CRANFIELD UNIVERSITY

MAULANA RANDA

DESIGN OF HARDWARE-ORIENTED SECURITY TOWARDS TRUSTED ELECTRONICS

SCHOOL OF AEROSPACE, TRANSPORT, AND MANUFACTURING

Doctor of Philosophy
Academic Year: 2017 - 2020

Supervisor: Prof. Ian K Jennions
Associate Supervisor: Dr. Mohammad Samie
July 2020

CRANFIELD UNIVERSITY

SCHOOL OF AEROSPACE, TRANSPORT, AND MANUFACTURING

Doctor of Philosophy

Academic Year 2017 - 2020

MAULANA RANDA

Design of Hardware-oriented Security Towards Trusted Electronics

Supervisor: Prof. Ian K Jennions
Associate Supervisor: Dr. Mohammad Samie
July 2020

This thesis is submitted in partial fulfilment of the requirements for the degree of Doctor of Philosophy

ABSTRACT

While the Internet of Things (IoT) becomes one of the critical components in the cyber-physical system of industry 4.0, its root of trust still lacks consideration. The purpose of this thesis was to increase the root of trust in electronic devices by enhance the reliability, testability, and security of the bottom layer of the IoT system, which is the Very Large-Scale Integration (VLSI) device. This was achieved by implement a new class of security primitive to secure the IJTAG network as an access point for testing and programming. The proposed security primitive expands the properties of a Physically Unclonable Function (PUF) to generate two different responses from a single challenge. The development of such feature was done using the ring counter circuit as the source of randomness of the PUF to increase the efficiency of the proposed PUF. The efficiency of the newly developed PUF was measured by comparing its properties with the properties of a legacy PUF. The randomness test done for the PUF shows that it has a limitation when implemented in sub-nm devices. However, when it was implemented in current 28nm silicon technology, it increases the sensitivity of the PUF as a sensor to detect malicious modification to the FPGA configuration file. Moreover, the efficiency of the developed bimodal PUF increases by 20.4% compared to the legacy PUF. This shows that the proposed security primitive proves to be more dependable and trustworthy than the previously proposed approach.

Keywords:

Hardware Security, Physically Unclonable Function, Random Number Generator, JTAG, IJTAG, FPGA

ACKNOWLEDGEMENTS

I would like to thank all those who have supported me in accomplishing this thesis. Without their endorsement, this thesis would not have been possible.

Foremost, I would like to pay my special regards to Dr. Anne Kusmayati for her encouragement for me to undertake a PhD and for the financial support I received through the Research and Development Agency of The Ministry of Defense of the Republic of Indonesia.

I would like to express the deepest gratitude to my supervisor, Professor Ian Jennions, for giving me the opportunity to work in Integrated Vehicle Health Management (IVHM) Centre and encouraging me to have the freedom to develop my ideas. These chances, together with his guidance, have helped to lead me through the process.

I am also deeply indebted to Dr. Mohammad Samie, my associate supervisor, who was also with me along this journey. His enthusiasm, constructive criticism, and fruitful discussions without which this thesis would be sorely diminished are most kindly acknowledged.

I wish to thank my wife and children for their support and patience over the course of my study period

Finally, I dedicate this work to my mother and father.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

PUF	Physically Unclonable Function
SoR	Source of Randomness
IoT	Internet of Things
SR	Shift Register
RC	Ring Counter
RO	Ring Oscillator
EDA	Electronic Design Automation
DRC	Design Rule Check
CPS	Cyber-Physical System
IP	Intellectual Properties
RoT	Root of Trust
ECT	Embedded Core Test
JTAG	Joint Test Access Group
IJTAG	Internal JTAG
TRNG	True Random Number Generator
PRNG	Pseudo-Random Number Generator
RNG	Random Number Generator
VLSI	Very Large-Scale Integration
RCRNG	Ring Counter-based Random Number Generator
RCPUF	Ring Counter-based Physically Unclonable Function
ROPUF	Ring Oscillator-based Physically Unclonable Function
BRCPUF	Bimodal Ring Counter-based Physically Unclonable Function
BROPUF	Bimodal Ring Oscillator-based Physically Unclonable Function
nm	nano millimetre
CRP	Challenge-Response Pair
HDL	Hardware Description Language
VHDL	VHSIC-HDL, Very High-Speed Integrated Circuit Hardware Description Language
NTV	Near-Threshold Voltage
LUT	Look-Up Table
FPGA	Field Programmable Gate Array

FIPS	Federal Information Processing Standard
NIST	National Institute of Standard and Technology
AIS	Application Notes and Interpretation of the Scheme
PAR	Placement and Route
VM	Virtual Machine
IC	Integrated Circuit
ID	Identity
TERO	Transient Effect Ring Oscillator
ASIC	Application Specific Integrated Circuit
SRAM	Static Random-Access Memory
LFSR	Linear Feedback Shift Register
BIST	Built-in Self-Test
TAP	Test Access Port
IUT	Instrument Under Test
SIB	Segment Insertion Bit
TDR	Test Data Register
TDI	Test Data In
TDO	Test Data Out
TMS	Test Mode Select
TRST	Test Reset
TCK	Test Clock
FSM	Finite State Machine

1 INTRODUCTION

1.1 Background

One of the moving components of industry 4.0 is the presence of a Cyber-Physical System (CPS). It is defined as a networked-embedded system that interacts with the environment [1], such as through autonomous vehicles or an industrial control system. The core of the CPS is the Internet of Things (IoT), which acts as an interface between the physical world and the cyber system. While bringing in many advantages, the IoT also brings in a couple of challenges, such as the security of the system. For an IoT device that has no area, time, power, and energy restriction (such as in a manufacturing plant), the security of the system can be easily implemented. However, for smaller IoT devices such as medical implants, where all of the parameters are constrained, the application of security measures needs to be well-thought-out. As suggested by Verbauwhede [2], the security of the system needs to be present in every different level of the system as a whole, as can be seen in Figure 1-1.

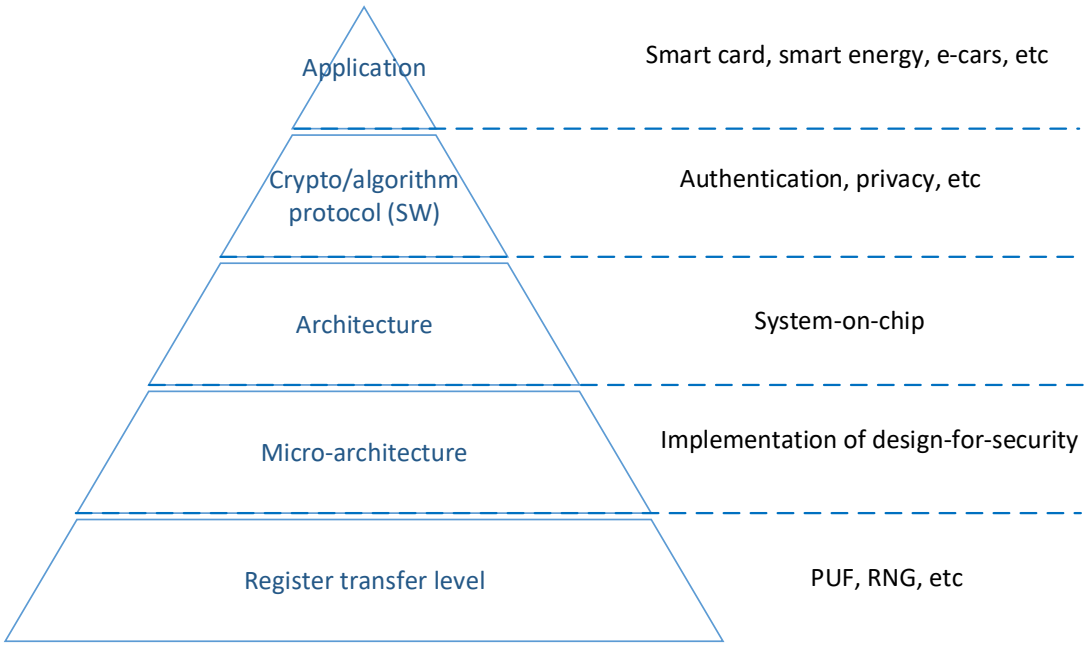

Figure 1-1: Hierarchy of security

A system or component needs to be trusted in order for it to run as expected for its intended purpose. In the old days, the threat to a system was more apparent in the communication channel between the nodes. One of the popular solutions to such an old-style threat is to protect the data through the implementation of a strong cryptographic protocol. However, the threat model for a modern system not only comes into play in the communication channel itself but also the nodes. When the component that provides the cryptographic protocol is not secure, the service that it provides is also threatened. Therefore, a Root of Trust (RoT) needs to be built into a system. Referring to Figure 1-1, Verbauwhede [2] defines the RoT of a system as a measure to account for the trustworthiness of the systems located in the lower-level layers. The extreme end of this definition means that in order to have a secure system, the lowest layer of the system, the transistor level, needs to be secure too.

The ever-evolving world of semiconductors has led to more complex Very Large-Scale Integration (VLSI) devices in terms of structure and functionality. While this advancement provides an advantage as there is more of a possibility of solving different kinds of problems, it also becomes a problem in itself. The increase in functional complexity means that more things can go wrong if it is not well-designed. For example, a different type of confidential information stored in IoT devices can be compromised if there is not enough protection. A Cloud-computing service with shared resources can become an easy target for intellectual property thievery [3]. The complexity of providing a secure environment for this kind of technology is enhanced by the fact that it is getting harder to perform affordable and fast testing procedures before the product is deployed in the market. It is almost impossible to physically probe all pins in the VLSI device to test its functionality.

The IEEE 1149.1 JTAG, IEEE 1500 Embedded Core Test (ECT), and IEEE 1687 Internal JTAG (IJTAG) have become the standard tests used to overcome the aforementioned testing challenges. The three standards have their own specificities and were made to be used side by side.

Other than the favourable circumstances that it brings in to the testability of the embedded instruments, JTAG confronts various security-related difficulties when it is utilised as the future standard of embedded instrument testing. The first is that the JTAG does not have a built-in security mechanism to forestall unauthorised access to the embedded instrument. A report about security breaks by means of JTAG can be found in the news [4] and papers [5]. Majeric [6] and Elnaggar [7] presented the JTAG fault injection attack and data integrity attack consecutively to modify the Intellectual Property (IP) core maliciously, steal classified information, reverse engineer the IP, and even stop the entire framework if it is undermined. Hence the security of the IJTAG network that gives access to the gadget is essential.

While it is also of importance to secure the embedded core, it is still a challenge to provide security to the ECT standard. The reason for this is that the ECT standard is prescriptive, which means that a modification to the standard will break the compliance of the system to the standard. On the other hand, the IJTAG standard is descriptive, which means that the engineer can have different implementations of the standard so long as they follow the description of the standard. Hence the security of the IJTAG will be the main topic of discussion in this thesis.

There is ongoing research into providing better security in IJTAG networks such as the use of a static secret key ([8], [9]) and dynamic secret key ([10], [11]) to unlock the Segment Insertion Bit (SIB). However, they only secure access to the IJTAG network without securing the output data from the embedded instrument. An echeloned IJTAG data protection mechanism [12] that is proposed to secure the access, as well as the output data, is also facing a scalability problem.

1.2 Research Gaps/Industrial Needs:

Having noticed the challenges, it is identified that building the security and root of trust into electronics is a critical industrial requirements for the future embedded systems. It is a need now to develop novel security mechanisms for devices'

access ports; especially, the widely used IJTAG network additional to the conventional approach of data security using some form of cryptography. From a hardware implementation point of view, building the security in the embedded systems requires characterisation of embedded devices based on their performance variations raised from physical changes and manufacturing process. This is usually conducted by observing physical resources, which expose the hardware in various performance variations; and employing it to develop electronic signatures, device identification, and digital fingerprint. This; obviously, involves various fields in design, test, and manufacturing of electronics, while each has specific gaps; however, we limit the research to address the following observed challenges:

- **Gaps in the hardware root of trust**
  The bottom layer of the root of trust lies within the transistor level of the device. However, the current access port and test mechanisms are primarily developed without any consideration of the root of trust. Therefore, it is of importance to secure access to the test mechanism as well as securing its data to achieve better security.
- **Gaps in observing the source of randomness in the hardware level**
  The emergence of sub-nano millimetre electronic device needs to be well secured against adversaries. However, the current security measures are only available in the form of algorithmically generated a random number, which can be broken easily using deep learning. Thus, a true random number generator based on the physical variation of the device needs to be developed and characterised to understand its performance in sub-nm technology.
- **Gaps in the device identification**
  One of the critical aspects of security is to be able to identify whether a device is an original, counterfeited or modified part. A security primitive that can be used for such purpose is the Physically Unclonable Function (PUF). However, the current mechanism for device identification using PUF relies on the response of the PUF, which easily altered by

environmental change. Thus, it is of importance to develop a new mechanism that has better resistance to environmental change.

- **Gaps in performance analysis**

  The performance analysis of a PUF is done by comparing the response variance of a PUF architecture implemented in two or more devices exposed to environmental variation. This mechanism requires the device to be in a test mode, which increases the down time of the system. However, it is always a desire to lower the down time. Thus, it is of importance to develop a new PUF performance analysis mechanism that can be performed in mission mode; and hence, decreasing the down time.

1.3 Research Aim and Objectives

1.3.1 Problem Description

Given embedded electronics as a collection of primitive components, circuitries, building blocks in the form of IPs fabricated as ASIC/FPGAs/SoCs; there are still technical problems for exploiting the device's performance variation in the development of root of trust. Such devices are equipped with test access ports, networks, and mechanisms, which allow external device-to-device and internal instrument-to-instrument communications needed for testing the function of specific hardware entities within the device. They are not intended for testing the performance of the device needed to assess the system concerning the security. Therefore, existing advanced test mechanisms do not have the capability for observing sources of randomness induced to the system unintentionally due to manufacturing process variation. Such variations are monitored using parameters such as power consumption, temperature, and propagation delays for the same particular devices against the specific working condition. Monitoring such signals requires enhancing internal test network with a proper sensor network distributed across the entire device. As only a few sensors are fabricated into the embedded electronics, designers would need to employ available resources and primitives to form sensors for observing the sign of variation from

device to device. This leaves us with the following problems for building root of trust:

Given embedded electronics with n number of sensors built using m primitives distributed across the entire system, device performance variation is observed in the form of propagation delays which drive electronic fingerprint, specific for a particular device under test, different from all other similar prototypes. The generated fingerprint is the key element needed for controlling access to the hardware as well as the encryption of the data generated by the hardware. This requires additional care in:

- Identifying proper signals linked with sources of randomness in the device
- Architecting right set of sensors using available primitives fabricated within the device
- Generating fingerprints from signals observed from the source of the randomness
- Integrating sensor network and the required signal path with the test access mechanism
- Building the root of trust

The conventional design flow does not yet allow engineers to construct the device identity and the root of trust, which is essential for the protection of vulnerabilities in ASICs, FPGAs and SoCs. In response to the problem mentioned earlier, this dissertation explores the accessibility of the conventional test access mechanism to break the bottleneck of the development of security and trust in embedded electronics.

1.3.2 Hypothesis

Building bimodal characteristics into the performance of the physically unclonable function (which drives electronic fingerprint from the source of randomness observed by the random number generators) enables designers to construct a

unique solution for obfuscating both the hardware and data, which lead to the establishment of the root of trust essential for the test access mechanisms. In this regard, the customizability of the security primitives within the test access allows for the exploration of the devices' source of randomness to secure the access and information of the IPs within the embedded electronics.

1.3.3 Aim

In order to address the aforementioned research gaps, this thesis aims to improve the security of the embedded system through the development and implementation of design-for-security.

1.3.4 Objectives

Based on the aim of the research above, the following key objectives have been defined to achieve the aim:

1. Develop and characterise a novel random number generator design based on the ring counter circuit (RCRNG).
2. Develop and characterise a novel digital physically unclonable function based on the ring counter circuit (RCPUF).
3. To develop and characterise bimodal RCPUF (BRCPUF) to secure access to the IJTAG network as well as its output data.

1.4 Research Methodology

In response to the gaps mentioned earlier, we propose a novel security mechanism developed based on a Physically Unclonable Function (PUF), which is a product of the utilisation of the physical randomness of an object/device that is easy to produce, but non-invertable and unpredictable [13]. A PUF can also be defined as a constrained True Random Number Generator (TRNG). Therefore, an understanding of TRNG is a fundamental requirement to develop a PUF.

Build upon the requirement as mentioned above; this thesis will first discuss the development of a true random number generator. The findings and experience drawn from the development of a TRNG will be used to develop a PUF which will then be used to develop the bimodal PUF to secure the IJTAG network.

The research methodology is divided into 4 phases, as illustrated in Figure 1-2. The first phase is where the gaps in the field of research are discovered by performing an extensive literature review. The scope of the research is then defined to focus on the area of study. From there on, the aim and objectives of the research are formalised. The research design is then developed as a guideline for the experiment to be performed in accordance with the aim and objectives of the research.

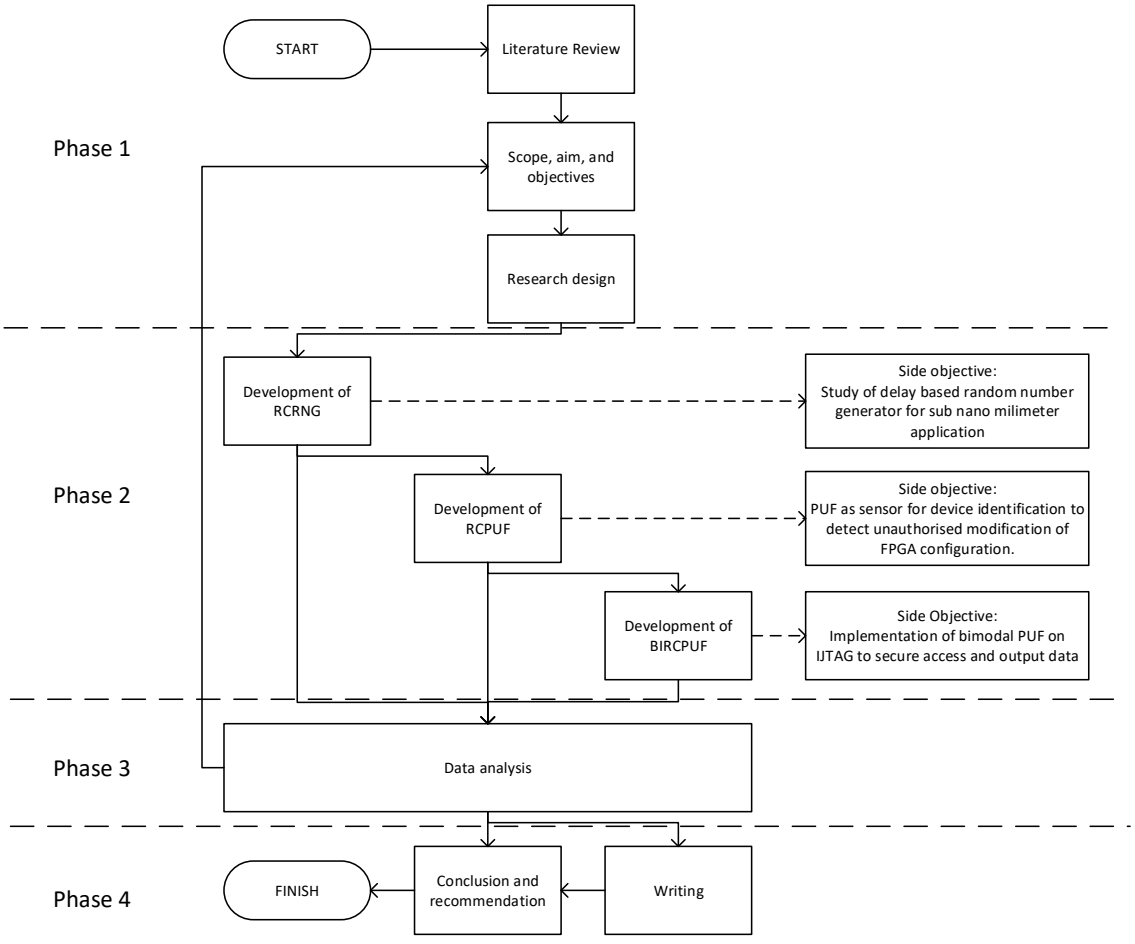

Figure 1-2: Flowchart of the research

The second phase is the experimental phase. The initial idea of this research is to find an efficient design-for-security element to secure an embedded system. For that matter, this research proposed a new class of Physically Unclonable Function (PUF) which will then be called a bimodal PUF. In order to develop a bimodal PUF, knowledge on how to develop a regular PUF is a prerequisite. Moreover, an understanding of the Random Number Generator (RNG) is needed to construct a PUF as it is the basic building block of PUF. Therefore 3 experiments are designed to achieve the aim and objectives of the research.

The first experiment is to develop a random number generator based on the Ring Counter (RC) circuit. Apart from getting an understanding of how the RNG works and behave, this experiment also aimed to implement the developed RNG as a model to study the characteristics of a delay-based RNG in sub-nano millimetre technology. The reason why this topic is chosen to be studied is that the behaviour of the TRNG as one of the critical components to provide security in sub-nano millimetre (nm) devices is not well-studied. It is essential to understand its characteristics when it is implemented in the sub-nm, so then the hardware security designer knows what to expect and can develop a better implementation of TRNG for security purposes in sub-nm devices.

After gaining a better understanding of the development of a TRNG, the second experiment is conducted to develop a PUF based on the ring counter circuit. As has already been mentioned above, the basic building block of a PUF is a random number generator. In a digital circuit, the PUF is created by selectively choosing an array for the Source of Randomness (SoR) comparison, so then the output is reproducible. A challenge in the form of a binary number is used to select which source of randomness is to be compared. The output of the PUF is then called the response. Each challenge will, ideally, generate a unique response. This can also be called a challenge-response pair (CRP). The source of randomness for the PUF is based on the ring counter circuit. Aside from characterising and gaining a better understanding of how the PUF works, this experiment also aimed to implement the developed PUF as an affordable high-sensitivity digital sensor

to detect unauthorised modifications to the configuration file of multi-tenant FPGA. This has now become an emerging trend in Cloud computing.

Lastly, the third experiment is conducted to develop a new class of physically unclonable functions called a bimodal PUF. It is aimed to be a design-for-security solution for embedded devices. The bimodal PUF is characterised and implemented in the IJTAG network to not only efficiently secure the access to the embedded instruments but also to secure the output data of the embedded instruments.

All three experiments have a common thread in that they all utilise the ring counter circuit as the main source of randomness. To the best of the author's knowledge, no publication uses a ring counter as the primary source of randomness in their research. Therefore, the utilisation of the ring counter circuit as the main source of randomness is claimed to be one of the novelties in this research. The other innovations in this research are as follows:

1. Study on the behaviour of a delay-based random number generator in sub-nano millimetre technology.
2. A novel high-sensitivity digital sensor to detect unauthorised modifications of the configuration file of a multi-tenant FPGA.
3. A new definition of the uniqueness and reliability parameters for PUF characterisation.
4. A new class of PUF that can generate two simultaneous responses from a single challenge. This new class of PUF will be called a bimodal PUF.

1.4.1 Research limitation

The majority of publications about hardware security include a discussion on the performance of the proposed work under different environmental conditions. For example, silicon-based hardware security primitives are said to incur a behavioural change under different temperatures and/or supply voltage stress. The system may not behave as it is intended to be or may leak confidential information that the security primitives are trying to hide.

However, this thesis will not include such a test in the discussion part of each chapter. Nonetheless, it does not lower the confidence level of the obtained experimental result. The reason for this is that in order to perform such a measurement, the decapsulation process needs to be done to the chip-under-test [14]. All security primitives proposed in this thesis (TRNG, PUF, bimodal PUF) are implemented as on-chip security. This means that the security mechanism and the object have tried to secure what resides in the same chip. Consequently, when an attacker is trying to break the security measures by increasing the supply voltage or/and the temperature, and they conduct the measurement without the decapsulating process, they will not be able to get an accurate measurement, and therefore the obtained information will not be accurate. If the attacker performs the decapsulation procedure, it will change the physical properties of the security mechanism. Consequently, the response generated by the security mechanism will also change. Moreover, the temperature/voltage stress applied to the chip might even break the information that the attacker is trying to get as it operates beyond the specification given by the manufacturer.

1.5 The organisation of the thesis

The main content of the thesis is divided into three chapters in a paper format. What this means is that each chapter/paper will have a literature review, aim and objectives, research methodology and analysis. However, all three chapters/paper do not stand on their own. They are more of a stepping stone to the chapter following the previous one. Chapter 2 of this thesis discusses the development of a random number generator which will be used as a building block to build a PUF in Chapter 3. Similarly, the PUF developed in Chapter 3 will be used as the foundation to develop the bimodal PUF in Chapter 4. The last chapter of this thesis is the conclusion that serves as a retrospective examination of what has been done in relation to the aim and objectives of the thesis. Chapter 5 also discuss the possibility of future work that can be done as a further development of the things presented in this thesis.

1.6 Risk and mitigation plan

1.6.1 Hardware description language

Hardware design involves the use of Hardware Description Language (HDL). There are three types of HDL; Verilog, System Verilog and VHDL (VHSIC-HDL and Very High-Speed Integrated Circuit Hardware Description Language). For this thesis, the author chooses to use the VHDL because it has non-C like syntax. This is easier to be used by people who do not have a strong background in programming. Nevertheless, the author needs some time to adapt to the VHDL as well as the complexity of the Electronic Design Automation (EDA) software used, e.g. ISE 14.7 from Xilinx.

The mitigation plan used to overcome this situation is to do the task manually, i.e. to hard-code the VHDL file. Nevertheless, this limitation did not reduce the confidence level of the results obtained in the experiment.

1.6.2 Modified IJTAG integration

The configurability of the IJTAG to include segment insertion bit is what separates it with the JTAG. The IJTAG network needs to be synthesised together along with other logic of the ASIC. However, because of the time and resource limitation, the ASIC implementation of the proposed IJTAG network cannot be done.

As a solution to this situation, the author implements the modified IJTAG network in FPGA. FPGA primitives and resources are used to model the IJTAG functionality and integrate the proposed PUF for the IJTAG security mechanism.

1.6.3 Randomness measurement

The Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications NIST SP 800-22 [15] standard was used to measure the randomness of the RNG in this thesis. The standard consists of 15 statistical tests. The standard decides whether an RNG has the right amount of

randomness or not by thresholding the output of each of the statistical tests. Because of this, it is possible for an RNG that falls below the threshold by just a small amount to be considered to fail the test.

To overcome this situation, the NIST SP 800-22 suggests that the tester perform a graphical analysis to determine the high-level performance of the RNG under-test. The graphical analysis can be done by plotting the results of the statistical tests in a bar chart or by plotting the generated random number using a heat map.

1.7 List of Published/Submitted Work

1.7.1 Journal Publications

M. Randa, M. Samie, I. Jennions. "Delay-Based True Random Number Generator in Sub-Nanomillimeter IoT Devices," in *Electronics* 2020, *9*, 817.

1.7.2 Conference Publications

M. Randa, M. Bozdal, M. Samie, I. Jennions. "Layered Security for IEEE 1687 Using a Bimodal Physically Unclonable Function," in *Procedia Manufacturing*. 2018; 16:24-30.

M. Bozdal, M. Randa, M. Samie, I. Jennions. "Hardware Trojan Enabled Denial of Service Attack on CAN Bus," in *Procedia Manufacturing*. 2018;16:47-52.

1.7.3 Virtual Conference Presentations

J. Buu-Sao, M. Samie, M. Randa, et. al., "IoT Security – Hardware Perspective", December 2018, the IoT Day Slam 2018, VIRTUAL Internet of Things Conference: https://iotslam.com/session/iot-security-hardware-perspective/.

1.7.4 Under Submission for Journal Publication

M. Randa, M. Samie, I. Jennions. "A High-Sensitivity Digital Sensor For The Detection of Unauthorised Modifications of Multi-Tenant FPGA Configuration Files Based On A Ring Counter Physically Unclonable Function,". – Under review for IEEE Access, July 2020

1.8 References

[1] G. Loukas, “A Cyber-Physical World,” in *Cyber-Physical Attacks*, London: Elsevier, 2015, pp. 1–19.

[2] I. Verbauwhede, “Hardware Security Knowledge Area,” in *The Cyber Security Body Of Knowledge*, 1.0., Andrew Martin and G. Danezis, Eds. Bristol: University of Bristol, 2019.

[3] J. Robertson and M. Riley, “The Big Hack: How China Used a Tiny Chip to Infiltrate U.S. Companies - Bloomberg,” *Bloomberg.com*, 2018. [Online]. Available: https://www.bloomberg.com/news/features/2018-10-04/the-big-hack-how-china-used-a-tiny-chip-to-infiltrate-america-s-top-companies. [Accessed: 02-Jun-2020].

[4] R. Johnson, “Sergei Skorobogatov Defends Backdoor Claims - Business Insider,” *Business Insider*, 2012. [Online]. Available: https://www.businessinsider.com/sergei-skorobogatov-defends-backdoor-claims-2012-5?r=US&IR=T. [Accessed: 30-May-2020].

[5] S. Skorobogatov and C. Woods, “Breakthrough silicon scanning discovers backdoor in military chip,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7428 LNCS, pp. 23–40, 2012.

[6] F. Majéric, B. Gonzalvo, and L. Bossuet, “JTAG Fault Injection Attack,” *IEEE Embedded Systems Letters*, vol. 10, no. 3, pp. 65–68, Sep. 2018.

[7] R. Elnaggar, R. Karri, and K. Chakrabarty, “Securing IJTAG against data-

integrity attacks," in *Proceedings of the IEEE VLSI Test Symposium*, 2018, vol. 2018-April, pp. 1–6.

[8] J. Dworak, A. Crouch, J. Potter, A. Zygmontowicz, and M. Thornton, "Don't forget to lock your SIB: Hiding instruments using P16871," in *Proceedings - International Test Conference*, 2013, pp. 1–10.

[9] H. Liu and V. D. Agrawal, "Securing IEEE 1687-2014 Standard Instrumentation Access by LFSR Key," in *Proceedings of the Asian Test Symposium*, 2015, vol. 2016-Febru, pp. 91–96.

[10] R. Baranowski, M. A. Kochte, and H. J. Wunderlich, "Fine-grained access management in reconfigurable scan networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 937–946, Jun. 2015.

[11] K. Sudeendra Kumar, N. Satheesh, A. Mahapatra, S. Sahoo, and K. K. Mahapatra, "Securing IEEE 1687 standard on-chip instrumentation access using PUF," in *Proceedings - 2016 IEEE International Symposium on Nanoelectronic and Information Systems, iNIS 2016*, 2017, pp. 56–61.

