DESIGN, IMPLEMENTATION AND ANALYSIS OF EFFICIENT FPGA BASED
PHYSICAL UNCLONABLE FUNCTIONS

by

Bilal Habib
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Electrical and Computer Engineering

Committee:

_____ Dr. Kris Gaj, Dissertation Director

_____ Dr. Jens-Peter Kaps, Dissertation Co-Director

_____ Dr. Houman Homayoun, Committee Member

_____ Dr. Huzefa Rangwala, Committee Member

_____ Dr. Monson H. Hayes, Department Chair

_____ Dr. Kenneth S. Ball, Dean, Volgenau School of
                                                          Engineering

Date: _____ Summer Semester 2016
                                                George Mason University
                                                Fairfax, VA

Design, Implementation and Analysis of Efficient FPGA based Physical Unclonable Functions

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Bilal Habib
Master of Science
George Washington University, 2010
Bachelor of Science
University of Engineering and Technology, Pakistan, 2005

Director: Kris Gaj, Associate Professor
Co-Director: Jens-Peter Kaps, Associate Professor
Department of Electrical and Computer Engineering

Summer Semester 2016
George Mason University
Fairfax, VA

Dedication

I dedicate this to my loving wife Madeha. She has been a constant source of support during my graduate studies. She worked tirelessly at home and at her job, so that I can focus more on my PhD. To my son Ghani, who has been a cheerleader in my doctoral studies. Lastly, to my mom, Parveen and my late father Habib ur Rehman, who always prayed for my success.

Acknowledgements

I would like to thank my PhD advisor Prof. Kris Gaj for his continuous guidance, support and mentoring. He urged me to explore new ideas and think along mathematical lines for analysis. He thoroughly reviewed my publications before submission. His comments greatly improved the quality of my papers.

I am also very grateful to my co-advisor Prof. Jens-Peter Kaps for his valuable directions. His suggestions in the implementation of low area hardware designs were remarkably useful.

Additionally, I am very thankful to my committee members Prof. Houman Homayoun and Prof. Huzaifa Rangwala for taking keen interest in my research.

I would like to thank my colleagues and friends in the CERG lab. They include Marcin, Umar, Rabia, Aaron, Rajesh, Panasayya and Ahmad. They always provided a congenial company during my stay in the lab.

Additionally, I would like to thank McQ Inc for allowing me to use their equipment for testing.

I am grateful to the Government of Pakistan and the Provost Office of graduate education for their generous financial support, without which all this was not possible.

Lastly, I would like to appreciate the ECE department of George Mason University, which hired me as a teaching assistant for a significant number of semesters. It helped me to cover my expenses during my studies.

Table of Contents

List of Tables

List of Figures

Abstract

DESIGN, IMPLEMENTATION AND ANALYSIS OF EFFICIENT FPGA BASED
PHYSICAL UNCLONABLE FUNCTIONS

Bilal Habib, Ph.D.

George Mason University, 2016

Dissertation Director: Dr. Kris Gaj

With the advent of Internet of Things (IoTs), secure communication between devices is a
big challenge. Billions of new devices and sensors are going to be connected to internet.
To ensure secure communication with the devices we need hardware primitives that are
well suited to the requirements of IoTs. Recent research has led to an increased interest in
security measures, especially in solutions that are physically unique and unclonable.
Physical Unclonable Function (PUF) has been found to be a strong candidate for this
purpose. Since PUF extracts the inherent manufacturing variations of a hardware chip,
therefore it can be used as a finger print of devices. Eventually these finger prints can be
used to authenticate devices and also to generate secure keys for cryptographic functions.
This thesis describes the development of efficient and reliable PUF for FPGAs. Novel
PUFs have been designed for this purpose. Furthermore, it also covers the generation and
analysis of PUF responses in a more coherent and systematic method. For the generation
of PUF responses different bit-generation schemes have been used and their results have

been compared with each other. This novel study was done to determine the best scheme among the most popular schemes developed so far by different researchers. Software scripts were developed for all the schemes. Similarly, for the analysis, new metrics have been presented for the evaluation of PUF responses. Additionally, software scripts have been developed for analysis of PUF responses. These scripts can be applied to any type of PUF.

Design, development, implementation and testing of two major types of PUF have been carried out. One is a memory based PUF: SR-Latch based design. The second is a delay based PUF: Ring oscillator based design. Both designs have been thoroughly tested on FPGA devices. Performance metrics of both designs have been presented and compared to the state of the art PUFs. Experiments were carried out on different FPGA technologies. It was done to prove the applicability and portability of our designs.

One of the major requirements of PUF intended for IoT applications is that the device area must be efficiently utilized. The current state of the art PUFs are expensive for low area implementation. Therefore, in this work a highly efficient PUF has been developed and tested for FPGAs. Additionally, the PUF is very reliable for use at different environmental conditions. It makes it further attractive because the device can be used in broad range of temperature and voltage variations. To regenerate the same PUF response under different conditions we used the error correction scheme. We also presented different schemes that are suitable from the security point of view.

Lastly we presented a prototype of an efficient SR-Latch based PUF design, with two times improvement in area over the state of the art, thus making it very attractive for low-area designs. This PUF is able to reliably generate a 128-bit cryptographic key.

