on a generally undesired phenomena called Random Telegraph Noise. This innovative approach has been protected by a patent.
- Design for both families of hash functions in the SHA-2 and SHA-3 standards.
- Design of hardware accelerators for Post-Quantum Cryptography algorithms.
- The validation of the proposed hardware designs through the following implementations:
  - A mixed-signal Application-Specific Integrated Circuit integration for the implementation of a novel Physical Unclonable Function, based on Random Telegraph Noise, in a 65-nanometer advanced lithographic node widely used in Complementary Metal-Oxide-Semiconductor fabrication.
  - The full digital implementation of hash functions on programmable logic included in commercial Systems-on-Chips. To ease its re-use, they are encapsulated in Intellectual-Property modules and interconnected to embedded processors. A set of drivers and tests is also provided to facilitate the integration into embedded software running on the processors.
  - A full-digital Application-Specific Integrated Circuit integration of the SHA-256 in a 65 nanometer Complementary Metal-Oxide-Semiconductor technology.
  - The hardware Post-Quantum Cryptography accelerators are implemented following a hardware/software co-design methodology using Systems-on-Chips.
- Analysis of the performance of each implementation and its comparison with similar approaches in the state-of-the-art.

- Integration of different crypto primitives in a hardware Root-Of-Trust on a Systems-on-Chip and validate it in several use cases.

The research activities of this dissertation have been funded by the following projects: VIGILANT ("The Variability Challenge in Nano-CMOS: From device Modeling to IC Design for Mitigation and Exploitation", PID2019-103869RB-C31) founded by the Spanish government, and SPIRS ("Secure Platform for ICT systems Rooted at the Silicon manufacturing process", GA: 952622) founded by the European Commission. On the one hand, the participation in VIGILANT project opened the possibility to explore the inherent variability in the Complementary Metal-Oxide-Semiconductor field to generate unique identifiers. On the other hand, the collaboration in the SPIRS platform has enabled the use of hardware-developed modules in cybersecurity environments through the Root-Of-Trusts.

The dissertation is organized in five chapters. The first chapter addresses the context of this dissertation among cybersecurity field. The Root-Of-Trust selection design leads to the next three chapters divided in each crypto primitive design. The selected use cases are shown in the next chapter in which the final Root-Of-Trust design is also presented. The final conclusions are summarized in the last chapter.

Table of contents

List of Figures

List of Tables

List of Acronyms

AMS	Analog/Mixed-Signal
ANSI	American National Standards Institute
ASB	Analog Sensing Block
ASIC	Application Specific Integrated Circuit
AU	Arithmetic Unit
AXI	Advanced eXtensible Interface
BRAM	Block Random Access Memory
BTI	Bias Temperature Instability
CAD	Computer Aided Design
CAVP	Cryptographic Algorithm Validation Program
CIA	Confidentiality, Integrity, and Availability
CMOS	Complementary Metal-Oxide-Semiconductor
CPU	Central Process Unit
DLP	Discrete Logarithm Problem
DMA	Direct Memory Access
DPA	Differential Power Analysis
DUT	Device Under Test
ENISA	European Union Agency for Cybersecurity
FF	Flip-Flop
FIFO	First In, First Out
FIPS	Federal Information Processing Standard
FPGA	Field Programmable Gate Array
HDL	Hardware Description Language
HLS	High Level Synthesis
HMAC	Hash-based Message Authentication Code
HPS	Hoffstein, Pipher, and Silverman
HRSS	Hülsing, Rijnveld, Schanck, and Schwabe
HW	Hardware

IA	Instrumentation Amplifier
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IoT	Internet of Things
IP	Intellectual Property
KEM	Key Encapsulation Mechanism
LFSR	Linear Feedback Shift Register
LSB	Least-Significant Bit
LUT	Look-Up Table
MAC	Message Authentication Code
MCF	Maximum Current Fluctuation
MPF	Maximum Parameter Fluctuation
MSB	Most-Significant Bit
MUX	Multiplexer
MVF	Maximum Voltage Fluctuation
NIST	National Institute of Standards and Technology
NTRU	Nth-degree Truncated polynomial Ring Unit
NVM	Non-Volatile Memory
OTA	Operational Transconductance Amplifier
PDH	Peak Detection and Hold
PEX	Parasitic EXtraction
PISO	Parallel-In Serial-Out
PKCS	Public-Key Cryptography Standards
PL	Programmable Logic
PoC	Proof-of-Concept
PQC	Post-Quantum Cryptography
PS	Processing System
PUF	Physical Unclonable Function
PYNQ	Python Productivity for Zynq
RAM	Random Access Memory
RO	Ring Oscillator
RoT	Root-of-Trust
RSA	Rivest, Shamir and Adleman
RTL	Register Transfer Level
RTN	Random Telegraph Noise
SCA	Side-Channel Attack

SHA	Secure Hash Algorithm
SIPO	Serial-In Parallel-Out
SoC	System-on-Chip
SPA	Simple Power Analysis
SRAM	Static Random Access Memory
SVP	Shortest Vector Problem
SW	Software
TRNG	True Random Number Generator
VLSI	Very Large Scale Integration

Chapter 1

Introduction

1.1 Cybersecurity Context

Cybersecurity is a crucial component of the digital ecosystem in Europe, as it plays a central role in safeguarding European interests, enabling the European Union (EU) economy to function at its full potential, and ensuring that citizens can safely and securely access online services [1]. This encompasses a wide range of sensitive information, including personally identifiable information, protected health information, personal data, intellectual property, and governmental and industry information systems [2]. The impact of cybersecurity is especially relevant in the following strategic points:

- *Critical infrastructures*: As society increasingly depends on digital infrastructure, it is essential to investigate cybersecurity to protect critical systems like power grids, healthcare systems, and financial institutions [3].
- *Data Privacy Preserving*: As data privacy concerns continue to mount, it is increasingly important for organizations to investigate cybersecurity in order to comply with regulations and safeguard sensitive information from unauthorized access [4].
- *Economic and National Security Implications*: Cybersecurity incidents can have severe economic and national security implications. Investigating and understanding these aspects helps in developing strategies to mitigate risks [5].
- *Global Communications*: In a globally interconnected world, investigating cybersecurity is essential for international collaboration in addressing cyber threats and ensuring the security of shared digital resources [6].

There are always security breaches that attackers could exploit with the potential to jeopardize the safety and well-being of individuals. There are some recent examples in which some security aspects of citizens have been compromised either in international, national or local context:

- The US State of Maine's data was accessed due to the MOVEit Transfer breach, exposing 1.3 million individuals [7].
- The hacked phone of the Prime Minister of Spain, Pedro Sánchez, using Pegasus exposed more than 3GB of sensitive information [8].
- The attack to the Seville town-hall that paralyzed all processes during more than a month, even affecting to the salary payment of many workers [9].

To this end, there are organizations or agencies that are continuously investigating and reporting new procedures or standards that improve the security of systems. The *European Union Agency for Cybersecurity* (ENISA) [10] is a center of expertise for cybersecurity in Europe in different areas as shown Figure 1.1. It works closely with EU member states and the private sector to develop and promote cybersecurity standards, best practices, and guidelines. ENISA also provides support to EU member states in the event of a cyber crisis. The *National Institute of Standards and Technology* (NIST) [11] is a non-regulatory agency of the United States Department of Commerce that promotes innovation and industrial competitiveness by

Figure 1.1 The European Cybersecurity Skills Framework (ECSF) of ENISA [10].

advancing measurement science, standards, and technology. NIST's cybersecurity program focuses on developing standards, guidelines, and best practices to manage cybersecurity-related risk. The *Internet Engineering Task Force* (IETF) is a large open international community of network designers, operators, vendors, and researchers concerned with the evolution of the Internet architecture and the smooth operation of the Internet. IETF develops and promotes voluntary Internet standards, in particular the standards that comprise the Internet protocol suite (TCP/IP) [12]. Moreover, in the national framework there is the *Centro Criptológico Nacional* (CCN) that is a Spanish government agency responsible for coordinating the actions of different government agencies that use cryptographic methods or procedures, ensuring the security of information technologies in that area [13].

1.2 The deployment of an information system: the CIA Triad

The CIA Triad, illustrated in Figure 1.2, was first mentioned in a NIST publication in 1977 [14]. This sets a fundamental cybersecurity model that forms the basis for the development of security systems. The three letters in "CIA" triad stand for **Confidentiality**, **Integrity**, and **Availability**:

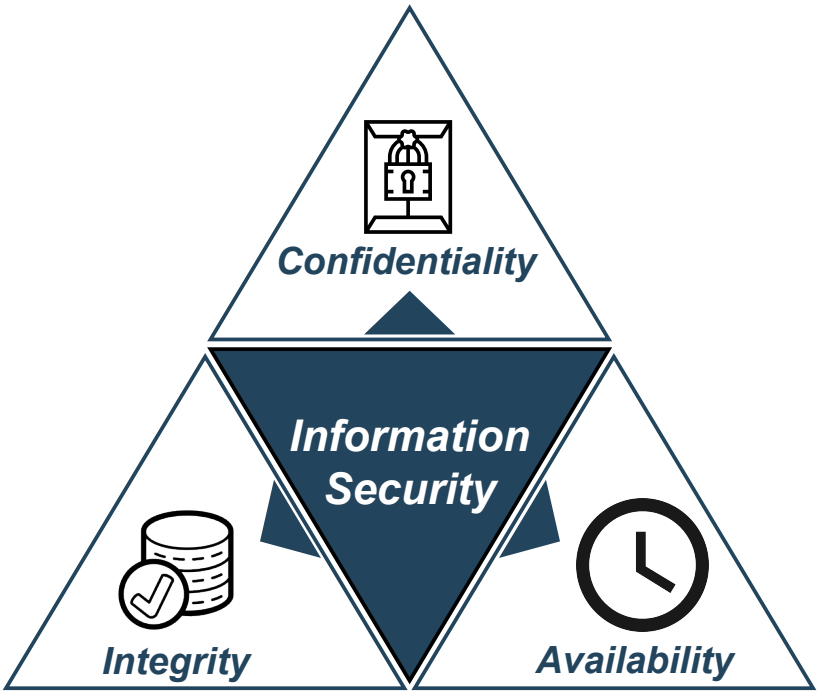


Figure 1.2 The CIA Triad: Confidentiality, Integrity and Availability

- **Confidentiality**: safeguarding sensitive information from unauthorized access. This involves implementing measures such as encryption, access controls, and secure authentication mechanisms. By ensuring confidentiality, organizations can protect proprietary information, personal data, and classified materials [15].

- **Integrity**: guarantees the accuracy and trustworthiness of data, preventing unauthorized modifications or corruption. Technologies such as cryptographic hash functions and digital signatures play a crucial role in ensuring data integrity. By maintaining the integrity of information, organizations can foster trust among users and stakeholders [16].

- **Availability**: ensuring that information and services are accessible when needed. This involves implementing redundancy, failover mechanisms, and disaster recovery plans to mitigate the impact of system failures, natural disasters, or cyberattacks. By prioritizing availability, organizations can minimize downtime and maintain continuous operations [17].

Since the CIA Triad was proposed, new menaces in the field of cybersecurity have emerged. The change in the nature of cyber threats leads to the importance of adapting to these new challenges. For that, it is necessary to explore specific security models, frameworks, and technologies in greater detail.

1.3 Root-of-Trust. From hardware perspective

The establishment of a Root-of-Trust (RoT) [18] in cybersecurity is intricately linked to the fundamental principles of the CIA Triad. The RoT serves as the cornerstone for building a secure computing environment, aligning with each aspect of the triad. As foundational security primitives, RoTs encompass hardware, firmware, and/or software components, delivering a suite of essential, trust-dependent functions.

Several organizations released standards that define sets of security services within RoT:

- NIST standardizes integrity, measurement, reporting, storage, update, and verification services [19].

- ISO 7498-2 manages authentication, access control, confidentiality, integrity, and non-repudiation services [20].

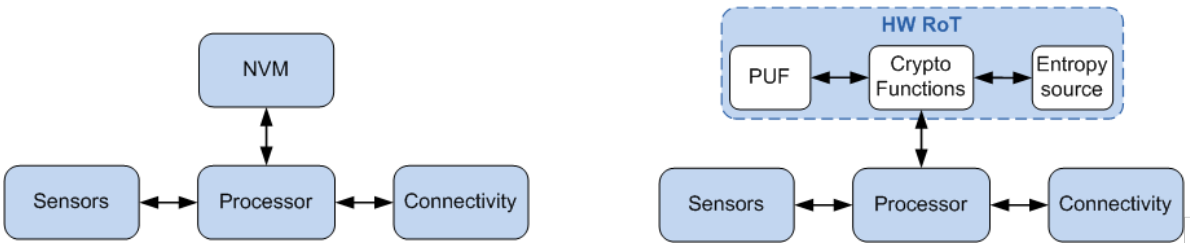
- The Trusted Computing Group works in the measurement, reporting, and storage services [21].

In the realm of **Confidentiality**, the RoT plays a pivotal role by securely storing cryptographic keys and facilitating secure key management processes. By safeguarding these keys, it ensures that sensitive data remains confidential, accessible only to authorized entities with the necessary credentials. Some examples include the use of symmetric or asymmetric algorithms in the context of public key cryptography, hash functions such as SHA-256 or message verification functions based on hashes.

Moving on to **Integrity**, the RoT acts as a guardian of the system's trustworthiness. For example, during the boot process, the RoT verifies the integrity of essential components, including firmware and the operating system. If any tampering or unauthorized modifications are detected, the RoT can halt the boot sequence, preventing the system from compromising its integrity. This capability is crucial in maintaining the integrity of the system's state and preventing malicious actors from manipulating critical components. The core of the security enhance of this part will be the addition of Physical Unclonable Functions (PUFs). They can provide unique identifiers linked to the underlying hardware, enhancing authentication, and contributing to various security functions critical for maintaining the confidentiality, integrity, and trustworthiness of the computing environment.

Lastly, in the context of **Availability**, the RoT indirectly contributes by reducing the risk of compromised and unauthorized access. By ensuring the integrity and confidentiality of the system, the RoT mitigates the likelihood of security incidents that could lead to service disruptions, aligning with the overarching goal of sustaining system availability.

Hardware RoTs, illustrated in Figure 1.3b, hold a preference over their software counterparts [22], shown in Figure 1.3a, due to their inherent immutability, reduced susceptibility to attacks, and a track record of more dependable performance. Certain methods aim to enhance the safety of cryptographic key storage by replacing Non-Volatile Memories (NVMs) with PUFs. Their reliability instills a higher level of confidence in their capability to execute trusted functions seamlessly. While software RoTs offer the advantage of swift deployment across diverse



(a) SW RoT approach (b) HW RoT approach

Figure 1.3 Different Root-of-Trust approaches

platforms, their reliance on mutable code introduces a trade-off in terms of the robust assurance provided by their hardware counterparts.

As the cybersecurity threats constantly evolve, the hardware RoT must be continually adapted to address the latest challenges and advancements in the field. This adaptability is crucial for ensuring that the RoT remains resilient against emerging cybersecurity threats, ranging from sophisticated attacks to novel vulnerabilities. Regular updates and enhancements to the hardware RoT's design and functionality are imperative to meet the demands of an ever-changing cybersecurity environment.

1.4 The Quantum menace

Quantum computers represent a groundbreaking advancement in computational capabilities with the potential to revolutionize various fields [23]. Unlike classical computers that use bits to represent either a 0 or a 1, quantum computers leverage quantum bits or qubits, which can exist in multiple states simultaneously due to the principles of superposition and entanglement. This unique feature enables quantum computers to perform parallel computations and solve certain problems exponentially faster than classical computers.

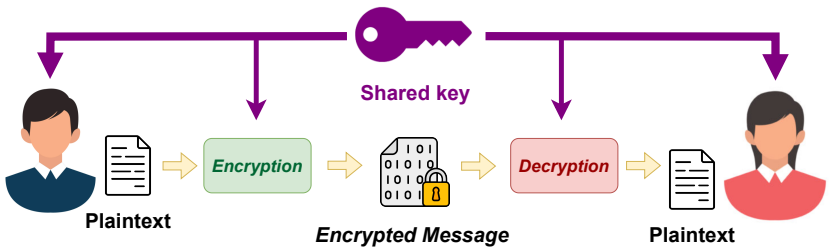
Some cases of advent of quantum computing is the IBM Quantum Platform [24] and Google Quantum Supremacy [25]. IBM Quantum Platform is a cloud-based quantum computing platform that provides access to IBM's quantum processors and simulators. They count with a Quantum development Roadmap, making clear that the advance of quantum computing is unstoppable. Meanwhile, Google Quantum Supremacy is a term used to describe the ability of a quantum computer to perform a calculation that would be practically impossible for a classical computer to perform in a reasonable amount of time.

Although it is not clear when a quantum computer will be powerful enough to surpass the classical computer, the advent of quantum computing poses a significant challenge to traditional cybersecurity paradigms. They raise concerns about the potential obsolescence of widely-used cryptographic algorithms by performing complex computations at unprecedented speeds, threatening the security of current encryption methods based on the difficulty of certain mathematical problems. This directly threatens the security mechanisms based on symmetric and asymmetric cryptography.

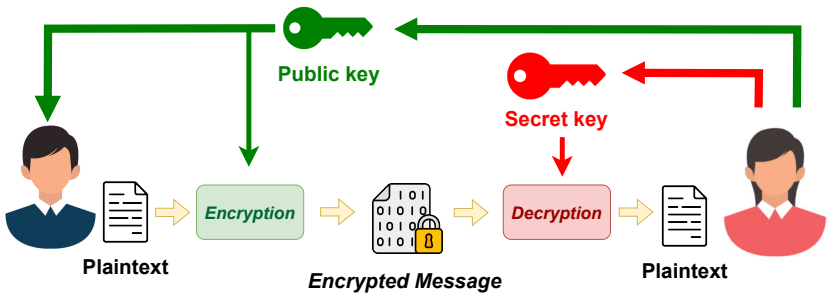
Symmetric cryptography employs a single secret key for both encryption and decryption processes as shown Figure 1.4a. The efficiency and speed of symmetric key algorithms make them well-suited for securing data communication and storage. The most popular and used symmetric algorithm is Advanced Encryption Standard (AES) [26] which is specified by the NIST under standard FIPS PUB 197 [27]. In the quantum computing context, Grover's

algorithm [28] poses a theoretical threat, suggesting that a quantum computer could search through the key space quadratically faster, potentially reducing the effective key length. That is, challenging the security assumptions of symmetric encryption algorithms, which rely on the difficulty of brute-force attacks. The strategy followed by NIST was to duplicate the keys length from 128-bits to 256-bits.

On the other hand, asymmetric cryptography, also known as public-key cryptography, is a type of cryptographic scheme where distinct pairs of keys are employed for encryption and decryption as shown Figure 1.4b. The widely used RSA (Rivest–Shamir–Adleman) algorithm [29] is a prominent example. In asymmetric cryptography, one key in the pair is designated as public and can be openly shared, while the other remains private. Information encrypted with the public key can only be decrypted using the corresponding private key. NIST has released several standards that use the RSA algorithm such as the NIST SP 800-56B [30]. The security of the scheme relies on the difficulty of factoring large numbers or solving discrete logarithm problems, which can be efficiently solved by quantum computers using algorithms like Shor's algorithm [31]. To address this challenge, the field of Post-Quantum Cryptography (PQC) is emerging, aiming to develop cryptographic algorithms that remain secure even in the era of quantum computing. To that end, NIST initiated a PQC algorithm standardization contest in 2017 [32] whose aim was to define the first PQC standard. The first draft standard of those selected algorithms was published in August 2023 [33].



(a) Symmetric cryptography scheme



(b) Asymmetric cryptography scheme

Figure 1.4 Cryptography schemes

1.5 Dissertation Overview

Internet of Things (IoT) has set a new paradigm where interconnected devices play the main role [34]. IoT, characterized by a network of embedded devices with sensors, processing units and connectivity to the internet, brings unprecedented opportunities for efficiency and convenience [35]. However, the intersection of IoT and hardware security poses unique challenges, demanding robust measures to safeguard sensitive data and mitigate potential threats to interconnected systems [36]. Not only that, but implementations on, for example, wearable devices impose severe restrictions that require low power consumption and small sizes [37].

Intense competition among companies to bring electronic devices in the first place has demanded the development of strategies to reduce time-to-market. The re-usability of Intellectual Property (IP) cores and an automatic design methodology reduce significantly the duration of the design cycle of embedded systems [38] in the IoT ecosystem. However, the need to deploy functional systems in a short time has sometimes led to serious security problems by presenting information leaks that could reveal confidential data. Two well-known examples are Spectre and Meltdown, which are vulnerabilities that affect modern microprocessors. Spectre is a vulnerability that uses side-channel timing attacks to extract private data based on speculative execution resulting from a branch miss-prediction [39]. Meltdown is related to a micro-architectural attack that exploits out-of-order execution to reveal the contents of kernel memory in many Intel and some ARM processors [40].

In light of the confidential data leaks observed in embedded systems, the hypothesis of this dissertation is that the design of dedicated hardware for a RoT could be the most suitable option to secure the embedded systems widely adopted in the IoT ecosystem. To corroborate this, this dissertation will pursue the following goals:

- To explore, design and implement the elements of a hardware RoT.
- To analysis the most suitable cryptographic primitives to ensure information security around the three key principles: confidentiality, integrity and availability.
- To validate the proposed RoT implementation on a System-on-Chip (SoC) that combines one embedded processor with Programmable Logic (PL) for prototyping.
- To provide use cases that corroborate the feasibility of the proposed solutions in message verification and PQC cryptosystems.
- To design digital and mixed-signal integrated circuits of cryptographic primitives to achieve high-performance implementations.

In order to achieve these objectives, the structure of this dissertation is organized as follows:

- A novel PUF design is presented in Chapter 2 based on generally undesired phenomena called Random Telegraph Noise (RTN). After performing several evaluations, a mixed-signal circuit was implemented into an Application Specific Integrated Circuit (ASIC).
- Given the crucial role that hash functions have become in the field of cryptography, two such functions (SHA-2 and SHA-3) are studied and implemented in Chapter 3. For that, it is provided a set of drivers and tests that eases the integration into software implementations. Subsequently, one of these hash functions was incorporated in accordance with the digital flow in a Very Large Scale Integration (VLSI) circuit.
- Chapter 4 addresses the potential for improving the performance of PQC algorithms, with the NTRU (Nth-degree Truncated polynomial Ring Unit) algorithm chosen for this purpose. An exhaustive analysis was conducted to determine the most suitable NTRU implementation for the IoT context from the proposed options. Additionally, a comprehensive study was undertaken to explore the potential for extracting information from these implementations through Side-Channel Attacks (SCAs).
- Chapter 5 proposes two use cases that utilize a combination of various cryptographic primitives included in the RoT suggested in this dissertation.
- The dissertation ultimately culminates in Chapter 6, where the conclusions drawn from the results and conclusions obtained of each chapter.

Chapter 2

Physical Unclonable Functions

2.1 Introduction

PUFs have emerged as a potential solution to build trusted anchors that provide secure hardware solutions for consumer and industrial IoT devices [41]. Based on their properties, PUFs can be used to generate unique identifiers that facilitate device authentication to prevent spoofing and counterfeiting [42]. They also introduce an extra hardware-based layer for building lightweight encryption schemes, as they can be used to obfuscate the secret keys used by ciphers, ensuring the confidentiality of data exchanged by the electronic device in which the PUF is embedded or attached [22]. In addition, PUFs can provide seeds to be used in the creation of public and private key pairs for public-key cryptography, increasing system security by avoiding the need to share secret keys [43].

A PUF works as a black box that maps an input challenge sequence to an output response conforming the so-called challenge-response pair, as Figure 2.1 illustrates. For that, it should

Figure 2.1 Conceptual representation of a PUF

feature three characteristics: *uniqueness* (different PUF instances should return different responses for the same challenge), *unpredictability* (the response cannot be anticipated) and reliability (the same response should be obtained for the same challenge applied to the same PUF instance, that is, the PUF response must be robust or stable over time and varying environmental conditions) [44]. The quality of the PUF, according to these three features, depends ultimately on the specific PUF implementation, its behavior in the presence of impeding factors (like noise or aging for silicon PUFs), and the response post-processing [45].

All reported PUFs use at its very core a source of entropy [46]: for example, silicon PUFs use the inherent randomness of the manufacturing process (e.g., inducing variations of the transistor's threshold voltage) to implement the challenge-response pair. For instance, Arbiter PUFs are based on time delay variations [43][47], transistor-pair PUFs utilize the mismatch-induced variations in the subthreshold drain currents of transistors [43, 47, 48], Ring Oscillator (RO) PUFs use the variability in oscillation frequencies [49], and Static Random Access Memory (SRAM) PUFs are based on the cells' power-up value (set by the subtle differences in the threshold voltages of the transistors in the cross-coupled inverters) [50]. This work explores the implementation of a different type of PUF based on Complementary Metal-Oxide-Semiconductor (CMOS) technology where the entropy source comes from a physical phenomenon whose impact is otherwise undesired: the RTN. The application of this phenomenon is straightforward as RTN has proven to be resistant against aging.

This dissertation delves into the inception, evaluation, and final design of an RTN-based PUF in an ASIC, following an analog design flow. It begins with the birth of the idea for an RTN-based PUF, exploring the initial thought process, the theoretical underpinnings, and the creative sparks that led to its conception. The following part focuses on the rigorous evaluation of the RTN-based PUF. This involves a detailed analysis of its performance, reliability, and robustness under various conditions. The evaluation process is critical in identifying potential improvements and understanding the limitations of the RTN-based PUF. The final part discusses the integration of the RTN-based PUF into an ASIC. This section outlines the steps taken to incorporate the PUF into an analog design flow, ensuring its compatibility and functionality within the ASIC. It also addresses the challenges encountered during the circuit design process and the solutions implemented to overcome them.

2.2 RTN-based PUF

As it was above mentioned, two of the most implemented PUFs among Silicon-based PUFs are those based on SRAMs or ROs. In any case, both PUF implementations require, typically, a large number of transistors for their implementation, leading to high area overhead. In devices

where such area constraints are very demanding (i.e., wearable devices), this can be a problem. To reduce the area used as much as possible, PUFs whose rationale is the exploitation of some feature and the comparison of said feature in a pair of transistors have been proposed. Both the TCO-PUF ("Two Choses One" PUF) [47] and the SCA-PUF (Subthreshold Current Array PUF) [51] are relevant in this type of PUF architectures. These designs use the comparison of the generated current level in the subthreshold region of the selected transistors. One main drawback of these approaches is that the entropy source is not immune to aging and temperature-induced variations, thus potentially impacting the PUF response as well.

The PUF presented in this dissertation explores the implementation of a different type of CMOS-based PUF where the entropy source comes from a physical phenomenon named as RTN. The utilization of this phenomenon as a foundation for PUF development is justified by several potential benefits: (1) the requirement of only one transistor and certain biasing conditions for RTN manifestation suggests that it would likely require less area for implementation compared to other solutions [52]; and, (2) the degradation due to aging is significantly reduced as the bias conditions of the transistor do not need to reach their nominal operating voltages to observe the RTN, ensuring the longevity and reliability of the PUF [53].

Although it will be deeply detailed later, in practice, RTN is observed as discrete and random shifts in the drain current of a transistor, where it is essential to highlight that two identically designed transistors may show different RTN-induced current shifts. Very few works have been reported that use RTN as underlying entropy source:

- The solutions described in [54] and [55] exploit a phenomenon called BTI (Bias Temperature Instability) in CMOS transistors. Even though the physical mechanism known as RTN is mentioned in these works, the main method presented is based on the previous application of high voltages to the transistor gate, thus creating BTI discharging events (rather than pure RTN events) that really are the core of the proposed PUF. A disadvantage of these two approaches is that the stress phases at voltages higher than the nominal one that need to be applied cause a premature aging of the PUFs, making them potentially unreliable over time.

- In the work presented in [56] the response of the PUF is extracted from the fluctuations that the RTN mechanism causes in the oscillation frequencies of the ROs. By taking two identical ROs and comparing the number of frequency fluctuations per unit time that each RO manifests as a result of the presence of RTN in the transistors of the inverter chains, a response can be obtained. However, it is difficult to determine whether these fluctuations are really due exclusively to RTN and not to some type of electronic noise.

2.2.1 The entropy source: RTN

The RTN can be defined as a transient phenomenon that occurs in the threshold voltage of transistors in the form of discrete and random shifts. In CMOS transistors, it can be observed through two parameters: in the form of voltage (through variations in the threshold voltages) or in the form of current (through drain current variations caused by these changes in the threshold voltages). The origin of this phenomenon lies in the existence of defects in the channel-oxide interface of the transistor and in the capture and emission events of charge carriers in these defects [57].

The parameters that characterize the RTN phenomenon are the number of defects in the transistor, the amplitude of the change in the threshold voltage ΔV_{th}, that will cause a shift in the drain current ΔI_D, that is associated to the trapping/detrapping events of each defect, and their time constants (the capture time (t_c) and the emission time (t_e)). Both, the number of defects and the dynamics of carrier capture/emission, are unique and differentiating characteristics of each transistor, which means that the RTN phenomenon can be potentially used to generate unique identities through the PUF.

An example of variations in a drain current generated by the variation in the threshold voltage with RTN is shown in Figure 2.2. This example corresponds to a simple case in which the current trace displays only one detectable RTN defect, which causes its current to alternate between two levels. The time constants, τ_e and τ_c, of this defect are the average of the corresponding times-to-emission t_{ei} and times-to-capture t_{ci}, respectively. In general, there are more than just a single RTN defect.

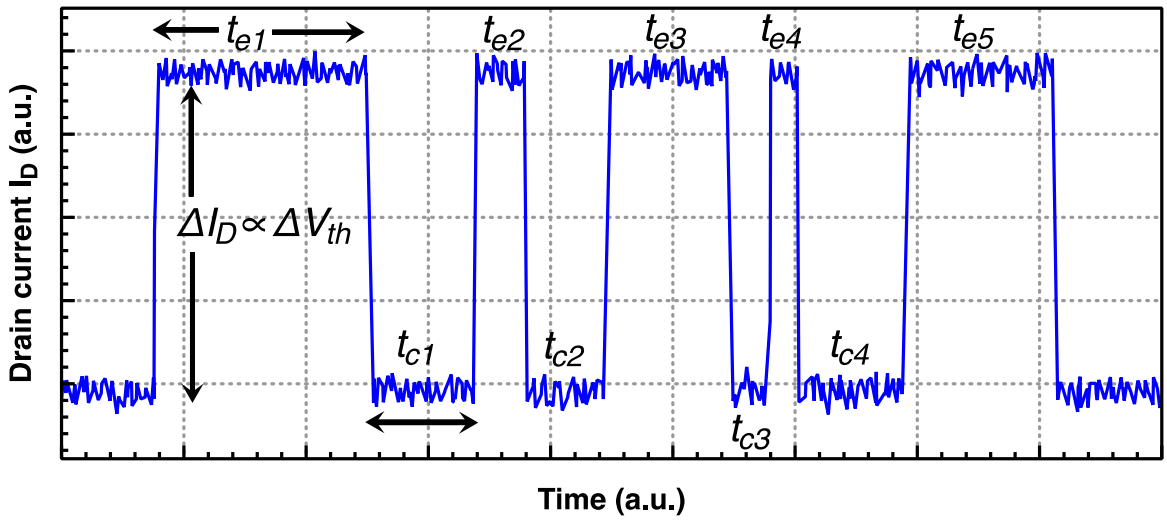


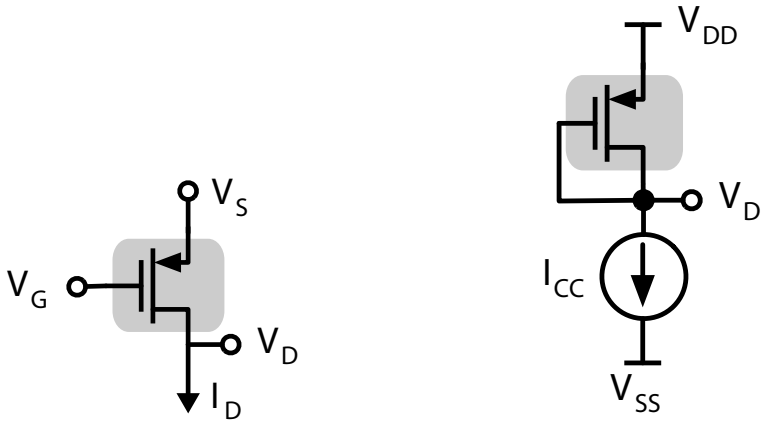
Figure 2.2 Illustrating RTN in the drain current (generated from variations in the threshold voltage) of a transistor where the RTN parameters have been indicated.

It is essential to highlight the fact that two identically designed transistors may show different RTN-induced threshold voltage shifts. These variations between transistors are the entropy source of the RTN-based PUF presented in this dissertation.

2.2.2 How to extract information: The Maximum Parameter Fluctuation

The design of this PUF requires a metric that can capture, in a comprehensive manner, the amount of RTN present in each transistor and that is amenable to evaluation using the simplest possible circuitry. This metric is the so-called Maximum Parameter Fluctuation (MPF). As explained above, RTN fluctuations can be observed either in the voltage at the gate node or in the transistor's drain current. The concept of MPF captures both fluctuations (i.e., the observed parameter P in the MPF metric can be either voltage or drain current). The current version of the metric, *Maximum Current Fluctuation* (MCF) and the voltage version of the metric, *Maximum Voltage Fluctuation* (MVF) were first reported in [58] and in [59], respectively, as methods to encapsulate all the RTN information.

To observe the RTN-induced fluctuations, the first method uses the drain current, I_D, set from a certain source voltage, V_S, gate voltage, V_G, and drain voltage, V_D, illustrated in Figure 2.3a for the case of PMOS transistors. The second method, known as constant-current method [60] may be used to obtain the voltage fluctuations at the gate node. This method consists in using a current source, I_{CC}, at the drain of the transistor, short-circuiting the gate and the drain nodes (in a typical diode configuration) and observing the RTN-induced fluctuations of the voltage at the drain node, V_D (which is equivalent to the gate node) as Figure 2.3b shows



(a) Current-mode PMOS configuration (b) Voltage-mode PMOS configuration

Figure 2.3 Methods to extract RTN information

for the case of PMOS transistors. The NMOS transistors version follows the same strategy of configuration.

The MPF can be calculated at any time instant, t', defined by $t_0 \leq t' \leq t_f$, where t_0 and t_f is the initial and final time of the measurement window. For that, it is required to compute the Cumulative Maximum Parameter (CMAXP), which is split in the Cumulative Maximum Voltage (CMAXV), for voltage, and the Cumulative Maximum Current (CMAXC), for drain current. Equation 2.1 shows the CMAXV and CMAXC formulation.

$$CMAXV(t') = \max_{\forall t \in [t_0,t']} V_D(t) \; ; \; CMAXC(t') = \max_{\forall t \in [t_0,t']} I_D(t) \tag{2.1}$$

Similarly, the Cumulative Minimum Voltage (CMINV), and the Cumulative Minimum Current (CMINC), can be defined as Equation 2.2 shows.

$$CMINV(t') = \min_{\forall t \in [t_0,t']} V_D(t) \; ; \; CMINC(t') = \min_{\forall t \in [t_0,t']} I_D(t) \tag{2.2}$$

The MPF at t' is then defined as the difference between $CMAXP(t')$ (either $CMAXV(t')$ or $CMAXC(t')$) and $CMINP(t')$ (either $CMINV(t')$ or $CMINC(t')$) as Equation 2.3 shows.

$$MPF(t') = CMAXP(t') - CMINP(t') \tag{2.3}$$

In other words, the MPF value will be determined by the cumulative impact of RTN either on the transistor voltage at the drain node or on the transistor current (both related to threshold voltages shifts) because of the trapping/detrapping events observed in a time interval defined by $t_{MPF} = t_f - t_0$, (defined identically for voltage, t_{MVF}, or current, t_{MCF}). Here it is necessary to keep in mind that the longer this time, the more RTN information is captured in the MPF.

An example of MCF measurement (the same can be applied to MVF) is shown in Figure 2.4. MPF is a continuous function of time but to carry out the required comparison needed in the PUF, only the value of the MPF at the end of the time interval (t_f) will be used (considering $t_0 = 0$). The MPF metric is a non-negative, ever-increasing function that, over time, will "capture" all RTN-induced variations both in the drain voltage and in the drain current (reflecting the number of defects, the shifts caused by each defect as well as its temporal characteristics).

2.2.3 Conceptual architecture of the RTN-based PUF

The PUF design should fulfill a way to extract the information contained in the MPF (using the RTN as entropy source) and somehow obtain a bit as response. For that, four designs have been proposed whose operation principle is the comparison between two MPF values. The first proposed design for the PUF can use an architecture where the MPF of only one transistor is

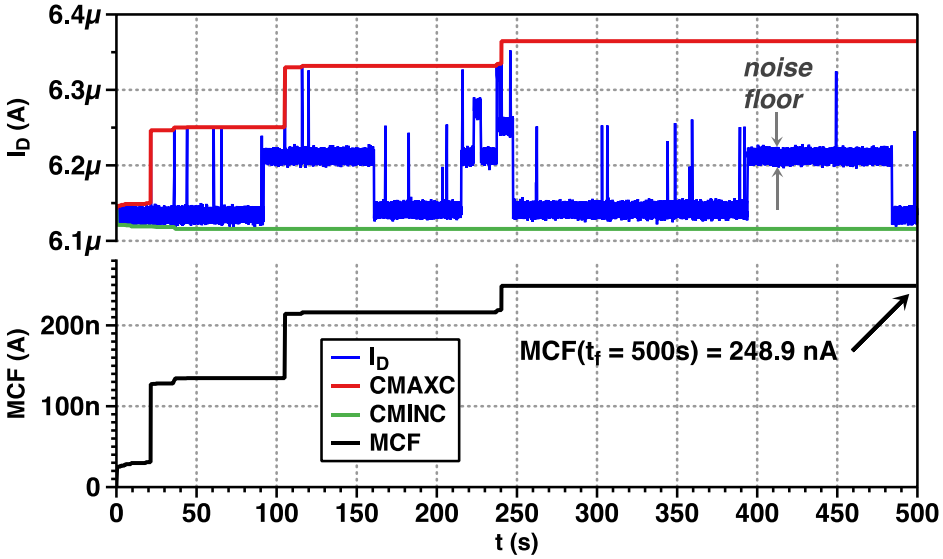
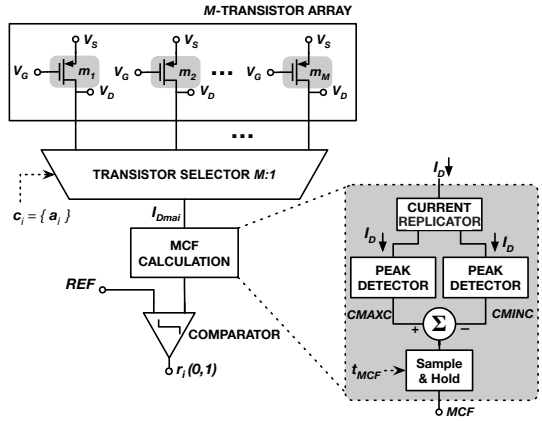


Figure 2.4 A drain current trace (top) with CMAXC, CMINC and MCF (bottom) computed over a time interval t_{MCF} of 500s.

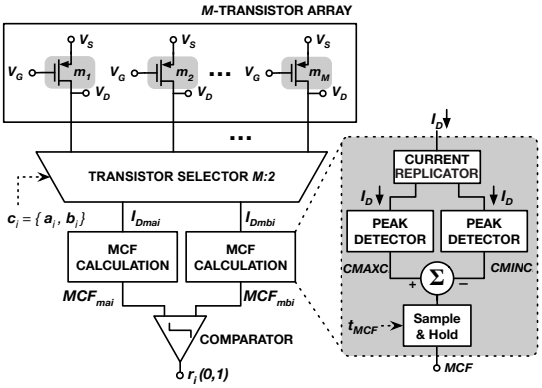
compared to an MPF reference. A differential architecture is also possible, where the MPF calculation is carried out on a pair of transistors and then compared. Both architectural options can exploit either voltage or current. The advantage of using differential architectures over non-differential ones is that differential design can mitigate and cancel out first-order environmental dependencies such as aging, temperature, and supply voltage since these should affect the two identical structures similarly. These designs are shown in Figure 2.5 and summarized in the following:

- Using the transistors' current configuration to obtain an MCF value and comparing it with a reference. It is shown in Figure 2.5a.
- Using the transistors' current configuration to obtain an MCF value and comparing it with the MCF of another transistor. It is shown in Figure 2.5b.
- Using the transistors' voltage configuration to obtain an MVF value and comparing it with a reference. It is shown in Figure 2.5c.
- Using the transistors' voltage configuration to obtain an MVF value and comparing it with the MVF of another transistor. It is shown in Figure 2.5d.

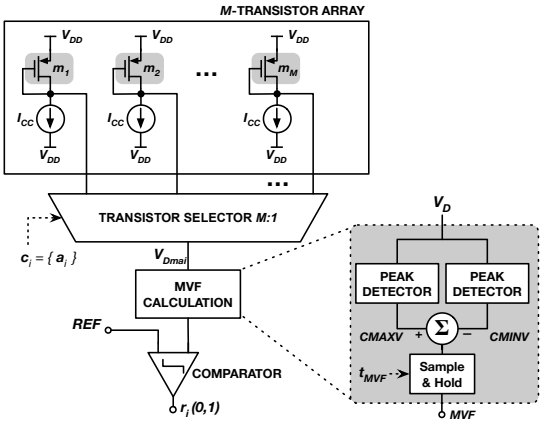
In the non-differential architecture (Figure 2.5a and Figure 2.5c), a single transistor, m_a, specified by a challenge c_i (which specifies the address in the array of the specific transistor selection and that is used as PUF input) from an array of a number, M, of equally designed



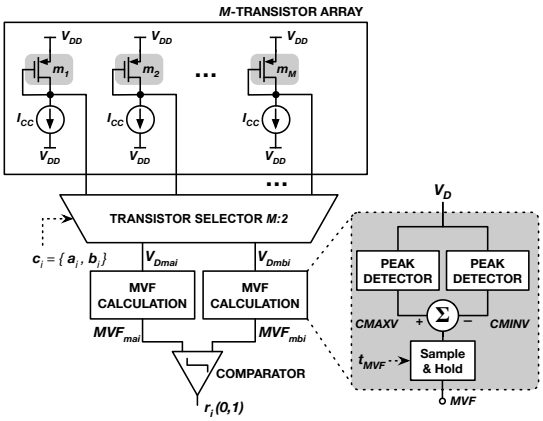
(a) Current-mode configuration comparing with a reference.



(b) Current-mode configuration comparing with another transistor.



(c) Voltage-mode configuration comparing with a reference.



(d) Voltage-mode configuration comparing with another transistor.

Figure 2.5 The different architectures proposed for the RTN-based PUF.

transistors (same type, same size), is selected using an $M:1$ multiplexer. In this case, the core element of the PUF is a single transistor. All transistors must have the same biasing conditions. Once the transistor is selected, the parameter chosen (either voltage or current) is used for the MPF calculation module, where the MPFs is computed for a preset time interval t_{MPF}. Then, the result is compared to an MPF reference, determining an output: if $MPF_{m_a} > MPF_{REF}$, the output given is "1" ("0", otherwise). This output can be used as, at least a basis of, a PUF response r_i. To obtain an n-bit response $R = (r_1, r_2, \ldots, r_n)$, a sequence of n challenges $C = (c_1, c_2, \ldots, c_n)$ is given to the PUF device.

Alternatively, the differential architecture (Figure 2.5b and Figure 2.5d), starts from an array with an even number, M, of equally designed and biased transistors. In this case, the core element of the PUF is a pair of transistors, m_a and m_b, specified by a challenge c_i (now, this

challenge c_i specifies the addresses in the array of the transistors in the selected pair), which is selected using an $M:2$ multiplexer. Once the pair is selected, the parameters chosen (either voltage or current of each transistor) are used for the MPF calculation module. The results, MPF_{m_a} and MPF_{m_b}, are then compared to decide which has higher MPF value, thus determining an output: if $MPF_{m_a} > MPF_{m_b}$, the output given is "1" ("0", otherwise). This output can be used at least as a basis of a PUF response r_i. To obtain an n-bit response $R = (r_1, r_2, \ldots, r_n)$, a sequence of n challenges $C = (c_1, c_2, \ldots, c_n)$ is given to the PUF device.

Apart from the array, one MPF calculator (an MVF calculator for Figure 2.5c and 2.5d, and an MCF calculator for Figure 2.5a and 2.5b) is necessary to compute the MCF of each selected transistor (depending on the architecture). It receives either the drain voltage from the selected current-driven transistor (V_{Dmai}) or the drain current signal from the selected voltage-driven transistor (I_{Dmai}). At the end of the process, it returns the calculated MPF value (MVF_{mai} or MCF_{mai}). In the case of the differential architecture, two MPF calculators will be necessary. Additionally, the MPF calculator comprises:

- A current replicator for the case of MCF calculator in which the current signal is duplicated.
- A couple of peak detectors (connected to the current replicator only for the case of current-mode configuration) that receive the signals. One of them oversees the maximum, while the other detects the minimum, obtaining the maximum (CMAXP) and minimum (CMINP) accumulated parameter (voltage or current) signal respectively.
- A summation block, connected to the peak detectors, that subtracts the minimum signal from the maximum signal to obtain the MPF value (either the MVF or the MCF).
- A sample and hold block that controls the summation block and resets at $t = t_f$, making sure that the calculation is performed at each time interval, t_{MPF} (t_{MVF} or t_{MCF}). At the beginning of this time interval, both CMAXP and CMINP are reset to the initial values (in which $CMAXP = CMINP = V_D(t_0)|I_D(t_0)$). Thus, a different MPF can be obtained at each time interval t_{MPF}.

And finally, there is the MPF comparator, associated to the MPF calculators that receives the MPF signal from the voltage-driven transistor or current-driven transistor. It performs the comparison to a reference or with another MPF value, deciding which is larger and obtaining a one-bit response, r_i.

In summary, the challenge c_i can be a single digital word a_i for the non-differential architecture or a pair of digital words a_i and b_i for the differential architecture that select a specific transistor m_a or a specific pair of transistors m_a, m_b, respectively, from the array, and,

then the corresponding single-bit response r_i ("0" or "1") is obtained. The function that maps a challenge to a response is thus the difference in amount of RTN captured during a certain time interval, either as a difference with respect to the reference or between the two transistors, with this amount being measured by the MPFs.

A relevant parameter in these architectures (both non-differential and differential architecture) is the duration of the time interval used to compute the MPF, $t_{MPF} = t_f - t_0$. The longer this interval, the more RTN emission/capture events can be captured into the MPF metric, but, on the other hand, the longer it takes for the PUF device to return an n-bit response. A too short of a time interval can be an issue as well, as RTN may not be observed at all. In this regard, an additional constraint on the minimum value of t_{MPF} results from considering the time constants of the RTN defects and the ability of the MPF calculation module in capturing fast RTN events since any time-to-emission t_{ei} or time-to-capture t_{ci} below this ability threshold will go undetected.

Since this PUF is based on RTN, which is unique and unpredictable, it can be expected that the proposed PUF device will have a unique and unpredictable response as well. RTN is, nonetheless, a time-varying phenomenon. From one use to the next, there is a probability that a response bit flips. However, the use of the MPF metric decreases the flip probability since MPF represents but an "accumulation" of RTN and is thus expected to remain similar on different requests of the same challenge. Despite this, the reproducibility over factors like time or temperature can be further improved (thus minimizing the probability of a bit flip) by using a bit preselection method at the point of manufacture. The bit preselection method assesses the reproducibility of each bit (i.e., the reproducibility of the response of each transistor, in the non-differential architecture of each transistor pair, in the differential architecture) and the resulting ranking is used to take only the most stable available bits to form the PUF. The bit preselection method can be carried out for both voltage and current as the parameter chosen.

2.2.4 The bit selection method for the RTN-based PUF

To improve the quality of the PUF, during the manufacturing process, a bit selection method is proposed. This technique ranks each bit (e.g., each transistor pair) and the resulting ranking is used to select only the most stable bits to form the PUF. This selection is performed through the use of the probability (P). It is defined as shown Equation 2.4 where n_0 and n_1 are the number of zeroes and ones, respectively, obtained in the overall set of responses L. An ideal bit is determined by $P = 1$ (the response bit, from the MCF comparison of the two transistors in the pair, would always be either "1" or "0"). Figure 2.6 illustrates an example of a probability computation of one pair of transistors that works as the basis of this selection method. In this instance, the two upper plots correspond to the drain current of two transistors, m_a and m_b,

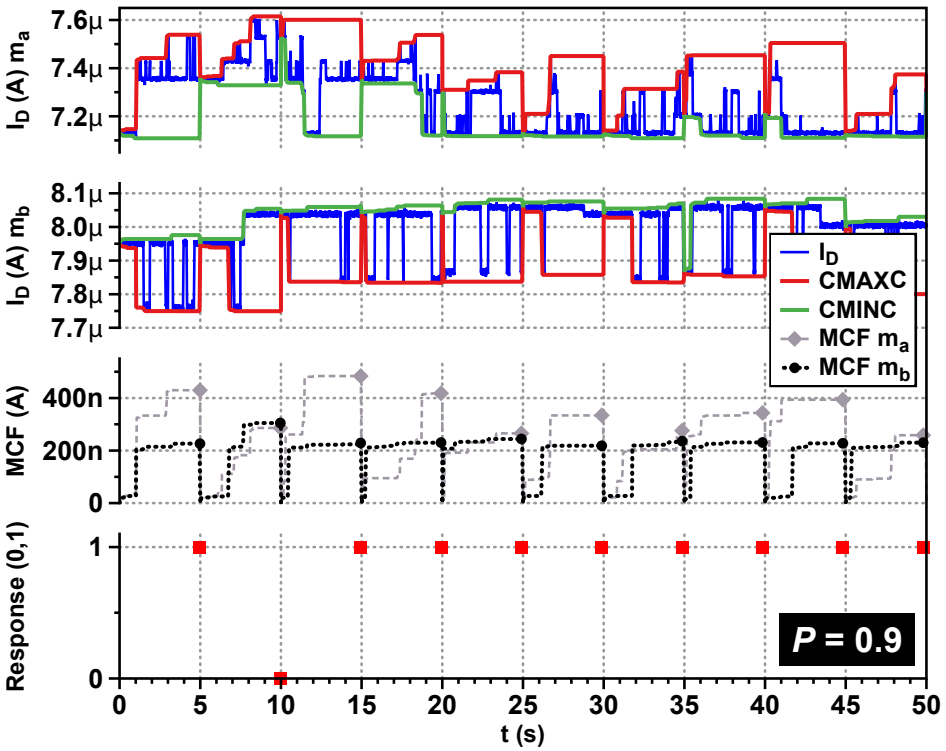


Figure 2.6 Illustrating probability calculation for the bit selection method: two current traces (top two plots), the corresponding MCFs ($t_{MCF} = 5s$) and the response ("0" or "1") from comparing the MCFs (bottom).

where CMAXC and CMINC have been calculated in ten intervals of $t_{MCF} = 5s$. The third plot represents the MCF computation for each transistor, while the fourth plot illustrates the bit response for each interval, resulting in a probability of $P = 0.9$.

$$P = \max\left(\frac{n_0}{L}, \frac{n_1}{L}\right) \tag{2.4}$$

2.2.5 Metrics to evaluate the RTN-based PUF

The evaluation of the quality of this PUF design was carried out using features that quantitatively describe its performance: the *uniqueness* (different PUF instances should return different responses for the same challenge), *unpredictability* (the response cannot be anticipated), and *reliability* (the same response should be obtained for the same challenge applied to the same PUF instance, that is, the PUF response must be robust or stable over time and varying environmental conditions). These features are quantified through a set of specific metrics [61]:

- The "intra-chip Hamming Distance", denoted as $HDintra$, is used to measure *reliability*. First, a n-bit response is obtained from an instance i, denoted by R_i. The same n-bit response is extracted at a different operating condition (temperature or supply voltage) or time instants with a value R'_i. Taking m samples of R'_i (being the t-th sample, $R'_{i,t}$) it is possible to obtain $HDintra$ as Equation 2.5 shows. HD represents the Hamming Distance (i.e., the number of positions of two arrays of length, l, at which their corresponding value are different) as shown Equation 2.6. The ideal value of $HDintra$ is 0% for a $reliability = 100\%$, ($reliability = 100\% - HDintra$).

$$HDintra = \frac{1}{m}\sum_{t=1}^{m}\frac{HD(R_i, R'_{i,t})}{n} \times 100\% \tag{2.5}$$

$$HD(x,y) = \sum_{j=1}^{l}|x_j - y_j| \tag{2.6}$$

- The concept of "inter-chip Hamming Distance", denoted as $HDinter$, is used to measure *uniqueness*. If there are two instances, i and j ($i \neq j$), each with n-bit responses, R_i and R_j respectively, for a given challenge C, then the average inter-chip HD among k instances is defined in Equation 2.7. Its ideal value is 50%.

$$HDinter = \frac{2}{k(k-1)}\sum_{i=1}^{k-1}\sum_{j=i+1}^{k}\frac{HD(R_i, R_j)}{n} \times 100\% \tag{2.7}$$

- The metric of "Hamming Weight", denoted as HW, is used to measure *unpredictability* as Equation 2.8 shows. The term $r_{i,l}$ represents the l-th binary bit of an n-bit response from instance i. Its ideal value is also 50%.

$$HW = \frac{1}{n}\sum_{l=1}^{n} r_{i,l} \times 100\% \tag{2.8}$$

2.2.6 Verifying the PUF performance

After proposing the conceptual architecture of the RTN-based PUF, the subsequent step involves conducting various evaluations. The RTN simulator introduced in [62] serves as a crucial tool in this evaluation, enabling the generation of RTN current traces under variations in the size of transistors or the biasing conditions for the PMOS configuration (shown in Figure 2.3a). For that, among the different designs proposed, the design of Figure 2.5b in which the transistor generate a drain current that experiments the RTN variation is the one which has been used.

For this evaluation, 500-s long, drain current traces of 1,000 80nm/60nm PMOS transistors (biased with $|V_{GS}| = 0.6V$ and $|V_{DS}| = 0.1V$) were obtained with the RTN simulator mentioned above. This simulator can generate current traces with a fixed time step, including both manufacturing variations and RTN fluctuations. Having a time step of $1ms$ would be the equivalent to considering a settling time for the MCF calculation and comparator circuits of $1ms$ when operating on real current traces in the fabricated PUF. Therefore, taking $t_{MCF} = 1s$ (complying with the rule of setting t_{MCF} three orders of magnitude larger than the settling time), on 500-s long current traces, yields 500 values of the MCF metric for each transistor. To compose a single RTN-based PUF instance, $M = 500$ transistors were randomly selected (the maximum length of a response R is then 250 bits). For a sound evaluation of the proposed PUF quality, 1,000 of these PUF instances were generated. The evaluation results in this Section uses the collection of challenges C_{eval} illustrated in Figure 2.7: the transistors to build a PUF instance are stored in a list; then, the challenge C_{eval} pairs transistors that are consecutive in that list (the 1st with the 2nd, the 3rd with the 4th, and so on).

The evaluation is carried out using MATLAB. The RTN-based PUF is simulated through basic math functions where each current trace is processed individually and proper computations are performed (e.g., the MCF is calculated using Equation 2.3). This simulation also takes into consideration potential non-idealities that might occur in a silicon implementation of either the MCF calculator or the comparator. For that, it was set a minimum value to compute the differences. In this scenario, the MCF calculator can only detect alterations in the current

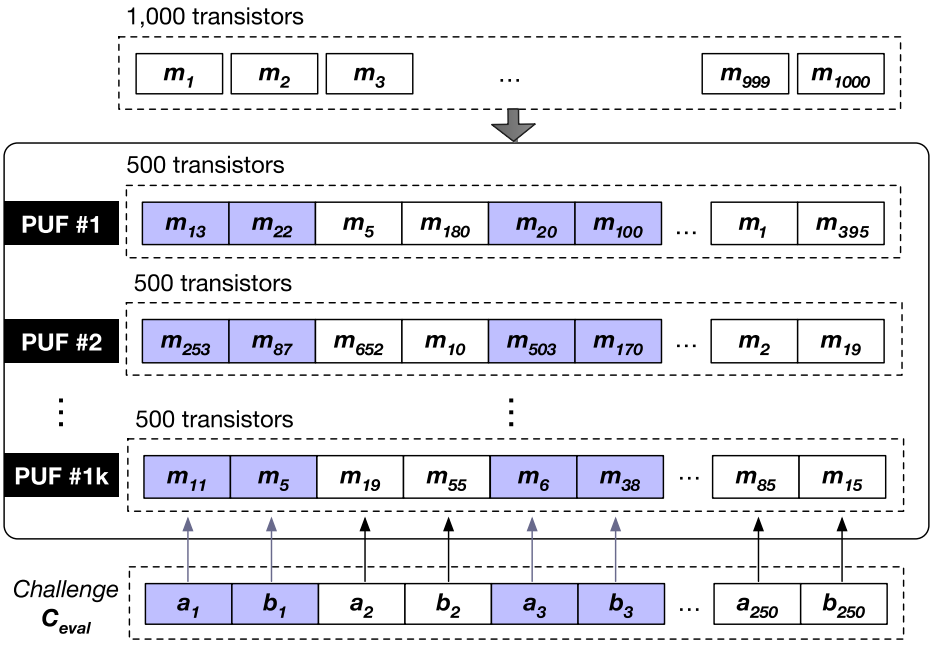


Figure 2.7 The challenge used for the evaluation of the PUF quality.

traces that exceed $20nA$, and the comparator can produce a response if the difference between MCF values surpasses $20nA$. If not, it yields a random value. These threshold values were established based on the resolution performance of some current comparators discussed in the literature [63, 64]. Figure 2.8 shows the result of the bit selection process on all generated 1,000 PUF instances. On every instance, 500 values of the MCF are computed (on a current trace of 500s with $t_{MCF} = 1s$) for each transistor in every pair. Then, the probability for every pair is computed using P in Equation 2.4 ($L = 500$). For every value p_i of the probability in the horizontal axis, the plot shows the statistics of the number of pairs with a probability equal or larger than p_i. As it can be seen, the better the probability, the fewer number of pairs. With the bit selection process done, *reliability*, *uniqueness* and *unpredictability* can be evaluated. For *reliability*, $HDintra$ has been computed for each PUF instance and every p_i value. The results are shown in Figure 2.9a in terms of *reliability* easing the comprehension of this concept for the readers, where the average number of pairs from Figure 2.8 is also plotted. Note that for $P = 1$, there are ≈ 20 pairs and the value of *reliability* is close to $\approx 100\%$ ($HDintra \approx 0$). If such high *reliability* degree is required for a 128-bits response, then an array of $\approx 2,300$ transistors would be sufficient. For *uniqueness* and *unpredictability*, $HDinter$ and HW have been computed and depicted in Figure 2.9b. $HDinter$ is always around the ideal value of 50%. HW presents more fluctuations, especially around high values of P, but nevertheless remain close to the ideal value on the whole range.