[12] S. Kan, J. Dworak, and J. G. Dunham, "Echeloned IJTAG data protection," in *Proceedings of the 2016 IEEE Asian Hardware Oriented Security and Trust Symposium, AsianHOST 2016*, 2017, pp. 1–6.

[13] S. Mulhem and W. Adi, "New Mathblocks-Based Feistel-Like Ciphers for Creating Clone-Resistant FPGA Devices," *Cryptography*, vol. 3, no. 4, p. 28, Dec. 2019.

[14] M. Hutter and J. M. Schmidt, "The temperature side channel and heating fault attacks," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8419 LNCS, pp. 219–235.

[15] A. Rukhin *et al.*, "Special Publication 800-22 Revision 1a A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," Apr. 2010.

2 DELAY-BASED TRUE RANDOM NUMBER GENERATOR IN SUB-NANOMILLIMETER IOT DEVICES

2.1 Abstract

True Random Number Generators (TRNGs) use physical phenomena as their source of randomness. In electronics, one of the most popular structures to build a TRNG is constructed based on the circuits that form propagation delays such as a ring oscillator, shift register, and routing paths. This type of TRNG has been well-researched within the current technology of electronics. However, in the future where electronics will use sub-nano millimetre (nm) technology, the components become smaller and work on near-threshold voltage (NTV). This condition has an effect on the timing-critical circuit as the distribution of the process variation becomes non-gaussian. Therefore, there is an urge to assess the behaviour of the current delay-based TRNG system in sub-nm technology. In this research, a model of TRNG implementation in sub-nm technology was created through the use of a specific Look-Up Table (LUT) in the Field-Programmable Gate Array (FPGA), known as SRL16E. The characterization of the TRNG was presented, and it shows a promising result, in that the delay-based TRNG will work properly with some constraints in sub-nm technology.

2.2 Introduction

In the era of the internet of things (IoT), everyone feels the need for privacy and security because their private data is floating around in the connected cloud [1]. As IoT-based systems have both hardware and software requirements, there is always a potential for systems to be hacked if the hardware is not as well-secured to a suitable level as the software. Research in recent years has demonstrated the existence of malware that could be removed from the system if appropriate software-level countermeasures are set up, correctly [2]. Hackers might target such malware for hacking the physical systems. Therefore, hardware security is also an essential requirement, besides the security of the software, to ensure that

the security of the system and the privacy of the user's data are well-established [2].

Cryptography is now essential for securing access to both the data and hardware, which is necessary for IoT-based systems [3]. A key is a vital aspect for cryptography, and it can be created using a Random Number Generator (RNG). There are two types of random number generator; a true random number generator (TRNG) and a pseudo-random number generator (PRNG). The comparison between TRNG and PRNG has been summarized in Table 2-1.

Table 2-1: Comparison Between TRNG and PRNG

	TRNG	PRNG
Source of randomness	Physical phenomenon	Mathematical algorithm
Uniformity	Yes	Yes
Independence	Yes	No (Periodic/deterministic)
Efficiency	low	high

While a pseudo-RNG (PRNG) is simple to implement and sufficient enough for many applications, there is always a desire to have a TRNG, especially for highly critical systems. The reason for this is that PRNG was created from a computational algorithm that has deterministic properties. When the algorithm behind the PRNG is compromised, the random number that it generates is also compromised. On the other hand, a TRNG utilizes a physical system that has intrinsic randomness, which can be extracted to create an RNG. This results in non-deterministic properties for the TRNG.

There have been various designs and technologies suggested for architecting TRNGs for different types of IoT. For a big-sized IoT, such as a smart-fridge, smart-toaster etc., a TRNG that uses optical scattering [4], [5] and radioactive decay [6] as its source of randomness (SoR) can be used. While these TRNGs are bulky and have low efficiency, they have an excellent randomness property. For smaller IoT devices such as a smartphone, the use of sensors such as an

accelerometer and gyroscope as the source of randomness have been reported to have excellent results [7]–[9]. However, the implementation of these approaches still relies on external data processing, e.g. a PC, which is impossible to include in resource-constrained devices such as an implanted IoT like a pacemaker. For this type of IoT, a TRNG that utilizes the intrinsic parameters of the devices is preferred as they do not have to rely on the external source of randomness.

An all-digital RNG implemented in 65nm and 14nm technology was proposed in [4] and [5] respectively. Pamula [4] proposes a high-quality RNG based on a processed low-quality RNG with intrinsic SoR. Their analysis shows excellent performance. However, the technology used is too big for an implanted IoT. In [5], the author implements TRNG in the latest semiconductor technology. However, the source of randomness used is not always available in the IoT devices, making it difficult to achieve in IoT. In modern FPGA technology, the SRL16E is standard, and it has the potential to be used in TRNG. The author in [9, 10] uses the SRL16E and configures it to be a ring counter in order for it to become one of the components of their TRNG. However, they only use the ring counter as a complementary component to increase the periodicity of the RNG and not as the primary source of randomness.

Moreover, the size of transistors in the future will become smaller beyond nano-millimetre technology [6]. This causes the electronic devices to run at a near-threshold voltage (NTV) [7]. These phenomena have an impact on the critical timing of the device because the distribution of the process variation is non-gaussian [8]. A couple of research studies have been done to address this issue [9], [12]. However, from the extensive literature review, a report on the effect of NTV in the time-critical application such as a delay-based random number generator is not in existence.

This chapter presents a study on the implementation of delay-based TRNG intending to explore TRNG performance and properties in sub-nm technology. The sub-nm delay-based RNG was modelled in FPGA using a ring counter based on the SRL16E configuration of Xilinx's LUT as the main source of randomness.

The rest of the chapter is organised as follows: Section 2.3 provides an introduction to RNG implementation in the FPGA and the metrics for RNG characterization. The experimental setup, practical limitations, and a framework for location selection are presented in section 2.4. The results, findings, and statistical analysis are discussed in Section 2.5. Finally, this chapter will be concluded in Section 2.6.

2.3 Related Works

2.3.1 Random Number Generator

The idea of a random number generator is based on stochastic modelling in which an observable random variable can be obtained from a random phenomenon. In a random number generator, let S be the state space of the generator, which is also a subset of a set Ω. The random variable generated is part of the random space U that is extracted using the extraction function g. What is being obtained by g is a mapped state space S by function *f* so that $f{:}S\rightarrow S$.

$$\begin{aligned} &\Omega \in S \\ &f: S \rightarrow S \\ &g: S \rightarrow U \end{aligned} \qquad \textbf{(2-1)}$$

In a True Random Number Generator (TRNG), f is the physical source of randomness and g is the logic or function used to process the source of randomness further. In a Pseudo-Random Number Generator (PRNG), f and g are the mathematical algorithms used to generate the random number. Both TRNG and PRNG need an initial condition. In TRNG, the initial condition is any current state of the physical system while in PRNG, the initial condition needs to be provided by the seed x_0. Figure 2-1 is given to illustrate this mechanism.

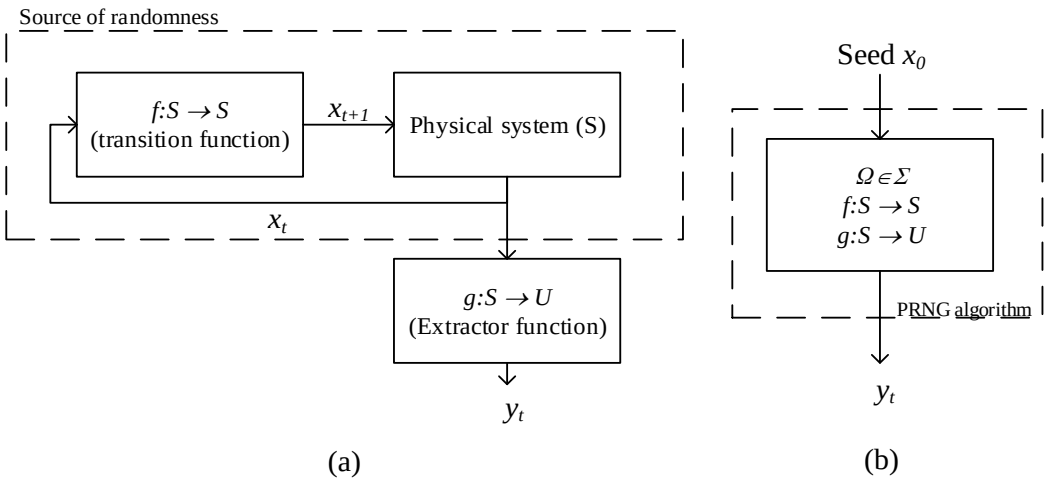

Figure 2-1: RNG configuration (a) TRNG and (b) PRNG

In semiconductor devices, one of the sources of randomness is from the shift in the D.C. current, also known as the burst noise [13]. This phenomenon happens because of the modulation of the current flowing over a physical barrier. The magnitude of this current can be calculated using the Schottky equation, as in (2-2).

$$\overline{i^2} = 2qI_D\Delta f \qquad \textbf{(2-2)}$$

Where q is the electronic charge, I_D is the average value of the random current pulses at the drain of the transistor and Δf is the measurement bandwidth. From (2-2), it can be seen that the bigger the bandwidth of the measurement, the higher the current will become. It also suggests that the higher the pulsating current at the drain, the more that the noise will increase. Burst noise is mostly caused by a random variation such as crystallographic defects in the bipolar junction transistor. Impurities can slip into the defect during the manufacturing process and form a low resistance current path. When the current flows over this resistance, some of it will leak, meaning that the output has current inconsistency.

Another source of randomness in semiconductor devices that comes from a random variation in the manufacturing process is the flicker noise. Flicker noise is also known as $1/f$ noise because it is mainly affecting the lower frequency range, i.e., Megahertz frequency. Two theories can be used to explain the flicker noise phenomenon, namely the number fluctuation theory [14] and mobility fluctuation theory [15]. Number fluctuation theory explains that flicker noise happens because there is an inconsistency in the number of electrons that can pass through the defective current path at any given time. On the contrary, mobility fluctuation theory states that flicker noise does not have any correlation with the number of electrons that pass the defective current path. Still, the velocity inconsistency of those electrons causes this. However, both theories agree that the leading cause of the flicker noise comes from the defective current path of the transistor, which is a random variation of the manufacturing process.

2.3.2 Random Number Generator in FPGA

FPGA refers to embedded electronics comprised of a vast number of digital circuitries known as primitives that can be employed to configure a wide range of different applications. There are two methods of RNG integration for FPGA applications, either through building mathematics models of RNGs using FPGA primitives which results in PRNGs [15] or by utilizing the random variation of the FPGA manufacturing process, thus creating TRNGs [15]. Although FPGA provides a sufficient level of randomness with high throughput for PRNG applications, the random number that it generates is no longer secure if the function behind the PRNG is compromised. This rest of this section focuses on the implementation of TRNG in FPGA.

There are two main components used to build the TRNG in FPGA: the source of randomness and the extractor. First, the source of randomness is the combination of the state space S and transition function f, as in (2-1). The transition function f is to prepare the state space S for generating the next random number.

An example of a digital circuit that can be used as a source of randomness for TRNG when implemented in FPGA is the shift-register. The shift register consists of a chain of flip-flops. With the FPGA from Xilinx, the shift-register can be simplified by utilizing the SRL16E [16] which is a particular mode of the LUT in Xilinx's FPGA. This configuration will significantly reduce silicon area usage.

The second component to build TRNG in FPGA is the extractor. One of the easiest ways to create an extractor function in FPGA is by using a comparator circuit to compare the quality of the two sources of randomness. If one source of randomness is better than the other, it will generate bit "1", and if it is the other way around, it will produce bit "0".

A simplified block diagram of TRNG for the implementation in FPGA is presented in Figure 2-2. The oscillation frequency of the two Sources of Randomness (SoR) will be counted by the binary counter and compared in a comparator circuit to generate the 1-bit random number. The 1-bit random number will be stored in a register and concatenated. After the first random number generation is finished, the finite state machine will tell the counter to start the SoR frequency counting routine again. In order to generate a 128-bit random number, it needs to run 128 times.

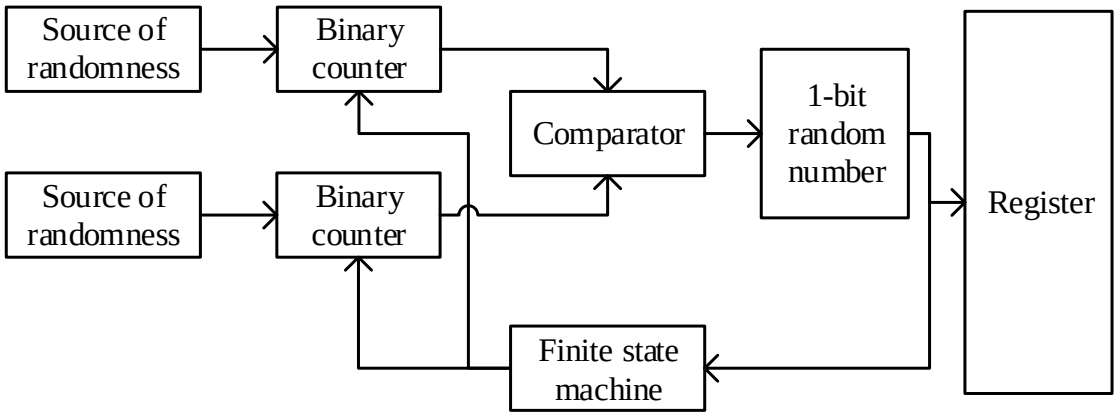

Figure 2-2: Block diagram of TRNG implementation

2.3.3 Test for Randomness

There are a couple of concepts that have been proposed in the literature to measure the quality of the bits produced by the random number generator. One of the parameters that can be easily tested with a non-statistical method is the frequency test. The frequency test aims to measure the uniformity of the bits produced by the RNG. An ideal uniform binary random number contains the same number of ones and zeros. This means that it has 50% uniformity while 100% uniformity means that the random number is made up of all ones or all zeros.

Another concept to measure the quality of a random number generator is by using a statistical test. There are a couple of suites used to test the randomness of an RNG such as DieHARD [17], FIPS140-2 [18], AIS-31[19], NIST SP800-22 [20], and TestU01 [21]. Table 2-2 shows the differences between each test suite.

Table 2-2: Minimum Input for Different RNG Test Suites

Name of the test suite	No. of test types	minimum input (bit)	Year of publication
DieHARD	18	2.5 million	1995
FIPS 140-2	11	16	2001
AIS-31	9	3 million	2001
TestU01	266	32 (max)	2007
NIST SP800-22	15	1 million	2010

Every random number test suite measures the p-value of the RNG. The p-value refers to the probability that the RNG under test will have the same quality as the referenced RNG used in the test suite. The p-value is chosen to represent the quality of an RNG to understand whether an RNG is good or bad. Nevertheless, it does not provide any information on which part of the RNG makes it a lousy RNG.

The p-value is compared to a significance level α, which is set by the tester. If the p-value is lower than α, then it means that the RNG is rejected as being a good

RNG. For example, the National Institute of Standard and Technologies (NIST) recommends setting the value of α to 1%. This means that there is a 1% probability that the RNG under test will be as good as the referenced RNG. However, as every RNG test suite is a statistical test, there are two types of error. Type I, also known as a false-positive error, happens when the test suite fails to detect a lower p-value of the RNG under test when it has a small p-value.

On the other hand, type II, also known as a false negative error, happens when the test suite fails to detect a higher p-value of RNG under-test when it has a high p-value. According to [17], a small p-value does not mean that the RNG is terrible. Instead, it tells us that there is a high chance of type II error which is more important from a practical point of view.

George Marsaglia published dieHARD in 1995 as an improvement of a random number of quality measurement techniques developed by Donald Knuth. His idea was to fix the p-value to a pre-chosen interval $[\alpha, 1 - \alpha]$. Beforehand, the p-value is not fixed, which makes it challenging to interpret the result of the randomness test.

The Federal Information Processing Standard (FIPS) 140-2 is a standard created by NIST in 2001. They also created a new standard called the NIST SP800-22. This is the latest tool used to quantify the quality of a random number generator. The difference between the two is that FIPS is more a qualitative way to standardize a random number generator. In contrast, NIST 800-22 is a more quantitative way to measure a random number generator. However, FIPS140-2 has been criticized by industries because it takes too long to get a random number certified. The certification process cannot be done by the creator of the random number themselves—it must be done by a third-party company.

AIS-31 is an improvised version of FIPS 140-1. It also introduces a new testing technique that focuses on how to measure the quality of a random number that has been post-processed. The tests that are included from FIPS 140-1 are the mono bit test, the poker test, the run test and the most extended run test. The other test used is the autocorrelation test, the uniform distribution test (which

includes 2 sub-tests), a comparative test for a multinomial test and the last one is the entropy test.

TestU01 is considered to be the most comprehensive test as it combines 266 test suites from the existing test suite available in the literature and commercial products. It divides the test into three packs: 1) "Small Crush" which consists of 10 tests, 2) "Crush" with 96 tests and 3) "Big Crush" which consists of 160 tests. However, it only able to handle 32-bit inputs which are too limited for modern RNGs in a cryptographic application.

In this experiment, NIST SP 800-22 Rev. 1a [16] will be used. It is a current standard that is widely used and accepted to measure the randomness of the random number generator. It consists of 15 statistical tests as described in Table 2-3. Every test has several parameters such as minimum bit length (n), block length (m or M), and several sub-tests. The number of n needs to be supplied by the user while m and M are parameters that can be set within the test suite.

Table 2-3: Statistical Tests Within NIST SP 800-22

TEST NAME	n	m or M
Frequency Test	$n \geq 100$	-
Frequency Test within a Block	$n \geq 100$	$20 \leq M \leq n/100$
Runs Test	$n \geq 100$	-
Longest-Run-of-Ones	$n \geq 128$	-
Binary Matrix Rank	$n \geq 38912$	-
FFT	$n \geq 1000$	-
Non-overlapping Template	$n \geq 8m-8$	$2 \leq m \leq 21$
Overlapping Template	$n \geq 10^6$	-
Maurer's Universal Statistical	$n \geq 387840$	-
Linear Complexity	$n \geq 10^6$	$500 \leq M \leq 5000$
Serial Test		$2 < m < [\log_2 n] - 2$
Approximate Entropy		$m < [\log_2 n] - 5$
Cumulative Sums	100	-
Random Excursions	$n \geq 10^6$	-
Random Excursions Variant	$n \geq 10^6$	-

NIST SP800-22 is widely used in industry and commercial RNG products because it is considered as having a low tolerance to error. Because of this, it is hard to pass the NIST SP800-22 unless the RNG is perfect. This claim is confirmed by [17], which mentions that high numbers of good RNG have difficulty passing 20% of the NIST test.

2.3.4 Metrics

Cryptography applications need a high rate of random number generation. The parameter used to measure the rate of the random number generation is known as the throughput. Throughput is calculated using (2-3).

$$throughput = \frac{n \times f_{max}}{latency} \tag{2-3}$$

where n is the number of bit-length of the generated random number, f_{max} is the maximum working frequency of the design, and $latency$ is the number of cycles taken to generate the 1 bit of random number. f_{max} is obtained by looking at the post-route-and-placement report of the FPGA and not the maximum frequency of the FPGA board.

2.4 Experimentation

2.4.1 Design of Ring Counter RNG (RCRNG)

The RNG based on the ring counter circuit will be implemented in the Kintex-7 FPGA development board, which consists of 7K325T FPGA from Xilinx. It utilizes 28 nm technology which is still widely used in critical systems nowadays, such as in avionics and radar technology. The Kintex-7 FPGA is categorized as a −2L device, means that it has a nominal voltage of 0.9 Volts. It is understandable that the threshold voltage for 28 nm devices is 0.4 Volts, and the experiment should

ideally run at that voltage level. However, this experiment uses the nominal voltage of the FPGA because lowering the voltage beyond the recommended voltage can harm the FPGA. While the differences between the nominal voltage used and the threshold voltage is an exciting topic to discuss, this experiment is focused on the effect of the non-uniform distribution of the process variation caused by the near-threshold voltage to the delay-based TRNG. This will leave the research on the environmental impacts, such as voltage and temperature differences, to others.

In this experiment, SRL16E will be used as the primary source of randomness for TRNG. The motivation behind it is to test the feasibility of upcoming silicon technology where the size of the transistor will become smaller. As stated in [16], LUT in SRL16E mode has very short wiring, so the delay should be slight enough to affect the timing or power consumption. This property will be used as the model for future delay-based TRNGs in FPGAs where the wiring is tiny. However, the differences in the delay are too small to be measured with today's technology. Therefore, in this experiment, the configuration of the shift-register from SRL16E for creating the ring counter is used. A ring counter is a shift-register with a feedback loop. The introduction of the loop will increase the delay to the measurable value of today's measurement technology.

The main component to build the RC-based TRNG is sliceM. It contains LUTs that can be programmed as a 16-bit shift-register in the form of SRL16E from the UNISIM library. By instantiating the LUT as shift-register, the resource usage of FPGA can be minimized.

The idea of using a ring counter as a source of randomness for TRNG is the same as the idea of using a ring oscillator to create a delay of a system clock. Two ring counters initialized as 10101010…. or 01010101010… will oscillate when activated. Depending on the process variation of the components used to create the ring counter, the oscillation frequency will be different from one ring counter to another. A 1-bit random number can be generated by comparing the frequency of two ring counters. In this experiment, the 16-bit ring counter was initialized only to have one bit of 1 and 15 bits of 0. This configuration was used to create a more

significant delay, so then the signal analyzer can easily see the difference in frequency. However, this configuration will increase the latency of the design and affect the overall throughput.

The TRNG is built based on the block diagram shown in Figure 2-2 without the finite state machine. This configuration is then stacked in parallel, as in Figure 2-3. The reason for this is that by using a parallel configuration, it is possible to generate an n-bit of random numbers in one run. This configuration also increases the confidence level of the measurement and the bit generation because it minimizes the effect of temperature and voltage change.

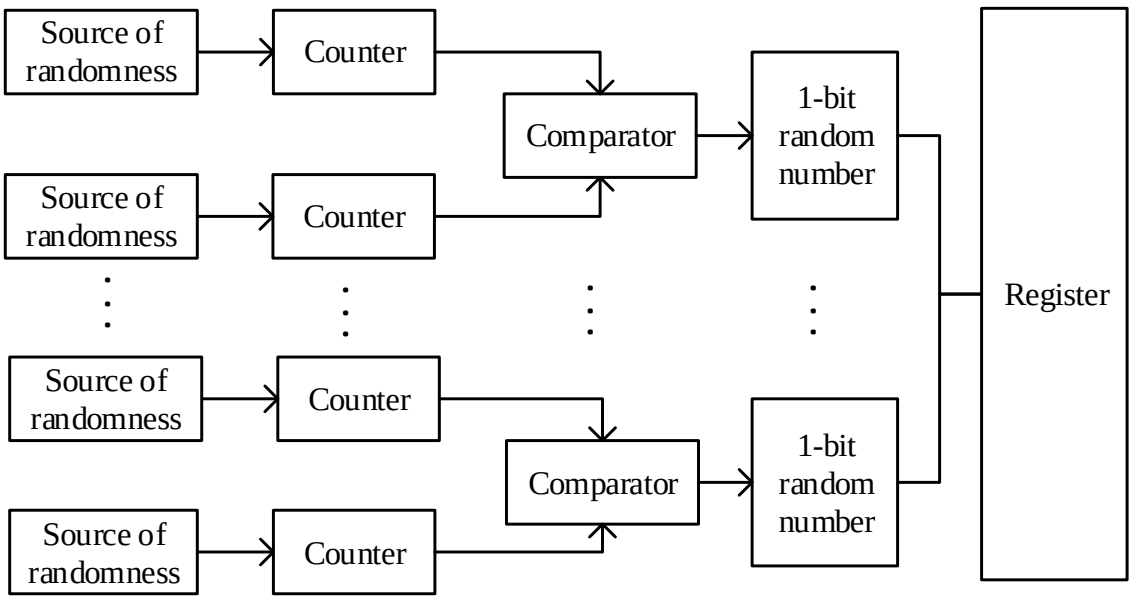

Figure 2-3: **Block diagram of TRNG implementation**

2.4.2 Experimental Limitation

Before implementing the RCRNG, there are a couple of things that need to be considered. The first is to find the location on the FPGA floorplan where the pair of RCs that will be compared can produce the best entropy for the random number generator. This can be done by inspecting every possible location in the FPGA floorplan. After that, every likely pair of RCs also needs to be checked to

find the best entropy. However, there are 16000 sliceMs in Kintex-7 7K325T FPGA. A single sliceM consists of 4 LUT that can be programmed as a four 16-bit shift register (SRL16E). Therefore, there are $2^4 = 16$ possible combinations on a single sliceM. Testing all of the possible combinations of all of the potential locations means to test 256.000 possible combinations, which will be time-consuming. Therefore, some constraint needs to be applied to the experiment by limiting the number of RC pairs that will be tested as follows:

1. The test will only be done by comparing the neighbouring LUTs on the same slice. This makes it only possible to compare two pairs of LUT per sliceM.
2. To acquire the data, an integrated logic analyser, in this case, Chipscope Pro 14.7, was used. Even though it is a powerful tool to debug the circuit design of FPGA, there are some practical limitations. For Kinetix-7 FPGA, the maximum number of signals that it can read at a single time is 4096. Hence in order to test all the possible pairs by applying the constraint on point 1), the measurement needs to be done (16 x 16000)/4096 times, or about 62 times, which is a time-consuming process. For this reason, the test will be limited to as close as to the maximum number of signals of Chipscope as possible, which is 4000 RC pair. Each pair will be captured 1000 times in order to be able to understand the uniformity of the ring counter pair.
3. The process of placement will be done manually by applying the location constraint to the ring counter pair and the relative location constraint to the counter, so then it is located close to the ring counter.

Even though there are some limitations in this experiment, it still gives a clear idea about the steps needed to find the best location for the RC pair to generate a random number with the best entropy. The flowchart in Figure 2-4 is given for a better understanding of the location selection process.

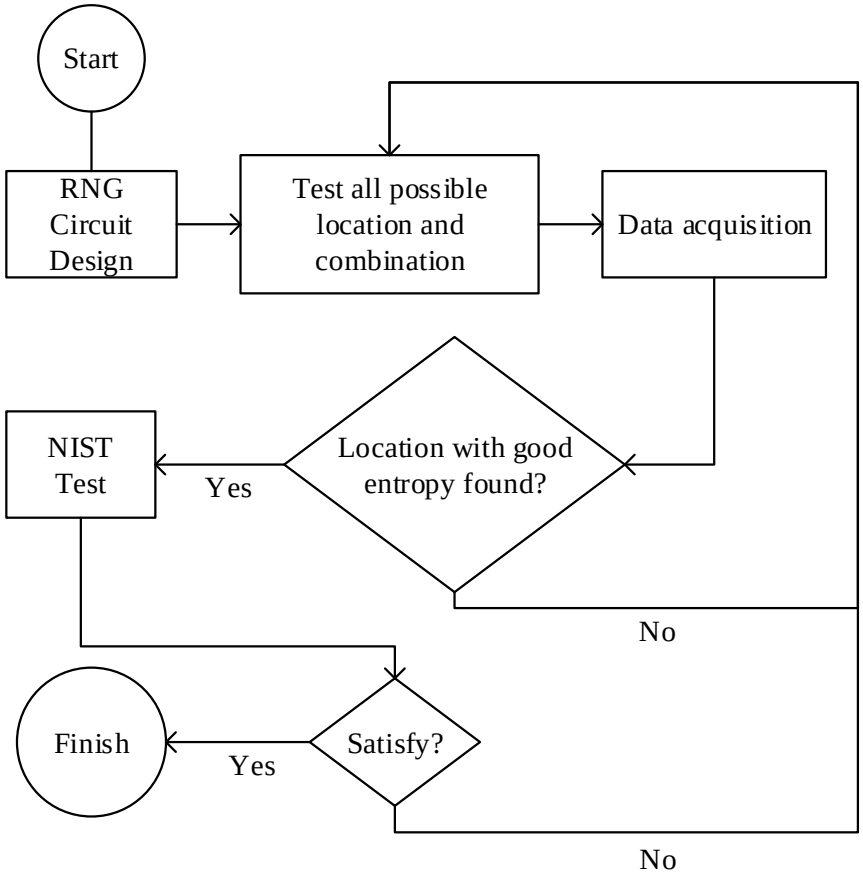

Figure 2-4: Location selection flowchart

Secondly, concerning the more technical aspect of the design, the delay in every part of the circuit needs to be the same up to the counter logic. In FPGA, the delay on the ring counter is not a problem, and it is assumed to be the same. This is because, in FPGA, the ring counter was made by instantiating a LUT which means that no wiring is needed to connect the component that builds the ring counter. However, it is a bit of a challenge to make sure that the delay between the ring counter and the counter is the same. This is because the manual routing tool from ISE is complicated to use. Therefore in this experiment, the delay from the ring counter to the counter is made as small and as similar as possible by forcing the placement of the counter to be as close as possible relative to the ring counter circuit.

Lastly, it is desirable to create a hard macro of the RCRNG circuit (at least from ring counter to counter circuit) to fix the location, to lessen any delays between the components and to make sure that there is no additional logic inserted into the circuit. However, it remains a big challenge for FPGA designers to create a hard macro from an instance that has an initialization value in one of the components of the hard macro. In this case, the ring counter circuit needs to have an initial value which will have consequences on the presence of a power net. In ISE 14.7, the tool does not accept any power nets inside a hard macro. In this experiment, to make sure that there is no additional logic added to the path between the ring counter and the counter, they need to be forcibly located as close as possible relative to the ring counter circuit. This can be done by using `rloc` constraint.

2.5 Findings and Analysis

From the 4000 pairs of the ring counter, the uniformity of the bits generated from each RC pair can be calculated. In Figure 2-5, the RC pair that have 100% uniformity is not shown to clarify the graph. Perfect uniformity in a bit string is reached when the number of ones is the same as the number of zeros, indicating that the uniformity is 50%. From Figure 2-5, there are 45 RC pairs that have precisely 50% uniformity. However, the initial design was to create a 128-bit random number. Therefore another run of tests is needed to find the RC pairs that have 50% uniformity. After undertaking the process for another two times, 32 and 55 RC pairs were found after the second and third location finding process, as shown in Figure 2-6 and Figure 2-7. In total, 132 locations with 50% of uniformity were found which is sufficient to build the 128-bit RNG.