1.   Introduction

Physical Unclonable Functions (PUFs) are physical primitives which produce unclonable and device-specific measurements of silicon Integrated Circuits (ICs). These measurements are then processed to generate device IDs or Keys. It is very similar to the biometric feature extraction of humans. These features can be used for authentication and identification purposes. In ICs, manufacturing process variations results in physical uniqueness due to sub-microscopic level differences. Even with extreme precision, the two ICs manufactured under same conditions cannot have similar features. Thus manufacturer cannot control the development of these features during the manufacturing process. Due to this property the term 'unclonability' is used to describe it. The extraction of these IC specific features is carried out by using different PUF designs. The primary goal of any PUF construction is to extract these features cleanly and reliably. The process of extraction and the subsequent measurements must not be affected by any environmental conditions like voltage, temperature and radiation. It must be mentioned that in humans the biometric features cannot be acquired and they are 'inherent', similarly in ICs the sub-microscopic level features cannot be developed or grown on the IC fabric after the manufacturing process is complete. A common setting in which these physical primitives are used is to apply a stimulus and measure the response. The response of this primitive can be

interpreted as the result of evaluating a function. Since this function is similar to mathematical functions, therefore these functions are referred as Physical Unclonable Functions. Due to the cryptographic setting in which PUFs are used, stimulus and response are called Challenge Response Pair (CRP).

Among other uses, PUFs enable device identification and authentication [9, 29], binding software to hardware platforms [12, 16, 21, and 41] and secure storage of cryptographic secrets [5, 4]. Furthermore, PUFs can be integrated into cryptographic algorithms [22], remote attestation protocols [43] or countering reverse engineering [60]. Today, PUF-based security products are already announced for the market, mainly targeting IP-protection, anti-counterfeiting and RFID applications [63, 64].

1.1.　Motivation

In the last couple of decades there has been an exponential increase in the digital information processing and communication systems. With this phenomenal increase, security challenges are becoming significant. An on-chip PUF (Physical Unclonable Function) can solve these challenges effectively and efficiently. A PUF is a chip-dependant unclonable challenge-response function that can be used to uniquely identify a specific integrated circuit. Furthermore, the PUF itself is tamper resistant against physically invasive attacks. Due to these attributes, a PUF offers security against intellectual property (IP) theft and counterfeiting, and solves issues such as chip authentication, reverse engineering, trusted computing, and secure key generation. The motivation for FPGA based PUF comes from the fact that FPGAs as opposed to ASICs offer a flexible and secure

solution to IP implementations in hardware. The reason for this flexibility is because FPGAs can be configured at any time in the field without any cost associated with it. Similarly FPGAs can offer secure IP implementation. In a practical application, the IP reads the PUF output and compares it with some built-in constant (chip-ID) and if both of them match then it enables the IP to run on this particular FPGA device. This way the IP vendor makes sure that the IP is licensed only for a selected device. The chip ID can be retrieved by the manufacturer during enrollment. Furthermore PUFs can be employed to verify if the system having an FPGA as one component which came from a genuine source. This can be done by extracting the chip-dependent PUF output in the field and comparing it with the one supplied by a genuine source. Another motivation of FPGA based PUFs is that FPGAs offer quick product customization as per market demands compared to ASIC, therefore it is important to investigate the security features of FPGA based PUF designs.

2.		Previous Work

Initially the term PUF was proposed by Pappu et al in [1]. Since 2002, silicon based PUFs have been extensively investigated. The initial proposal of a delay based arbiter PUF was made in [2]. Arbiter PUF was further explored by [3, 4, and 7] to investigate reliability and security features. Although the Arbiter-PUF offers strong PUF properties, it is prone to machine-learning attacks [49]. In [5], the robustness of optical PUF is proposed. PUF based secure processor is presented in [6]. In [9], the unclonability of PUF based RFID tags is investigated. In addition, a security protocol for RFID tags is also presented. In [11], the idea of a Ring-Oscillator (RO) based PUF is presented. In this PUF the challenge is the selection of a pair of ROs. The response is the one-bit comparison result of the frequencies of those ROs. A large-scale characterization of RO based PUF has been done in [23]. In [10] the first SRAM-PUF is presented, in which the start-up values of uninitialized Embedded RAMs are used as a PUF response. From 2008-2013, FPGAs produced by Xilinx and Altera, the start-up values of memory locations were controlled by the chip manufacturer, which rendered SRAM PUF useless for FPGAs. It must be mentioned that in new 28nm (Zynq) FPGAs, SRAM PUF can be employed by using the techniques of power gating [66]. In this technique the Block RAM of FPGA is disconnected from the power line of chip. In [16], Butterfly-PUF is presented, which requires symmetric paths between registers for causing metastability. FPGA tools do not offer complete access to

symmetric design at the wire level, therefore, routing schemes make it hard to achieve symmetric butterfly design on FPGAs. This fact has been verified by [27] for both Arbiter and Butterfly PUF. In [30] and [39], the concept of programmable delay lines is presented, in which the LUT delays are used to create a metastable condition which is further used to develop a PUF and TRNG respectively. In [34], Maiti et al. presented an RO PUF, in which multiplexers were introduced in the ring to select different paths inside the ring. In [37], the number of configurations of ring oscillator has been improved by introducing a latch in the path of a ring, making it impossible to compare a latch-path with no-latch-path. In addition to RO PUF, extensive research has been done to investigate the Latch PUF. In [38], an Error Correction Code (ECC) encoder and decoder have been developed for PUF based light weight applications. In [44] and [53] it has been revealed that robust responses from SR-Latch based PUF can be derived. In Latch PUF, the set reset latch is triggered into a metastable state. After some time the latch stables into a particular state. The response bit is based on the stable state of the SR-Latch. In [44], the response bits are derived from the location of the random latches. Bitline PUF has been proposed in [62]. It is an extension of SRAM PUF. In [67] the interface for secure PUF has been described.