A first quality comparison is made with other reported differential PUFs using transistor arrays (like the one presented here) but taking variations in the threshold voltages caused by static manufacturing randomness as entropy source to generate a response [47, 48, 51] but ignoring RTN. Table 2.1 reports this comparison with averages of the quality metrics. For

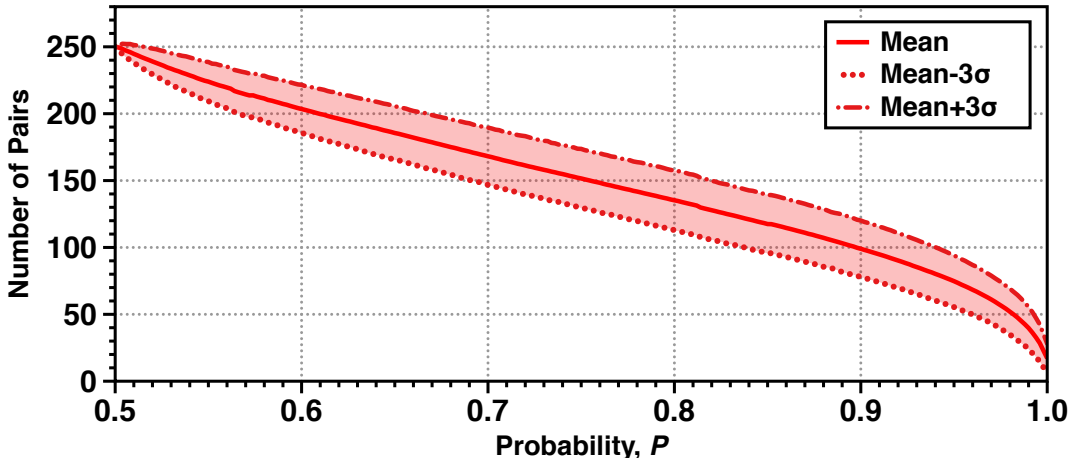
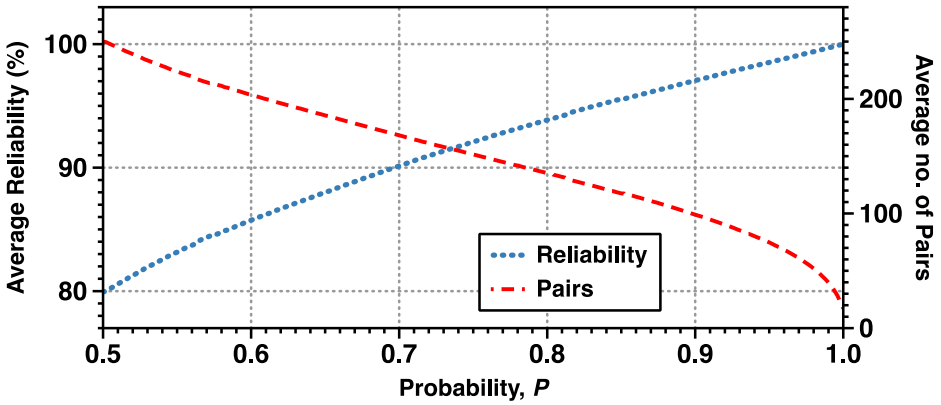
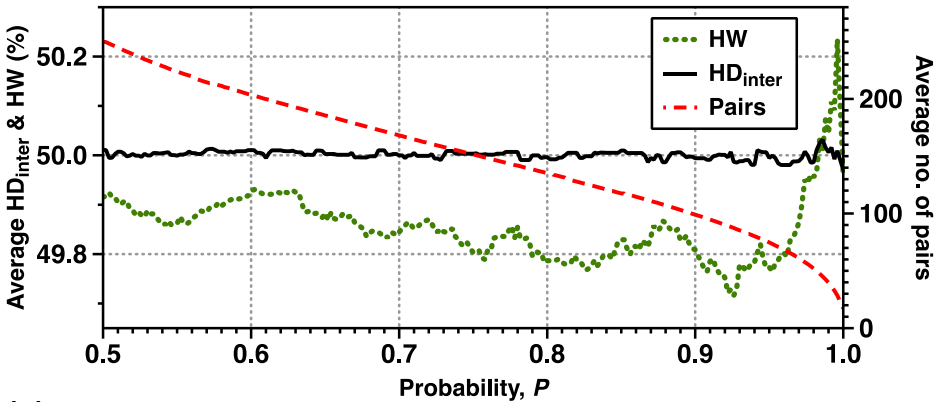


Figure 2.8 Resulting number of pairs from the bit selection carried out in the 1.000 RTN-based PUF instances.



(a) *Reliability* of the proposed RTN-based PUF.



(b) *HDinter* and *HW* of the proposed RTN-based PUF.

Figure 2.9 Results of the metrics of the proposed RTN-based PUF.

a 32-bit response, it is shown that using RTN outperforms other approaches where entropy comes solely from the mismatch-induced variability. Another comparison can be made with a differential PUF that does use RTN [56] in ROs and comparing with the area usage of the RTN-based PUF transistors array for similar levels of PUF quality. As shown in Table 2.2, the proposed RTN-based PUF can attain the same values of the averaged quality metrics (reliability, uniqueness, and unpredictability) but using fewer transistors. Finally, Table 2.2 also reports a comparison with a non-differential PUF using SRAM cells [65]. Again, equal quality levels are achieved with a lower number of transistors. To ensure a fair comparison, it would be essential to include the transistors that will be utilized in the implementation of the MCF calculator and the comparator.

As a conclusion, the evaluation results show that the RTN-based PUF outperforms other PUFs that use a pair of transistors and the comparison of voltage or current as the basic component. Moreover, it can attain the same quality level of a PUF that also uses RTN as

Table 2.1 PUF comparison with [47], [48] and [51]. ($n = 32$)

	[47]	[48]	[51]	***This dissertation***
Reliability (%)	91.58	-	94.2	99.8
HDinter (%)	50.23	49.9	49.9	49.99
HW (%)	-	53.94	53.94	50.09
Technology	130-nm	65-nm	130-nm	65-nm

Table 2.2 PUF comparison with [56] and [65]. ($n = 128$)

	[56]	[65]	***This dissertation***
Reliability (%)	99.99	99.99	99.99
HDinter (%)	48.0	-	49.97
HW (%)	-	50	49.96
Technology	65-nm	65-nm	65-nm
Number of transistors	4,352 (256 17T ROs)	4,992 (832 6T-SRAM cells)	2,286

entropy source. Thus, the aim of this section which was the verification of the PUF architecture has been corroborated with these results.

2.2.7 Studying the impact of the size and biasing condition of the entropy-generating transistors

The importance of this evaluation abides in that there are several factors that can influence the presence of RTN. First, the number of RTN defects is determined by the channel area. Second, both the occupancy and the relative shift amplitude $\Delta I_D/I_D$ of an RTN defect are strongly dependent on the carrier concentration [66] and, therefore, on the gate voltage, $|V_{GS}|$. Third, the charge uniformity from source to drain is significantly influenced by $|V_{DS}|$ and this can impact the RTN behavior [67], especially in nanometric scale devices with short channel effects. The aim of this evaluation is to find the importance of these factors in the performance of the PUF. For that, the strategy of evaluation followed in the previous section has been also applied here (i.e., the use of the RTN simulator, the strategy to generate transistor pairs shown in Figure 2.7 and the evaluation on1,000 PUF instances). The experiments have been carried out in two groups, one to study the effect of the channel area and one to analyze the impact of biasing. The values of the parametric variations (channel geometries, expressed by Width/Length in nm for each biasing combination) for this design of experiments are listed in Table 2.3.

Table 2.3 Summary of parametric variations

$\lvert V_{GS}\rvert$(V) \ $\lvert V_{DS}\rvert$(V)	**0.1**	**0.6**	**1.2**
0.6	80/60, 200/60, 300/80	80/60	80/60
0.8	80/60	80/60	80/60
1.2	80/60	80/60	80/60

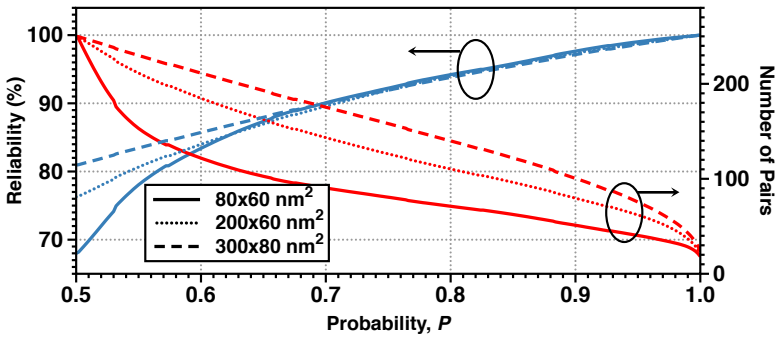


Figure 2.10 Impact of channel area on *reliability* and *number of pairs*.

For the transistors' size evaluation three channel areas were explored: $4,800nm^2$ (corresponding to the smallest transistor sizes, W/L = 80nm/60nm), $12,000nm^2$ (W/L = 200nm/60nm) and $24,000nm^2$ (W/L = 300nm/80nm). Figure 2.10 shows how reliability and the number of pairs change with channel area. The largest area devices seem to show better reliability for lower values of P. However, for the range of interest of reliability (close to 100%, $P > 0.9$), all geometries attain the same level of reliability. A particularly interesting insight into the

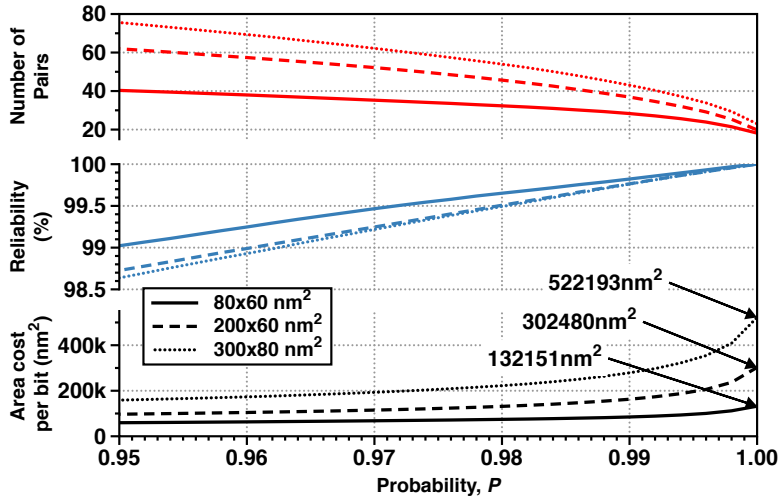


Figure 2.11 Area cost per stable bit for different channel areas.

influence of the channel area can be obtained if studying the area cost to implement a PUF with the better reliability available. As shown in Figure 2.11, this cost (at the bottom plot) refers to the silicon area needed to implement a perfectly stable bit ($P = 1$) to obtain a highly reliable PUF ($reliability = 100\%$). If 250 pairs yield 20 pairs with $P = 1$, thus providing perfect reliability, a response of length n would require an $n \times 250/20$ transistor array. The cost of such a stable bit is then $n \times W \times L \times 250/NP$ (with NP being the number of pairs at probability P). It can be concluded that the smallest area provides the lowest area cost while this cost increases almost fourfold for the largest channel. Finally, Figure 2.12 shows that HW and $HDinter$ are barely impacted by the channel area value, and, in every case, the values remain close to the ideal ones of 50%.

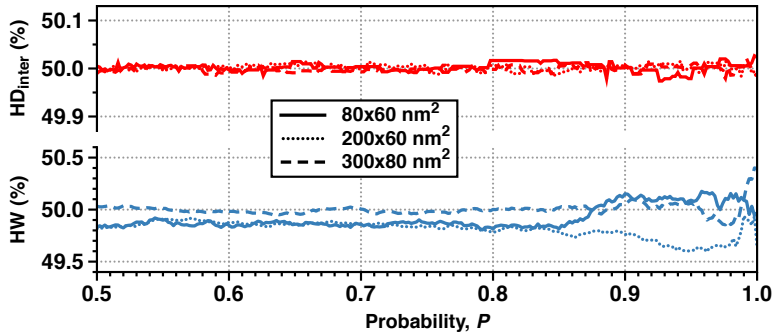


Figure 2.12 HW (*unpredictability*) and $HDinter$ (*uniqueness*) for different channel areas.

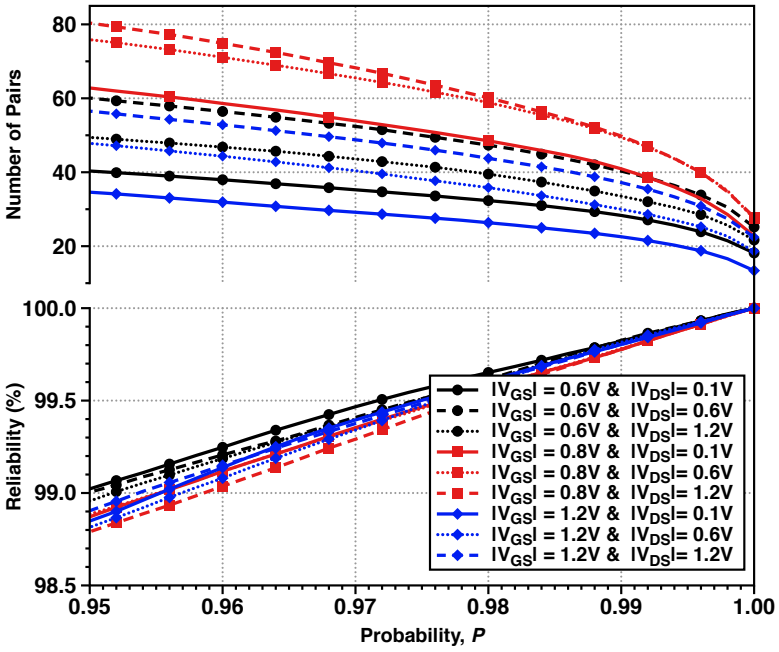


Figure 2.13 Impact of drain and gate voltages on *reliability* and *number of pairs*.

For the transistors' biasing conditions evaluation, it has been used the less-area-cost identified size (W/L = 80nm/60nm) with the help of the RTN simulator that can also simulate different biasing conditions. The impact that the biasing conditions (i.e., source-drain and source-gate voltages) have on reliability and the number of pairs is shown in Figure 2.13 (for $P > 0.95$ only since this is the range of interest in terms of *reliability*). A transistor in saturation yields better quality figures than in the linear regime and so it does if, additionally, the gate voltage is slightly over half the supply voltage of 1.2V. In terms of area cost per stable bit (shown in Figure 2.14), biasing with $|V_{GS}| = 0.8V$ and $|V_{DS}| = 1.2V$ provides twice the reduction in area than using $|V_{GS}| = 1.2V$ and $|V_{DS}| = 0.1V$, and a 1.5 reduction with respect to $|V_{GS}| = 0.6V$ and $|V_{DS}| = 0.1V$. One important note to make here is that biasing with higher voltages may, however, cause premature degradation through aging. To corroborate this, an aging simulation was done with an accurate stochastic simulator [68]. Indeed, biasing with $|V_{GS}| = 0.8V$, $|V_{DS}| = 1.2V$ may cause the transistor to age as far as 3.9 times faster than when biasing with $|V_{GS}| = 0.6V$ and $|V_{DS}| = 0.1V$. However, these results indicate variations in the

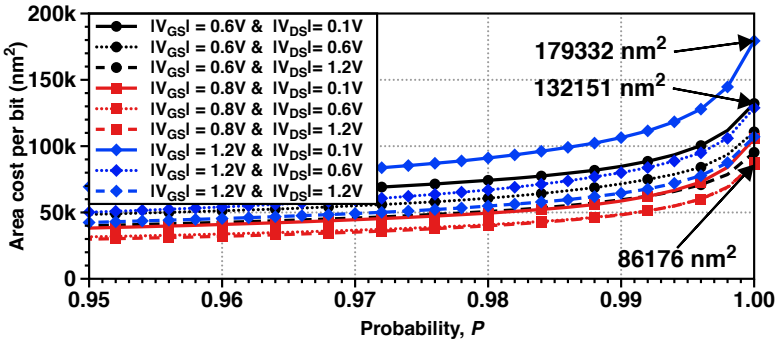


Figure 2.14 Area cost per stable bit for different biasing conditions.

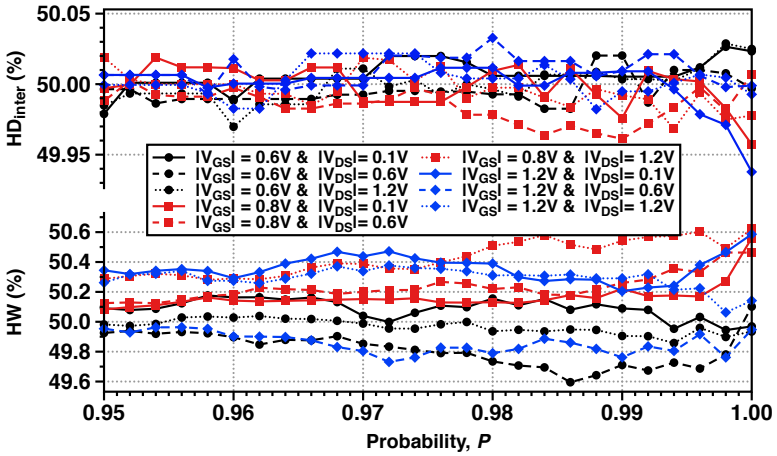


Figure 2.15 *HW* (*unpredictability*) and *HDinter* (*uniqueness*) for different biasing conditions.

V_{th} of the transistors, which may not necessarily imply a variation in the presence of RTN, thereby leaving the PUF unaffected. Nevertheless, a prudent measure would be to operate within the mid-supply range to mitigate the risk of premature aging. In terms of HW and $HDinter$, as in the case with the channel area, no significant design guidelines can be extracted from the results shown in Figure 2.15.

As a conclusion of this study, results show that there is a trade-off between reliability and silicon area cost per stable bit. This trade-off, however, vanishes with perfect reliability, with the smallest transistor (W/L = 80nm/60nm) being the cheapest option. In terms of biasing, the best characteristics are attained with $|V_{GS}| = 0.8V$ and $|V_{DS}| = 1.2V$, but with a critical downside: faster (up to 3.9X) degradation due to aging compared to when using lower bias voltages.

2.2.8 Evaluating the non-idealities in the building blocks of the RTN-based PUF

This evaluation has tried to shed light in the impact that non-idealities of some building blocks (described in Figure 2.5b) have in the PUF reliability. For that, this study has been divided in two phases. In the first phase, a parametric analysis using MATLAB is carried out to explore the ideal PUF performance (i.e., without non-idealities in its building blocks and selecting only the most reliable implementations of the PUF, i.e., using bits with $P \geq 0.99$), and also how non-idealities in the MCF calculators and the comparators impact the PUF performance. In a second phase, the PUF will be simulated with Spectre AMS to (1) check the design at the electrical level and (2) include non-idealities in the Analog MUX. The goal of this high-level design procedure is to refine the specifications for its building blocks so that the next phase of transistor-level design can be carried out. In doing so, a mixture of languages and simulation techniques are put in place thus demonstrating a methodology to design hardware security primitives.

To perform a high-level evaluation of the PUF and how a non-ideal implementation of the blocks may impact its quality metrics, several non-idealities have been introduced in the form of parametric variables. For the MCF calculator, a current threshold (MCF_{th}) is introduced below which no change in the drain current can be detected following the same MATLAB simulations described in the previous PUF analysis. This represents a resolution limitation in the circuits that eventually implement this building block. For the Comparator and the Analog Multiplexer (whose schematic is shown in Figure 2.16 for the sake of clarity), the non-idealities considered have been the *offset* and the ratio of the on (R_{on}) and off (R_{off}) resistances of the analog switches, respectively. For the sake of illustration, Figure 2.17 shows how a

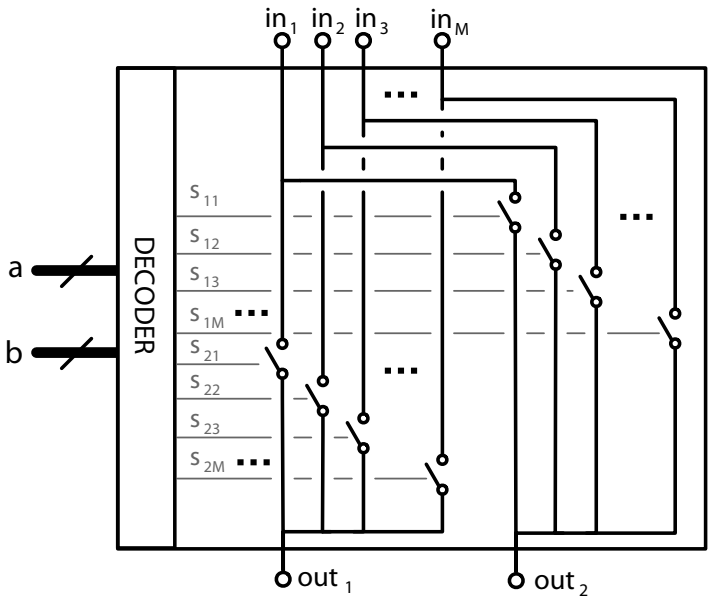


Figure 2.16 Schematic of the Analog MUX.

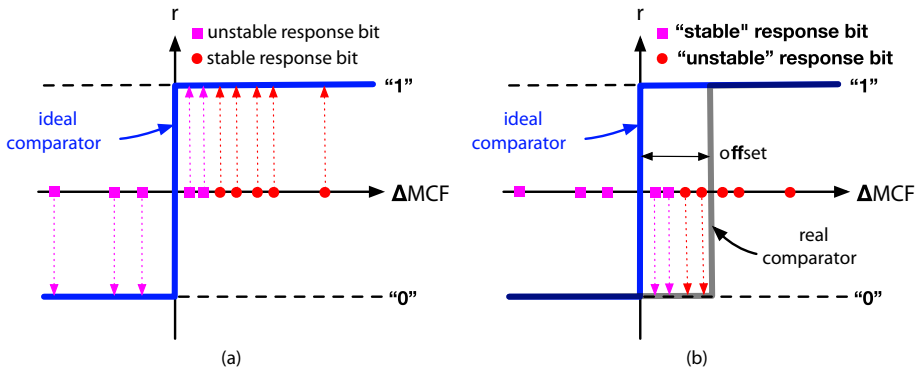


Figure 2.17 Impact of the comparator offset in the bit stability.

non-zero comparator *offset* would impact the PUF quality through the bit stability: for an ideal comparator (Figure 2.17(a)), a fully stable response bit (red circles) could become unstable when an *offset* is present (Figure 2.17(b)); similarly, an unstable bit in an ideal situation could become stable for a sufficiently large *offset*. Logically, this alteration will have consequences not only for the bit stability, but also, as it will be shown later, for the rest of the quality metrics. Therefore, it is essential that maximum allowable values are found for the non-idealities so that minimum quality metrics are attained later at the transistor level.

An initial coarse parametric sweep was carried as follows: from $0nA$ to $100nA$ in $20nA$ increments for MCF_{th}, and in the range of $\pm 100nA$ in $20nA$ increments, for the comparator *offset*. For each PUF instance and each parametric combination, the numerical simulation

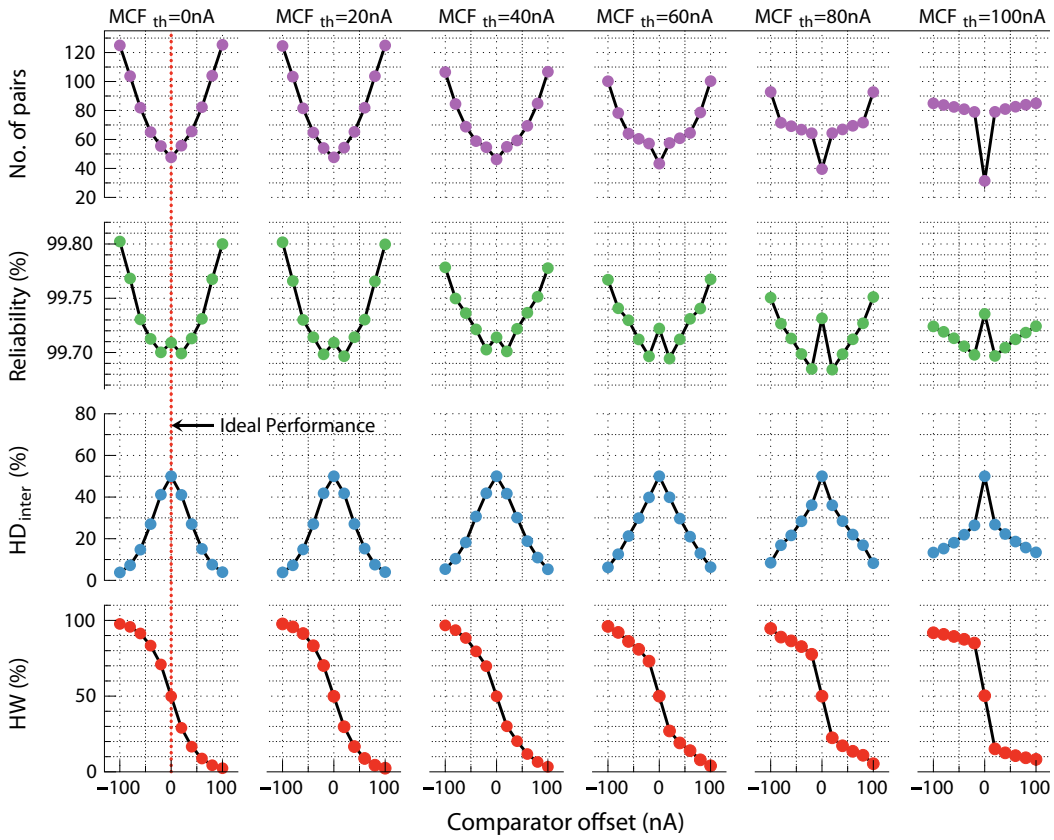


Figure 2.18 Initial coarse parametric analysis of the impact of *offset* and MCF_{th}.

results are shown in Figure 2.18. This figure represents the values of the PUF quality metrics averaged over the 1,000 PUF implementations. Several conclusions can be extracted at this point: (1) the *offset* has a large impact on all metrics while MCF_{th} plays a significant role only when it is larger than $60nA$; (2) the number of stable pairs improves when the *offset* increases (meaning that less silicon area is required to attain a reliable PUF response); (3) the PUF reliability initially decreases with increasing *offset* to later improve slightly but, overall, remains at optimal values because the selected pairs are quite stable; (4) the former apparent improvements are counteracted by dramatical degradations of the HW and $HDinter$ metrics. In fact, a slight change in *offset* produces a large deviation of these two metrics from their ideal value (50%) and this deviation worsens with increasing MCF_{th}. Figure 2.19 offers a deeper look at the HW and $HDinter$ degradations when *offset* varies between $\pm 5nA$ with $1nA$ steps: only when $MCF_{th} \leq 20nA$ and *offset* is in the $\pm 4nA$ range, solutions can be found with HW between the more acceptable values of 45% and 55%, and with $HDinter$ over 49.5%. In summary, for this RTN-based PUF to perform acceptably well ($reliability \geq 99\%$, number of stable pairs

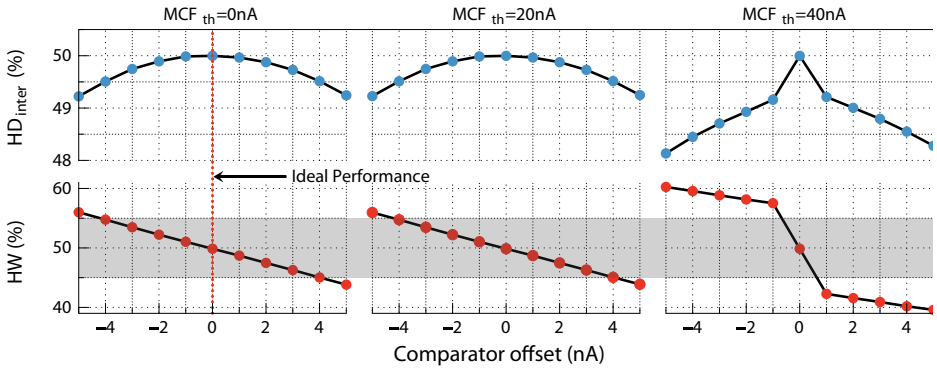


Figure 2.19 Detailed parametric analysis of the impact of *offset* and MCF_{th}.

≥ 40, $45\% \leq HW \leq 55\%$, $HDinter \geq 49\%$), the comparator *offset* should be no larger than $\pm 4nA$ and the MCF calculators should detect RTN events with as low as $20nA$ of amplitude.

Next, to evaluate the impact of imperfect switches in the Analog MUX, one PUF implementation from the pool of 1,000 ones was selected. This evaluation required of an electrical simulator since imperfect switching ultimately distorts the flow of current towards the MCF calculators. Such a distortion was thus simulated using Spectre AMS and a variety of high-level description languages. To use Spectre AMS in combination with the RTN simulator, different high-level description languages have been used: (1) the RTN shifts are added to the transistors by including a voltage source at their gates to emulate the shift in V_{th}; (2) the Analog MUX is a combination of a digital decoder and a collection of analog switches, so a Verilog-AMS module has been used; (3) the MCF calculator is a purely analog implementation, so Verilog-A was used there; (4) the comparator module, is a Verilog-AMS block since it receives input signals and outputs the digital PUF response. A control unit was also implemented in Verilog to control the selection of transistors and the execution of the MCF comparison. To find out the impact of R_{on} and R_{off} of the analog switches in the MUX, the implementation with quality metrics (obtained in the first phase of high-level design) in Table 2.4 was simulated using Spectre AMS. These metrics correspond to the worst-case scenario of the non-idealities (i.e., the offset and MCF_{th} being $4nA$ and $20nA$, respectively), so the potential effect of the imperfect analog switches can be better assessed.

Table 2.4 PUF quality metrics of the selected implementation

No. Pairs	***Reliability (%)***	***HW (%)***	***HDinter (%)***
42	99.81	52.38	49.28

Table 2.5 Parametric Sweep of R_{on} and R_{off}

Resistance	*Start value*	*Final value*	*Steps per decade*
R_{on}	100Ω	$100M\Omega$	1
R_{off}	$1k\Omega$	$100M\Omega$	1

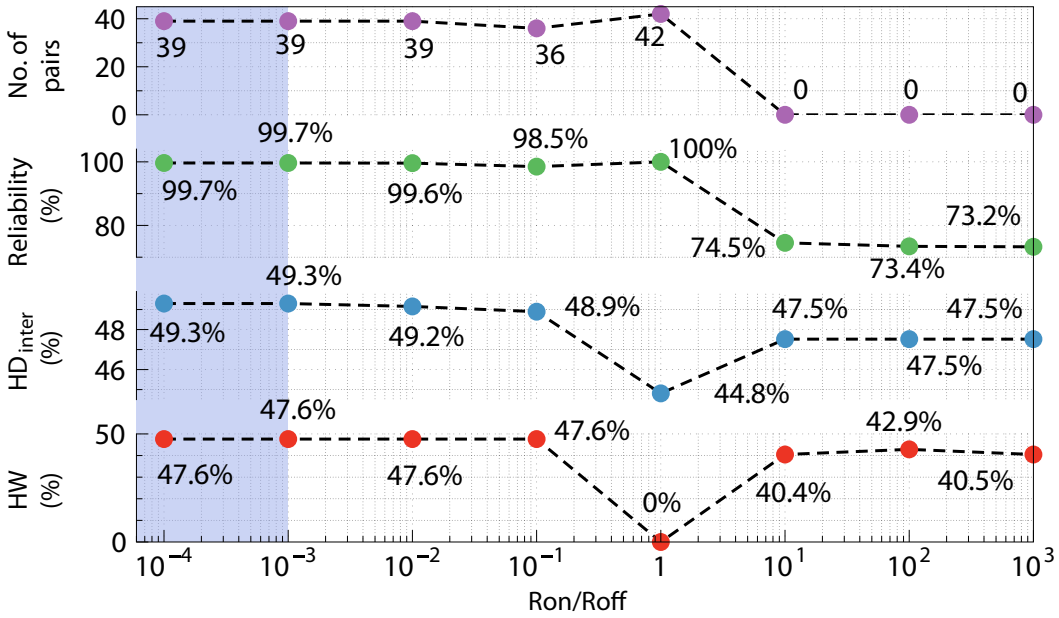


Figure 2.20 Detailed parametric analysis of the impact of imperfect switching.

A parametric analysis was carried out following the sweeps in Table 2.5. For each combination of resistances, the number of stable pairs, reliability, *HW* and *HDinter* were evaluated. The results are depicted in Figure 2.20. When $R_{on} > R_{off}$, (i.e., $R_{on}/R_{off} > 1$) both the reliability and the number of stable pairs are degraded. What happens is that the MCF calculators receive not only the current from the incorrect transistor in the pair, but also current of all other transistors in the array that should not be part of the challenge. Then, no stable pair can be found as the input signals to the MCF calculators can be assimilated to a random noise where all RTN information is lost (and, being it random, *HW* is thus close to 50%). In reality, due to the inherent design of a CMOS transmission gate, this would not be physically plausible. However, it has been considered throughout the entire range of values.

For $R_{on} = R_{off}$, the currents of all transistors spill equally into both MCF calculators, so the two inputs of the comparator are two identical signals. Therefore, with the comparator offset being $4nA$, all responses are "0" in all cases. That is why $HW = 0\%$ (so the response of the resulting PUF is totally predictable), but *reliability* is 100% and all considered pairs are

found to be stable. It is only when $R_{on}/R_{off} < 1e-3$ that finite resistances are not negatively impacting the PUF quality. Regarding *uniqueness*, there is very low correlation between imperfect switching and $HDinter$ mainly because, as shown in Figure 2.19, the other PUF implementations are already very unpredictable. Finally, note that the differences with the ideal case of $R_{on} \to 0$ and $R_{off} \to \infty$ (see Table 2.4), may be explained by the logical differences between a numerical simulation, which idealizes the system performance, and an electrical simulation, which is closer to reality, especially in dynamically complex circuits.

As a conclusion, both numerical and electrical simulations have been combined to expose the influence that non-idealities in the PUF building blocks have on the PUF quality metrics. The outcome of this high-level are design specifications for the building blocks (MCF calculator, comparator and Analog MUX). The results show that the PUF quality comes not only from the entropy source but also from the quality of its building blocks.

2.2.9 Summary of the RTN-based PUF realization results

In this section the RTN-based PUF conception has been detailed. For that, the method to generate a response bit comprises four different architectures: comparing the RTN information provided by either one or two transistors, and either in terms of voltage or drain current. For the sake of evaluation, regarding the possibility of generating drain current traces containing RTN, only one of the architectures has been used for the PUF evaluation. The evaluations have provided promising results. First, it is demonstrated that the RTN-based PUF can generate a highly reliable response (regarding the PUF metrics). Second, the best size to provide the lowest area-cost is W/L = 80nm/60nm for a 65-nm CMOS technology. Besides, the best biasing condition for that is $|V_{GS}| = 0.8V$ and $|V_{DS}| = 1.2V$. And, third, it has been also demonstrated that the building blocks necessary to conform the PUF may also harm its quality. This supports the importance of well-designed building blocks to improve the PUF performance. With all this information, the next step taken was the integration into a physical design.

2.3 RTN-based PUF low-level design

Once the conception and verification of the RTN-based PUF was proved the next step was the low-level design for the implementation in an ASIC. Evaluating the current version of the PUF has confirmed the feasibility of the RTN-based PUF concept and shed light on how the necessary building blocks for obtaining the response bit can impact PUF performance. In this case, for the low-level design, the PUF architecture chosen is the one depicted in Figure 2.5d, where the RTN is shown in the transistor's drain voltage. Employing the voltage architecture

has yielded several benefits. Firstly, it avoids potential issues that may arise in current sensing, simplifying the design by utilizing voltages. Secondly, the examination of the bias current value (ICC) selected in the Constant-Current method [60] enable parameter scaling that may improve the quality of the PUF. Thirdly, in case aging impacted the PUF, adjusting voltage levels becomes a simpler task. For that, the analog design flow with UMC 65nm process technology has been applied.

2.3.1 Floorplan of the RTN-based PUF integration scheme

As it was above mentioned the architecture selected to be implemented is illustrated in Figure 2.5d. Instead of using the drain current, the threshold voltage, impacted by RTN, is obtained using the Constant-Current (CC) method [60]. Setting the PMOS transistor in diode configuration and driving it with a fixed current, the voltage at its drain (V_D) can provide a measure of the threshold voltage. A relationship between transistor size and the optimum value of the current *ICC* is also defined. Since the transistors used in this implementation have the size ratio $W/L = 80nm/60nm$, following the indication presented in [60], the optimum *ICC* for a current-mode implementation should be $100nA$.

An important aspect here is that different *ICC* values provide different amplified versions of the threshold voltage at the transistor drain. This has an additional advantage over using the MCF: the RTN-induced shifts on V_D are larger and, therefore, the sensing capabilities of the MVF calculation blocks and the comparator resolution can be relaxed. To demonstrate this, a modification of the already-used RTN simulator presented in [62] was used. This new version provided drain voltage traces that contain RTN. From a pool of 1,000 RTN simulated traces, Table 2.6 shows the cumulative maximum and the minimum voltages detected ($CMAXV$ and $CMINV$) and the maximum and minimum MVF computed in $100\mu s$ (MVF_{MAX} and MVF_{MIN}). The tradeoff here, as far as circuit design is concerned, can be appreciated by looking at the maximum and minimum common-mode voltage of the voltage signal acquired (CM_{MAX} and CM_{MIN}) at node V_D. Lower *ICC* values lead to wider MVF ranges, resulting in a less demanding comparator resolution. However, these wider MVF ranges also require wider, and, therefore more difficult to design, input common-mode ranges for the analog circuits in the MVF calculation module. On the other hand, detecting the RTN may become more challenging when the *ICC* values are very low as the transistor enters into the sub-threshold region.

The concept shown in Figure 2.5d has been translated into a physical realization with the following approach. On the one hand, both the transistors and the selector are organized into a single array of M unit cells, each one containing the transistor itself, a set of switches, and a digital control block. On the other hand, there is an Analog Sensing Block (ASB), comprising

Table 2.6 Comparison between different ICCs (voltages in mV)

ICC	CMAXV	CMINV	MVF_{MAX}	MVF_{MIN}	CM_{MAX}	CM_{MIN}
100nA	996.7	677.1	175.0	14.8	979.5	738.6
200nA	931.2	645.4	104.8	8.5	923.6	690.0
500nA	858.9	607.3	53.0	4.1	851.5	629.0
700nA	832.9	593.3	44.4	3.3	828.9	602.8
1μA	804.9	569.3	32.8	2.6	803.0	573.4
2μA	742.8	502.7	26.7	1.7	741.6	505.3
3μA	700.9	457.7	17.4	1.3	700.2	459.2
5μA	638.9	387.7	12.7	1.0	637.1	388.7
6μA	613.6	358.6	1.2	0.9	611.3	359.4

the MVF calculator and the comparator that processes the drain voltages and outputs a response to a given challenge. This organization is depicted in Figure 2.21.

The ASB contains two MVF calculation modules. Each module has two Peak Detection and Hold (PDH) blocks, to obtain *CMAXV* and *CMINV*, a subtractor block (SUB), to provide the MVF value for the selected transistor, and a comparator, to ultimately attain the response. As for the unit cell, detailed in the inset of Figure 2.21, the transistor is connected, when set as a member of the pair defined by the challenge, to a current source (*ICC*) to evaluate its threshold

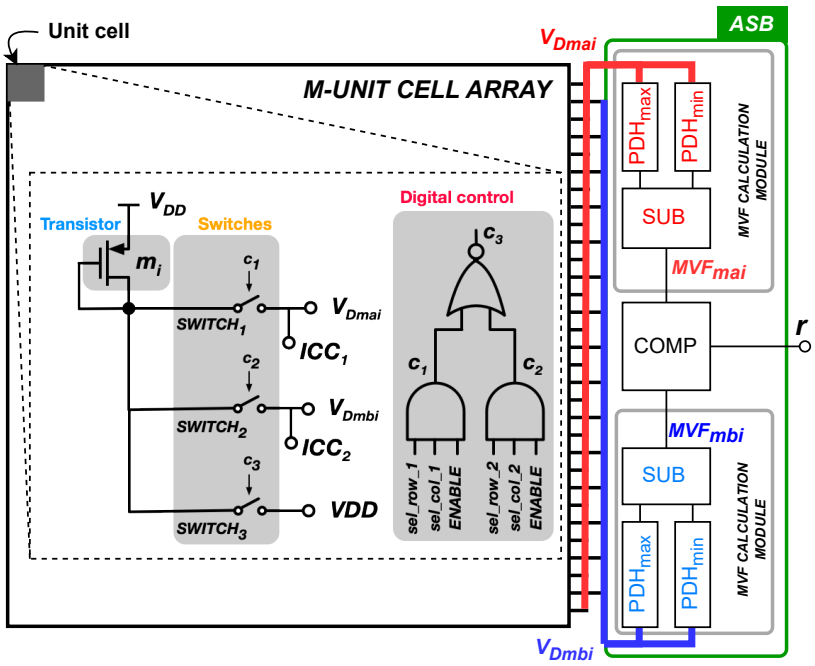


Figure 2.21 Floorplan of the RTN-PUF showing the unit cell and the ASB.

voltage using the CC method. The switches enable the transistor to be set either as the first (with `SWITCH 1`) or the second (with `SWITCH 2`) member of the pair in the challenge, or (with `SWITCH 3`) in a standby mode when the transistor is not part of the challenge and to prevent excessive power dissipation. As shown with the digital control block schematic in Figure 2.21, these switches are activated depending upon the location (row and column) of the transistor:

- `SWITCH 1` is activated when the row (`sel_row_1`) and the column (`sel_col_1`) signals of the first member of the pair selection are activated.
- `SWITCH 2` is activated similarly for the second member of the pair (with `sel_row_2` and `sel_col_2`).
- `SWITCH 3` is activated if the transistor is not selected or the ENABLE signal is not activated, remaining in the standby mode.

An array of 4,096 unit cells has been considered for the RTN-based PUF. This would allow generating 2,048 pairs and, therefore, a 2,048-bit response. With all these considerations the next step was the design of this array.

2.3.2 Transistors Array design

The design of the switches deserves, as demonstrated in Section 2.2.8, careful consideration. The strategy here was to consider two elements: (1) a limit for the detection of RTN changes (i.e., the fastest RTN event that can be completely detected); (2) the parasitic resistances and capacitances from the layout of the switches as well as from the routing lines used throughout the entire array. Both elements bring along a design constraint. On the one hand, the lower the limit, the more complex the required detection circuit becomes. On the other hand, the larger the parasitics, the more severe the lowpass filtering out of RTN information, inevitably arising from having RC elements in any integrated circuit as shown in Equation 2.9 (where the subscript T denotes the total resistances and capacitances affecting the transistor that is generating the signal with RTN).

$$f = \frac{1}{2\pi R_T C_T} \tag{2.9}$$

The initial estimation of such parasitics reveals that the total capacitance (C_T) connected to the transistor can be approximated as expressed in Equation 2.10. This calculation is a sum of the capacitance of the routing lines (C_{LINE}), the parasitic capacitances of the transmission gates in the array ($C_{GS-SWITCH}$), the parasitic capacitances of the input transistors of the required OTAs in the MVF calculator (C_{GS-OTA}), and the load capacitance (C_L). After designing the

array structure, C_{LINE} was determined to be $1pF$ as a maximum value. The load capacitance, C_L, which represents the connection with the pad, can be estimated as about $1pF$. The parasitic capacitances due to the transistors in the switches and in the OTAs inputs were calculated with the transistor size as an initial approximation at $W/L = 3\mu m/0.5\mu m$, with each $C_{GS} = 100fF$. According to Equation 2.10, C_T can be bounded by $2.6pF$. However, to account for any variations due to fabrication or any unconsidered factors, the total capacitance connected to the transistor is approximately $C_T = 5pF$.

$$C_T = C_{LINE} + 2 \cdot C_{GS-SWITCH} + 4 \cdot C_{GS-OTA} + C_L \tag{2.10}$$

On the other hand, since the sensing block responsible for computing the MPF is now specifically designed, it is possible to reduce the detection time of the voltage peak caused by the RTN to $100ns$. This sets the maximum R_T, derived from Equation 2.9, at $3,183\Omega$. This total resistance (R_T), as Equation 2.11 shows, is composed of the routing line resistance (around 100Ω) and the on-resistance R_{on} of the switch, the latter depending on the size and type of CMOS transistors used. From Equation 2.11, it is possible to get the maximum R_{on} allowed, which is estimated at $3k\Omega$.

$$R_T = R_{LINE} + R_{on} \tag{2.11}$$

Once the specification has been obtained, the next step is to design the switches. To that end, CMOS transmission gates [69], with low threshold voltage transistors, have been used as shown Figure 2.22. The first simulation was a static one in which the switch is evaluated in open and

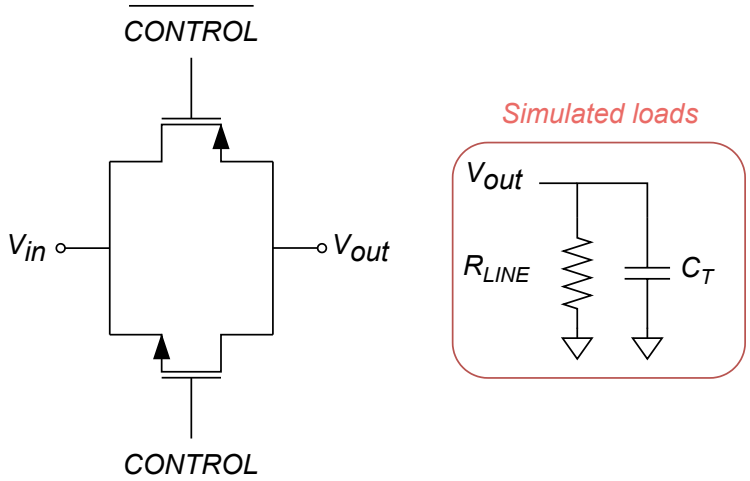


Figure 2.22 CMOS transmission gates implemented in the transistors' array. In the red box the load resistance and capacitor used for the simulations.

close state varying the input voltage (V_{in}) from 0 to 1.2V. The length (L) of the transistors that comprise the transmission gates was set in $100nm$ to avoid possible current leakage that could affect the rest of the circuitry. Then, several widths were evaluated: $W = 1\mu m$, $W = 2\mu m$ and $W = 4\mu m$. To account for manufacturing variations, 1,000 instances of transmission gates were generated using Monte-Carlo simulation. Table 2.7 displays the worst and best-case resistance values for R_{on} and R_{off} of the set of instances evaluated for each size. While the transmission gate with $W = 4\mu m$ and $L = 100nm$, delivers maximum performance, a balanced trade-off between size and performance is selected: $W = 2\mu m$ and $L = 100nm$. The worst R_{on} occurs around $V_{in} = 0.7V$, where neither NMOS nor PMOS are operating optimally. For R_{off}, the lowest value corresponds to the input voltage being equal to 0V and the highest one when it is equal to 1.2V. These results are in line with the established thresholds for R_{on} and R_{off} ratios that were obtained in Section 2.2.8.

The array design that conforms the PUF is rooted in the unit cell design. For the array of 4,096 transistors considered, a square-shaped floorplan of 64 rows x 64 columns has been implemented. A decoder has been used for the row and column of each of the two unit cells that make up the challenge (i.e., 4 decoders in total, with 64 output bits). The input of the decoders (i.e., the specific number of the transistor selected out of the 4,096 in the array) is a 12-bit digital word. The 6 least significant bits are used for the row selection, while the 6 most significant bits are used for the column selection. To reduce the number of chip pads devoted to this 12-bit word for unit cell selection, serializers, following the serial-in parallel-out architecture, have been implemented. For that, two levels of registers have been used, working as follows: once the 12-bit digital word for the selected transistor has been introduced in the first-level registers, a load signal activates the data loading in the second-level registers. The use of two serializers (one per transistor of the pair making up the challenge) is thus required.

Table 2.7 Worst and best case of each transistor size evaluated to comprise the transmission gates

Size	$\mathbf{R_{on}}(\Omega)(V_{in} = 0.7V)$		$\mathbf{R_{off}}(\Omega)$	
	Worst Case	***Best Case***	***Worst Case***	***Best Case***
$W = 1\mu m$ $L = 100nm$	4.46k	2.90k	278k (0 V) 342k (1.2 V)	1.44M (0 V) 1.50M (1.2 V)
$W = 2\mu m$ $L = 100nm$	2.18k	1.50k	253k (0 V) 310k (1.2 V)	1.41M (0 V) 1.47M (1.2 V)
$W = 4\mu m$ $L = 100nm$	1.09k	766	229k (0 V) 283k (1.2 V)	1.39M (0 V) 1.47M (1.2 V)

Figure B.1 in Appendix B.1 shows the layout implementation of the PUF array with 4,096 unit cells. An inset has been included to show the layout of the unit cell as well. The total size of this array is $730\mu m \times 700\mu m$. The array has been designed in a modular and regular fashion so that it can be easily expanded to generate a larger number of stable response bits.

Once a new given transistor is selected (as 1^{st} or 2^{nd} member of the pair in a new given challenge), its drain voltage generally differs to the drain voltage of the previous transistor. Due to the on- and off-resistance of the unit cell switches and the routing parasitics, the voltage that the ASB is evaluating suffer a dynamic behavior as shown Figure 2.23. Phase A represents the measurement of the previous transistor with a drain voltage, $VD\#1$, that in a time $t = t_s$ is deselected, selecting a new one in Phase B with a drain voltage, $VD\#2$. It is therefore important to determine how long will it take to settle since, if the ASB processes these prematurely, the extracted MVF values may be incorrect and the response may end up being the wrong one. The evaluation of this settling time in the selection process has a twofold purpose: (1) to confirm that the layout has been properly done and (2) to know how fast a response can be reliably obtained from the RTN-based PUF once a challenge is given.

To measure how well the signal settles, the evaluation tracks its variation rate and extracts at which three threshold times (indicated as th_1, th_2, and th_3 in Figure 2.23) these rates gradually decrease below $10mV/\mu s$, $1mV/\mu s$ and $0.1mV/\mu s$. The signal is considered settled (and thus no impact on the PUF response occurs because of this dynamic behavior) when it reaches threshold time th_3 (i.e., a variation rate of $0.1mV/\mu s$ or less). To simplify this evaluation and reduce the computational requirements, only 3 unit cells have been considered in the selection process.

In addition, temperature variations have been included to cover the Military Extended range testing for 3 temperatures: -55^oC, 27^oC and 125^oC. At each temperature, a Monte-Carlo

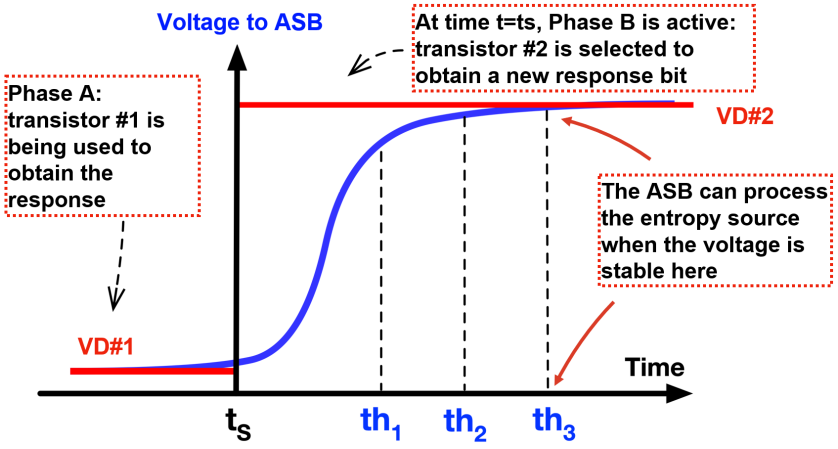


Figure 2.23 Dynamic behavior and settling during the selection process.

simulation with 200 samples has been run. The test bench used is defined in Table 2.8. The 3 unit cells selected are: (1) unit cell #0 (at the left topmost corner of the array), labeled "ORIGINAL"; (2) the one just below, labeled "NEAR"; and (3) the farthest one from unit cell #0, labeled "FAR". The test bench considers three consecutive phases (A, B and C) to analyze if and how the deselection → selection process depends on each unit cell being the 1^{st} or 2^{nd} member of the pair and thus quantify any difference whatsoever between nodes *VDmai* and *VDmbi*. For *VDmai*, the 1^{st} member of the pair is "ORIGINAL" (phase A), then "NEAR" (phase B) and then "FAR" (phase C). Similarly, for *VDmbi*, the 2^{nd} member of the pair is "FAR" (phase A), then "ORIGINAL" (phase B) and then "NEAR" (phase C).

Table 2.8 Array test bench definition

CELL	***NUM.***	***ROW***	***COL***	***Phase A***	***Phase B***	***Phase C***
ORIGINAL	0	0	0	1st	2nd	Standby
NEAR	1	1	0	Standby	1st	2nd
FAR	4,095	63	63	2nd	Standby	1st

As mentioned above, due to the unit cell switches and the routing parasitics, the evaluation carried out obtains the worst case of threshold time th_3 (in Figure 2.23) at which the output of the array (the voltages *VDmai* and *VDmbi*) is stable and settled and, therefore, no dynamic-induced error hinders the MVF detection and PUF response. These worst cases $(\mu + 3\sigma)$ are shown in Table 2.9 for all 3 threshold times, all temperatures and each unit cell being the 1^{st} and 2^{nd} member of the transistor pair in the challenge (following the test bench of Table 2.8). The results in Table 2.9 show that: 1) both voltages at *VDmai* and *VDmbi* settle considerably fast as there are small differences between the threshold times for the same unit cell; 2) the threshold times between cells are very similar meaning that the location in the array (and thus routing parasitics from the implemented layout) does not have any impact; 3) the selection as 1^{st} and 2^{nd} member of the pair yields similar threshold times; 4) temperature has an almost negligible impact. Additionally, and in the face of the worst-case value of threshold time th_3 (obtained for cell "NEAR" at 125^oC when selected as 1^{st} member of the pair as highlighted in Table 2.9) a guideline for use is attained: to establish a wait time of $10\mu s$ to have a properly stabilized input for the ASB to process. This wait time is to be added to the time for the ASB to carry out the MVF extraction (t_{MVF}) and comparison.

Concluding, the strategy has been, instead of using the drain current to encapsulate the RTN, to apply a more practical approach based on monitoring voltages. The strategy followed for the RTN-based PUF core building block consists of an array of unit cells that contain the transistor, from which entropy is extracted, and auxiliary elements (switches and a digital control circuit). This unit cell and the array layout have had to be carefully designed for such entropy to be used

Table 2.9 Evaluation of the 4,096 unit cell array (threshold times in μs)

T (^{o}C)	*CELL*	$\mathbf{th_1}(\mathbf{10mV}/\mu\mathbf{s})$ 1st	2nd	$\mathbf{th_2}(\mathbf{1mV}/\mu\mathbf{s})$ 1st	2nd	$\mathbf{th_3}(\mathbf{0.1mV}/\mu\mathbf{s})$ 1st	2nd
-55	ORIG.	2.61	2.72	4.76	3.49	6.03	4.46
	NEAR	2.57	1.90	5.96	3.52	5.65	4.97
	FAR	2.00	0.82	3.54	7.59	4.86	6.64
27	ORIG.	3.07	3.28	4.40	4.66	6.20	6.06
	NEAR	0.97	2.13	6.93	4.18	7.06	6.13
	FAR	2.32	0.83	4.21	7.44	6.18	6.12
125	ORIG.	3.61	3.84	4.79	5.47	6.10	6.53
	NEAR	0.96	2.34	7.20	5.04	**8.01**	7.60
	FAR	2.33	0.82	4.84	7.51	7.71	6.88

without introducing either ohmic losses or timing limitations that may degrade the response quality and turnaround times. However, the ASIC design does not finish here, the next step is to evaluate and implement the building blocks required to extract the response bit from the RTN.