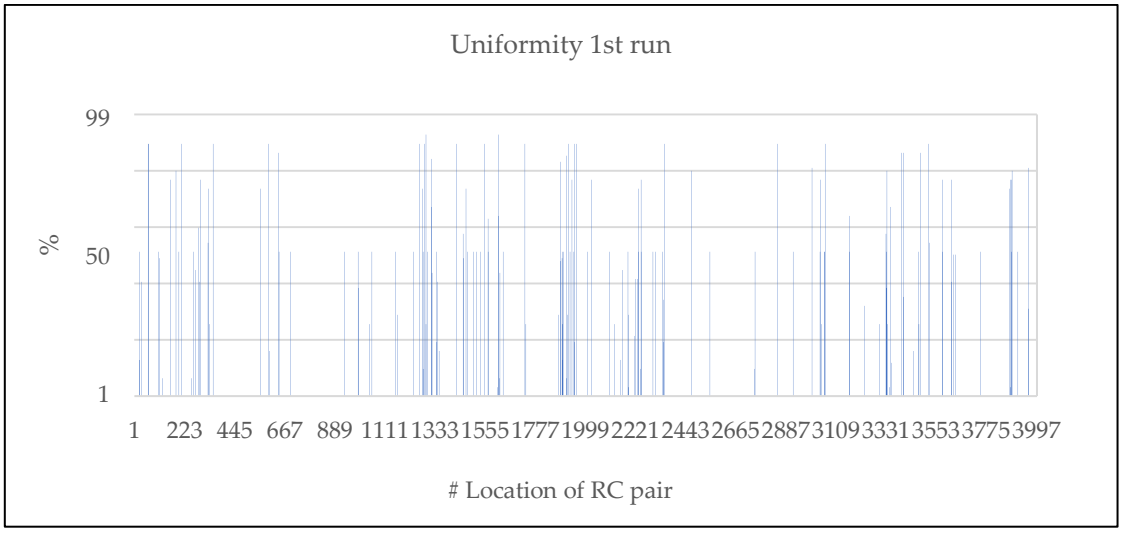

Figure 2-5: Uniformity of the RC pairs on the first run

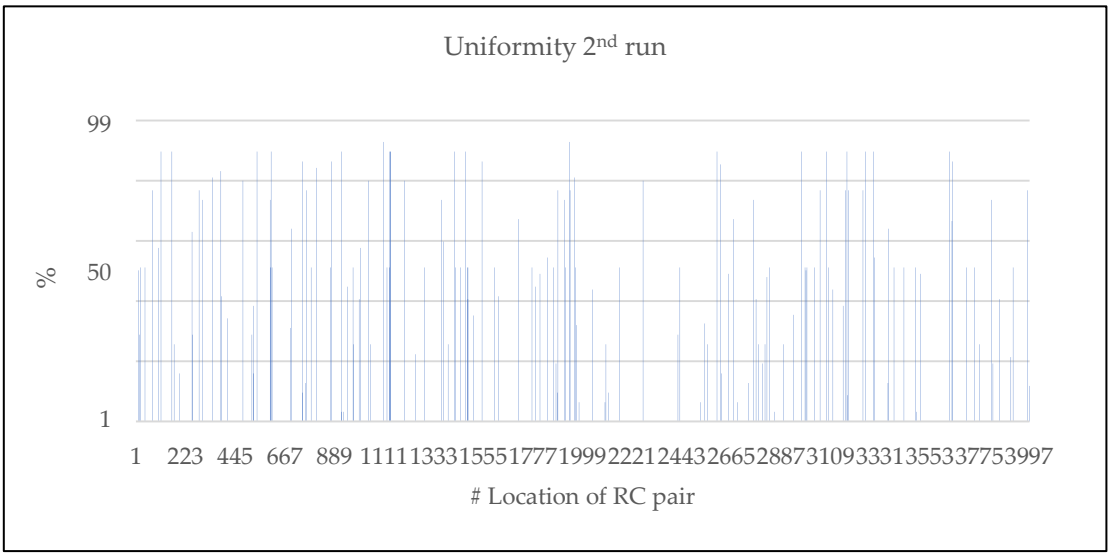

Figure 2-6: Uniformity of the RC pairs on the second run

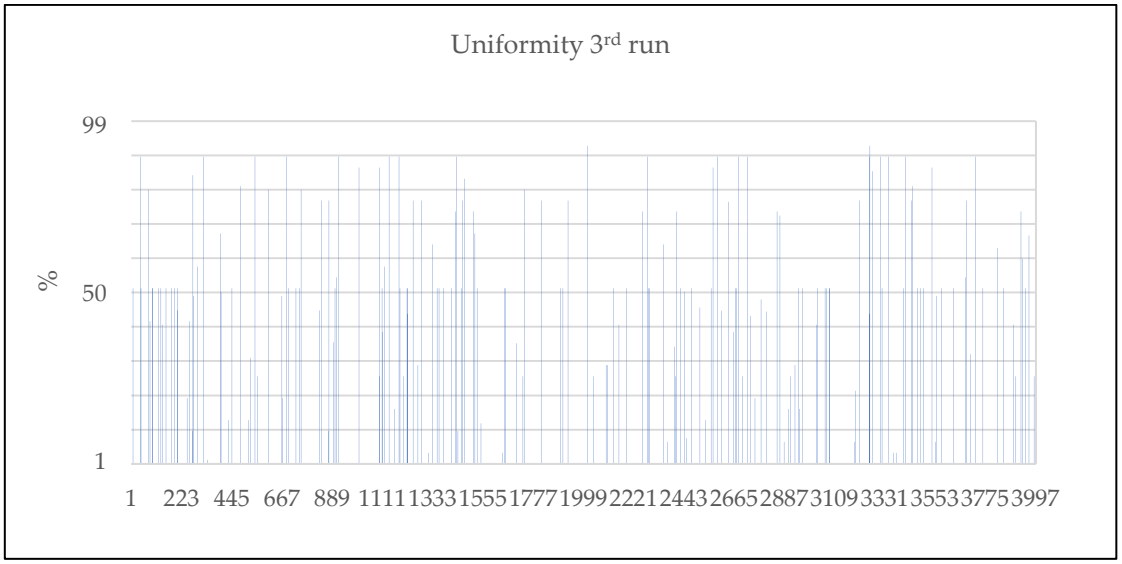

Figure 2-7: Uniformity of the RC pairs on the third run

NIST SP800-22 was used to measure the quality of the random number. Because some of the tests in the NIST test suite need at least 10^6 bits of data, at least 10000 bitstreams are required for a 128-bit RNG. This will translate into sequence length (128) and bitstreams (10000) for the input of the test suite. Raw data from the pre-selected RC pair is fed into the test suite, and the results are as shown in Table 2-4. The first ten columns are ten bins from 0 to 1. What is in the bin is the p-value that falls within the range of that bin. For example, 3024 in the first row and the first column means that there is a 3024 p-value that has a value between 0 to 0.1 in the frequency test. For each experiment, the optimum result is achieved when the p-value is distributed uniformly across all bins. The p-value column is the uniformity of the p-value. The χ^2 test determines the uniformity of the p-value. The optimum value for the uniformity of the p-value is 1. However, according to the guideline of the NIST test suite, it mentions that the minimum value of 0.01 for the uniformity of the p-value is enough for the RNG under-test to pass each test.

The NIST test, however, has a rigorous rule where the recommended significance level is between 0.1% - 1%. This means that it will only tolerate an error of 1%. For example, if the number of p-values that falls within a bin is outside of the

range of ±1%, then it will be considered an error and will fail the χ^2 test. As an example, in this experiment, 10000 bitstreams were produced. If the bitstreams are divided into ten bins, each bin should have a 1000±1% p-value fall into it. According to Table 2-4, none of the bins satisfies this rule. Therefore when the program calculates the uniformity of the p-value, it will give error `igamc: UNDERFLOW`. This means that the calculated uniformity of the p-value is too small. This is the reason why the value of the p-value column is all zeros.

Table 2-4: NIST SP 800-22 Test Results

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-value	Statistical test
3024	1067	1395	728	780	841	0	850	846	469	0	Frequency
3024	1067	1395	728	780	841	0	850	846	469	0	Block Frequency
2903	1247	782	898	948	586	554	541	925	616	0	Cumulative Sums
1092	1019	1104	1041	905	1070	764	1046	1091	868	0	Runs
2208	1227	1013	995	827	1030	610	800	646	644	0	Longest Run
10000	0	0	0	0	0	0	0	0	0	0	Rank
1156	1292	0	1987	0	2566	0	0	2999	0	0	FFT
1503	0	0	0	0	0	0	0	0	8497	0	Nonoverlapping Temp
10000	0	0	0	0	0	0	0	0	0	0	Overlapping Template
0	0	0	0	0	0	0	0	0	0	0	Universal
0	0	0	0	0	0	0	0	0	10000	0	Approximate Entropy
0	0	0	0	0	0	0	0	0	0	0	Random Excursions
0	0	0	0	0	0	0	0	0	0	0	Rand Excursions Var
1380	0	0	0	7654	0	0	0	0	966	0	Serial
10000	0	0	0	0	0	0	0	0	0	0	Linear Complexity

Aside from using the χ^2 test, the NIST SP 800-22 also suggests another way to analyze the uniformity of the p-value, which is using a graphical plot of the p-value. Figure 2-8 has been given to investigate the results of the NIST further. Figure 2-8 is the representation of Table 2-4 in graph format. To clarify the graph, the maximum range of the y-axis is limited to 1500. In this graph, most of the p-values fall outside the 1000±1% range but not by much. The only bin that falls way over the tolerance range is C1. This result is an indication of a type II error, where most of the p-value falls into the low bin. By looking at the proportion of

sequences that pass the test, even though it falls below the tolerance range of 9900/10000, the result is not bad at all. As already discussed in [17], even the built-in PRNG of NIST SP800-22 has only a 15% probability of passing all tests. Therefore, it can be said that the result of this experiment does not mean that the RNG fails to produce an excellent random number but rather a type II errors in the statistical measurement which is sometimes more useful from a practical point of view.

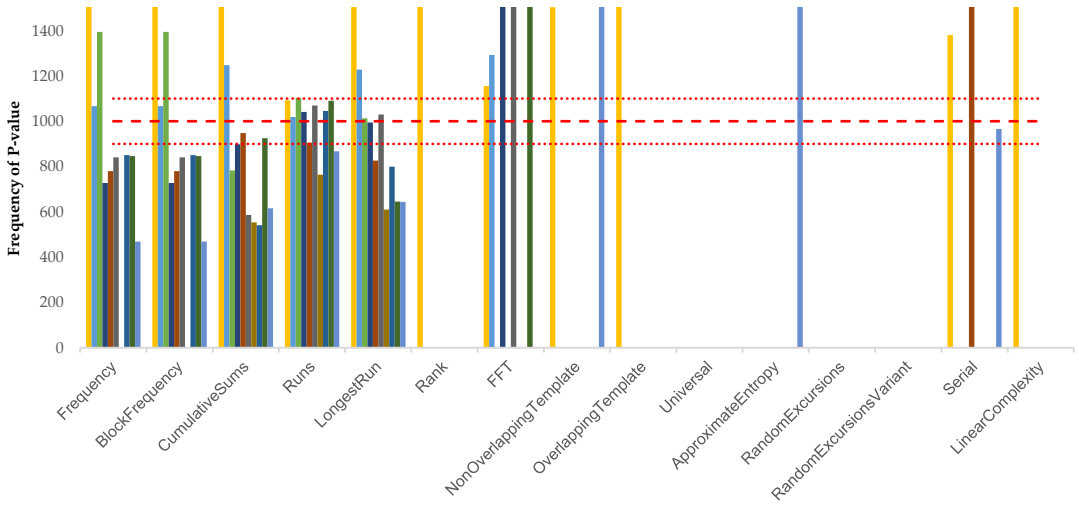

Figure 2-8: Graphical presentation of the NIST test result

The results from Table 2-4 leads to the suspicion that something is not right with the NIST test suite because of the parametrical error. Therefore, a test using the NIST built-in PRNG was done to verify that the NIST test suite is working as it is intended to be. In this case, the built-in PRNG that was used is the linear congruential generator. The test was run using 1 million, 10 million, 100 million and 1000 million bits to see the effect of the number of input bits on the results of the test. The results have been shown in Figure 2-9. When the NIST test suite is fed with the minimum input recommended by the standard, it did not return any meaningful data. The uniformity is not valid using the minimum input even though the generator under test is from the built-in PRNG. When the number of input bits

increased, the uniformity of the p-value improves. When 100 million and 1000 million input bits were given, the test result returned the same p-value and the same uniformity of p-value as well. From this test, it can be concluded that in order to get a meaningful result from the NIST test, a more significant number of bits is needed than the recommended minimum input bit mentioned on the standard.

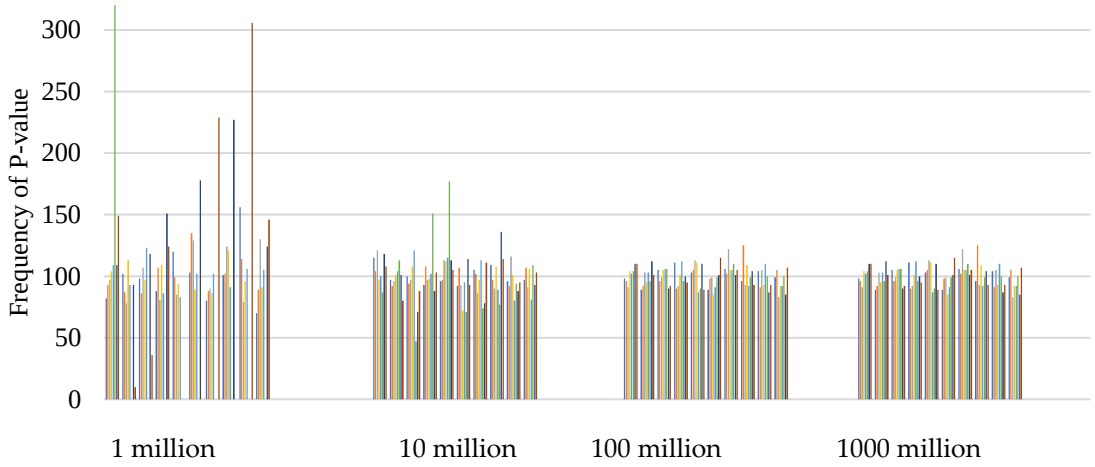

Figure 2-9: Comparison of the different input bit lengths in the NIST test

Based on this finding, another NIST test was conducted using the experimental data. This time, 10 million bits were used as the input of the test suite and this number was increased to 100 million to see the effect of increasing the number of input bits and how it relates to the output obtained from the NIST test. First, 10 million bits was divided into 1000 sequences with a length of 10000 each. The result can be seen in Table 2-5. It shows that there is an improvement in the uniformity of the p-value as expected. However, when the number of input bits was increased to 100 million, the test returned an error message saying that the number of bits is insufficient. The same error message also reported by [18]. Nevertheless, the result from Table 2-4 and Table 2-5 agree with the trend in Figure 2-9. This means that despite the inability to acquire the results for the NIST

test with a higher input bit, the RCRNG can pass the NIST test when it is tested with a larger input bit.

Table 2-5: NIST test results with 10 million input bits

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-VALUE	STATISTICAL TEST
535	82	83	55	44	41	54	39	30	37	0	Frequency
1000	0	0	0	0	0	0	0	0	0	0	Block Frequency
772	105	57	31	18	9	7	0	1	0	0	Cumulative Sums
922	18	8	9	12	8	7	2	7	7	0	Runs
706	111	60	38	28	24	8	11	6	8	0	Longest Run
104	150	78	103	172	54	66	86	96	91	0	Rank
614	124	46	46	33	31	30	34	15	27	0	FFT
102	85	95	123	91	94	124	96	105	85	0.042531	Nonoverlapping Template
611	165	30	45	51	32	25	15	16	10	0	Overlapping Template
0	0	0	0	0	0	0	0	0	0	0	Universal
1000	0	0	0	0	0	0	0	0	0	0	Approximate Entropy
0	0	0	0	0	0	0	0	0	0	0	Random Excursions
0	0	0	0	0	0	0	0	0	0	0	Random Excursions Variant
999	1	0	0	0	0	0	0	0	0	0	Serial
116	72	100	83	102	119	100	100	112	96	0.029401	Linear Complexity

Another discovery from the test result, as shown in Table 2-4 and Table 2-5 is the distribution of the p-value from the experimental data that gravitate towards the smaller p-value (column C1). This can be interpreted as one indication of small periodicity. The reason for this phenomenon might come from the non-gaussian distribution of the process variation of the NTV devices. It can be concluded that when the SRL16E was used as a model source of randomness for RNG in sub-nano millimetre electronics, the RNG can still perform well but with small periodicity.

Using the XPower Analyzer tool from Xilinx, the estimated power consumption is 0.157 Watts. Table 2-6 presents the resource utilization and throughput of the proposed design compared to the other TRNG implementation in FPGA. Based on the post PAR (Placement and Route) analysis, the maximum frequency for

this design is 74 MHz. In this experiment, the latency is 16×16 because of the initialization of the ring counter. Therefore the throughput of the design is calculated as 37 Mbps using equation (1). From Table 2-6, it appears that the RC-based TRNG has the right balance between FPGA resource utilization and its throughput.

Table 2-6: Throughput comparison between the TRNG implementation in FPGA

RNG Type	Resource utilization (LUT)	Throughput (Mbps)
PLL [19]	6144	69
Ring Oscillator [20]	3968	13.8
Metastability [21]	8960	50
Chaotic Oscillator [22]	43732	58,76
Ring Oscillator [23]	7296	4.77
Ring Counter	2048	37

The throughput can be increased by increasing the number of ones at the initialization stage of the ring counter. It can be increased up to 0.6 Gbps when all of the bits on the ring counter are initiated as ones. However, there is a drawback to this. The faster the ring counter overflows the frequency counter, the harder it is for the comparator to see any differences in frequency. It will think that the frequency of the two ring counter is the same, and it will generate the same bit every time. For the application of a random number generator, this property is unwanted. However, for the application of a physically unclonable function, this configuration will create a more stable bit generation which is preferred by many researchers.

2.6 Conclusion

In this chapter, a random number generator based on the ring counter circuit has been implemented in FPGA. The framework for the construction process was

described as well as the process of location selection to get the best randomness out of the ring counter pair. Because of the limitation of the IDE tools used, there are a couple of practical limitations in the experiments. This limitation has been explained thoroughly and overcome. The evaluation using the NIST SP800-22 statistical suite was also presented, and the results have been discussed thoroughly. One comment for the NIST test suite is that one needs to have a significant input a bit beyond its recommended minimum input to get a meaningful result.

In terms of the adaptability of delay-based RNG for sub-nano millimetre technology, it is shown that the current delay-based RNG can still be implemented. Even though the path delay is small and negligible, there are still some differences in delay or frequency that can be extracted to construct a random number generator. However, one should take note that the periodicity of delay-based RNG in the sub-nano millimetre will be small.

2.7 References

[1] V. Chellappan and K. M. Sivalingam, “Security and privacy in the Internet of Things,” in *Internet of Things*, Elsevier, 2016, pp. 183–200.

[2] J. Dofe, J. Frey, and Q. Yu, “Hardware security assurance in emerging IoT applications,” in *Proceedings - IEEE International Symposium on Circuits and Systems*, 2016, vol. 2016-July, pp. 2050–2053.

[3] S. Satpathy *et al.*, “An All-Digital Unified Static/Dynamic Entropy Generator Featuring Self-Calibrating Hierarchical von Neumann Extraction for Secure Privacy-Preserving Mutual Authentication in IoT Mote Platforms,” in *IEEE Symposium on VLSI Circuits, Digest of Technical Papers*, 2018, vol. 2018-June, pp. 169–170.

[4] V. R. Pamula, X. Sun, S. Kim, F. Ur Rahman, B. Zhang, and V. S. Sathe, “An All-Digital True-Random-Number Generator with Integrated De-

correlation and Bias Correction at 3.2-to-86 MB/S, 2.58 PJ/Bit in 65-NM CMOS," in *IEEE Symposium on VLSI Circuits, Digest of Technical Papers*, 2018, vol. 2018-June, pp. 173–174.

[5] S. Mathew *et al.*, "μrNG: A 300-950mV 323Gbps/W all-digital full-entropy true random number generator in 14nm FinFET CMOS," in *European Solid-State Circuits Conference*, 2015, vol. 2015-October, pp. 116–119.

[6] IRDS, "International roadmap for devices and systems 2017 edition," 2018.

[7] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253–266, Feb. 2010.

[8] A. S. Mutschler, "Near-Threshold Issues Deepen." [Online]. Available: https://semiengineering.com/near-threshold-issues-widen/. [Accessed: 20-Jan-2020].

[9] J. Zhou, T. T. H. Kim, and Y. Lian, "Near-threshold processor design techniques for power-constrained computing devices," in *Proceedings of International Conference on ASIC*, 2017, vol. 2017-October, pp. 920–923.

[10] D. B. Thomas and W. Luk, "FPGA-Optimised Uniform Random Number Generators Using LUTs and Shift Registers," in *2010 International Conference on Field Programmable Logic and Applications*, 2010, pp. 77–82.

[11] D. B. Thomas and W. Luk, "The LUT-SR Family of Uniform Random Number Generators for FPGA Architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 4, pp. 761–770, Apr. 2013.

[12] H. Kaul, M. Anders, S. Hsu, A. Agarwal, R. Krishnamurthy, and S. Borkar, "Near-threshold voltage (NTV) design: Opportunities and challenges," in *Proceedings - Design Automation Conference*, 2012, pp. 1153–1158.

[13] K. B. Cook and A. J. Brodersen, “Physical origins of burst noise in transistors,” *Solid State Electronics*, vol. 14, no. 12, pp. 1237–1242, 1971.

[14] 1930- McWhorter, Alan L. (Alan Louis), “1/f noise and related surface effects in germanium,” 1955.

[15] F. N. Hooge, “1/f noise is no surface effect,” *Physics Letters A*, vol. 29, no. 3, pp. 139–140, Apr. 1969.

[16] A. Rukhin *et al.*, “Special Publication 800-22 Revision 1a A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications,” Apr. 2010.

[17] “On the interpretation of results from the NIST statistical test suite.” [Online]. Available: https://www.researchgate.net/publication/287224641_On_the_interpretation_of_results_from_the_NIST_statistical_test_suite. [Accessed: 01-Oct-2019].

[18] Terry Moore, “GitHub - terrillmoore/NIST-Statistical-Test-Suite: The code from NIST SP-800-22 for testing random-number generators, along with docs for reference.” [Online]. Available: https://github.com/terrillmoore/NIST-Statistical-Test-Suite. [Accessed: 29-Mar-2020].

[19] V. Fischer and M. Drutarovský, “True random number generator embedded in reconfigurable hardware,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 2523, pp. 415–430, 2003.

[20] M. Dichtl and J. D. Golić, “High-Speed True Random Number Generation with Logic Gates Only,” in *Cryptographic Hardware and Embedded Systems - CHES 2007*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 45–62.

[21] I. Vasyltsov, E. Hambardzumyan, Y. S. Kim, and B. Karpinskyy, “Fast

digital TRNG based on metastable ring oscillator," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008, vol. 5154 LNCS, pp. 164–180.

[22] İ. Koyuncu and A. Turan Özcerit, "The design and realization of a new high speed FPGA-based chaotic true random number generator," *Computers and Electrical Engineering*, vol. 58, pp. 203–214, Feb. 2017.

[23] E. A. , M. T. , A. B. O. Taner Tuncer, "Implementation of Non-periodic Sampling TrueRandom Number Generator on FPGA," *Journal of Microelectronics, Electronic Components and Materials*, vol. 44, no. 4, pp. 296–302, 2014.

3 A HIGH-SENSITIVITY SENSOR FOR THE DETECTION OF UNAUTHORISED MODIFICATIONS OF FPGA CONFIGURATION BASED ON A PHYSICALLY UNCLONABLE FUNCTION

3.1 Abstract

Although the application of a Field Programmable Gate Array (FPGA) as in multi-tenant Cloud-based services adds to the affordability of high-speed computation, it demonstrates emerging security problems such as virtual machine attacks. A malicious tenant makes its way into the other tenant's FPGA estate using the vulnerability of the lengthy wires used in a Cloud-based FPGA. As a consequence, there is a thread of FPGA modification created by altering the configuration files for inserting a Hardware Trojan. This is known to be a source of confidential data leakage or malicious behaviour. This chapter proposes the implementation technique of a novel physical unclonable function that facilitates the early detection of unauthorised modifications to the FPGA configuration file. The proposed security measures are proven to have a high level of sensitivity to small changes, and they have excellent resistance to physical tampering and ageing.

3.2 Introduction

The demand for high-speed and affordable computation is some of the moving factors behind the rise in multi-tenant Field Programmable Gate Arrays (FPGAs) Cloud services such as from Amazon [1] and Microsoft [2]. The genomic research, big data analytics, and real-time video processing are some of the applications that benefit from this kind of service. Each tenant of the multi-tenant FPGA Cloud service shares the same hardware resources within their Virtual Machine (VM). The virtualisation was done to prevent an intervention between each tenant.

While it offers a new possibility for fast computing, this scheme exposes the systems to advanced security issues such as data leakage from long wires using

side-channel attacks [3]–[5]. Moreover, the FPGA's ability in runtime reconfiguration makes it possible for malicious tenants to modify the configuration file of other tenants without their consent. Such an attack, which is also known as a VM escape attack [6], compromises the host's physical machine when it comes to gaining access and control over the other VMs. It is sub-classed as a VM hopping attack [6] when it occurs in the VM in the same physical machine. Alternatively, it is sub-classed as a mobility attack [6] where a compromised physical machine with a malicious VM is moved to other systems. In such a case, the malicious VM can infect the other VMs in the new location as well.

Solutions to overcome the security shortcomings presently include bitstream encryption to preventing reverse engineering [7], authentication for the bitstream integrity check [8], and Physically Unclonable Function (PUF) for device identification [9]. While the encryption and authentication already a standard in modern FPGA, the industry still reluctant to integrate the PUF into the FPGA. The problem comes from the instability of the key generated by PUF, which requires additional care toward building error correction algorithms. Additionally, PUF is yet susceptible to physical deterioration which will affect its performance in the long run. However, Such shortages are negligible for less critical implementation, such as device identification.

PUF extracts the imperfection of the Integrated Circuit (IC) manufacturing process and utilises it for security purposes, such as in the development of a random number generator [10], encryption/decryption key [11], and device identification [12]. There has been ongoing research regarding the development of PUFs for device identification (ID) in FPGA. Gu [13] proposed a PUF model for device identification using D-flipflop and cross-coupled NAND gates. He also presented an error correction mechanism to improve the reliability of the generated ID. Ring oscillators, as one of the most popular primitives for PUF design, were also recommended to create a device ID for FPGAs. Haile [14] proposed a way to increase the reliability of the PUF response by implementing a reconfigurable ring oscillator to reduce the temperature and voltage interference. While a reliable PUF for device ID generation has the advantage of

identifying a device, it is not suitable to be used as a detector of unauthorised modifications to the FPGA configuration file that can create a deviation in the ID. The reason for this is that the introduction of an error correction mechanism will force the deviate ID into the correct one. A slight change in ID, which is useful information to detect modifications to the FPGA configuration file, will thus be discarded. So, in order to utilise the PUF as a detector of unauthorised modifications to the FPGA configuration file, the presence of an error correction algorithm is not necessary. What is important is how to implement the PUF, so then it becomes sensitive to a slight change in the FPGA configuration file.

An on-board digital sensor needs to be implemented to detect unauthorised modifications to the FPGA configuration file, such as the insertion of a Hardware Trojan. Kitsos [15] presented the utilisation of the Transient Effect Ring Oscillator (TERO) as a sensor to detect Hardware Trojan in FPGAs. Although the sensitivity of such sensors can be increased by adjusting the length of the TERO, the proposed method cannot alleviate the need for a golden reference which is known as an excellent and Trojan-free FPGA. Techniques involving side-channel analysis allow for the detection of Hardware Trojans without the need for golden references as presented by Fournaris [16]. He combined the side-channel study with a logical test suite to trigger the hardware Trojan. Furthermore, he developed an array of ring oscillators as a digital sensor to detect the presence of Hardware Trojan without a golden reference. While offering excellent performance, the proposed method uses a significant amount of FPGA resources.

Traditionally, it is the PUF's pure response that is directly used as a measure for the detection of the FPGA modifications induced by Hardware Trojans. However, there are various parameters including uniqueness, reliability, throughput and so on that are suggested for characterising the PUF and analysing its performance. Some of these parameters have the potential to provide a vision in relation to the detection of malicious activities. In respect to the uniqueness of PUFs, most of the research is directed toward the use of inter-chip distance. For instance, by measuring the Hamming distance between the PUF responses among the different implementations using a variety of devices [17]. When the Hamming

distance is close to 50%, it can be concluded that the PUF has an ideal inter-distance parameter. However, this parameter is a bit misleading when it comes to measuring the quality of PUF because each device has a manufacturing flaw that is different from the others. Therefore when the same PUF design is implemented in two identical apparatus, the responses will likely be different from one to another because the devices have different random process variations. Hence the racing condition that occurs in the individual devices also varies.

Reliability is one of the critical parameters used to detect the unauthorised modification of FPGA configuration files. The majority of the literature defines reliability as an intra-chip distance parameter used to measure the efficiency of reproducing the response bits. Hamming distance is employed to evaluate the consistency of the generated responses against the changes that are due to varying operating conditions; for instance, changes in the subject in terms of supplying voltage fluctuations and temperature. However, this definition prevents the reliability measurement from being done while the FPGA is on a mission mode. This is as it is too risky to change the voltage level, and it plays with the environmental temperature while in mission mode. There is an urge to have a new definition to measure the reliability of the PUF response, so then it can be used in mission mode and so then it can be utilised as a way to detect unauthorised modifications to the FPGA configuration file.

From the literature, the challenges associated with the malicious modification detection of an FPGA configuration file are known to be a) difficulties related to acquiring a golden reference, b) the limitation of the FPGA resources, and c) the practicality of the implementation and interpretation of the detection technique. Therefore it is of the utmost importance to develop strategies for detecting Hardware Trojans and their associated malicious modifications, which should be efficient and straightforward for both implementation and computation.

In this regard, we hypothesise that the average reliability of the PUF can be used as a measure for the detection of FPGA modifications instead of the conventional techniques that employ the PUF pure response directly as a measure. As an advantage, it eliminates the need for error correction algorithms. Moreover, the

proposed method will not suffer from either physical deterioration or the ageing process of the FPGA. Building on this hypothesis, this chapter presents an implementation of a highly sensitive PUF-based device identification and employs it for the detection of malicious modifications in the FPGA. It requires building a novel PUF configuration architected as a Ring Counter-Based PUF which is constructed based on the SRL16E mode of Xilinx's LUT to simplify the overall FPGA implementation flows. The proposed architecture is then tested to characterise its various parameters and metrics, including uniqueness, reliability, and throughput. The results from the experiment indicate that the change in a single logic gate can be detected using the proposed mechanism. With the goals mentioned earlier, this chapter makes the following contributions:

- First, a high-sensitivity digital sensor based on the Ring Counter PUF (RCPUF) is developed to detect unauthorised modifications of multi-tenant FPGA configuration files.
- A new definition of uniqueness parameter is then proposed for a better understanding of the characteristics of the PUF in a device without the need for external references.
- A new definition for the reliable measurement of PUF is proposed so then it can be used for the detection of unauthorised modifications to the FPGA configuration files without affecting the mission mode of the FPGA.
- Finally, The PUF will be characterised to measure their performance against the parameters of uniqueness, reliability, and throughput.