2.1.   Types of PUFs

Fig. 1 shows the types and sub-types of different PUF designs.

```
                          ┌──────────────┐
                          │     PUF      │
                          └──────┬───────┘
              ┌──────────────────┴──────────────────────┐
              ▼                                          ▼
    ┌──────────────────┐                      ┌──────────────────┐
    │  Non-Silicon PUF │                      │   Silicon PUF    │
    └────────┬─────────┘                      └────────┬─────────┘
             ▼                            ┌─────────────┴──────────┐
                                          ▼                        ▼
```

Non-Silicon PUF	Memory Based	Delay Based
• Paper PUF • Optical PUF • CD-PUF • Magnetic PUF • Acoustical PUF • RF-DNA..	• SRAM PUF • Butterfly PUF • Latch PUF • Flip Flop PUF • Bistable ring PUF • MECCA PUF • Bit-line PUF	• Ring Oscillator PUF • Arbiter PUF • Glitch PUF • LUT based PUF **Other types:** • Current based PUF •
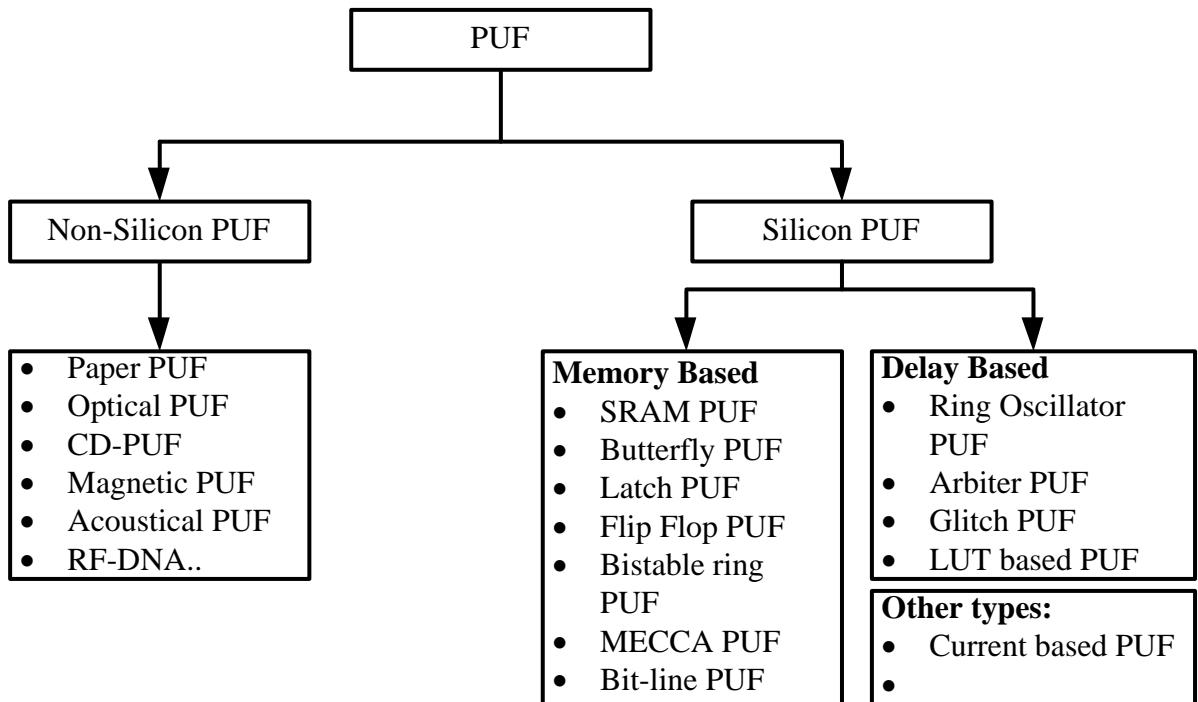
Figure 1: PUF Types and classifications

a) SRAM-based PUF

 It was initially proposed by Guajardo et al. [10], the SRAM-based PUF uses the initialization values of dedicated SRAM blocks. They consider a range of memory

locations as challenges and start-up values at these locations as responses. These values

depend on the small asymmetry between two cross-coupled inverters Fig. 2, ensuring that

the start-up values will always be the same with high probability. Guajardo et al. defined

this kind of PUF as intrinsic, because the PUF generating circuit is directly present in the

design to protect. The main drawback of SRAM-based PUF with FPGAs is that most FPGA

manufacturers initialize the embedded memory blocks to zero before loading the bitstream

to avoid shortcuts in the reconfigurable circuitry. It must be added that SRAM PUF has

been adopted by Microsemi and Altera [72] in their devices.
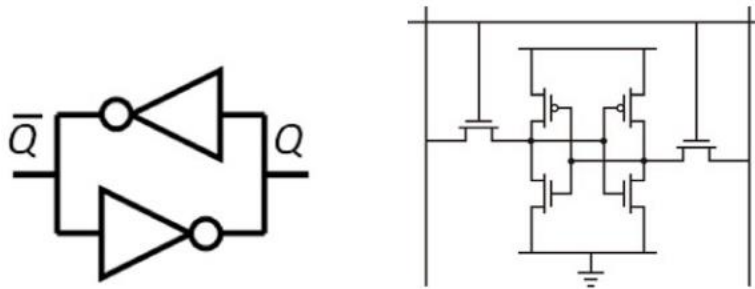
Figure 2: Logical circuit of an SRAM PUF (left). Electrical circuit of an SRAM cell (right) in CMOS [28].

b) Flip-flop PUF

It was proposed by Maes et al. [18], the flip-flop PUF uses flip-flops start-up values as responses similarly to the SRAM-based PUF. Maes et al. imagined this PUF because it is possible to prevent flip-flops from being reset. Hence, this allows having an efficient PUF suitable for every FPGA.