2.3.3 Analog Sensing block implementation

The architecture for the ASB in the PUF responsible for measuring and processing all the RTN information is depicted in Figure 2.24. The voltage signal generated in each transistor serves as the input to a pair of Peak Detect and Hold circuits (PDH_{MAX} and PDH_{MIN}) to calculate the maxima ($CMAXV$) and the minima ($CMINV$) envelopes, respectively. The subsequent block is a difference amplifier that internally subtracts the two values obtained, giving the MVF value for the selected transistors (M1 and M2 in this case). This subtraction (ΔMVF) is added to a reference voltage V_{REF} to set the common-mode operation of the comparator.

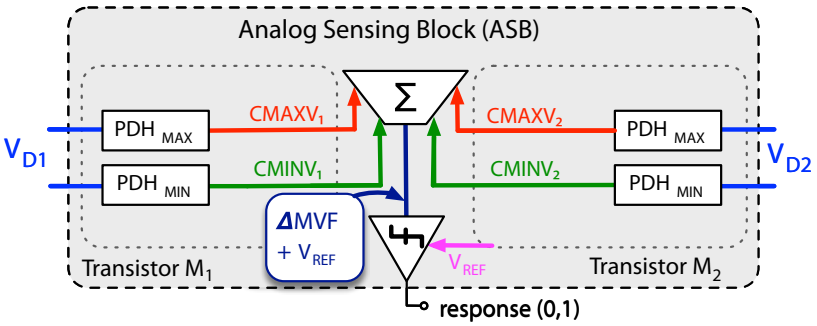


Figure 2.24 Schematic of the proposed ASB

The PUF response is directly obtained by comparing the addition result ($\Delta MVF + V_{REF}$) with the reference voltage V_{REF} to finally give the binary response (0,1) depending if it is larger or smaller than V_{REF}. Thus, this step is divided in the design of the PDH, the subtractor and the comparator.

2.3.3.1 PDH design

The design and implementation of the PDH circuits, shown in Figure 2.25, have been carried out inspired by the ideas in [70][71] and adapting those ideas for RTN detection. The PDH circuit relies on an amplifying element (an OTA) connected to a load made by the series combination of a hold capacitor C_H and a resistor R_C (whose purpose will be made apparent later) through a current mirror.

One important consideration as far as the usage of the PDH circuit within the RTN-based PUF is that the random offset from the OTA is one major non-ideality that may seriously impact the values of the detected peaks and, thus, the MVF values that are ultimately attained. This is because such offset can be of the same order of magnitude as the RTN-induced shifts present in the transistors, so it is mandatory to overcome this issue. To do so, an offset cancellation approach is used, as illustrated also in Figure 2.25.

This approach uses three main phases of operation. During the Write phase (switches W on), represented in the red circuitry, the maximum value of V_D plus the input-referred offset (V_L) of the OTA are stored on node L using capacitor C_H. The PDH circuit operates in this

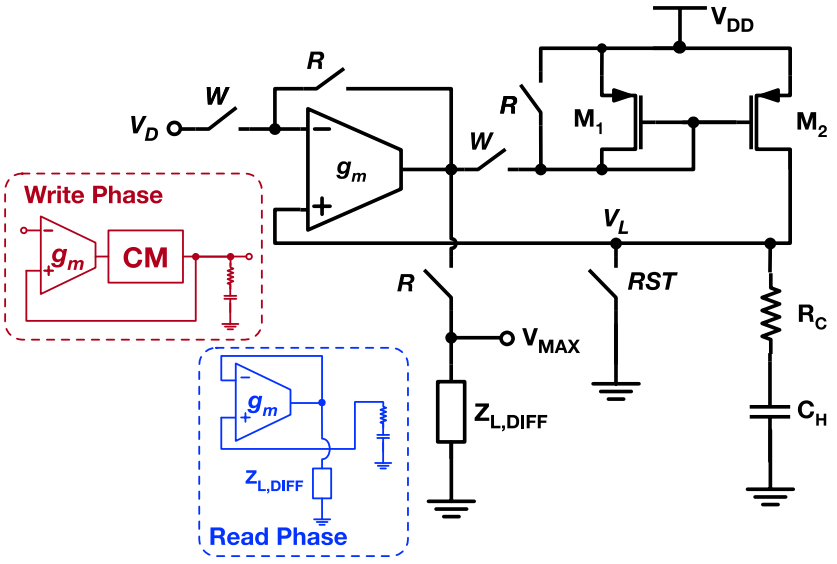


Figure 2.25 Final design of the PDH circuit. The circuits red and blue are the equivalent black circuit in each phase. CM represents the current-mirror.

phase until time t_{MVF} is reached. In the Read phase (switches R on), represented in the blue circuitry, the stored value is transferred via the OTA configured as a unity-gain buffer to the subsequent stage, the subtractor (represented in Figure 2.25 by $Z_{L,DIFF}$). During this phase, the OTA subtracts V_L at the output. The PDH circuit operates in this phase until the comparator processes both MVFs. There is a third operation phase, Reset (switch RST on), in which node L is grounded to set the circuit for detection of new maxima from V_D for a new pair of transistors. Although, the PDH circuit in Figure 2.25 serves as the PDH_{MAX} circuit in the ASB, the PDH_{MIN} circuit can be designed by just replacing the PMOS current mirror with an NMOS counterpart.

Once the topology of the PDH is clear several steps were taken to reach the final design in which the OTA design was the more complex one [72]. The process can be summarized in the next points:

- The design process was conducted using a top-down approach. The initial step involves transmitting specifications to determine the OTA requirements, as well as the geometries of the remaining elements (the current mirror, and the values of C_H and R_C) that optimize the performance of the PDH circuit. In this phase, a Verilog-A model is utilized to substitute the transistor-level description of the OTA, encapsulating all the second-order non-ideal effects considered previously. The subsequent step involves the transistor-level design of the OTA, guided by the requirements established in the first phase.

- In terms of the PDH circuit performance specifications (which are the top level in this methodology), an ideal dynamic behavior and minimal detection errors are sought. Figure 2.26 depicts the various error sources that have been taken into account to evaluate the quality of the PDH circuit. A single step in this context is used as a representative for the input V_D where one RTN event (that is, a trapping followed by a detrapping) needs to be detected. This event should represent a worst-case scenario for the PDH

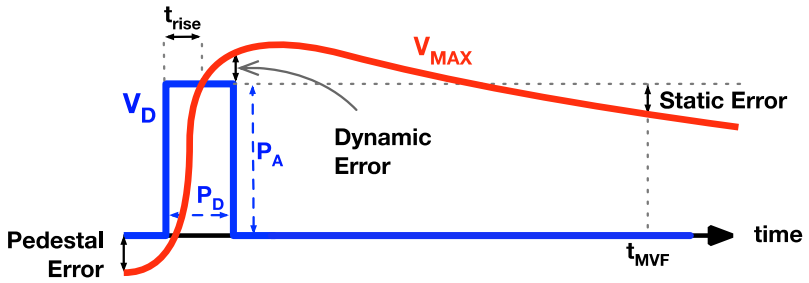


Figure 2.26 Illustration of the errors to consider during the first phase of the top-down design of the PDH circuit. V_{MAX} represents the CMAXV.

circuit. The errors considered in the design of the PDH circuit include the *Pedestal Error* (which accounts for the baseline from which detection may commence), the *Static Error* (measured post t_{MVF}), and the *Dynamic Error* (which gauges the dynamic performance of the PDH circuit). Furthermore, a constraint is placed on the time required for the voltage V_{MAX} (CMAXV) to reach the peak of the step, with $t_{rise} \leq 100ns$.

- During this initial design phase, all these errors were required to be less than $1mV$, as the RTN-induced shifts to be detected are above this value (see Table 2.6), and this has been considered a reasonable balance with design complexity. Moreover, to represent rapid RTN events (thus representing the worst-case scenario for the PDH circuit), the input step signal V_D has a rise time of $1ns$, a pulse amplitude (PA) of $20mV$, and a pulse duration (PD) of $250ns$. Simultaneously, this signal has an input voltage common-mode range ($V_{D,CM}$) of $[350mV - 980mV]$ (see columns CM_{MAX} and CM_{MIN} of Table 2.6), a factor that will be critical in the design of the OTA. This extensive range of common-mode operation could potentially cause issues during the implementation phase. Therefore, it was decided to slightly narrow the range to $[400mV - 900mV]$, which in turn enhances the performance in other specifications. In that sense, several ICCs were discarded (100nA, 200nA and $6\mu A$) for future consideration.

- The transmission of specifications is conducted with the understanding that the PDH circuit is operational during the Write phase. The outcome of the initial design step provides the OTA performance specifications as shown in Table 2.10. In terms of the current mirror, the aspect ratios of the transistors utilized were $W/L = 480nm/200nm$ for the PMOS current mirror in the PDH_{MAX} circuit and $W/L = 160nm/100nm$ for the NMOS current mirror in the PDH_{MIN} circuit. The hold impedance to be used consists of $R_C = 13k\Omega$ and $C_H = 3pF$.

Table 2.10 OTA required and attained specifications

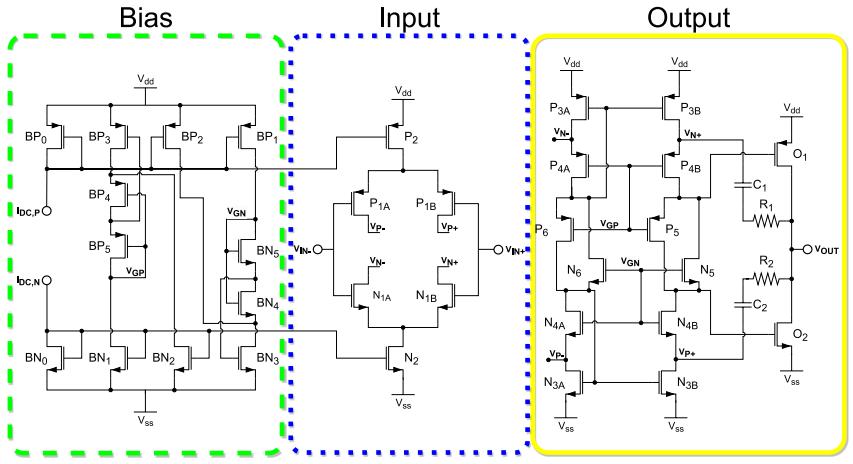
Parameter	$\mathbf{A_o}$ **(dB)**	$\mathbf{f_u}$ **(MHz)**	$\mathbf{R_{out}}$ $(\mathbf{k\Omega})$	**CMRR (dB)**	**SR** $(\mathbf{V/\mu s})$
Required	> 60	> 250	> 10	> 70	> 100
Attained (PEX)	75	322.2	12.6	91.7	158.3

- The top-down process continues to complete the transistor-level design of the OTA. Given the specifications and $V_{D,CM}$, a two-stage rail-to-rail folded-cascode (FD) OTA [73] has been found to be an optimal solution for the PDH circuit. The high f_u of the OTA is determined by Miller compensation capacitors (C_{1-2}) and nulling resistors (R_{1-2}), which guarantee loop stability. Additionally, the entire cell's biasing is achieved by the circuitry

shown on the left side, which sets the correct voltage levels at the gates of cascode transistors (N_4-P_4) and floating current sources (P_{5-6}, N_{5-6}). The input stage in full rail-to-rail (N_1-P_1) enables the circuit to operate across the entire $V_{D,CM}$ range, while a push-pull stage consisting of O_1 and O_2 transistors defines the output.

- The schematic of this OTA is shown in Figure 2.27 where the geometries and values have been listed. Specific matching techniques were used, such as common-centroid and interdigitation techniques, to harden the design against local variations, preventing degrading the performance of critical blocks (differential pairs and current mirrors). Table 2.10 shows the comparison between the required versus the attained specification of the OTA after parasitic extraction. For that, it was necessary a layout implementation shown in Figure B.2 in Appendix B.1.

- A final validation of the proposed PDH_{MAX} circuit with offset cancellation is performed using PEX simulations over a 100-run Monte Carlo analysis, utilizing the input signal depicted in Figure 2.26 with a step of $20mV$. The results of this verification are presented in Table 2.11. To that end, the *Static Error* was used, which compares the theoretical output value to the value obtained at the end of the Write cycle, with $t_{MVF} = 100\mu s$. The PDH_{MAX} circuit was simulated across the $V_{D,CM}$ range to evaluate its performance for various RTN input signals. The findings indicate that the optimal performance is achieved for $V_{D,CM}$ at the mid rail ($600mV$), as both differential input pair stages (NMOS and PMOS) are biased and the overall transconductance of the OTA increases.



Device	*Value*
N_{1A}, N_{1B}	15/0.5 µm/µm
P_{1A}, P_{1B}	60/0.5 µm/µm
N_2, BN_1, BN_2	36/1 µm/µm
P_2, BP_1, BP_2	72/1 µm/µm
N_{3A}, N_{3B}, BN_3	35/1 µm/µm
P_{3A}, P_{3B}, BP_3	112/1 µm/µm
N_{4A}, N_{4B}, BN_4	5/0.3 µm/µm
P_{4A}, P_{4B}, BP_4	15/0.3 µm/µm
N_5, N_6, BN_5	20/0.5 µm/µm
P_5, P_6, BP_5	60/0.5 µm/µm
O_1	75/1 µm/µm
O_2	30/1 µm/µm
R_1, R_2	5.85 kΩ
C_1, C_2	241.6 fF

Figure 2.27 Implementation of the rail-to-rail OTA with its biasing circuitry and the aspect ratios used.

- An additional analysis was performed to evaluate the performance of the PDH circuit with experimental RTN traces. Using 30 such traces, a 100-run Monte Carlo simulation was executed for each one. The outcomes are summarized in Table 2.12, where the *Static Error* for both PDH circuits is demonstrated to meet the initial specification of a mean maximum error of $1mV$ in detecting CMINV and CMAXV.

Table 2.11 PDH_{MAX} circuit performance for a 20-mV step input

$V_{D,CM}$	Static Error (mV)	
	μ	σ
400	0.773	0.144
600	0.647	0.149
900	1.052	0.286

Table 2.12 PDHs performance for 30 real RTN traces

Circuit	Static Error (mV)	
	μ	σ
PDH_{MAX}	0.5319	0.141
PDH_{MIN}	0.8405	0.250

2.3.3.2 Subtractor design

The subtractor already shown in Figure 2.24 was implemented using an Instrumentation Amplifier (IA) [74] whose schematic is shown in Figure 2.28. In this case, it is possible to enhance the performance of the circuit by employing a variable resistor $R1$ that can be selected to adjust the IA gain. Another applied strategy was to reuse the OTA from the PDHs, with an output impedance enhanced by modifying the output-stage transistors.

The possibility to use diverse values of *ICC* was planned during the array implementation in order to influence on how large o subtle are the RTN shifts detected. The set of *ICCs* considered is shown in Table 2.6, removing those *ICC* discarded in the PDH design: $100nA$, $200nA$ and $6\mu A$. From these considerations, the design of an adjustable $R1$ will depend mainly of (1) the output swing of the amplifiers of the IA which has to be carefully considered, and (2) considering the best possible range of the inputs of the subsequent comparator. This will also lead to relaxed specs in the comparator because of the gain enhanced provided by $R1$ in those *ICC* values in which it is necessary. The gain of the subtractor can be represented as depicted in Equation 2.12. Therefore, it is necessary to configure R_1 and R_2 in the first stage, and R_3 and R_4 in the second stage, such that the PDHs' outputs can be amplified through different

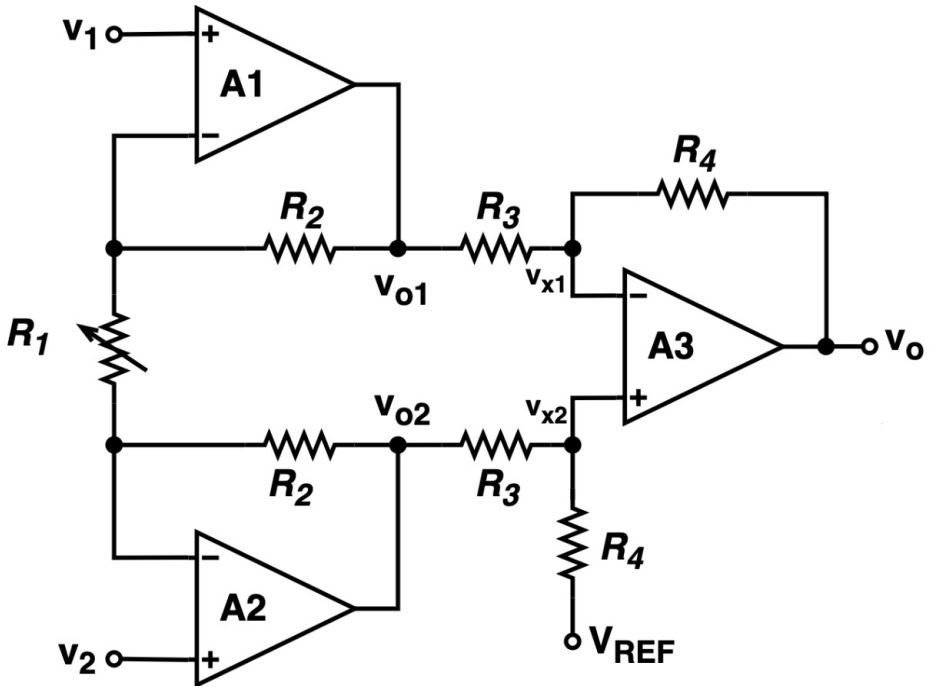


Figure 2.28 Schematic of the IA implemented.

gain stages. The gain of the second stage is set to 2 (this decision will become meaningful later), so $R_4 = 2 \cdot R_3$. With these values, a parametric analysis was conducted to determine the optimal values of R_1, R_2, and R_3 that minimize the area used and could lead to larger ΔMVF ranges in the comparator. Thus, it could be possible to determine the resistances as follows: $R2 = 3k\Omega$, $R3 = 4.8k\Omega$ and $R4 = 9.6k\Omega$, as well as the different values of $R1$ that can be selected depending on the value of ICC shown in Table 2.13. The adjusted $R1$ corresponds to the inclusion of the switches that select between different values of $R1$. The switches used are the same used for the array, detailed in Section 2.2.8. The multiplier column denotes how many resistors are intended to be connected for minimizing the variability in the manufacturing process using common-centroid and interdigitation techniques. Finally, V_{REF} is set to $600mV$ to leverage the operation in the middle range of the input common-mode range of the IA.

$$G = G_1 \cdot G_2 \; ; \; G_1 = 1 + 2 \cdot \frac{R_2}{R_1} \; ; \; G_2 = \frac{R_4}{R_3} \tag{2.12}$$

Another important aspect to deal with is the error that the offset introduce in the gain stage of the IA. Considering that there are two gain stages, any unwanted variations in the input stage will be amplified at the output of the IA. A total of 1,000 IA samples were generated and simulated with varying input common-modes $[400mV, 600mV, 900mV]$ through Monte Carlo simulations. The results indicated that the output offsets could reach values up to

Table 2.13 RTN traces characterization in terms of ICC values

ICC	Used R1 (kΩ)	Adjusted R1 (kΩ)	Area (μm^2)	Gain	Multiplier
500nA	9.43	9.53	41	1.63	x2
700nA	2.94	3.04	13	2.97	x2
1 μA	1.51	1.56	26	4.84	x4
2 μA	0.86	0.91	15	7.61	x4
3 μA	0.53	0.58	9	11.43	x4
5 μA	0.37	0.42	7	15.19	x4

$\pm 8mV$. These values may have a large negative impact on the PUF response. To avoid that, an autozeroing offset cancellation at the output strategy is proposed and implemented finally for offset reduction as Figure 2.29 shows [75]. This offset cancelling approach (with an output hold capacitor, C_H) is the easiest way to implement such a strategy for compensating the error. In practice, the offset can be cancelled in just one phase, but an unexpected effect emerged in the cancellation process that restricts the offset cancellation to two phases. This is primarily because the schematic shown in Figure 2.29 does not account for parasitic capacitances in the input transistors of the IA. Consequently, the gain factor k of the offset cancellation, depicted in Equation 2.13, depends now on the load capacitor, C_L (the capacitance of the IA input transistors). Ideally, when $C_L \to 0$, $k = 1$ completely eliminating the offset. One solution is to set a large C_H to mitigate the effect, but to avoid excessive area usage, two cycles of the same offset cancellation were chosen as the solution ($C_H = 1pF$).

$$V_o/V_i = k = \frac{C_L C_H}{C_L C_H + C_L^2} \tag{2.13}$$

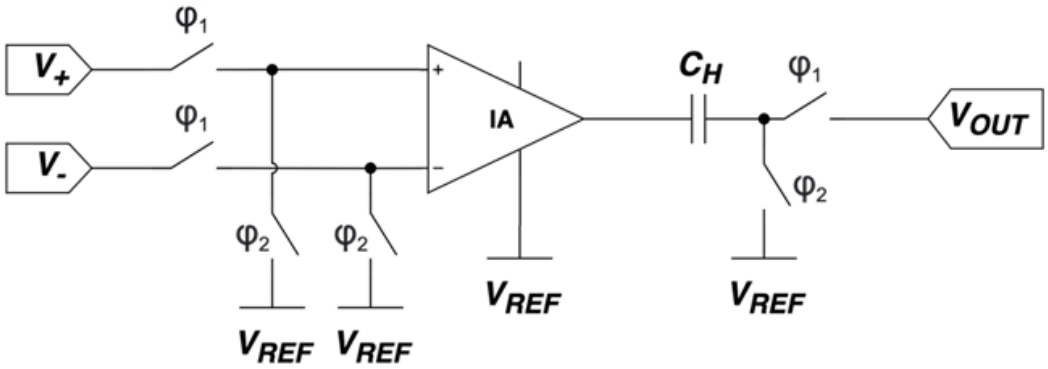


Figure 2.29 Autozeroing offset-free IA schematic.

The last improvement in the ASB topology was the merging of the two IA blocks into a single block. This modification, illustrated in Figure 2.30, greatly reduces the variability due to manufacturing process between MVFs, resulting in a similar impact on both. The output signal of this new design can be mathematically expressed in Equation 2.14. The IA second-stage gain above considered as 2, now can cancel the fraction in the equation. As a result, the error introduced in the PUF is minimized since only one IA is used. By introducing the obtained value in the comparator on the positive terminal, with its negative terminal to V_{REF}, the comparator straightforwardly gives the PUF bit response.

$$V_{out,IA} = G_{IA} \cdot \frac{1}{2}(V_{max,M1} + V_{min,M2} - (V_{max,M2} + V_{min,M1})) \tag{2.14}$$

2.3.3.3 Comparator design

The ASB final stage corresponds to the generation response on the PUF by processing both IA outputs containing the MVF calculation connected to a single-stage comparator circuit. This block processes the difference between the two inputs. A positive differential input at the comparator will generate a '1' as a response bit, while a negative differential input will generate a '0'.

After testing multiple topologies by increasing the number of gain stages (single, double and Miller compensation), it was found that a two-stage OTA fits with the required gain (more than 60dB for a $1mV$ input resolution) as well as with the required common-mode input range of the signal (from 400mV to 900mV). The comparator design process adhered to the method outlined in [76], utilizing the previously mentioned specifications. After determining the transistor sizes through manual calculations, a series of simulations were conducted to verify their functionality. With the proposed transistor sizes, two kinds of simulations were performed: firstly, DC simulations were used to confirm that a response could be obtained

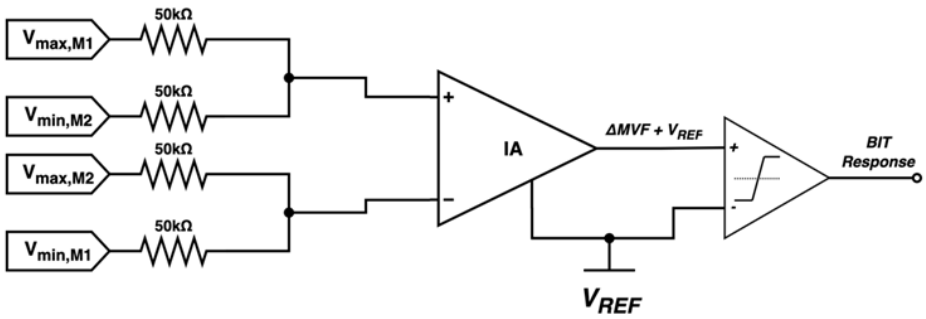


Figure 2.30 Multiple inputs resistor pondered IA plus comparator scheme. V_{MAX} and V_{MIN} correspond to CMAXV and CMINV respectively.

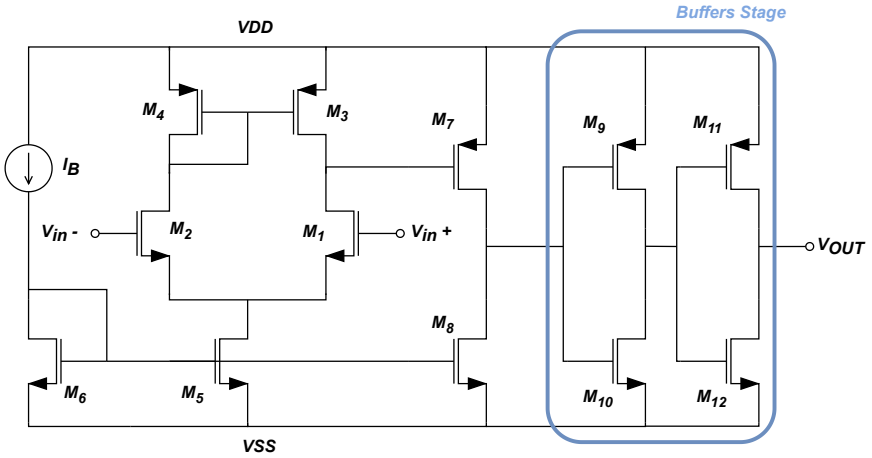
from various common-mode input ranges of the signal; secondly, Monte-Carlo variations were used to examine the offset variation, with size adjustments made to minimize its effect. The importance of both simulations lies in their ability to examine the key specifications that can influence the performance of the comparator, and consequently, impact the PUF response. The schematic of the design is shown in Figure 2.31 in which the transistor sizes of the design are also shown.

Also, the final comparator will not make use of offset-cancellation structures since a well-designed, simple approach copes well with the offset and does not introduce a significant error. In addition to this, a pair of buffers were incorporated into the comparator output to boost its performance, adhering to the approach detailed in [76]. This method enables the comparator to handle substantial capacitive loads without sacrificing speed. The final layout design is shown in Figure B.3 in Appendix B.1.

2.3.3.4 Final ASB design

Once all the module of the ASB have been designed separately, the next step is to merge them to complete the ASB design. The floorplan is conceptually shown in Figure 2.32 and reflects the crucial decision to cross-position the PDH blocks. This helps maintain the signal processing on both branches as symmetrical as possible, thus avoiding the undesired effects introduced by manufacturing process variations. The final layout is depicted in Figure B.4 in Appendix B.1 where each module of the ASB has been highlighted to ease the visualization of the modules' distribution.

Tests using simulated RTN traces were conducted on the entire ASB design to confirm its proper functionality. The results of the parasitic extracted ASB are presented in Table 2.14 for



Transistors	W (um)	L (nm)
M_1 & M_2	48.10	650
M_3 & M_4	32.50	650
M_5 & M_6	73.50	750
M_7	33.52	730
M_8	27	600
M_9 & M_{11}	10	480
M_{10} & M_{12}	1	510

I_B = 200 uA

Figure 2.31 Comparator schematic based on two-stage OTA topology.

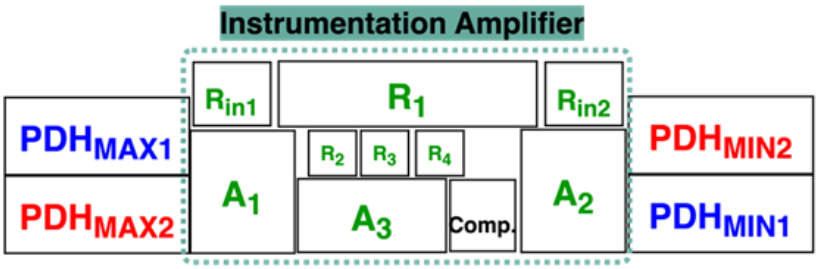


Figure 2.32 Final floorplan of the layout distribution for the ASB.

some combinations of RTN traces. The first two rows represent the same selection of traces (labeled as 62 and 22) but with their positions on the ASB inputs swapped. The displayed data includes CMAXV and CMINV for $t_{MCF} = 100\mu s$, the resulting MVF of each trace, the ΔMVF calculated after the IA operation, and the response returned by the comparator as well as the expected response. These data indicate that the results align with the expected outcomes, demonstrating the ASB's ability to return the anticipated response. While there are slight differences in the measurement of CMAXV and CMINV when the traces are swapped, these do not impact the final result as these differences remain under the previously established upper limit of $1mV$. The third row displays a different selection of traces (labeled as 93 and 48) in a worst-case scenario for the detection in the IA, where it can be observed that the expected result is once again returned.

Table 2.14 ASB characterization using simulated RTN traces and parasitic extraction

Transistor	**Trace**	**CMAXV**	**CMINV**	**MVF**	ΔMVF	**Resp.**	**Exp. Resp.**
M_1	*48*	644.7	636.2	8.5	5.5	1	1
M_2	*22*	672	669	3			
M_1	*22*	672.9	668.2	4.7	-2	0	0
M_2	*48*	643.7	637	6.7			
M_1	*93*	651.4	645.3	6.1	-0.6	0	0
M_2	*48*	643.6	636.9	6.7			
M_1	*48*	642.9	636	6.9	2.3	1	1
M_2	*93*	649.8	645.1	4.7			

2.3.4 Biasing blocks

The last blocks design involves, on the one hand, performing all the biasing current (*IDC*) generation for any amplifier module in the circuitry within the chip design, and on the other hand, generating all the *ICC* identified as suitable (see Table 2.13) to be used in the PUF through an external digital selection.

2.3.4.1 Generation the bias current and reference voltage within chip design

The bias currents utilized in the analog block (i.e., amplifiers) are established with a bandgap (BG) and several current mirror circuits. These circuits set the golden reference current and, reduce or amplify the current signals by copying them, respectively. Additionally, the reference voltage level, V_{REF}, employed in the ASB blocks is generated using an off-chip high-precision resistor. A simplified diagram depicting the generation and propagation of bias currents is shown in Figure 2.33. It can be observed that the golden reference current was set at $50\mu A$. This allows for a reduction in the size relation in the current mirror stage, thereby minimizing the effects of variability due to the manufacturing process. The implemented BG, which includes the start-up circuit responsible for activating the BG when the chip is also turned on, is described in [77]. Additionally, the reference voltage level, V_{REF}, used in the ASB blocks is generated off-chip using a high-precision resistor.

The sizes of the transistors of the BG core and the start-up circuits were initially determined using a parametric analysis of the most suitable size combinations. These were then fine-tuned through several Monte Carlo simulations that included temperature variations. After

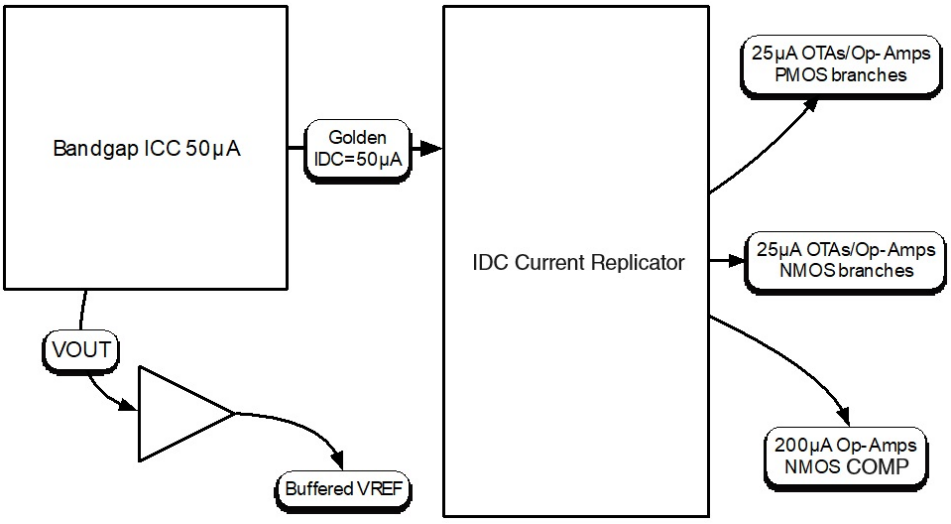


Figure 2.33 Bias current generation within chip design.

the sizes of the transistors were established, a Monte Carlo evaluation was conducted with 200 samples for a temperature range between -20ºC and 80ºC. Initially, the results indicate that there is a fluctuation in the generated V_{REF} voltage for a constant off-chip resistor of $\Delta V_{REF}[-20^oC, 80^oC] = [0.5, 51]mV$. The most extreme case exhibits a variation of $51mV$, while 75% of cases display a variation below $10mV$. A variation of $10mV$ will lead to a change in the gold current of $0.89\mu A$. In the case of the replication of $25\mu A$, this will be $0.45\mu A$. For the most extreme case ($51mV$), the variation in the gold current will be $4.45\mu A$, while the variation in $25\mu A$ will be $2.23\mu A$. Despite this level of variation, after evaluating the ASB with parasitic extraction under differences in the gold current of $[45\mu A, 50\mu A, 55\mu A]$, it was determined that these variations are tolerable for the proper functioning of the circuit. However, these variations can be mitigated with the correct setting of the off-chip resistance, allowing the gold response to be set at $50\mu A$.

2.3.4.2 Generation of the *ICCs*

The process of generating different *ICC* sources adheres to the same strategy outlined above: a gold current can be generated from a BG circuit, which when replicated can produce the set of *ICCs* identified in Table 2.13. The goal is to externally select the desired *ICC* to be applied and replicate it for the two transistors in each of the three arrays on the chip. The purpose of the third array will be explained later. The *ICC* tuning is carried out using a binary control signal plus different transistors to generate the maximum *ICCs* possible. The current generated for the main transistor is $3.2\mu A$ and thus appropriately reduced to obtain $1.6\mu A$, $800nA$, $400nA$, and $200nA$ subsequently. By activating/deactivating these transistors, attaining the desired current levels for *ICC* becomes possible. For the sake of illustration, if the control signal is set in (0,0,1,0,1,0), it will generate $1\mu A$, being the binary word (MSB..., LSB). Figure 2.34 illustrates the scheme used for the *ICC* current generation and distribution.

In the initial strategy, the generation of *ICC* was designed to share the BG circuit with the Bias Current generation. However, due to technical issues related to a general lack of precision in achieving smaller current levels, it appeared that the most effective solution involved: (1) an additional BG circuit and precision resistor to implement the higher values of *ICC*, setting now the gold current in $3.2\mu A$; and (2) in both the schematic and layout, the use of individually physically separated transistors, as opposed to multiple fingers on the same diffusion block, to enhance the performance of the current mirrors, as this makes the output current more accurate.

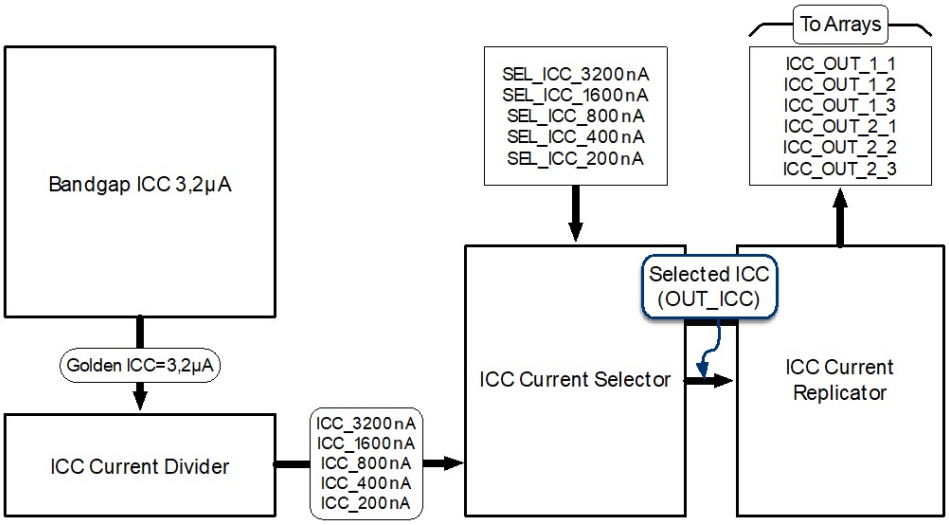


Figure 2.34 *ICC* generation within chip design.

2.3.5 Final layout design

Once, on the one hand, the design of the array and the unity cells that will contain the transistors that will generate a voltage signal containing the RTN has been completed, and on the other hand, the blocks that will be used to extract all that information and transform it into a response bit have been described, the next step is to complete the integration of the RTN-based PUF. This structure can provide a bit response using the RTN as entropy source from a set of 4,096 transistors distributed in an array. Two of these RTN-based PUFs were considered to be included in one single die. This will allow to study inter- and intra-die PUF metrics.

Once all the parts required for the ASIC integration have been described, designed and proved, they can be integrated in the same design. The result is depicted in Figure 2.35, resulting in the so-called MILESTONE-I design. It includes two identical large PUFs comprising an array of 4,096 unit cells and a single Test PUF with aging capabilities but fewer unit cells (1,024 precisely). Additionally, a biasing circuitry composed of two bandgap circuits is intended to provide the proper *ICC* currents to the PUF arrays and generate the current and biasing voltages of the analog sensing circuits across the ASIC. The final PADs distribution after the bounding process of MILESTONE-I is shown in Figure 2.36. The test section was designed to perform several evaluations under different conditions for the transistor array and the ASB within the chip:

- One of the considerations in the implementation of the RTN-based PUF is the impact of aging on the PUF response. As discussed in the conceptualization of the RTN-based PUF, using RTN as an entropy source could potentially mitigate the effects of aging on the involved transistors. To confirm it, accelerated aging capabilities have been enabled in a

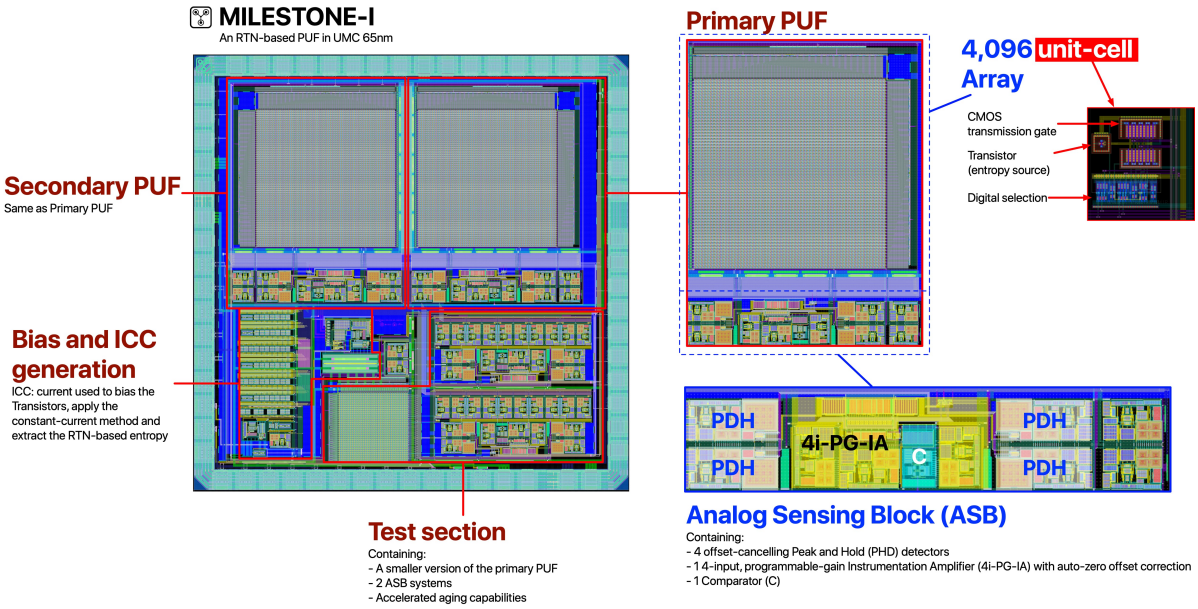


Figure 2.35 MILESTONE-I layout

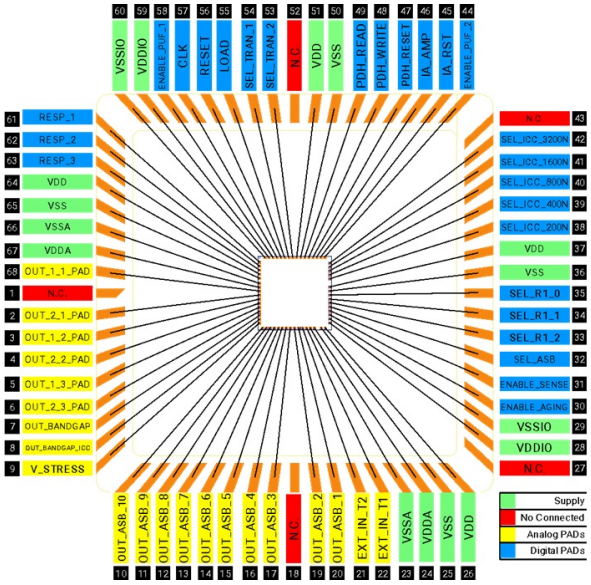


Figure 2.36 MILESTONE-I PADs distribution.

new array of 1,024 unit cells (the third array mentioned earlier). This implementation only required a modification of the unit cell to include a new switch in its design. In this array, when the `ENABLE_AGING` signal is activated, the transistors' V_{DS} voltage is set to V_{STRESS}. This will allow to study the impact of aging on RTN, and consequently, on the performance of the PUF.

- In the evaluation of the ASB, the first strategy involved the integration of ten outputs with internal access to various points within the ASB design. These points function as internal probes, enabling off-chip observation of internal activities. The output of the OTA of each PDH calculation module (PDH_{MAX} and PDH_{MIN}) for each selected transistor (M_1 and M_2) can be monitored, which allows for the verification of circuit timing and further validation of this crucial part of the analog system. Additionally, the output of these four PDH modules can be externally observed, facilitating the comparison of the envelope calculation with the input reference signal. Lastly, the IA output, both pre and post-processing through the offset-canceling mechanism, can also be observed. This feature allows for the evaluation of the autozeroing mechanism's performance on the IA.

- The second approach in the ASB evaluation involved incorporating the option to input an external RTN signal. This signal can be utilized as a reference for generating a response bit, which allows for a thorough evaluation of the ASB's functionality using its internal nodes.

- The test capabilities also cover the possibility to access the RTN signals that the selected transistors are generating. This is the signal that the ASB modules use for the PUF operation in any of the three arrays.

2.3.5.1 MILESTONE-I timing schedule

Another crucial aspect of the chip design is the adherence to a timing schedule in order to generate a PUF response. This timing schedule is segmented into three phases: the operation reset (in GREEN), the selection of the two transistors (in BLUE), and the MVF measurement and response extraction (in RED). Figure 2.37 illustrates the proposed timing schedule for obtaining a PUF response. These phases are further broken down as follows:

1. GREEN: The signals `RESET` and `PDH_RESET` are activated. The first one (`RESET`) carries out the serializers, and the decoders reset. In contrast, the second one performs the reset of the PDH-ASB modules. Notice that this last one (`PDH_RESET`) is active on the high level while the `RESET` signal is active on the low level.

2. LOAD_DATA_SEL: During specific clock cycles, the data of the selected transistors are loaded in the serializers' first stage. In the case of the array of 1,024 transistors, 10 clock cycles are necessary, while in the case of 4,096 transistors, 12 clock cycles are needed.

3. LOAD: The data stored in the serializers' first stage are loaded in the second stage, keeping the identical selected transistors during all measurements.

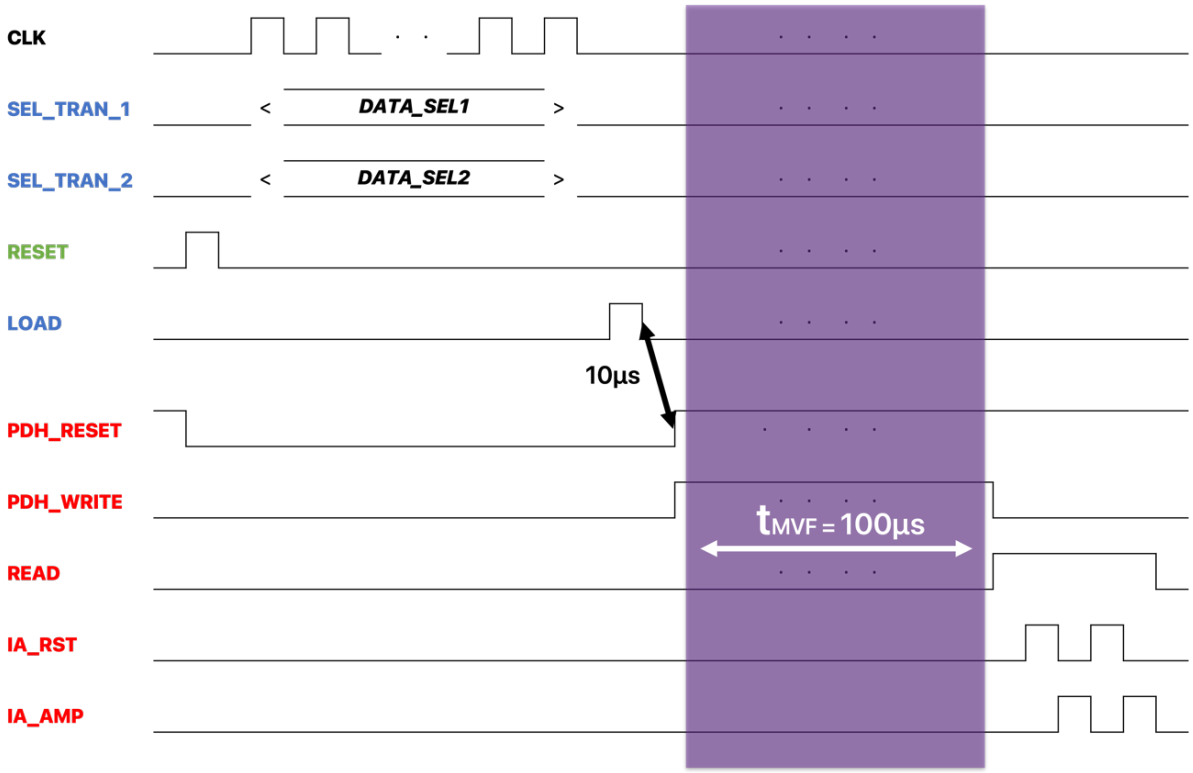


Figure 2.37 The timing schedule to generate a PUF response.

4. STABILIZATION: Given a new challenge, a stabilization process is required to ensure that the voltage received by the ASBs is not influenced by the previously selected transistor.

5. MEASUREMENT: The measurement of the RTN, as well as the response extraction, are performed in a time t_{MVF}. The longer this step is active, the more information it is extracted from the RTN signal.

6. READ: The comparison response from the transistors is read during this clock cycle. This step is necessary as subsequent blocks, such as the IA and comparator, need to process the value. Initially set to $10\mu s$, this value can potentially be reduced and largely depends on the method of IA offset cancellation, given that the comparator is assumed to perform the calculation almost instantaneously.

7. IA OFFSET STORAGE (IA_RESET): The offset is stored at the hold capacitor in this part. The inputs of the IA are connected to V_{REF}, so the output is intended to measure this offset value.

8. IA AMPLIFYING: The response from the IA is outputted without the offset influence ideally. This two-step process is repeated twice to ensure the voltage obtained corresponds to the MVF with the highest precision possible.

2.3.6 Conclusions of the low-level design

In this section the RTN-based PUF conceptualized in the previous section has become a reality by being fully integrated into an ASIC. For this purpose, following the analog design flow, specifications were previously established as well as a floorplan of which elements are necessary to carry out the integration of the PUF. The task was concluded with the MILESTONE-I design. This chip integrates two full RTN-based PUFs that allow the evaluation of intra- and inter-chip PUF statistics. It also integrates the possibility to evaluate the PUF performance under aging stress conditions, as well as additional circuitry for biasing and generating the necessary currents to bias the transistors. This innovative approach serves as a Proof-of-Concept (PoC) to determine the feasibility of constructing PUFs with such entropy source.

2.4 Conclusions

To sum up, PUFs have been recognized as an essential element in the RoT structure. While various types of PUFs exist, silicon-based PUFs have become notable in the IoT realm due to their superior quality and minimal space requirements for the implementation. Among this type of PUFs, this dissertation has presented a novel architecture that utilizes RTN as entropy source.

The initial phase of this endeavor involved characterizing the RTN-based PUF using established performance metrics commonly used for PUF evaluation such as reliability, uniqueness and unpredictability. The evaluation yielded encouraging outcomes, facilitating the identification of key specifications required for silicon implementation, such as the type of transistors to be used, their dimensions, and the biasing conditions.

Following this, the creation of the ASIC implementation required a detailed analog design process. This involved segmenting the design procedure into individual tasks, like the design of the transistor array and the sensing blocks needed to acquire the response bit. These elements were then combined, leading to the establishment of the first-ever RTN-based PUF, named as MILESTONE-I.

Nonetheless, there are avenues yet to be explored in the advancement and deployment of the RTN-based PUF. Once experimental data are collected, numerous queries about the fluctuation of PUF metrics under aging or temperature stress conditions remain unresolved.

Furthermore, alternative methods to minimize the PUF's area need to be investigated. Another significant factor is the potential of the PUF as a True Random Number Generator (TRNG), which continues to be a promising area for future exploration. Numerous suggestions take advantage of the inherently fewer stable pair challenge-response characteristic of a PUF to generate random bits, offering an intriguing path for additional research.

In summation, this dissertation represents a significant step forward in the field of PUFs, specifically in the utilization of RTN as an entropy source. The achievements thus far, including the successful development of MILESTONE-I, lay the foundation for future advancements and applications in secure IoT systems and beyond.

Chapter 3

Hash Functions

3.1 Introduction

In contemporary computer science and information security, hash functions stand as fundamental cryptographic primitives. They play a pivotal role in various applications, ranging from data integrity verification to password storage and digital signatures. A hash function, in essence, is a deterministic algorithm that takes an input (or 'message') and returns a fixed-size string of bytes, typically a digest, which is unique to the input. This property ensures that even a minor alteration in the input will lead to a significantly different hash value.

Hash functions have found extensive employment in securing data transmission over networks, to the extent that they have permeated into fields as diverse as data retrieval, fingerprinting, and data structure design, becoming indispensable tools for computer scientists, software engineers, and information security practitioners. They play a very important role as cryptographic primitives in a wide variety of applications for security purposes [78], such as:

- Verification of message integrity: comparing message digests (hash digests of the message) calculated before and after transmission. This comparison can help determine if the message or file has been altered.
- Signature generation and verification: many digital signature schemes perform signature calculations after hashing the message. In the verification stage, the same hash algorithm is used to corroborate that the message has not been tampered with since it was signed.
- Password hashing: a strong password storage strategy is critical to mitigate data breaches. One solution is to store the hash digest of each password instead of its plaintext version.
- Key derivation: hash-based Keys Derivation Functions (KDFs) derive one or more secret keys from a secret value using hash functions.

A hash function must satisfy three key properties: 1) Collision resistance, it should be computationally challenging to find two distinct inputs that produce the same hash value; 2) Preimage resistance, given a randomly selected hash value, it should be computationally challenging to discover an input message that results in this hash value; and 3) Second preimage resistance, it should be computationally challenging to find a second input that produces the same hash value as any other specified input. These properties ensure the security and reliability of the hash function.

Among the family of cryptographic hash functions published by the NIST, SHA-1 is included, although deprecated since 2011. SHA-2 [79] and SHA-3 [80] are the two hash algorithms that are still in use (see the table with indicators of security strengths for the above properties in [81]). This chapter presents several SHA-2 and SHA-3 hardware implementations, that cover the next points:

- The design of a efficient design that performs a competitive trade-off between the messages processing time and the required area for the Field Programmable Gate Array (FPGA) implementation.
- An extensive comparison with the state-of-the-art FPGA implementations in which it is surpassed the existing literature in terms of efficiency.
- The design of a highly configurable IP, whose configuration enables the possibility to facilitate the implementation of different instances of the SHA-2 and SHA-3 hash functions. For that, it is included an AXI4-Lite interconnection protocol that eases the communication with the system processor.
- It is also included all drivers required to install and invoke the IP module in software environments. The set of instructions and use cases are public in a repository where the performance can be evaluated for testing purposes or to be included in a demonstrator.
- Finally, the ASIC implementation of the SHA-2 hash function is also presented.

3.2 SHA-2

3.2.1 Introduction

The SHA-2 hash function is widely used in security protocols and applications, including TLS and SSL which are cryptographic protocols designed to provide a secure communication channel between clients and servers over the internet. Digital signatures, Secure/Multipurpose

Internet Mail Extensions (S/MIME) email certificates, Pretty Good Privacy (PGP), and IPsec also use SHA-2. And hashing with SHA-2 is also required by law for specific government applications like protecting sensitive data. Additionally, SHA-2 is the industry standard for hashing algorithms (NIST FIPS 180-4 [79]), and it is used in numerous applications to validate and sign digital security certificates and documents.

Many hardware implementations of SHA-2 on FPGAs and ASICs have been reported in scientific literature, as well as commercial IP cores. Two types of architecture can be distinguished. On the one hand, iteration-based structures reuse a single transformation block to perform the SHA-2 hash function and offer efficient implementations in terms of area occupation [82]. On the other hand, pipeline-based structures process multiple messages to improve the throughput [83]. In halfway, quasi-pipelining structures where the compression operations are divided into multiple stages to make the critical paths shorter [84]. Additionally, the work in [84] also redesigns the accumulator register of the compressor as shift register to improve the hardware utilization. In [85], a good compromise between area and throughput of a pipelined SHA-256 hash function implementation is researched using one structure with 4 pipeline stages. Other existing works have used optimization techniques based on arithmetic components to increase timing performance. The work in [86] employs optimized Carry-Save adders in the hash computation to carry out a three-operand addition. This process is as time-efficient as a two-operand adder, thereby reducing the duration of the critical path.

3.2.2 Mathematical background

Table 3.1 shows the parameters of the family of hash functions included in the SHA-2 standard [79].

Table 3.1 SHA-2 family hash functions parameters [79].

Instance	Output Size	Block Size	Rounds
SHA-224 [1]	224	512	64
SHA-256	256	512	64
SHA-384	384	1024	80
SHA-512	512	1024	80
SHA-512/224 [1]	224	1024	80
SHA-512/256	256	1024	80

[1] It is admitted as Legacy until 2025 by SOG-IS [87]

The stages into which the SHA-2 function is divided are the following:

1. *Padding*: The padding process as it is described in [79] is applied depending on whether SHA-2 instance is applied. For a message, $\mathtt{m}$, with a length in bits of l the padding works appending a bit $\mathtt{1}$ at the end of the message. After that, it is appending k zeroes being the non-negative solution to the equation $l+1+k \equiv 448 \bmod 512$ for the case of SHA-224 and SHA-256, or $l+1+k \equiv 896 \bmod 1024$ for the case of SHA-384, SHA-512, SHA-512/224 and SHA-512/256.

2. *Parsing the message*: Once the padding is performed, the message along with the padding, M, is divided in blocks whose size is shown in Table 3.1. This results in $M^{(N)}$ blocks being N the number of total blocks $(M^{(1)}, M^{(2)} \dots M^{(N)})$. Inside the hash function each block is divided in sixteen words 32 bits for the case of SHA-224 and SHA-256, or 64 bits for the case of SHA-384, SHA-512, SHA-512/224 and SHA-512/256. So, the i-th word of the j-th message block is expressed as $M_i^{(j)}$.