The rest of the chapter is organised as follows. Section 3.3 will discuss the related works on PUF implementation, classification, and characterisation. The proposed PUF was constructed using the SRL16E mode of Xilinx's LUT along with the required experimental setup presented in sections 3.4 and 3.5, respectively. Furthermore, the results and findings will be discussed in section 3.6. Finally, the chapter will be concluded in section 3.7.

3.3 Related Works

3.3.1 Physically unclonable function

The idea of a PUF initially was introduced by R. Pappu in 2008 [18] as a solution to overcome problems in number theory-based one-way functions by transferring the medium's microstructure to a fixed-length string of binary digits. He proposed PUF as a "physical one-way function" for modern cryptographic practices. In a digital circuit, creating a PUF is done by converting the physical imperfections of the manufacturing process within an integrated circuit into a useful binary digit of a fixed-length. Referring to this idea, the PUF is a fixed-length binary digit that comes from a continuous flow of random binary digits, known as a Random Number Generator (RNG). The binary digits of the RNG are extracted as a subset of Ω from an initial entropy source S. The set S is preconditioned by the mapping function *f* to be ready for extraction using the extraction function g, and so it becomes a new random space *U* as shown in Figure 3-1.

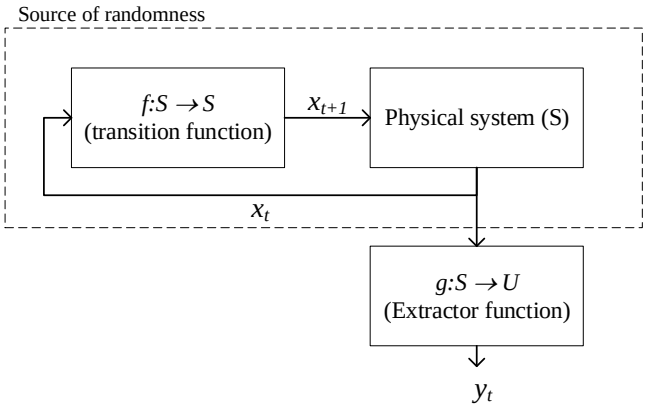

Figure 3-1: Block diagram of a random number generator

In a digital circuit, *S* is the physical source of randomness (SoR), and g is the logic used to process the SoR further. Like all digital circuitry, the SoR also needs an initial condition. It can be in the form of the current state of the physical system or the need for it to be given externally by a seed x_0. Figure 1 illustrates this mechanism.

There are different ways to classify PUFs. One of them is by looking at how the response is generated, e.g., weak PUF and strong PUF. Weak PUF [19] is a PUF where the generation process is done purely by utilising the random variation of the devices. Because of this, this type of PUF tends only to produce a few usable responses, i.e., a response with high reliability, and in some cases, only one usable response. On the other hand, the response generation of a strong PUF [20] incorporates mathematical algorithms to process the random variation of a device further so then the chance to get a more usable response is higher.

Figure 3-2 provides a high-level schematic of PUF implementation in an integrated circuit. The source of randomness can be in the form of a single physical entity such as an oscillating crystal or in the form of a digital circuit such as a ring oscillator.

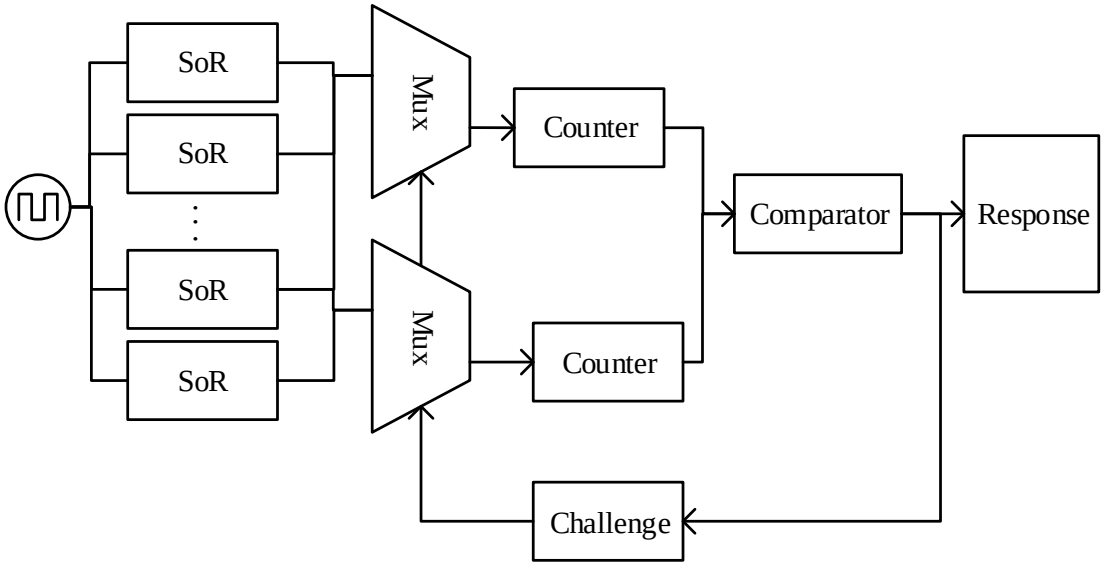

Figure 3-2: High-level schematic of delay-based PUF implementation

Table 3-1: Various types of PUF

Application	Source of Randomness	PUF name	Advantage	Disadvantage
Non-Silicon	Optical PUF		Large CRP space	Bulky
Silicon	Time-delay based	Arbiter PUF	Large CRP space	Vulnerable to modelling attacks
		Ring oscillator PUF	Flexible implementation	Low reliability
		Ring counter PUF		
	Intrinsic PUF	SRAM PUF	High reliability	Limited CRP space
	Mismatch-based	Latch PUF	Imitate the behaviour of SRAM PUF for devices without SRAM	
		Flip-flop PUF		
		Butterfly PUF		
	Reconfigurable PUF	Physically reconfigurable	Useful for key renewal and revoke	Vulnerable to DoS attacks
		Logically reconfigurable		

Table 3-1 shows the various types of PUF. A pair of PUF challenges with their corresponding response is called a challenge-response pair (CRP). While having the advantage of large CRP spaces, the characterisation of the optical PUF [21] needs a bulky external device which makes it less practical and costly to produce. A time-delay-based PUF such as the arbiter PUF [22], ring oscillator PUF [23], and ring counter PUF have an advantage in that they are very flexible in terms of implementation. What it means is that their architecture can be applied to any electronic device ranging from an Application Specific Integrated Circuit (ASIC)

to a Field Programmable Gate Array (FPGA). However, the reliability of this type of PUF is lower compared to the other type of PUF. The reason for this is because their flexible architecture needs to be specifically tailored for every implementation. This makes it hard to get a level of acceptable reliability without any iterative implementation process. The intrinsic PUF is based on primitives that are already available in the device such as the Static RAM (SRAM) PUF [24] without having to configure it. Because of their rigid structure, this type of PUF tends to have high reliability. Researchers try to imitate the behaviour of SRAM in the devices that do not have the same primitive behaviour according to the development of mismatched-based PUFs such as latch PUF [25], flip-flop PUF [26], and butterfly PUF [27]. Lastly, the recent development of PUFs presents with the ability to change the CRP through the introduction of a reconfigurable PUF [28]. This type of PUF has an advantage in that they can have a larger CRP space. However, the reconfiguration process still has challenges in terms of security.

One of the FPGA primitives that is widely used to build a PUF is the LUT (Look-Up Table). The LUT is one of the main building blocks of FPGA as it is used to create the logic element of the circuit. It can be configured to any kind of logic gates as well as to a memory element such as the shift register. A chain of LUT in series, while each configured as an inverter, creates a PUF-based ring oscillator [29]. The LUT in Xilinx's FPGA can also be configured into a 16-bit shift-register using the SRL16E mode. This particular configuration reduces the use of the FPGA resources by 16 times compared to the traditional approach for building registers using a flip-flop chain. SRL16E can be utilised in many ways for the development of PUFs. Thomas [30], [31] uses the SRL16E as a complementary component to increase the periodicity of the random number generator (RNG), which is the building block of PUF.

3.3.2 PUF characterisation

Different definitions of the parameters for PUF characterisation have been extensively discussed in [17]. Despite the differences in how they construct their

definition, there is one thing that they have in common. The majority, if not all, of the definitions of PUF characterisation, utilise Hamming distance as a quantifier. The Hamming distance between the PUF responses of the same PUF implementation in different devices is used to measure the uniqueness of the PUF by [17]. This definition requires the tester to have 2 or more devices to implement the same PUF, to generate the response, and to finally calculate the Hamming distance of the PUF under test. Without the presence of the second implementation or device, it is a bit of a challenge to get the uniqueness of the PUF under-test.

Maiti [17] defines the reliability of the PUF response as a complement of the averaged Hamming distance of the PUF response when it is measured at different device temperatures or voltage levels. The requirement to have a response to different temperatures and voltage levels limits the measurement in terms of it only being performed in test mode. The characterisation in a test mode will increase the downtime of a system. It is too risky to characterise the PUF using this definition as it will jeopardise the functionality of the system.

Another parameter that is widely used in PUF characterisation is throughput. It is a parameter used to measure the generation time of the PUF response. It is essential to understand the throughput of the PUF, so then it can be efficiently used in cryptography applications. The throughput is accounted as in equation (2-3).

3.4 Proposed works

3.4.1 Ring PUF as a digital sensor

A ring counter is a shift-register with a feedback loop. It can be utilised as a delay element used to build the PUF. The idea of using the ring counter as a building block for the PUF is similar to the concept of using a ring oscillator to create a delay in the system clock. Depending on the process variation of the components involved in the structure of the ring counter, the oscillation frequency will be different from one ring counter to another. Figure 3-3 shows the initialisation of

the ring counter. The 16-bit ring counter was initialised to have one bit of 1 and 15 bits of 0. This configuration is used to create a significant amount of delay, so then the frequency counter can easily capture the frequency differences between the two ring counters.

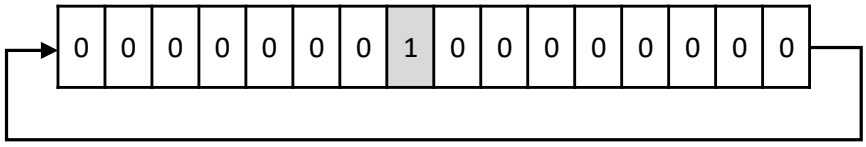

Figure 3-3: Initialisation of the ring counter

To meet the requirement of the sensor to make a measurement in mission mode with minimal influence on temperature and voltage fluctuation, the PUF is configured as in Figure 3-4. This configuration allows the PUF to run in parallel. Consequently, all parts of the responses are generated at the same time. This eliminates the influence of temperature and voltage change at the time of generation, and it increases the confidence level of the measurement.

The PUF consists of 256 pairs of ring counter (RC) used to produce a 256-bit response. The ring counter is implemented using the SRL16E mode Xilinx's LUT to simplify the design and reduce the silicon area usage. The SRL16E will be configured as a shift-register, and the output is connected back to its input; thus, it becomes a ring counter. Each RC pair has a 2-bit input. However, because of the limitations of time and resources, the challenge is only of an 8-bit length instead of 512-bit. The challenge will be used repeatedly for every 4 RC pairs. The response generation mechanism for each RC pair is given in the pseudo-code in Table 3-2.

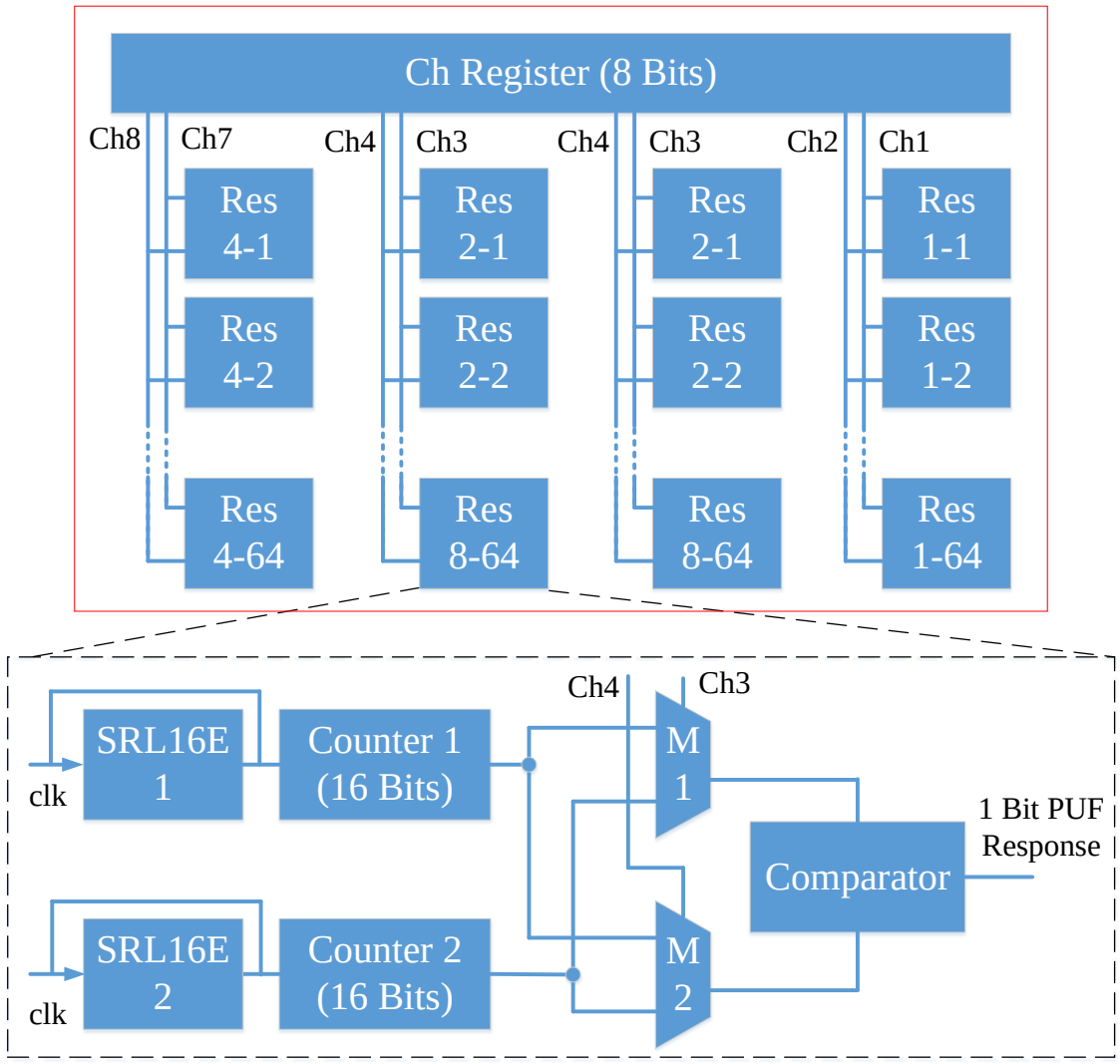

Figure 3-4: Block diagram of RC-based PUF

Table 3-2: Pseudo code for the RCPUF mechanism

```
mechanism RCPUF is
//component
challenge = {ch1, ch2}
SoR = {s1, s2}
counter = {c1, c2}
mux = {m1, m2}
comparator
//input-output
m1 {
input c1, input c2, select ch1, output muxout1
     }
m2 {
input c1, input c2, select ch2, output muxout2
```

```
     }
comparator {
input muxout1, input muxout2, output PUF_response
     }
//processing
     while c1 or c2 !overflow
          c1 = s1
          c2 = s2
     else
               stop all counter
               hold counter value
               shift counter value to mux
          then if
               muxout1 > muxout2
                    generate "1"
               else
                    generate "0"
```

3.4.2 Uniqueness

In contrast with the traditional definition of uniqueness, we define it as the comparison of the number of actual unique responses that the PUF generates with the maximum number of unique responses that it should be able to generate, as given in equation (3-1).

$$Uniqueness = \frac{R_u}{2^C} \times 100\% \quad \textbf{(3-1)}$$

Where R_u is the number of unique responses of a PUF which can be calculated using 'tabulate' function on MATLAB. 2^C is the number of maximum responses to the C-bit challenge, which is the number of challenges from the PUF.

3.4.3 Average reliability

Figure 3-5 shows the response of the PUF when a challenge is applied to it a couple of times. It appears that it is not generating a single unique response. Instead, it has a response variance with slightly different hamming distances from one another. The response with the highest reproduction rate will be called the dominant response, and it will be the formal response to that challenge. The

reliability of the Challenge-Response Pair (CRP) is the reproduction rate of the dominant response. Subsequently, the overall reliability of the PUF is then given by averaging the reliability of each dominant response as represented in equation (3-2):

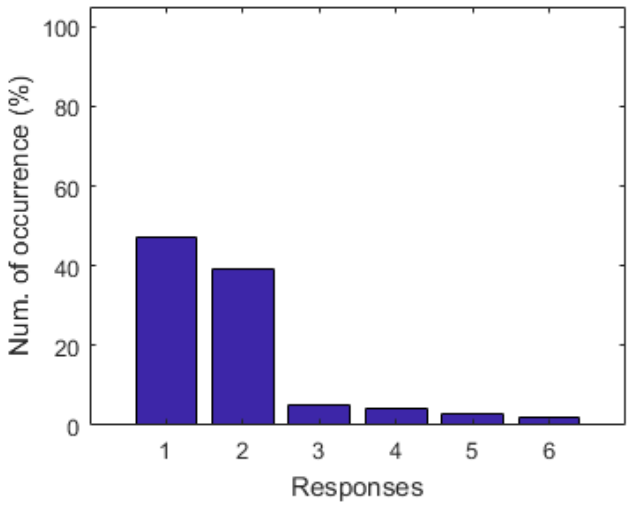

Figure 3-5: Response variance from a single challenge

$$\mu_r = \frac{\sum r_c}{2^C} \quad \textbf{(3-2)}$$

Where μ_r represents the average reliability of the dominant response, and r_c represents the percentage of occurrence of the dominant response to a challenge.

3.5 Experimental setup

The RCPUF will be implemented in the Kintex-7 FPGA development board. Figure 3-6 illustrates the process used for PUF characterisation. Assume that 100 iterations get 100 PUF responses, which can be used to obtain the dominant response to a specific challenge. The iteration process was done until all of the possible challenges are shifted 100 times to the PUF. In this case, because the challenge to the RCPUF is 8-bit binary, there are 256 possible challenges. For

each CRP, the value of each dominant response is grouped to obtain the R_u for use in the uniqueness calculation using equation . Next, the reliability of each dominant response is used to calculate the average reliability and to get the signature plot of the device. The throughput can be calculated by obtaining the working frequency after the placement and route (PAR) process occur, and after substituting the values in equation .

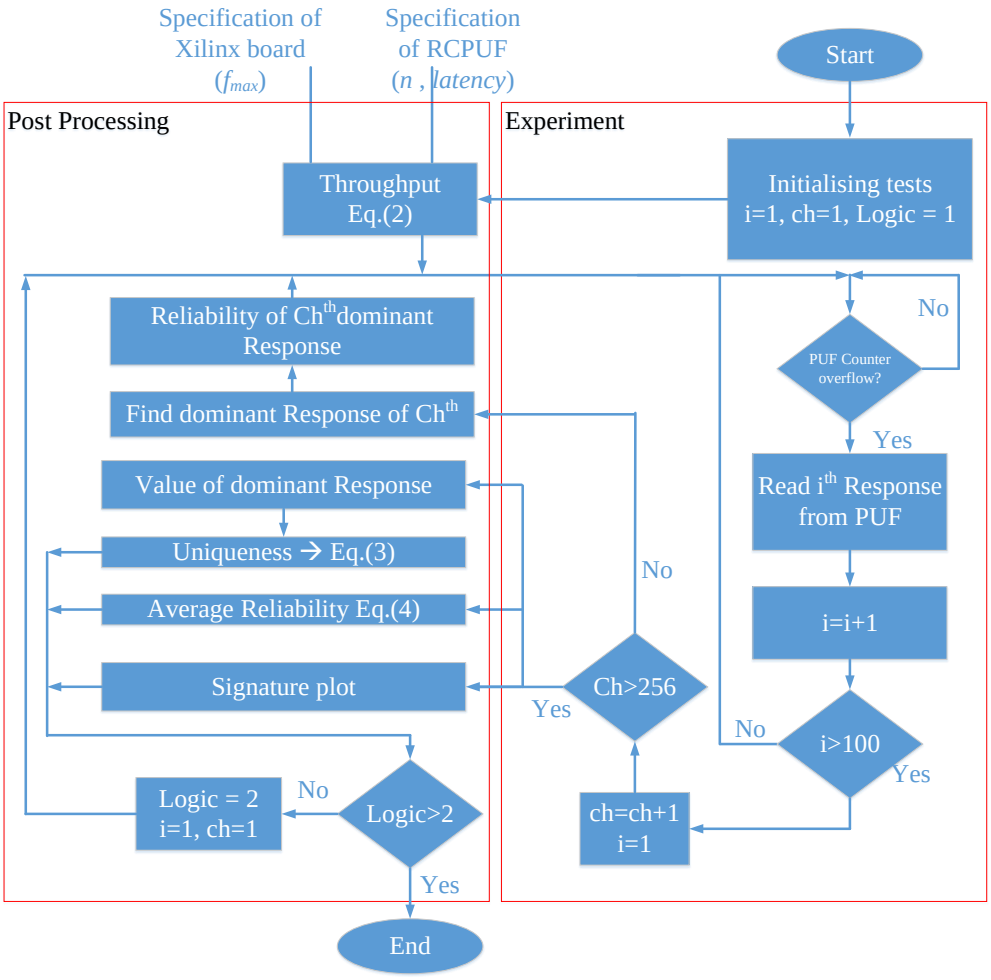

Figure 3-6: Flowchart for PUF characterisation and the detection of the FPGA configuration file modification

To test if the RCPUF-based sensor can detect any modification to the FPGA configuration file, the logic that is implemented into the FPGA is changed. The logical part of the device is represented by a linear-feedback shift-register (LFSR) in a built-in self-test (BIST). A built-in self-test (BIST) is a circuit that allows an IC

to perform tests by itself. It involves a set of test patterns to check most of the functionality of the device. The BIST consists of the memory used to store all of the test patterns and a state machine to choose which test pattern is to be fed to the device. The state machine can be realised using LFSR.

LFSR is a shift-register with a serial input (lfsr_tap) in the form of a linear function of its previous states. The input is driven by XORing, which is a particular set of the register's outputs. This means that the state of the LFSR depends on the chosen parts of the outputs and the register's initial value (also known as seeds). When it is appropriately configured, LFSR creates an infinite state. However, because it has a finite number of registers, the LFSR will eventually come back to its initial state.

The change in the logic to simulate the unauthorised modification is performed in the 10-bit LFSR circuit. The original feedback loop consists of an XOR for the seventh bit and tenth bit, as seen in Figure 3-7. The unauthorised modification occurs by changing the XOR and the OR gate, which reduces the number of LFSR states. As a result, the BIST will not be able to perform the self-test thoroughly. This affects the reliability of the device indirectly. Such a logical modification provides valuable input to measure the sensitivity of PUF identification. If a single gate modification is not detected, further amendments are continued until the unauthorised modification is detected. If the PUF sees the induced unauthorised single gate modification, then the PUF's sensitivity is known to be sufficient for detecting any significant changes to the FPGA configuration file.

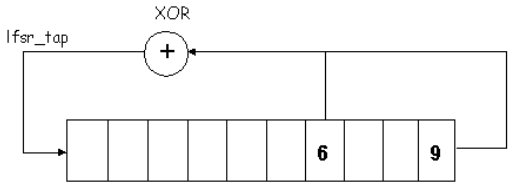

Figure 3-7: LFSR setup

In this experiment, any change in the reliability of the PUF responses indicates a modification to the FPGA configuration. The reliability of the PUF responses depends on the frequency stability of the source of randomness. The frequency stability is not only affected by environmental conditions such as temperature and voltage fluctuation, but they are also affected by the resonance frequency of the neighbouring circuit and wires [33]. The configuration in Figure 3-4 is used to minimise the environmental influence, so then a better measurement can be achieved.

Although the electronic design automation (EDA) tool can perform an automatic placement in the circuit to avoid the resonance effect. We tend to employ the resonance effect to detect unauthorised modifications to the FPGA configuration in our experiment. Four experimental setups, given in Table 3-3, have been prepared to validate the hypothesis mentioned earlier. Setups (A) and (B) include cases where the PUF does not have a constrained location but where the implemented logic might not or might be changed respectively, as in cases (A) and (B). Such a change in logic represents the unauthorised modification of the FPGA configuration file. The same method was done for the setups of (C) and (D). However, both have a constrained location in the PUF.

Table 3-3: Experimental setup

	No logic change	w/ logic change
Unconstrained	A	B
Constrained	C	D

For this experiment, only the location of the PUF is fixed. The location of the BIST module will be placed and routed automatically by the ISE synthesis tool. “LOC” attributes are used in the constraint file to fix the location of the PUF. It is also necessary to use the “keep” attribute in the VHDL file and the “S” attribute in the

constraint file to keep the synthesis tool in order not to get rid of the ring counter component.

Chipscope Pro 14.7 was used for data acquisition. Before the data acquisition started, a data acquisition module needs to be configured inside the FPGA. The purpose of this setup is to be able to capture the 256 PUF response in one go, as illustrated in Figure 3-8. On the left side of Figure 3-8, there is a set of registers for all possible challenges. Because the PUF was designed to have an 8-bit challenge, 256 registers with the length of 8-bit for each register were prepared. The finite state machine is a simple 8-bit binary counter that will call the next challenge when the previous challenge has already been shifted through the PUF.

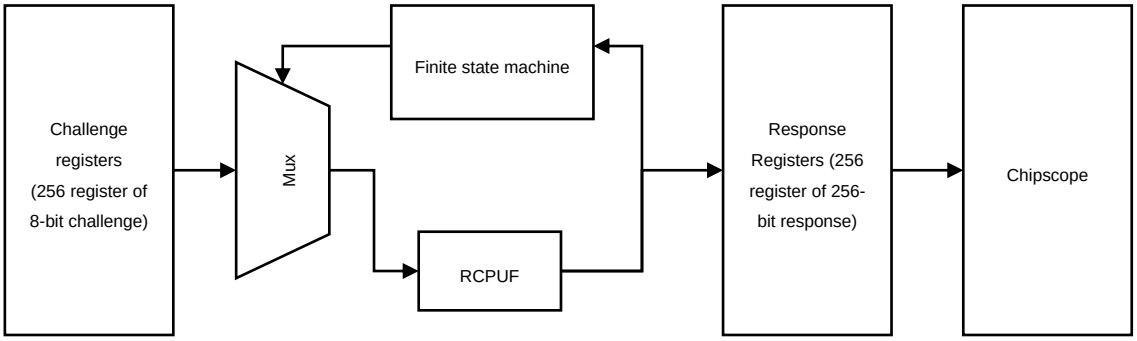

Figure 3-8: Data acquisition setup

3.6 Findings and discussion

3.6.1 RCPUF characterisation

The PUF characterisation was proceeded with in order to measure the uniqueness, reliability, and throughput of the proposed PUF. The Tabulate function from MATLAB was used to account for the number of unique numbers that PUF has. The next step is to calculate the uniqueness value using equation (3-1). Table 3-4 shows how many unique numbers each configuration has, along with its uniqueness. The average reliability was computed using equation (3-2).

Table 3-4: Reliability and uniqueness comparison of RCPUF

Parameter	Configuration			
	A	B	C	D
# of unique responses	217	224	96	140
Uniqueness (%)	84.76	87.50	37.50	54.68
Average Reliability (%)	5.86	5.70	62.08	59.51

There is an inverse correlation between the reliability and number of unique responses in the PUF, as indicated in Table 3-4. The unconstrained PUF has a high number of unique responses, but it is low in terms of its average reliability value. This result is similar to the result in [34] in which they have a randomly placed PUF. On the other hand, the constrained PUF has higher average reliability but a lower number of unique responses. In practice, the chance to get a unique response can be increased by increasing the length of the challenge. However, reliability is an intrinsic parameter that cannot be controlled. Therefore a constrained PUF has more advantages in terms of its higher than average reliability value.

It is true that for a weak PUF, there only exists one typical usable response that has the best stability [35]. However, with the improvement that has been made in the area of the PUF error correction algorithm, such as in [36], it will be beneficial to use a less stable response. Therefore the end-user will have more options to use the PUF, for example, for private/public key generation [37]. It will be more sensible to define the uniqueness of a PUF according to the number of actual unique responses that the PUF can generate. A PUF with an n-bit challenge will have 2^n possible unique responses. Still, it is almost impossible to have a maximum number of unique responses because of the random process variation.

For the throughput, information about the maximum working frequency for each configuration needs to be acquired. This information can be found in the post-PAR (placement and routing) report. Table 3-5 provides the counted maximum working frequency for each design, and the throughput was calculated using

equation (2-3). The latency for the proposed design is 16 x 16 because of the initialisation of the ring counter.

Table 3-5: Maximum working frequency and throughput

Parameter	Configuration			
	A	B	C	D
Max. frequency (MHz)	96.674	96.674	134.048	134.048
Throughput (Mbps)	96.674	96.674	134.048	134.048

Table 3-6 shows the throughput comparison between the RCPUF and the other PUF. Even though the throughput of the RCPUF is not as high as the other PUF in Table 3-6, it does not diminish its functionality in terms of detecting unauthorised modifications to the FPGA configuration file, which does not need high-speed performance.

Table 3-6: Throughput comparison

PUF Type	Throughput (Mbps)
Ring oscillator [38]	0.0018
Cross-couple inverter [39]	560
SRAM [40]	598
SR latches [34]	192
Ring counter	134

3.6.2 RCPUF implementation as a sensor

Having done the characterisation process of the RCPUF, the next step is to verify the placement of the PUF for the purpose of device identification. As already mentioned in Table 3-3, there are four different PUF configurations for this experiment. The parameter that will be used for this purpose is the reliability parameter. The result that was expected is that when there is a change to the configuration file of the FPGA, there will be a change in the fingerprint of the FPGA. In this experiment, the reliability parameter will be used to represent the change in the FPGA fingerprint. If the reliability changes, it can be assumed that there is a change in the fingerprint. Therefore the unauthorised change in the FPGA configuration file can be detected.