c) Butterfly PUF

It was proposed by Kumar et al. [16], the Butterfly PUF is another solution to overcome the SRAM PUF reset drawback. It consists in two cross-coupled latches initialized with two different values to have an unstable operating point. The latches are initialized on an external signal. When this one is released, the stable state depends on the slight differences between the connecting wires which are designed using symmetrical paths on the FPGA matrix. The Butterfly PUF needs manual routing to have symmetric paths and its performance highly depends on the targeted FPGA [51]. In Fig. 3, two cross coupled latches are shown.
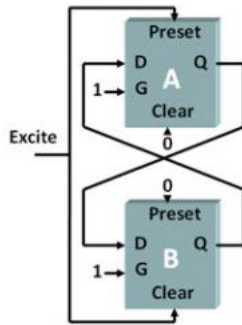
Figure 3: Schematic circuit of a butterfly PUF cell

d) MECCA PUF

Proposed in 2011 by Krishna et al [45]. In this PUF the write failure in SRAM memory cells is caused. It results into a data being written in the SRAM memory which is dependent on the chip. MECCA PUF has not been implemented on the real chip and the results come only from the simulation.

e) LUT-based PUF

It was proposed by Anderson [31], the LUT-based PUF harnesses the FPGA's LUT structure. It uses LUTs from the same basic logic block (slice or ALM), configured in shift-register, and the carry-chain logic. This PUF relies on delays introduced by the LUTs and the multiplexers. It uses the presence or absence of glitches along the carry chain to determine the output bit. This PUF has the advantage to be completely described in HDL.

f) Ring oscillator PUF

It was introduced by Suh et al. [11], it mainly relies on a self-oscillating circuit and a counter. The ring oscillator produces an oscillating signal with a delay-dependent frequency. Besides, the counter measures the number of positive edges over a period of time. The obtained value is a good representation of the ring oscillator intrinsic delay. The main drawbacks of this kind of PUF are the limited number of possible challenges and the significant dynamic power consumption. The input is the selection of a pair of oscillators while the output is the one bit result after comparing their frequencies as shown in Fig. 4.
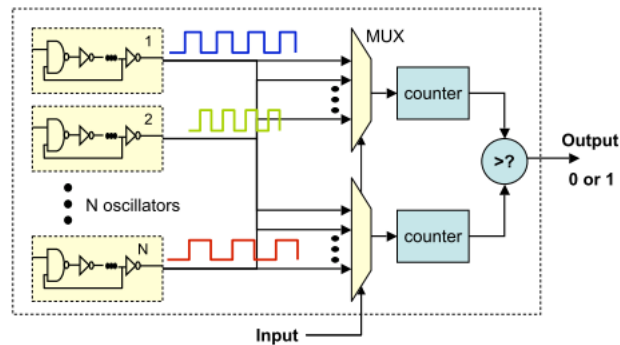
Figure 4: Ring oscillator based PUF circuit

Classification into Strong and Weak PUF

PUFs can be divided into strong PUFs and weak PUFs. The strong PUF includes the optical PUF, arbiter PUF, lightweight secure PUF, etc. The weak PUF mainly includes the memory-based PUF, RO PUF and glitch PUF. The security of strong PUFs is based on their high entropy content providing a huge number of unique challenge-response pairs (CRPs), which can be used in authentication protocols. On the other hand, weak PUFs exhibit only a small number of CRPs to be applied. Although they are not applicable to authentication protocols, the corresponding responses of weak PUFs can be used as a device unique key or seed for conventional encryption systems, while maintaining the advantages of physical unclonability. In order to enable the extraction of cryptographic keys from PUFs, the fuzzy extractor [17] is necessary. In [14], fuzzy extractor has been implemented on FPGA to generate cryptographic keys.

Issues associated with Ring Oscillators

- Systematic Variation

The frequency of RO PUF is dependent on the location of RO. In [8 and 34] it has been shown that the frequency of rings at the middle of a chip is higher than the rings located at the edges of the chip. This behavior is called 'Systematic Variation'. Fig. 5 shows this behavior. The solution proposed by Maiti et al in [34], is that while comparing the frequencies of two ring oscillators, care must be taken because too far rings will have known results from frequency comparison. They proposed to compare only the neighboring ring oscillators. In this way the effect of Systematic variation will be minimal.
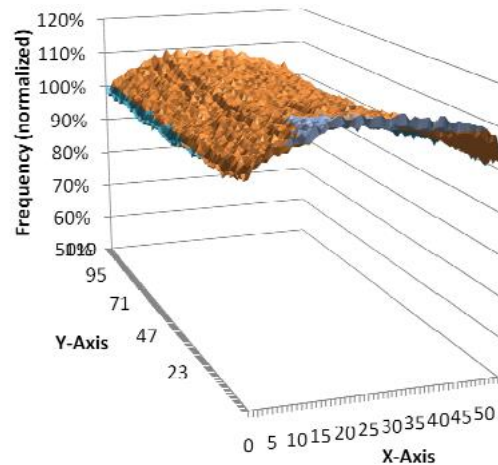
Figure 5: Systematic Variation Tuan et al [42]

- Dependence on Voltage

The frequency of rings is highly dependent on the voltage supplied at the core of FPGA. These two values are directly related with each other. This problem can be solved by using the PUF only at the rated voltage.

- Dependence on Temperature

The temperature and frequency of rings have inversely relationship with each other. Qu et al have proposed a method of 'Temperature Aware Rings co-operation' in [20]. With this method, 80% more rings can be used to generate a reliable bit.

- Locking of Rings

Costea et al [33] showed that rings can lock with other, resulting in exhibition of same frequency when allowed to oscillate. This locking behavior occurs if two rings are

implemented very close to each other inside the FPGA fabric and if they are allowed to oscillate at the same time. This problem can be solved by running only one ring to oscillate at any particular time. Or two rings should not share the same resources like switch boxes inside CLB.

- Implementation Size

The number of CRPs scales linearly with implementation size of regular RO-PUF. Therefore number of rings implemented are bounded by the logical resources available in FPGAs.