3. *Hash computation*: It is repeated for each message block and it is divided into the next parts:

   - *Message Schedule*: This part is carried out following Equations 3.1 for the case of SHA-224 and SHA-256, and 3.2 for the case of SHA-384, SHA-512, SHA-512/224 and SHA-512/256 for the t-th word of the j-th message block. The operations $\sigma_0^{\{256\}}$, $\sigma_1^{\{256\}}$, $\sigma_0^{\{512\}}$ and $\sigma_1^{\{512\}}$ are described in Appendix C.1.

$$W_t = \begin{cases} M_t^j & 0 \leq t \leq 15 \\ \sigma_1^{\{256\}}(W_{t-2}) + (W_{t-7}) + \sigma_0^{\{256\}}(W_{t-15}) + (W_{t-16}) & 16 \leq t \leq 63 \end{cases} \tag{3.1}$$

$$W_t = \begin{cases} M_t^j & 0 \leq t \leq 15 \\ \sigma_1^{\{512\}}(W_{t-2}) + (W_{t-7}) + \sigma_0^{\{512\}}(W_{t-15}) + (W_{t-16}) & 16 \leq t \leq 79 \end{cases} \tag{3.2}$$

   - *Hash initialization*: In this step the eight working variables are initialized. For the first iteration the initial values correspond to the values shown in Table C.3 for the case of SHA-224 and SHA-256, Table C.4 for the case of SHA-384 and SHA-512, and Table C.5 for the case of SHA-512/224 and SHA-512/256 in Appendix C.3. For the rest of cases, it is used the previous hash value in the j-th iteration as it is

described in Equation 3.3.

$$\begin{aligned} a &= H_0^{(j-1)} \\ b &= H_1^{(j-1)} \\ c &= H_2^{(j-1)} \\ d &= H_3^{(j-1)} \\ e &= H_4^{(j-1)} \\ f &= H_5^{(j-1)} \\ g &= H_6^{(j-1)} \\ h &= H_7^{(j-1)} \end{aligned} \tag{3.3}$$

- *Hash computation*: Using the equations described in Appendix C.1 the eight working variables change during 64 cycles as described in Equation 3.4 for the case of SHA-224 and SHA-256, and 80 cycles as described in Equation 3.5 for the case of SHA-384, SHA-512, SHA-512/224 and SHA-512/256. Apart from that, it is used the SHA-2 constants K_t shown in Table C.1 and Table C.2 in the Appendix C.2

$$\begin{aligned} T_1 &= h + \sum_1^{\{256\}}(e) + Ch(e,f,g) + K_t^{(256)} + W_t \\ T_2 &= \sum_0^{\{256\}}(a) + Maj(a,b,c) \\ a &= T_1 + T_2 \\ b &= a \\ c &= b \\ d &= c \\ e &= d + T_1 \\ f &= e \\ g &= f \\ h &= g \end{aligned} \tag{3.4}$$

$$\begin{aligned} T_1 &= h + \sum_1^{\{512\}}(e) + Ch(e,f,g) + K_t^{(512)} + W_t \\ T_2 &= \sum_0^{\{512\}}(a) + Maj(a,b,c) \\ a &= T_1 + T_2 \\ b &= a \\ c &= b \\ d &= c \\ e &= d + T_1 \\ f &= e \\ g &= f \\ h &= g \end{aligned} \tag{3.5}$$

- *Final hash*: The final hash is obtained adding the final result of the eight working works with the previous hash result. If it is the first block the adding is carried out with the initial values. Thus, Equation 3.6 described this operation.

$$\begin{aligned} H_0^j &= a + H_0^{(j-1)} \\ H_1^j &= b + H_1^{(j-1)} \\ H_2^j &= c + H_2^{(j-1)} \\ H_3^j &= d + H_3^{(j-1)} \\ H_4^j &= e + H_4^{(j-1)} \\ H_5^j &= f + H_5^{(j-1)} \\ H_6^j &= g + H_6^{(j-1)} \\ H_7^j &= h + H_7^{(j-1)} \end{aligned} \tag{3.6}$$

4. *Final hash truncation*: In order to obtain the desired output size relying on the SHA-2 instance and following the requirement shown in Table 3.1 it is necessary to compute a truncation. In the case of SHA-256 and SHA-512 the output hash is directly the concatenation of the eight words (32-bit and 64-bit word, respectively) obtained in the final hash operation. For the case of SHA-224, the output hash is formed using the concatenation of the seventh first hash words. For the case of SHA-384, SHA-512/256 and SHA-512/224, the output hash is formed using the sixth, fourth and third (and the half of the fourth) first hash words respectively.

3.2.3 Proposed scheme

The aim of the proposed SHA-2 scheme has been the minimization of the number of resources used in the hardware implementation and, secondly, to maximize the operation frequency. These two features will lead to an increase in the SHA-2 performance which will be deeply detailed later. The proposed architecture for the Message Schedule process can be seen in Figure 3.1. This design is based on a Linear Feedback Shift Register (LFSR) in which the inputs of the Arithmetic Unit (AU) are W_{i-2}, W_{i-7}, W_{i-15}, W_{i-16} or $REG[14]$, $REG[9]$, $REG[1]$ and $REG[0]$, respectively. While the *load* process is carrying out the first 16 message blocks, M_t are stored in the LFSR. The key to this scheme is that it does not require the message schedule function to be completed first in order to start the SHA-2 Rounds, but rather these coefficients are generated as the rounds are being executed. Thus, the number of hash operation cycles is 60 for SHA-224 and SHA-256, 80 for SHA-384, SHA-512, SHA-512/224 and SHA-512/256.

This leads to the design of the SHA-2 core scheme as shown in Figure 3.2. In this scheme, the memory of the hash result (*MEM H*) stores the initial hash value depending on the SHA-2

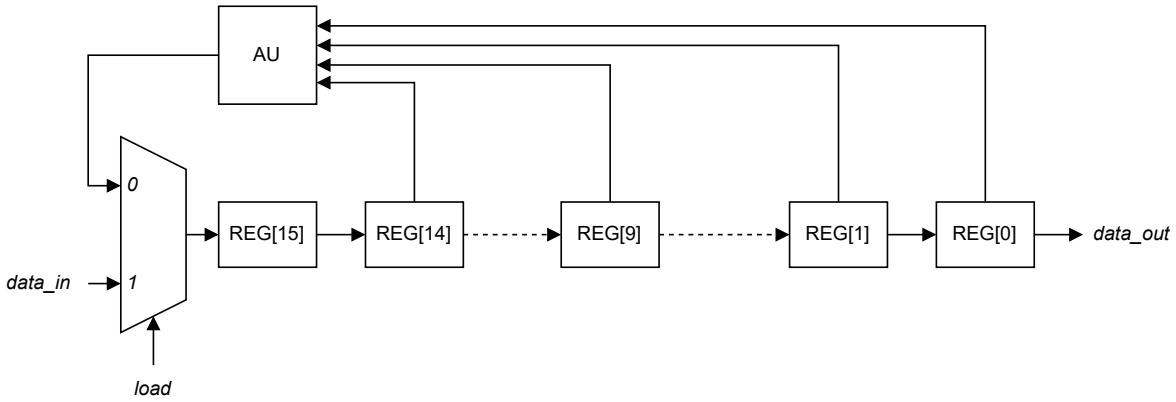


Figure 3.1 Message Schedule scheme.

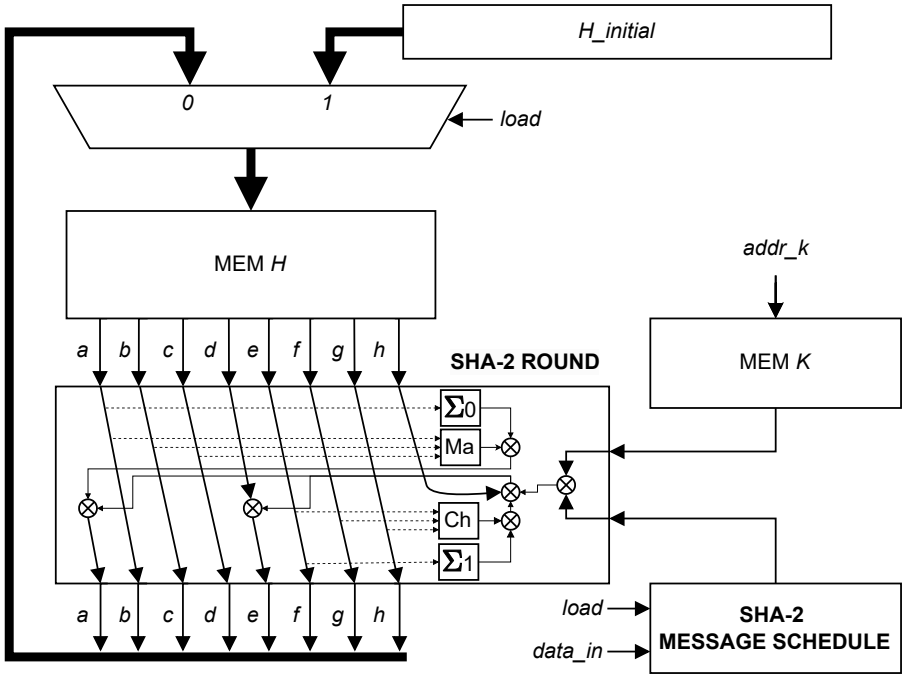


Figure 3.2 SHA-2 core scheme.

version as it is indicated in the FIPS 180-4 standard if *load* signal is activated, otherwise it stores the output of the SHA-2 Round module. This module requires the input of the MEM H itself as well as the *K* coefficients of the SHA-2 version stored in a ROM memory whose selection is controlled by an address pointer *addr_k*. The data generated by the SHA-2 Message Schedule module is directly used in the SHA-2 Round module.

After completing the design of the SHA-2 core, the SHA-2 IP Module can be wrapped using an AXI4-Lite communication protocol. Several modifications have been also applied

to the padding module as well as to the Control module. The aim of these modifications is to minimize to the maximum possible the resources used in its implementation. The input memory is used to synchronize the input data from the software with the input data of the SHA-2 core. The control module generates all logic control for the SHA-2 core as well as the address pointer for the k module.

3.2.4 Implementation of all SHA-2 versions

The SHA-2 core modifications required to include these versions are mainly related to the adaptation of the initial hash, illustrated in Figure 3.2, used at the beginning of the hash operation. Also, the coefficients of the k memory were also adapted to each version. The Control logic, shown in Figure 3.3, was modified to count 60 or 80 cycles depending on the version implemented.

To evaluate performance at the hardware level, two metrics have been used that are widely used in the state of the art. First, the ability to digest a certain number of bits per unit of time is called *throughput*, given in Equation 3.7. *BS* refers to the number of bits of the input block to be processed, f_{max} is the maximum possible operating frequency of the hardware module, N_{CC} is the total number of clock cycles required to complete the hash operation, and finally, N_{msg} symbolises the number of messages that can be digested at the same time [88]. Secondly, *efficiency* is another important parameter, which relates the capacity to process per unit of time

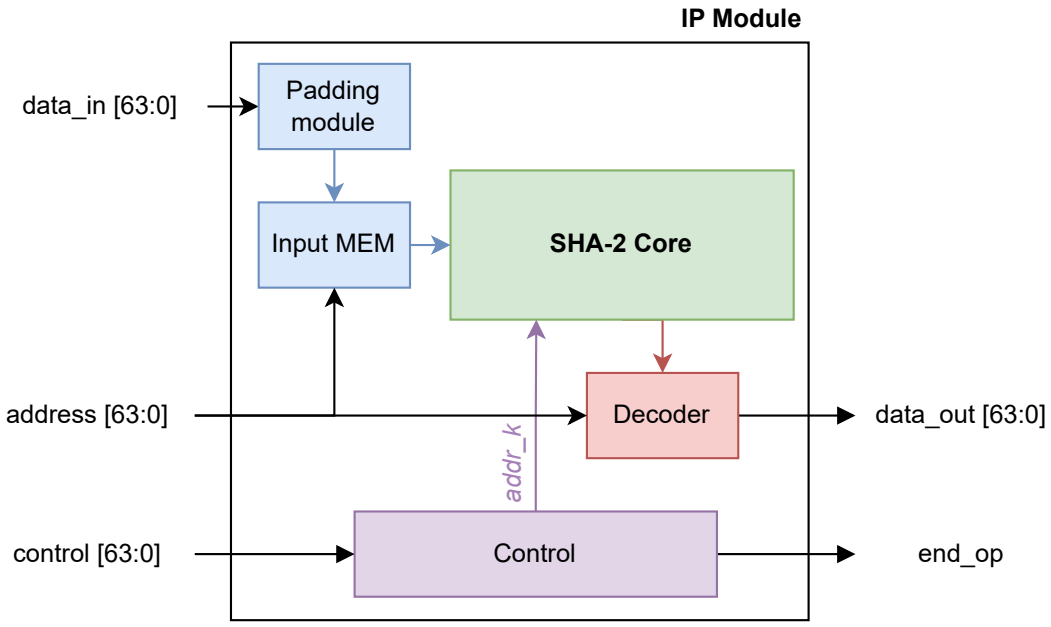


Figure 3.3 Schematic of the SHA-2 IP module.

(i.e. throughput) to the resources cost of a certain implementation, as shown in Equation 3.8. In this case, since the analysis is performed on FPGA implementations, the cost is related to the number of slices used.

$$Throughput = \frac{BS \cdot f_{max}}{N_{CC}} \times N_{msg} \tag{3.7}$$

$$Efficiency = \frac{Throughput}{Cost(\#Slices)} \tag{3.8}$$

Table 3.2 collects all the possible SHA-2 implementations in a Kintex UltraScale device in terms of throughput and efficiency, while Table 3.3 collects various SHA-2 implementations in the literature and provides a comparison to the SHA-2 implementations presented in this dissertation. This design is technology independent, so it is possible to implement it on a different board, such as a Virtex-7 board. In the case of the SHA-256 implementation, it is

Table 3.2 Performance of all hash function in the SHA-2 family.

SHA-2 Version	Block Size (bits)	CCs	Frequency (MHz)	Area (Slices)	Throughput (Gbps)	Efficiency (Mbps/Slice)
SHA-224	512	60	214.41	157	1.829	11.654
SHA-256	512	60	218.72	165	1.866	11.312
SHA-384	1024	80	185.19	352	2.370	6.730
SHA-512	1024	80	186.05	352	2.381	6.765
SHA-512/224	1024	80	185.77	352	2.377	6.755
SHA-512/256	1024	80	185.19	352	2.370	6.730

Table 3.3 Comparison of the SHA-2 HW implementation with the state of the art.

Ref.	Platform	SHA-2 ver.	Area (Slices)	Frequency (MHz)	Throughput (Gbps)	Efficiency (Mbps/Slice)
[89]	Virtex	SHA-256	755	174.00	1.370	1.830
[90]	Virtex	SHA-256	6136	35.10	2.077	0.338
[91]	Virtex-5	SHA-256	387	202.54	1.580	4.190
[92]	Virtex-4	SHA-256	610	170.75	1.345	2.200
[93]	Virtex-5	SHA-256	1895	411.30	3.290	1.740
This diss.	**Virtex-7**	**SHA-256**	**282**	**191.13**	**1.630**	**5.784**
[89]	Virtex	SHA-512	1667	141.00	1.780	1.010
[91]	Virtex-5	SHA-512	874	176.06	2.200	2.580
[94]	Virtex-5	SHA-512	1080	129.00	0.826	0.765
This diss.	**Virtex-7**	**SHA-512**	**576**	**173.52**	**2.221**	**3.856**

remained in a competitive position in terms of throughput, but it is in the case of *efficiency* that achieves a very significant improvement over other implementations in the literature. In the case of SHA-512, this design outperforms all implementations already presented in the literature, both in terms of throughput and efficiency.

3.2.5 Embedded system integration and results

The design of the IP module includes the user interface shown in Figure 3.4. As can be seen, it is possible to select the module according to the SHA-2 version desired: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, or SHA-512/256. This IP module can be found in an open repository available in [95] as `sha2_xl_3_0`. The compilation process to obtain each version only requires one parameter which is passed directly in the compiler call: `SHA_224`, `SHA_256`, `SHA_384`, `SHA_512`, `SHA_512/224` or `SHA_512/256`. The test suite includes all NIST tests for byte and bit inputs for each version released in [96] and [97], respectively.

For the performance evaluation, PYNQ-Z2 board [98] was selected, based on the Xilinx Zynq-7000 SoC which incorporates an ARM as the Processing System (PS) and a Xilinx Artix-7 as the Programmable Logic (PL), along with the PYNQ C-API provided in [99]. All SHA-2 instances have been considered in this evaluation as Figure 3.5 shows. The IP modules have been connected to the processor using the AXI Interconnect module supplied by Xilinx.

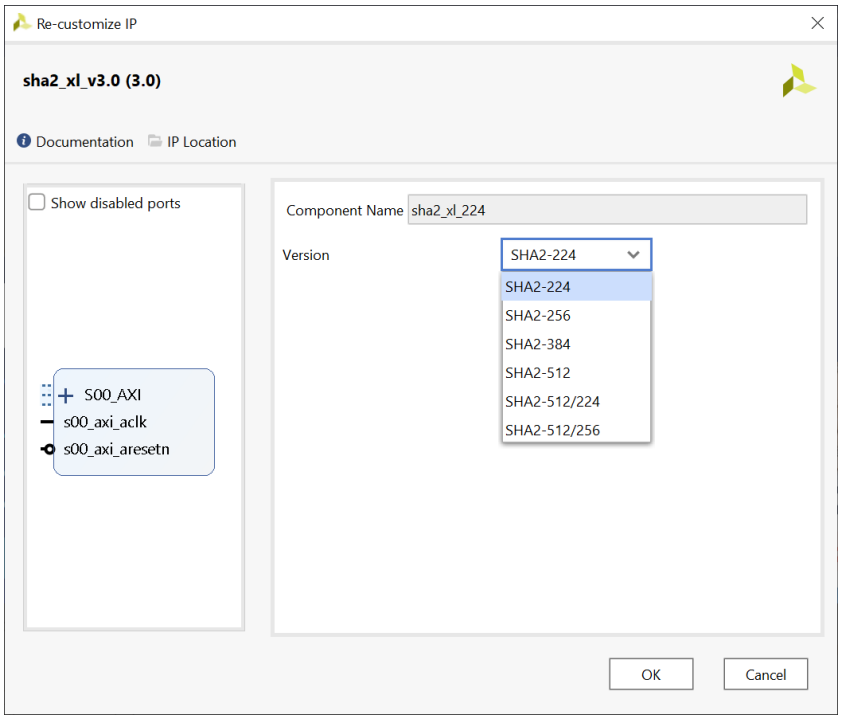


Figure 3.4 User interface of the SHA-2 IP Module.

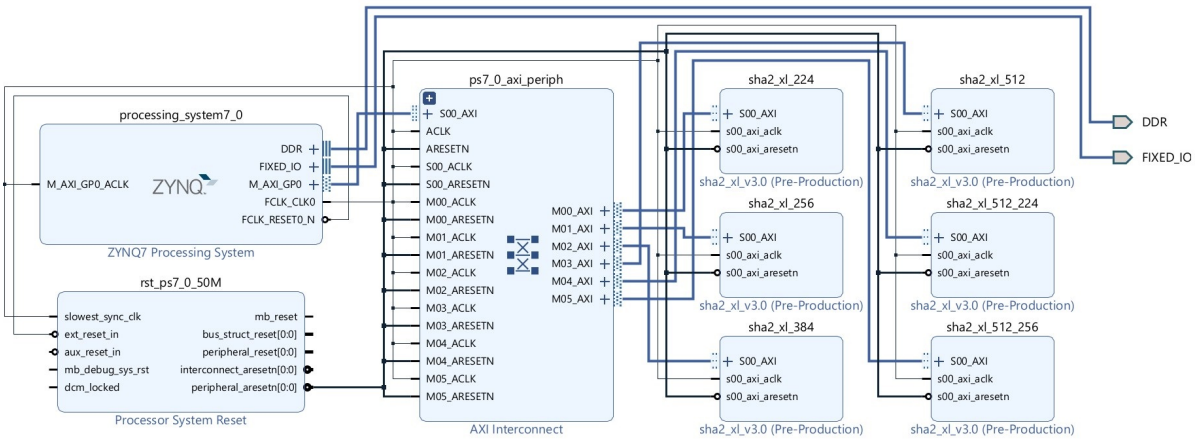


Figure 3.5 Block Diagram of the SHA-2 IP Module integration in an embedded system

The software implementation of SHA-2 [100] has been recently updated and is therefore highly optimized. As a result, this SW version performs the SHA-224 and SHA-256 operations in less time than the HW implementation as can be seen in Table 3.4. This is not the case for the other SHA-2 versions where the HW takes less time to process the data. This is due to the long time required to send and receive data to and from the IP module when using shared memory. A reduction in these times could be achieved by modifying the communication protocols. Apart from that, `sha2_xxx_demo` introduces new functionalities such as the possibility to perform hash functions for hexadecimal (`-m`) or plain ASCII text strings (`-t`) provided through the command line, as well as introducing these data through a file as hexadecimal (`-mf`) or plain ASCII text strings (`-tf`).

Table 3.4 Acceleration of the SHA-2 HW implementation vs SHA-2 SW implementation of [100]

SHA-2 Instance	Message Length	
	Smaller than Block Size	Bigger than Block Size
SHA-224	x0.68	x0.87
SHA-256	x0.67	x0.87
SHA-384	x1.54	x2.35
SHA-512	x1.44	x2.32
SHA-512/224	x1.64	x2.35
SHA-512/256	x1.64	x2.37

3.3 SHA-2 low-level design

After the prototyping of the proposed SHA-2 scheme on PL, its integration in a fully-digital dedicated ASIC on a nanometric technology will provide a reduction of size, memory resources and power consumption requirements, making them more suitable for wearable and/or low-power secure applications. The implementation of SHA-2 in a dedicated silicon ASIC brings new challenges in terms of viability, design, and obtained performances of the solution. The migration from FPGA to ASIC design is far from being automatic since great efforts in both front-end and, especially, back-end processes are needed. Also, specific design methodologies and Computer Aided Design (CAD) tools have to be used.

The selection of the technology between available foundries and integration processes has been made on the basis of the previous knowledge of the designers' group, the availability of CAD tools and technological libraries, the runs for integration provided by the Europractice consortium, and the trade-off between cost and expected performances. The selected technology was TSMC 65nm, to be fabricated under the MiniASIC Programme of Europractice and the funds for the integration comes from the European projects SPIRS [101].

The Hardware Description Language (HDL) description of the SHA-2 used in the FPGA implementation was the starting point for the ASIC integration. For that, a methodology for front-end digital design was applied. However, it was necessary to introduce several modifications as follows:

- The synthesis tool used in the FPGA design flow infers Block Random Access Memories (BRAMs) as memory units to storage message words (W_t) and message digests. In ASIC integration, registers and memory cells included in the TSMC technology library are used after synthesizing a technology independent HDL description.
- Ad-hoc serial input interface has been designed to reduce the number of input/output ports used in the ASIC integration of the SHA-256. This interface includes a protocol to control the data writing and reading, which will be detailed below.

3.3.1 Description of the SHA-256 ASIC implementation

The total number of input/output ports for the ASIC implementation has been drastically reduced down to 10 (the previous design implemented on programmable devices used more than 100 ports). Table 3.5 shows the name, direction and a short description of each port used in the ASIC integration of the SHA-256. The width of the input (`i_data_in_S`) and the output (`o_data_out_S`) data signals have been reduced down to one single bit. The width of the control signal (`i_control_S`) is also minimized down to 3 bits with the help of a decoding

Table 3.5 Port description in the SHA-256 design for the ASIC integration

Port name	Port Direction	Description
i_clk_S	Input	Clock signal
i_reset_n_S	Input	Synchronous active low reset
i_data_in_S	Input	Input bit data
i_control_0_S	Input	Bit 0 of Control [2:0]
i_control_1_S	Input	Bit 1 of Control [2:0]
i_control_2_S	Input	Bit 2 of Control [2:0]
i_enable_load_S	Input	Enable input data (load)
i_enable_read_S	Input	Enable output data (read)
o_data_out_S	Output	Output bit data
o_end_op_S	Output	End operation flag

block. The truth table of the 3 to 5 decoder is shown in Table 3.6. The `i_enable_load_S` and `i_enable_read_S` inputs enable an external control in the data loading and reading processes.

The block diagram of the SHA-256 design for the ASIC implementation is shown in Figure 3.6. This block uses an HDL description that is practically similar to that already presented in Section 3.2. The only difference is the adaptation of the inputs and outputs to the ad-hoc serial protocol to send/receive data to/from the ASIC. The control of the incoming data is governed with a module called CONTROL in Figure 3.6. It is in charge of generating the load signals for the input data, as well as the memory addresses for their storage. It contains a decoder whose output signal (`control_dec`) are associated to six possible control values required by the SHA-256 block. Each value corresponds to an action that is described in Table 3.7.

The input data of the SHA-256 is provided by the Serial-In Parallel-Out (SIPO) block in Figure 3.7. It implements a SIPO shift register where the input data is given bit by bit serially. The SIPO is built using 64 D Flip-Flops (FFs) which are connected in cascade as it is shown in Figure 3.7. For each clock pulse, the input data at all the FFs can be shifted by a single position

Table 3.6 Truth table of the decoder used in the Control module

i_control_2_S	i_control_1_S	i_control_0_S	control [2:0]	control_dec [4:0]
0	0	0	000	00000
0	0	1	001	00001
0	1	0	010	00010
0	1	1	011	00100
1	0	0	100	01000
1	0	1	101	10000

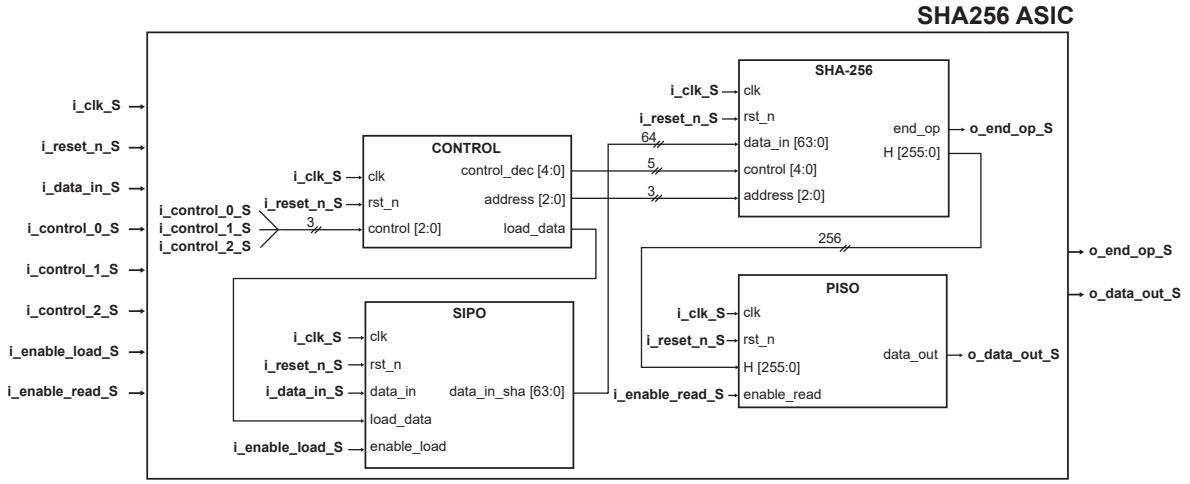


Figure 3.6 Block diagram of the ASIC integration of the SHA-256 algorithm

Table 3.7 Actions associated to the values of the control signals

control [2:0]	control_dec [4:0]	Action description
000	00000	NO ACTION
001	00001	RESET
010	00010	LOAD LENGTH
011	00100	LOAD DATA
100	01000	START OPERATION
101	10000	READ DATA

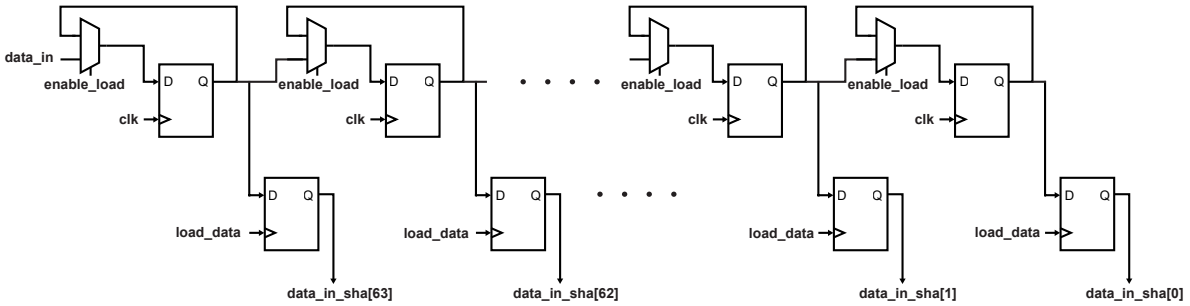


Figure 3.7 SIPO diagram used for input data of the SHA-256 block

if `enable_load` is active. During the first clock cycle, the Least Significant Bit (LSB) of the `data_in` must be provided. As long as the `enable_load` signal is active the input data will be shifted to the right until it reaches the end. At that point, the CONTROL block generates the `load_data` signal that stays high for one clock cycle and loads the data to the lower bank of 64 registers in parallel. The `load_data` input only remains active during one clock edge and

ends with the subsequent one. The `load_data` signal will stay low until the SIPO shift register is not ready with a new value of the input data, and therefore, the `data_in_sha` signal will not be modified. Each time the `load_data` signal is generated, the corresponding storage address is also updated by the CONTROL block.

A Parallel-In Serial-Out (PISO) shift register is used to change data output of the SHA-256 block from parallel to serial form (see Figure 3.6). This shift register works in a reverse way to the SIPO shift register, the `H` signal enters in a parallel way and comes out serially. The diagram of the PISO that includes 256 connected D FFs as shown in Figure 3.8. The initial HDL description of the SHA-256 for programmable devices was slightly modified to make it compatible with the PISO interface. When the `control_dec` signal activates the read data process, the SHA-256 block is prepared to send the output data. One control signal called `enable_read` is used to control the parallel input and serial output. When the `enable_read` signal is activated the Most Significant Bit (MSB) of `H` values shift from the left to the right and they are sent to the output serial port (`data_out`).

The timing diagrams for the SHA-256 ASIC integration are detailed in Figures 3.9, 3.10 and 3.11. A short description of these diagrams is given to fully understand the evolution of signals in the time domain:

1. During the first two clock cycles of Figure 3.9, the (`i_reset_n_S`) signal remains low (logic zero), activating the general reset of the ASIC module. This signal acts independently of the control value, which, for the sake of illustration, it is set in NO ACTION value (see Figure 3.9) since the values of `i_control_2_S`, `i_control_1_S` and `i_control_0_S` are null (see Table 3.7).

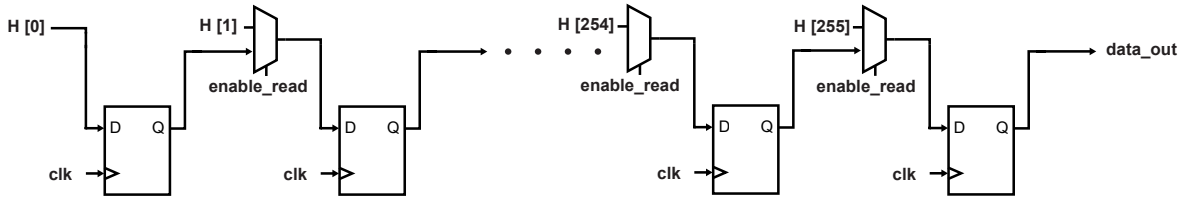


Figure 3.8 PISO diagram used for output data of the SHA-256 block

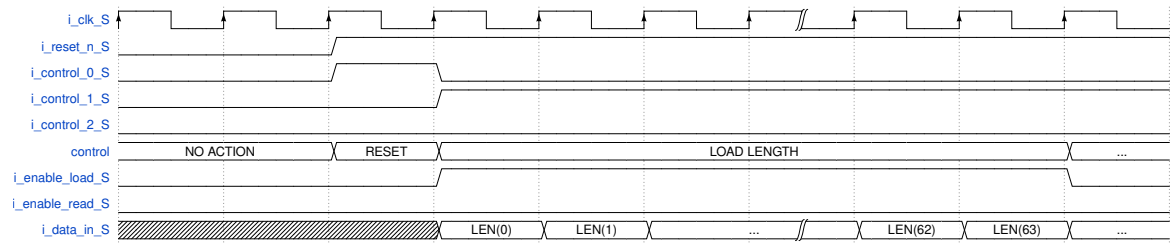


Figure 3.9 Timing diagram to load the message length in the SHA-256 ASIC integration

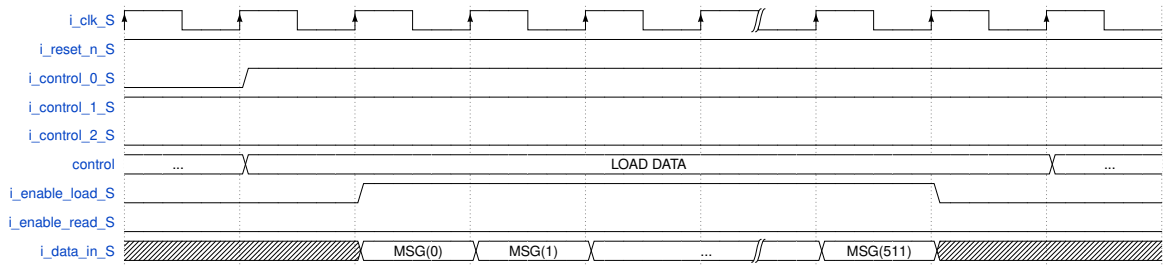


Figure 3.10 Timing diagram to load the message in the SHA-256 ASIC integration

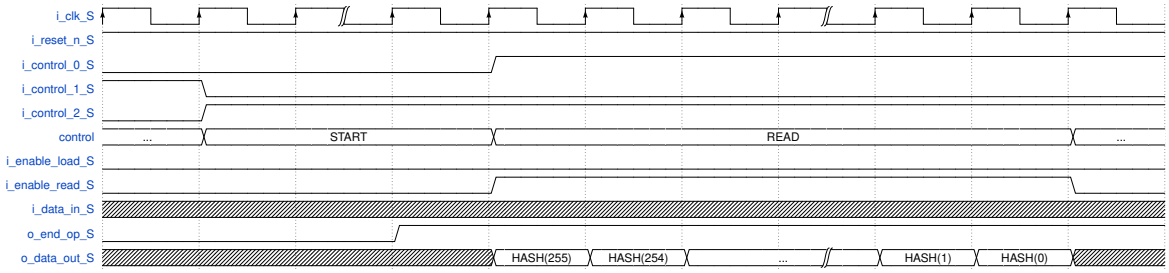


Figure 3.11 Timing diagram to read the digest value in the SHA-256 ASIC integration

2. The SHA-256 ASIC design is a power-up or power-on reset circuit, which means that the reset signal has to be asserted to a high value (logic one) after power is first applied to the circuit and becomes stable. In the third clock cycle, the `i_reset_n_S` signal is deactivated (its value goes to logic one with the rising edge of the clock signal). While the `control` signal evolves to the RESET value according to the values of `i_control_2_S`, `i_control_1_S` and `i_control_0_S`. This corroborates that the external reset is independent of the internal *control* signal that is set by the three external signals (`i_control_X_S`).

3. During the fourth clock cycle, the control signals (`i_control_2_S`, `i_control_1_S` and `i_control_0_S`) are configured to reach the LOAD LENGTH value with the goal of providing the length of the message to be hashed. Thus, the input data protocol acts providing the LSB called LEN(0) during the fourth clock cycle through the `i_data_in_S` input. Figure 3.9 shows the evolution of the `i_data_in_S` signal from an indeterminate value (striped area) to LEN(0). Internally, data will be provided to the SIPO block. In parallel, the `i_enable_load_S` goes to a logic one to load the LEN(0) value to the first SIPO shift register.

4. In successive clock cycles (see Figure 3.9), the length of the message is transmitted serially, one bit per each clock cycle. In parallel, it is loaded to the SIPO registers since the `i_enable_load_S` remains active.

5. Once the message length has been written, the next step is to provide the message block to be hashed. The timing diagram to load the message is shown in Figure 3.10. As can be seen, the control signals (`i_control_2_S`, `i_control_1_S` and `i_control_0_S`) must change to LOAD DATA value. Again, the SIPO registers start to shift the bits of the message only if the `i_enable_load_S` signal remains active.

6. The hash process can start once the message length and the block message have been fully transmitted. At that moment, the input control signals can be configured to reach the START (see the `control` value in Figure 3.11).

7. When hashing ends, the signal (`o_end_op_S`) is activated and there are two different options: i) load a new message block if the message has not completely digested; ii) read the final hash value if the entire message has been processed. In the first case, it is returned to point 5 and the `control` signal again reaches the LOAD DATA value. In the second case, the `control` signal provides the READ value. For the sake of simplicity, it is only illustrated this second option in Figure 3.11. The reading process using the PISO interface will not begin while the enable read signal (`i_enable_read_S`) is not active. In the reading process, the MSB of the hash value (HASH(255)) is sent firstly as shown in Figure 3.11.

3.3.2 Synthesis and Validation

After completing the front-end design flow, a synthesized design of the SHA-256 implementation is obtained and validated using post-synthesis simulations. On the one hand, Table 3.8 shows the expected area occupation of the SHA-256 block after synthesis. The worst timing slack of this block after synthesizing it with a clock period of 10 ns is 2 ps. On the other hand, the testbenches are written in Verilog to corroborate in the simulator that:

- Instantiates and initializes the design.

Table 3.8 Post Synthesis SHA-256 Occupation

Module	Cell Count (GE)	Cell Area(μm^2)	Net Area(μm^2)	Total Area(μm^2)
SIPO	261	1317.240	742.423	2059.663
PISO	516	2507.400	1479.523	3986.923
Control	52	181.440	2594.252	2775.692
SHA-256 (Core)	19350	118787.400	70292.775	189080.175
Total	20179	122793.480	75108.973	197902.453

- Generates and applies stimulus to the design.
- Monitors the design output result and checks for functional correctness.

The testbenches include stimulus that read test vectors used in the Cryptographic Algorithm Validation Program (CAVP) provided by NIST [102]. Tests included in CAVP allow to validate the SHA-256 implemented in the ASIC according to FIPS 180-4 standard (Secure Hashing) available at [79]. Thus, the test benches follow the same structure:

1. Initialize all inputs to the design within the test bench at simulation time zero to properly begin simulation with known values.
2. Apply the reset pulse.
3. After 4 clock cycles, apply stimulus data to set the `i_control_X_S` control signals to RESET (see Table 3.7).
4. During 64 clock cycles, apply stimulus data to set the `i_control_X_S` control signals to LOAD LENGTH, and simultaneously, stimulus to send input data for hashing according test vectors of the CAVP.
5. During 512 clock cycles, apply stimulus data to set the `i_control_X_S` control signals to LOAD DATA.
6. Apply stimulus data to set the `i_control_X_S` control signals to START OPERATION, which starts the message digestion.
7. The testbench waits until the `o_end_op_S` signal is activated, at which point it returns to LOAD DATA if there is still information to be hashed, or to READ if the entire message has not been completely introduced.

After completing the post-synthesis simulations with stimulus of a pair of tests from NIST CAVP, Figure 3.12 and Figure 3.13 show the results obtained. It is possible to observe how the results obtained by the synthesized module and the results provided in the CAVP are the same, concluding that the module is working properly.

3.3.3 ASIC layout and tapeout

In this phase, the design place & route phase is performed as well as its verification against the synthesis netlist using formal equivalence check. Before tape-out, a complete sign-off verification process is made from Virtuoso Cadence Design. Once validated, the design is submitted for fabrication. Thus, the final layout of the ASIC is shown in Figure 3.14. As it

```
                ############ TEST NIST:          10 ############
LENGTH:                   80
MSG: 74cb9381d89f5aa73368
HASH_NIST: 73d6fad1caaa75b43b21733561fd3958bdc555194a037c2addec19dc2d7a52bd
HASH_SHA2: 73d6fad1caaa75b43b21733561fd3958bdc555194a037c2addec19dc2d7a52bd

CORRECT
```

Figure 3.12 Post-synthesis simulation result of NIST test number 10

```
                ############ TEST NIST:          69 ############
        LENGTH:                 4472
        MSG: 9d64d891d99bb8aba23a29a8f69b32482714e031d31dde3317b046d000f6b7fc421fa8212d91fb66dc46d531
eeafd5ea40302a215351f746c0c42523ba5a3e98bb7b13870d04bf3e0e13425c4fdc11a505ed57c90a90fbc447242b3ee032\
68a29594dd73c705808efc16a059e08dd118b4a34f178175151760de963f89d34c92b12e9b58ace694fadd73a576193b80bf\
ed0074bf5074cfba9e21da980fb366f39e76d1b8073e88ebf2d8d623827bad051f736d02e02688185fbc7ccaea69244fae2c\
15146e63b8ed0cb496f494b4b272bc8aac94c8f0dadb45fd015ab25b210170acd9f05afcc1786b758c6bc87d3d93449497d7\
637a345db161ecc9f00fc9b37677a4de55701f189fba0afba63baaf1584fc36d5819212a5299b39b2c0daad0302aea20d654\
4e3829f0b726b68686e7681ac3a91f543dcb79f2da30aecb30d23e252e7a661fcb619a98056f61d46e1fe473fd3d11b1c6bb\
c80be54d20cee843e0f4f65d7d49032f523e6a4830abacf56de9f46bd7c86865ad4359230a9f5dafc928b61c9456a1fbf142\
7a53cb82dff264eb2de7f9feaf739a47aa64c4a2fd70772f026a33cf1451e852a9e47ae083a159f62e23c0cae8402f775d84\
f77044204b765fb8e418d6cbb7dd7dacc74b148cbda95991f4c3cf65dd60e6f61b8dce59e6ad127b2dda65b3d0416a0f4939\
2f1f107354c4de6fa14f1482db5a9961f867b921ef33697a4db4d22cf37e69211fd2f2c2944f16252a86755baf0509835ee4\
33733a743f8f0b493e0eae8cb
        HASH_NIST: db593a375cb27df689cd78b5154949e5bc30094a05d704c0295d547385176662
        HASH_SHA2: db593a375cb27df689cd78b5154949e5bc30094a05d704c0295d547385176662

        CORRECT
```

Figure 3.13 Post-synthesis simulation result of NIST test number 69

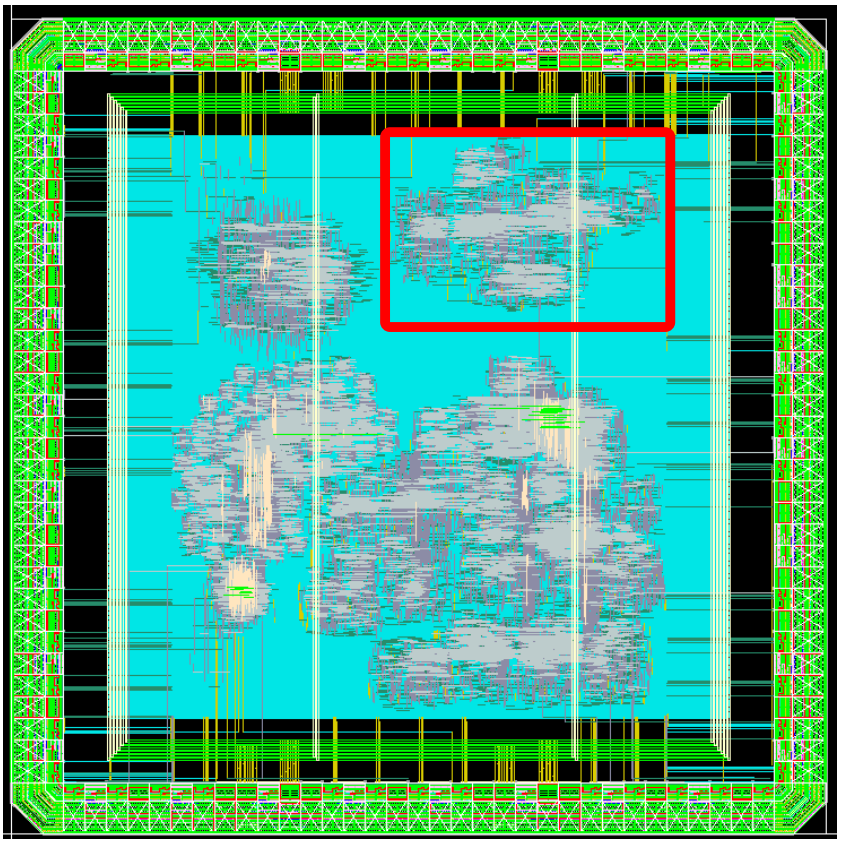


Figure 3.14 Layout ASIC. In red the block for SHA-256

can be seen, other blocks have been integrated in the same ASIC, however this dissertation is focused in the SHA-256 integration (marked with a red line).

The chip has a pad ring which includes 68 pads. Their distribution in the chip can be seen in Figure 3.15, in which the green color are the ones related with the SHA-256 implementation. The ASIC has 36 pads for inputs, 14 pads for outputs, 8 pads for VDD/VSS core and 10 pads for VDD/VSS ring pad. Output digital pads are selected with a driving strength of 2mA, whereas input digital pads are selected without pull option. The size of the ASIC layout is $1890\mu m$x$1890\mu m$.

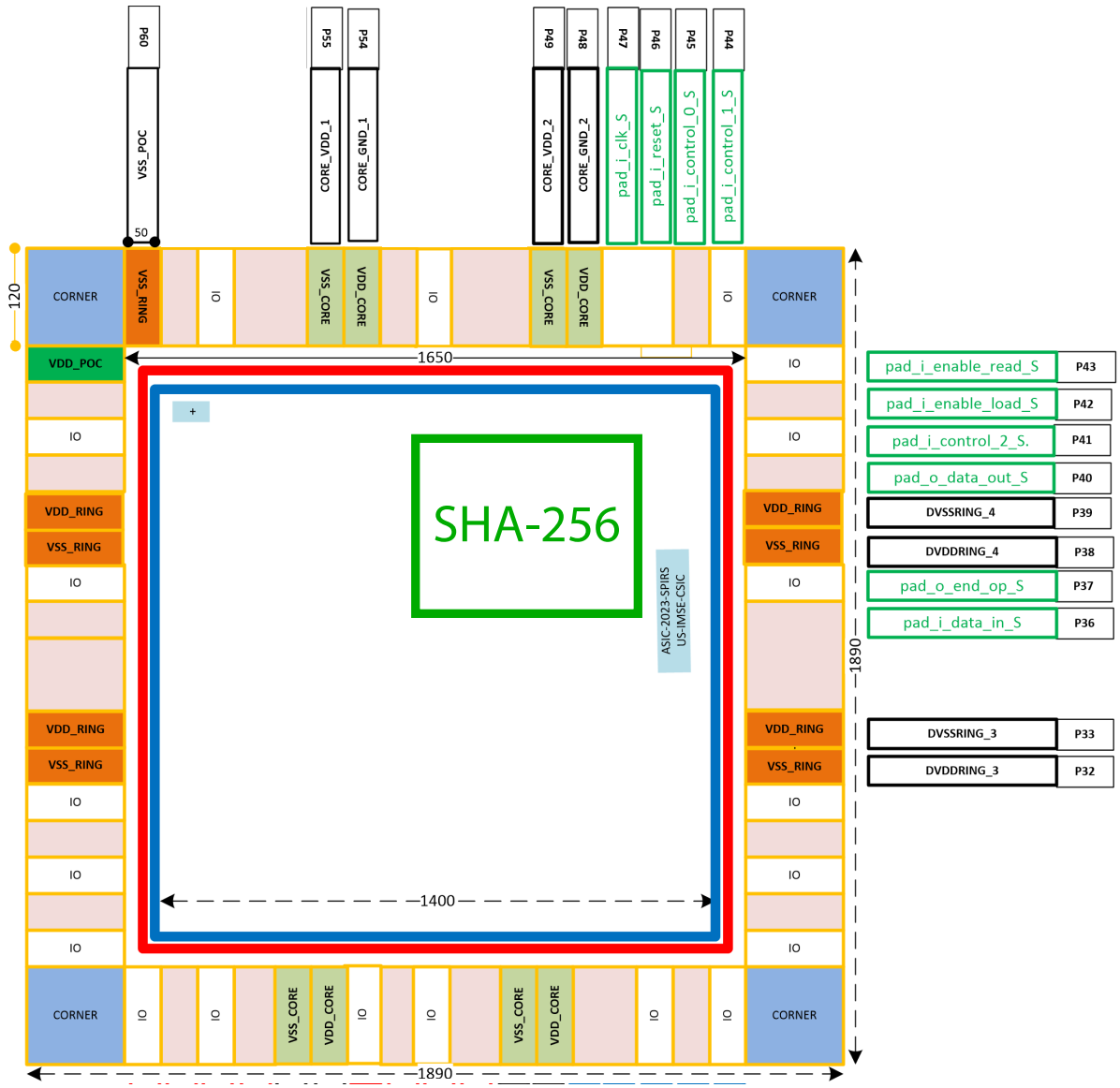


Figure 3.15 Distribution of the blocks and pads in the ASIC focusing in the SHA-256 block. In black, pads for supply voltages and ground

3.4 SHA-3 family

3.4.1 Introduction

Among the hash algorithms above mentioned, SHA-3 family (that includes SHA-3 and SHAKE algorithms) is considered more secure than SHA-2 for the same hash length (see the table with indicators of security strengths for the above properties in [81]). Additionally, compared to SHA-2, SHA-3 family has relatively fewer implementation costs and is much faster (in hardware implementations). The SHA-3 family is noted as PQC primitive since it is used internally by PQC algorithms (e.g. NTRU or CRYSTALS-Kyber, NIST-PQC-finalist) [103].

SHA-2 and SHA-1 are based on $Merkle-Damgråd$ (MD) construction scheme, whereas SHA-3 family possesses an architecture completely different. It was selected as the winner of the NIST hash function competition in 2012, succeeding the earlier SHA-2 family of hash functions [104]. Particularly, the Keccak algorithm was the winner of this NIST hash function competition for SHA-3. Keccak is based on a principle known as sponge construction, which relies on a random permutation. This technique enables the input (or "absorption" in sponge terms) of data of any size, and the output (or "squeezing") of data of any size, while working as a pseudorandom function with respect to all previous inputs. Several different digest bit lengths hash of 224, 256, 384, and 512 bits are available in SHA-3 family.

During the SHA-3 competition, several FPGA implementations were proposed. A detailed overview was published in the final SHA-3 report [104]. Further optimizations of lightweight architectures for SHA-3 implementations were proposed in [105] and [106]. Other works focus on high-speed FPGA implementations such as [107], [108], and [109]. Other approaches look for a competitive result in terms of efficiency with a good balance between throughput and area in [110] and [88]. More recently, a hardware design and implementation of the SHA-3 algorithm on FPGA is presented in [111].

This section aims to contribute and expand the work already presented in [112] in the following ways:

- The main goal is to present an efficient design of the Keccak function used in the SHA-3 hash function standard compatible with SHA-3 and SHAKE. The efficiency is corroborated by an optimal trade-off between the time processing of messages and the required area for FPGA implementation.
- A study and comparison of state-of-the-art FPGA implementations is carried out, achieving that the proposed solution exceeds the existing literature in efficiency.

- An IP module encapsulation of the SHA-3 and SHAKE function is also included, where the user can combine different parameters to e.g. make the module faster with higher area consumption, or swap the different SHA-3 family versions.
- The drivers required to install and invoke the IP module in software environments are also provided. There is also a set of instructions and use cases.

3.4.2 Keccak Function Background

The NIST declared the Keccak hash function [113] as the latest Secure Hash Algorithm-3 (SHA-3) after a competition in 2012. SHA-3 falls under the specifications outlined in the FIPS 202 standard [80], which provides the guidelines and requirements for the implementation and use of the hash function. The implementation of the Keccak function, regulated by the mentioned standard, is based on a sponge construction featuring an absorb and squeeze mechanism. This construction enables the Keccak function to handle input data of arbitrary length and transform them into an output of a specified length. This transformation involves multiple stages that systematically process the input data by employing a sequence of operations, ultimately generating the desired hash value.

The choice of each Keccak instance determines the configuration of the sponge construction, through the so-called bitrate (r) and capacity (c), where $r+c=b$. In the FIPS 202 standard [80], $b=1600$, defining the Keccak-f[1600] as primitive function. Thus, each SHA-3 instance has its own Keccak instance and therefore different bitrate, capacity, and output size (d) as it is shown in Table 3.9. These parameters not only define the structure of the construction but also play a crucial role in determining the overall security strength of the hash function.

Table 3.9 SHA-3 family hash functions parameters. [80]

Instance	d	r	c	Keccak Instance
SHA3-224	224	1152	448	Keccak[1152,448]
SHA3-256	256	1088	512	Keccak[1088,512]
SHA3-384	384	832	768	Keccak[832,768]
SHA3-512	512	576	1024	Keccak[576,1024]
SHAKE-128	d [2]	1344	256	Keccak[1344,256]
SHAKE-256	d [2]	1088	512	Keccak[1088,512]

[2] It is not predefined

The stages into which the SHA-3 function is divided are the following:

1. *Padding*: As stated in the FIPS 202 standard, the padding rule applied is the so-called `pad10*1`. In this process, the input is concatenated with a string of the form $P = 1||0^j||1$, where j is the minimum number of zeros to complete an entire input block.

2. *Absorbing*: Once the message is padded, it is divided into blocks of equal size. Each block is then XORed with the corresponding part of the internal state of the hash function. Normally this operation is only performed on the bits of the bitrate (r). This internal state is defined by a matrix of 25 elements (5x5) of 64 bits (i.e., $A[x,y]$ where $0 \leq x \leq 4$ and $0 \leq y \leq 4$).

3. *Keccak rounds*: Once the XOR operation is completed, the rounds of the Keccak function begin. There are 24 rounds in total, in which the internal state is operated through the following five sub-stages:

   - Theta (θ): It XORs every bit in the state with the parities of two columns in the matrix, as shown in Equations 3.9, 3.10, and 3.11. The *ROT* operation means a rotation of the array elements.

   $$C[x]{=}A[x,0]\oplus A[x,1]\oplus A[x,2]\oplus A[x,3]\oplus A[x,4] \quad \text{for } 0 \leq x \leq 4 \tag{3.9}$$

   $$D[x] = C[x-1] \oplus ROT(C[x+1],1) \quad \text{for } 0 \leq x \leq 4 \tag{3.10}$$

   $$A'[x,y] = A[x,y] \oplus D[x] \quad \text{for } 0 \leq x \leq 4 \text{ and } 0 \leq y \leq 4 \tag{3.11}$$

   - Rho (ρ) - Pi (π): It performs bit rotations on each matrix element depending on the value of a predefined matrix, known as offset matrix ($O\rho$) [113] as shown in Equation 3.12.

   $$A'[y,(2x+3y)] = ROT(A[x,y], O\rho[x,y]) \quad \text{for } 0 \leq x \leq 4 \text{ and } 0 \leq y \leq 4 \tag{3.12}$$

- Chi (χ): It performs an XOR operation of each bit with a non-linear function of two other bits in its row, as shown in Equation 3.13.

$$A'[x,y] = (A[x,y] \oplus !A[x+1,y]) \,\&\, A[x+2,y] \quad \text{for } 0 \leq x \leq 4 \text{ and } 0 \leq y \leq 4 \tag{3.13}$$

- Iota (ι): It modifies one element of the matrix state in a manner that depends on a predefined array depending on the Keccak round [80], known as *RC*, as shown in Equation 3.14.

$$A[0,0] = A[0,0] \oplus RC(\text{round}) \tag{3.14}$$

4. *Squeezing*: Once the Keccak rounds have finished, there are two possibilities, return to the absorbing stage with a new message block or, if there is no other block to digest, the result of the hash function will be the d LSBs of this last operation.

Keccak's mode of operation must be taken into account when designing and implementing the function in hardware.

3.4.3 Keccak Core Design

3.4.3.1 Basic version

The hardware design of the Keccak-f[1600] function aims to be functional with the least amount of resources needed for the implementation. The basic hardware implementation presented in this thesis is shown in Figure 3.16, where the Keccak[576,1024] core has been used to implement the SHA3-512 instance. On the one hand, the input stage is formed by the number of XOR operations according to the bitrate, which is in this case 576 bits for SHA3-512. This

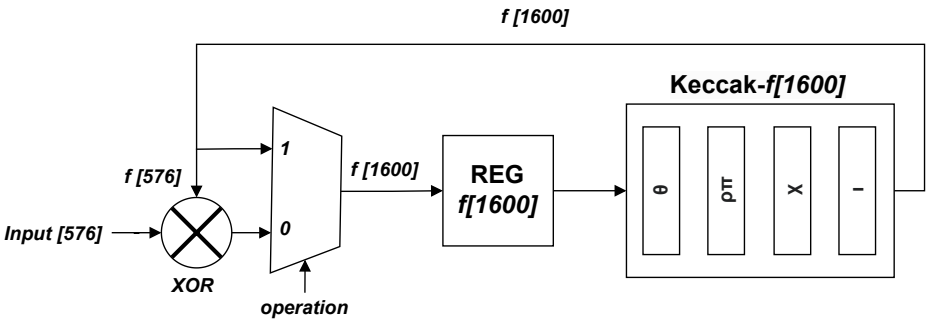


Figure 3.16 Basic version of the Keccak core (SHA3-512 example).

strategy, mentioned in [111], allows to reduce the number of resources used since the input block size must be concatenated with zeros to reach the 1600 bits of the Keccak input. On the other hand, the Keccak core design strategy has followed the architecture first presented in [114] that is also used in other works such as [107], [108] or [115].