The result of the experiment is shown in Figure 3-9. It does not show any information about the value of the individual response, but it only indicates the reliability of the PUF dominant responses. However, it is still able to detect an unauthorised change in the FPGA configuration file by observing the changes in the reliability plot of the PUF.

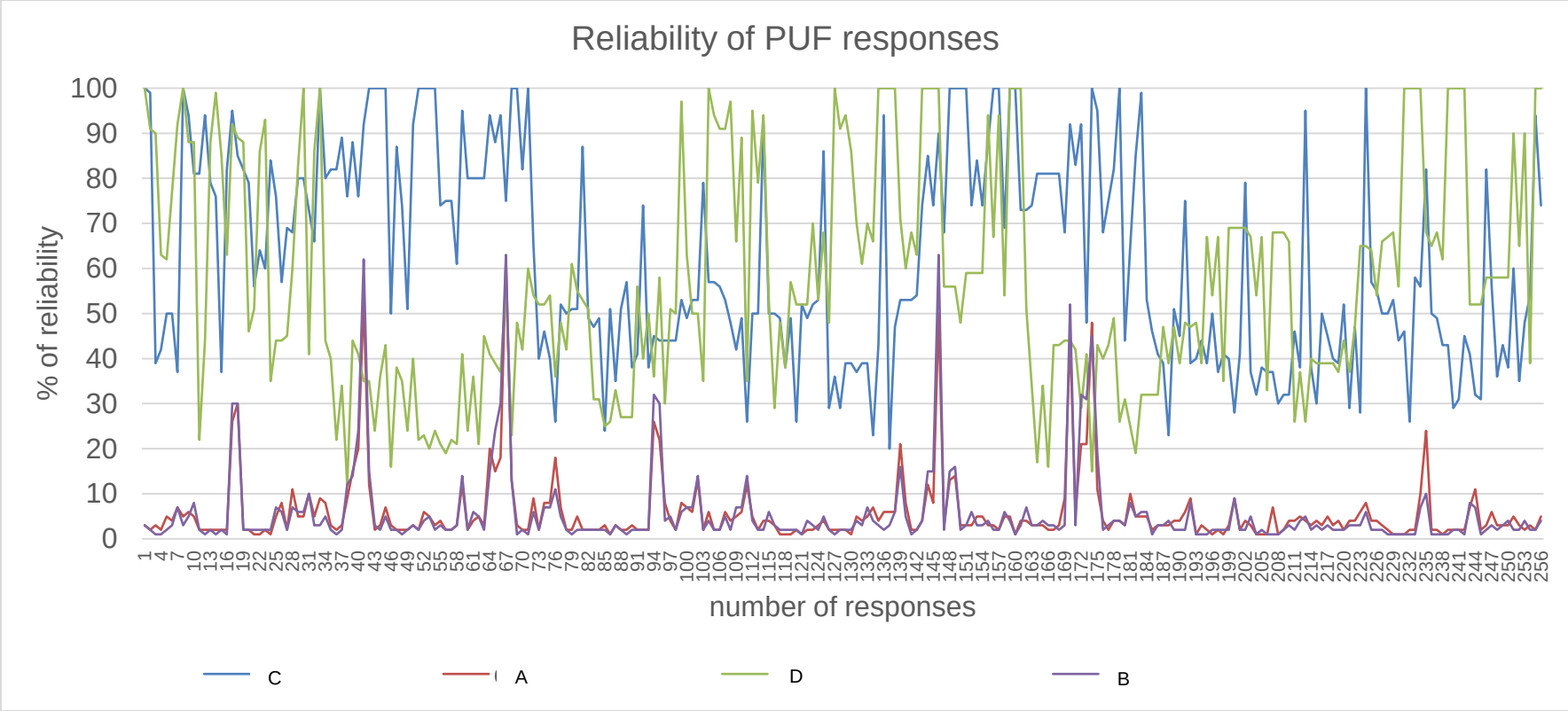

Figure 3-9: Comparison of the reliability of the PUF responses.

When the location of the PUF is not fixed, the reliability plot of the PUF dominant responses is not changed, regardless of whether there is a change to the FPGA configuration file or not. This indicates that the FPGA's fingerprint is also changed and that the unauthorised modification to the FPGA configuration file is detected.

Whenever the synthesis tool automatically picks the location of the PUF, it will try to find a site with the smallest disruption between each component to avoid the resonance effect. On the other hand, when a constraint is applied to fix the location of the PUF, the synthesis tool works hard to find the best possible routing for the design. Because of the constraint applied, the unconstrained logical circuit will eventually come across the path of the PUF component and disrupt the frequency of the ring counter [41]. Therefore any change made to the configuration file results in changes within the disruption. This turns to changes in the reliability of the PUF response when the location of the PUF is fixed.

The fact that the logic change in this experiment is represented by changing the XOR gate on the LFSR to the OR gate is an indication that the PUF as a sensor has a high level of sensitivity to detect unauthorised modifications to the FPGA configuration file.

3.6.3 Resistance to physical tampering and ageing

The developed detection mechanism has resistance to physical tampering, including temperature and/or voltage alteration, and ageing. This can be explained using the following example:

1. If an attacker is trying to completely break the detection mechanism by altering the environmental condition (temperature and/or supply voltage) with the purpose of changing the FPGA configuration file without being detected, then the user will be alerted with the fact that the detection mechanism has been compromised.
2. If the purpose of the attack is to change the reliability plot, the user will understand that there is an ongoing effort to tamper the detection mechanism

or modification of the FPGA configuration file. After all, it will take a significant amount of effort for the attacker to figure out that there is a detection mechanism that uses PUF reliability as its sensor. Therefore this kind of attack scenario is challenging to perform.

3. One of the possibilities for the attacker to change the FPGA configuration file without being detected is by stabilising the reliability plot while performing the modification of the FPGA configuration file. This kind of attack can be done by altering the temperature and/or supply voltage. However, there are some challenges that the attacker will face when performing such an attack:
    a. First, the attacker needs to able to figure out that the PUF reliability is used as a sensor.
    b. Second, to understand how much temperature or voltage change is needed to stabilise the reliability plot, the attacker needs to understand the device's sensitivity to environmental change.
    c. Third, in order to perform an accurate side-channel reading to understand the reliability plot, the device needs to be decapsulated [42]. Without the decapsulating process, there is no guarantee that side-channel reading is accurate.

The detection mechanism also has good resistance to the ageing process. This can be done by making the reliability measurement both continuous and in real-time. This can be easily done as the architecture of the PUF allows for the measurement of the reliability performed in mission mode as described in section 3.4.1.

Having mentioned the attack scenario above, it can be concluded that the detection mechanism has excellent resistance to environmental attacks and the ageing process.

3.7 Conclusion

In this chapter, a high-sensitivity and affordable sensor to detect unauthorised modifications to the configuration file of a multi-tenant FPGA service have been developed. The sensor is based on a novel physically unclonable function with a ring counter as its source of randomness. Instead of using the response of the PUF, the sensor uses the average reliability as its signature. This makes the sensor tamper-proof and eliminates the need for an error correction algorithm.

A comparison between a couple of PUF implementations has been made, and it has been concluded that the PUF needs to be located in a fixed location on the FPGA floorplan to be able to detect any unauthorised changes.

Furthermore, the characterisation of the novel ring counter PUF is performed. A new definition of PUF characterisation has been introduced in this chapter. Based on this new definition, it shows that there is an inverse correlation between the reliability of the PUF and the number of unique responses that it can produce. This new definition opens up a new possibility for PUF designers to balance the reliability and uniqueness of the PUF responses.

3.8 References

[1] "Amazon EC2 F1 Instances." [Online]. Available: https://aws.amazon.com/ec2/instance-types/f1/. [Accessed: 06-Apr-2020].

[2] A. M. Caulfield *et al.*, *A Cloud-Scale Acceleration Architecture*. .

[3] C. Ramesh *et al.*, "FPGA Side Channel Attacks without Physical Access," in *Proceedings - 26th IEEE International Symposium on Field-Programmable Custom Computing Machines, FCCM 2018*, 2018, pp. 45–52.

[4] G. Provelengios, C. Ramesh, S. B. Patil, K. Eguro, R. Tessier, and D. Holcomb, "Characterization of long wire data leakage in deep submicron FPGAS," in *FPGA 2019 - Proceedings of the 2019 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2019, pp.

292–297.

[5] I. Giechaskiel, K. Eguro, and K. B. Rasmussen, “Leakier Wires: Exploiting FPGA Long Wires for Covert-and Side-channel Attacks,” *ACM Transactions on Reconfigurable Technology and Systems*, vol. 12, no. 3, pp. 1–29, Sep. 2019.

[6] D. Hyde, “A Survey on the Security of Virtual Machines,” St. Louis, 2009.

[7] J. Zhang *et al.*, “Design and implementation of a delay-based PUF for FPGA IP protection,” in *Proceedings - 13th International Conference on Computer-Aided Design and Computer Graphics, CAD/Graphics 2013*, 2013, pp. 107–114.

[8] S. Trimberger, J. Moore, and W. Lu, “Authenticated encryption for FPGA bitstreams,” in *FPGA’ 2011, Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*, 2011, pp. 83–86.

[9] C. Gu and M. O ’neill, “Ultra-compact and Robust FPGA-based PUF Identification Generator,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2015, p. 934.

[10] S. Buchovecká, R. Lórencz, F. Kodýtek, and J. Buček, “True random number generator based on ring oscillator PUF circuit,” *Microprocessors and Microsystems*, vol. 53, pp. 33–41, Aug. 2017.

[11] B. Karpinskyy, Y. Lee, Y. Choi, Y. Kim, M. Noh, and S. Lee, “Physically unclonable function for secure key generation with a key error rate of 2E-38 in 45nm smart-card chips,” in *Digest of Technical Papers - IEEE International Solid-State Circuits Conference*, 2016, vol. 59, pp. 158–160.

[12] U. Guin, A. Singh, M. Alam, J. Canedo, and A. Skjellum, “A secure low-cost edge device authentication scheme for the internet of things,” in *Proceedings of the IEEE International Conference on VLSI Design*, 2018, vol. 2018-January, pp. 85–90.

[13] C. Gu, N. Hanley, and M. O'Neill, "Improved reliability of FPGA-based PUF identification generator design," *ACM Transactions on Reconfigurable Technology and Systems*, vol. 10, no. 3, May 2017.

[14] H. Yu, P. H. W. Leong, and Q. Xu, "An FPGA chip identification generator using configurable ring oscillator," in *Proceedings - 2010 International Conference on Field-Programmable Technology, FPT'10*, 2010, pp. 312–315.

[15] P. Kitsos, K. Stefanidis, and A. G. Voyiatzis, "TERO-Based Detection of Hardware Trojans on FPGA Implementation of the AES Algorithm," in *Proceedings - 19th Euromicro Conference on Digital System Design, DSD 2016*, 2016, pp. 678–681.

[16] A. P. Fournaris, L. Pyrgas, and P. Kitsos, "An FPGA hardware trojan detection approach based on multiple parameter analysis," in *Proceedings - 21st Euromicro Conference on Digital System Design, DSD 2018*, 2018, pp. 516–522.

[17] A. Maiti, V. Gunreddy, and P. Schaumont, "A systematic method to evaluate and compare the performance of physical unclonable functions," *Embedded Systems Design with FPGAs*, pp. 245–267, 2013.

[18] R. Pappu, "Physical One-Way Functions," *Science*, vol. 297, no. 5589, pp. 2026–2030, 2002.

[19] J. Guajardo, S. S. Kumar, G. J. Schrijen, and P. Tuyls, "FPGA intrinsic PUFs and their use for IP protection," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2007, vol. 4727 LNCS, pp. 63–80.

[20] U. Rührmair, H. Busch, and S. Katzenbeisser, "Strong PUFs: Models, constructions, and security proofs," in *Information Security and Cryptography*, no. 9783642143120, Springer International Publishing, 2010, pp. 79–96.

[21] R. Pappu, "Physical One-Way Functions," *Science*, vol. 297, no. 5589, pp.

2026–2030, Sep. 2002.

[22] D. P. Sahoo, D. Mukhopadhyay, R. S. Chakraborty, and P. H. Nguyen, “A Multiplexer-Based Arbiter PUF Composition with Enhanced Reliability and Security,” *IEEE Transactions on Computers*, vol. 67, no. 3, pp. 403–417, Mar. 2018.

[23] C. Q. Liu, Y. Cao, and C. H. Chang, “ACRO-PUF: A Low-power, Reliable and Aging-Resilient Current Starved Inverter-Based Ring Oscillator Physical Unclonable Function,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 64, no. 12, pp. 3138–3149, Dec. 2017.

[24] O. Willers, C. Huth, J. Guajardo, H. Seidel, and P. Deutsch, “On the feasibility of deriving cryptographic keys from MEMS sensors,” *Journal of Cryptographic Engineering*, vol. 10, no. 1, pp. 67–83, Apr. 2020.

[25] A. Ardakani, S. B. Shokouhi, and A. Reyhani-Masoleh, “Improving performance of FPGA-based SR-latch PUF using Transient Effect Ring Oscillator and programmable delay lines,” *Integration*, vol. 62, pp. 371–381, Jun. 2018.

[26] S. Khan, A. P. Shah, S. S. Chouhan, N. Gupta, J. G. Pandey, and S. K. Vishvakarma, “A symmetric D flip-flop based PUF with improved uniqueness,” *Microelectronics Reliability*, vol. 106, p. 113595, Mar. 2020.

[27] X. Xu *et al.*, “A highly reliable butterfly PUF in SRAM-based FPGAs,” *IEICE Electronics Express*, vol. 14, no. 14, Jul. 2017.

[28] S. Khan, A. P. Shah, N. Gupta, S. S. Chouhan, J. G. Pandey, and S. K. Vishvakarma, “An ultra-low power, reconfigurable, aging resilient RO PUF for IoT applications,” *Microelectronics Journal*, vol. 92, p. 104605, Oct. 2019.

[29] F. Kodytek, R. Lorencz, J. Bucek, and S. Buchovecka, “Temperature Dependence of ROPUF on FPGA,” *Proceedings - 19th Euromicro Conference on Digital System Design, DSD 2016*, pp. 698–702, 2016.

[30] D. B. Thomas and W. Luk, “FPGA-Optimised Uniform Random Number Generators Using LUTs and Shift Registers,” in *2010 International Conference on Field Programmable Logic and Applications*, 2010, pp. 77–82.

[31] D. B. Thomas and W. Luk, “The LUT-SR Family of Uniform Random Number Generators for FPGA Architectures,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 4, pp. 761–770, Apr. 2013.

[32] M. Bakiri, C. Guyeux, J.-F. Couchot, and A. K. Oudjida, “Survey on hardware implementation of random number generators on FPGA: Theory and experimental analyses,” *Computer Science Review*, vol. 27, pp. 135–153, Feb. 2018.

[33] Xilinx, “AR# 65459: Power - Mitigating the Effects of Power System Resonance.” [Online]. Available: https://www.xilinx.com/support/answers/65459.html. [Accessed: 29-Apr-2020].

[34] D. Yamamoto *et al.*, “Uniqueness enhancement of PUF responses based on the locations of random outputting RS latches,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2011, vol. 6917 LNCS, pp. 390–406.

[35] U. Ruhrmair *et al.*, “PUF modeling attacks on simulated and silicon data,” *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1876–1891, 2013.

[36] J. Delvaux, D. Gu, I. Verbauwhede, M. Hiller, and M. D. M. Yu, “Efficient fuzzy extraction of PUF-induced secrets: Theory and applications,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2016, vol. 9813 LNCS, pp. 412–431.

[37] R. Maes, A. Van Herrewege, and I. Verbauwhede, “PUFKY: A fully functional PUF-based cryptographic key generator,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7428 LNCS, pp. 302–319, 2012.

[38] C. Jin *et al.*, “FPGA Implementation of a Cryptographically-Secure PUF Based on Learning Parity with Noise,” *Cryptography*, vol. 1, no. 3, p. 23, 2017.

[39] S. Satpathy *et al.*, “An All-Digital Unified Static/Dynamic Entropy Generator Featuring Self-Calibrating Hierarchical von Neumann Extraction for Secure Privacy-Preserving Mutual Authentication in IoT Mote Platforms,” in *IEEE Symposium on VLSI Circuits, Digest of Technical Papers*, 2018, vol. 2018-June, pp. 169–170.

[40] S. Chen, B. Li, and C. Zhou, “FPGA implementation of SRAM PUFs based cryptographically secure pseudo-random number generator,” *Microprocessors and Microsystems*, vol. 59, pp. 57–68, Jun. 2018.

[41] D. Merli, F. Stumpf, and C. Eckert, “Improving the quality of Ring Oscillator PUFs on FPGAs,” in *Proceedings of the 5th Workshop on Embedded Systems Security, WESS ’10*, 2010.

[42] M. Hutter and J. M. Schmidt, “The temperature side channel and heating fault attacks,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2014, vol. 8419 LNCS, pp. 219–235.

4 LAYERED SECURITY FOR JTAG/IJTAG USING A BIMODAL PHYSICALLY UNCLONABLE FUNCTION

4.1 Abstract

While bringing in advantages in terms of testability, the IEEE 1687, also known as the IJTAG, also brings in new challenges in relation to security and scalability. Reports about the discovery of the malicious usage of the JTAG, which is what the IJTAG is based on, means that the IJTAG also possesses a similar property if it is not appropriately secured. Moreover, the fact that the IJTAG is prepared to be the standard of the future means that scalability needs to be well thought out to guarantee its performance. This section proposes an efficient layered security mechanism for JTAG/IJTAG using a new class of physically unclonable function (PUF) called a Bimodal PUF. It moves beyond the conventional single-challenge single-response PUF by introducing a second response to the PUF from the same single challenge. As an advantage, a double-response PUF forms a 2-layer security solution, one in the hardware layer by limiting the access to the embedded instrument and the second for the data layer by securing the output data that needs to be transmitted. Experiments conducted with FPGA show that compared to the traditional single challenge-single response PUF, the bimodal PUF has a doubled functionality while only adding 40% of the silicon area usage.

4.2 Introduction

A rapid shrinking down of the size of the transistors increases the complexity of VLSI devices and their testing procedures. The manufacturers are able to integrate more IPs into a single device as in on-chip embedded instruments, but the IPs are less accessible in relation to conventional probes. This, in turn, makes the necessary tests and quality checks more complex and challenging. It is a crucial requirement to develop methods and tools for enabling technical tests and proceeding with the characterisation of the VLSI devices. This is impossible to do in a short time. From an economic point of view, while the market competition requires manufacturers to release their products quickly, the extended testing

time means that there is a loss of profit and competition in the current challenging market.

The latest industrial solution used to overcome the problem is the IEEE 1500 Embedded Core Test (ECT) [1] and the IEEE 1687 [2], also known as the Internal JTAG (IJTAG). Both standards are an extension of the IEEE 1149.1 [3] JTAG, and they use the same Test Access Port (TAP) controller as JTAG for interfacing the Instruments-Under-Test (IUT) with the test equipment. The new standard makes it possible to employ boundary scan techniques for testing the embedded Intellectual Property (IP). The differences and similarities between the three standards are presented in Figure 4-1.

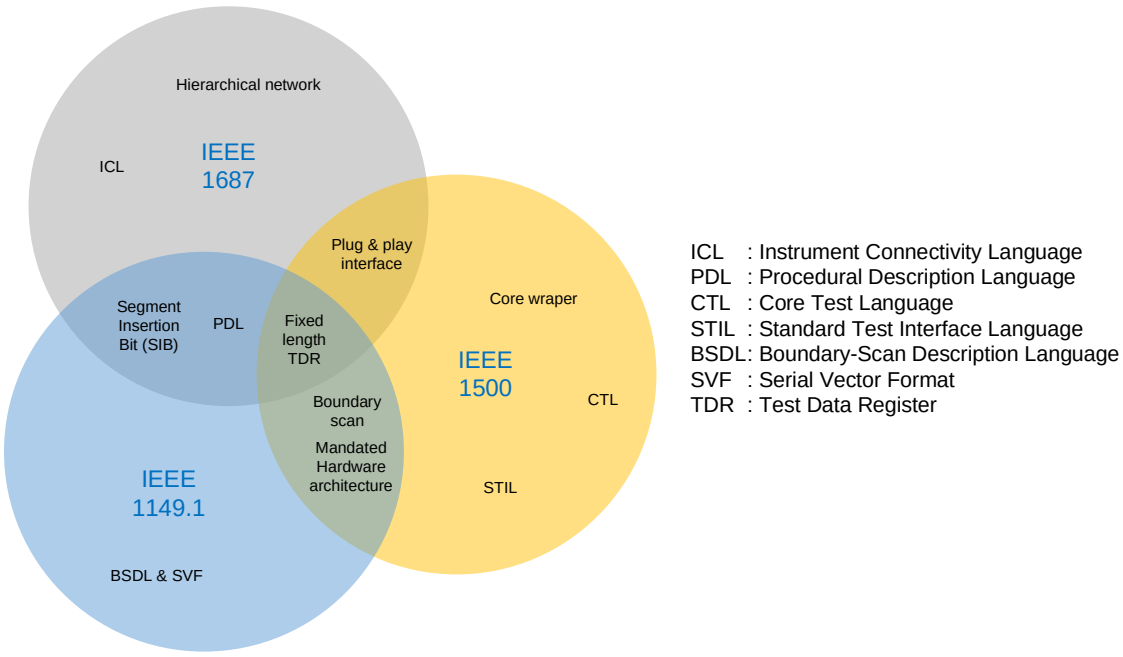

Figure 4-1: IEEE 1149.1 vs IEEE 1500 vs IEEE 1687

While it can be said that an embedded instrument is also an IP core, both have their own standards when it comes to testing their functionality. The differences between IEEE 1500 and IEEE 1687 are related to the depth of the scan boundary that it can perform. While the IEEE 1500 and IEEE 1149.1 have a mandatory hardware architecture that needs to be followed by a system to comply with the

standard, the IEEE 1687 is more of a descriptive standard rather than a prescriptive one. This makes it easier for it to be modified to add a new feature to the standard. The IEEE 1687 will, therefore, be the main topic discussed in this chapter.

One of the prominent features of IJTAG is its support when creating a dynamic scan chain by implementing a Segment Insertion Bit (SIB) that acts as a gate in front of the instrument on the chip as can be seen in Figure 4-2. The SIB has 2 working states, closed and open. Testing instruments thus require only opening the relevant SIBs of the chosen instruments. However, in 2013, JTAG was improved to also have the ability to have a variable scan chain through the introduction of the SIB as in the IJTAG.

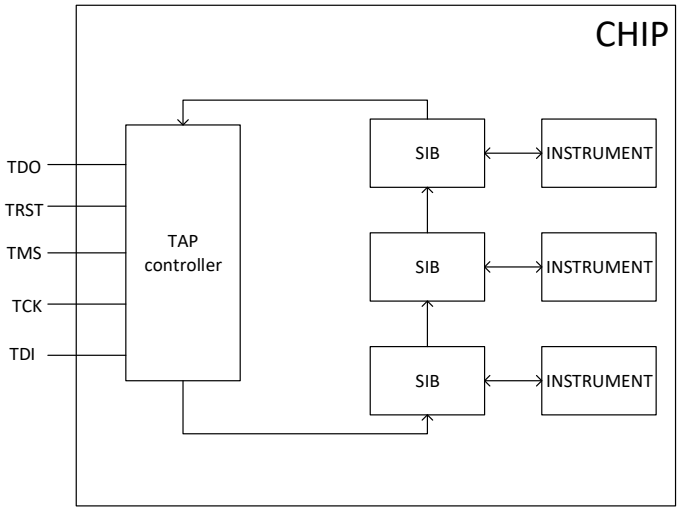

Figure 4-2: IJTAG network with SIB and 3 Instruments

Along with the advantages that it brings to the testability of the embedded instruments, IJTAG has several security-related challenges. The first one is that IJTAG does not have built-in security measures to prevent unauthorised access to the embedded instruments. A report about security breaches exploiting the TAP controller can be found in both the news [4] and academic papers [5].

Majeric [6] presented the first JTAG fault injection attack. The attack exploits the debugging capabilities of JTAG as a path for a fault injection attack. The attack will provide a user privilege escalation, so then a normal user can perform tasks previously limited to the administrator of the system. This kind of attack is harmful in a way, in that it can be used to steal confidential information and copyrighted material such as pirated software.

The architecture of IJTAG itself poses a security flaw in which the Test Data Register (TDR) that is embedded in the IJTAG compliance IPs is a secret that is only known by the IP maker. This flaw makes it possible for a malicious TDR to be inserted into the IP that can manipulate the data shifted to said IP. This kind of attack is called a data integrity attack [7]. Because the architecture of IJTAG originates from the initial JTAG, it is also vulnerable to attacks that applied to the JTAG. Moreover, on-chip instruments may contain confidential data, patented IPs and critical cores that will stop the whole system if they are compromised. Hence the security of the IJTAG network is essential.

Dworwak et al. proposed a security mechanism for IJTAG by architecting locking techniques for the SIB using an n-bit signal [8]. Liu et al. [9] proposed a secret key generation technique using the Linear-feedback Shift Register (LFSR). However, the methods presented by Dworwak [8], and Liu [9] only utilise a static secret key. Baranowski proposed a dynamic secret key generation using hash core [10]. It solved the primary drawback of the static key, but it has scalability issues. Sudeendra [11] proposed a method using the Physically Unclonable Function (PUF) for the secret key generation and comparing it to the secret key generated by LFSR. This technique has better scalability because it does not need external memory to keep the secret key. Echeloned IJTAG data protection was proposed in [12] to not only secure the access but also to secure the data from the embedded instruments. It uses two cipher cores to encrypt both the Test Data Input (TDI) and the Test Data Output (TDO) of the instrument. However, the presence of 2 ciphers can create an unnecessary and excessive use of silicon area. A graph colouring method to isolate the malicious instruments was proposed in [13] to secure the IJTAG network against either a data transmission

attack or a sniffing attack. Although such a solution secures systems against internal security breaches, the data is still left unprotected, and it can be sniffed by attackers from the outside of the network.

Besides the prevention techniques mentioned above, there is also the IJTAG attack detection technique that has been proposed by researchers. Xuanle [14] proposed the use of a machine learning system to detect illegitimate access to the IJTAG network by checking the number of shifting cycles. If the cycle is more than the pre-defined cycle, then this shows that there is something that needs to be investigated. However, this system cannot detect a more sophisticated attack, and this may result in false positives. Xuanle then improved his findings by implementing a Low-Density Parity Check (LDPC)-based feature reduction technique [15]. His experiment result increased the detection accuracy by increasing the previously insignificant amount of area overhead in the IJTAG network.

Having conducted a literature review, it can be concluded that the available security mechanism for the IJTAG network has a problem in which the security of the data is not taken care of adequately. Even though there exist proposals that try to secure the data coming from the network, their security mechanism is lacking in scalability.

PUF is believed to be one of the scalable hardware security primitives [16][17]. Its design variation has been explored extensively by researchers. However, the PUF has its own issue: its implementation takes up a lot of silicon area and/or FPGA resources. As a consequence, the benefits of the new PUF design over the design that it tries to improve on are not significant. For example, an increase in the PUF's reliability has to be paid for by using an error correction algorithm that takes up a lot of silicon area to the point where it sometimes has to be implemented externally. It can be concluded that the available PUF design is already too saturated and that there needs to be a breakthrough to get significant benefits from the already beneficial security primitives.

This chapter proposes a novel PUF-based security mechanism to prevent access to the IJTAG network as well as to prevent access to the output data of the

instrument by unauthorised parties. The developed PUF breaks the norm of the conventional PUF challenge-response pair by having two unique responses to a single challenge, hence the name ‘bimodal PUF’. The first response will be used to unlock the SIB, and the other one is used for obfuscating the output data coming out from the instrument. By having two unique responses to a single challenge, it is proven that the bimodal PUF shows a significant amount of improvement compared to the legacy PUF design.

An overview of the PUF has been discussed in the next section. The section on Proposed Works (4.4) discusses the development and characterisation of a new class of PUF used to achieve the proposed security mechanism. A novel IJTAG security protocol is presented in section 4.5, followed by the security analysis in section 4.5.2. Finally; section 4.6 concludes this chapter.

4.3 Related Works

4.3.1 IEEE 1687 (IJTAG)

The IJTAG standard offers rules that are convenient and straightforward to follow and implement by industries. It is mainly developed based on descriptions of the system/network-on-chip. Its hierarchical design makes it possible for the SIB to become a gateway to an embedded instrument or a doorway to a deeper layer of hierarchy, as illustrated in Figure 4-3.

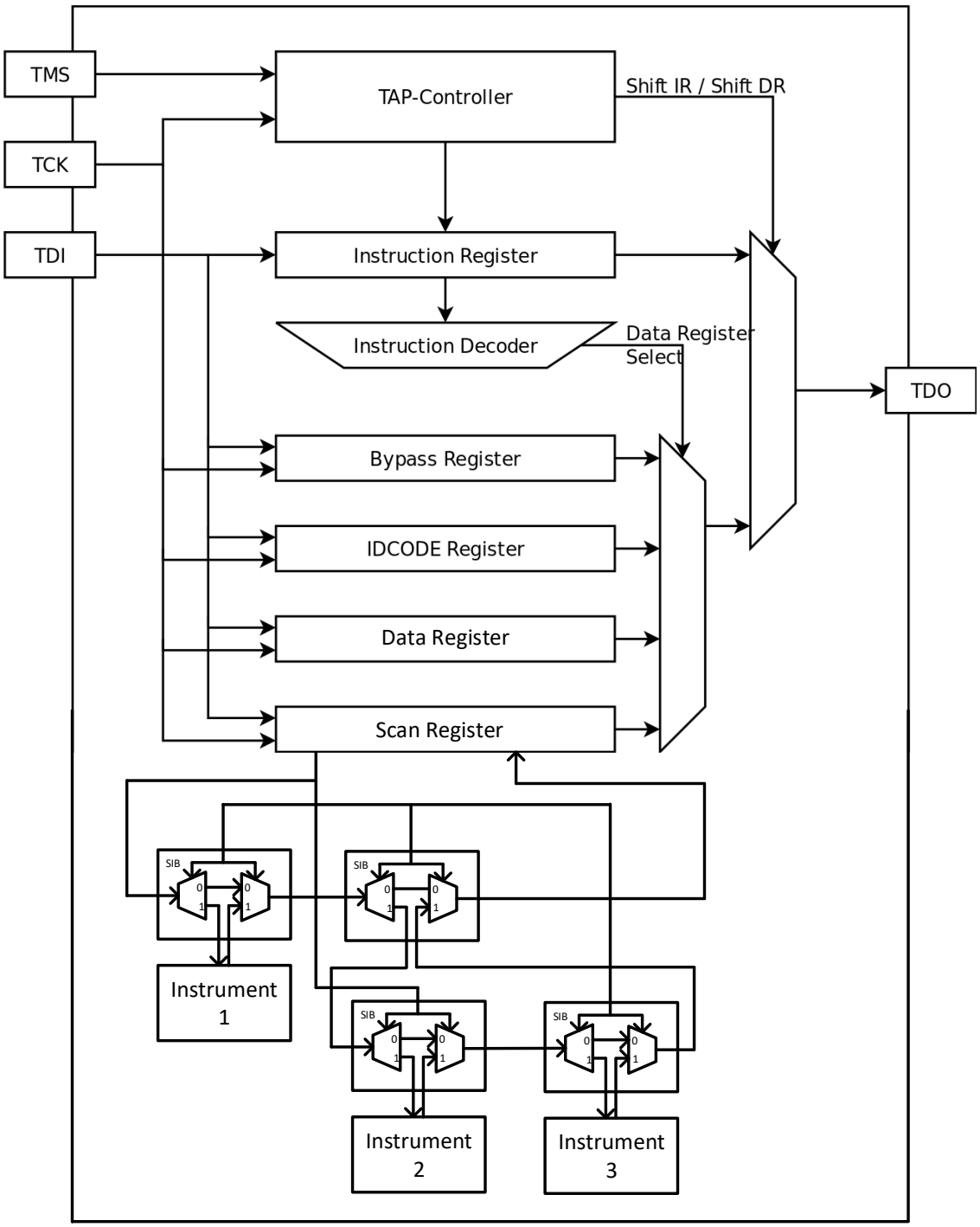

Figure 4-3: IJTAG hierarchical network with a SIB

The simplest form of SIB is a 1-to-2 demultiplexer connected directly to a 2-to-1 multiplexer. When the select input is “1”, the SIB will shift the data to the instrument. Otherwise, the SIB will bypass the data to the next SIB when the

select input is “0”. The bit length of the address as the select input of the SIB is the same as the number of SIBs in the IJTAG network.