New Directions and Improvements

The following are the improvements reported for RO-PUF.

a) Increasing the CR space: Maiti et al proposed that the challenge Response pairs of RO-PUF can be increased by using the Identity mapping functions.

b) Ganta et al proposed the concept of S-ArbRO. In S-ArbRO the CRP or RO PUF have been increased by integrating Arbiter PUF with RO PUF.

c) In [24], circuit level techniques have been proposed to improve the reliability of RO PUF. This technique is based on using the transistors in forward body bias.

d) In [32], it has been shown that the quality of RO PUF improves when ROs are enabled for longer duration. Additionally if ROs are placed and compared in a chain like mode then the quality metrics improves. It is also shown that surrounding logic badly affects

the frequency of ROs. We have also discovered this phenomenon in our investigation of RO PUF.

e) In [65], composite PUF has been proposed. The main advantage is the higher number of CRPs.

g) Arbiter PUF

In Arbiter PUF two path are applied a low to high signal as shown in the Fig. 6. At the end of these paths there is an arbiter which decides the fastest path. A corresponding binary value is generated as an output by the arbiter. The challenge is the select line for multiplexers. Therefore if n multiplexers are employed then the total CRPs are 2^n. Arbiter is usually a D-flip flop. The two inputs for this flip flop are connected to the clock and data signal. As shown in Fig. 7, the output of an arbiter is based on the speed of input signals.

In [47], it has been shown that arbiter PUF has very low entropy when implemented on ASIC. Additionally it is prone to model building attacks [49].
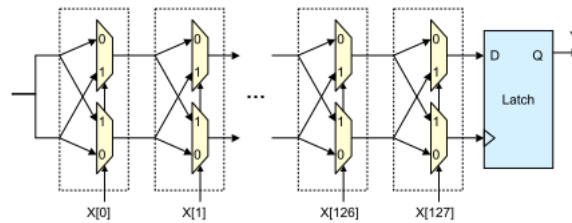
Figure 6: Arbiter PUF: Arbiter detects faster signal
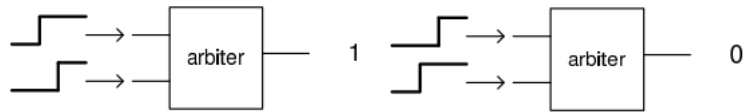
Figure 7: Arbiter output is based on the speed of two signals

h) Time-bounded PUF

It was introduced by Majzoobi et al., the time-bounded PUF relies on three flip-flops placed around the circuit under test (CUT): The Launch FF, the Sample FF, and the Capture FF. Initially, the flip-flops are set to zero. Then, the Launch FF is set to one on the rising edge of the clock. This signal propagates through the CUT and is sampled by the Sample FF on the falling edge of the clock. The CUT adds a challenge-dependent delay which may be greater than the half of the clock period. Hence, the sampled value depends on it and is xor-ed with the true launched value to be captured by the Capture FF.

i) Current based PUF

Proposed in 2011 by Majzoobi et al [40], it uses CMOS transistor leakage currents instead of delays and a sense amplifier instead of arbiter to implement a linear Strong PUF structure.

2.2.   Properties of PUF

For the evaluation of PUF, certain properties have been devised. These properties are based on statistical measures.

Table 1: Different PUF parameters

Maiti et al [58]	Uniformity Bit-aliasing Uniqueness Reliability
Hori et al [26]	Randomness Steadiness Correctness Diffuseness Uniqueness
Habib et al [73]	Worse case Uniqueness Worse case Reliability
Su et al [13]	Probability of Misidentification
Majoobi et al[15]	Single-bit Probability Conditional Probability
Yamamoto et al[44]	Variety

2.3. Target Applications

PUF structure is incorporated in the silicon devices for targeting two major applications. First one is the *identification* of silicon devices and another one is secure key generation for cryptographic functions. In the case of identification, the silicon devices go through the process of enrolment. In this step, challenge response pairs (CRP) are generated and then stored in the database. This step is carried out at room temperature and nominal voltage. Once these devices reach the users in the field, the device PUF response is again generated. It is like a fingerprint used for human identification. If this PUF response is equivalent to the response stored in the database, the devices are identical, otherwise they are different. Since we do not know the operational conditions in the field, therefore a PUF is said to be reliable if it can generate the same response even if the operating conditions, namely

voltage and temperature, are changed. Similarly, the response of PUF should be unique, so that the responses of any two devices are substantially different. Additionally the number of CRPs should be huge for this application. Another major application of PUF is the generation of *keys* for cryptographic functions. The minimal requirements for a secure key generation and storage are: A) a source of randomness to ensure that generated keys are unique and unpredictable, and B) Keys are secure and reproducible. A PUF-based key generator tries to take care of both requirements at the same time by using a PUF to harvest static but device-unique randomness, and by processing it into a cryptographic key. It also avoids the need for a protected non-volatile memory to store the key. Since we know that key must be reproducible, therefore error correction schemes are employed to regenerate the key reliably in the field.

2.4. Attacks

PUF prevents the attacker from revealing the underlying device specific secrets associated with FPGAs. There are several types of attacks that can be carried out against PUF circuits.

- Replay attack

If the adversary has physical access to PUF device and records the challenge Response pairs, then the adversary can fake the PUF by using the known CRPs. It can be thwarted by employing a PUF that has unlimited number of Challenge Response Pairs and using a new CRP every time. However, FPGAs have limited resources and it is impossible to build a circuit that has unlimited number of CRPs.

- Active attack

This attack can be invasive or non-Invasive. In case of Invasive attack an adversary tries to use Laser beams or ion beam to study the physical structure of PUF. It may permanently damage the PUF. Therefore the PUF secrets are not tempered. Till now, there has been no invasive attack reported in literature.

In case of non-Invasive attack the adversary tries to disrupt the challenge response behavior of PUF. For this purposes, the temperature or voltage are varied.