The input stage also contains a multiplexer that, under the control of an `operation` signal, chooses either to store in a register the input data operated on by the XOR gate or to store the output data of the Keccak-f[1600] (containing all the Keccak functions described above) for a given round. This storage process is performed in a 1600-bit FF register. The resulting data is read directly from this register after all rounds have been completed. The implementation has been conceived as a parameterized design, where changing the SHA-3 instance only requires small changes to the design, such as the number of XOR gates at the input stage which are related to the bitrate of the function.

3.4.3.2 Optimized version

Based on the initial strategy, two optimized versions of the operation emerge to improve the performance of the original Keccak core. On the one hand, one of the proposed schemes is shown in Figure 3.17, where a reduction of the initial number of cycles (24) is achieved by concatenating *STAGES* times the Keccak function. Thus, the number of *STAGES* becomes part of the set of parameters that can be adjusted in this proposed scheme, taking into account that this parameter must be a divisor of the number of cycles of the Keccak operation (24).

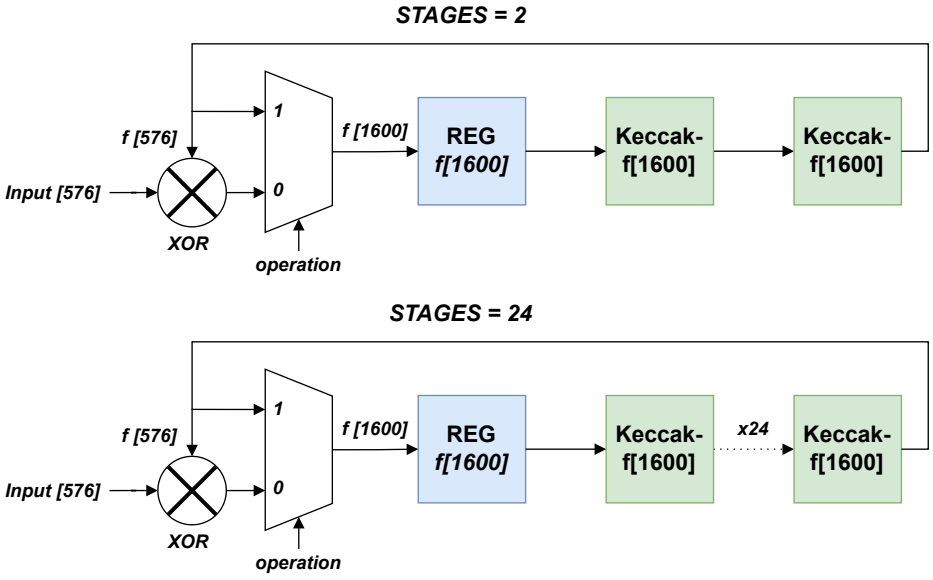


Figure 3.17 First optimized version of the Keccak core (SHA3-512 example)

For example, if $STAGES = 2$, it will take 12 cycles to complete the function, whereas if $STAGES = 24$, it will take only one cycle to complete the whole function. This strategy leads to an increase in resources and a decrease in the frequency of operation as the number of stages increases, which will need to be evaluated later. This scheme is partly based on the schemes presented in [107] and [88], without considering intermediate registers.

On the other hand, the second strategy, which is much more similar to [107] and [88], shown in Figure 3.18, aims to increase the number of blocks operated per clock cycle. In contrast to this dissertation, previous work did not provide a high tunability. For this purpose, a new parameter called $STAGES\_REG$ is added which, when enabled, allows the instantiation of intermediate registers between Keccak-f[1600], as shown in Figure 3.18. Contrary to the previous strategy, this does not reduce the number of cycles that each block requires but increases the number of blocks that can be operated per cycle, thus keeping the operating frequency constant without a significant increase in resources.

3.4.4 IP Module Integration

The SHA-3 function design presented in this dissertation has been encapsulated into an IP Module as shown in Figure 3.19. For that, the Keccak core was encapsulated together with its own Keccak control logic in the so-called SHA-3 core. The Keccak control logic is responsible for controlling the round counter as well as the input stage of the Keccak core once it is externally activated for starting the operation, while the control logic of this IP is responsible for the management of all the individual elements that comprise the encapsulation.

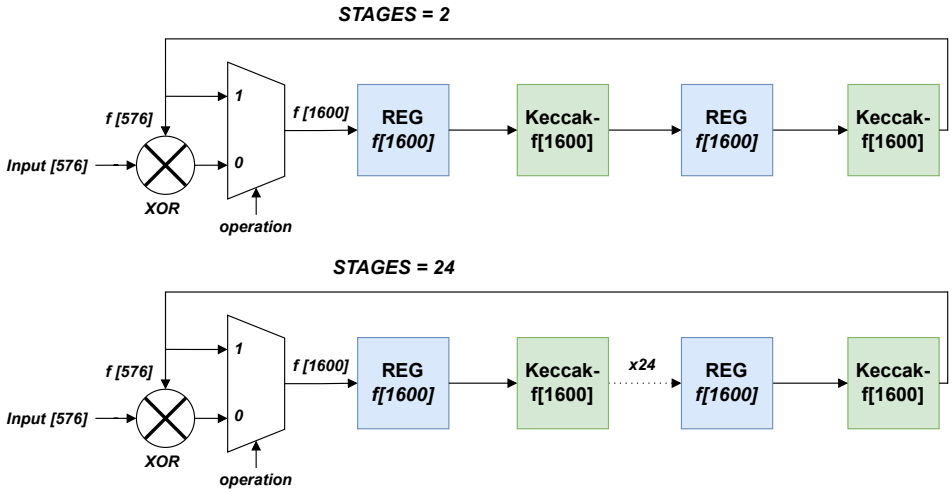


Figure 3.18 Second optimized version of the Keccak core (SHA3-512 example)

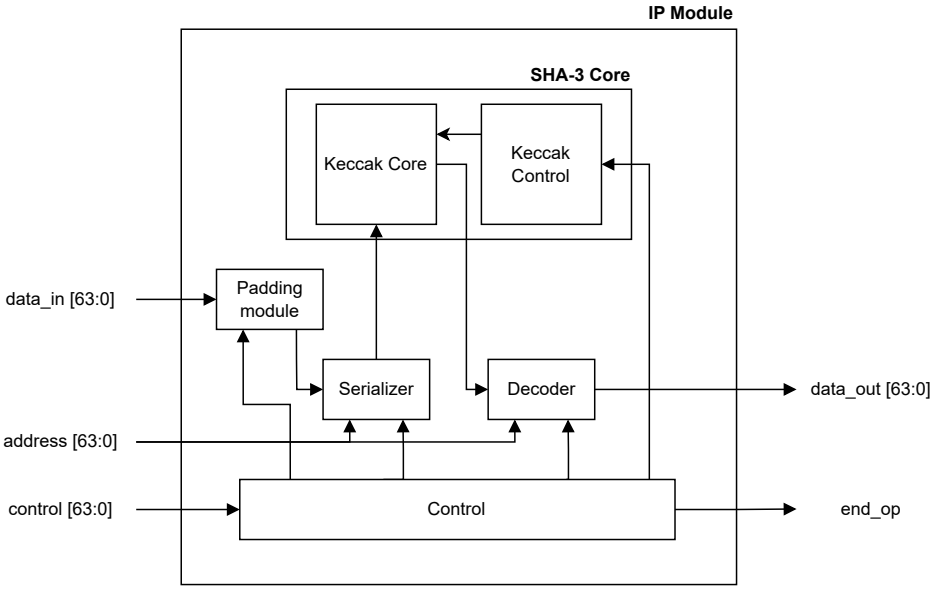


Figure 3.19 SHA-3 IP Module encapsulation

This control logic is based on a Finite State Machine with three states: LOAD, START, and READ. Each one of these states is externally controlled by the `control` signal. In the first of these states, the loading process is carried out on the serializer. Since the interface chosen to interconnect the IP Module has been AXI4-Lite with a bus size of 64 bits, it is necessary an interface that exchanges data serially/parallel and vice-versa. This is achieved by the serializer and decoder included in the design. Once all the data have been stored, the `control` signal can indicate that operation can begin, at which point the control module is set to the START state, and the SHA-3 core is activated. When the operation finishes, the `end_op` signal is activated and, depending on whether there is more data to operate, it can return to the LOAD state or, if it is the last, go to the READ state. Since the SHA-3 specifications do not specify a maximum length of the string to be processed, it is impossible to predict a previous string length in order to store it. For this reason, it can also be indicated by the `control` signal that the next block to be operated will require padding, carried out by the padding module.

The design of the IP module includes the user interface shown in Figure 3.20. As can be seen, it is possible to select the module according to the SHA-3 instance desired by the end user: SHA3-224, SHA3-256, SHA3-386 or SHA3-512. It is also possible to choose one of the two types of optimized versions as well as the amount of stages to be implemented. This IP module can be found in the repository of this dissertation [116] as `sha3_xl_1_0`.

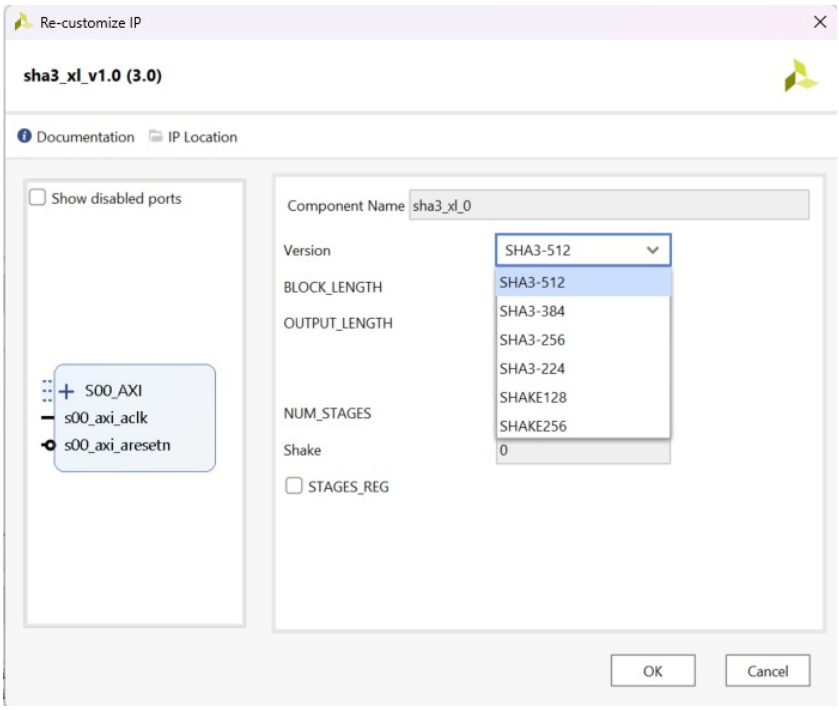


Figure 3.20 User interface of the SHA-3 IP Module

3.4.5 Embedded System Design

This section is completed with the inclusion of the SHA-3 IP module in an embedded system. For this integration, the PYNQ-Z2 [98] development board was used, which is based on the Xilinx Zynq-7000 SoC, containing on the one hand, an ARM as PS and, on the other, a Xilinx Artix-7 as PL. Due to the high tunability of the module, the evaluation with respect to the software has been carried out using all SHA-3 instances, as Figure 3.21 shows. This case is only considered for the tests performed. In a real application, it would be sufficient to instantiate the SHA-3 module that the designer considers appropriate. As shown in Figure 3.21, the connection of the IP modules to the processor has been done through the AXI Interconnect module provided by Xilinx.

The SHA-3 drivers' implementation was carried out in C and adjusted for the PYNQ environment by utilizing the PYNQ C-API that is available in [99]. This C-API provides an extensive collection of C routines that eases the loading of bitstreams and communication with hardware blocks on the PL of the Zynq device via memory-mapped and shared memory mechanisms. These C routines allow the communication between the processor and SHA-3 IP module. All the drivers and libraries necessary for the use of the SHA-3 IP module in software environment are included in the repository of this dissertation.

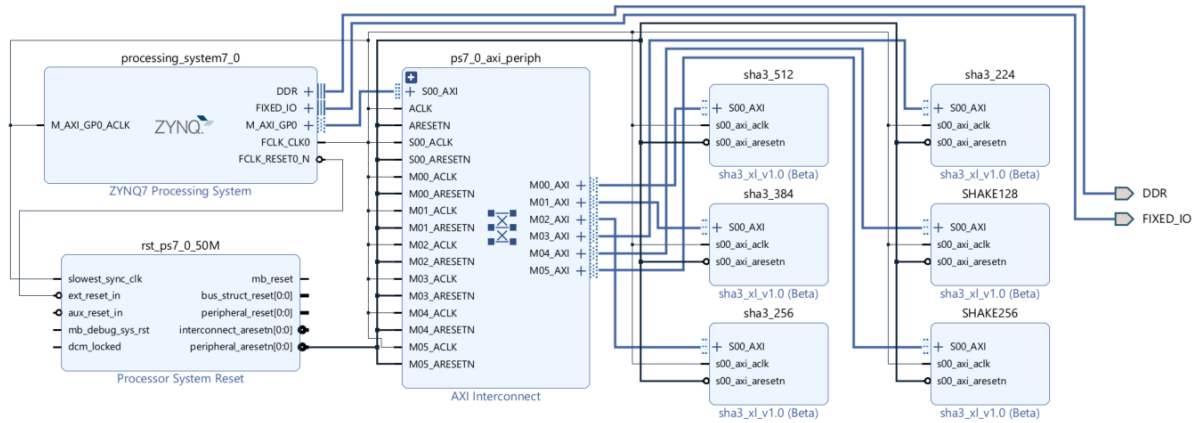


Figure 3.21 Block Diagram of the SHA-3 IP Module integration in a embedded system

3.4.6 Results

3.4.6.1 Performance at HW level

To evaluate performance at the hardware level, two metrics have been used that have been already defined in Section 3.2. One is called *throughput*, whose equation is given in Equation 3.7. And the other one is the *efficiency*, shown in Equation 3.8. For the case of *throughput*, there is a new parameter mentioned in [88] which symbolizes the number of messages that can be digested at the same time, N_{msg}. This parameter multiplies the value of previous *throughput* definition.

A Virtex-7 device has been used for the implementation of the designs to perform the evaluation. This is due to the majority of implementations reported in the state-of-the-art use this device to implement the SHA-3 function. However, the design is technologically independent, so it could have been synthesized and implemented on any other platform. The comparison between different strategies (i.e., number of `STAGES` and the use or not of intermediate registers, `STAGES_REG`) of the SHA-3 core (see Figure 3.19) for the SHA3-512 implementation is shown in Figure 3.22. Apart from the number of slices used, the two performance parameters mentioned above are included: *throughput* and *efficiency*, represented in the figure as `Thr` and `Eff` respectively. The presence or absence of intermediate registers is shown as 1 or 2 together with the parameter evaluated. Thus, Slices 1 and Slices 2 represent the number of slices used without and with intermediate registers, respectively.

As can be seen, the number of slices used increases with the number of `STAGES`. The higher the number of Keccak-f[1600] blocks, the more resources are required. However, the strategy that includes intermediate registers does not use more slices than the strategy that does not. This is because these new registers are instantiated in the slices used by the Keccak core logic.

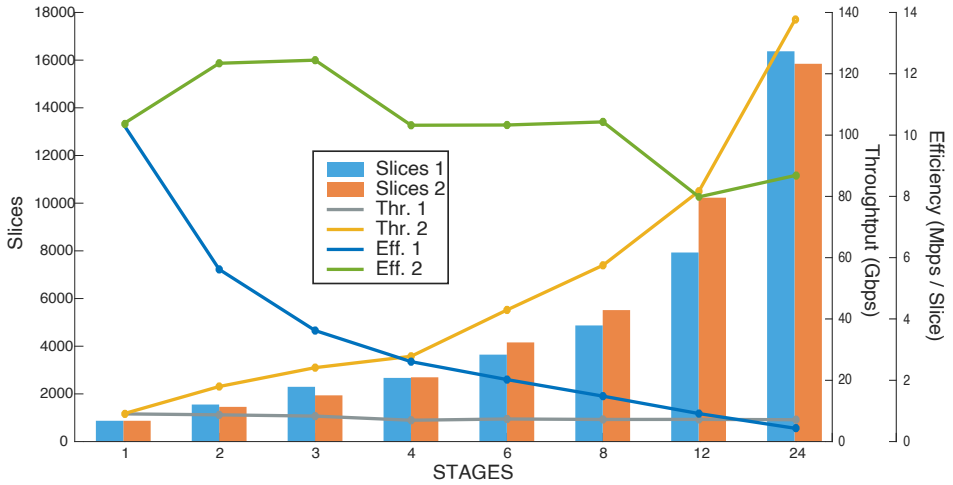


Figure 3.22 Comparison between different strategies of the SHA-3 core for the SHA3-512 implementation.

In terms of *throughput*, it can be seen that by maintaining the ability to process different blocks of messages at the same time, i.e. including the instantiation of intermediate registers, the throughput increases considerably over the other strategy. Finally, in terms of *efficiency*, it decreases considerably as the number of `STAGES` increases for a strategy without intermediate registers, while it decreases slightly for a strategy that includes them. The maximum of this efficiency is precisely in the latter strategy for a number of `STAGES` of 2 and 3.

The comparison between different implementations in the state-of-the-art has been done by exploiting the tunability of the module as much as possible, i.e. comparing between different instances of SHA-3 as well as the different design parameters: `STAGES` and `STAGES_REG`. Table 3.10 shows the comparison of the most competitive state-of-the-art implementations for different SHA-3 instances and different platforms. Since the configuration with $STAGES = 2$ and $STAGES\_REG = 1$ is one of the best performing, it was chosen for comparison. It can be seen, on the one hand, that the implementation carried out in this dissertation improves all the referenced implementations. On the other hand, it is shown how different instances of SHA-3 give completely different results. When the output size is reduced, the input block increases (e.g., from SHA3-512 to SHA3-256), which increases the number of slices occupied, thus increasing the throughput of the design.

In terms of integration, the first step was to integrate the SHA-3 core into an IP module, whose process has already been mentioned. Carrying out this integration, it increases the number of Look-Up Tables (LUTs) and FFs by about 600 and 900 respectively, resulting in an increase of about 500 slices. This increase in resources is independent of the SHA-3

Table 3.10 Comparison of the implementation of the SHA-3 core for different state-of-the-art works.

Ref.	Platform	SHA-3 Instance	Slices	Throughput (Gbps)	Efficiency (Mbps/Slice)
[88]	Virtex-6	SHA3-512	1406	16.51	11.47
[107]	Virtex-6	SHA3-512	2296	18.78	8.17
[110]	Virtex-5	SHA3-512	1163	7.80	6.06
This dissertation	**Virtex-7**	**SHA3-512**	**1456**	**17.97**	**12.34**
[109]	Virtex-7	SHA3-256	1618	20.08	12.90
[108]	Virtex-7	SHA3-256	1418	14.30	11.97
This dissertation	**Virtex-7**	**SHA3-256**	**1566**	**29.25**	**18.68**
[111]	Zynq-7000	SHA3-224	1424	12.8	8.98
This dissertation	**Zynq-7000**	**SHA3-224**	**1577**	**30.00**	**19.02**

instance, whose difference has already been considered in the core. The resource increase of the embedded system interconnection is approximately 700 LUTs and 800 FFs.

3.4.6.2 Performance at SW level

Among the set of functions that have been included in the software repository [116], the first one allows to evaluate the SHA-3 function implemented in hardware with respect to the NIST bit and byte tests [117][118]. This function allows to evaluate both type of tests for all available message lengths being compared in time with the SHA-3 software function found in the repository available in [119]. A double functional verification is included that compares the results obtained by the SHA-3 IP Module implementation with the same function in software and with the results included in the NIST tests. In the work repository can be found as `sha3_XXX_test` being XXX the implemented instance. The compilation process to obtain each instance only requires one parameter which is passed directly in the compiler call: `SHA3_224`, `SHA3_256`, `SHA3_384` or `SHA3_512`. This can be easily replicate using the `makefile` that accompanies the repository of this dissertation. Using the software provided in [119], the results of hardware acceleration versus software are shown in Table 3.11. These data have been obtained as the arithmetic mean of all NIST tests for all instances and considering a message whose length is either one or more than one input blocks. The results are quite promising, showing a 6 to 7 times speed-up of HW over SW execution time for the single block processing case, and 10 to 20 times faster for the multi-block case depending on the type of SHA-3 instance being used. These results could be improved by reducing the communication time between the software and the IP module where most of the processing time is spent.

Table 3.11 Acceleration of the SHA-3 family HW vs SHA-3 family SW [119] implementations.

SHA-3 Instance	*Message Length*	
	< Block Size	> Block Size
SHA3-224	x6.23	x10.89
SHA3-256	x6.41	x11.71
SHA3-384	x6.77	x14.54
SHA3-512	x7.33	x19.93
SHAKE128	x2.80	x8.18
SHAKE256	x3.07	x9.64

The actual functionality of the hash function is included in `sha3_XXX_demo` for SHA-3, and `shake_XXX_demo` for SHAKE. This function allows performing hash functions for hexadecimal strings (`-m`) or plain ASCII text strings (`-t`) provided through the command line, as well as introducing these data through a file as hexadecimal strings (`-mf`) or plain ASCII text strings (`-tf`). In this case, there is also an additional check on the software, which can be commented on or removed in a final version of the implementation. Results of some execution examples are shown in Figure 3.23, in which it is possible to observe how different calling always return an improvement in the execution time.

```
root@pynq:/home/xilinx/SHA3_xl# ./sha3_256_spirs -m abcd

        -- Results of the Execution --

 Msg. Length: 128 (bits) 16 (bytes)             Acceleration: 5.89             HW Pass: YES

 Hash Result:   0f1108bfb4ddb5cd6a8b05ad6dbc8244f0b0ef94cf77475a60a7bc952058425b


root@pynq:/home/xilinx/SHA3_xl# ./sha3_256_spirs -mf input_data.txt

        -- Results of the Execution --

 Msg. Length: 128 (bits) 16 (bytes)             Acceleration: 6.93             HW Pass: YES

 Hash Result:   0f1108bfb4ddb5cd6a8b05ad6dbc8244f0b0ef94cf77475a60a7bc952058425b


root@pynq:/home/xilinx/SHA3_xl# ./sha3_256_spirs -t abcd

        -- Results of the Execution --

 Msg. Length: 256 (bits) 32 (bytes)             Acceleration: 5.57             HW Pass: YES

 Hash Result:   6f6f129471590d2c91804c812b5750cd44cbdfb7238541c451e1ea2bc0193177


root@pynq:/home/xilinx/SHA3_xl# ./sha3_256_spirs -tf input_data.txt

        -- Results of the Execution --

 Msg. Length: 256 (bits) 32 (bytes)             Acceleration: 5.34             HW Pass: YES

 Hash Result:   6f6f129471590d2c91804c812b5750cd44cbdfb7238541c451e1ea2bc0193177
```

Figure 3.23 Results of the execution of the input chain "abcd" as hexadecimal, ASCII text and using input file

3.5 Conclusions

In conclusion, this dissertation underscores the pivotal role of hash functions in the realm of cybersecurity, with particular emphasis on the widespread use of the SHA-2 function in contemporary contexts. Notably, the recent introduction of SHA-3 marks a significant advancement in this domain.

The primary objective of this research endeavor has been to integrate these hash functions into a hardware RoT. To this end, the exploration started with the in-depth analysis of SHA-2 as the inaugural candidate. The outcome culminated in the establishment of a comprehensive open repository, providing a valuable resource for designers seeking to incorporate this function into their projects. Through a series of refinements and modifications, SHA-2 was further enhanced and subsequently implemented in an ASIC following a full digital flow implementation. Simultaneously, the SHA-3 hash function underwent a parallel journey of exploration and implementation, leading to its integration into a FPGA and subsequent release in an open repository.

Furthermore, it is worth noting that all relevant standards pertaining to both hash functions have been meticulously implemented and made openly available. The results of these implementations are highly encouraging, demonstrating that the incorporation of these functions need not entail an exorbitant allocation of resources. This finding holds significant implications for the broader cybersecurity landscape, affirming the feasibility and efficiency of their deployment.

In summation, this dissertation not only illuminates the critical importance of hash functions in cybersecurity but also charts a tangible path towards their integration within the RoT framework. The successful implementations of SHA-2 and SHA-3, alongside the provision of open repositories and adherence to standards, collectively signify a substantial leap forward in fortifying digital security measures. This dissertation lays a robust foundation for future endeavors in advancing the security landscape of digital systems.

Chapter 4

Post-Quantum Cryptography

4.1 Introduction

The security of most digital infrastructures relies on Public Key Cryptography (PKC), which enables secure communications between entities without sharing any pre-established secret. PKC provides i) protected channel establishment (key establishment) and ii) authentication of digital information (including authentication of individuals involved in a communication protocol through the application of digital signatures). The strength of current PKC techniques is based on the computation complexity of two mathematical problems: the factorization of large numbers and the computation of discrete logarithms. However, although these problems are complex for current state-of-the-art systems with high amounts of resources and computational power, they can be solved in a reasonable amount of time using quantum computers. As a consequence, the security of cryptographic protocols applied in our everyday life will be compromised in the near future. For instance, Shor's algorithm [31] highlights the capability of quantum computers in efficiently factoring integers. This exposes a weakness in the widely used RSA algorithm, which relies on the complexity of factoring a large biprime number. Additionally, Shor's algorithm can also solve the Discrete Logarithm Problem (DLP) in polynomial time. The DLP serves as the foundation for other cryptographic methods, such as Diffie-Hellman (DH), the Digital Signature Algorithm (DSA), and Elliptic Curve Cryptography (ECC).

The scientific community has developed PQC to deal with this threat. The roadmap of the EU Cybersecurity Strategy identifies PQC as a key enabling technology, as reported by ENISA in [120]. Moreover, the NIST started a PQC competition in 2016 to identify cryptographic algorithms able to withstand quantum computer attacks. During the development of this dissertation, the algorithms based on Key Encapsulation Mechanism (KEM) selected in the third round were published in 2020 and underwent modifications until 2022 [103]. Among those

selected were NTRU. However, it was not until July 2022 that NIST published its final decision on the algorithms to be standardized, with CRYSTALS-Kyber being selected [33]. Besides, it was not until August 2023 that the first draft of the standard based on CRYSTALS-Kyber was published as KEM, under the name ML-KEM (FIPS-203) [121]. Due to recent publications in this regard by NIST and the fact that these publications are not completely closed, it was decided not to include any information in this dissertation regarding CRYSTALS-Kyber or ML-KEM. Although it is a very clear and necessary way forward.

Proposals submitted to the NIST PQC contest included software implementations. However, the design of hardware-efficient solutions is an open challenge for the electronics engineering community. Recent studies present the use of hybrid hardware/software (HW/SW) co-design methodologies to combine flexibility and efficiency when implementing PQC-based algorithms [122, 123]. This chapter focuses on the hardware execution of the previously mentioned PQC algorithm, NTRU. It advances the field by presenting the acceleration of all parameter sets of this NTRU version, building upon the work related to NTRU acceleration [124]. Additionally, it incorporates various mechanisms to counteract SCA, thereby preventing the extraction of information through Simple Power Analysis (SPA), and safeguarding against timing attacks.

4.2 NTRU

4.2.1 Introduction

Among lattice-based PQC cryptosystems, the public key encryption scheme NTRU was consolidated as a reference since it offers certain advantages over other cryptosystems with the same security level, namely that it is faster and works with smaller key sizes [125]. NTRU's security is based on the Shortest Vector Problem (SVP), which is a difficult problem in lattice reduction. Until now, no algorithm has been developed to solve this problem in polynomial time. The NTRU public key cryptosystem was standardized by the Institute of Electrical and Electronics Engineers (IEEE) in 2008 as IEEE Std 1363.1-2008 [126] and by the American National Standards Institute (ANSI) in 2010 as ANSI Std X9.98-2010 [127]. The original version of NTRU has been progressively improved to be resilient against different types of attacks. NTRUEncrypt [128] and NTRU-HRSS-KEM [129] submissions in Round 1 of the NIST PQC standardization contest were merged in Round 2 to give rise to a new NTRU submission (NTRU [130]), which reached Round 3 [131]. The first list of PQ algorithms to be standardized has been recently announced [33], in which NTRU is not among those selected. However, NTRU-based algorithms are a fundamental pillar in PQC with a solid background. Advances to provide efficient NTRU implementations on embedded systems are

an open challenge, especially in certain scenarios where strict restrictions make the adoption of other PQC finalists with higher levels of complexity unfeasible.

There are a wide variety of implementations of NTRU encryption and decryption schemes on several platforms, such as software on embedded microcontrollers [132], FPGAs [133], and even an experimental study of hardware-dedicated building blocks for VLSI integrations [134]. In most cases, these implementations must be included in IoT environments where area and time constraints are very limited. This is because the evolution of programmable devices has progressed towards SoCs, which combine one or more embedded processor cores and PL, encouraging the development of hybrid implementations following HW/SW co-design methodologies. The idea behind HW/SW implementations is to exploit the flexibility coming from software with the efficiency of hardware realizations for the most demanded timing operations. In NTRU cryptosystem, the critical operation is the multiplication in the nth-degree truncated polynomial ring; thus, the efforts of the scientific community have been focused on its acceleration through hardware implementations. Most of the studies reported in the literature follow two well-distinguished methodologies. On one hand, some studies are based on a High-Level Synthesis (HLS) methodology, starting from a high-level description of the NTRU algorithm [135]. On the other hand, some employ a methodology based on a Register-Transfer Level (RTL) description for critical operations [136–138]. The main advantage of the first strategy is the reduction in development time due to the use of automatic synthesis tools that do not require a solid background of designers in hardware description languages. However, the second strategy generally offers the most efficient implementations in terms of timing, power consumption, and area, using ad hoc hardware realizations for critical operations.

This section presents an extension of the work presented in [124] based on the implementation of the NTRU third round version, following a HW/SW co-design methodology. This includes the evaluation of each NTRU parameter set as well as the decryption evaluation. Additionally, regarding side-channel attacks, this section completes the evaluation in terms of security started in [124] for the rest of parameter sets. That is crucial because despite the efficiency of post-quantum cryptography, implementations of lattice-based cryptography secure against side-channel attacks remain an open issue, as [139] and [140] point. The security implementation aspect of lattice-based cryptography has yet to be explored in this regard. Some advances related with timing attacks are included in [141]. Therefore, this dissertation (unlike others works presented on NTRU) tries to include a solution that can involve a mitigation against timing side-channel attacks. Moreover, this implementation follows a flexible design that can mitigate timing-based side-channel attacks and also make it suitable for the area or temporal limitations that are common in IoT environments, establishing a compromise between area and performance. The main contributions of this section are as follows:

- The specific solution for the NTRU polynomial multiplier which allows accelerate the multiplication, the encryption and decryption process without generating any security breaches related to timing attacks in the system.
- The design of a highly configurable IP module, whose configuration enables the possibility to easily implement the different parameter sets as well as different arithmetic units responsible for performing the multiplication operation.
- The design of an interconnection scheme based in an AXI4-Stream protocol that optimizes the bandwidth of communication infrastructures between the processor core and the IP.
- The evaluation of i) the resources used for each particular solution and comparison with other implementations in the literature; and ii) the acceleration factors achieved with the proposed implementations versus the software implementation of the NTRU third round version [142]. And the best trade-off between a high acceleration factor and a moderated value of area occupation.
- Following open policies, all software and hardware developments are available in a public repository to ease its re-use and corroborate the results [143].

4.2.2 The NTRU Encryption Scheme

4.2.2.1 Mathematical Background

The basis of the actual KEM used in the actual version of NTRU [131] is inherited from the first round of the NIST PQC contest, in which the NTRU-HRSS-KEM version was submitted. This version was in turn based on a variant of the Fujisaki–Okamoto transformation [144]. During the second round of the contest, two of the cryptography algorithms presented, NTRU-HRSS-KEM and NTRUEncrypt are combined to merge the NTRU presented in the third round, NTRU Round 3. This latest version of the algorithm included variants which can be summarized in the transformation of the original NTRU-HRSS (Hülsing, Rijnveld, Schanck, and Schwabe) [129] and NTRU-HPS (Hoffstein, Pipher, and Silverman) [145]. The difference between these variants is essentially the sample space of some of the polynomial that are involved in the cryptosystem.

The cryptography scheme of NTRU is based on polynomial convolution rings or quotient rings, which are a particular algebraic structure where polynomial operations are performed [145]. The characteristics of each quotient ring are set depending on the NTRU Round 3 security level, which is modulated by the sets of parameters defined in [131]. The polynomial degree is configured by the parameter n, and the modulus of the polynomial coefficients is set

by the parameter q as it can be observed in Table 4.1. In this scope, any polynomial whose coefficients are integers is denoted as $\mathbb{Z}[x]$.

Table 4.1 NTRU parameter set [131]

Parameter set	n	q
`ntruhps2048509`	509	2048
`ntruhps2048677`	677	2048
`ntruhps2048821`	821	4096
`ntruhrss2048701`	701	8192

The most important parts of the KEM is the encryption and the decryption process due to the fact that they involve the majority amount of time in the encapsulation and decapsulation, respectively. Focusing on the encryption and decryption schemes in NTRU Round 3 [131], it is required defining the quotient rings described by Equations 4.1, 4.2, and 4.3.

$$R/q = \mathbb{Z}[x] \,/\, (q, \Phi_1 \Phi_n) \tag{4.1}$$

$$S/q = \mathbb{Z}[x] \,/\, (q, \Phi_n) \tag{4.2}$$

$$S/3 = \mathbb{Z}[x] \,/\, (3, \Phi_n) \tag{4.3}$$

where

- Φ_1 is the polynomial $(x-1)$;
- Φ_n is the polynomial $(x^n-1)/(x-1) = x^{n-1} + x^{n-2} + \ldots + 1$;
- $(q, \Phi_1 \Phi_n)$ represents the operation modulus q for the coefficients and modulus $\Phi_1 \Phi_n$ for the polynomial degree;
- (q, Φ_n) represents the operation modulus q for the coefficients and modulus Φ_n for the polynomial degree;
- $(3, \Phi_n)$ represents the operation modulus 3 for the coefficients and modulus Φ_n for the polynomial degree.

Thus, Equations 4.1, 4.2, and 4.3 can be expressed as shown in Equations 4.4, 4.5, and 4.6.

$$R/q = \mathbb{Z}[x] \mod (q, x^n - 1) \tag{4.4}$$

$$S/q = \mathbb{Z}[x] \mod \left(q, \frac{x^n - 1}{x - 1}\right) \tag{4.5}$$

$$S/3 = \mathbb{Z}[x] \mod \left(3, \frac{x^n - 1}{x - 1}\right) \tag{4.6}$$

Therefore, R/q is a polynomial with a degree of $n-1$ at most, with coefficients of $\{-q/2, -q/2+1, \ldots, q/2-1\}$, whereas S/q is a polynomial with a degree of $n-2$ at most, with coefficients of $\{-q/2, -q/2+1, \ldots, q/2-1\}$, and $S/3$ is a polynomial with a degree of $n-2$ at most, with coefficients of $\{-1, 0, 1\}$, which constitutes the so-called *ternary* polynomial, $\mathcal{T}$. In this version of the NTRU, it is necessary to define a subset of ternary polynomials, represented by $\mathcal{T}(t)$, that contain exactly $t/2$ elements equal to $+1$ and other $t/2$ elements equal to -1.

The NTRU Round 3 encryption scheme initially requires two polynomials: the first one, which emerges from the public key $h(x) \in S/q$, and the blinding polynomial, which is a ternary polynomial, $r(x) \in \mathcal{T}$. Unlike previous versions of the NTRU cryptosystem, the number of nonzero coefficients of $r(x)$ is not known in this version. These two polynomials are multiplied according to the convolution product described in Equation 4.7:

$$e(x) = r(x) \times h(x) \mod (q, x^n - 1) \tag{4.7}$$

where $e(x) \in R/q$. On the other hand, the message, which is the other polynomial required, is transformed into a ternary polynomial with $q/16 - 1$ ones and $q/16 - 1$ minus-ones, $m(x) \in \mathcal{T}(q/8 - 2)$, to increase the message obfuscation in the encryption process. Unlike NTRUEncrypt, the padding mechanism of the message presented in [129] disappears. However, a change in the message representation from $S/3$ to R/q is required for NTRU Round 3. In NTRU-HPS, the new message $m'(x) \in S/3$ is equivalent to the message $m(x)$. In NTRU-HRSS, the message $m'(x)$ is obtained after a complex process described in [131], so-called *Lift* operation. The operation in which the encrypted message $c(x) \in R/q$ is obtained is described in Equation 4.8,

$$c(x) = e(x) + m'(x) \mod (q, x^n - 1) \tag{4.8}$$

In the case of the decryption scheme of the NTRU Round 3 algorithm, several convolution products are carried out in which some polynomials are required. First, two polynomials that merge from the secret key of the algorithm, $f(x)$ and $f_p(x)$ are used. The first polynomial is a ternary polynomial, thus $f(x) \in \mathcal{T}$, while $f_p(x) \in S/3$, being the inverse of $f(x)$ in this modulus. Second, the inverse of $h(x)$ is also required, $h_q \in S/q$. And, finally, the ciphertext that as it was mentioned above, $c(x) \in R/q$. The decryption process starts with a multiplication described in Equation 4.9 between the ciphertext, $c(x)$ and one of the polynomial obtained from the secret key, $f(x)$:

$$a(x) = c(x) \times f(x) \mod (q, x^n - 1) \tag{4.9}$$

The original message is partially recovered using the result of the previous operation and the other polynomial also generated from the secret key as is shown in Equation 4.10.

$$m'(x) = a(x) \times f_p(x) \mod \left(3, \frac{x^n - 1}{x - 1}\right) \tag{4.10}$$

In order to recover completely the original message, the *Lift* operation, mentioned above, is used in the case of NTRU-HRSS, not so in the case of NTRU-HPS.

In this encryption operation, the multiplication operation required to calculate $e(x)$ consumes the highest percentage of time in relation to the total encryption time [125, 146]. This operation is also used during the decryption process for the calculation of $a(x)$ so that, the goal of this chapter is to provide a new implementation, suitable for constrained devices, that accelerates the algorithm execution at the expense of a low increase in resources.

4.2.3 Hardware Implementation of Polynomial Multiplication

The multiplication described in Equation 4.7 or 4.9 follows a cyclic convolution process that can be expressed by Equation 4.11 (for the case of Equation 4.7),

$$e_k = \sum_{i+j=k \bmod N} (h_j \cdot r_i) \mod q \tag{4.11}$$

where the polynomial degree is expressed as N; e_k represents the k-th coefficient of $e(x)$; h_j the j-th coefficient of $h(x)$; and r_i the i-th coefficient of $r(x)$. It is trivial to prove that this operation requires $N \cdot N$ scalar multiplications to conclude. The first effort to accelerate this operation at the hardware level was reported in [147]. The main changes introduced in this work were i) the replacement of the multiplication by adding coefficients of the public key, $h(x)$, to the temporal result for each nonzero element in the blinding polynomial, $r(x)$, now as a binary polynomial; and ii) the substitution of $r(x)$ by $r_1(x) + r_2(x)$ reducing the total operation cycles to $(d_1 + d_2) \cdot N$, being d_1 and d_2 the nonzero coefficients of $r_1(x)$ and $r_2(x)$, respectively. This was a direct and basic implementation of the scalar multiplication that achieved good results at the hardware level. However, the techniques were progressively refined to achieve improved implementations, as in the case of [148], where the scalar multiplication was replaced by addition or subtraction depending on whether the coefficient r_j was 1 or -1, respectively. Power reduction methods were used to provide a design specially adapted to security applications (RFIDs and sensor nodes) reducing the amount of resources and power needed in a physical implementation. On the other hand, due to the large number of zero

elements contained in the polynomial $r(x)$, another important improvement in the hardware implementation of the polynomial multiplication was the exploitation of this fact in the works presented in [149] and [150]. They only consider the nonzero coefficients of $r(x)$ polynomial to implement the convolution product. The first one exploits their locations, whereas the second identifies the degrees of nonzero terms in the $r(x)$ polynomial during the load process. Using this consideration, the operation described in Equation 4.11 can be expressed as shown in Algorithm 1. Therefore, the multiplication operation will require $(N \cdot nnz)$ scalar multiplications and $(N - nnz)$ clock cycles of null coefficients of $r(x)$, where N is the degree of the polynomial and nnz the number of nonzero elements (including 1 and -1) in the $r(x)$ polynomial. This enhancement significantly reduces the time required to complete the convolution operation.

Algorithm 1 Accelerating the polynomial multiplication using nonzero elements

```
for i = 0 : N − 1 do
    if r_i ≠ 0 then
        for k = 0 : N − 1 do
            j = mod(k − i, N)
            e_k = e_k + (h_j · r_i)
        end for
    end if
end for
```

A parallelization process of this convolution product can be carried out by adding different scalar multiplications per operation cycle, considerably reducing the overall operation time. With the consequent cost in terms of resources, the work in [151] presents a fully parallelized hardware in which N scalar multiplications are performed in each cycle using a LFSR structure. In [152] and [137], an improvement over [151] is proposed, where the multiplication operation is accelerated by analyzing when two, three, or four consecutive zeros are presented in the obfuscation polynomial $r(x)$. In [137] and [153], it was analyzed whether the total number of cycles or in other words, the total time required to complete the multiplication operation when it is accelerated considering consecutive elements can generate a security breach by using timing attacks that could allow the coefficients of the obfuscation polynomial $r(x)$ to be induced. On the other hand, the use of fully parallel structures to reduce the number of cycles implies a high cost in terms of resources, which constrained implementations cannot afford. To solve these drawbacks, a low-resource architecture for NTRUEncrypt based on a partial parallelization of the scalar multiplications that does not present any security breach against timing attacks is proposed in [138]. In this case, the operation could be accelerated using only the $2 \cdot dr$ nonzero coefficients of the polynomial $r(x)$ (dr is the number of coefficients that are 1 and -1), i.e., $r(x) \in \mathcal{T}(dr)$, resulting in a number of operations equal to $(N \cdot 2dr) + (N - 2dr)$. More

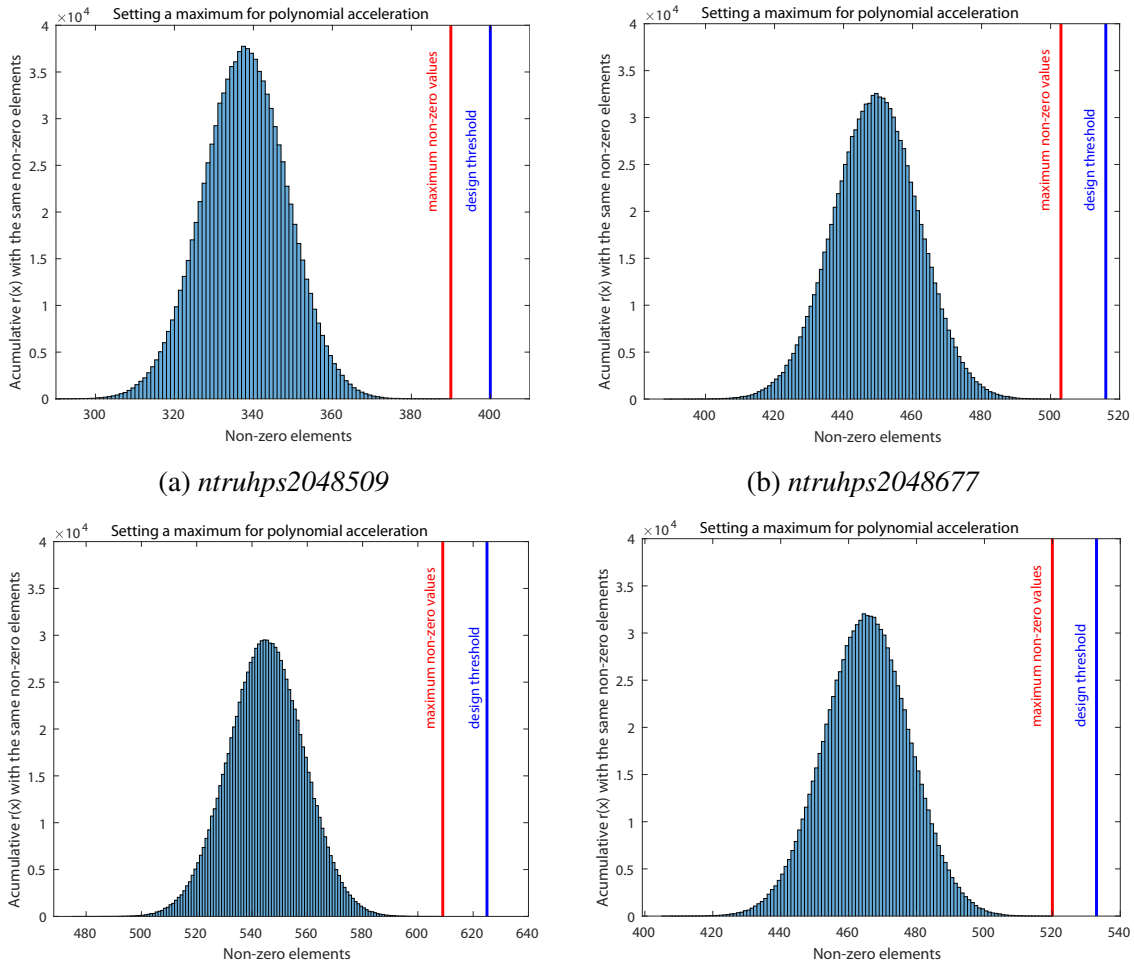
recently, the work presented in [154] described a full hardware implementation of the three multiplications required in NTRU Round 3. However, such implementation is not performed on the software structure of the cryptosystem presented in [131], using a large amount of resources to complete the operation in parallel, making its implementation on resource-constrained devices unfeasible.

4.2.4 Robust Acceleration Against Timing Attacks

In the version of NTRU submitted to the third round of the NIST PQC contest, as described above, $r(x) \in \mathcal{T}$, i.e., the number of nonzero coefficients, is not fixed, being impossible to predict the number of ones, minus-ones, and zeros before generating the polynomial. Using a pseudocode such as the one described in Algorithm 1 would cause a security breach if timing attacks were performed (by disclosing the information of the number of nonzero coefficients contained in $r(x)$). The goal of this proposal is to achieve some degree of acceleration of the $N \cdot N$ clock cycles of the convolution product without revealing any sensitive information of the polynomial $r(x)$.

Since it is not possible to know in advance the number of nonzero elements that the polynomial $r(x)$ will have, the strategy to accelerate this process consists of the evaluation of a significant number of $r(x)$ possible generations in order to establish an upper limit. For the set of parameters shown above in Table 4.1, Figure 4.1 shows the distributions of the number of nonzero elements when the polynomial $r(x)$ is generated 10^6 times for each parameter set. The red line represents the maximum number of nonzero elements obtained, or in other words, this case has a probability of 0.0001% (1 over 10^6) that happens. Establishing this threshold as a limit to perform the multiplication operation would not generate any security breaches since there is no temporal distinction between different generations of $r(x)$. The hardware multiplier, therefore, will have to operate for at least that number of cycles so as not to raise suspicions about the number of nonzero elements contained in the polynomial $r(x)$.

In a real implementation, for a specific use case, margins must be established from this threshold of nonzero coefficients. In this particular case for the specific set of parameters the confidence margin is set by the blue line in Figure 4.1. Fitting the number of nonzero coefficients to a normal distribution, the design threshold corresponds to a certain probability shown in Table 4.2. Thus, it can be considered that is practically impossible to obtain a polynomial $r(x)$ with more nonzero coefficients than the threshold or confident limit established. In the case that this happens, establishing some design mechanism that allows for regeneration of the polynomial $r(x)$ at the software level would solve this remote problem. Hereinafter, this design threshold value will be referred to as max_{coef}. This limit of coefficients will include a large number of nonzero coefficients and a small number of zero coefficients.



(a) *ntruhps2048509*

(b) *ntruhps2048677*

(c) *ntruhps2048821*

(d) *ntruhrss2048701*

Figure 4.1 Distribution of nonzero elements in different $r(x)$ generations. The red line represents the maximum obtained, while the blue line defines a confidence threshold for the implementation.

Table 4.2 NTRU confident limits

Parameter set	N	Max. non-zero value	Confident Limit (CL)	Probability of CL (%)
`ntruhps2048509`	509	390	400	$1.2270 \cdot 10^{-7}$
`ntruhps2048677`	677	503	516	$1.0382 \cdot 10^{-7}$
`ntruhps2048821`	821	609	625	$1.6783 \cdot 10^{-7}$
`ntruhrss2048701`	701	520	533	$1.0557 \cdot 10^{-7}$

These changes can be applied to the operation of the multiplier, so the pseudocode presented in Algorithm 1 can be upgraded to the pseudocode of Algorithm 2. In this case, the total number of cycles $total_{cycles}$ required to complete the operation will be $total_{cycles} = N * max_{coef} + (N - max_{coef})$. Setting a fixed number of clock cycles avoids the possible leakage of timing information that would allow the number of nonzero coefficients of $r(x)$ used in the multiplication to be known. The inclusion of this countermeasure does not entail a significant reduction in acceleration that can be achieved by taking advantage of the fact that $r(x)$ is ternary. To complete the operation, it is necessary to know the number of nonzero coefficients of the polynomial $r(x)$, nnz. This number must be calculated for each polynomial in each execution, which can be carried out and stored internally at a stage prior to the multiplication phase. Therefore, with this information, the number of zeros to be computed as if they were nonzero elements, i.e., those below the threshold max_{coef}, corresponds to the variable defined as $number_{zeros-max}$ or nzm. Therefore, in order to avoid any leakage of timing information, the acceleration produced by the elimination of the cycles corresponding to the null coefficients of $r(x)$ will only take place when the threshold determined by max_{coef} is exceeded. Thus, the pseudocode is designed so that the operation is only performed if one of the following conditions occurs: 1) the maximum number of zeros nzm has not yet been reached; and 2) the limit has been exceeded and the coefficient of $r(x)$ is not zero. This analysis will lead to an implementation of eight different solution two per parameter set: one for $max_{coef} = N$ (in which it is not performed any algorithm acceleration) and other one for $max_{coef} = CL$.

Algorithm 2 Accelerating the polynomial multiplication considering nonzero elements and avoiding timing attacks

$nzm = max_{coef} - nnz$

for $i = 0 : N-1$ **do**
  if $r_i = 0$ **then**
    $nz = nz + 1$
  end if
  if $nz \leq nzm$ **or** ($r_i \neq 0$ **and** $nz > nzm$) **then**
    for $k = 0 : N-1$ **do**
      $j = mod(k-i, N)$
      $e_k = e_k + (h_j \cdot r_i)$
    end for
  end if
end for

4.2.5 IP Module Design and Integration

4.2.5.1 Design of the Arithmetic Unit

The design goal of this hardware-level multiplier is to be fully compatible with the reference version of NTRU submitted to the third round of the NIST PQC contest [131], so that software routines can be interchanged with hardware routines in a much easier and more efficient way. This implies that some particularities in both arithmetic and data types at the software level must be taken into account for software/hardware implementation. In the NTRU Round 3 scheme [131], the coefficients of the polynomials $r(x)$ and $h(x)$ are computed as modulus 2 and modulus 2048, respectively. The efficient AU presented in [136], [137] and [138] has been slightly modified. The new architecture uses a logic gate AND instead of a multiplexer. The AU is controlled by the r_i coefficient, whose operation is summarized in Table 4.3 and shown in Figure 4.2.

Table 4.3 AU operation in function of r_i.

r_i	Operation
00	$e_{out_k} = e_{in_k}$
01	$e_{out_k} = e_{in_k} + h_j$
11	$e_{out_k} = e_{in_k} - h_j$

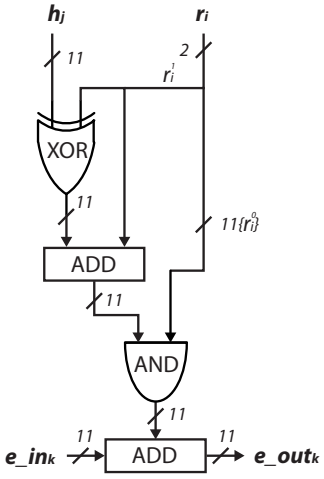


Figure 4.2 Block diagram of the AU designed in this dissertation.

4.2.5.2 Core Design

The architecture of the polynomial multiplier used in the NTRU Round 3 encryption scheme is described in this section. The module was developed using the RTL-based design flow provided by Xilinx Vivado tools, and the Verilog HDL was used for hardware description. A simplified block diagram that contains the main functional blocks necessary for the hardware implementation of the polynomial multiplication operation is shown in Figure 4.3.

Multiplier operation is coordinated by the Control Unit block. It manages all operations in different phases: *Load* coefficients, *Operate*, and *Read* result. It generates the indices i, j, and k, which are used in each phase as memory addresses in the Memory block. This component contains the coefficients of the input polynomials, $r(x)$ and $h(x)$, stored in the *Load* phase. It also stores the partial results during the *Operate* phase, as well as the resulting polynomial, $e(x)$, that will be provided by the module in the *Read* phase. The memories included in this block were implemented as dual-port memories using the BRAMs usually available in many programmable devices. During the *Operate* phase, the Control Unit obeys the operation described by the pseudocode in Algorithm 2; while the AU carries out either addition or subtraction over h_j, depending on the value of r_i, updating the partial results e_k. The number of total clock cycles required to complete the operation, CC_{op}, including the clock cycles for coefficient loading, CC_{load}, and reading, CC_{read} (both require one clock cycle per coefficient), and the number of total clock cycles for the multiplication, CC_{mult}, is described by Equation 4.12. The number of coefficients, N, and the threshold fixed to avoid timing attacks,

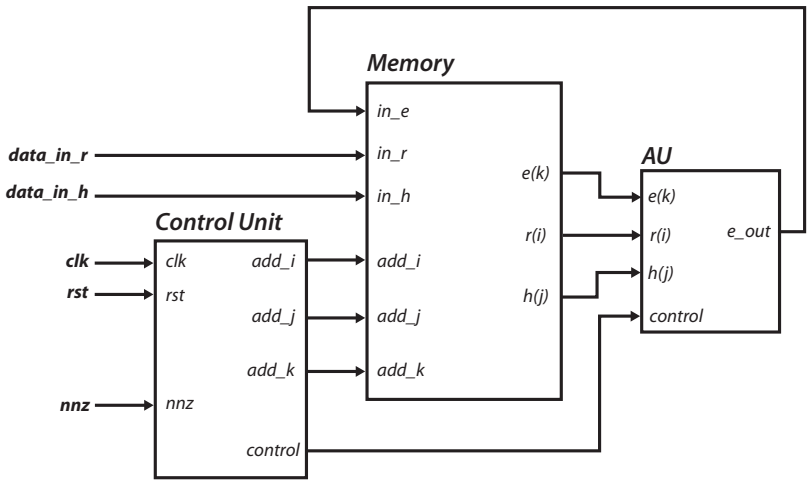


Figure 4.3 Simplified block diagram of the hardware polynomial multiplier architecture.

max_{coef}, are the main factors determining the overall system runtime.

$$\begin{aligned} CC_{op} &= CC_{load} + CC_{mult} + CC_{read} \\ &= 2N + N \cdot max_{coef} + (N - max_{coef}) \\ &= N \cdot (max_{coef} + 3) - max_{coef} \\ &\approx N \cdot max_{coef} \end{aligned} \tag{4.12}$$

4.2.5.3 Parallelizing the Multiplication Process

In the scheme described above, the number of clock cycles required for the system to complete the multiplication operation can be further reduced. A possible solution in this proposed scheme is acceleration using AUs operating in parallel. The strategy is the inclusion of M AUs, where M is the parallelization degree of the system so that the multiplication module can operate on M coefficients per cycle. To introduce the ability to parallelize the system through the M parameter, the pseudocode presented in Algorithm 2 is redefined in Algorithm 3, in which the the total clock cycles due to the operation of the AUs is reduced by a factor M. This reduction will depend on the degree of the polynomial and the number of AUs instantiated in parallel.