The IJTAG uses the same Test Access Port (TAP) controller as the JTAG. The TAP controller is a 16-state Finite State Machine (FSM) that regulates the data flow to and from the electronic devices. The state diagram of a TAP controller is, as shown in Figure 4-4.

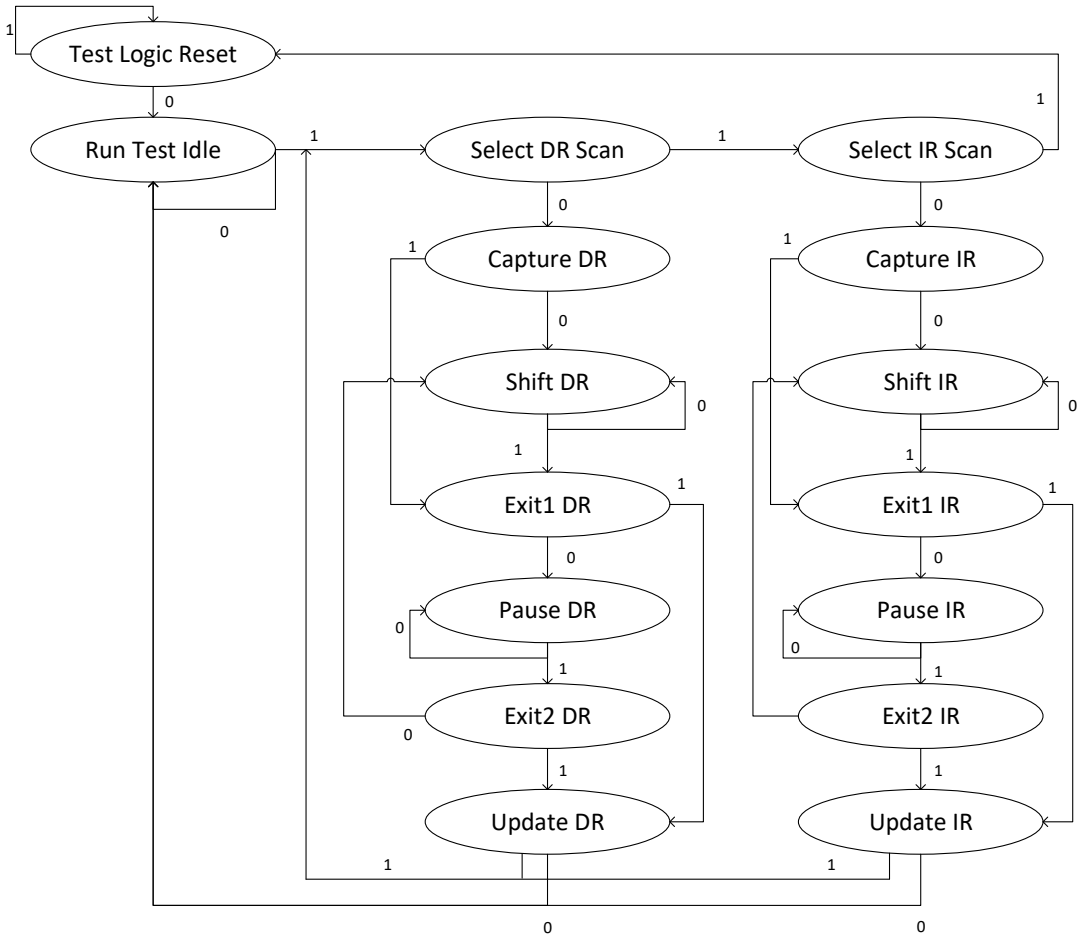

Figure 4-4: State diagram of the TAP controller

From Figure 4-4, it can be seen that to reset the state of the TAP controller, it always takes five clock cycle no matter where the initial state of the controller is. For this reason, TRST becomes an optional pin to have in a TAP controller.

4.3.2 Physically Unclonable Function (PUF)

A PUF can be defined as a product of the utilisation of physical randomness of an object/device that is easy to produce but non-invertable and unpredictable [18]. Formally, PUF maps a set of finite numbers (challenges) onto a set of finite numbers (response), in which both is a part of sample space S, as in equation (4-1) [18].

$$x_1, x_2, \dots, x_n \rightarrow y_1, y_2, \dots, y_n ; (x_i, y_i \in S, 1 \leq i \leq n) \quad \textbf{(4-1)}$$

Because the PUF is created from a physical system, its entropy and the ability to produce a set of responses is limited by its physical dimension. If it is assumed that the PUF is located on a sphere with a radius of R, then its entropy will be limited by equation (4-2) [19].

$$H \leq \alpha \cdot R^2 \quad \textbf{(4-2)}$$

Where H is the entropy of the system, and α is the physical properties of the sphere. In a silicon device, the maximum entropy of a PUF implemented in the silicon is bound by N silicon cells such as logic, memory, flip-flop, etc., with a C information capacity in a single cell. Therefore equation (4-2) can be written as equation (4-3) [20].

$$H \leq C \cdot N \quad \textbf{(4-3)}$$

From equations (4-2) and (4-3), it can be seen that the entropy of the PUF is not unlimited but cannot be predicted until its architecture is implemented on a physical system.

PUF has properties that make it suitable for key generation for use in cryptographic applications. For instance, a PUF can produce a randomly generated response every time the same challenge is given to it. The response might always be the same, or it may change slightly, depending on the reliability of the PUF. On the other hand, when a challenge is given to 2 different PUFs, the 2 PUFs will generate two different responses. This indicates that no PUF implementation is the same, even when it is implemented in the same FPGA family. This PUF behaviour is produced as a side effect of the manufacturing variance, such as the variations in the transistors' length, width, and thickness.

Even though PUF looks promising for use in security applications, the industry still hesitates to implement it natively in their product. The first reason for this is that most of the designs suggested for PUFs are too costly for implementation, e.g. they take a lot of silicon area usage. Additionally, PUFs have reliability issues related to producing a reproducible response without any error correction algorithms in place. The addition of an error correction algorithm will increase the silicon area usage even more. Therefore there is a desire to create a more efficient and reliable PUF.

One of the FPGA primitives that is widely used to build a PUF is a LUT (Look-Up Table). A LUT is one of the main building blocks of an FPGA as it is used to create the logical element of the circuit. It can be configured to any kind of logic gate as well as to a memory element such as a shift register. A chain of LUTs in series, with each configured as an inverter, creates a PUF-based ring oscillator [21]. For instance, the LUT in Xilinx's FPGA can be configured into a 16-bit shift-register using the SRL16E mode. This particular configuration reduces the use of FPGA's resources by 16 times compared to the traditional approach for building registers using a flip-flop chain. SRL16E can be utilised in many ways for the development of PUFs. Thomas [22], [23] used the SRL16E as a complementary component to

increase the periodicity of the random number generator (RNG), which is the building block of the PUF.

4.3.3 PUF Metrics

There are three main parameters that characterise a PUF. These parameters are reliability, uniqueness, and throughput.

The uniqueness of a PUF is measured by how many unique responses the PUF can generate. A PUF with an C-bit challenge will have 2^C possible unique responses. However, because of the random process variation, it is almost impossible to have a maximum number of unique responses.

A novel definition of the uniqueness of a PUF is proposed in Chapter 3 by comparing the real unique response that it can produce with the maximum unique responses that it should be able to produce.

When a challenge is applied 100 times to a PUF, it will not only generate a single unique response, but it will also generate other responses with a slight hamming distance difference between them. The response with the highest rate of occurrence is called the dominant response, and it will be chosen as the formal response to that challenge. The percentage of occurrences of the dominant response is used as the basis of the reliability of the specific PUF response. The overall reliability of the PUF is given by averaging the reliability of each of the responses as given in equation (3-2).

For some applications such as cryptography, the PUF response needs to be generated at high speed. The parameter used to measure the speed of the PUF response generation is called the throughput, calculated using equation (2-3).

Where n is the number of bit-length of the generated PUF response, f_{max} is the maximum working frequency of the design, and the latency is the number of cycles used to generate the one bit of PUF response. f_{max} is not the maximum frequency of the FPGA board but it was obtained by looking at the post-route-and-placement report of the FPGA. For the latency, if the PUF is based on the parallel 16-stage ring oscillator, then this means the number of lags is 16. This

means that it takes a 16 clock cycle to measure the frequency of the ring oscillator. If it only takes two clock cycles to produce a 32-bit random number, then this means that the number of lags is 2.

4.3.4 Splittable random number generator

Splittable Random Number Generators (RNGs) are widely used in functional programming [24]. A regular RNG, also known as a linear RNG, involves the mapping of a set of numbers onto another set of numbers through a random phenomenon. In functional programming, a splittable RNG is created by dividing a random number from a linear RNG using some of the methods described in Table 4-1. All of the methods split a string of random numbers into two random numbers with the same bit length. Assume that the original random number has an i bit length.

Table 4-1: Splitting method for the splittable RNG

Splitting method	1st part	2nd part
Half-and-half	$n_1, n_2, n_3, \dots, n_{i/2}$	$n_{(i/2)+1}, n_{(i/2)+2}, n_{(i/2)+3}, \dots, n_i$
Odd-even	$n_1, n_3, n_5, \dots, n_{2i-1}$	$n_2, n_4, n_6, \dots, n_{2i}$
Bunny hop	$f(i)$	$g(i)$

There are some constraints to these splitting methods. The half-and-half and odd-even are only applicable if the original random number has an even bit-length. However, since the common bit-length of a random number for the application of security of a digital system rarely has an odd bit-length, this limitation is not a hindrance. The bunny-hop splitting method constructs the split random number using a function repeatedly to pick the i^{th} bit for the splitted random number. It has a disadvantage in that not every bit of the original random number will be

chosen to construct the split random number. The efficiency of the bunny-hop splitting method depends on what function is used, e.g. $f(i)$ and $g(i)$, and the length of the original and split random number.

4.4 Proposed Works

4.4.1 IJTAG Security Mechanism

There are a couple of options that can be used to implement a PUF into IJTAG security measures as follows:

1. The implementation of a single PUF with a reusable response as in Figure 4-5. The first use of the response is to secure access to the IJTAG network. The same response will also be used to secure the output data of the embedded instrument. While this implementation looks feasible, the reuse of the PUF response will threaten the security of the whole system. If the attacker knows the challenge used to generate the correct response, they can easily get access to the IJTAG network as well as decipher the output data.

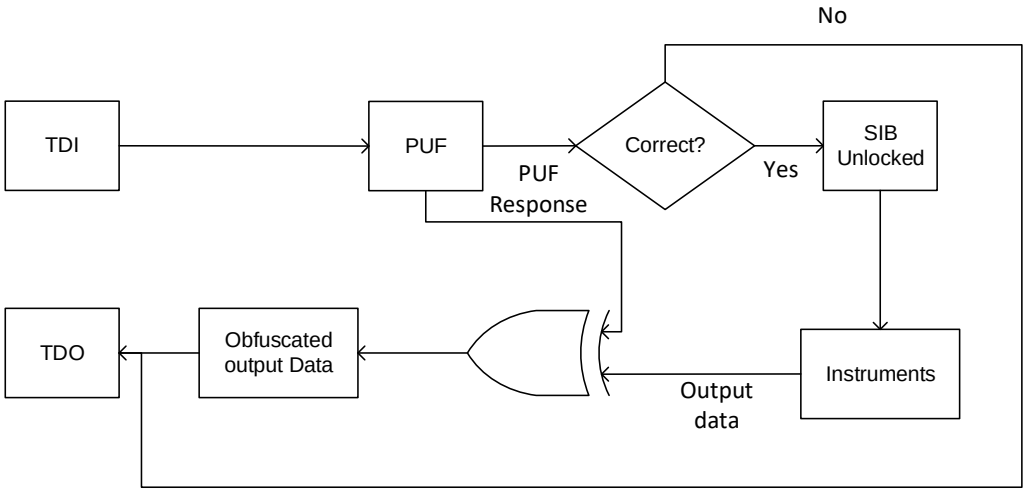

Figure 4-5: Flowchart of the IJTAG security with a reusable PUF response

2. Implementation of 2 PUFs as in Figure 4-6. The response from the first PUF can be used to secure access to the IJTAG network, and the response from the second PUF can be used to chiper the output data. From a practical point of view, this system will work just fine. However, the efficiency of this new system is unlikely to be improved. The reason for this is that while the reliability and the uniqueness of the system might increase, the implementation of 2 PUF instances will also increase the area occupation and power consumption of the system.

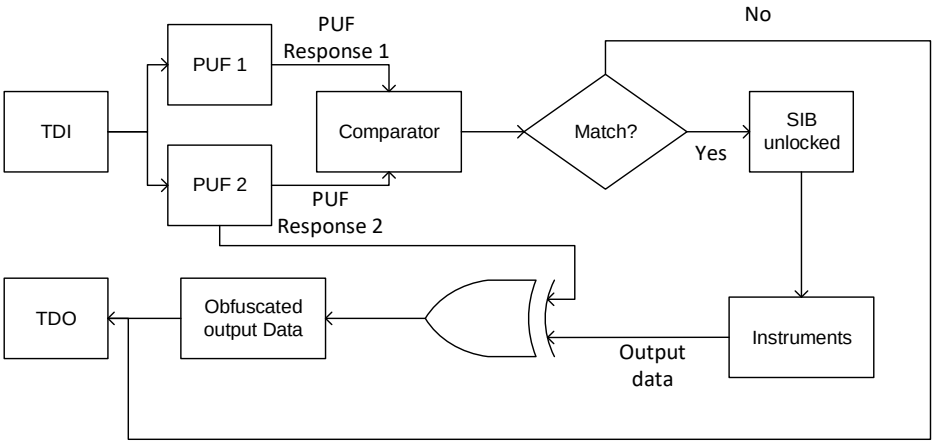

Figure 4-6: Flowchart of the IJTAG security with 2 PUF implementations

3. Because it is impossible to achieve an efficient PUF implementation to secure the IJTAG network by increasing the number of PUFs implemented, the only option left is to find a way to increase the functionality without adding to the area usage and power consumption. While the throughput of a PUF depends on the device where the PUF being implemented, the overall reliability and uniqueness of the PUF response can be increased by increasing the number of responses that can be generated by the PUF. This is where the bimodal PUF become a possible option to efficiently secure the IJTAG network without having a scalability issue, as illustrated in Figure 4-7.

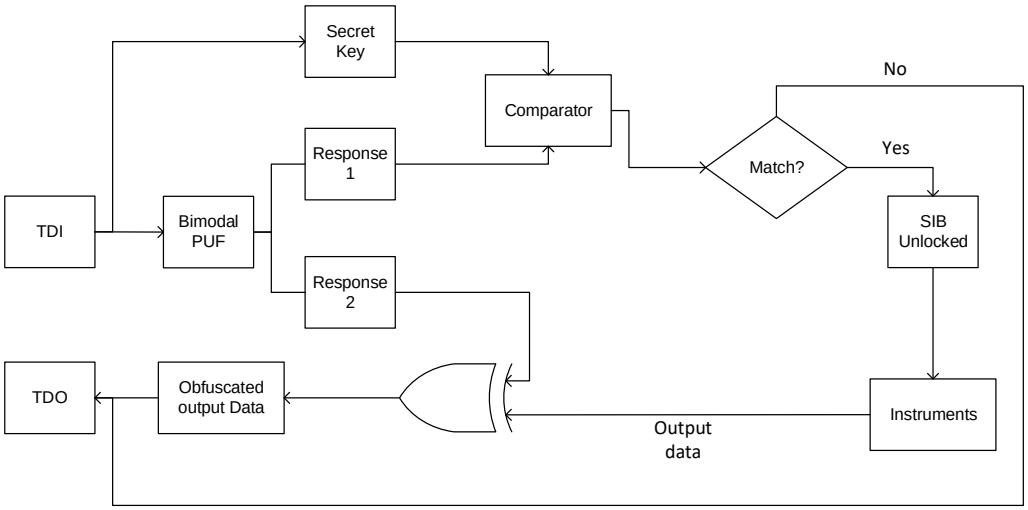

Figure 4-7: Flowchart of the IJTAG security with the bimodal PUF

Figure 4-8 illustrates the data flow in the IJTAG network with the addition of the proposed bimodal PUF. The first response of the bimodal PUF will be used to unlock the SIB, and the second response from the same challenge will be used to obfuscate the output data.

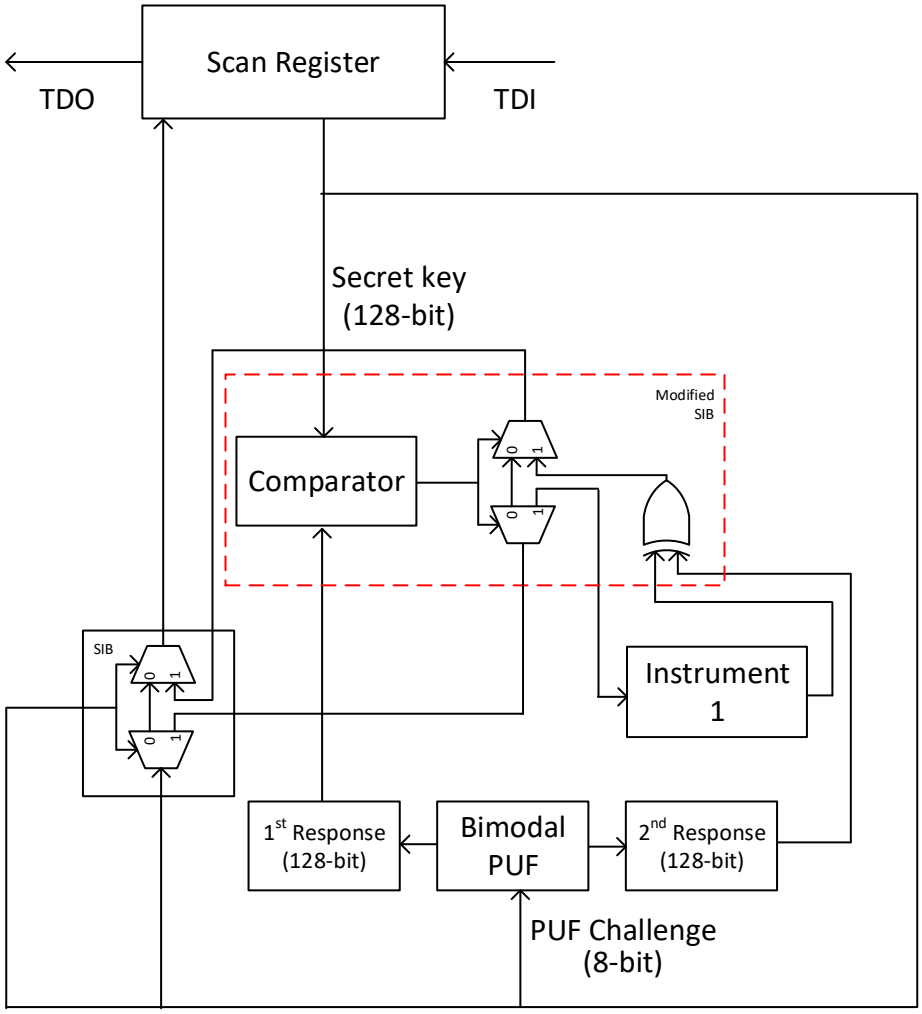

Figure 4-8: Multi-Layer security mechanism for the IJTAG network using the bimodal PUF

In order to support the lock-key mechanism, the SIB is configured as in Figure 4-8. The select input for the mux-demux of the modified SIB comes from the one-bit output of a comparator. It compares the first responses of the bimodal PUF with the secret keys that are assumed to be safe/unknown to the adversary. The challenge of the PUF is also assumed to be secure/unknown to the adversary.

The secret key and the challenge to the PUF is assumed to be safe/unknown to the adversary. Even if one of them is known, the adversary still needs to find out the other. There is another assumption that can be made regarding the secret

key as can be seen in Table 4-2. However, the assumption used in this experiment has the right balance between security and silicon area usage.

Table 4-2: Assumption comparison for the secret key and PUF challenge confidentiality

PUF challenge	Secret key	Impact
Public	Public	Total data loss
Public	Secret	Attacker can use brute force to guess the secret key
Secret	Public	Attacker can use brute force to guess the challenge
Secret	Secret	Attacker can use brute force but the time needed increases exponentially

The output data is obfuscated to make it secure from any unauthorised parties who might use it illegally for malicious purposes such as stealing confidential data or acquiring the logic of the instrument for reverse engineering. Data obfuscation provides the same level of security as conventional straight forward encryption techniques. The downside of traditional encryption techniques is that attackers will easily recognise if the data has been encrypted. Therefore they can easily find the counter to that encryption mechanism and steal the data. Hence a straight forward encryption technique alone is not enough to secure the output data. On the other hand, data obfuscation produces readable data, but in an obfuscated form, so it is delivered as a ‘fake’ data if not de-obfuscated. The attacker will not realise that the data is being obfuscated and they will assume that it is the correct data from the instrument. Even if the attacker intends to inject a signal to activate the hardware Trojan hidden inside the instrument, the output of the hardware Trojan will not affect the other instruments as the signal is obfuscated regardless.

4.4.2 Cost benefit analysis

The efficiency of a new PUF architecture compared to its predecessor, the PUF architecture, can be measured using equation (4-4).

$$\pi = \Delta F - \Delta C \quad \textbf{(4-4)}$$

π is the efficiency (in percent) of the new PUF design compared to the previous attempt to get at least the same level of functionality F. C is the cost needed to build a PUF. A positive value for π means that the new PUF design has a better implementation compared to its predecessor/reference design. In this regard, the cost is a function on the FPGA resource or silicon area usage (A) and power consumption (P). Meanwhile, functionality is the function of the PUF characteristics such as reliability (r), uniqueness (u), and throughput (t). Therefore equation (4-4) can be written as equation (**4**-**5**).

$$\pi = \Delta F(r, u, t) - \Delta C(A, P) \quad (4\text{-}5)$$

r, u, and A are represented in percentage. However, the throughput is a unit of Hertz and power is a unit of Watt. Therefore the conversion of frequency and power to a unit of percentage is needed as in equations (4-6) and (4-7).

$$t = \frac{t_{Hz}}{t_{max}} \times 100\% \quad (4\text{-}6)$$

$$P = \frac{P_{Watt}}{P_{Max}} \times 100\% \quad (4\text{-}7)$$

Where t_{Hz} is the throughput of the PUF and t_{max} is the maximum throughput of the PUF when implemented in the fastest ideal silicon. P_{Watt} is the power consumption of the design while P_{max} is the maximum power that can be handled

by the device. However, to find the value of t_{max} and P_{max} in the design is still a challenge. This value will be kept as a variable, and it will be assumed to be the same if the compared PUF design is implemented in the same device.

Because π is a unit of percentage, its maximum value should not go higher than 100%. However, its variable, r, u, t, A, and P is also stated in percent. These variables need to be normalised with a contribution factor, so then equation (4-5) can be expanded into equation (4-8).

$$\pi = [\alpha(r_2 - r_1) + \beta(u_2 - u_1) + \gamma(t_2 - t_1)] - [\delta(A_2 - A_1) + \varepsilon(P_2 - P_1)] \quad (4\text{-}8)$$

$$\alpha + \beta + \gamma = \delta + \varepsilon = 1 \; ; 0 \leq \alpha, \beta, \gamma, \delta, \varepsilon \leq 1 \quad (4\text{-}9)$$

Where subscript 1 is for the reference design, and subscript two is for the new design. α, β, γ are the contribution factors of r, u, t concerning the functionality of the PUF while δ and ε are the contribution factors of A and P related to the cost of the PUF.

Since there are two different responses generated by the bimodal PUF to a single challenge, the efficiency of a bimodal PUF compared to the legacy PUF design can be written as equation (4-10).

$$\pi = [\alpha((r_{21} - r_{11}) + (r_{22} - r_{12})) + \beta(u_2 - u_1) + \gamma(t_2 - t_1)] - [\delta(A_2 - A_1) + \varepsilon(P_2 - P_1)] \quad (4\text{-}10)$$

Where the r_{21} and r_{22} are the first and the second responses of the bimodal PUF. If the reference PUF is a regular PUF, then the $r_{12} = 0$ and equation (4-10) become equation (4-11).

$$\pi = [\alpha(r_{22} + r_{21} - r_{11}) + \beta(u_2 - u_1) + \gamma(t_2 - t_1)] - [\delta(A_2 - A_1) + \varepsilon(P_2 - P_1)] \quad (4\text{-}11)$$

From equation (4-11), it can be concluded that the bimodal PUF is the only option to increase the efficiency of the new PUF architecture without sacrificing its scalability.

4.4.3 Bimodal PUF

The implication of equation (4-3) in the PUF design is that its entropy function can be predicted once it is implemented in a physical system. Let E be the entropy equation of a PUF as a function of physical variance v as in equation (4-12).

$$E(v); 0 \leq v \leq v_i \quad (4\text{-}12)$$

Equation (**4-12**) implies that the entropy function will likely, but not necessarily, have a positive gradient.

We define the conventional PUF response R as the result of a black-box function of entropy $b(E(v))$ which itself is a result of the black box function to the PUF challenge $b(C)$. Formally, this definition can be written as in equations (4-13) and (4-14).

$$E(v) = b(C) \quad (4\text{-}13)$$

$$R = b(E(v)) \quad (4\text{-}14)$$

A function is a particular case of a relation in which one and only one output come from one or more inputs. In relation, one or more outputs can be obtained from one or more inputs. Because the things that connect the PUF challenge to the entropy and the entropy to the PUF response are unknown (black box), it is possible for it to be in the form of either a function or relation. To generalise things, we define a PUF as a relation rather than a function. After all, the entropy is predictable after the PUF is implemented. As a consequence, the black box properties of a PUF are not valid anymore. It is only a black box for an observer that has no information about the PUF's implementation and response.

By using this definition to define a PUF, or rather a PUR (Physically Unclonable Relation), it is possible for a PUF to have more than one response to a single challenge. However, to keep things in context according to the aim of this experiment, we will limit this possibility to only two responses.

Every relation needs three things that have to be defined: the range set $\{0,1\}^l$ (challenge), the domain set $\{0,1\}^m$ (response), and the rule of the assignment. Because the relationship between entropy and response is a black box, it is only possible to define a part of the relation, so then two responses can be obtained from a single challenge. We define the bimodal PUF as a PUF instance that, in parallel, generates two sequences of values that are statistically independent but repeatable given the same challenge, subject to its reliability and uniqueness. Given this definition, considering equations (4-13) and (4-14) and inspired by the splittable random number generator in functional programming [24], the response of a conventional PUF is to split after the generation process.

Let $R = \{0,1\}^m$ be the response of a conventional PUF with the length of m-bit. If r_1 and r_2 is the first and the second response of the bimodal PUF, then equation (4-15) and (4-16) represent the rule of assignment for the bimodal PUF.

$$r_1 = \{0,1\}^{\frac{m_i}{2}} \text{ for } 1 \leq i \leq \frac{m}{2} \tag{4-15}$$

$$r_2 = \{0,1\}^{\frac{m_i}{2}} \text{ for } \frac{m}{2} < i \leq m \tag{4-16}$$

There are different ways to split a random number, as mentioned in Table 4-1. However, this chapter will only discuss the splitting mechanism as in equation (4-15) and (4-16) and leave the other mechanism to others.

4.4.4 Design and architecture of the bimodal PUF

Figure 4-9 illustrates the available options to develop the bimodal PUF. First, a different type of PUF as the core of the system can be chosen. This experiment will use a LUT-based ring counter as the source of randomness. A ring counter is a shift-register with a feedback loop. In ASIC, the ring counter was built by chaining together a number of flip-flops and making a feedback loop from its output. The same design principle can also be applied to FPGA.