- Passive attack

The adversary uses the side-channel analysis to determine the relationship between the PUF circuit and the voltage variation or power consumption.

- Modeling attack

In modeling attack the adversary develops a mathematical model for the underlying PUF. This can be done by knowing the challenge response pairs of a PUF. Successful modeling attacks have been reported for Arbiter PUF [49]. It can be prevented by employing a PUF circuit that has true random outputs for corresponding inputs. It implies that there is no relationship between two challenge response pairs.

- Chosen Challenge Attack

In this attack an adversary can select the desired challenges and model the relationship between challenges and responses. To thwart this attack Controlled PUF [2, 25] has been introduced. In these designs hash functions are employed at the input

and output of PUF. Thus breaking the link between input and output of a PUF as shown in the Fig. 8.
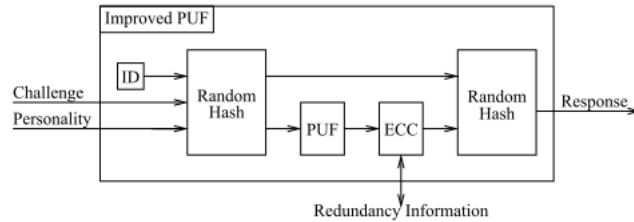
Figure 8: Controlled PUF [2]

- Cloning attack

In this attack an adversary will make the exact copy or clone of a PUF. It is assumed that the clone and the original circuit will have same Challenge Response Pairs (CRPs). Recently, Helfmeier et al. [52] demonstrated the first successful physical cloning of an SRAM PUF based on the fact that SRAM cells emit near infrared light when it is read and the cell's power-up value can be obtained from the emitted light. Hence, SRAM PUFs are not well suited as secure PUFs. Cloning other PUFs such as RO PUFs or SR-Latch PUFs have not been reported to this date. Therefore, both PUFs are still considered unclonable.

- Reverse Engineering attack

It is like a passive attack. In [60] Gate-level characterization (GLC) technique has been used to measure and extract the gate-level physical properties, such as threshold voltage and effective channel length.

2.5. Challenges

- Portability

When PUF design is implemented on different FPGA platforms, the results are not always similar. The robustness and efficiency metrics vary depending on the FPGA technology used. Currently there is no literature available to describe the effects of FPGA technology on the final PUF output. We observed this behavior when similar Ring Oscillator (RO) PUF was implemented on Spartan-3e (90 nm) devices and then on Spartan-6 (45 nm) devices. The uniqueness and reliability were found to be significantly different. However, care must be taken to ensure that this comparison is not affected by the external FPGA circuit like the noise present in the voltage regulator or the variations due to temperature. In addition, aging affects must also be considered during this evaluation.

In this research we implemented same type of PUF on different FPGA families. We presented our results to demonstrate the extent of portability.

- Aging effect

The PUF circuit behaves differently on the similar devices due to the aging effect. This phenomenon has been investigated in [36] for RO PUF. In Fig. 9, the affect of aging on RO frequency has been shown. This problem has been addressed in [59] for RO PUF.
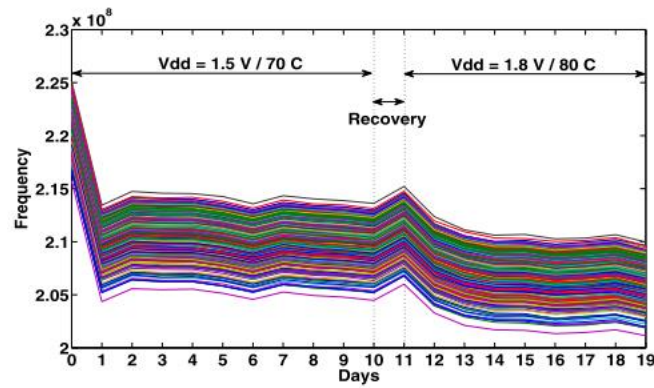
Figure 9: RO Frequency variation with aging under T+V stress [36]

However, in depth analysis and investigation needs to be carried out for other PUF designs especially the memory based PUF.

Weak points of different PUF designs

Apart from the Ageing and Portability, there are other pros and cons of PUFs. These are based on the individual type or subtype of PUF design. They are listed in the Table below.

Table 2: PUF advantages and disadvantages

PUF Type	Subcategory PUF Type	Disadvantages
Memory Based	SRAM	Cannot be used in Altera and Xilinx FPGAs, because start-up values are controlled.
	D flip flop	Cannot be used in Altera and Xilinx FPGAs, because start-up values are controlled.
	Butterfly	Not recommended for FPGAs

22

	Mecca	Design has not been reported to be implemented in practice yet, which presents a further risk in developing a solution
Delay Based	Arbiter	Prone to modeling attacks
	Ring oscillator	Expensive, since the number of rings required is large.
	S-ArbRO	Systematic variation can influence PUF output, if physically farther rings are compared

2.6. General Error Correction

The output of PUF in the field is affected by noise. For reliability purposes the noisy bits have to be corrected. For this purpose error correction schemes are employed. One of the well known schemes used for this purpose is called Fuzzy extraction. It was proposed by Dodis et al in [17]. More recently Kang et al proposed error correction schemes for PUF in [61, 69]. This scheme consists of two steps: Generation and Reproduction as shown in the Fig. 10.

In the generation, PUF bits are generated at the room temperature and nominal voltage. These bits are then encoded using BCH encoder. The encoded bits are stored in a database as helper data. In the reproduction step, the noisy PUF bits are generated in the field. The same helper data is used to correct the noisy bits. It is done using BCH decoder. In this process, some information is leaked to the adversary in the form of helper data.