Algorithm 3 Parallelizing the polynomial multiplication considering nonzero elements and avoiding timing attacks

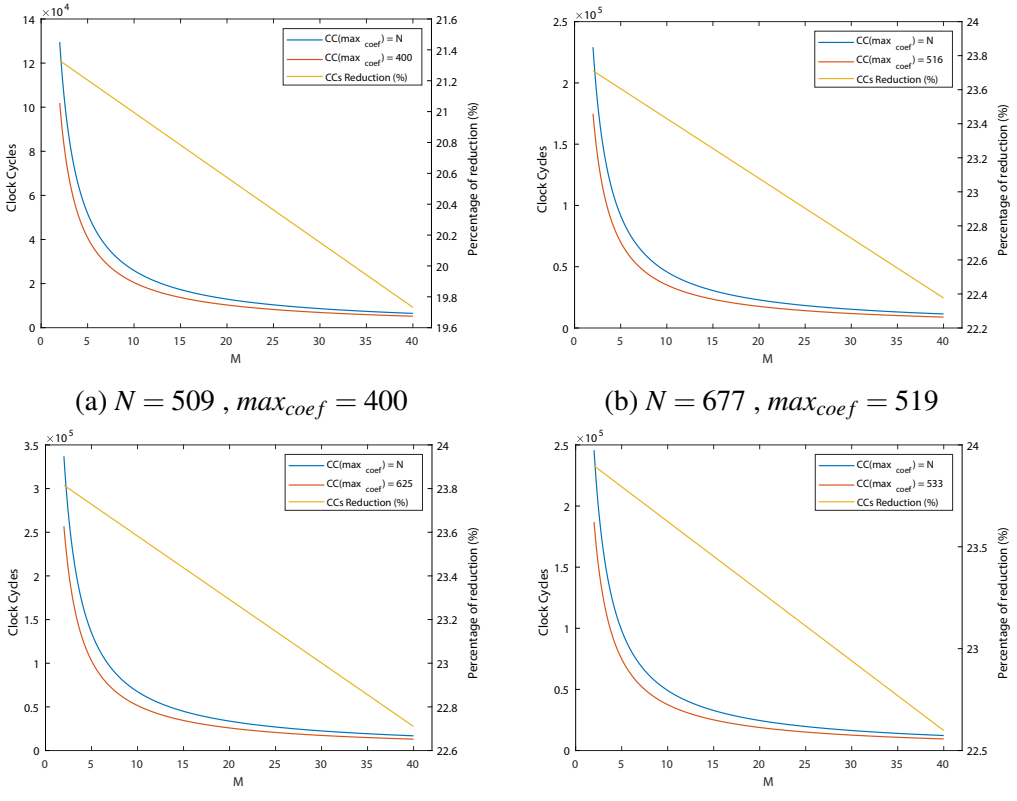
$M \leftarrow$ Parallelization parameter
$nzm = max_{coef} - number_{non-zero}$

for $i = 0 : N-1$ **do**
  if $r_i = 0$ **then**
    $nz = nz + 1$
  end if
  if $nz \leq nzm$ **or** ($r_i \neq 0$ **and** $nz > nzm$) **then**
    for $k = 0 : M : N-1$ **do**
      $j1 = mod(k-i, N)$
      $e_k = e_k + (h_{j1} \cdot r_i)$
      $j2 = mod(k-i+1, N)$
      $e_{k+1} = e_{k+1} + (h_{j2} \cdot r_i)$
      $\vdots$
      $jM = mod(k-i+M-1, N)$
      $e_{k+M-1} = e_{k+M-1} + (h_{j1} \cdot r_i)$
    end for
  end if
end for

Therefore, the number of total clock cycles of the parallel operation, CC^*_{op}, including both the loading and reading of the coefficients as well as the multiplication operation, CC^*_{mult}, is reduced according to Equation 4.13. In this case, the estimation of the total number of total clock cycles is directly related to the M parameter. Large values of M involve a considerable reduction in the number of clock cycles used for coefficient multiplication. Thus, the times required for writing and reading the coefficients to and from the module should not be underestimated when evaluating the total time for the module operation.

$$\begin{aligned} CC^*_{op} &= CC_{load} + CC^*_{mult} + CC_{read} \\ &= 2N + \left\lceil \frac{N}{M} \right\rceil \cdot max_{coef} + (N - max_{coef}) \\ &= 3N + max_{coef} \cdot \left(\left\lceil \frac{N}{M} \right\rceil - 1 \right) \end{aligned} \tag{4.13}$$

In Figure 4.4, a theoretical calculation to compare both strategies in terms of total clock cycles required to perform the multiplication operation in the function of M is shown. For its representation, a continuous function of the number of clock cycles as a function of M has been



(a) $N = 509$, $max_{coef} = 400$

(b) $N = 677$, $max_{coef} = 519$

(c) $N = 821$, $max_{coef} = 625$

(d) $N = 701$, $max_{coef} = 533$

Figure 4.4 Comparison in terms of clock cycles between strategies versus M.

considered. Essentially, it is the representation of the expression presented in Equation 4.13. It can be seen how the number of cycles decays very fast until about $M = 10$, where the clock cycles needed for loading and reading the coefficients begin to be significant. Furthermore, the yellow line shows the difference of clock cycles between the two strategies of max_{coef} in terms of percentage. The method detailed in Section 4.2.4 allows for a reduction around the 20% of the operation time in parallelized solutions.

Since the acceleration strategy is based on the parallelization of the operation, at the hardware level, it is necessary to replicate the AUs M times. This also involves a replication of the Memory block, which increases the size in order to provide both the addresses and coefficients to complete the multiplication correctly. The implementation strategy followed for the AUs, bus sizes, and Memory block replication was detailed in [138]. The Control Unit must be also modified in order to generate the index addresses, taking into account that there is more than one coefficient operating in the same clock cycle. The block diagram of the hardware implementation considering acceleration is shown in Figure 4.5.

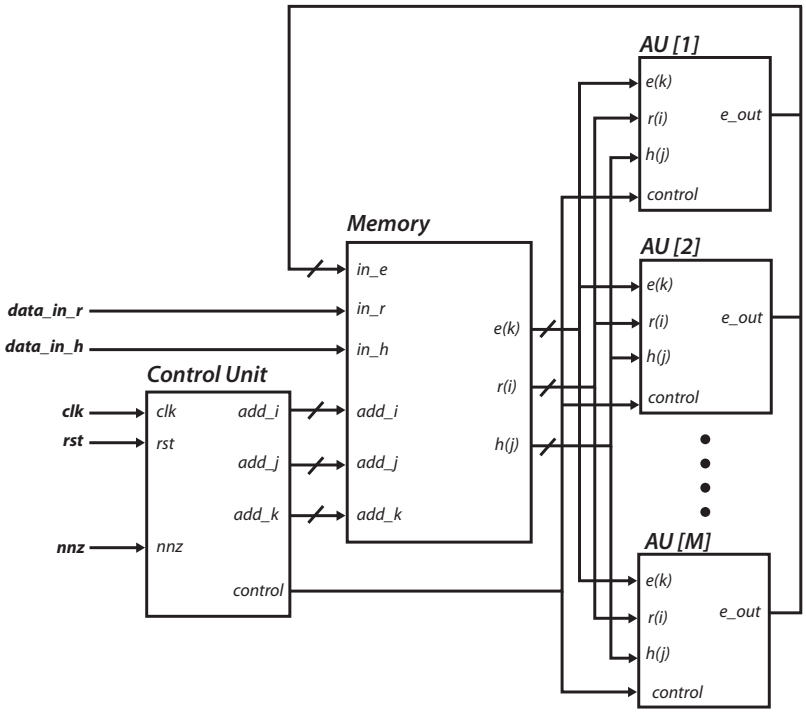


Figure 4.5 Block diagram of the hardware polynomial multiplier architecture considering parallelization.

4.2.5.4 Embedded System Integration

The hardware architecture detailed above must be interconnected with a PS to build an hybrid implementation. The IP module and the PS are connected using standard interconnection buses that facilitate design reusability. In this case, the cryptosystem defined at the software level and executed on a general-purpose processor sends the coefficients of the input polynomials; thus, this communication protocol also has to be designed following these considerations. In order to use the proposed multiplier on SoC solutions—for example, those that incorporate an ARM processor—the most suitable option is to use the AXI bus.

The AXI interconnection interface is implemented through AXI4-Stream interface. For a correct synchronization between the data sent by the processor and the IP module, it is necessary to instantiate First-In, First-Out (FIFO) structures both at the input and output. Figure 4.6 shows the final multiplication module, in which the FIFOs have been integrated into the blocks called *Data In* and *Data Out*. The memory addresses used to store the $r(x)$ and $h(x)$ coefficients in the *Load* phase and to read the $e(x)$ coefficients in the *Read* phase are internally provided by the Control Unit.

The use of standard communication protocols such as AXI4-Stream makes the module fully integrable with other devices. Apart from that, the design of the IP module is completely reusable, being able to change implementation parameters related to the cryptosystem, such

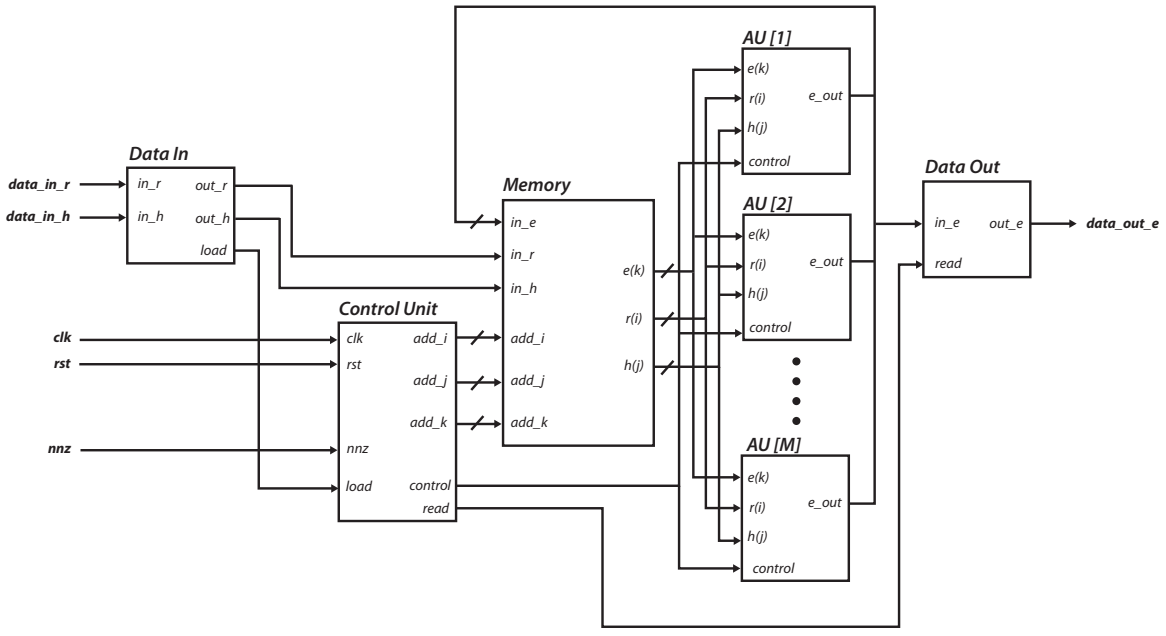


Figure 4.6 Block diagram of the hardware polynomial multiplier architecture, considering parallelization and including AXI4-Stream interconnection interfaces.

as the degree of the polynomial, N, as well as others that influence the timing performance, such as max_{coef} or M. Figure 4.7 shows the IP integrator window in which can be observed the variety of parameters the final user can modify. Therefore, first, it is fully functional on any of the parameter sets defined in the NTRU Round 3 version, and second, the implementation can be adapted to be more or less restrictive in terms of area and timing performance depending on the constraints imposed on the system to be implemented. The IP module is located in the NTRU repository [143] with the name of `ntru_ms2xs_8.0`.

At the hardware level, to complete the interconnection scheme between the PS and the IP module, shown in Figure 4.8, some extra modules are required: the Direct Memory Access (DMA), AXI Interconnect, and AXI SmartConnect blocks handle the exchange of information related to the polynomial coefficients. The IP module receives the coefficients of the input operands at the beginning of the operation and sends the coefficients of the result at the end of the operation from/to the DMA blocks, respectively. For the design and implementation of the IP module, the Vivado 2023.1 design tool was used. This facilitates both the design and its integration in embedded systems, making the design of a parametrizable IP module possible.

Specifically, this dissertation has used the development board PYNQ-Z2, which integrates the Xilinx Zynq-7000 SoC (XC7Z020-1CG400C). The device consists of a PS that operates at

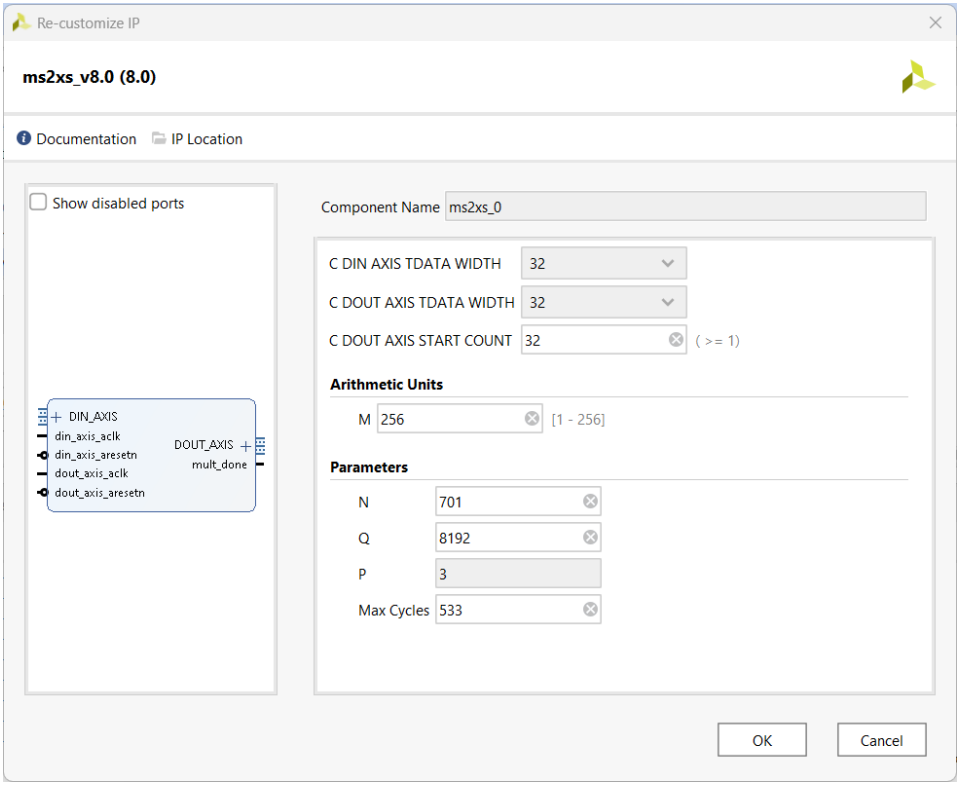


Figure 4.7 IP Integrator window

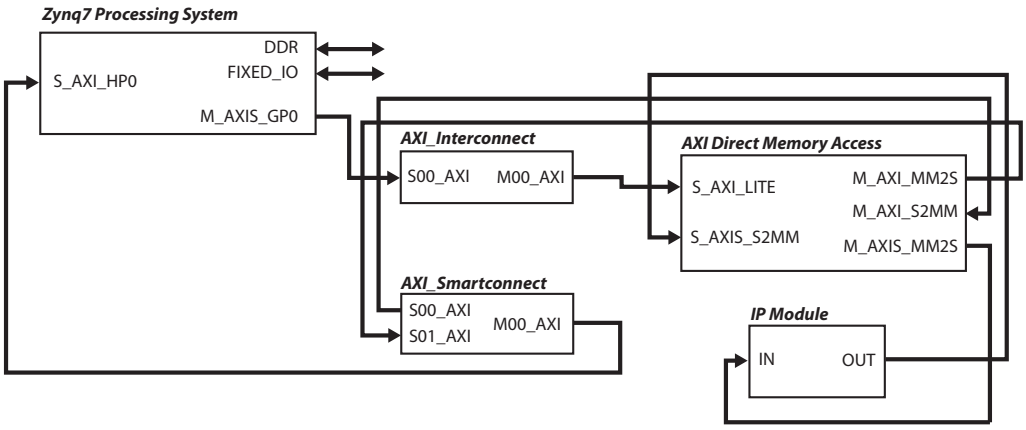


Figure 4.8 Block diagram of the complete embedded system and the necessary blocks to interconnect the IP module with the Zynq Processor.

650 MHz and includes a dual-core ARM-Cortex-A9, along with PL from the Xilinx Artix-7 FPGA family. The selected development board supports the PYNQ environment [98]. This describes a framework for Python, which operates on an embedded Linux operating system. The framework streamlines the process of connecting hardware modules with software components.

At the software level, the NTRU Round 3 version implementation used was the optimized one presented in [131]. This is a complete C implementation of the NTRU KEM scheme presented in the third round of the NIST contest. For this dissertation, a completely new set of tests was designed, directly providing results relative to the acceleration of the encryption and decryption scheme using dedicated hardware. The implementation of NTRU in C was adjusted for the PYNQ environment by utilizing C-API, which is made available in [99]. This C-API offers a comprehensive set of C routines that can be compiled to produce executable code. PYNQ C-API includes features that make it easier to load bitstreams and communicate with hardware blocks located on the PL of the Zynq device through memory-mapped and shared memory mechanisms. The utilization of these features not only simplifies the creation of software drivers required to manage the hardware multipliers, but also eases the programming of a series of tests to validate and characterize their operation.

The software repository is included in [143] in the compressed file `NTRU_3Round.rar`. In these files there are two versions of the NTRU hardware-accelerated version: `Test_XXX` and `Demo_XXX`, where `XXX` corresponds to the N parameter in each NTRU parameter set. The first one was developed to perform different types of tests in which it is possible to evaluate the NTRU parameters set adding the ability to select the parallelization degree, M or the max_{coef} parameter. For that, there is a set of bitstreams already generated that contain the NTRU IP module of the particular specifications defined in the section above. And, the second one is

a demonstration demo in which it is possible to generate the pair of keys, encapsulate and decapsulate using the HW module.

4.2.6 Results

In this Section, the multiplication module is evaluated in terms of resource consumption, and the acceleration factors in both multiplication and encryption processes are analyzed. Different versions of the multiplication module for NTRU Round 3 were implemented in this dissertation for multiple values of M and considering or not the limit established by max_{coef}.

4.2.6.1 Resource Consumption

Using the parametrizable IP Module, it is possible to modify the degree of the polynomial, N, and the number of AUs in parallel, M, as well as the maximum number of coefficients for acceleration, max_{coef} as it was shown above. For the evaluation, the complete NTRU parameter set was used. Thus, The IP module was implemented, varying the values of N, max_{coef} and M for each specific parameter set. Although it is possible to select any value, for the evaluation and in order to comprehensively cover the entire possible range of the parallelization coefficient, M, that generates the different parallel implementations, the first sixteen values and all powers of two starting from 32 up to 256, both inclusive, were used.

Appendix D.1 shows the tables of the use of LUTs, FFs, and BRAMs of the implementations as combination of max_{coef} and N already presented in Table 4.2. For the case of $max_{coef} = N$ there is an increase in execution time of around 10%, while the resource occupation is slightly lower with respect to the $max_{coef} = CL$. In terms of the parallelization coefficient, M, results show that as the value doubles, the resource consumption close to doubles in the case of LUTs and BRAMs, but not in the case of FFs, where the trend is simply upward. For the highest value of M, 256, while there is hardly any variation in occupancy relative to FFs, around 30% of LUTs and 90% of BRAMs are occupied in a Zynq-7000 platform. With the increase in M, the resources required for the implementation also increase, being even more critical to resource consumption. Summarizing the results, up to an index $M = 32$, occupancy is approximately below 10%, being even around 0.5%-2% in total for $M = 4$.

With respect to the embedded system integration, there is a minimum use of resources necessary to instantiate the DMAs and the communication infrastructure, adding a resource consumption of approximately 2400 LUTs, 3200 FFs, and 2 BRAMs. The percentage of occupation due to all these connections in the embedded system is independent of the size of the IP module and clearly notable for the small values of M.

Table 4.4 firstly shows the comparison in terms of area occupation and timing performance for `ntruhps2048509`, with respect to the work presented in [154], which is based as is this dissertation on the NTRU submitted to NIST PQC Round 3. The comparison between the two implementations shows that the one presented in this section has less resource occupation with the same timing performance. Moreover, this dissertation provides both a detailed analysis of the potential risks of timing attacks associated with this NTRU version and a balance between cost and performance, which is crucial in the context of IoT. Thus, our contribution in this regard is a strong and reliable version for IoT that requires minimal additional resources. In fact, 256 AUs were used for the comparison. However, this number can be reduced to any value that satisfies the occupancy constraints of a particular IoT device. Additionally, Table 4.4 includes several works in the literature, such as [137], [138], [152], and [153], which present implementations for the NTRU version standardized in IEEE-1363.1, different from the one used in this dissertation. The works presented in [137], [152], and [153] implement parallel structures to perform the NTRU multiplication, while the work presented in [138] shows a serial implementation that is very close to this dissertation.

Table 4.4 Resources and timing performance comparison between this dissertation for `ntruhps2048509`, a recent work of the latest NTRU version, and other works of the previous standard.

NTRU Version	Work	LUT	FF	#CC	Latency μs	#AU
Round-3	***This dissertation***	16707	2992	1018	12.90	256
	[154]	56218	21406	821	12.32	509
IEEE-1363.1	[137]	29194	19096	245	3.23	541
	[138]	603	90	7107	71.07	8
	[152]	30300	-	343	3.62	541
	[153]	38240	-	541	-	541

4.2.6.2 Analysis of Acceleration Factors

Another aspect that requires detailed analysis is the time reduction that the IP module achieves in the multiplication process as well as in the encryption and decryption of the NTRU Round 3 software version. That is, how much it allows the operation of the multiplication to speed up when comparing a full software version, executed on the embedded system processor, versus the use of the hardware IP module. The set of tables contained in Appendix D.2 shows the acceleration produced by the use of the IP module with both strategies of max_{coef} for each parameter set for the multiplication, and the complete encryption and decryption process,

respectively. For each value of M, 1000 tests were performed, with software and hardware mean times obtained. The values of both the multiplication and the encryption time in the software are the mean values of the whole tests performed.

Using this IP module in the NTRU Round 3 version as the value of M increases, the hardware operation time is reduced, also increasing the acceleration in the multiplication operation, and in the encryption and decryption scheme. This impact in the acceleration is slightly reduced in the case of decryption in which only one of the three multiplication performed is the one that is accelerated using the IP Module. For $M = 1$ of all implementations, accelerations around 5 are obtained in both multiplication and encryption, being slightly higher in the case of the multiplication. In the case of decryption in which only one multiplication is accelerated, the accelerations of the complete decryption process are over 1.3.

For the multiplication case in the set of tables, the maximum acceleration factors range between 70 and 120 depending on the selected set of parameters with the use of the maximum degree of parallelization. The results also reveal that the differences between both strategies ($max_{coef} = CL$ and $max_{coef} = N$) are more significant for intermediate M values. The strategy of $max_{coef} = CL$ presents certain temporal advantages, reducing around 10% of the time required with respect to the second one, $max_{coef} = N$. As is also observed, a limit in the acceleration is reached when M is increased (i.e., there are 128 different values between $M = 128$ and $M = 256$ in which the acceleration barely changes). This is mainly due to the fact that the operating time accelerates to such an extent that the times required to exchange coefficients to and from the IP Module are no longer insignificant. This behavior was already predicted earlier in Figure 4.4.

In the case of the encryption scheme, since the software time required for multiplication is equivalent to about 95% of the software time required for encryption, the time reductions follow the same trend. This can be verified where as in the case of multiplication—the central values of M are where the difference between the use of both values ($max_{coef} = CL$ and $max_{coef} = N$) is the greatest. On the other hand, as the degree of parallelization increases, a limit as in the case of multiplication is reached. In this case, the acceleration factor obtained range between 35 and 70. The difference with respect to multiplication is mainly due to the fact that in the encryption process of the NTRU Round-3 version, other software functions different from the polynomial multiplication are executed, which are necessary and should not be disregarded. No matter how much the multiplication function is accelerated, the execution of these operations requires time that cannot be avoided. This is particularly evident in the set of parameters `ntruhrss701` in which the operations around the encryption process involve much more time than the rest of parameters sets based on `hps`. In the case of the decryption scheme, the reduction of time carried out by the IP module in the multiplication is faded out by the amount of operations

performed in the decryption process. The increase in the parallelization degree gets a maximum acceleration of around 1.4 for any solution presented in this dissertation.

4.2.6.3 Optimizing Area and Acceleration

In constrained devices, it is especially important to analyze the resources required to implement a function. In other words, evaluating how the specific use of each resource is capable of separately speeding up the algorithm to a greater or lesser extent is fundamental for establishing certain design decisions regarding area optimization. Since in this section, results have been presented related to LUT, FF, and BRAM occupancy, as well as accelerations as a function of the parallelization index M, it is possible to study how this parameter M jointly affects the resources used and acceleration obtained. For this purpose, a new figure of merit, *Efficiency*, E, is defined for each type of resource as the quotient between the acceleration obtained and the resources (LUTs, FFs, and BRAMs) used for each M:

$$E_{LUT}(M) = \frac{Acc.(M)}{LUT(M)} \tag{4.14}$$

$$E_{FF}(M) = \frac{Acc.(M)}{FF(M)} \tag{4.15}$$

$$E_{BRAM}(M) = \frac{Acc.(M)}{BRAM(M)} \tag{4.16}$$

Optimization of this figure of merit occurs when few resources are used and high acceleration is obtained. This combination causes the efficiency to tend towards higher values. For simplicity in evaluating these results, only the resources used to implement the IP module are used. An analogy could easily be drawn in terms of occupancy with the embedded system. Thus, using the results presented in the set of tables contained in Appendix D.1 concerning the number of resources used to implement the IP module, as well as the data in Appendix D.2 concerning the acceleration of multiplication, it is possible to quantify the efficiency parameter defined above. Results are shown in the set of tables contained in Appendix D.3. The maximum *Efficiency* of each resource type and of each strategy is in bold taking as a reference the value only in the multiplication (marked in red).

Table 4.5 collects all the most efficiency implementations of Appendix D.3 using the multiplication acceleration as reference. It is shown the maximum efficiency in terms of LUTs, FFs and BRAMs of each configuration. All cases are delimited of a M value between 4 and 13, which are the medium-low part of the all generated M. Apart from that, the majority of cases share the same optimized M value between the strategy of $max_{coef} = CL$ and $max_{coef} = N$. For the LUTs case, the maximum optimization occurs between a M value of 4 and 8. For that

Table 4.5 Summary of the *M* selected for the maximum efficiency in terms of resource occupancy and timing performance of the IP module.

N	max_{coef}	Max. Eff.	M	LUTs	FFs	BRAM	Acc. (x)
509	400	***LUT***	**4**	285	119	4.5	20.67
		FF	**7**	437	124	7.5	30.28
		BRAM	**8**	584	205	4.5	32.81
	509	***LUT***	**6**	348	105	6.5	23.40
		FF	**7**	399	106	7.5	26.00
		BRAM	**8**	556	187	4.5	28.35
677	516	***LUT***	**8**	476	128	8.5	37.46
		FF	**10**	582	133	10.5	43.08
		BRAM	**11**	801	251	6.0	45.49
	677	***LUT***	**8**	451	108	8.5	31.23
		FF	**10**	558	113	10.5	36.40
		BRAM	**11**	773	231	6.0	38.64
821	625	***LUT***	**8**	510	129	8.5	40.06
		FF	**12**	713	133	12.5	51.89
		BRAM	**13**	981	287	7.0	54.25
	821	***LUT***	**8**	481	109	8.5	32.94
		FF	**12**	679	113	12.5	43.74
		BRAM	**13**	929	267	7.0	45.99
701	533	***LUT***	**6**	433	130	6.5	32.26
		FF	**10**	634	133	10.5	43.54
		BRAM	**11**	881	273	6.0	46.32
	701	***LUT***	**6**	403	110	6.5	25.52
		FF	**10**	605	113	10.5	36.62
		BRAM	**11**	854	253	6.0	39.18

values of *M*, the resource expenditure associated with LUTs are the ones that provides the highest accelerations with the lowest occupancy. Following the same analogy, FFs and BRAMs, seem to follow the same trend in which the optimum *M* is a bit higher than the LUTs optimized value. In this case, the *M* value are between 7 and 12, and 8 and 13 for the FFs and BRAMs case respectivetly.

From this selection of implementations, it is necessary to decide which is the most limited resource in a future cryptographic implementation where this module fulfills a given function. It can be seen that LUTs and FFs are kept at around 1% and 0.20%, respectively, a significantly low occupancy value. Additionally, the high efficiency of the BRAMs has its origin in the fact that the number of BRAMs used does not follow a linear tendency, with fluctuations when M is not a power of 2.

4.3 Single-Power Analysis in NTRU AU

4.3.1 Introduction

One crucial aspect among the PQC study and implementation is to guarantee that no sensitive information is leaked. Additionally, several post-quantum constructions are particularly vulnerable to SCAs that exploit specifically chosen ciphertexts to amplify the observed leakage, known as Chosen-Ciphertext Side-Channel Analysis, [155]. The security evaluation of post-quantum schemes against attacks and the development of effective countermeasures to mitigate them are open research topics [156].

In passive attacks, sensitive information is inferred by observing physical characteristics of the device such as execution time, power consumption, or electromagnetic emanation during the normal operation mode [157]. In the case of power SCAs, two categories are distinguished SPA and Differential Power Analysis (DPA) attacks. SPA is based on visual inspection of one or few power consumption measurements collected over a period of time. DPA is a method for analyzing large number of power consumption traces using statistical analysis and error correction techniques. This approach takes multiple power consumption power traces during the normal operation of the device for different input plaintexts for the same key. Calculating the correlation between the hypothetical power consumption and the measured power consumption for an attacked intermediate value, the secret key can be obtained with the maximum correlation value. Given enough traces, even tiny correlations can be detected. Using DPA, an adversary can obtain secret keys by analyzing correlations obtained from multiple cryptographic operations. DPA is a more complex and powerful technique than SPA, but the use of SPA is very useful when data-dependent features in the power traces are significant. Based on the SPA, a designer can make decisions to achieve an efficient and secure implementation of a cryptographic algorithm.

The study presented in this Section is focused on the NTRU submitted to the NIST post-quantum standardization project. Some of the proposed NTRU implementations in the literature are vulnerable to both power [158] and timing analysis [159]. A simple and effective

countermeasure to timing attacks is to perform implementations that run in constant time, whereas the protection against power analysis attacks is usually more complex.

Several power analysis attacks using various techniques have been reported on the different NIST PQC candidates. A side-channel assisted chosen ciphertext attack on KYBER is presented in [160], a SCA on the masked implementation of SABER is detailed in [161], and two vulnerabilities in the NTRU algorithm and a side-channel assisted attack to exploit them are reported in [162]. Finally, single-trace SCAs on the message encoding of several of the NIST PQC finalists are described in [163].

This Section addresses the issue of SPA performed on PQC, with a particular focus on the NTRU cryptosystem already presented in Section 4.2. The main objective is to analyze the effectiveness of these information gathering measures in the context of PQC. The study includes experimental analysis of polynomial multiplication based on the NTRU cryptosystem, which can be susceptible to SCAs using power measurements. Several countermeasures and strategies are also suggested to reduce potential information leakage in this scenario. Overall, this Section aims to focus on the importance of securing systems that use PQC that are still vulnerable to side-channel attacks.

The AU used is the one described in Figure 4.2, whose mode of operation is described in Table 4.3. To hardware description, the Verilog HDL was employed, while the RTL-based design flow provided by Xilinx Vivado tools was utilized in developing the multiplier module. In terms of occupation, the AU implementation requires 22 LUTs in Xilinx Spartan6 family series. As can be observed in Table 4.3, when the AU is operating using the coefficients of $r(x)$, $r_i = 01$ and $r_i = 11$, it performs the addition or the subtraction of the coefficient of $h(x)$, h_j, respectively. However, when a zero coefficient of $r(x)$ ($r_i = 00$) acts as input of the AU, no operation is performed, maintaining at the output the same result as at the input. Therefore, it is evident that, since there is a clear difference in the behavior of the AU, there must also be a difference in the power consumption of the arithmetic module depending on the coefficient of $r(x)$ it has at its input. This may constitute an input vector for an attacker who wants to obtain information regarding the obfuscation polynomial, $r(x)$. To do this, a SPA of the power consumption of the arithmetic circuit would be sufficient to see if there are differences in power consumption corresponding to different values of the coefficients of $r(x)$.

4.3.2 Experimental Setup

To analyze the SPA flaws of different hardware proposals, the design has been implemented in FPGA. It has been used the SAKURA-G board [164], specifically designed to test cryptographic hardware implementations against SCAs. The SAKURA-G board, is composed by 2 FPGAs, namely controller FPGA and crypto FPGA (both Xilinx Spartan6 FPGA), where the cryptogra-

phic module can be isolated from the rest of the design to measure its power consumption with high precision. SAKURA-G has specific circuitry to measure the power consumption traces, which, along with a PC and an oscilloscope, allows the SPA attack to be carried out.

Figure 4.9 shows an scheme of the used experimental setup. The Device Under Test (DUT) is the attacked hardware module implemented on the crypto Spartan6 FPGA of SAKURA-G board. The used PC (i7, 64 RAM, Windows 10) runs Matlab and the software to control the oscilloscope. The used oscilloscope is the PicoScope 3406D, with bandwidth of 200MHz and 250MS/s. Several power-consumption traces have been experimentally measured for each implementation and then processed under Matlab to perform the SPA attack.

4.3.3 SPA of the NTRU AU

The results of the power analysis of the AU used in this section are shown in Figure 4.10, where the analyzed power trace is shown in blue in the upper graph. These data correspond to a selection of a few cycles of operation. Specifically, about 45 coefficients of $r(x)$ being operated each of them N times by the coefficients of $h(x)$. The bottom graph is used to clarify the data presented in the power trace using `findchangepts` Matlab function. The orange line represents the jumps that are occurring in the power consumption of the module. Just below this line are represented the r_i coefficients that are being used in each cycle (framed by vertical black lines) to operate in the arithmetic module. As can be seen, firstly, there is a difference in the power consumption of the multiplication module and secondly, this difference is exclusively due to when null coefficients in $r(x)$ are being used.

The above can be summarized as follows: two thresholds of power consumption corresponding to null elements and non-null elements could be clearly established. This is a clear

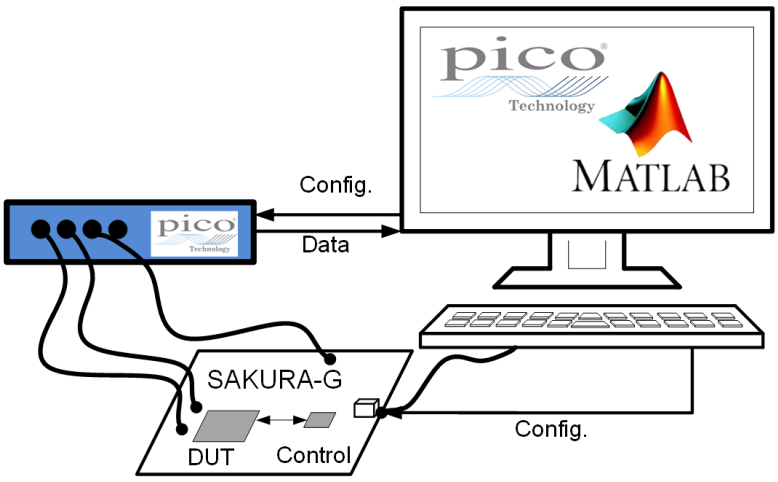


Figure 4.9 Experimental setup scheme.

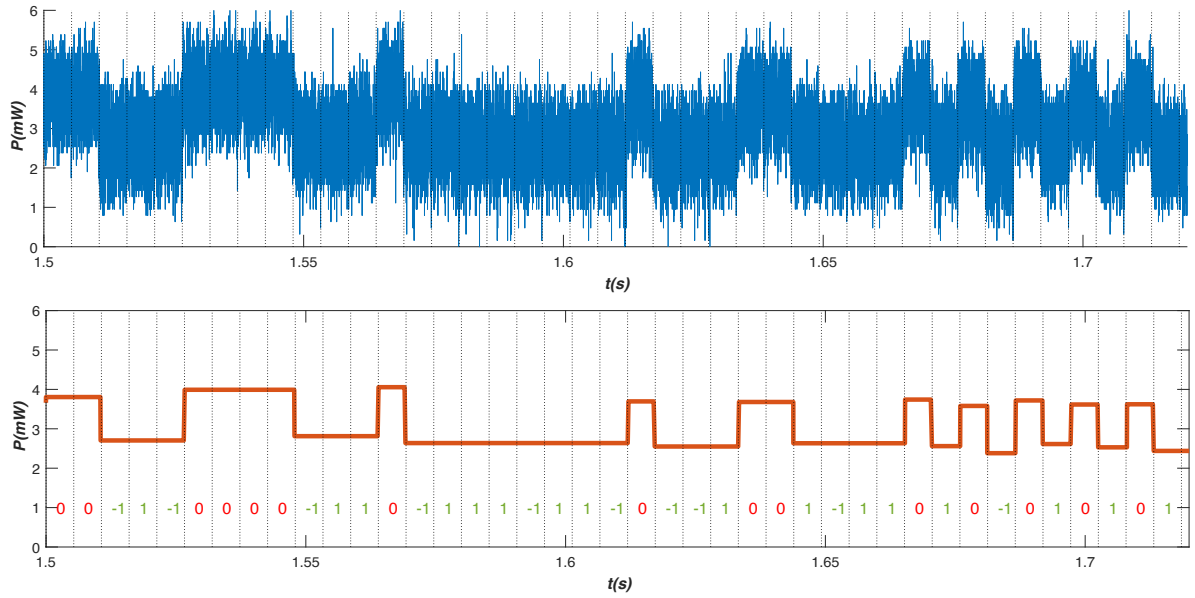


Figure 4.10 SPA of the NTRU multiplication module

vulnerability that allows knowing not only the number of null elements, but also their position in $r(x)$, which would mean a reduction of the search space for a possible attack on the cryptosystem. Therefore, several countermeasures are proposed that could mitigate the problem in advance.

4.3.4 SPA of the countermeasures proposed

The design of countermeasures in this dissertation seeks to mitigate the differences in power consumption that affect the current AU by maintaining the functionality. The first proposed countermeasure tries to obfuscate the power consumption instantiating a dummy AU, without considering its output as can be observed in Figure 4.11. The function of this AU is to operate in the opposite way than the original one, thereby maintaining equal power consumption when working with 0 (then the dummy would act as a -1) as when not (then the dummy would act as a 0). The second proposed countermeasure tries to mitigate the differences in the power consumption instantiating 3 AUs whose inputs are the three possible values of the coefficients of $r(x)$, as it can be observed in Figure 4.12. The output of the operation is controlled by a multiplexer that uses the coefficient r_i to select between inputs. In this case, since the 3 AUs are operating at the same time, the power consumption should not suffer great differences beyond the difference in consumption that may occur in the selection of the multiplexer. In terms of occupation, the first proposed countermeasure requires 45 LUTs, while the second one 110 LUTs in Xilinx Spartan6 family series.

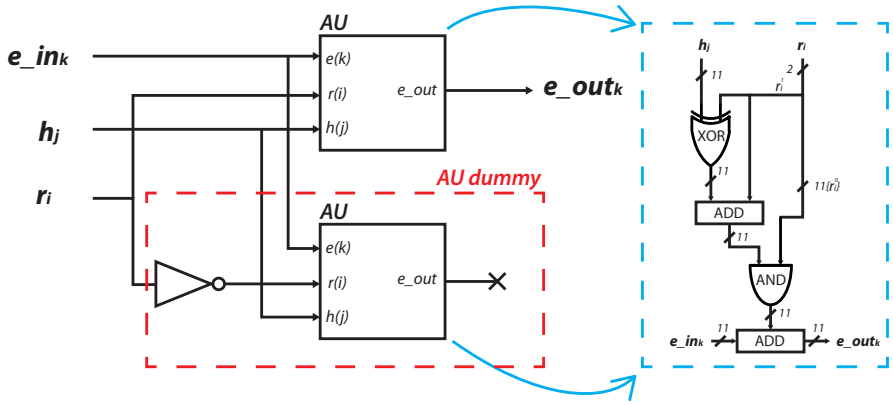


Figure 4.11 First countermeasure proposed.

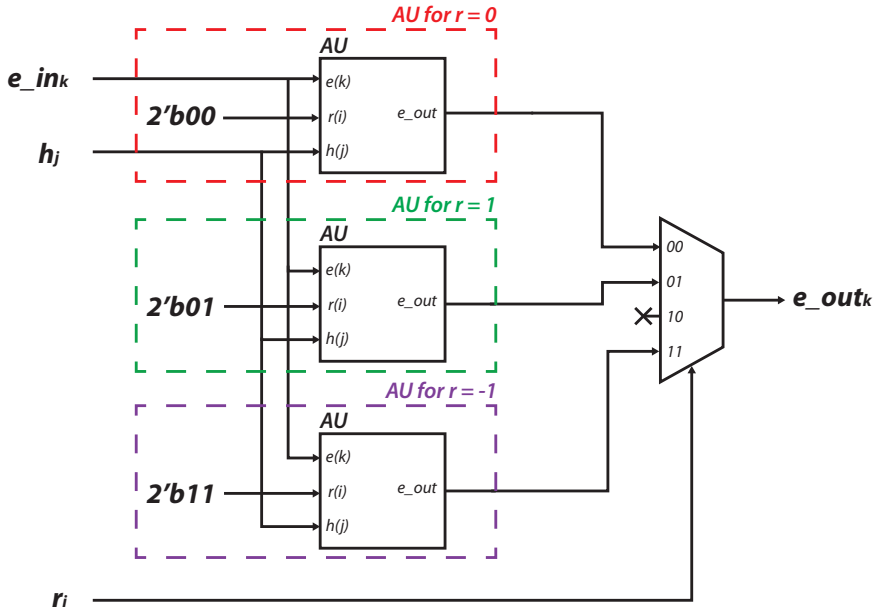


Figure 4.12 Second countermeasure proposed.

SPAs are also performed to corroborate if there are still differences in the power consumption of these two proposed countermeasures. The results are shown in Figure 4.13. In these graphs it is possible to visualize the power traces in blue, as well as, over them, an orange line that allows to clarify and observe the existence of changes in the power due to changes in the coefficients of $r(x)$, for which the same Matlab function used in Figure 4.10 has been applied. In order from top to bottom, the power consumption of the NTRU multiplication module with the original AU, with the first countermeasure and, with the second countermeasure are shown. The last graph shows the comparison between the different power consumption of the three designs. As can be observed, there is hardly any difference in the power consumption either between

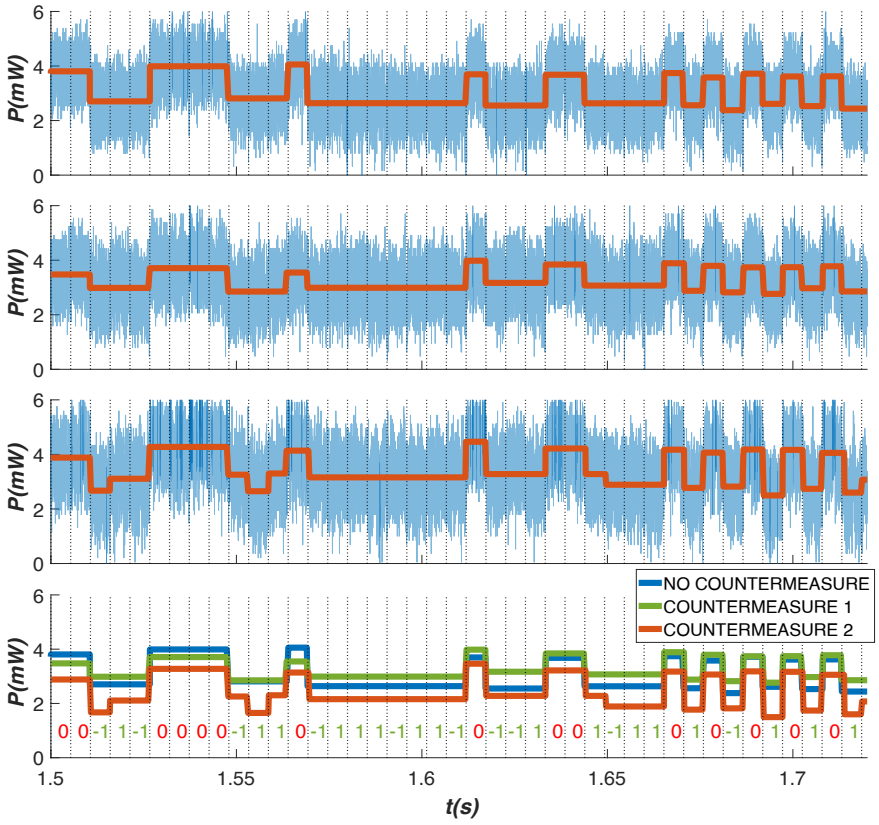


Figure 4.13 SPA of the NTRU multiplication module with the original AU and the two countermeasures proposed.

different countermeasures or between the countermeasures and the original case. This supposes that the proposed countermeasures are continuing to produce a noticeable impact on the power usage, even when a null coefficient is employed. As a result, it is still feasible to determine the quantity and location of the zero coefficient of $r(x)$, not being effective in mitigating the differences in power consumption, and therefore, generating information leaks. This may be because the extra circuitry added (the inverter in one case and the multiplexer in another) is very sensitive to changes in the input bits and therefore their inclusion still leaks information. The control logic could also have some effect when operating with zero elements. At this point, from the algorithmic point of view, it is interesting to study the SPA of the recently-published accelerated version of the NTRU cryptosystem.

4.3.5 SPA of the accelerated algorithm

Since the previously formulated countermeasures have not been successful in reducing the differences in power consumption that may lead to information leakage, a new analysis is

proposed to corroborate if the latest version of the hardware implementation of the algorithm first presented in [150] still produces this type of information leakage. The idea behind this countermeasure is to hide the most sensitive coefficients without affecting the operation. As previously mentioned, the number of cycles required to perform the multiplication operation would be $N \cdot N$, however, when $r_i = 0$ no operation is performed, so at the algorithmic level not operating or operating with 0 it is exactly the same. This was already applied in [138] or [124] to reduce the number of cycles in the complete operation of the NTRU cryptosystem.

Regarding this idea, the SPA can corroborate if there are still differences in the power consumption. The results can be shown in Figure 4.14. In the same way as in the previous figures, the power traces are in blue and a line is shown in orange that makes it easier to see if there are jumps in power consumption. In the upper graph, the power consumption of the NTRU multiplier with the original AU presented in Figure 4.2 is shown while, in the middle graph, the application of the algorithm acceleration is displayed. Although the accelerated algorithm must be applied from the start of the multiplication operation, in this case, to illustrate the change in power consumption, such acceleration is applied to the middle of the display window.

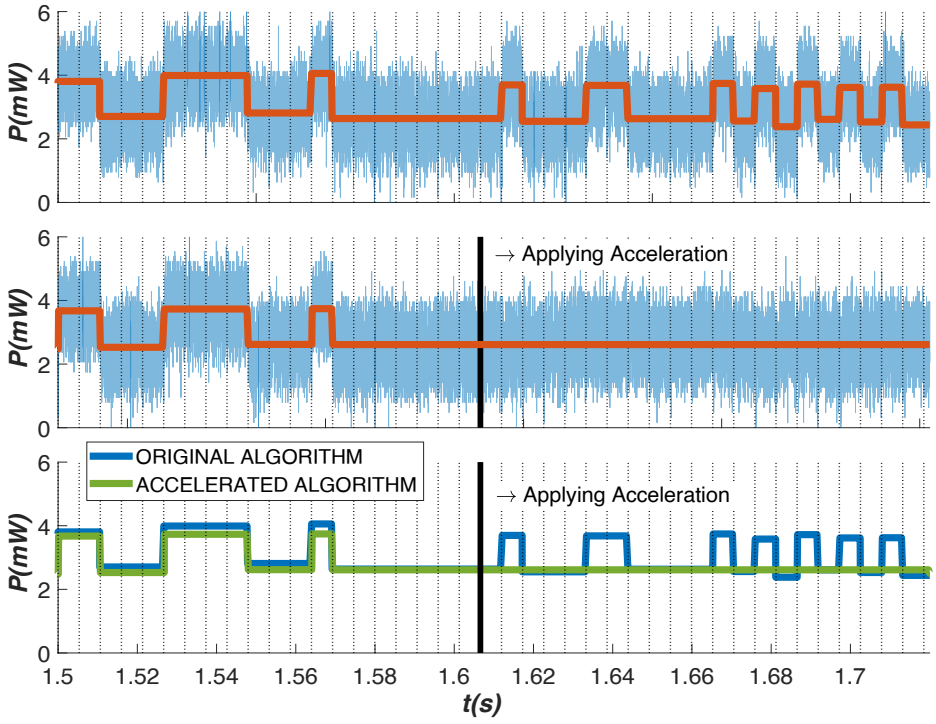


Figure 4.14 SPA of the NTRU multiplication module with the original algorithm versus the application of the accelerated version.

When applying the acceleration, it can be observed how when jumping the null coefficients there is no longer a difference in power between different values of r_i. Apart from that, Figure 4.15 shows one of the operation cycles in detail from the previous analysis. Since a jump is occurring when a null-coefficient appears in the accelerated version of the algorithm, it is conceivable that the null-coefficient jump itself may cause some information leakage. In the detail it can be seen how it is impossible to locate where the null-coefficient jump is occurring in the accelerated version. In this way, an attacker who is carrying out a SCA attack, through the use of a SPA, will not be able to obtain any type of internal information, either the number or the position of the null coefficients. This solution, therefore, is not only faster but also avoids information leaks caused by AU.

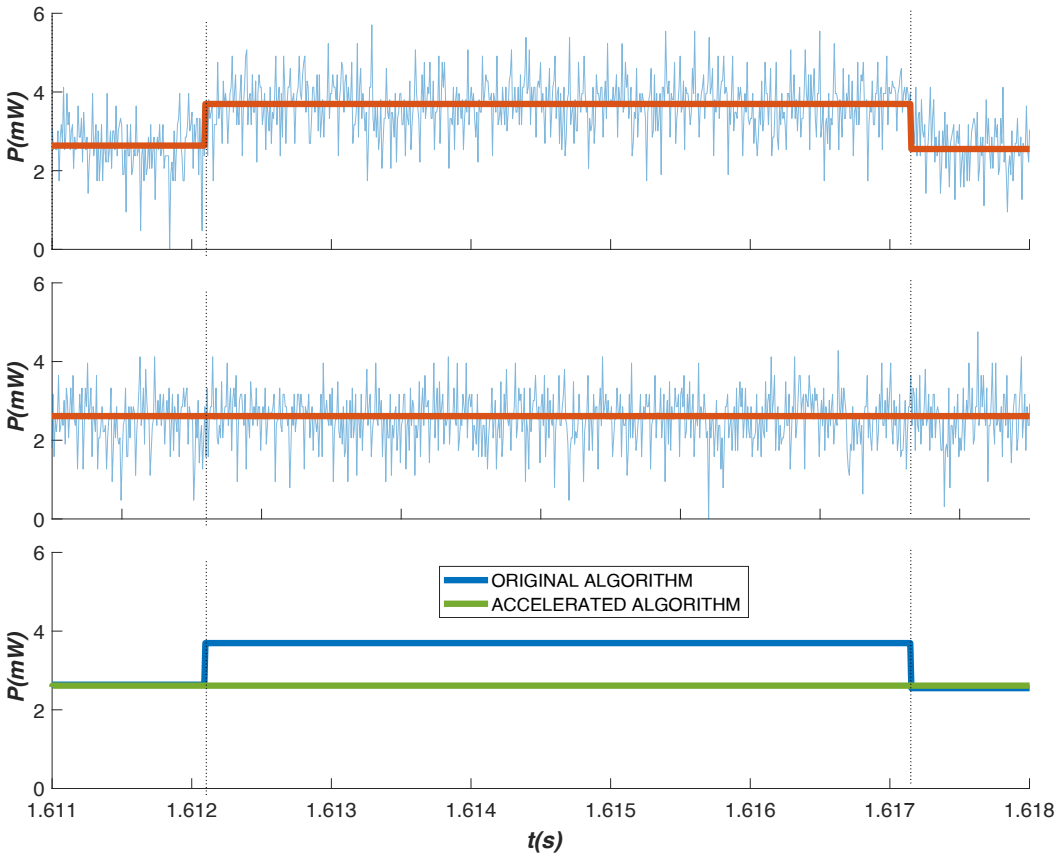


Figure 4.15 Detail of the SPA of the NTRU multiplication module with the original algorithm versus the application of the accelerated version.

4.4 Conclusions

In conclusion, this dissertation has delved into the critical realm of Post-Quantum Cryptography, with a particular focus on the NTRU algorithm as one of the selected candidates. Recognizing the imperative to reduce the execution time of these algorithms, a pivotal strategy emerged: the implementation of hardware-based operations.

To this end, a comprehensive library of the latest version of NTRU was meticulously developed and released. The acceleration approach was twofold: first, the parallelization of the AU was determined by the discretion of the designer, and second, a strategy was employed to considering only nonzero coefficients in the blind polynomial. However, it was also revealed that this strategy could potentially introduce vulnerabilities to timing attacks. To address this concern, an exhaustive analysis of vulnerabilities was conducted, resulting in the proposition of a countermeasure involving controlled zero-jumping, effectively establishing a security threshold. Subsequently, a comprehensive study was undertaken to determine the most efficient strategy, weighing both acceleration and resource utilization, with and without countermeasures.

Another vulnerability came to light in the AU, presenting a challenge that resisted resolution despite the introduction of several proposed countermeasures. Consequently, the advocated strategy was a resolute commitment to zero-jumping. Nevertheless, further evaluations in the domain of SCA remain imperative. An in-depth assessment employing methodologies such as Machine Learning within SPA holds promise. Additionally, DPA could be a viable avenue of exploration within the proposed design.

Furthermore, in consideration of the latest published draft of NIST PQC standards, a noteworthy advancement emanates from this dissertation: the successful hardware implementation of the ML-KEM algorithm. This achievement not only contributes to the evolving landscape of post-quantum cryptographic solutions but also underscores the significance of robust hardware implementations in fortifying digital security.

In summary, this section represents a significant contribution to the burgeoning field of PQC, particularly in the context of NTRU and hardware-based operations. The methodologies, strategies, and countermeasures proposed and evaluated herein lay a valuable foundation for future advancements in cryptographic protocols and implementations. This dissertation not only addresses current challenges but also paves the way for more secure digital communication in the post-quantum era.

Chapter 5

Final RoT design: Use Cases

5.1 Introduction

This chapter is essentially a practical demonstration, or a PoC, of the RoT that has been designed in this dissertation. The primary objective here is to validate and verify the interoperability of each module when they are put to test in real-world scenarios. This is crucial as it helps in understanding how each module interacts with others and functions in a live environment. After the individual modules of the RoT have been meticulously designed, the next step involves their integration into a unified RoT. This integration is visually represented in Figure 5.1. This figure serves as a graphical representation of the integrated modules, providing a clear and concise view of the entire RoT structure.

The ultimate goal is to utilize the RoT in conjunction with a processor. To achieve this, a SoC architecture, specifically the Zynq-7000, has been chosen. The board selected for that was the Pynq-Z2 since it is a versatile platform that offers on the one hand, the ability

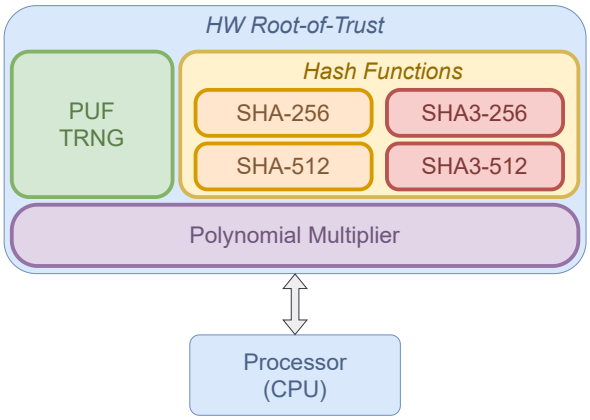


Figure 5.1 Schematic of the PoC RoT

to easily integrate digital designs into the built-in FPGA, and on the other hand, allowing a straightforward connection with the processor, facilitating the integration between them. These primitives form the building blocks of many cryptographic algorithms and protocols, to that end, they are fundamental elements that can be used in various scenarios. Those primitives are:

- *PUF/TRNG*: The significance of PUFs and TRNGs in cryptography necessitates their inclusion in the RoT. Although the digital design of the RoT on FPGA makes it impossible to incorporate the PUF developed in this dissertation, the section serves as a PoC. Therefore, the conclusions drawn from this are applicable to the use of an RTN-based PUF. The PUF and TRNG used in this PoC are the ones presented in the referenced work [165].

- *Hash Function*: The RoT incorporates widely used hash functions in cryptographic implementations. These include SHA-2 based functions (SHA-256 and SHA-512) and SHA-3 based functions (SHA3-256 and SHA3-512), which are developed in the respective sections 3.2 and 3.4, respectively.

- *Polynomial Multiplier*: The inclusion of this module has resulted in a substantial performance enhancement in the domain of PQC, particularly within the NTRU algorithm.