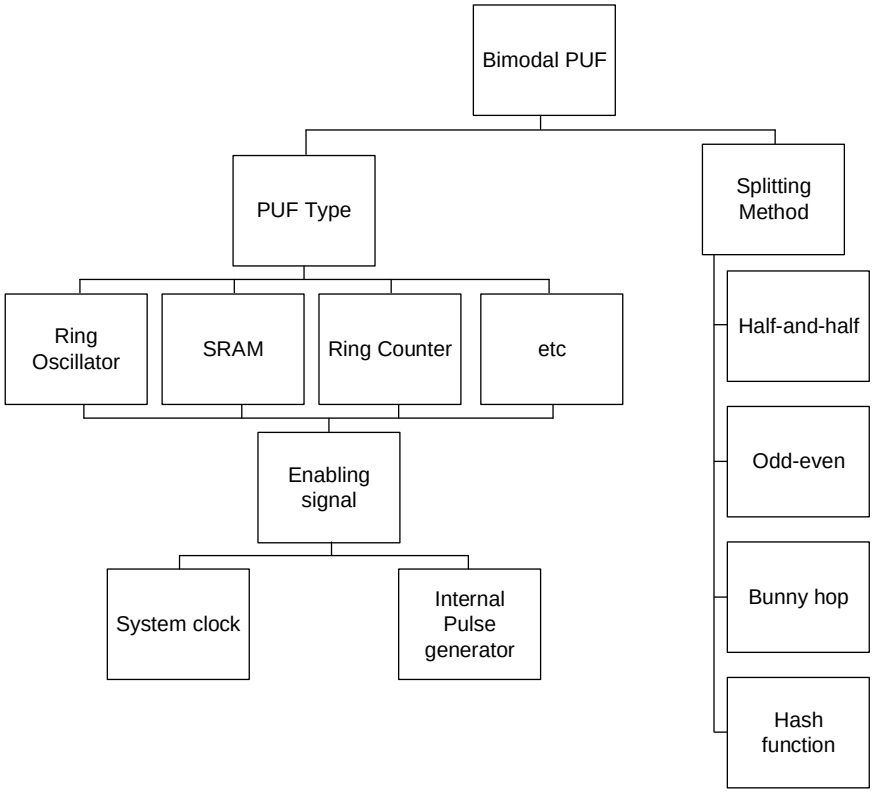

Figure 4-9: Classification of bimodal PUF

The source of randomness in a PUF can be enabled using either a system clock or an internal pulse generator. The system will create a PUF in a ready state so long as the system is supplied with power. However, if the PUF is not required to be active at any time and it only needs to be activated when the embedded instrument is to be accessed, then an internal pulse generator as the enabling signal is a more affordable option. The reason for this is because when the PUF is always active, it will dissipate more heat and in return, it will decrease the life span of the silicon. However, when the PUF is only activated when needed, it will not have such a problem, and its life span will increase. The internal pulse generator was implemented using the shift register mode of the SRL16E mode of a LUT.

Lastly, there are a number of splitting methods that can be used to generate the bimodality of the PUF responses as mentioned in Table 4-1. While comparing the performance of the different splitting methods is an exciting topic to discuss, this chapter only discusses the use of the half-and-half splitting method. This will leave the research on the performance comparison between each splitting method to others.

The bimodal PUF was configured as in Figure 4-10, making it possible for every bit in the responses to be generated at the same time while also minimising the environmental influences such as temperature and voltage variance on the generated responses. Thus, a more confident data acquisition result can be achieved.

The idea of using a ring counter as the source of randomness for PUF is similar to the idea of using a ring oscillator to create a delay in the system clock. Two ring counters initialised as 10101010…. or 01010101010… will oscillate when activated. Depending on the process variation of the components used to create the ring counter, the oscillation frequency will be different from one ring counter to another. A 1-bit random number can be generated by comparing the frequency of the two ring counters. In this experiment, the 16-bit ring counter was initialised only to have one bit of 1 and 15 bits of 0. This configuration was used to create a more significant delay, so then the signal analyser can easily see any difference

in frequency. However, this configuration will increase the latency of the design and affect the overall throughput.

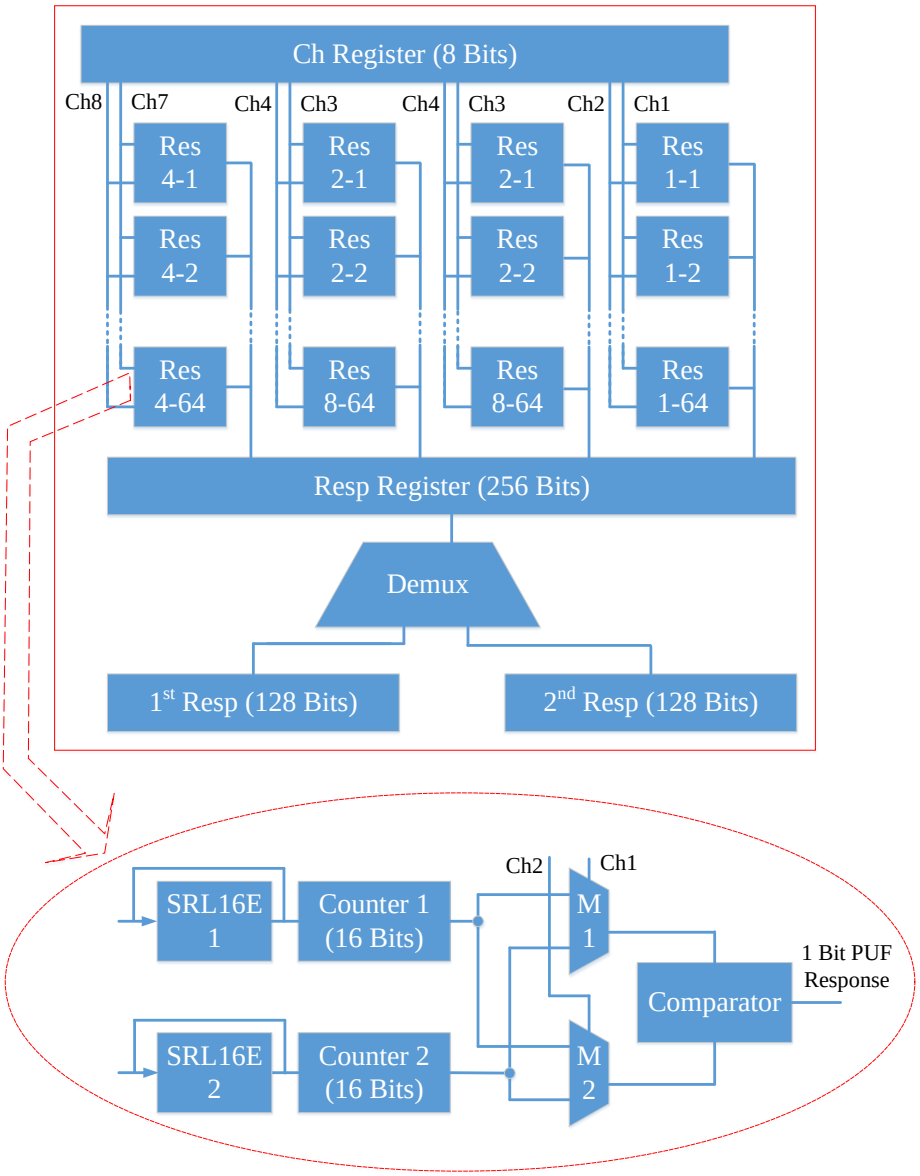

Figure 4-10: RC-based bimodal PUF

The bimodal PUF consists of 256 pairs of ring counters (RCs) to produce two different sets of responses with the length of 128-bit each. The ring counter was implemented using the SRL16E mode in Xilinx's LUT to simplify the design and reduce the silicon area usage. The SRL16E was configured as a shift-register,

and the output was connected back to its input; thus, it becomes a ring counter. Each RC pair has 2-bit input. However, because of the limitations in terms of time and resources, the challenge only had an 8-bit length instead of a 512-bit length. The challenge will be used repeatedly for every 4 RC pair. The response generation mechanism for each RC pair was given in the pseudo-code shown in Table 4-3. A demultiplexer was added at the output of the PUF to split the response in half. This is needed to create the bimodality feature of the proposed PUF.

Table 4-3: Pseudo-code for the RCPUF mechanism

```
mechanism RCPUF is
//component
challenge = {ch1, ch2}
SoR = {s1, s2}
counter = {c1, c2}
mux = {m1, m2}
comparator
//input-output
m1 {
input c1, input c2, select ch1, output muxout1
      }
m2 {
input c1, input c2, select ch2, output muxout2
      }
comparator {
input muxout1, input muxout2, output PUF_response
      }
//processing
      while c1 or c2 !overflow
            c1 = s1
            c2 = s2
      else
                  stop all counter
                  hold counter value
                  shift counter value to mux
            then if
                  muxout1 > muxout2
                        generate "1"
                  else
                        generate "0"
```

4.4.5 Experimental Setup

Kintex-7 was used for the implementation and performance verification of the proposed techniques. There were four experiments performed in this chapter, as illustrated in Figure 4-11.

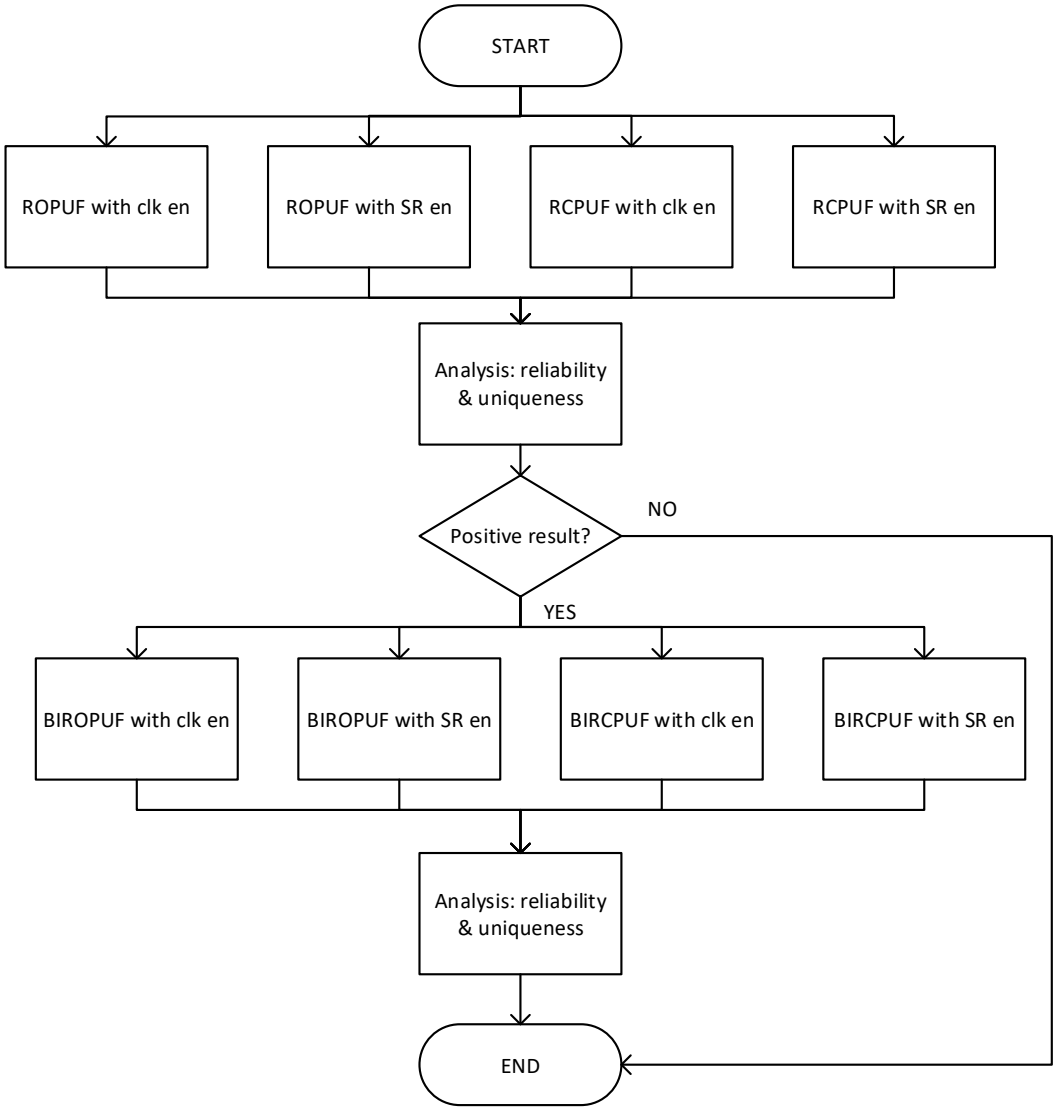

Figure 4-11: Experiment flow chart

First, a performance comparison between the Ring Oscillator PUF (ROPUF) and the Ring Counter PUF (RCPUF) with a different enabling signal was performed.

The enabling signal used was the system clock and the 16-bit Shift Register (SR). The SR represents the internal enabling pulse. The shift register was initialised to 1010101010101010, so then the interval between high and low is similar to the system clock. The reliability and uniqueness of each setup were analysed to verify the hypothesis. Next, 4-bimodal PUF, Bimodal ROPUF (BROPUF) and Bimodal RCPUF (BRCPUF) were tested and analysed using the same setup as the previous experiment. The characterisation process of the Bimodal PUF was done following the same procedure as in Chapter 3 of this thesis.

Chipscope Pro 14.7 was used for data acquisition. A data acquisition module was configured on the FPGA to capture the bimodal PUF response in a single iteration. The data acquisition setup is illustrated in Figure 4-12. The finite state machine is an 8-bit binary counter. The finite state machine will call the next challenge when the previous challenge has already shifted through the PUF.

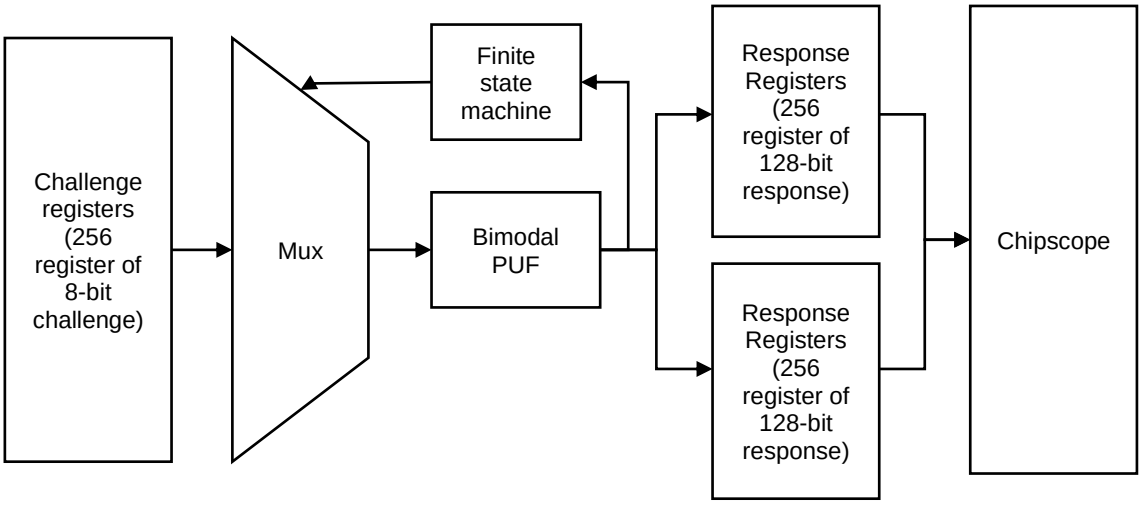

Figure 4-12: Data acquisition setup

4.5 Findings and Discussions

4.5.1 Bimodal PUF Characterisation

From the data acquired in the previous section, the bimodal PUF can be characterised to get information about its uniqueness, reliability, and throughput.

To obtain the information on how many unique numbers the PUF has, the “tabulate” function on MATLAB was used. The next step was to calculate the uniqueness value using equation . Figure 4-13 shows the comparison of the uniqueness and average reliability of regular PUF when a different enabling signal is applied. It can be seen that there is an inverse correlation between the reliability and uniqueness of the PUF. This finding is similar to the results in Chapter 3 of this thesis. There is also an increase in the reliability for both ROPUF and RCPUF when a shift register is used as an internal pulse generator-based enabling signal. The reliability of ROPUF is doubled when the shift register-based internal pulse generator is used. On the other hand, the reliability improvement in RCPUF is not that significant compared to the uniqueness decrement.

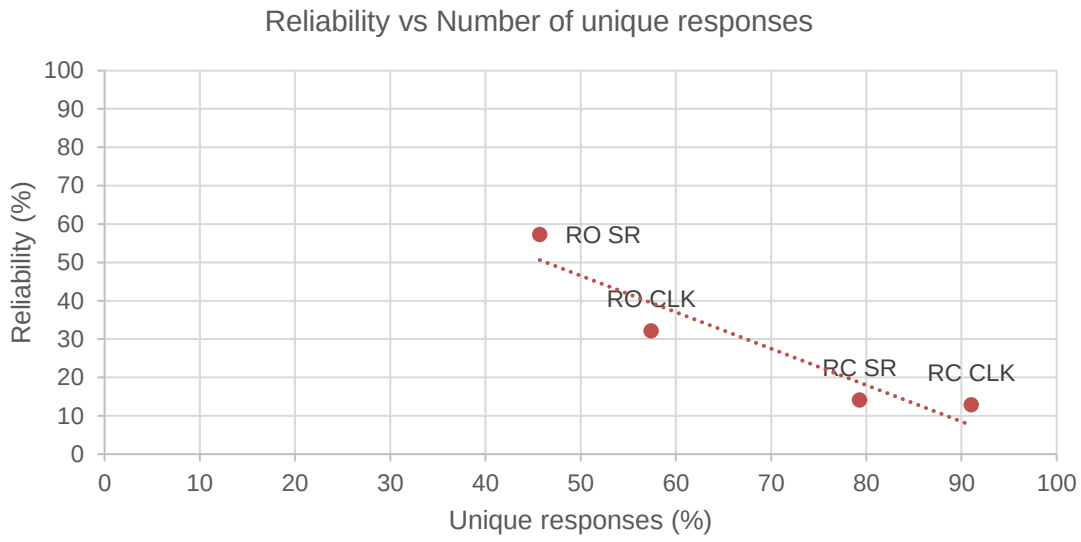

Figure 4-13: Reliability VS Number of unique responses in the regular PUF

The efficiency (π) calculation was done using equation (4-8). However, since the contribution factor (α, β, γ) and (δ, ε) is unknown, it will be assumed that these parameters are equal, following the constraint given in equation (4-9). Therefore, $\alpha = \beta = \gamma = 1/3$ and $\delta = \varepsilon = 1/2$. Another assumption that needs to be made is regarding the P_{max} in equations (4-6) and (4-7). Because the PUF is implemented

in the same device, it is assumed that the P_{max} is the same for every design in this experiment. Thus, the differences in power consumption (ΔP) between the legacy design and the new design is 0, as shown in Table 4-4. The total LUT available in Kintex-7 FPGA is 203,800 LUT and it will be used to calculate the percentage of resource usage A. Using the XPower Analyzer, it was discovered that the power consumption and junction temperature for all designs is 162 mWatt and 25.3 Celsius respectively. The throughput for each PUF design can vary as shown in Table 4-4. The throughput is converted into a unit of percentage by assuming that t_{max} is 800 MHz, which is the same as the maximum I/O switching frequency of Kintex-7 FPGA as stated in its data sheet [25].

Table 4-4: Efficiency of the new PUF design compared to the legacy PUF

Var	ROPUF CLK	ROPUF SR	RCPUF CLK	RCPUF SR
r_1	31	59	12	15
r_2	0	0	0	0
u	58	45	91	78
t	73.43	74.62	118.37	119.78
A	2048	2049	1280	1281
ΔP	0	0	0	0

Figure 4-14 illustrates the efficiency of the new PUF design compared to the legacy ROPUF design with a system clock as its enabling signal. It can be seen that even though the increase in efficiency is below 10%, the use of a ring counter as the source of randomness for PUF is a benefit compared to the legacy ROPUF. It can also be seen that by changing the source of randomness from a ring oscillator to a ring counter, it increases the efficiency of the new design compared to changing its enabling signal from a system clock to an internal pulse generator.

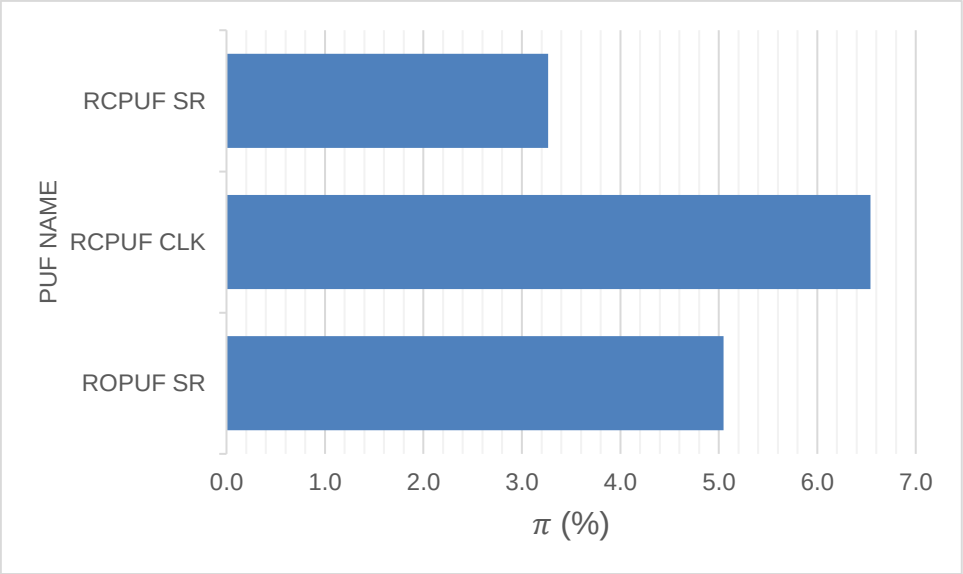

Figure 4-14: Efficiency of the new PUF designs compared to the legacy PUF (ROPUF with a system clock as its enabling signal)

Since the first experiment returned a positive result, i.e. an efficiency increase based on equation (4-4), following the flowchart in Figure 4-11, the experiment continued with the implementation of the bimodal PUF to see if it will return a similar behaviour as the regular PUF. A comparison between the average reliability and the uniqueness of the bimodal PUF has been presented in Figure 4-15. It shows that the use of a shift register-based internal pulse generator as the enabling signal can increase the reliability of the bimodal PUF. However, in contrast with the results in Figure 4-13, the reliability of the bimodal RCPUF increases by 40% when using the shift-register-based internal pulse generator. Figure 4-15 also shows that the reliability of the bimodal ROPUF decreases but not by a significant amount.

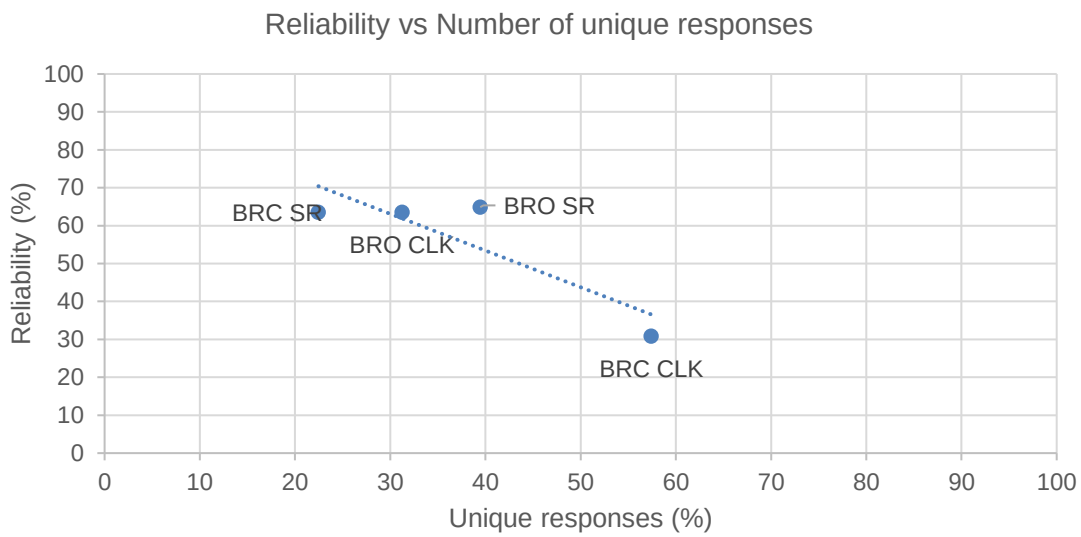

Figure 4-15: Reliability VS Number of unique responses from the bimodal PUF

Table 4-5 and Figure 4-16: illustrate the efficiency of the new bimodal PUF design compared to the legacy ROPUF.

Table 4-5: Efficiency of the bimodal PUF design compared to the legacy PUF

var	BROPUF CLK	BROPUF SR	BRCPUF CLK	BRCPUF SR
r_1	31	70	41	31
r_2	96	60	21	96
u	32	39	57	23
t	84.52	116.85	74.88	76.20
A	4096	4097	2816	2817
ΔP	0	0	0	0

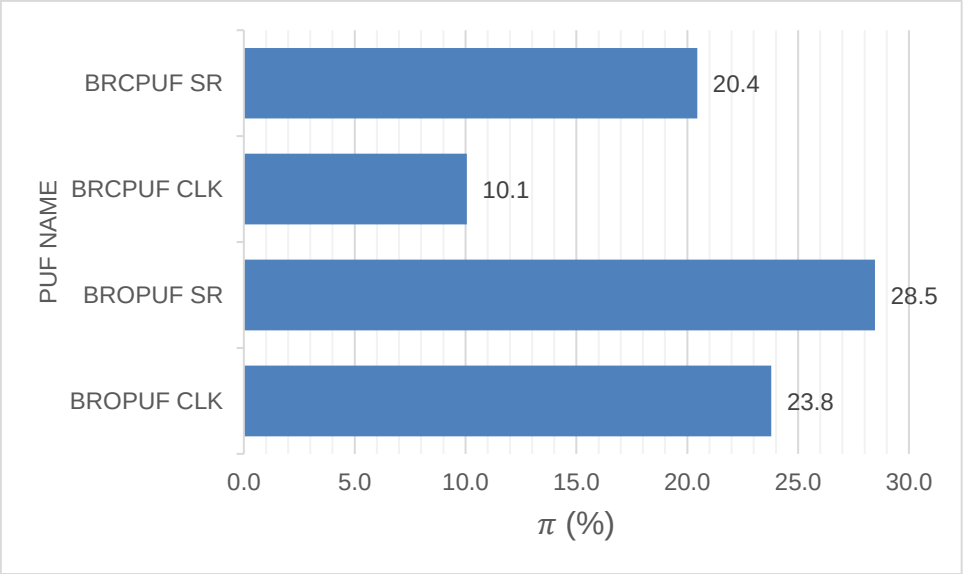

Figure 4-16: Efficiency of the bimodal PUF designs compared to the legacy ROPUF

It can be seen that the efficiency of the new design is increased compared to the efficiency in Figure 4-14. It can also be seen that by changing the source of randomness from a ring oscillator to a ring counter, it increases the efficiency of the new design compared to changing its enabling signal from a system clock to an internal pulse generator. Figure 4-12 shows that the bimodal ROPUF has better efficiency than the bimodal RCPUF. However, to overcome the scalability issue of IJTAG security measures, resource usage is a critical factor that needs to be considered. Figure 4-17 displays a closer look into the FPGA resource usage of each PUF design. It can be seen that the bimodal RCPUF only adds 40% of the FPGA resources compared to the regular PUF, but it has two different responses that can be used to add more functionality. Therefore, it can be concluded that the bimodal RCPUF is an ideal solution for IJTAG security measures without the presence of a scalability problem.

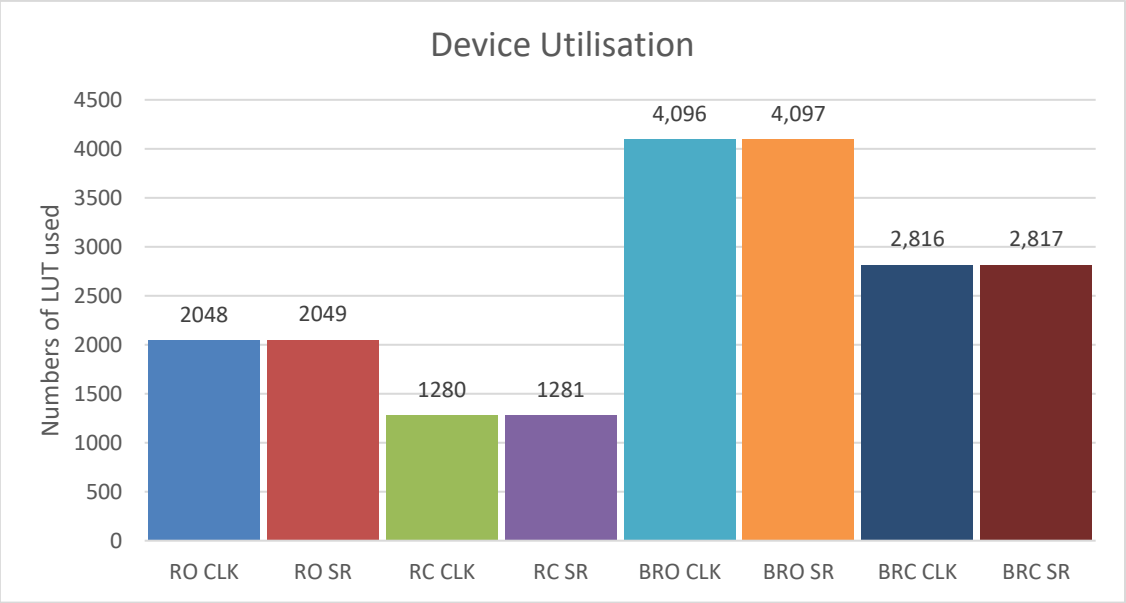

Figure 4-17: Comparison of device utilisation

4.5.2 Security Analysis

In this section, the time needed to unlock the SIB will be discussed. For an authorised user who has the right answer to the challenge to unlock the SIB, the unlocking process takes N number of clocks. It requires five clocks to get the TAP controller ready (TRST). If the PUF has a C-bit challenge, it will take a C clock cycle to shift the challenge to the PUF. If the challenge generates a correct response, then it will be shifted to the comparator. At the same time, an R-bit secret key is shifted to the comparator. The comparison process takes another two clock cycles. In total, the clock cycle to unlock the SIB is $N = 5 + C + R + 2 = C + R + 7$ clock cycle. For the case of the proposed security mechanism, the challenge is 8-bit, and the response is 128-bit. Therefore the total clock cycle to gain access to the embedded instrument is a 143 clock cycle. Compared to a similar approach in [11] that takes up to 65560 clock cycles to unlock the SIB, it is clear that the proposed security mechanism is faster when it comes to performing the unlocking process.

If the attacker succeeds in guessing the challenge for PUF, but they have no idea about the obfuscation provided by the second PUF response, they will never get

the correct output data from the instrument. Thus the extracted data cannot be used for reverse engineering. Even if the hardware Trojan activation command is shifted to the instrument, the output will not affect another instrument as the output data was obfuscated.

With this layered security protocol, not only it will prevent the instrument from being accessed by an unauthorised party, but it will also preclude the use of stolen data. It also protects another instrument from the effect of Hardware Trojan activation.

There is a limitation in this security mechanism. The output data comes out from the instrument and TDO in an obfuscated form. This means that the output data cannot be used as an input for other instruments in the same network. Therefore, this security mechanism cannot be used in mission mode, only in testing mode. To use this mechanism in mission mode, one can use the mechanism as described in [26] where an LFSR is utilised to de-obfuscate the output data before it comes out of the instrument.