A more efficient implementation scheme for PUF error correction has been proposed in [61]. It is shown in Fig. 11,
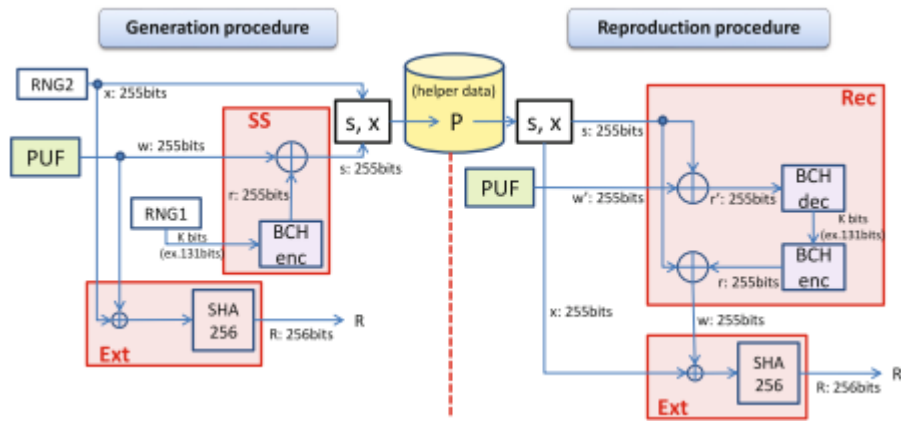
Figure 10: Implementation diagram for fuzzy extractor [69]
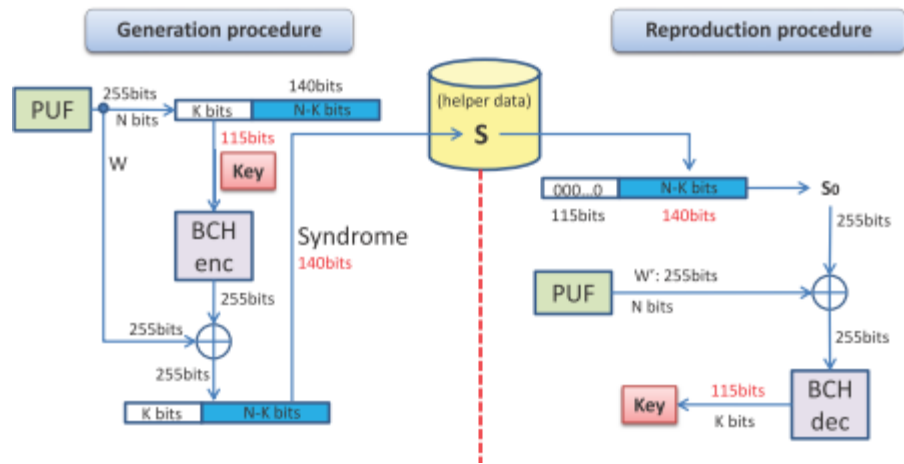
Figure 11: Efficient Implementation diagram for fuzzy extractor [61]

As shown in the above figure, hash function is removed. It results into considerable improvement in area and speed of error correction scheme.

3. Goals and Contributions

My research goal was primarily focused on three areas of PUF space. These areas are shown below in Fig. 12.

Robustness
Reliable
Unique
Stable
Uniform

Efficiency
Area
Power
Time

Portability
Spartan-3e
Spartan-6
Artix-7
Cyclone-V
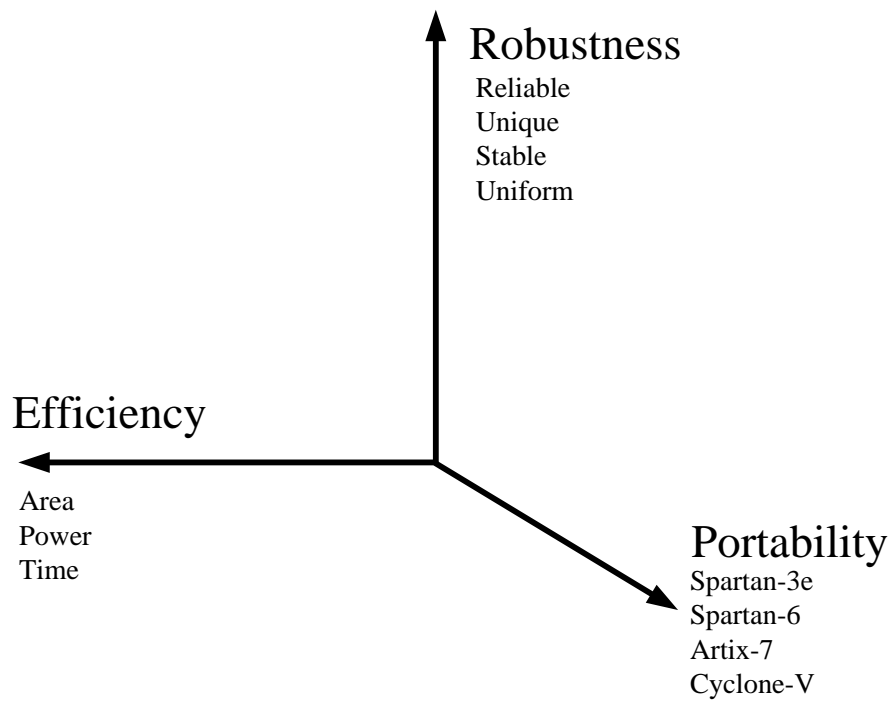
Figure 12: Dimensions of my proposed research

As evident from the above figure the three main areas of PUF space are: Robustness, Efficiency and Portability. My research goal was to develop a PUF that is Robust, Efficient and Portable at the same time. Robustness means that it should be able to generate a response that is unique, uniform and highly reliable. Portable means that it should be easily ported to FPGAs of different families and types. Efficiency means that our design should consume less power, area and time. We have to integrate all three dimensions while developing a PUF based system. Ignoring any one of them will seriously affect the quality of PUF. To comprehensively cover the PUF space we planned to develop two new types of PUFs. One is memory based and the other is delay based PUF. In the memory based PUF we planned to investigate an SR-Latch design. Similarly in the delay based PUF the goal was to develop a new RO-PUF design. Both designs are briefly explained in section 3.1 and 3.2 respectively. The dimensions of PUF space namely Robustness, Efficiency and Portability are covered in section 3.3 to 3.5.