The integration process was carried out using the IP Integrator tool available in Vivado 2023.1. The schematic of this integration, which serves as a visual representation of the design, is illustrated in Figure 5.2. Each module within the design has its own resource occupancy, in terms of LUTs, FFs and BRAMs, which refers to the amount of hardware resources it

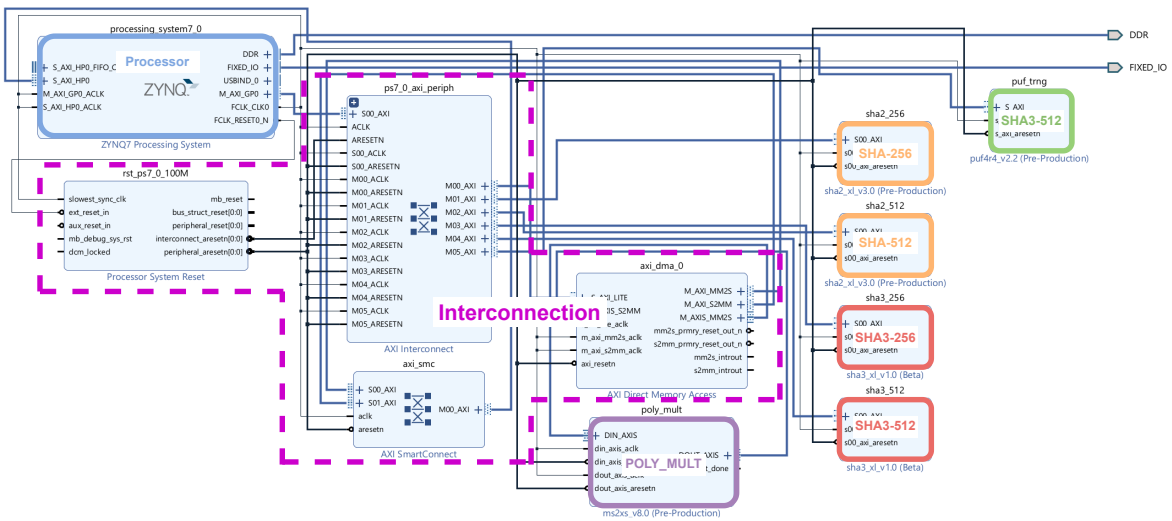


Figure 5.2 Block diagram of the PoC RoT

consumes on the FPGA. The resource occupancy of each individual module is detailed in Table 5.1. The final RoT design requires various interconnection modules. These modules serve as the communication infrastructure, enabling data transfer between different IP cores within the design. However, these interconnection modules also consume hardware resources, leading to an increase in the overall resource occupancy of the final RoT. This is an important consideration in the design process, as it impacts the scalability and complexity of the system.

Table 5.1 Resources occupancy of each module in the final RoT design.

Module	LUTs	FFs	BRAMs
Processor	24	0	0
Interconnection	4095	5688	2.5
SHA-256	1612	1196	0.5
SHA-512	3192	2097	1
SHA3-256	3859	3046	0
SHA3-512	3618	2532	0
NTRU Polynomial Multiplier	359	130	4.5
PUF / TRNG	367	390	0
Total	17126	15079	8.5

5.2 Message verification: HMAC

The Hash-based Message Authentication Code (HMAC) is a widely recognized scheme within the Message Authentication Code (MAC) family. It's designed to verify the integrity and authenticity of a message simultaneously. The HMAC process involves two main components: a cryptographic hash function and a secret cryptographic key. The combination of these two components in HMAC provides robust security features. The secret key ensures that only a sender and receiver who have access to the same secret key can generate or verify the HMAC respectively. The cryptographic hash function guarantees that even minor changes to the message will result in a significantly different hash, making it nearly impossible for attackers to modify the message undetected. Therefore, HMAC is a powerful tool in ensuring data integrity and authenticity, making it a cornerstone of secure communications in the digital world.

It was first introduced in [166] as a mechanism for message authentication across various hash functions and standardized by NIST in FIPS 198-1 [167]. It is used in several modern protocols such as Internet Protocol security (IPsec) [168] or in the standard for key derivation functions known as Password-Based Key Derivation Function 2 (PBKDF2) being part of Public-Key Cryptography Standards (PKCS) series, specifically PKCS #5 v2.0 [169]. PBKDF2

is a key derivation function that takes a password as input and generates a cryptographic key as output. The key derivation process is designed to be computationally expensive, making it difficult for an attacker to guess the original password. This added computational work makes password cracking much more difficult, and is known as key stretching.

There are many variants of HMAC, which are HMAC-SHA1, HMAC-SHA2, HMAC-MD5, that corresponds to the use of SHA-1, SHA2, and Message-Digest 5 (MD5) hash functions. Since there are many collisions attacks on SHA-1 [170] and MD5 [171] reported in the literature, they have been discarded to be used in this RoT. For that, the hash functions selected have been SHA-2 and SHA-3 to perform the HMAC function. Although SHA-3 is not recognized in FIPS 198-1 as hash function to use in HMAC, NIST expects to include it in the standard NIST SP 800-224 in the future.

For the better understanding of the proposed implementation, the mathematical expression of the HMAC taken from FIPS 198-1 is provided:

$$\mathrm{HMAC}(K,m) = \mathrm{H}\Big(\big(K' \oplus opad\big) \parallel \mathrm{H}\big(\big(K' \oplus ipad\big) \parallel m\big)\Big) \tag{5.1}$$

$$K' = \begin{cases} \mathrm{H}(K) & \text{if } K \text{ is larger than block size} \\ K & \text{otherwise} \end{cases} \tag{5.2}$$

where

- H is the SHA-2 (256) cryptographic hash function.
- m is the message to be authenticated.
- K is the secret key.
- K' is a block-sized key derived from the secret key, K; either by padding to the right with 0s up to the block size, or by hashing down to less than or equal to the block size first and then padding to the right with zeros.
- $\parallel$ denotes concatenation.
- $\oplus$ denotes bitwise exclusive or (XOR).
- $opad$ is the block-sized outer padding, consisting of repeated bytes valued $0x5c$.
- $ipad$ is the block-sized inner padding, consisting of repeated bytes valued $0x36$.

In the context of the proposed RoT, the key generation of this HMAC implementation can be done by means of a PUF that can function as an identity generator or as a randomness generator

via a TRNG. The HMAC function developed in this dissertation can be found in [172]. This function allows performing HMAC for a hexadecimal message (`-m`) and a hexadecimal key (`-k`). The use of different hash functions within HMAC is selected with `-s`. The SHA-256 is selected with `-s = 1`; SHA-512 with `-s = 2`; SHA3-256 with `-s = 3` and SHA3-512 with `-s = 4`. The PUF can be used with the selection as ID generator (`-p`) or as TRNG (`-t`). The long of the key in bits is selected with `-n`, with a maximum of 256 bits. Some execution example are shown in Figures 5.3 and 5.4, in which it is possible to observe how different calling return an improvement in the execution time versus the HMAC software implementation.

```
root@pynq:/home/xilinx/HMAC_v2# ./HMAC -m 0123456789 -k 0123456789 -s 2

 --- HMAC (SHA-512) ---

 hmac_hw:
58baeb7ff096b478cfa7dad314a972474bcb968e4736537c62bead6b33c434cb
a98a158136d09765e62819954f280c81fbfcd95f959f765e57234a5a662ad102

 hmac_sw:
58baeb7ff096b478cfa7dad314a972474bcb968e4736537c62bead6b33c434cb
a98a158136d09765e62819954f280c81fbfcd95f959f765e57234a5a662ad102

 HW Time:        (   183 us.)
 SW Time:        (   295 us.)
    Acc.:        1.61
```

Figure 5.3 Execution example of the HMAC function

```
root@pynq:/home/xilinx/HMAC_v2# ./HMAC -m 0123456789 -p -s 4

 --- HMAC (SHA3-512) ---
   ---  ID: 64000 ---

 key (l=128):
9e4c3056ed799b8a0478ef2734dec0f4

 hmac_hw:
7fc9bd61096458d1ada04b6427bd278fa5a111c906c54f20d80334b39150e4c8
217b3e9321afbfb1fd35598164215b10e66f89a4a3575a11f287982f1b61481a

 hmac_sw:
7fc9bd61096458d1ada04b6427bd278fa5a111c906c54f20d80334b39150e4c8
217b3e9321afbfb1fd35598164215b10e66f89a4a3575a11f287982f1b61481a

 HW Time:        (   154 us.)
 SW Time:        (  1074 us.)
    Acc.:        6.97
```

(a) HMAC with PUF

```
root@pynq:/home/xilinx/HMAC_v2# ./HMAC -m 0123456789 -t -s 4

 --- HMAC (SHA3-512) ---
   ---  ID: 64000 ---

 key (l=128):
8b6a3d58f5a47386a10e57042ab7080f

 hmac_hw:
df8cfe26f65eaf693d6631d8bcd2402a9cdf5d8e5e5a1875fc05171f9df06667
0ad40ac4110c2d72605a44dba10a395412bf361008ab750b5f8c48b05f01f984

 hmac_sw:
df8cfe26f65eaf693d6631d8bcd2402a9cdf5d8e5e5a1875fc05171f9df06667
0ad40ac4110c2d72605a44dba10a395412bf361008ab750b5f8c48b05f01f984

 HW Time:        (   154 us.)
 SW Time:        (  1088 us.)
    Acc.:        7.06
```

(b) HMAC with TRNG

Figure 5.4 Execution example of the HMAC function using the PUF/TRNG.

5.3 Adding new functionalities to the NTRU cryptosystem

The primary objective of this section is to accomplish a comprehensive hardware integration of the cryptographic functions inherent in the NTRU cryptosystem. The aim is to ensure that as many cryptographic functions as possible are thoroughly implemented in hardware. This comprehensive approach not only enhances the robustness of the cryptosystem but also ensures its reliability and security. To that end, in the NTRU implementation utilized in this dissertation, there are certain functions that continue to be executed in software. These include hash functions based on SHA3, specifically SHA3-256 and SHA3-512. These functions play a crucial role in the overall operation of the NTRU algorithm. On the other hand, the generation of keys in this implementation requires a seed. This seed can be produced using the TRNG which ensures that the keys generated are truly random and secure. The polynomial multiplier, a key component of the NTRU algorithm, is applied in the same manner as described in the NTRU section of the dissertation. This consistency in application ensures the integrity and security of the cryptographic process.

As depicted in Figure 5.5, the TRNG plays a role as a seed generator in the process of key pair generation. This use of TRNG ensures the generation of truly random and secure keys,

```
root@pynq:/home/xilinx/NTRU_3Round_DEMO# ./Demo_509 -k -v 1

 --- HARDWARE INITIALIZATION ---

 Generating keys ...
 Generating random seed ...    ---  ID: 64000 --- Complete
 Seed:
0ac419a81eb21924a30f7729a07a2c879e8055dbb30e3ad21c4913c0b4e8ba3a1ec31e7205642d6f66c9fde398a7245c
        Complete
 Storing keys ...        Complete

root@pynq:/home/xilinx/NTRU_3Round_DEMO# ./Demo_509 -e -v 1

 --- HARDWARE INITIALIZATION ---

 Loading public key ...          Complete
 Generating shared secret and ciphertext in SW and HW/SW...
 Performing SHA3-256 in HW ...  Complete
        Complete

     Time SW:    14906 us.
     Time HW:     2208 us.
 Acceleration:   6.750906

 Storing ciphertext ...          Complete
 ss: fba94ea952da1c062cfb1a3d3340f075d49acdd0e6ceb3f055a435b6243eef4e
root@pynq:/home/xilinx/NTRU_3Round_DEMO# ./Demo_509 -d -v 1

 --- HARDWARE INITIALIZATION ---

 Loading secret key ...          Complete
 Loading ciphertext ...          Complete
 Recovering shared secret in SW and HW/SW...
 Performing SHA3-256 in HW ...  Complete         Complete

     Time SW:    45425 us.
     Time HW:    32515 us.
 Acceleration:   1.397048

 ss_SW: fba94ea952da1c062cfb1a3d3340f075d49acdd0e6ceb3f055a435b6243eef4e
 ss_HW: fba94ea952da1c062cfb1a3d3340f075d49acdd0e6ceb3f055a435b6243eef4e
```

Figure 5.5 Completing the NTRU hardware implementation.

which is a fundamental aspect of the NTRU cryptosystem. In addition to this, the figure also illustrates how, within the context of `ntruhps2048509`, certain hash functions such as SHA3-256 have been transitioned to a hardware implementation. This transition to hardware not only enhances the efficiency of these functions but also contributes to the overall performance and security of the NTRU cryptosystem. The hardware implementation of these cryptographic functions is a significant step towards achieving a fully integrated hardware-based NTRU cryptosystem.

Chapter 6

Conclusions

The establishment of a hardware RoT significantly bolsters the security of an embedded system. This enhancement is not only robust but also efficient, making it a critical component in the design of secure systems. The considerations for implementation within the framework of the IoT have led to a streamlined design of the RoT. This design is mindful of the constraints of IoT devices, particularly the size of transistors or the occupancy of logic resources in terms of FPGA implementations. The result is a compact yet powerful RoT that fits within the limited resources of IoT devices while providing robust security. In addition to the primary conclusion, there are several other insights extracted from this dissertation. These insights shed light on various aspects of the cryptosystem and contribute to our understanding of its design and implementation. They serve as valuable outcomes that can guide future research and development in this field. These conclusions are:

- Taking inspiration of the CIA triad as model designed to guide policies for information security, the design of a RoT that encompasses a set of crypto primitives is a suitable solution. The approach taken in this dissertation is primarily from a hardware-oriented perspective. This proposed RoT has included a PUF, which can be used as an identifier generator. They also incorporate hash functions, represented by SHA-2 and SHA-3, which are essential for ensuring data integrity. Moreover, it includes a cryptographic primitive that can securely accelerate the computation of PQC algorithms. One such PQC algorithm is NTRU, which is known for its security and efficiency. The integration of these components within a RoT provides a robust and secure hardware solution for addressing the principles of the CIA Triad in the IoT framework.

- The initial module developed in this dissertation was a PUF based on a typically undesirable phenomenon known as RTN. Several architectures were proposed in which, basically, the response of the PUF is obtained comparing one metric that encapsulate all

the RTN information, the MPF. The initial stages involved designing and evaluating the PUF from a software perspective, which included conducting several simulations. Once positive results were obtained from these simulations, the next phase involved integrating this PUF into an ASIC. This process followed an analog design flow, culminating in the creation of the first silicon-based RTN-PUF, referred to as MILESTONE-I. The successful results of this endeavor led to the filing of an international patent.

- The following modules developed were those associated with hash functions. After studying the mathematical background of both selected hash functions (SHA-2 and SHA-3), the first step involved implementing the RTL in an FPGA. To this end, a comprehensive set of parameters was incorporated into the integration of these hash functions into IP modules, allowing the implementation of any of the hash versions collected in the standards. One of these designs, specifically SHA-256, was chosen for integration into an ASIC, following a fully digital design flow.

- The final module that was developed was related to the acceleration of one of the algorithms selected by NIST in the third round of the PQC contest: the NTRU. In that algorithm, the critical component is the polynomial multiplication. To address this, a RTL design was developed, incorporating all the parameter sets outlined in the documentation. Consideration was given not only to performance but also to potential information leakage associated with timing and power attacks. The results demonstrated that it is feasible to enhance the speed of the algorithm while adding an additional layer of security against SCAs.

- Lastly, a final design of the RoT was proposed, integrating all modules into a single hardware platform. This comprehensive design was evaluated in various scenarios, demonstrating the reusability of each module. The results were highly promising, underscoring the effectiveness of this integrated approach in enhancing the security and efficiency of the embedded systems.

References

[1] A. G. Rodríguez, "A quantum cybersecurity agenda for europe," https://www.epc.eu/en/publications/A-quantum-cybersecurity-agenda-for-Europe~526b9c, 2023, accessed: Jan. 12, 2024.

[2] UpGuard, "Why is cybersecurity important?" https://www.upguard.com/blog/cybersecurity-important, 2023, accessed: Jan. 12, 2024.

[3] D. Gritzalis, M. Theocharidou, and G. Stergiopoulos, *Critical Infrastructure Security and Resilience: Theories, Methods, Tools and Technologies*. Springer International Publishing, 2019.

[4] D. J. Solove, *Understanding Privacy*. Harvard University Press, 2008, gWU Legal Studies Research Paper No. 420, GWU Law School Public Law Research Paper No. 420. [Online]. Available: https://ssrn.com/abstract=1127888

[5] J. Brenner, *America the Vulnerable: Inside the New Threat Matrix of Digital Espionage, Crime, and Warfare*. Penguin Press, 2011.

[6] J. S. Nye Jr., *The Future of Power*. PublicAffairs, 2011.

[7] E. Times, "Personal data of almost entire population of us state of maine hacked," https://ciso.economictimes.indiatimes.com/news/data-breaches/personal-data-of-almost-entire-population-of-us-state-of-maine-hacked/105140471, 2023, accessed: Jan. 12, 2024.

[8] ——, "El gobierno revela que los teléfonos de pedro sánchez y margarita robles han sido infectados con pegasus," https://www.eldiario.es/politica/gobierno-revela-telefonos-pedro-sanchez-margarita-robles-han-sido-infectados-pegasus_1_8959353.html, 2022, accessed: Jan. 12, 2024.

[9] ——, "Ciberataque en sevilla: un mes de penumbra en la web del ayuntamiento," https://www.diariodesevilla.es/sevilla/Ciberataque-Sevilla-mes-web-Ayuntamiento_0_1836116663.html, 2023, accessed: Jan. 12, 2024.

[10] E. U. A. for Cybersecurity, "European union agency for cybersecurity," https://www.enisa.europa.eu/, 2023, accessed: Jan. 12, 2024.

[11] N. I. of Standards and Technology, "Computer security resource center," https://csrc.nist.gov/, 2023, accessed: Jan. 12, 2024.

[12] I. E. T. Force, "Internet engineering task force," https://www.ietf.org/, 2023, accessed: Jan. 12, 2024.

[13] C. C. Nacional, "Centro criptológico nacional," https://www.ccn-cert.cni.es/es/, 2023, accessed: Jan. 12, 2024.

[14] A. J. Neumann, N. Statland, and R. D. Webb, "Post-processing audit tools and techniques," pp. 11–3–11–4, 1977. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nbsspecialpublication500-20.pdf

[15] H. Smith, T. Dinev, and H. Xu, "Information privacy research: An interdisciplinary review," *MIS Quarterly*, vol. 35, pp. 989–1015, 12 2011.

[16] G. Sivathanu, C. P. Wright, and E. Zadok, "Ensuring data integrity in storage: Techniques and applications," in *Proceedings of the 2005 ACM Workshop on Storage Security and Survivability*, ser. StorageSS '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 26–36. [Online]. Available: https://doi.org/10.1145/1103780.1103784

[17] S. Qadir and S. M. K. Quadri, "Information availability: An insight into the most important attribute of information security," *Journal of Information Security*, vol. 7, pp. 185–194, 2016.

[18] W. D. Casper and S. M. Papa, *Root of Trust*. Boston, MA: Springer US, 2011, pp. 1057–1060. [Online]. Available: https://doi.org/10.1007/978-1-4419-5906-5_789

[19] L. Chen, J. Franklin, and A. Regenscheid, "Guidelines on hardware-rooted security in mobile devices," https://csrc.nist.gov/pubs/sp/800/164/ipd, 2012.

[20] International Organization for Standardization, "Information processing systems — open systems interconnection — basic reference model — part 2: Security architecture," https://www.iso.org/standard/14256.html, 1989.

[21] Trusted Computing Group, "Trusted computing group glossary," http://www.trustedcomputinggroup.org/developers/glossary, 2016, accessed: Jan. 12, 2024.

[22] M. Alioto, "Trends in hardware-security: from basics to asics," *IEEE Solid-State Circuits Magazine*, vol. 11, no. 3, pp. 56–74, 2019.

[23] J. Preskill, "Quantum computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, aug 2018. [Online]. Available: https://doi.org/10.22331%2Fq-2018-08-06-79

[24] IBM, "Ibm quantum platform," https://quantum-computing.ibm.com/, 2021.

[25] J. Martinis and S. Boixo, "Quantum supremacy using a programmable superconducting processor," *Google AI Blog*, October 2019.

[26] J. Daemen and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard*. Springer, 01 2002.

[27] National Institute of Standards and Technology, "Advanced encryption standard (aes)," https://csrc.nist.gov/pubs/fips/197/final, 2001.

[28] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, ser. STOC '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 212–219. [Online]. Available: https://doi.org/10.1145/237814.237866

[29] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, p. 120–126, feb 1978. [Online]. Available: https://doi.org/10.1145/359340.359342

[30] E. Barker, L. Chen, A. Roginsky, A. Vassilev, R. Davis, and S. Simon, "Recommendation for pair-wise key-establishment using integer factorization cryptography," https://csrc.nist.gov/pubs/sp/800/56/b/r2/final, 2019, accessed: Nov. 12, 2023.

[31] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, 1994, pp. 124–134.

[32] N. I. of Standards and T. (NIST), "Post-quantum cryptography | csrc," 2023, accessed: Nov. 20, 2023. [Online]. Available: https://csrc.nist.gov/projects/post-quantum-cryptography

[33] National Institute of Standards and Technology, "Selected algorithms and key sizes for post-quantum cryptography," 2022, accessed: September 29, 2023. [Online]. Available: https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022

[34] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128610001568

[35] S. Madakam, R. Ramaswamy, and S. Tripathi, "Internet of things (iot): A literature review," *Journal of Computer and Communications*, vol. 3, pp. 164–173, 04 2015.

[36] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 57, no. 10, pp. 2266–2279, 2013, towards a Science of Cyber Security Security and Identity Architecture for the Future Internet. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128613000054

[37] M. El-Haii, M. Chamoun, A. Fadlallah, and A. Serhrouchni, "Analysis of cryptographic algorithms on iot hardware platforms," in *2018 2nd Cyber Security in Networking Conference (CSNet)*, 2018, pp. 1–5.

[38] M. Keating and P. Bricaud, *Reuse Methodology Manual for System-on-a-Chip Designs*. Springer US, 01 2002.

[39] P. Kocher, J. Horn, A. Fogh, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom, "Spectre attacks: Exploiting speculative execution," in *2019 IEEE Symposium on Security and Privacy (SP)*, 2019, pp. 1–19.

[40] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, A. Fogh, J. Horn, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg, "Meltdown: Reading kernel memory from user space," in *27th USENIX Security Symposium (USENIX Security 18)*. Baltimore, MD: USENIX Association, 2018, pp. 973–990. [Online]. Available: https://www.usenix.org/conference/usenixsecurity18/presentation/lipp

[41] A. Shamsoshoara, A. Korenda, F. Afghah, and S. Zeadally, "A survey on physical unclonable function (PUF)-based security solutions for Internet of Things," *Computer Networks*, vol. 183, p. 107593, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128620312275

[42] S. P and P. M. Krishnammal, "Study of different silicon physical unclonable functions," in *2016 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET)*, 2016, pp. 81–85.

[43] J. Lee, D. Lim, B. Gassend, G. Suh, M. van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits for identification and authentication applications," in *2004 Symposium on VLSI Circuits. Digest of Technical Papers (IEEE Cat. No.04CH37525)*, 2004, pp. 176–179.

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

[45] The reliability of sram puf. [Accessed: Oct. 25, 2023]. [Online]. Available: https://www.intrinsic-id.com/resources/whitepapers/landing-page-white-paper-reliability-sram-puf/

[46] T. McGrath, I. E. Bagci, Z. M. Wang, U. Roedig, and R. J. Young, "A PUF taxonomy," *Applied Physics Reviews*, vol. 6, no. 1, p. 011303, 02 2019. [Online]. Available: https://doi.org/10.1063/1.5079407

[47] M. S. Mispan, B. Halak, Z. Chen, and M. Zwolinski, "Tco-puf: A subthreshold physical unclonable function," in *2015 11th Conference on Ph.D. Research in Microelectronics and Electronics (PRIME)*, 2015, pp. 105–108.

[48] M. S. Mispan, B. Halak, and M. Zwolinski, "Nbti aging evaluation of puf-based differential architectures," in *2016 IEEE 22nd International Symposium on On-Line Testing and Robust System Design (IOLTS)*, 2016, pp. 103–108.

[49] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas, "Silicon physical random functions," in *Proc. of the 9th ACM Conference on Computer and Communications Security*, 2002, pp. 148–160.

[50] J. Guajardo, S. S. Kumar, G.-J. Schrijen, and P. Tuyls, "Fpga intrinsic pufs and their use for ip protection," in *Proc. of the 9th International Workshop on Cryptographic Hardware and Embedded Systems*, 2007, pp. 63–80.

[51] H. Zhuang, X. Xi, N. Sun, and M. Orshansky, "A strong subthreshold current array puf resilient to machine learning attacks," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 1, pp. 135–144, 2020.

[52] E. Camacho-Ruiz, R. Castro-Lopez, E. Roca, P. Brox, and F. V. Fernandez, "A novel physical unclonable function using rtn," in *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2022, pp. 160–164.

[53] A. B. Manut, J. F. Zhang, M. Duan, Z. Ji, W. D. Zhang, B. Kaczer, T. Schram, N. Horiguchi, and G. Groeseneken, "Impact of hot carrier aging on random telegraph noise and within a device fluctuation," *IEEE Journal of the Electron Devices Society*, vol. 4, no. 1, pp. 15–21, 2016.

[54] J. Chen, T. Tanamoto, H. Noguchi, and Y. Mitani, "Further investigations on traps stabilities in random telegraph signal noise and the application to a novel concept physical unclonable function (puf) with robust reliabilities," in *Proc. Symposium on VLSI Technology (VLSI Technology)*, 2015, pp. T40–T41.

[55] J. Brown, "Designing, implementing, and testing hardware for cybersecurity," Ph.D. dissertation, Liverpool John Moores University, 2020.

[56] M. Yoshinaga, H. Awano, M. Hiromoto, and T. Sato, "Physically unclonable function using rtn-induced delay fluctuation in ring oscillators," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 2619–2622.

[57] T. Grasser and et al., "Switching oxide traps as the missing link between negative bias temperature instability and random telegraph noise," in *Proc. IEEE International Electron Devices Meeting*, 2009, pp. 1–4.

[58] P. Saraza-Canflanca, J. Martin-Martinez, R. Castro-Lopez, E. Roca, R. Rodriguez, F. V. Fernandez, and M. Nafria, "Statistical characterization of time-dependent variability defects using the maximum current fluctuation," *IEEE Transactions on Electron Devices*, vol. 68, no. 8, pp. 4039–4044, 2021.

[59] E. Camacho-Ruiz, F. J. Rubio-Barbero, R. Castro-Lopez, E. Roca, and F. V. Fernandez, "Design considerations for a cmos 65-nm rtn-based puf," in *2023 19th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2023, pp. 1–4.

[60] A. Ortiz-Conde, F. García-Sánchez, J. Muci, A. Barrios, J. Liou, and C.-S. Ho, "Revisiting mosfet threshold voltage extraction methods," *Microelectronics and Reliability*, vol. 53, pp. 90–104, 09 2013.

[61] A. Maiti, V. Gunreddy, and P. Schaumont, "A systematic method to evaluate and compare the performance of physical unclonable functions," in *Embedded Systems Design with FPGAs*, P. Athanas, D. Pnevmatikatos, and N. Sklavos, Eds. Springer New York, 2013, pp. 245–267.

[62] P. Saraza-Canflanca, E. Camacho-Ruiz, R. Castro-Lopez, E. Roca, J. Martin-Martinez, R. Rodriguez, M. Nafria, and F. V. Fernandez, "Simulating the impact of random telegraph noise on integrated circuits," in *SMACD / PRIME 2021; International Conference on SMACD and 16th Conference on PRIME*, 2021, pp. 1–4.

[63] V. Bhatia, M. Madan, B. Kaur, N. Pandey, and A. Bhattacharyya, "A novel cc-ii based current comparator and its application as current mode flash adc," in *IMPACT-2013*, 2013, pp. 217–222.

[64] V. Thrivikramaru and R. K. Baghel, "High speed low power cmos current comparator," in *2012 International Conference on Communication Systems and Network Technologies*, 2012, pp. 764–768.

[65] A. Santana-Andreo, P. Saraza-Canflanca, H. Carrasco-Lopez, P. Brox, R. Castro-Lopez, E. Roca, and F. Fernandez, "A drv-based bit selection method for sram puf key generation and its impact on eccs," *Integration*, vol. 85, pp. 1–9, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167926022000220

[66] M. J. Kirton and M. J. Uren, "Capture and emission kinetics of individual Si:SiO2 interface states," *Applied Physics Letters*, vol. 48, no. 19, pp. 1270–1272, 05 1986. [Online]. Available: https://doi.org/10.1063/1.97000

[67] A. Tataridou, G. Ghibaudo, and C. Theodorou, ""pinch to detect": A method to increase the number of detectable rtn traps in nano-scale mosfets," in *2021 IEEE International Reliability Physics Symposium (IRPS)*, 2021, pp. 1–5.

[68] P. Martín-Lloret, A. Toro-Frías, R. Castro-López, E. Roca, F. Fernández, J. Martin-Martinez, R. Rodriguez, and M. Nafria, "Case: A reliability simulation tool for analog ics," in *2017 14th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2017, pp. 1–4.

[69] X. Chen and N. A. Touba, "Chapter 2 - fundamentals of cmos design," in *Electronic Design Automation*, L.-T. Wang, Y.-W. Chang, and K.-T. T. Cheng, Eds. Boston: Morgan Kaufmann, 2009, pp. 39–95. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780123743640500096

[70] G. De Geronimo, P. O'Connor, and A. Kandasamy, "Analog cmos peak detect and hold circuits. part 1. analysis of the classical configuration," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 484, no. 1, pp. 533–543, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900201020599

[71] ——, "Analog cmos peak detect and hold circuits. part 2. the two-phase offset-free and derandomizing configuration," *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 484, no. 1, pp. 544–556, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168900201020605

[72] F. J. Rubio-Barbero, E. Camacho-Ruiz, R. Castro-Lopez, E. Roca, and F. Fernandez, "A peak detect & hold circuit to measure and exploit rtn in a 65-nm cmos puf," in *2023 19th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD)*, 2023, pp. 1–4.

[73] R. Hogervorst, J. Tero, R. Eschauzier, and J. Huijsing, "A compact power-efficient 3 v cmos rail-to-rail input/output operational amplifier for vlsi cell libraries," *IEEE Journal of Solid-State Circuits*, vol. 29, no. 12, pp. 1505–1513, 1994.

[74] R. Coughlin and F. Driscoll, *Operational Amplifiers and Linear Integrated Circuits*. Prentice Hall, 2001. [Online]. Available: https://books.google.es/books?id=d-VHPgAACAAJ

[75] R. Poujois and J. Borel, "A low drift fully integrated mosfet operational amplifier," *IEEE Journal of Solid-State Circuits*, vol. 13, no. 4, pp. 499–503, 1978.

[76] P. Allen and D. Holberg, *CMOS Analog Circuit Design*. Oxford University Press, 2016. [Online]. Available: https://books.google.es/books?id=SWBKnQAACAAJ

[77] G. Traversi, F. D. Canio, L. Gaioni, M. Manghisoni, L. Ratti, and V. Re, "Design of bandgap reference circuits in a 65 nm cmos technology for hl-lhc applications," *Journal of Instrumentation*, vol. 10, no. 02, p. C02004, feb 2015. [Online]. Available: https://dx.doi.org/10.1088/1748-0221/10/02/C02004

[78] W. Stallings, *Cryptography and Network Security: Principles and Practice*, 8th ed. USA: PEARSON Education Limited, 2022.

[79] "FIPS 180-4 The Secure Hash Standard (SHS)," August 2015, https://doi.org/10.6028/NIST.FIPS.180-4.

[80] National Institute of Standards and Technology (NIST), "FIPS 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions," Federal Information Processing Standards Publication, 2015, Available online: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.202.pdf.

[81] "Hash Functions," https://csrc.nist.gov/projects/hash-functions, accessed: Sep. 5, 2023.

[82] J. Docherty and A. Koelmans, "A flexible hardware implementation of SHA-1 and SHA-2 Hash Functions," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, 2011, pp. 1932–1935.

[83] Y. Zhang, Z. He, M. Wan, M. Zhan, M. Zhang, K. Peng, M. Song, and H. Gu, "A New Message Expansion Structure for Full Pipeline SHA-2," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 68, no. 4, pp. 1553–1566, 2021.

[84] M. Macchetti and L. Dadda, "Quasi-pipelined hash circuits," in *17th IEEE Symposium on Computer Arithmetic (ARITH'05)*, 2005, pp. 222–229.

[85] H. Michail, G. Athanasiou, V. Kelefouras, G. Theodoridis, T. Stouraitis, and C. Goutis, "Area-Throughput Trade-Offs for SHA-1 and SHA-256 Hash Functions Pipelined Designs," *Journal of Circuits, Systems and Computers*, vol. 25, p. 1650032, 12 2015.

[86] R. Chaves, G. Kuzmanov, L. Sousa, and S. Vassiliadis, "Cost-Efficient SHA Hardware Accelerators," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 16, no. 8, pp. 999–1008, 2008.

[87] SOG-IS Crypto Working Group, "SOG-IS Crypto Evaluation Scheme Agreed Cryptographic Mechanisms," https://www.sogis.eu/documents/cc/crypto/SOGIS-Agreed-Cryptographic-Mechanisms-1.2.pdf, accessed: Jan. 12, 2024.

[88] M. M. Wong, J. Haj-Yahya, S. Sau, and A. Chattopadhyay, "A new high throughput and area efficient sha-3 implementation," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2018, pp. 1–5.

[89] R. Chaves, G. Kuzmanov, L. Sousa, and S. Vassiliadis, "Improving SHA-2 Hardware Implementations," in *8th International Workshop in Cryptographic Hardware and Embedded Systems - CHES 2006*, 10 2006, pp. 298–310.

[90] H. Michail, G. Athanasiou, A. Kritikakou, C. Goutis, A. Gregoriades, and V. Papadopoulou, "Ultra high speed sha-256 hashing cryptographic module for ipsec hardware/software codesign," in *2010 International Conference on Security and Cryptography (SECRYPT)*, 2010, pp. 1–5.

[91] H. Mestiri, F. Kahri, B. Bouallegue, and M. Machhout, "Efficient FPGA Hardware Implementation of Secure Hash Function SHA-2," *International Journal of Computer Network and Information Security (IJCNIS)*, vol. 7, no. 1, pp. 9–15, 2015.

[92] M. Padhi and R. Chaudhari, "An optimized pipelined architecture of SHA-256 hash function," in *7th International Symposium on Embedded Computing and System Design (ISED)*, 2017, pp. 1–4.

[93] V. D. Phan, H. L. Pham, T. H. Tran, and Y. Nakashima, "High Performance Multicore SHA-256 Accelerator using Fully Parallel Computation and Local Memory," in *IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS)*, 2021, pp. 1–3.

[94] S.-H. Lee and K.-W. Shin, "An efficient implementation of SHA processor including three hash algorithms (SHA-512, SHA-512/224, SHA-512/256)," in *International Conference on Electronics, Information, and Communication (ICEIC)*, 2018, pp. 1–4.

[95] E. Camacho-Ruiz, S. Sánchez-Solano, M. C. Martínez-Rodriguez, and P. Brox, "SHA-2 Repository of this dissertation," https://gitlab.com/hwsec/sha2.

[96] "SHA-2 Hash Function Test Vectors for Hashing Byte-Oriented Messages," https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/shs/shabytetestvectors.zip, accessed: Sep. 4, 2023.

[97] "SHA-2 Hash Function Test Vectors for Hashing Bit-Oriented Messages," https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/shs/shabittestvectors.zip, accessed: Sep. 4, 2023.

[98] "PYNQ—Python Productivity for Zynq," http://www.pynq.io, accessed: Sep. 24, 2023.

[99] N. Brown, "PYNQ API: C API for PYNQ FPGA Board," https://github.com/mesham/pynq_api, accessed: Sep. 24, 2023.

[100] "SHA2 of Oryx Embedded," https://www.oryx-embedded.com/doc/dir_a9aab978e0be629e504b25df915d67e8.html, accessed: Sep. 4, 2023.

[101] European Commission, "Secure platform for ict systems rooted at the silicon manufacturing process (spirs)," 2021, grant Agreement Number: 952622. [Online]. Available: https://cordis.europa.eu/project/id/952622

[102] "Cryptographic Algorithm Validation Program (CAVP)," October 2016, https://csrc.nist.gov/Projects/Cryptographic-Algorithm-Validation-Program/Secure-Hashing#shavs.

[103] National Institute of Standards and Technology, "Nistir 8413, cybersecurity framework version 1.1 (final)," https://csrc.nist.gov/publications/detail/nistir/8413/final, accessed: September 29, 2023.

[104] S.-J. Chang, R. Perlner, W. Burr, M. Turan, J. Kelsey, S. Paul, and L. Bassham, "Third-Round Report of the SHA-3 Cryptographic Hash Algorithm Competition, (NIST," http://nvlpubs.nist.gov/nistpubs/ir/2012/NIST.IR.7896.pdf, 2012.

[105] S. M. Jungk B., "Hobbit - smaller but faster than a dwarf: revisiting lightweight sha-3 fpga implementations," in *2016 IEEE International Conference on Reconfigurable Computing and FPGAs - ReConFig '16*, 2016.

[106] B. Jungk and M. Stöttinger, "Serialized lightweight sha-3 fpga implementations," *Microprocessors and Microsystems*, vol. 71, p. 102857, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0141933117302818

[107] L. Ioannou, H. E. Michail, and A. G. Voyiatzis, "High performance pipelined fpga implementation of the sha-3 hash algorithm," in *2015 4th Mediterranean Conference on Embedded Computing (MECO)*, 2015, pp. 68–71.

[108] F. Kahri, H. Mestiri, B. Bouallegue, and M. Machhout, "High speed fpga implementation of cryptographic keccak hash function crypto-processor," *Journal of Circuits, Systems and Computers*, vol. 25, no. 04, p. 1650026, 2016.

[109] G. S. Athanasiou, G.-P. Makkas, and G. Theodoridis, "High throughput pipelined fpga implementation of the new sha-3 cryptographic hash algorithm," in *2014 6th International Symposium on Communications, Control and Signal Processing (ISCCSP)*, 2014, pp. 538–541.

[110] M. Sundal and R. Chaves, "Efficient fpga implementation of the sha-3 hash function," in *2017 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2017, pp. 86–91.

[111] X. Zhou, L. Wu, and X. Zhang, "Hardware design of sha-3 for pqc classic mceliece," in *2021 IEEE 15th International Conference on Anti-counterfeiting, Security, and Identification (ASID)*, 2021, pp. 140–144.

[112] E. Camacho-Ruiz, S. Sánchez-Solano, M. C. Martínez-Rodríguez, and P. Brox, "A complete sha-3 hardware library based on a high efficiency keccak design," in *2023 IEEE Nordic Circuits and Systems Conference (NorCAS)*, 2023, pp. 1–7.

[113] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche, *Keccak*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[114] A. Arshad, D.-e.-S. Kundi, and A. Aziz, "Compact implementation of sha3-512 on fpga," in *2014 Conference on Information Assurance and Cyber Security (CIACS)*, 2014, pp. 29–33.

[115] H. S. Jacinto, L. Daoud, and N. Rafla, "High level synthesis using vivado hls for optimizations of sha-3," in *2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2017, pp. 563–566.

[116] E. Camacho-Ruiz, S. Sánchez-Solano, M. C. Martínez-Rodriguez, and P. Brox, "SHA-3 Repository of this dissertation," https://gitlab.com/hwsec/sha3_shake.

[117] "SHA-3 Hash Function Test Vectors for Hashing Bit-Oriented Messages," https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/sha3/sha-3bittestvectors.zip, accessed: Nov. 24, 2023.

[118] "SHA-3 Hash Function Test Vectors for Hashing Byte-Oriented Messages," https://csrc.nist.gov/CSRC/media/Projects/Cryptographic-Algorithm-Validation-Program/documents/sha3/sha-3bytetestvectors.zip, accessed: Nov. 24, 2023.

[119] "Tiny-SHA3 of mjosaarinen," https://github.com/mjosaarinen/tiny_sha3, accessed: Jul. 24, 2023.

[120] European Union Agency for Cybersecurity, "Enisa - research and innovation," https://www.enisa.europa.eu/topics/research-and-innovation/research-and-innovation, accessed: September 29, 2023.

[121] National Institute of Standards and Technology. (2022) Fips 203: Standard for personal identity verification (piv) of federal employees and contractors. Accessed: September 29, 2023. [Online]. Available: https://csrc.nist.gov/publications/detail/fips/203/ipd

[122] V. Kostalabros, J. Ribes-González, O. Farràs, M. Moretó, and C. Hernandez, "Hls-based hw/sw co-design of the post-quantum classic mceliece cryptosystem," in *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*, 2021, pp. 52–59.

[123] D. B. Roy, T. Fritzmann, and G. Sigl, "Efficient hardware/software co-design for post-quantum crypto algorithm sike on arm and risc-v based microcontrollers," in *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, 2020, pp. 1–9.

[124] E. Camacho-Ruiz, M. C. Martínez-Rodríguez, S. Sánchez-Solano, and P. Brox, "Timing-attack-resistant acceleration of ntru round 3 encryption on resource-constrained embedded systems," *Cryptography*, vol. 7, no. 2, 2023. [Online]. Available: https://www.mdpi.com/2410-387X/7/2/29

[125] V. B. Dang, F. Farahmand, M. Andrzejczak, K. Mohajerani, D. T. Nguyen, and K. Gaj, "Implementation and benchmarking of round 2 candidates in the nist post-quantum cryptography standardization process using hardware and software/hardware co-design approaches," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 795, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:220249576

[126] IEEE, "Ieee standard specification for public key cryptographic techniques based on hard problems over lattices," *IEEE Std 1363.1-2008*, pp. 1–81, 2009.

[127] American National Standards Institute. (2010) Ansi x9.98-2010 - financial services - corporate treasury management. Accessed: September 29, 2023. [Online]. Available: https://webstore.ansi.org/standards/ascx9/ansix9982010

[128] National Institute of Standards and Technology, "Ntruencrypt submission," Available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/NTRUEncrypt.zip, accessed: September 29, 2023.

[129] ——, "Ntru-hrss-kem submission," Available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-1/submissions/NTRU_HRSS_KEM.zip, accessed: September 29, 2023.

[130] ——, "Ntru-round2 submission," Available at https://csrc.nist.gov/CSRC/media/Projects/Post-Quantum-Cryptography/documents/round-2/submissions/NTRU-Round2.zip, accessed: September 29, 2023.

[131] C. Chen, O. Danba, J. Hoffstein, A. Hülsing, J. Rijneveld, J. M. Schanck, T. Saito, P. Schwabe, W. Whyte, K. Xagawa, T. Yakamaka, and Z. Zhang, "NTRU: Algorithm Specifications And Supporting Documentation (version 3)," 2020.

[132] O. M. Guillen, T. Pöppelmann, J. M. Bermudo Mera, E. F. Bongenaar, G. Sigl, and J. Sepulveda, "Towards post-quantum security for iot endpoints with ntru," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, 2017, pp. 698–703.

[133] F. Farahmand, M. U. Sharif, K. Briggs, and K. Gaj, "A high-speed constant-time hardware implementation of ntruencrypt sves," in *2018 International Conference on Field-Programmable Technology (FPT)*, 2018, pp. 190–197.

[134] M. Imran, Z. U. Abideen, and S. Pagliarini, "An experimental study of building blocks of lattice-based nist post-quantum cryptographic algorithms," *Electronics*, vol. 9, no. 11, 2020. [Online]. Available: https://www.mdpi.com/2079-9292/9/11/1953

[135] F. Farahmand, D. T. Nguyen, V. B. Dang, A. Ferozpuri, and K. Gaj, "Software/hardware codesign of the post quantum cryptography algorithm ntruencrypt using high-level synthesis and register-transfer level design methodologies," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*, 2019, pp. 225–231.

[136] E. Camacho-Ruiz, M. C. Martínez-Rodríguez, S. Sánchez-Solano, and P. Brox, "Accelerating the development of ntru algorithm on embedded systems," in *2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS)*, 2020, pp. 1–6.

[137] E. Camacho-Ruiz, S. Sánchez-Solano, P. Brox, and M. C. Martínez-Rodríguez, "Timing-optimized hardware implementation to accelerate polynomial multiplication in the ntru algorithm," *J. Emerg. Technol. Comput. Syst.*, vol. 17, no. 3, may 2021. [Online]. Available: https://doi.org/10.1145/3445979

[138] S. Sánchez-Solano, E. Camacho-Ruiz, M. C. Martínez-Rodríguez, and P. Brox, "Multi-unit serial polynomial multiplier to accelerate ntru-based cryptographic schemes in iot embedded systems," *Sensors*, vol. 22, no. 5, 2022. [Online]. Available: https://www.mdpi.com/1424-8220/22/5/2057

[139] R. Primas, P. Pessl, and S. Mangard, "Single-trace side-channel attacks on masked lattice-based encryption," Cryptology ePrint Archive, Paper 2017/594, 2017, https://eprint.iacr.org/2017/594. [Online]. Available: https://eprint.iacr.org/2017/594

[140] A. Aysu, Y. Tobah, M. Tiwari, A. Gerstlauer, and M. Orshansky, "Horizontal side-channel vulnerabilities of post-quantum key exchange protocols," in *2018 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2018, pp. 81–88.

[141] E. Karimi, Y. Fei, and D. Kaeli, "Hardware/software obfuscation against timing side-channel attack on a gpu," in *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, 2020, pp. 122–131.

[142] National Institute of Standards and Technology, "Ntru-round3 submission," Available at https://csrc.nist.gov/CSRC/media/Projects/post-quantum-cryptography/documents/round-3/submissions/NTRU-Round3.zip, accessed: September 29, 2023.

[143] E. Camacho-Ruiz, S. Sánchez-Solano, M. C. Martínez-Rodriguez, and P. Brox, "NTRU Round 3 repository," https://gitlab.com/hwsec/ntru_3round.

[144] E. E. Targhi and D. Unruh, "Post-quantum security of the fujisaki-okamoto and oaep transforms," in *Theory of Cryptography*, M. Hirt and A. Smith, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 192–216.

[145] J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem," in *Algorithmic Number Theory*, J. P. Buhler, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 1998, pp. 267–288.

[146] V. B. Dang, K. Mohajerani, and K. Gaj, "High-speed hardware architectures and fpga benchmarking of crystals-kyber, ntru, and saber," *IEEE Transactions on Computers*, vol. 72, no. 2, pp. 306–320, 2023.

[147] D. V. Bailey, D. Coffin, A. Elbirt, J. H. Silverman, and A. D. Woodbury, "Ntru in constrained devices," in *Cryptographic Hardware and Embedded Systems — CHES 2001*, Ç. K. Koç, D. Naccache, and C. Paar, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 262–272.

[148] A. C. Atici, L. Batina, J. Fan, I. Verbauwhede, and S. Berna Ors Yalcin, "Low-cost implementations of ntru for pervasive security," in *2008 International Conference on Application-Specific Systems, Architectures and Processors*, 2008, pp. 79–84.

[149] A. A. Kamal and A. M. Youssef, "An fpga implementation of the ntruencrypt cryptosystem," in *2009 International Conference on Microelectronics - ICM*, 2009, pp. 209–212.

[150] X. Zhan, R. Zhang, Z. Xiong, Z. Zheng, and L. Zhenglin, "Efficient implementations of ntru in wireless network," *Communications and Network*, vol. 05, pp. 485–492, 01 2013.

[151] B. Liu and H. Wu, "Efficient architecture and implementation for ntruencrypt system," in *2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2015, pp. 1–4.

[152] ——, "Efficient multiplication architecture over truncated polynomial ring for ntruencrypt system," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 1174–1177.

[153] K. Braun, T. Fritzmann, G. Maringer, T. Schamberger, and J. Sepúlveda, "Secure and compact full ntru hardware implementation," in *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, 2018, pp. 89–94.

[154] Z. Qin, R. Tong, X. Wu, G. Bai, L. Wu, and L. Su, "A compact full hardware implementation of pqc algorithm ntru," in *2021 International Conference on Communications, Information System and Computer Engineering (CISCE)*, 2021, pp. 792–797.

[155] M. Azouaoui, Y. Kuzovkova, T. Schneider, and C. van Vredendaal, "Post-quantum authenticated encryption against chosen-ciphertext side-channel attacks," Cryptology ePrint Archive, Paper 2022/916, 2022, https://eprint.iacr.org/2022/916. [Online]. Available: https://eprint.iacr.org/2022/916

[156] J. Park, N. N. Anandakumar, D. Saha, D. Mehta, N. Pundir, F. Rahman, F. Farahmandi, and M. M. Tehranipoor, "Pqc-sep: Power side-channel evaluation platform for post-quantum cryptography algorithms," *IACR Cryptol. ePrint Arch.*, vol. 2022, p. 527, 2022.

[157] Y. Zhou and D. Feng, "Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing," Cryptology ePrint Archive, Paper 2005/388, 2005, https://eprint.iacr.org/2005/388. [Online]. Available: https://eprint.iacr.org/2005/388

[158] P. Ravi, A. Chattopadhyay, and S. Bhasin, "Practical side-channel and fault attacks on lattice-based cryptography," in *2021 IFIP/IEEE 29th International Conference on Very Large Scale Integration (VLSI-SoC)*, 2021, pp. 1–2.

[159] B. Liu and H. Wu, "Efficient multiplication architecture over truncated polynomial ring for ntruencrypt system," in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2016, pp. 1174–1177.

[160] P. Ravi, S. Sinha Roy, A. Chattopadhyay, and S. Bhasin, "Generic side-channel attacks on cca-secure lattice-based pke and kems," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2020, no. 3, p. 307–335, Jun. 2020. [Online]. Available: https://tches.iacr.org/index.php/TCHES/article/view/8592

[161] K. Ngo, E. Dubrova, Q. Guo, and T. Johansson, "A side-channel attack on a masked ind-cca secure saber kem," Cryptology ePrint Archive, Paper 2021/079, 2021, https://eprint.iacr.org/2021/079. [Online]. Available: https://eprint.iacr.org/2021/079

[162] A. Askeland and S. Rønjom, "A side-channel assisted attack on ntru," Cryptology ePrint Archive, Paper 2021/790, 2021, https://eprint.iacr.org/2021/790. [Online]. Available: https://eprint.iacr.org/2021/790

[163] B.-Y. Sim, J. Kwon, J. Lee, I.-J. Kim, T. Lee, J. Han, H. Yoon, J. Cho, and D.-G. Han, "Single-trace attacks on the message encoding of lattice-based kems," Cryptology ePrint Archive, Paper 2020/992, 2020, https://eprint.iacr.org/2020/992. [Online]. Available: https://eprint.iacr.org/2020/992

[164] H. Guntur, J. Ishii, and A. Satoh, "Side-channel attack user reference architecture board sakura-g," in *IEEE 3rd Global Conference on Consumer Electronics (GCCE'14)*. IEEE, 2014, pp. 271–274.

[165] L. F. Rojas-Muñoz, S. Sánchez-Solano, M. C. Martínez-Rodríguez, and P. Brox, "On-line evaluation and monitoring of security features of an ro-based puf/trng for iot devices," *Sensors*, vol. 23, no. 8, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/8/4070

[166] M. Bellare, R. Canetti, and H. Krawczyk, "Message authentication using hash functions—the hmac construction," *CryptoBytes*, vol. 2, 02 1998.

[167] N. I. of Standards and Technology, "Fips 198-1, the keyed-hash message authentication code (hmac)," https://csrc.nist.gov/pubs/fips/198-1/final, 2008, accessed on November 15, 2023.

[168] S. Kelly and S. Frankel, "Using hmac-sha-256, hmac-sha-384, and hmac-sha-512 with ipsec," https://datatracker.ietf.org/doc/html/rfc4868, 2007, accessed on November 15, 2023.

[169] K. Moriarty, B. Kaliski, and A. Rusch, "Pkcs #5: Password-based cryptography specification version 2.1," https://datatracker.ietf.org/doc/html/rfc8018, 2017, accessed on November 15, 2023.

[170] X. Wang, Y. L. Yin, and H. Yu, "Finding collisions in the full sha-1," in *Advances in Cryptology – CRYPTO 2005*, V. Shoup, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 17–36.

[171] X. Wang and H. Yu, "How to break md5 and other hash functions," in *Advances in Cryptology – EUROCRYPT 2005*, R. Cramer, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 19–35.

[172] E. Camacho-Ruiz, S. Sánchez-Solano, M. C. Martínez-Rodriguez, and P. Brox, "HMAC Repository of this dissertation," https://gitlab.com/hwsec/hmac_swhw.

Appendix A

Brief CV

Eros Camacho-Ruiz received the B. Sc. Degree in Physics from the University of Córdoba, Spain, in 2017, and the M. Sc. Degree in Microelectronics from the University of Seville, Spain, in 2020. Since 2020, he is a predoctoral researcher at Instituto de Microelectrónica de Sevilla (IMSE), Centro Nacional de Microelectrónica (CNM), CSIC / University of Seville. He has been funded by an FPU grant from the Spanish Government. His main research interest is the study and development of PUFs in analog and digital devices; and the design and implementation of Post-Quantum algorithms in embedded systems.

A.1 Journal Papers

[J1] E. Camacho-Ruiz, S. Sánchez-Solano, P. Brox, and M. C. Martínez-Rodríguez. "Timing-Optimized Hardware Implementation to Accelerate Polynomial Multiplication in the NTRU Algorithm". J. Emerg. Technol. Comput. Syst. 17, 3, Article 35, 2021.

[J2] M. C. Martínez-Rodríguez, E. Camacho-Ruiz, P. Brox, and S. Sánchez-Solano, "A Configurable RO-PUF for Securing Embedded Systems Implemented on Programmable Devices," Electronics, vol. 10, no. 16, p. 1957, Aug. 2021.

[J3] S. Sánchez-Solano, E. Camacho-Ruiz, M. C. Martínez-Rodríguez, and P. Brox, "Multi-Unit Serial Polynomial Multiplier to Accelerate NTRU-Based Cryptographic Schemes in IoT Embedded Systems," Sensors, vol. 22, no. 5, p. 2057, Mar. 2022.

[J4] M. C. Martínez-Rodríguez, L. F. Rojas-Muñoz, E. Camacho-Ruiz, S. Sánchez-Solano, and P. Brox, "Efficient RO-PUF for Generation of Identifiers and Keys in Resource-Constrained Embedded Systems," Cryptography, vol. 6, no. 4, p. 51, Oct. 2022.

[J5] E. Camacho-Ruiz, M. C. Martínez-Rodríguez, S. Sánchez-Solano, and P. Brox, "Timing-Attack-Resistant Acceleration of NTRU Round 3 Encryption on Resource-Constrained Embedded Systems," Cryptography, vol. 7, no. 2, p. 29, Jun. 2023.

A.2 Conference Papers

[C1] E. Camacho-Ruiz, M. C. Martínez-Rodríguez, S. Sánchez-Solano and P. Brox, "Accelerating the Development of NTRU Algorithm on Embedded Systems," 2020 XXXV Conference on Design of Circuits and Integrated Systems (DCIS), Segovia, Spain, 2020, pp. 1-6.

[C2] E. Camacho-Ruiz, P. Saraza-Canflanca, R. Castro-Lopez, E. Roca, P. Brox and F. V. Fernandez, "A study of SRAM PUFs reliability using the Static Noise Margin," SMACD / PRIME 2021; International Conference on SMACD and 16th Conference on PRIME, online, 2021, pp. 1-4.

[C3] P. Saraza-Canflanca et al., "Simulating the impact of Random Telegraph Noise on integrated circuits," SMACD / PRIME 2021; International Conference on SMACD and 16th Conference on PRIME, online, 2021, pp. 1-4.

[C4] M. C. Martínez-Rodríguez, E. Camacho-Ruiz, S. Sánchez-Solano and P. Brox, "Design Flow to Evaluate the Performance of Ring Oscillator PUFs on FPGAs," 2021 XXXVI Conference on Design of Circuits and Integrated Systems (DCIS), Vila do Conde, Portugal, 2021, pp. 1-6.

[C5] E. Camacho-Ruiz, R. Castro-Lopez, E. Roca, P. Brox and F. V. Fernandez, "A novel Physical Unclonable Function using RTN," 2022 IEEE International Symposium on Circuits and Systems (ISCAS), Austin, TX, USA, 2022, pp. 160-164.

[C6] E. Camacho-Ruiz, A. Santana-Andreo, R. Castro-Lopez, E. Roca and F. V. Fernandez, "On the use of an RTN simulator to explore the quality trade-offs of a novel RTN-based PUF," 2022 18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Villasimius, Italy, 2022, pp. 1-4.

[C7] E. Camacho-Ruiz, R. Castro-Lopez, E. Roca and F. V. Fernandez, "High-level design of a novel PUF based on RTN," 2022 18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Villasimius, Italy, 2022, pp. 1-4.

[C8] F. J. Rubio-Barbero, E. Camacho-Ruiz, R. Castro-Lopez, E. Roca and F. V. Fernandez, "A Peak Detect & Hold circuit to measure and exploit RTN in a 65-nm CMOS PUF," 2023 19th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Funchal, Portugal, 2023, pp. 1-4.

[C9] E. Camacho-Ruiz, F. J. Rubio-Barbero, R. Castro-Lopez, E. Roca and F. V. Fernandez, "Design considerations for a CMOS 65-nm RTN-based PUF," 2023 19th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design (SMACD), Funchal, Portugal, 2023, pp. 1-4.

[C10] E. Camacho-Ruiz, S. Sánchez-Solano, M. C. Martínez-Rodríguez and P. Brox, "A complete SHA-3 hardware library based on a high efficiency Keccak design," 2023 IEEE Nordic Circuits and Systems Conference (NorCAS), Aalborg, Denmark, 2023, pp. 1-7.

[C11] E. Camacho-Ruiz, S. Sánchez-Solano, M. C. Martínez-Rodríguez, E. Tena-Sanchez and P. Brox, "A Simple Power Analysis of an FPGA implementation of a polynomial multiplier for the NTRU cryptosystem," 2023 38th Conference on Design of Circuits and Integrated Systems (DCIS), Málaga, Spain, 2023, pp. 1-6.