Table 4-6 compares the proposed IJTAG security measures with different measures from the literature. The proposed security excels when compared to the other proposed security measures in that it not only provides security focused on the access to the IJTAG network, but it also secures the output data from the embedded instrument. While paper [12] also provides access protection and data security, its implementation using a chiper core makes it have a high area overhead, affecting scalability. Hardness is a measure of how easy security can be broken by an adversary. Papers [8] and [9] are the easiest as they only use one static secret key (password) to unlock the access to the IJTAG network. The dynamic password means that the secret key can be changed depending on what challenge applies to the PUF. The secret key will also be different for every IJTAG implementation, and its security measures will be on a different chip.

Table 4-6: Comparison of the proposed method with other IJTAG security measures

Parameter	Paper [8]	Paper [9]	Paper [10]	Paper [11]	Paper [12]	proposed
Area overhead	Low	Medium	High	Medium	High	Medium
Scalability	High	Medium	High	Low	Low	High
Password	Static	Static	Dynamic	Dynamic	Dynamic	Dynamic
Hardness	Low	Medium	High	High	High	High
Data Protection	N/A	N/A	N/A	N/A	Yes	Yes

4.6 Conclusion

A multi-layer security mechanism to protect access and the data read in the IJTAG network using bimodal PUF is proposed. The proposed technique benefits from a single challenge and generates two responses that are used to unlock the hardware and obfuscate the output data. Hence the utilisation of PUF to obfuscate the output data provides a high level of data protection. The analysis shows that it is faster to unlock when accessed by the authorised party compared to previous work. This means that the time needed for testing will be faster while maintaining its security. The use of bimodal PUF for generating secret keys also adds to the advantages as each chip has a unique characteristic due to the process variation. This can be used to create a dynamic key for every single chip.

A PUF performance comparison between the system clock and the internal pulse generator as an enabling signal has also been made. It was discovered that the reliability of the PUF response is increased when an internal pulse generator is used. However, it sacrifices the uniqueness of the PUF as the reliability and uniqueness are discovered to have an inverse correlation. Therefore, it is up to the designer to choose which configuration to use to meet the requirements of the system.

4.7 Reference

[1] IEEE Computer Society, *1500 - 2005 - IEEE standard testability method for embedded core-based integrated circuits*. Institute of Electrical and Electronics Engineers, 2011.

[2] IEEE Computer Society. Test Technology Standards Committee., Institute of Electrical and Electronics Engineers., and IEEE-SA Standards Board., *IEEE standard for access and control of instrumentation embedded within a semiconductor device - 1687*. .

[3] Test Technology Standards Committee, *IEEE Standard Test Access Port and Boundary Scan Architecture*, vol. 2001. 2001.

[4] R. Johnson, “Sergei Skorobogatov Defends Backdoor Claims - Business Insider,” *Business Insider*, 2012. [Online]. Available: https://www.businessinsider.com/sergei-skorobogatov-defends-backdoor-claims-2012-5?r=US&IR=T. [Accessed: 30-May-2020].

[5] S. Skorobogatov and C. Woods, “Breakthrough silicon scanning discovers backdoor in military chip,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 7428 LNCS, pp. 23–40, 2012.

[6] F. Majéric, B. Gonzalvo, and L. Bossuet, “JTAG Fault Injection Attack,” *IEEE Embedded Systems Letters*, vol. 10, no. 3, pp. 65–68, Sep. 2018.

[7] R. Elnaggar, R. Karri, and K. Chakrabarty, “Securing IJTAG against data-integrity attacks,” in *Proceedings of the IEEE VLSI Test Symposium*, 2018, vol. 2018-April, pp. 1–6.

[8] J. Dworak, A. Crouch, J. Potter, A. Zygmontowicz, and M. Thornton, “Don’t forget to lock your SIB: Hiding instruments using P16871,” in *Proceedings - International Test Conference*, 2013, pp. 1–10.

[9] H. Liu and V. D. Agrawal, “Securing IEEE 1687-2014 Standard Instrumentation Access by LFSR Key,” in *Proceedings of the Asian Test*

Symposium, 2015, vol. 2016-Febru, pp. 91–96.

[10] R. Baranowski, M. A. Kochte, and H. J. Wunderlich, “Fine-grained access management in reconfigurable scan networks,” *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 34, no. 6, pp. 937–946, Jun. 2015.

[11] K. Sudeendra Kumar, N. Satheesh, A. Mahapatra, S. Sahoo, and K. K. Mahapatra, “Securing IEEE 1687 standard on-chip instrumentation access using PUF,” in *Proceedings - 2016 IEEE International Symposium on Nanoelectronic and Information Systems, iNIS 2016*, 2017, pp. 56–61.

[12] S. Kan, J. Dworak, and J. G. Dunham, “Echeloned IJTAG data protection,” in *Proceedings of the 2016 IEEE Asian Hardware Oriented Security and Trust Symposium, AsianHOST 2016*, 2017, pp. 1–6.

[13] A. Das and N. A. Touba, “A Graph Theory Approach towards IJTAG Security via Controlled Scan Chain Isolation,” in *2019 IEEE 37th VLSI Test Symposium (VTS)*, 2019, vol. 2019-April, pp. 1–6.

[14] Xuanle Ren, Vítor Grade Tavares, and R. D. Shawn Blanton, “Detection of illegitimate access to JTAG via statistical learning in chip - IEEE Conference Publication,” in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015.

[15] X. Ren, R. D. S. Blanton, and V. G. Tavares, “Detection of IJTAG attacks using LDPC-based feature reduction and machine learning,” in *Proceedings of the European Test Workshop*, 2018, vol. 2018-May, pp. 1–6.

[16] A. Maiti, R. Nagesh, A. Reddy, and P. Schaumont, “Physical Unclonable Function and True Random Number Generator : a Compact and Scalable Implementation,” in *Great Lakes Symposium on VLSI (GLSVLSI)*, 2009.

[17] M. Naveed Aman, S. Taneja, B. Sikdar, K. C. Chua, and M. Alioto, “Token-based security for the internet of things with dynamic energy-quality tradeoff,” *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 2843–2859,

Apr. 2019.

[18] S. Mulhem and W. Adi, “New Mathblocks-Based Feistel-Like Ciphers for Creating Clone-Resistant FPGA Devices,” *Cryptography*, vol. 3, no. 4, p. 28, Dec. 2019.

[19] J. D. Bekenstein, “How does the entropy/information bound work?,” in *Foundations of Physics*, 2005, vol. 35, no. 11, pp. 1805–1823.

[20] J. Wu and M. O’Neill, “On Foundation and Construction of Physical Unclonable Functions,” 2010.

[21] F. Kodytek, R. Lorencz, J. Bucek, and S. Buchovecka, “Temperature Dependence of ROPUF on FPGA,” *Proceedings - 19th Euromicro Conference on Digital System Design, DSD 2016*, pp. 698–702, 2016.

[22] D. B. Thomas and W. Luk, “FPGA-Optimised Uniform Random Number Generators Using LUTs and Shift Registers,” in *2010 International Conference on Field Programmable Logic and Applications*, 2010, pp. 77–82.

[23] D. B. Thomas and W. Luk, “The LUT-SR Family of Uniform Random Number Generators for FPGA Architectures,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 4, pp. 761–770, Apr. 2013.

[24] H. G. Schaathun, “Evaluation of splittable pseudo-random generators,” *Journal of Functional Programming*, vol. 25, Feb. 2015.

[25] Xilinx and Inc, “Kintex-7 FPGAs Data Sheet: DC and AC Switching Characteristics,” 2011.

[26] M. Randa, M. Bozdal, M. Samie, and I. K. Jennions, “Layered Security for IEEE 1687 Using a Bimodal Physically Unclonable Function,” *Procedia Manufacturing*, vol. 16, pp. 24–30, Jan. 2018.

5 CONCLUSION AND FUTURE WORKS

5.1 Addressing the Aim and Objectives of the Research

The advancement of semiconductor technology has both advantages and challenges such as its testability. While the IJTAG has become the new standard to overcome the testability problem of embedded instruments, it also presents a security concern. Future embedded systems, with added functionality and of a smaller size, present with an overlapping problem in terms of reliability and security of the systems. This thesis aims to improve the security of the embedded system by advancing the design-for-testability to design-for-security.

Past approaches to design-for-security have been discovered to have inefficiency and scalability issues. This thesis presents a novel physically unclonable function as a form of primitive security that allows for the untangling of the issues mentioned above. It adds to the security of both the IJTAG network and the data while also performing as a digital sensor for observing unauthorised modifications of the configuration file for applications in multi-tenant FPGA.

This aim has been achieved by fulfilling the following objectives:

Objective 1: *Develop and characterise a novel random number generator design based on the ring counter circuit.* A True Random Number Generator (TRNG) based on the Ring Counter (RC) circuit has been developed and used as a model to study the behaviour of delay-based random number generators when implemented in sub-nano millimetre (sub-nm) IoT devices. It has been observed that the delay-based random number generator is still able to perform well in sub-nm devices despite its limitation in a short periodicity. The experiment also revealed the fact that using the suggested minimum input of the NIST SP800-22 does not return a meaningful result. Thus, it is a good practice to have an input that is 100 times bigger than the minimum recommendation of the NIST SP 800-22 standard.

Objective 2: *Develop and characterise a novel digital physically unclonable function based on the ring counter circuit*. A physically unclonable function utilising ring counter circuits (RCPUF) as its source of randomness has been

developed and implemented as a digital sensor to detect unauthorised modifications to the FPGA configuration file in multi-tenant FPGA. The digital sensor based on the RCPUF is proven to have a high sensitivity to changes in the FPGA configuration file as it can react a one logic gate change. The digital sensor is also affordable in terms of the FPGA resource usage as it was designed not to require the use of an error correction algorithm. This experiment also presents a new definition of the reliability and uniqueness parameters for PUF characterisation. This new definition minimises the environmental influence on the measured PUF parameter. Therefore the data obtained from the measurement is more accurate and trusted.

Objective 3: *To develop and characterise bimodal RCPUF (BRCPUF) to secure access to the IJTAG network as well as its output data.* A new class of PUF that can produce two simultaneous responses from a single challenge has been developed. The novel PUF is called bimodal PUF. The bimodal PUF is based on the RCPUF as in objective 2. On the process of developing the bimodal PUF, a comparison between the use of a system clock and internal pulse generator was made. It was discovered that the internal pulse generator increases the reliability of the PUF as well as the bimodal PUF. However, there is an inverse correlation between the uniqueness and the reliability parameter. Therefore there is a trade-off that needs to be considered to satisfy the requirements of the system.

5.2 Future work

The hardware-oriented security measures presented in this thesis are already capable of providing a secure and trusted environment for the system that it is assigned to, which has been amply demonstrated in Chapters 2, 3, and 4. However, from the study conducted during the research, there are many exciting pieces of research left to be conducted. Nevertheless, to keep future research in context with the research of this thesis, the following topic might be of interest to conduct in the future.

5.2.1 Environmental influence on the sub-nm TRNG

In Chapter 2, a study on the behaviour of a TRNG in sub-nm technology has been conducted with a focus on how the exceptional structure of the sub-nm device affects the behaviour of the TRNG. However, while the environmental variation within the range given by the current semiconductor manufacturer is known to be harmless to the behaviour of the system, the delicate structure of the sub-nm semiconductor device might present with different behaviour when operated with an environmental variance, either negative or positive. Sub-nm technology, which possibly is the quantum computer that requires a temperature close to the absolute zero to work, is now getting closer to working at room temperature through the development of diamond-based material. This advancement requires the TRNG that becomes the root of the trust in the sub-nm device to be tested in different environmental situations, such as a variance in temperature or variance in voltage supply.

5.2.2 Integration of design-for-security in the Electronic Design Automation (EDA) tool Design Rule Checking (DRC)

Modern EDA tools are designed to maximise the performance of the design by performing a design rule check. However, the recent discovery of Meltdown [1] and Spectre [2] has raised the awareness of semiconductor industries were improving performance without considering security can be fatal to the privacy and security of the users. However, there have still been no improvements made to the EDA tools in terms of including a security aspect in the Design Rule Checking (DRC) routine. With the multitude of research that has been conducted about hardware security, it is now possible to compile the roots of malicious behaviour or a security threat at the register transfer level of design. Traditionally, a formal analysis is done to discover the flaws or malicious code in design. However, the laborious process of a formal analysis is what makes the design house reluctant to perform the task. Therefore an improvement in the EDA tool to include the security aspect in their DRC process would be an interesting topic to work on, considering that not many people have done similar research.

5.3 References

[1] M. Lipp et al., “Meltdown,” World Watch, vol. 9, no. 3, pp. 23–31, Jan. 2018.

[2] P. Kocher et al., “Spectre Attacks: Exploiting Speculative Execution,” Communications of the ACM, vol. 63, no. 7, pp. 93–101, Jan. 2018.

APPENDICES

Appendix A Raw data of the ring counter based random number generator

The NIST SP 800-22 rev 1a suggest that to be able to use their statistical test suite for random number generator characterisation, at least one million bit of random number need to be provided as an input. However, it is discovered in section 2.5 that in order to produce a meaningful and consistent result, the test suite need to be tested with higher input bit. In our experiment, at least ten million bit input is needed to get a consistent input.

According to the NIST SP 800-22 rev 1a, there are two method to analyse the result of the statistical test suite. It is recommended to always use the first method. However, when using the first analysis method did not satisfy the user, the NIST standard allow the user to use the second method. The first method is by quantitatively looking at the p-value output as in Table 2-4. The second method to analyse the result is more qualitative, which is by using graphical presentation, which means plotting the p-value table into a bar diagram or similar diagram. The second method needed the user to have their intuition to judge whether the random number generator under-test is satisfy their randomness requirement or not.

Along with the research, it is discovered that a heat map of the random number generated also can provide a meaningful information to analyse the characteristic of the random number generator. Figure A-1 is a heat map of one million random number generated by the ring counter-based random number generator. It shows that there are 32 consecutive repetition of random number with 1000-bit length. This mean, if the RNG is set to produce 1000-bit for every generation, it will produce the same string of number for 32 clock cycle. In real case scenario, this behaviour is more than enough to be used for cryptography or any other security related application.

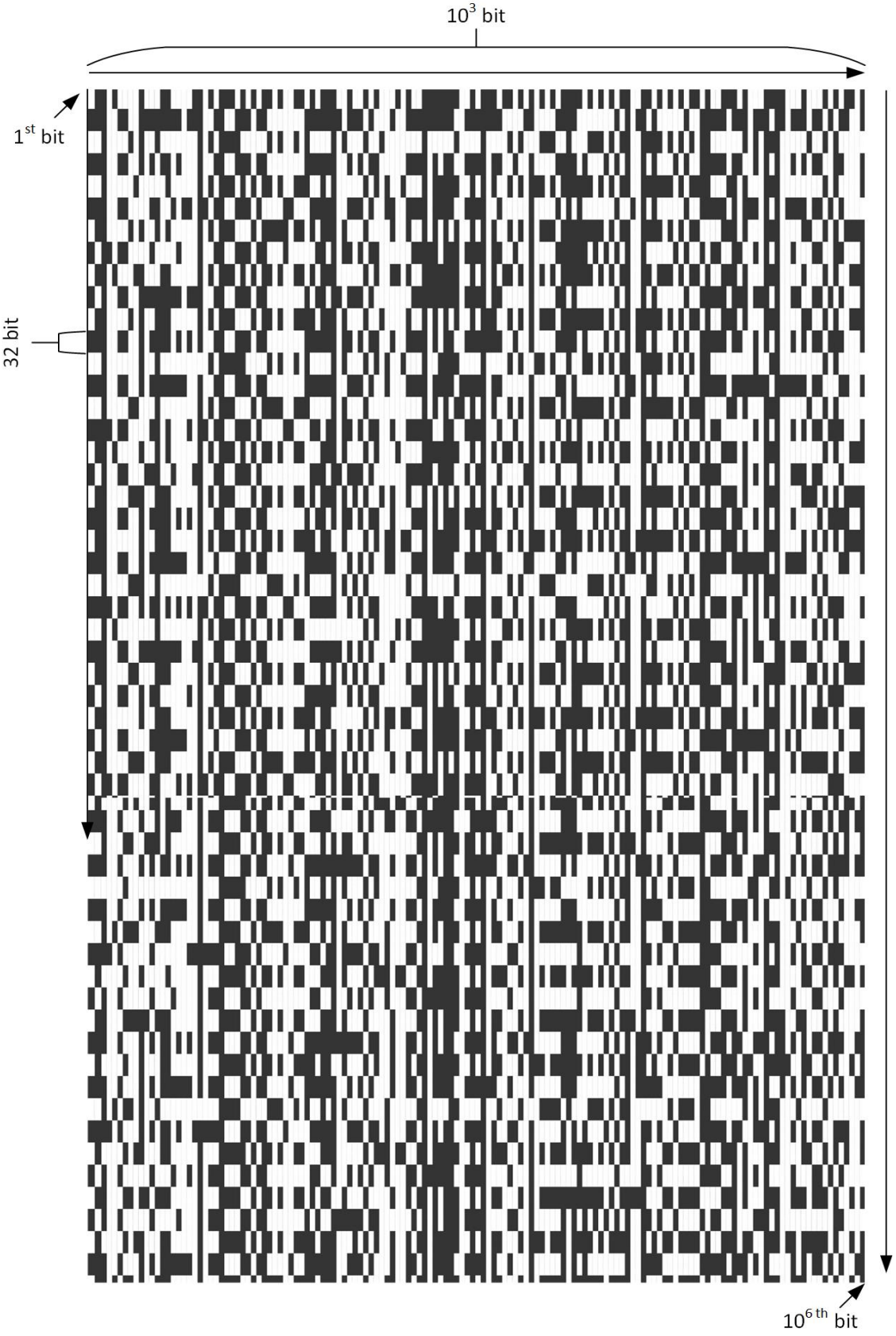

Figure A-1: Heat map of one million bit random number

Appendix B Source code for ring counter-based random number generator and physically unclonable function

The ring counter-based random number generator is implemented in Kintex-7 FPGA from Xilinx. It implemented using VHDL using the ISE 14.7 as the IDE. Here we break down the source code for a better understanding of the thesis.

Table B-1: Source code for RCRNG comparator

top_file		In hierarchical design of VHDL, top file is the interface where an instance of a function underneath it gets implemented. All components that needed to build the RCRNG is declared in this file and instantiated. The component declaration defines the ports of all function underneath the top file. In the case of RCRNG, the components declared on the top file are the enabling signal from SRL16E and the core file, which consist of the source of randomness and the signal processing component such as frequency counter and comparator. To simplify the implementation process, the core file is not hard-coded but generated using the `generate` command.
	Core components	The core components for the RCRNG is declared in this file. It consists of a pair of SRL16E-based ring counter, frequency counter, counter stopping logic, and comparator.

		Frequency_counter	Each ring counter connected to a frequency counter. The frequency counter implemented using a 16-bit binary counter.
		counter_stop	This logic is implemented to stop the other counter from counting whenever a counter is overflow.
		comparator	The comparator logic is used to compare the value of a pair of counters after they stop counting.

If the location of the generated ring counter is not important, there is no need to create a constraint file (*.ucf) to fix the location of the ring counter. However, if the location of the ring counter needs to be fixed, a constraint file need to be created to fix the location of ring counter using the `loc` command.

The following are the code for each file listed above.

Table B-2: Source code for RCRNG top file

```
library IEEE;
Library UNISIM;
use UNISIM.vcomponents.all;
use IEEE.STD_LOGIC_1164.ALL;

entity TOP_FILE is
    Port (
       clk: in std_logic;
       clk2: in std_logic;
       clr: in STD_LOGIC
       );
end TOP_FILE;

architecture Behavioral of TOP_FILE is

attribute loc: string;
attribute rloc: string;
attribute rloc of RN_gen: label is "X0Y0";

signal ro0_out, ro1_out, sro, oflow1, oflow2, compout_flow,
stop, clr1, resp, comp_out: std_logic;
signal countout1, countout0, comp_in1, comp_in2:
std_logic_vector (15 downto 0);
```

```
attribute keep: string;
attribute keep of sro, ro0_out, ro1_out, oflow1, oflow2,
countout1, countout0, compout_flow
,comp_out, comp_in1, comp_in2: signal is "true";

attribute s: string;
attribute s of ro0_out, ro1_out, oflow1, oflow2, countout1,
countout0, stop
: signal is "yes";

COMPONENT CORE_FILE
     PORT(
          clr1: IN std_logic;
          clk: IN std_logic;
          resp: OUT std_logic
          );
     END COMPONENT;

begin
RN_gen:
          for i in 0 to 4000 generate
          begin
          RON_1: CORE_FILE
          PORT MAP(
          resp => resp,
          clr1 => clk2,
          clk => sro
          );
   end generate;

   SRL16E_inst: SRL16e
   generic map (
      INIT => X"AAAA")
   port map (
      Q => sro,      -- SRL data output
      A0 => '1',     -- Select[0] input
      A1 => '1',     -- Select[1] input
      A2 => '1',     -- Select[2] input
      A3 => '1',     -- Select[3] input
      CE => '1',     -- Clock enable input
      CLK => CLK,    -- Clock input
      D => sro       -- SRL data input
   );
end Behavioral;
```

Table B-3: Source code for RCRNG core components

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
library UNISIM;
use UNISIM.VComponents.all;


entity CORE_FILE is
    Port (
    resp: out std_logic;
      clr1: in std_logic;
      clk: in std_logic
    );
end CORE_FILE;

architecture Behavioral of CORE_FILE is

signal ro0_out, ro1_out, oflow1, oflow2, compout_flow,
counter_in, stop: std_logic;

signal countout1, countout0: std_logic_vector (15 downto 0);

attribute keep: string;
attribute keep of ro0_out, ro1_out, oflow1, oflow2, countout1,
countout0, stop
: signal is "true";

attribute s: string;
attribute s of ro0_out, ro1_out, oflow1, oflow2, countout1,
countout0, stop
: signal is "yes";

------------- component instantiate --------------------------
component counter is
    port (
    count_in: in std_logic;
    clr: in std_logic;
    q: out std_logic_vector(15 downto 0);
    oflow: out std_logic;
    enable: in std_logic
    );
end component counter;

component comparator is
     port (
      comp_in1, comp_in2: in std_logic_vector(15 downto 0);
    comp_out: out std_logic
     );
end component comparator;

component counterstop is
```

```
port (
      oflow1: in STD_LOGIC;
      oflow2: in STD_LOGIC;
      clr: in std_logic;
      stop: out STD_LOGIC
);
end component counterstop;
-------------------------------------------------------------
begin

------------------- INST BASED RO -------------------------
      U1: SRL16e
      generic map (
      INIT => X"80")
      port map (
       Q => ro0_out
       A0 => '1',
       A1 => '1',
       A2 => '1',
       A3 => '1',
       CE => '1',
       CLK => CLK,
       D => ro0_out
);
   U2: SRL16e
   generic map (
   INIT => X"80")
   port map (
      Q => ro1_out,
      A0 => '1',
      A1 => '1',
      A2 => '1',
      A3 => '1',
      CE => '1',
      CLK => CLK,
      D => ro1_out
);

--------------------- counter -------------------------------
count_u0: counter
port map (
enable => stop,
count_in => ro0_out,
q => countout0,
oflow => oflow1,
clr=> clr1
);

count_u1: counter
port map (
enable => stop,
count_in => ro1_out,
q => countout1,
```

```
oflow => oflow2,
clr=> clr1
);

counterstop_u0: counterstop
port map (
oflow1 => oflow1,
oflow2 => oflow2,
stop => stop,
clr => clr1
);

comparator_u0: comparator
port map (
      comp_in1 => countout0,
      comp_in2 => countout1,
    comp_out => resp
);

end architecture behavioral;
```

Table B-4: Source code for RCRNG frequency counter

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity counter is
generic (n: natural:=16);

port(
    count_in, clr: in std_logic;
    q: out std_logic_vector(n-1 downto 0);
    oflow: out std_logic;
    enable: in std_logic
);
end counter;

architecture archi of counter is
    signal tmp: std_logic_vector (n-1 downto 0);
    signal halt: std_logic;

begin
    process (count_in)

    begin
        if(count_in'event and count_in ='1')then
           if (enable = '0') then
            if (clr='1') then
```

```
                tmp <= (others => '0'); -- making it zero
            else
                tmp <= tmp + 1;
                      end if;
                            end if;
         end if;
        end process;

   process (tmp)
    begin
        if tmp = 65535 then
            oflow <= '1';
        else
            oflow <= '0';
        end if;
    end process;


  q <= tmp;

end archi;
```

Table B-5: Source code for RCRNG counter_stop

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity counterstop is
    Port (
      clr: in std_logic;
      oflow1: in STD_LOGIC;
      oflow2: in STD_LOGIC;
      stop: out STD_LOGIC
                  );
end counterstop;

architecture Behavioral of counterstop is

begin
process
begin
     if (clr='1') then
           stop <= '0';
     else
           stop <= oflow1 or oflow2;
     end if;
end process;
end Behavioral;
```

Table B-6: Source code for RCRNG comparator

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
library UNISIM;
use UNISIM.VComponents.all;

entity comparator is
    Port (
    comp_in1, comp_in2: in std_logic_vector(15 downto 0);
    comp_out: out std_logic

    );
end comparator;

architecture Behavioral of comparator is

attribute keep: string;
attribute keep of comp_in1, comp_in2, comp_out: signal is true";

begin
process
    begin
           if comp_in1 < comp_in2
                then comp_out <= '1';
                else comp_out <= '0';
                end if;
end process;
end Behavioral;
```

The following is the hierarchical structure of the ring counter-based physically unclonable function (RCPUF).

Table B-7: Hierarchical structure of the RCPUF

Top file			
	Core file		
		mux	Multiplexer is implemented to select which ring counter to be compared.
		Counter	Each ring counter connected to a frequency counter. The frequency

			counter implemented using a 16-bit binary counter.
		Counter_stop	This logic is implemented to stop the other counter from counting whenever a counter is overflow.
		Comparator	The comparator logic is used to compare the value of a pair of counters after they stop counting.

In general, it has the same structure as the RCRNG hierarchical structure. The only difference is that in RCPUF we can choose which ring counter to be compared. This made possible by adding a 2-to-1 multiplexer that has input from the ring counter, and the select input from the challenge of the PUF. Therefore, we will only show the core file to show how the wiring and integration of the mux to the RCPUF architecture.

Table B-8: Source code for RCPUF core file

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;
library UNISIM;
use UNISIM.VComponents.all;


entity CORE_FILE is
    Port (
      resp: out std_logic;
      clr1: in std_logic;
      clk: in std_logic;
      chal1: in std_logic;
      chal2: in std_logic

    );
end CORE_FILE;

architecture Behavioral of CORE_FILE is

signal ro0_out, ro1_out, ro2_out, ro3_out, oflow1, oflow2,
compout_flow, counter_in , stop, regresetflow, q_clr, pulse_i
: std_logic;
```

```
signal countout1, countout0, muxout1, muxout0: std_logic_vector
(15 downto 0);


signal q_resetreg: std_logic_vector (2 downto 0);
signal  chal1_i,  chal2_i,  chal3_i,  chal4_i,  chal5_i,  chal6_i,
chal7_i, chal8_i: std_logic_vector (7 downto 0);

attribute keep: string;
attribute keep of chal1, chal2, ro0_out, ro1_out,
ro2_out,  ro3_out,  muxout1,  muxout0,  oflow1,  oflow2,  countout1,
countout0, compout_flow, counter_in, regresetflow, q_clr, stop,
pulse_i, q_resetreg: signal is "true";

------------- component instantiate --------------------------
      COMPONENT ro3_macro
      PORT(
            input: IN std_logic;
            output: out std_logic
            );
      END COMPONENT;

component ro_linear is
    port (
      clk: in std_logic;
      wave: inout STD_LOGIC
    );
end component ro_linear;

component mux is
 Port (
      sel: in STD_LOGIC_VECTOR (1 downto 0);
      min0, min1: inout STD_LOGIC_vector (15 downto 0);
    mout: inout STD_LOGIC_vector (15 downto 0)
);
end component mux;

component counter is
    port (
      count_in: in std_logic;
      clr: in std_logic;
      q: out std_logic_vector(15 downto 0);
      oflow: out std_logic;
      enable: in std_logic
    );
end component counter;

component comparator is
      port (
      comp_in1, comp_in2: in std_logic_vector(15 downto 0);
      comp_out: out std_logic
      );
end component comparator;
```

```
component counterstop is
port (
      oflow1: in STD_LOGIC;
      oflow2: in STD_LOGIC;
      clr: in std_logic;
      stop: out STD_LOGIC
);
end component counterstop;

COMPONENT pulse_stop
      PORT(
            clk: IN std_logic;
            reset: in std_logic;
            flag: IN std_logic;
            pulse: OUT std_logic
            );
END COMPONENT;

component reg2 is
port (
      clk: in std_logic;
      regin: in std_logic;
      regout: out std_logic_vector(7 downto 0);
      reset: in std_logic
);
end component reg2;

      COMPONENT RO0
      PORT(
            input: IN std_logic;
            output: OUT std_logic
            );
      END COMPONENT;



      COMPONENT RO1
      PORT(
            input: IN std_logic;
            output: OUT std_logic
            );
      END COMPONENT;



------------------------------------------------------------
begin

------------------- INST BASED RO --------------------------
U1: SRL16e
   generic map (
      INIT => X"80")
```

```
   port map (
      Q => ro0_out,
      A0 => '1',
      A1 => '1',
      A2 => '1',
      A3 => '1',
      CE => '1',
      CLK => CLK,
      D => ro0_out
);

U2: SRL16e
   generic map (
      INIT => X"80")
   port map (
      Q => ro1_out,
      A0 => '1',
      A1 => '1',
      A2 => '1',
      A3 => '1',
      CE => '1',
      CLK => CLK,
      D => ro1_out
);

---------------------- mux ----------------------

mux_u0: mux
port map (
      sel => chal1,
      mout => muxout0,
      min0 => countout0,
      min1 => countout1
);

mux_u1: mux
port map (
     sel => chal2,
     mout => muxout1,
     min0 => countout0,
     min1 => countout1
);

--------------------- counter -------------------------------
count_u0: counter
port map (
enable => stop,
count_in => ro0_out,
q => countout0,
oflow => oflow1,
clr=> clr1
);
```

```
count_u1: counter
port map (
enable => stop,
count_in => ro1_out,
q => countout1,
oflow => oflow2,
clr=> clr1
);

counterstop_u0: counterstop
port map (
oflow1 => oflow1,
oflow2 => oflow2,
stop => stop,
clr => clr1
);



------------------ comparator --------------------------------

comparator_u0 : comparator
port map (

       comp_in1 => muxout0,
       comp_in2 => muxout1,
    comp_out => resp

);

end architecture behavioral ;
```