A.3 Other merits

[O1] E. Camacho-Ruiz, R. Castro-Lopez, E. Roca, P. Brox, and F. V. Fernandez, "Method and device for physical unclonable function (puf) based on random telegraph noise (rtn)," Universidad de Sevilla (40 %), CSIC (60 %). PCT Patent PCT/EP2023/057 799, 2023.

[O2] Predoctoral stay at "Network and Information Security Group (NISEC)" of the Tampere University, Finland. 11 June 2023 - 8 September 2023. Tampere. Finland.

A.4 Projects

TOGETHER Towards Trusted Low-Power Things: Devices, Circuits and Architectures (TEC2016-75151-C3-3-R); PI: Dr. Francisco V. Fernández Fernández and Dr. Rafael Castro López; Supported by Ministerio de Economía, Industria y Competitividad; 2016 - 2021.

VIGILANT The Variability Challenge in Nano-CMOS: From device Modeling to IC Design for Mitigation and Exploitation (PID2019-103869RB-C31); PI: Dr. Francisco V. Fernández Fernández and Dr. Rafael Castro López; Supported by Ministerio de Ciencia e Innovación; 2020 - 2023.

SPIRS Secure Platform for ICT systems Rooted at the Silicon manufacturing process (GA: 952622); PI: Dr. Piedad Brox Jiménez; Supported by EU H2020 research and innovation programme, European Commission; 2021 - 2024.

QUBIP Quantum-oriented Update to Browsers and Infrastructures for the PQ Transitions (GA: 101119746); PI: Dr. Andrea Vesco; Supported by EU Horizon Europe research and innovation programme, European Commission; 2023 - 2026.

Appendix B

RTN-based PUF ASIC integration

B.1 Layout images

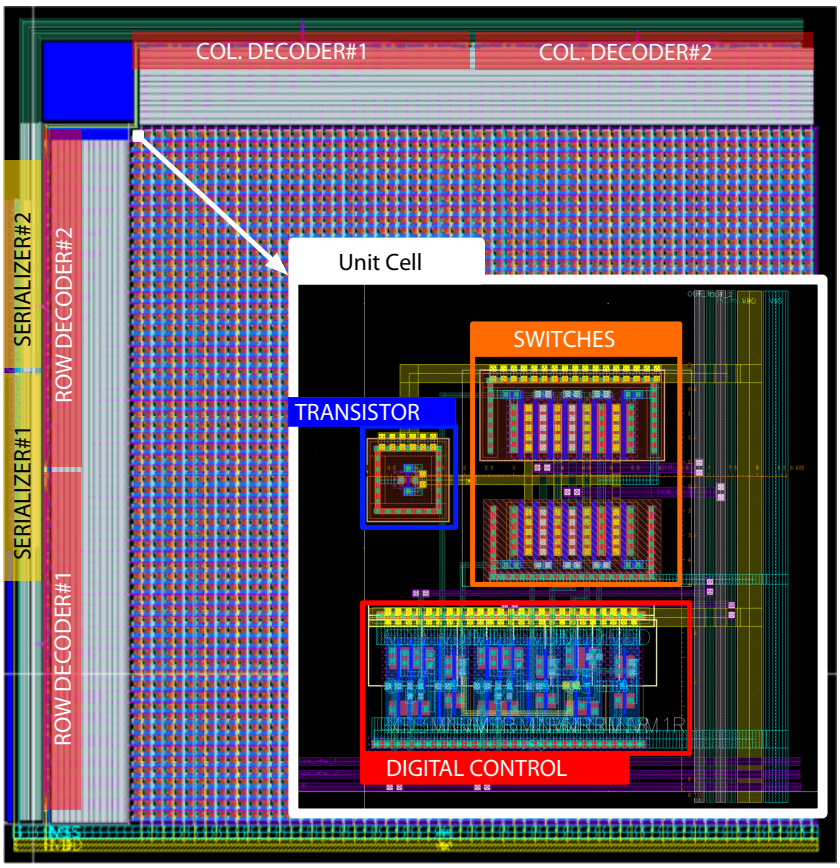


Figure B.1 Layout of the array of 4,096 transistors and inset showing the layout of the unit cell.

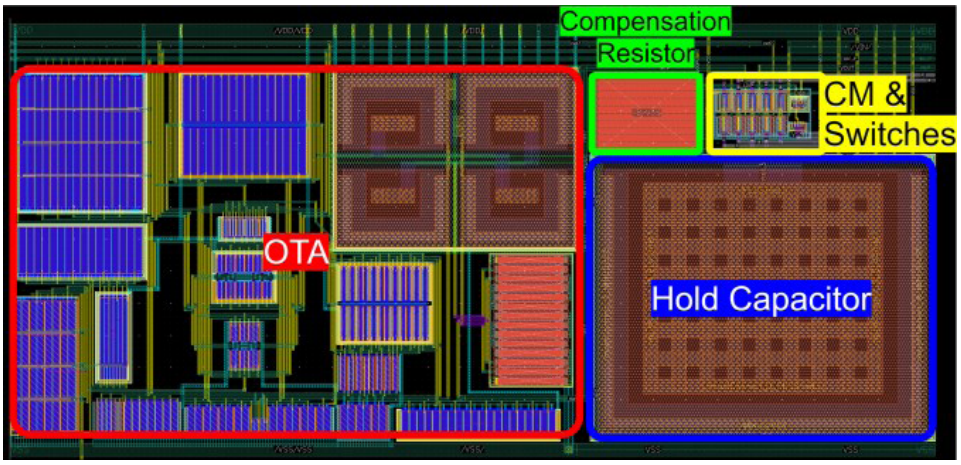


Figure B.2 Layout design of the PDH_{MAX} circuit.

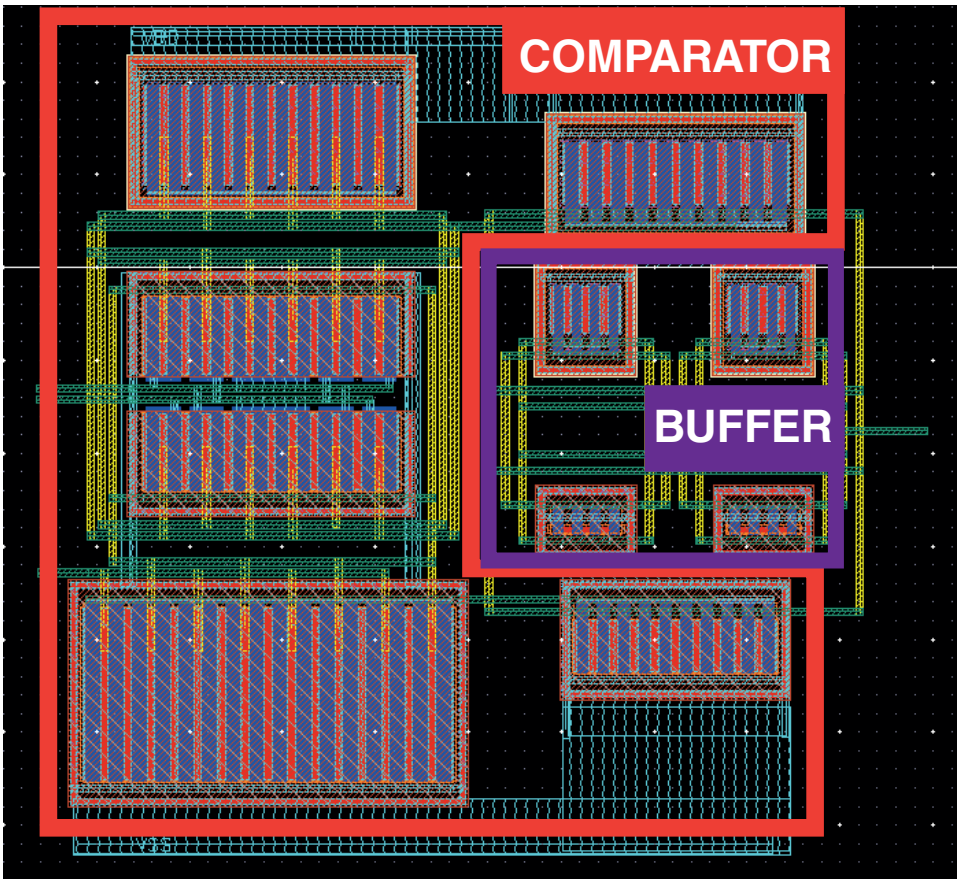


Figure B.3 Comparator final layout implementation.

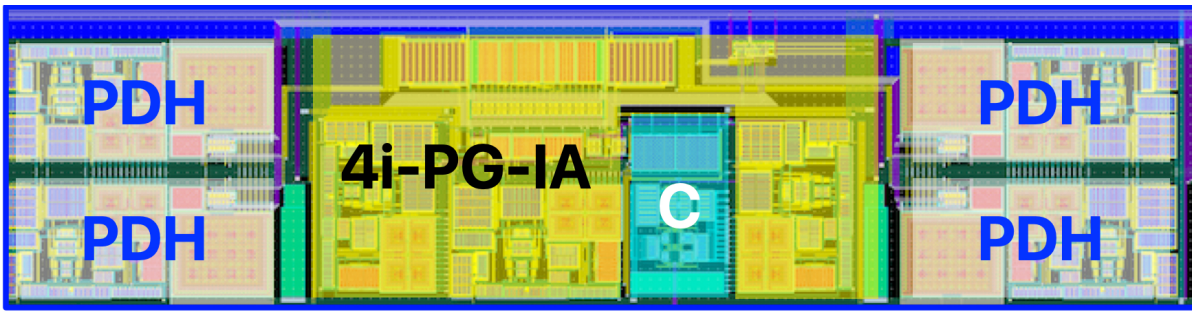


Figure B.4 Final layout of the ASB.

Appendix C

SHA2

C.1 Mathematical Equations

$$SHR^n(x) = x >> n \tag{C.1}$$

$$ROTR^n(x) = (x << (w-n) \mid (x >> n)) \text{ w-bit word} \tag{C.2}$$

$$\begin{aligned}
\sigma_0^{\{256\}}(x) &= ROTR^7(x) \oplus ROTR^{18}(x) \oplus SHR^3(x) && \text{(C.3)}\\
\sigma_1^{\{256\}}(x) &= ROTR^{17}(x) \oplus ROTR^{19}(x) \oplus SHR^{10}(x) && \text{(C.4)}\\
\sum_0^{\{256\}}(x) &= ROTR^2(x) \oplus ROTR^{13}(x) \oplus ROTR^{22}(x) && \text{(C.5)}\\
\sum_1^{\{256\}}(x) &= ROTR^6(x) \oplus ROTR^{11}(x) \oplus ROTR^{25}(x) && \text{(C.6)}
\end{aligned}$$

$$\begin{aligned}
\sigma_0^{\{512\}}(x) &= ROTR^1(x) \oplus ROTR^8(x) \oplus SHR^7(x) && \text{(C.7)}\\
\sigma_1^{\{512\}}(x) &= ROTR^{19}(x) \oplus ROTR^{61}(x) \oplus SHR^6(x) && \text{(C.8)}\\
\sum_0^{\{512\}}(x) &= ROTR^{28}(x) \oplus ROTR^{34}(x) \oplus ROTR^{39}(x) && \text{(C.9)}\\
\sum_1^{\{512\}}(x) &= ROTR^{14}(x) \oplus ROTR^{18}(x) \oplus ROTR^{41}(x) && \text{(C.10)}
\end{aligned}$$

$$\begin{aligned}
Ch(x,y,z) &= (x \,\&\, y) \oplus (!x \,\&\, z) && \text{(C.11)}\\
Maj(x,y,z) &= (x \,\&\, y) \oplus (x \,\&\, z) \oplus (y \,\&\, z) && \text{(C.12)}
\end{aligned}$$

C.2 SHA-2 Constants

Table C.1 SHA-224 and SHA-256 list of constants from upper-left to bottom-right

428a2f98	71374491	b5c0fbcf	e9b5dba5	3956c25b	59f111f1	923f82a4	ab1c5ed5
d807aa98	12835b01	243185be	550c7dc3	72be5d74	80deb1fe	9bdc06a7	c19bf174
e49b69c1	efbe4786	0fc19dc6	240ca1cc	2de92c6f	4a7484aa	5cb0a9dc	76f988da
983e5152	a831c66d	b00327c8	bf597fc7	c6e00bf3	d5a79147	06ca6351	14292967
27b70a85	2e1b2138	4d2c6dfc	53380d13	650a7354	766a0abb	81c2c92e	92722c85
a2bfe8a1	a81a664b	c24b8b70	c76c51a3	d192e819	d6990624	f40e3585	106aa070
19a4c116	1e376c08	2748774c	34b0bcb5	391c0cb3	4ed8aa4a	5b9cca4f	682e6ff3
748f82ee	78a5636f	84c87814	8cc70208	90befffa	a4506ceb	bef9a3f7	c67178f2

Table C.2 SHA-384, SHA-512, SHA-512/224 and SHA-512/256 list of constants from upper-left to bottom-right

428a2f98d728ae22	7137449123ef65cd	b5c0fbcfec4d3b2f	e9b5dba58189dbbc
3956c25bf348b538	59f111f1b605d019	923f82a4af194f9b	ab1c5ed5da6d8118
d807aa98a3030242	12835b0145706fbe	243185be4ee4b28c	550c7dc3d5ffb4e2
72be5d74f27b896f	80deb1fe3b1696b1	9bdc06a725c71235	c19bf174cf692694
e49b69c19ef14ad2	efbe4786384f25e3	0fc19dc68b8cd5b5	240ca1cc77ac9c65
2de92c6f592b0275	4a7484aa6ea6e483	5cb0a9dcbd41fbd4	76f988da831153b5
983e5152ee66dfab	a831c66d2db43210	b00327c898fb213f	bf597fc7beef0ee4
c6e00bf33da88fc2	d5a79147930aa725	06ca6351e003826f	142929670a0e6e70
27b70a8546d22ffc	2e1b21385c26c926	4d2c6dfc5ac42aed	53380d139d95b3df
650a73548baf63de	766a0abb3c77b2a8	81c2c92e47edaee6	92722c851482353b
a2bfe8a14cf10364	a81a664bbc423001	c24b8b70d0f89791	c76c51a30654be30
d192e819d6ef5218	d69906245565a910	f40e35855771202a	106aa07032bbd1b8
19a4c116b8d2d0c8	1e376c085141ab53	2748774cdf8eeb99	34b0bcb5e19b48a8
391c0cb3c5c95a63	4ed8aa4ae3418acb	5b9cca4f7763e373	682e6ff3d6b2b8a3
748f82ee5defb2fc	78a5636f43172f60	84c87814a1f0ab72	8cc702081a6439ec
90befffa23631e28	a4506cebde82bde9	bef9a3f7b2c67915	c67178f2e372532b
ca273eceea26619c	d186b8c721c0c207	eada7dd6cde0eb1e	f57d4f7fee6ed178
06f067aa72176fba	0a637dc5a2c898a6	113f9804bef90dae	1b710b35131c471b
28db77f523047d84	32caab7b40c72493	3c9ebe0a15c9bebc	431d67c49c100d4c
4cc5d4becb3e42b6	597f299cfc657e2a	5fcb6fab3ad6faec	6c44198c4a475817

C.3 SHA-2 Initial Values

Table C.3 SHA-224 and SHA-256 initial hash values

Initial values	SHA-224	SHA-256
H_0^0	c1059ed8	6a09e667
H_1^0	367cd507	bb67ae85
H_2^0	3070dd17	3c6ef372
H_3^0	f70e5939	a54ff53a
H_4^0	ffc00b31	510e527f
H_5^0	68581511	9b05688c
H_6^0	64f98fa7	1f83d9ab
H_7^0	befa4fa4	5be0cd19

Table C.4 SHA-384 and SHA-512 initial hash values

Initial values	SHA-384	SHA-512
H_0^0	cbbb9d5dc1059ed8	6a09e667f3bcc908
H_1^0	629a292a367cd507	bb67ae8584caa73b
H_2^0	9159015a3070dd17	3c6ef372fe94f82b
H_3^0	152fecd8f70e5939	a54ff53a5f1d36f1
H_4^0	67332667ffc00b31	510e527fade682d1
H_5^0	8eb44a8768581511	9b05688c2b3e6c1f
H_6^0	db0c2e0d64f98fa7	1f83d9abfb41bd6b
H_7^0	47b5481dbefa4fa4	5be0cd19137e2179

Table C.5 SHA-512/224 and SHA-512/256 initial hash values

Initial values	SHA-512/224	SHA-512/256
H_0^0	8C3D37C819544DA2	22312194FC2BF72C
H_1^0	73E1996689DCD4D6	9F555FA3C84C64C2
H_2^0	1DFAB7AE32FF9C82	2393B86B6F53B151
H_3^0	679DD514582F9FCF	963877195940EABD
H_4^0	0F6D2B697BD44DA8	96283EE2A88EFFE3
H_5^0	77E36F7304C48942	BE5E1E2553863992
H_6^0	3F9D85A86A1D36C8	2B0199FC2C85B8AA
H_7^0	1112E6AD91D692A1	0EB72DDC81C52CA2

Appendix D

NTRU

D.1 IP module resource occupation and timing performance results

Table D.1 IP module resource occupation and timing performance results for $N = 509$ and $max_{coef} = 400$.

M	*LUTs*	*FFs*	*BRAM*	*Clk (MHz)*	CC_{mult}	*Latency* (μs)
1	147	96	1.5	96.01	203709	2121.83
2	207	121	2.5	97.46	102109	1047.74
3	257	125	3.5	96.22	68109	707.86
4	285	119	4.5	97.68	51309	525.25
5	336	124	5.5	94.61	40909	432.41
6	381	123	6.5	98.03	34109	347.95
7	437	124	7.5	96.62	29309	303.35
8	584	205	4.5	97.00	25709	265.03
9	634	222	5	94.98	22909	241.21
10	704	232	5.5	97.24	20509	210.91
11	767	244	6	91.78	18909	206.03
12	810	253	6.5	94.36	17309	183.44
13	892	266	7	94.30	16109	170.82
14	946	276	7.5	94.02	14909	158.57
15	997	288	8	91.79	13709	149.36
16	921	291	8.5	93.99	12909	137.34
32	1747	472	16.5	92.45	6509	70.41
64	4115	828	32.5	87.36	3309	37.88
128	7951	1546	64.5	79.18	1709	21.58
256	16159	3070	128	77.86	909	11.68

Table D.2 IP module resource occupation and timing performance results for $N = 509$ and $max_{coef} = 509$.

M	*LUTs*	*FFs*	*BRAM*	*Clk (MHz)*	CC_{mult}	*Latency* (μs)
1	119	78	1.5	101.29	259081	2557.64
2	176	103	2.5	98.46	129795	1318.19
3	228	107	3.5	99.59	86530	868.84
4	260	101	4.5	99.22	65152	656.60
5	316	106	5.5	93.43	51918	555.67
6	348	105	6.5	96.49	43265	448.35
7	399	106	7.5	96.51	37157	384.98
8	556	187	4.5	97.94	32576	332.60
9	603	204	5	95.31	29013	304.37
10	678	214	5.5	95.45	25959	271.94
11	739	226	6	92.61	23923	258.29
12	785	235	6.5	94.31	21887	232.06
13	872	248	7	94.28	20360	215.93
14	923	258	7.5	94.63	18833	199.00
15	971	270	8	92.18	17306	187.73
16	897	273	8.5	94.53	16288	172.29
32	1721	454	16.5	90.27	8144	90.21
64	4088	816	32.5	90.08	4072	45.20
128	8097	1540	64.5	83.78	2036	24.29
256	16707	2992	128.5	78.86	1018	12.90

Table D.3 IP module resource occupation and timing performance results for $N = 677$ and $max_{coef} = 516$.

M	*LUTs*	*FFs*	*BRAM*	*Clk (MHz)*	$\mathbf{CC_{mult}}$	*Latency* (μs)
1	167	104	1.5	97.03	349493	3601.87
2	238	132	2.5	95.76	175085	1828.41
3	272	132	3.5	95.28	116777	1225.57
4	309	130	4.5	95.00	87881	925.04
5	364	135	5.5	93.85	70337	749.44
6	398	130	6.5	95.50	58469	612.23
7	444	131	7.5	93.34	50213	537.93
8	476	128	8.5	94.30	44021	466.84
9	531	134	9.5	93.36	39377	421.77
10	582	133	10.5	93.21	35249	378.15
11	801	251	6	93.34	32153	344.49
12	851	260	6.5	94.10	29573	314.27
13	935	273	7	91.68	27509	300.04
14	988	283	7.5	91.85	25445	277.02
15	1037	295	8	91.11	23897	262.29
16	1103	302	8.5	93.70	22349	238.51
32	1823	484	16.5	90.25	11513	127.56
64	4205	846	32.5	82.31	5837	70.91
128	8206	1562	64.5	81.65	3257	39.89
256	17454	3044	128.5	75.15	1709	22.74

Table D.4 IP module resource occupation and timing performance results for $N = 677$ and $max_{coef} = 677$.

M	*LUTs*	*FFs*	*BRAM*	*Clk (MHz)*	CC_{mult}	*Latency* (μs)
1	137	84	1.5	96.85	349493	3608.52
2	198	112	2.5	97.71	175085	1791.82
3	241	112	3.5	97.04	116777	1203.39
4	291	110	4.5	94.66	87881	928.37
5	332	115	5.5	93.74	70337	750.36
6	366	110	6.5	93.90	58469	622.69
7	419	111	7.5	92.76	50213	541.30
8	451	108	8.5	94.73	44021	464.69
9	500	114	9.5	91.50	39377	430.35
10	558	113	10.5	93.32	35249	377.73
11	773	231	6	91.18	32153	352.62
12	824	240	6.5	90.93	29573	325.21
13	906	253	7	91.82	27509	299.60
14	959	263	7.5	91.64	25445	277.66
15	1025	275	8	93.34	23897	256.03
16	1068	282	8.5	91.97	22349	243.00
32	1787	464	16.5	91.52	11513	125.80
64	4176	826	32.5	84.92	5837	68.74
128	8356	1552	64.5	78.34	3257	41.58
256	17389	3039	128.5	74.81	1709	22.85

Table D.5 IP module resource occupation and timing performance results for $N = 821$ and $max_{coef} = 625$.

M	***LUTs***	***FFs***	***BRAM***	***Clk (MHz)***	$\mathbf{CC_{mult}}$	***Latency (μs)***
1	166	105	1.5	96.36	349493	3627.04
2	237	133	2.5	99.40	175085	1761.36
3	290	137	3.5	97.24	116777	1200.93
4	326	131	4.5	94.64	87881	928.55
5	373	136	5.5	97.41	70337	722.08
6	419	135	6.5	94.11	58469	621.29
7	472	132	7.5	93.08	50213	539.49
8	510	129	8.5	94.70	44021	464.86
9	562	135	9.5	95.04	39377	414.32
10	612	134	10.5	94.40	35249	373.39
11	656	135	11.5	92.88	32153	346.16
12	713	133	12.5	93.42	29573	316.55
13	981	287	7	92.66	27509	296.88
14	1033	298	7.5	93.95	25445	270.84
15	1087	311	8	93.34	23897	256.03
16	1159	319	8.5	92.28	22349	242.17
32	1919	518	16.5	91.98	11513	125.17
64	4458	914	32.5	86.89	5837	67.17
128	8725	1697	64.5	84.04	3257	38.76
256	18091	3283	128.5	77.63	1709	22.02

Table D.6 IP module resource occupation and timing performance results for $N = 821$ and $max_{coef} = 821$.

M	*LUTs*	*FFs*	*BRAM*	*Clk (MHz)*	$\mathbf{CC_{mult}}$	*Latency* (μs)
1	150	85	1.5	98.61	349493	3544.21
2	207	113	2.5	98.37	175085	1779.91
3	260	117	3.5	95.54	116777	1222.30
4	306	111	4.5	98.51	87881	892.08
5	347	116	5.5	95.62	70337	735.58
6	390	115	6.5	93.35	58469	626.32
7	442	112	7.5	93.03	50213	539.74
8	481	109	8.5	96.70	44021	455.22
9	532	115	9.5	93.19	39377	422.55
10	590	114	10.5	93.22	35249	378.12
11	640	115	11.5	91.72	32153	350.56
12	679	113	12.5	92.78	29573	318.74
13	929	267	7	91.79	27509	299.68
14	1005	278	7.5	92.20	25445	275.98
15	1064	291	8	92.94	23897	257.13
16	1134	299	8.5	93.99	22349	237.77
32	1893	497	16.5	92.23	11513	124.82
64	4418	892	32.5	85.21	5837	68.50
128	8640	1660	64.5	80.59	3257	40.41
256	18233	3277	128.5	77.91	1709	21.94

Table D.7 IP module resource occupation and timing performance results for $N = 701$ and $max_{coef} = 533$.

M	***LUTs***	***FFs***	***BRAM***	***Clk (MHz)***	$\mathbf{CC_{mult}}$	***Latency (μs)***
1	176	104	1.5	96.64	349493	3616.55
2	255	132	2.5	96.81	175085	1808.63
3	293	132	3.5	97.02	116777	1203.62
4	330	130	4.5	97.37	87881	902.54
5	390	135	5.5	93.06	70337	755.84
6	433	130	6.5	93.40	58469	626.03
7	487	131	7.5	93.08	50213	539.49
8	529	128	8.5	94.30	44021	466.80
9	581	134	9.5	93.17	39377	422.63
10	634	133	10.5	93.99	35249	375.01
11	881	273	6	93.41	32153	344.23
12	932	284	6.5	91.95	29573	321.61
13	1025	299	7	92.13	27509	298.58
14	1089	311	7.5	93.05	25445	273.46
15	1159	325	8	91.22	23897	261.98
16	1225	334	8.5	91.95	22349	243.05
32	2068	550	16.5	90.91	11513	126.64
64	4708	970	32.5	84.88	5837	68.77
128	9441	1830	64.5	80.75	3257	40.33
256	19587	3562	128.5	74.77	1709	22.86

Table D.8 IP module resource occupation and timing performance results for $N = 701$ and $max_{coef} = 701$.

M	*LUTs*	*FFs*	*BRAM*	*Clk (MHz)*	$\mathbf{CC_{mult}}$	*Latency* (μs)
1	148	84	1.5	94.97	349493	3680.16
2	222	112	2.5	97.94	175085	1787.62
3	261	112	3.5	95.15	116777	1227.33
4	298	110	4.5	94.98	87881	925.21
5	367	115	5.5	92.28	70337	762.24
6	403	110	6.5	95.95	58469	609.36
7	462	111	7.5	94.79	50213	529.75
8	506	108	8.5	92.83	44021	474.19
9	553	114	9.5	92.27	39377	426.77
10	605	113	10.5	91.87	35249	383.69
11	854	253	6	93.89	32153	342.46
12	921	264	6.5	92.44	29573	319.92
13	1009	279	7	92.47	27509	297.48
14	1076	291	7.5	88.95	25445	286.05
15	1138	305	8	92.06	23897	259.59
16	1205	314	8.5	92.14	22349	242.55
32	2070	529	16.5	89.97	11513	127.97
64	4694	955	32.5	84.27	5837	69.27
128	9430	1805	64.5	77.66	3257	41.94
256	19643	3588	128.5	72.49	1709	23.58

D.2 Multiplication, encryption and decryption acceleration using the hardware implementation with respect to the time required for the software results

Table D.9 Multiplication, encryption and decryption acceleration using the hardware implementation with respect to the time required for the software results for $N = 509$ and $max_{coef} = 400$.

	Multiplication			***Encryption***			***Decryption***		
M	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**
1		2218	**6.47**		2346	**6.16**		32564	**1.37**
2		1201	**11.92**		1327	**10.88**		31512	**1.41**
3		861	**16.63**		987	**14.62**		31173	**1.43**
4		693	**20.67**		818	**17.64**		30999	**1.44**
5		589	**24.32**		714	**20.20**		30896	**1.44**
6		521	**27.55**		646	**22.37**		30827	**1.45**
7		473	**30.28**		600	**24.06**		30781	**1.45**
8		436	**32.81**		563	**25.62**		30749	**1.45**
9		409	**35.04**		535	**26.98**		30721	**1.45**
10	14337	385	**37.16**	14451	511	**28.23**	44638	30699	**1.45**
11		369	**38.82**		494	**29.19**		30672	**1.45**
12		353	**40.57**		479	**30.13**		30662	**1.45**
13		340	**42.07**		467	**30.93**		30650	**1.45**
14		329	**43.58**		456	**31.67**		30635	**1.45**
15		317	**45.14**		443	**32.54**		30627	**1.46**
16		309	**46.45**		436	**33.14**		30616	**1.46**
32		244	**58.71**		369	**39.06**		30545	**1.46**
64		212	**67.47**		340	**42.42**		30517	**1.46**
128		196	**73.11**		321	**44.89**		30494	**1.46**
256		189	**75.88**		315	**45.87**		30495	**1.46**

Table D.10 Multiplication, encryption and decryption acceleration using the hardware implementation with respect to the time required for the software results for $N = 509$ and $max_{coef} = 509$.

	Multiplication			***Encryption***			***Decryption***		
M	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**
1	14337	2770	**5.17**	14451	2896	**4.98**	44638	33074	**1.35**
2		1478	**9.69**		1604	**9.00**		31786	**1.40**
3		1045	**13.71**		1171	**12.32**		31349	**1.42**
4		830	**17.24**		956	**15.09**		31132	**1.43**
5		698	**20.50**		824	**17.51**		31001	**1.44**
6		612	**23.40**		738	**19.55**		30919	**1.44**
7		551	**26.00**		677	**21.30**		30855	**1.44**
8		505	**28.35**		631	**22.88**		30812	**1.45**
9		469	**30.50**		595	**24.25**		30777	**1.45**
10		439	**32.63**		565	**25.54**		30747	**1.45**
11		419	**34.21**		544	**26.50**		30727	**1.45**
12		398	**35.96**		524	**27.55**		30702	**1.45**
13		383	**37.35**		509	**28.34**		30690	**1.45**
14		368	**38.92**		494	**29.19**		30674	**1.45**
15		352	**40.62**		478	**30.19**		30660	**1.45**
16		342	**41.87**		468	**30.83**		30650	**1.45**
32		261	**54.89**		387	**37.28**		30567	**1.46**
64		221	**64.92**		346	**41.67**		30529	**1.46**
128		200	**71.70**		325	**44.36**		30507	**1.46**
256		189	**75.64**		315	**45.82**		30494	**1.46**

Table D.11 Multiplication, encryption and decryption acceleration using the hardware implementation with respect to the time required for the software results for $N = 607$ and $max_{coef} = 516$.

	Multiplication			***Encryption***			***Decryption***		
M	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**
1		3730	**6.79**		3904	**6.52**		56513	**1.38**
2		1986	**12.74**		2160	**11.78**		54759	**1.42**
3		1403	**18.04**		1576	**16.15**		54180	**1.44**
4		1114	**22.72**		1287	**19.78**		53890	**1.45**
5		939	**26.96**		1113	**22.88**		53712	**1.45**
6		820	**30.88**		993	**25.63**		53594	**1.46**
7		737	**34.32**		911	**27.93**		53512	**1.46**
8		676	**37.46**		849	**29.98**		53446	**1.46**
9		629	**40.26**		802	**31.73**		53403	**1.46**
10	25320	588	**43.08**	25471	761	**33.46**	78046	53364	**1.46**
11		556	**45.49**		730	**34.86**		53330	**1.46**
12		531	**47.65**		705	**36.12**		53306	**1.46**
13		510	**49.61**		684	**37.22**		53284	**1.46**
14		490	**51.67**		663	**38.37**		53259	**1.46**
15		474	**53.38**		648	**39.31**		53249	**1.47**
16		459	**55.18**		632	**40.27**		53230	**1.47**
32		350	**72.21**		524	**48.57**		53123	**1.47**
64		294	**86.11**		468	**54.45**		53069	**1.47**
128		268	**94.60**		441	**57.70**		53040	**1.47**
256		253	**100.14**		426	**59.73**		53043	**1.47**

Table D.12 Multiplication, encryption and decryption acceleration using the hardware implementation with respect to the time required for the software results for $N = 607$ and $max_{coef} = 607$.

	Multiplication			***Encryption***			***Decryption***		
M	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**
1		4819	**5.25**		4993	**5.10**		57615	**1.35**
2		2530	**10.01**		2704	**9.42**		55335	**1.41**
3		1765	**14.34**		1939	**13.13**		54546	**1.43**
4		1387	**18.25**		1561	**16.31**		54172	**1.44**
5		1156	**21.89**		1330	**19.14**		53939	**1.45**
6		1002	**25.30**		1176	**21.68**		53836	**1.45**
7		893	**28.34**		1067	**23.86**		53701	**1.45**
8		811	**31.23**		985	**25.87**		53600	**1.46**
9		750	**33.76**		924	**27.57**		53594	**1.46**
10	25320	695	**36.40**	25471	869	**29.29**	78046	53478	**1.46**
11		655	**38.64**		829	**30.72**		53436	**1.46**
12		621	**40.74**		795	**32.02**		53408	**1.46**
13		594	**42.62**		768	**33.17**		53377	**1.46**
14		567	**44.67**		740	**34.40**		53351	**1.46**
15		546	**46.33**		720	**35.35**		53332	**1.46**
16		526	**48.12**		700	**36.37**		53339	**1.46**
32		384	**65.93**		558	**45.69**		53173	**1.47**
64		309	**81.84**		483	**52.72**		53090	**1.47**
128		276	**91.82**		449	**56.65**		53064	**1.47**
256		255	**99.12**		429	**59.35**		53041	**1.47**

Table D.13 Multiplication, encryption and decryption acceleration using the hardware implementation with respect to the time required for the software results for $N = 821$ and $max_{coef} = 625$.

	Multiplication			***Encryption***			***Decryption***		
M	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**
1		5417	**6.87**		5618	**6.65**		82460	**1.38**
2		2855	**13.03**		3055	**12.23**		79876	**1.43**
3		1998	**18.62**		2198	**17.00**		79025	**1.44**
4		1572	**23.66**		1773	**21.08**		78592	**1.45**
5		1317	**28.25**		1517	**24.64**		78337	**1.46**
6		1142	**32.57**		1343	**27.83**		78171	**1.46**
7		1022	**36.39**		1222	**30.57**		78048	**1.46**
8		929	**40.06**		1130	**33.09**		77985	**1.46**
9		861	**43.23**		1061	**35.23**		77888	**1.47**
10	37214	805	**46.24**	37385	1005	**37.20**	114180	77835	**1.47**
11		754	**49.33**		955	**39.15**		77774	**1.47**
12		717	**51.89**		917	**40.75**		77745	**1.47**
13		686	**54.25**		886	**42.19**		77715	**1.47**
14		654	**56.89**		855	**43.73**		77682	**1.47**
15		629	**59.13**		830	**45.05**		77657	**1.47**
16		611	**60.85**		812	**46.02**		77639	**1.47**
32		448	**83.00**		648	**57.64**		77477	**1.47**
64		367	**101.38**		567	**65.89**		77420	**1.47**
128		329	**112.96**		530	**70.53**		77356	**1.48**
256		310	**119.85**		511	**73.18**		77349	**1.48**

Table D.14 Multiplication, encryption and decryption acceleration using the hardware implementation with respect to the time required for the software results for $N = 821$ and $max_{coef} = 821$.

	Multiplication			***Encryption***			***Decryption***		
M	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**
1		7025	**5.30**		7224	**5.17**		84056	**1.36**
2		3658	**10.17**		3858	**9.69**		80692	**1.42**
3		2534	**14.68**		2734	**13.67**		79632	**1.43**
4		1975	**18.84**		2175	**17.19**		79007	**1.44**
5		1638	**22.71**		1837	**20.34**		78663	**1.45**
6		1409	**26.41**		1608	**23.24**		78443	**1.46**
7		1252	**29.71**		1452	**25.74**		78287	**1.46**
8		1130	**32.94**		1329	**28.13**		78163	**1.46**
9		1039	**35.80**		1239	**30.16**		78070	**1.46**
10	37214	965	**38.56**	37385	1165	**32.09**	114180	77989	**1.46**
11		900	**41.35**		1100	**33.99**		77933	**1.46**
12		851	**43.74**		1050	**35.59**		77888	**1.47**
13		809	**45.99**		1009	**37.06**		77847	**1.47**
14		768	**48.47**		967	**38.64**		77798	**1.47**
15		736	**50.56**		935	**39.96**		77781	**1.47**
16		711	**52.35**		910	**41.06**		77766	**1.47**
32		497	**74.80**		698	**53.56**		77517	**1.47**
64		390	**95.39**		591	**63.29**		77416	**1.47**
128		341	**109.09**		541	**69.07**		77368	**1.48**
256		316	**117.55**		517	**72.33**		77344	**1.48**

Table D.15 Multiplication, encryption and decryption acceleration using the hardware implementation with respect to the time required for the software results for $N = 701$ and $max_{coef} = 533$.

	Multiplication			***Encryption***			***Decryption***		
M	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**
1		3981	**6.82**		4509	**6.13**		60834	**1.38**
2		2116	**12.82**		2642	**10.46**		58965	**1.42**
3		1492	**18.18**		2018	**13.69**		58342	**1.44**
4		1183	**22.94**		1709	**16.17**		58052	**1.45**
5		997	**27.23**		1524	**18.14**		57865	**1.45**
6		868	**31.26**		1394	**19.82**		57725	**1.45**
7		783	**34.67**		1309	**21.11**		57635	**1.46**
8		714	**38.02**		1240	**22.28**		57566	**1.46**
9		661	**41.07**		1187	**23.27**		57511	**1.46**
10	27134	623	**43.54**	27633	1150	**24.03**	83897	57472	**1.46**
11		586	**46.32**		1112	**24.85**		57433	**1.46**
12		559	**48.51**		1086	**25.44**		57408	**1.46**
13		533	**50.88**		1060	**26.07**		57398	**1.46**
14		518	**52.43**		1045	**26.44**		57372	**1.46**
15		496	**54.73**		1022	**27.03**		57350	**1.46**
16		480	**56.58**		1006	**27.45**		57329	**1.46**
32		362	**74.94**		889	**31.09**		57212	**1.47**
64		304	**89.34**		830	**33.27**		57150	**1.47**
128		277	**98.05**		803	**34.39**		57126	**1.47**
256		260	**104.25**		787	**35.09**		57113	**1.47**

Table D.16 Multiplication, encryption and decryption acceleration using the hardware implementation with respect to the time required for the software results for $N = 701$ and $max_{coef} = 701$.

	Multiplication			***Encryption***			***Decryption***		
M	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**	**SW (*us*)**	**HW (*us*)**	**Acc. (x)**
1		5157	**5.26**		5685	**4.86**		62009	**1.35**
2		2703	**10.04**		3231	**8.55**		59568	**1.41**
3		1883	**14.41**		2411	**11.46**		58729	**1.43**
4		1477	**18.37**		2004	**13.78**		58319	**1.44**
5		1231	**22.04**		1758	**15.71**		58078	**1.44**
6		1063	**25.52**		1590	**17.38**		57908	**1.45**
7		951	**28.53**		1478	**18.69**		57796	**1.45**
8		860	**31.54**		1388	**19.92**		57722	**1.45**
9		790	**34.36**		1317	**20.98**		57634	**1.46**
10	27134	741	**36.62**	27633	1268	**21.78**	83897	57587	**1.46**
11		692	**39.18**		1220	**22.64**		57534	**1.46**
12		657	**41.29**		1184	**23.33**		57500	**1.46**
13		623	**43.58**		1150	**24.02**		57464	**1.46**
14		601	**45.16**		1129	**24.49**		57468	**1.46**
15		573	**47.37**		1100	**25.13**		57438	**1.46**
16		551	**49.22**		1078	**25.62**		57398	**1.46**
32		397	**68.28**		924	**29.90**		57270	**1.47**
64		320	**84.80**		847	**32.63**		57167	**1.47**
128		285	**95.16**		813	**33.98**		57136	**1.47**
256		264	**102.80**		791	**34.92**		57127	**1.47**

D.3 Optimizing area and acceleration results

Table D.17 Multiplication, encryption and decryption efficiency of each resource for $N = 509$ and $max_{coef} = 400$.

	Multiplication			***Encryption***			***Decryption***		
M	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**
1	0.044	0.067	4.310	0.042	0.064	4.106	0.009	0.014	0.914
2	0.058	0.099	4.769	0.053	0.090	4.351	0.007	0.012	0.566
3	0.065	0.133	4.752	0.057	0.117	4.177	0.006	0.011	0.408
4	**0.073**	**0.174**	**4.594**	**0.062**	**0.148**	**3.920**	**0.005**	**0.012**	**0.319**
5	0.072	0.196	4.421	0.060	0.163	3.673	0.004	0.012	0.262
6	0.072	0.224	4.238	0.059	0.182	3.442	0.004	0.012	0.222
7	**0.069**	**0.244**	**4.038**	**0.055**	**0.194**	**3.207**	**0.003**	**0.012**	**0.193**
8	**0.056**	**0.160**	**7.290**	**0.044**	**0.125**	**5.693**	**0.002**	**0.007**	**0.322**
9	0.055	0.158	7.008	0.043	0.122	5.396	0.002	0.007	0.290
10	0.053	0.160	6.757	0.040	0.122	5.133	0.002	0.006	0.264
11	0.051	0.159	6.470	0.038	0.120	4.866	0.002	0.006	0.242
12	0.050	0.160	6.242	0.037	0.119	4.635	0.002	0.006	0.224
13	0.047	0.158	6.010	0.035	0.116	4.419	0.002	0.005	0.208
14	0.046	0.158	5.810	0.033	0.115	4.223	0.002	0.005	0.194
15	0.045	0.157	5.643	0.033	0.113	4.068	0.001	0.005	0.182
16	0.050	0.160	5.464	0.036	0.114	3.899	0.002	0.005	0.171
32	0.034	0.124	3.558	0.022	0.083	2.367	0.001	0.003	0.088
64	0.016	0.081	2.076	0.010	0.051	1.305	0.000	0.002	0.045
128	0.009	0.047	1.134	0.006	0.029	0.696	0.000	0.001	0.023
256	0.005	0.025	0.593	0.003	0.015	0.358	0.000	0.000	0.011

Table D.18 Multiplication, encryption and decryption efficiency of each resource for $N = 509$ and $max_{coef} = 509$.

	Multiplication			*Encryption*			*Decryption*		
M	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**
1	0.043	0.066	3.446	0.042	0.064	3.322	0.011	0.017	0.898
2	0.055	0.094	3.876	0.051	0.087	3.600	0.008	0.014	0.561
3	0.060	0.128	3.916	0.054	0.115	3.521	0.006	0.013	0.406
4	0.066	0.171	3.830	0.058	0.149	3.353	0.006	0.014	0.318
5	0.065	0.193	3.727	0.055	0.165	3.183	0.005	0.014	0.261
6	**0.067**	**0.223**	**3.599**	**0.056**	**0.186**	**3.008**	**0.004**	**0.014**	**0.222**
7	**0.065**	**0.245**	**3.467**	**0.053**	**0.201**	**2.841**	**0.004**	**0.014**	**0.193**
8	**0.051**	**0.152**	**6.299**	**0.041**	**0.122**	**5.084**	**0.003**	**0.008**	**0.321**
9	0.051	0.150	6.101	0.040	0.119	4.851	0.002	0.007	0.290
10	0.048	0.152	5.933	0.038	0.119	4.644	0.002	0.007	0.264
11	0.046	0.151	5.701	0.036	0.117	4.417	0.002	0.006	0.242
12	0.046	0.153	5.532	0.035	0.117	4.239	0.002	0.006	0.223
13	0.043	0.151	5.335	0.033	0.114	4.049	0.002	0.006	0.207
14	0.042	0.151	5.190	0.032	0.113	3.892	0.002	0.006	0.194
15	0.042	0.150	5.077	0.031	0.112	3.773	0.001	0.005	0.182
16	0.047	0.153	4.926	0.034	0.113	3.628	0.002	0.005	0.171
32	0.032	0.121	3.326	0.022	0.082	2.260	0.001	0.003	0.088
64	0.016	0.080	1.998	0.010	0.051	1.282	0.000	0.002	0.045
128	0.009	0.047	1.112	0.005	0.029	0.688	0.000	0.001	0.023
256	0.005	0.025	0.589	0.003	0.015	0.357	0.000	0.000	0.011

Table D.19 Multiplication, encryption and decryption efficiency of each resource for $N = 677$ and $max_{coef} = 516$.

	Multiplication			***Encryption***			***Decryption***		
M	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**
1	0.041	0.065	4.524	0.039	0.063	4.348	0.008	0.013	0.921
2	0.054	0.097	5.096	0.050	0.089	4.714	0.006	0.011	0.570
3	0.066	0.137	5.155	0.059	0.122	4.615	0.005	0.011	0.411
4	0.074	0.175	5.050	0.064	0.152	4.395	0.005	0.011	0.322
5	0.074	0.200	4.902	0.063	0.169	4.160	0.004	0.011	0.264
6	0.078	0.238	4.750	0.064	0.197	3.944	0.004	0.011	0.224
7	0.077	0.262	4.576	0.063	0.213	3.725	0.003	0.011	0.194
8	**0.079**	**0.293**	**4.407**	**0.063**	**0.234**	**3.527**	**0.003**	**0.011**	**0.172**
9	0.076	0.300	4.238	0.060	0.237	3.340	0.003	0.011	0.154
10	**0.074**	**0.324**	**4.103**	**0.057**	**0.252**	**3.186**	**0.003**	**0.011**	**0.139**
11	**0.057**	**0.181**	**7.582**	**0.044**	**0.139**	**5.810**	**0.002**	**0.006**	**0.244**
12	0.056	0.183	7.331	0.042	0.139	5.557	0.002	0.006	0.225
13	0.053	0.182	7.087	0.040	0.136	5.317	0.002	0.005	0.209
14	0.052	0.183	6.890	0.039	0.136	5.116	0.001	0.005	0.195
15	0.051	0.181	6.673	0.038	0.133	4.914	0.001	0.005	0.183
16	0.050	0.183	6.491	0.037	0.133	4.738	0.001	0.005	0.172
32	0.040	0.149	4.376	0.027	0.100	2.944	0.001	0.003	0.089
64	0.020	0.102	2.650	0.013	0.064	1.675	0.000	0.002	0.045
128	0.012	0.061	1.467	0.007	0.037	0.895	0.000	0.001	0.023
256	0.006	0.033	0.779	0.003	0.020	0.465	0.000	0.000	0.011

Table D.20 Multiplication, encryption and decryption efficiency of each resource for $N = 677$ and $max_{coef} = 677$.

	Multiplication			*Encryption*			*Decryption*		
M	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**
1	0.038	0.063	3.503	0.037	0.061	3.401	0.010	0.016	0.903
2	0.051	0.089	4.005	0.048	0.084	3.769	0.007	0.013	0.564
3	0.059	0.128	4.096	0.054	0.117	3.751	0.006	0.013	0.409
4	0.063	0.166	4.056	0.056	0.148	3.625	0.005	0.013	0.320
5	0.066	0.190	3.981	0.058	0.166	3.480	0.004	0.013	0.263
6	0.069	0.230	3.893	0.059	0.197	3.335	0.004	0.013	0.223
7	0.068	0.255	3.779	0.057	0.215	3.182	0.003	0.013	0.194
8	**0.069**	**0.289**	**3.674**	**0.057**	**0.240**	**3.043**	**0.003**	**0.013**	**0.171**
9	0.068	0.296	3.554	0.055	0.242	2.902	0.003	0.013	0.153
10	**0.065**	**0.322**	**3.467**	**0.052**	**0.259**	**2.789**	**0.003**	**0.013**	**0.139**
11	**0.050**	**0.167**	**6.440**	**0.040**	**0.133**	**5.121**	**0.002**	**0.006**	**0.243**
12	0.049	0.170	6.268	0.039	0.133	4.926	0.002	0.006	0.225
13	0.047	0.168	6.088	0.037	0.131	4.738	0.002	0.006	0.209
14	0.047	0.170	5.956	0.036	0.131	4.586	0.002	0.006	0.195
15	0.045	0.168	5.791	0.034	0.129	4.419	0.001	0.005	0.183
16	0.045	0.171	5.661	0.034	0.129	4.279	0.001	0.005	0.172
32	0.037	0.142	3.996	0.026	0.098	2.769	0.001	0.003	0.089
64	0.020	0.099	2.518	0.013	0.064	1.622	0.000	0.002	0.045
128	0.011	0.059	1.424	0.007	0.036	0.878	0.000	0.001	0.023
256	0.006	0.033	0.771	0.003	0.020	0.462	0.000	0.000	0.011

Table D.21 Multiplication, encryption and decryption efficiency of each resource for $N = 821$ and $max_{coef} = 625$.

	Multiplication			***Encryption***			***Decryption***		
M	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**
1	0.041	0.065	4.580	0.040	0.063	4.436	0.008	0.013	0.923
2	0.055	0.098	5.213	0.052	0.092	4.894	0.006	0.011	0.572
3	0.064	0.136	5.320	0.059	0.124	4.857	0.005	0.011	0.413
4	0.073	0.181	5.258	0.065	0.161	4.685	0.004	0.011	0.323
5	0.076	0.208	5.137	0.066	0.181	4.480	0.004	0.011	0.265
6	0.078	0.241	5.010	0.066	0.206	4.281	0.003	0.011	0.225
7	0.077	0.276	4.853	0.065	0.232	4.076	0.003	0.011	0.195
8	**0.079**	**0.311**	**4.713**	**0.065**	**0.257**	**3.893**	**0.003**	**0.011**	**0.172**
9	0.077	0.320	4.551	0.063	0.261	3.709	0.003	0.011	0.154
10	0.076	0.345	4.404	0.061	0.278	3.542	0.002	0.011	0.140
11	0.075	0.365	4.289	0.060	0.290	3.404	0.002	0.011	0.128
12	**0.073**	**0.390**	**4.151**	**0.057**	**0.306**	**3.260**	**0.002**	**0.011**	**0.117**
13	**0.055**	**0.189**	**7.750**	**0.043**	**0.147**	**6.026**	**0.001**	**0.005**	**0.210**
14	0.055	0.191	7.585	0.042	0.147	5.831	0.001	0.005	0.196
15	0.054	0.190	7.391	0.041	0.145	5.631	0.001	0.005	0.184
16	0.053	0.191	7.159	0.040	0.144	5.414	0.001	0.005	0.173
32	0.043	0.160	5.030	0.030	0.111	3.494	0.001	0.003	0.089
64	0.023	0.111	3.119	0.015	0.072	2.027	0.000	0.002	0.045
128	0.013	0.067	1.751	0.008	0.042	1.093	0.000	0.001	0.023
256	0.007	0.037	0.933	0.004	0.022	0.570	0.000	0.000	0.011

Table D.22 Multiplication, encryption and decryption efficiency of each resource for $N = 821$ and $max_{coef} = 821$.

	Multiplication			*Encryption*			*Decryption*		
M	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**
1	0.035	0.062	3.532	0.034	0.061	3.450	0.009	0.016	0.906
2	0.049	0.090	4.070	0.047	0.086	3.876	0.007	0.013	0.566
3	0.056	0.126	4.196	0.053	0.117	3.907	0.006	0.012	0.410
4	0.062	0.170	4.187	0.056	0.155	3.819	0.005	0.013	0.321
5	0.065	0.196	4.130	0.059	0.175	3.698	0.004	0.013	0.264
6	0.068	0.230	4.064	0.060	0.202	3.575	0.004	0.013	0.224
7	0.067	0.265	3.961	0.058	0.230	3.432	0.003	0.013	0.194
8	**0.068**	**0.302**	**3.876**	**0.058**	**0.258**	**3.309**	**0.003**	**0.013**	**0.172**
9	0.067	0.311	3.768	0.057	0.262	3.175	0.003	0.013	0.154
10	0.065	0.338	3.673	0.054	0.282	3.056	0.002	0.013	0.139
11	0.065	0.360	3.596	0.053	0.296	2.956	0.002	0.013	0.127
12	**0.064**	**0.387**	**3.499**	**0.052**	**0.315**	**2.847**	**0.002**	**0.013**	**0.117**
13	**0.050**	**0.172**	**6.570**	**0.040**	**0.139**	**5.294**	**0.002**	**0.005**	**0.209**
14	0.048	0.174	6.463	0.038	0.139	5.151	0.001	0.005	0.196
15	0.048	0.174	6.321	0.038	0.137	4.996	0.001	0.005	0.183
16	0.046	0.175	6.159	0.036	0.137	4.830	0.001	0.005	0.173
32	0.040	0.151	4.533	0.028	0.108	3.246	0.001	0.003	0.089
64	0.022	0.107	2.935	0.014	0.071	1.947	0.000	0.002	0.045
128	0.013	0.066	1.691	0.008	0.042	1.071	0.000	0.001	0.023
256	0.006	0.036	0.915	0.004	0.022	0.563	0.000	0.000	0.011

Table D.23 Multiplication, encryption and decryption efficiency of each resource for $N = 701$ and $max_{coef} = 533$.

	Multiplication			***Encryption***			***Decryption***		
M	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**
1	0.039	0.066	4.544	0.035	0.059	4.086	0.008	0.013	0.919
2	0.050	0.097	5.130	0.041	0.079	4.183	0.006	0.011	0.569
3	0.062	0.138	5.196	0.047	0.104	3.911	0.005	0.011	0.411
4	0.070	0.176	5.098	0.049	0.124	3.592	0.004	0.011	0.321
5	0.070	0.202	4.950	0.047	0.134	3.298	0.004	0.011	0.264
6	**0.072**	**0.240**	**4.809**	**0.046**	**0.152**	**3.049**	**0.003**	**0.011**	**0.224**
7	0.071	0.265	4.622	0.043	0.161	2.815	0.003	0.011	0.194
8	0.072	0.297	4.473	0.042	0.174	2.621	0.003	0.011	0.171
9	0.071	0.306	4.323	0.040	0.174	2.450	0.003	0.011	0.154
10	**0.069**	**0.327**	**4.147**	**0.038**	**0.181**	**2.288**	**0.002**	**0.011**	**0.139**
11	**0.053**	**0.170**	**7.720**	**0.028**	**0.091**	**4.142**	**0.002**	**0.005**	**0.243**
12	0.052	0.171	7.463	0.027	0.090	3.914	0.002	0.005	0.225
13	0.050	0.170	7.268	0.025	0.087	3.724	0.001	0.005	0.209
14	0.048	0.169	6.990	0.024	0.085	3.525	0.001	0.005	0.195
15	0.047	0.168	6.841	0.023	0.083	3.378	0.001	0.005	0.183
16	0.046	0.169	6.656	0.022	0.082	3.230	0.001	0.004	0.172
32	0.036	0.136	4.542	0.015	0.057	1.884	0.001	0.003	0.089
64	0.019	0.092	2.749	0.007	0.034	1.024	0.000	0.002	0.045
128	0.010	0.054	1.520	0.004	0.019	0.533	0.000	0.001	0.023
256	0.005	0.029	0.811	0.002	0.010	0.273	0.000	0.000	0.011

Table D.24 Multiplication, encryption and decryption efficiency of each resource for $N = 701$ and $max_{coef} = 701$.

	Multiplication			***Encryption***			***Decryption***		
M	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**	**E_{LUT}**	**E_{FF}**	**E_{BRAM}**
1	0.036	0.063	3.508	0.033	0.058	3.241	0.009	0.016	0.902
2	0.045	0.090	4.015	0.039	0.076	3.421	0.006	0.013	0.563
3	0.055	0.129	4.117	0.044	0.102	3.275	0.005	0.013	0.408
4	0.062	0.167	4.082	0.046	0.125	3.063	0.005	0.013	0.320
5	0.060	0.192	4.008	0.043	0.137	2.857	0.004	0.013	0.263
6	**0.063**	**0.232**	**3.927**	**0.043**	**0.158**	**2.673**	**0.004**	**0.013**	**0.223**
7	0.062	0.257	3.804	0.040	0.168	2.492	0.003	0.013	0.194
8	0.062	0.292	3.711	0.039	0.184	2.343	0.003	0.013	0.171
9	0.062	0.301	3.617	0.038	0.184	2.208	0.003	0.013	0.153
10	**0.061**	**0.324**	**3.488**	**0.036**	**0.193**	**2.074**	**0.002**	**0.013**	**0.139**
11	**0.046**	**0.155**	**6.530**	**0.027**	**0.089**	**3.774**	**0.002**	**0.006**	**0.243**
12	0.045	0.156	6.353	0.025	0.088	3.589	0.002	0.006	0.224
13	0.043	0.156	6.225	0.024	0.086	3.431	0.001	0.005	0.209
14	0.042	0.155	6.022	0.023	0.084	3.265	0.001	0.005	0.195
15	0.042	0.155	5.921	0.022	0.082	3.141	0.001	0.005	0.183
16	0.041	0.157	5.791	0.021	0.082	3.014	0.001	0.005	0.172
32	0.033	0.129	4.138	0.014	0.057	1.812	0.001	0.003	0.089
64	0.018	0.089	2.609	0.007	0.034	1.004	0.000	0.002	0.045
128	0.010	0.053	1.475	0.004	0.019	0.533	0.000	0.001	0.023
256	0.005	0.029	0.800	0.002	0.010	0.272	0.000	0.000	0.011