3.1. Development of new SR-Latch PUF

The state of the art SR-Latch PUF [44, 53] is very expensive from area point of view. It requires 2 CLBs of an FPGA to generate a single bit of PUF response. It is prone to the affect of nearby logic. Thus the reliability of PUF bit is severely affected if the tool configures the same CLB with external logic. In our proposed design we solved this problem by utilizing all the logic resources of a latch CLBs. Thus latch is not affected by the nearby logic.

Our design approach is based on determining the length of metastable state. We trigger the latch into a metastable state. During the metastable state, oscillations are generated by the latch and the counter counts it. The duration of metastable state of a latch is based on the inherent manufacturing variation. Thus entropy harvested is based on the manufacturing variation. Strong latches are selected for bit generation, the remaining latches are discarded. A latch is regarded as a strong one, if it repeats for the same duration of oscillation during metastable state. Once strong latches are determined, then PUF response bits are generated by comparing the number of oscillations of strong latches.

To achieve area efficiency we configured more latches per CLB. With our design we utilized all the LUTs available inside a CLB, thus achieving a 100% LUTs efficiency. We compared our design with the state of the art [53]. In depth details of SR-Latch PUF have been covered in chapter 5.

3.2.   Development of new RO-PUF

We developed a novel FPGA friendly Ring Oscillator (RO) based Physical Unclonable Function (PUF). In this design the internal variations of FPGA Look-Up Tables (LUTs) are exploited to generate a PUF response. Statistical tests were performed to study the strength of this PUF. Moreover, stability is compared with the state of the art reported in literature to date. Our design has been tested on 31 Spartan-3e devices and the results are promising. Furthermore, we also analyzed the frequencies to extract the random variation offered by our design.

In our design each RO is made from a single AND gate and three inverters. It uses one LUT for an AND gate and three LUTs for inverters. We used one LUT-input to connect in a ring, while the remaining LUT-inputs are varied in order to generate programmable delays. We compare our design with the state of the art design [34]. In depth details of our RO-PUF design are covered in chapter 6.

3.3. Robustness

Robustness deals with the ability of the PUF to exhibit 'reliable', 'unique' and 'uniform' responses. In our experiments, we compared our results with the state of the art.

We developed a reliable and efficient SR-latch PUF in this work and compared the results to the state of the art implementation. A novel method of mode calculation is used to determine strong latches. The derived design has been verified on 25 Xilinx Spartan-6 FPGAs (XC6SLX16) and 10 Xilinx Zynq SoC (XC7Z010) devices. The uniqueness is close to the ideal value of 50%. We also did the entropy analysis. We calculated bit-dependent bias entropy bound based on PUF responses of 25 Spartan-6 FPGAs. We proposed two error correcting schemes for our PUF design. It was shown that the bit flips at extreme voltage and temperature were in the range of our proposed error correction schemes.

Additionally, we developed new matrices to analyze the quality of PUF. Software based Python scripts were developed to generate the PUF IDs using different schemes and then

analyze the result. This software tool can be used for any type of PUF. Furthermore, the tool has been made available for research community. It is freely available at [74].

3.4. Portability

Portability of PUF design means that different FPGA devices can be configured with the same design. These devices may belong to different FPGA vendors. Similarly devices may belong to same vendor but different families. Lastly devices may belong to the same family of a particular vendor. These three levels of portability can be explained with the following figure,
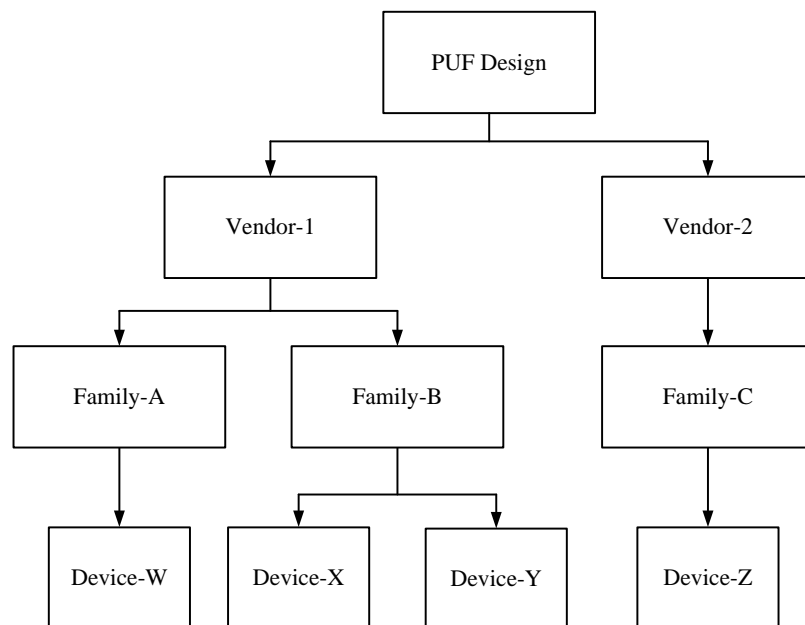
Figure 12: Types of Portability

As shown above, the three types of portability are,

1) Intra-Family portability

In this case devices belong to the same family of a particular vendor. As shown above XY shows this type of portability.

2) Inter-Family portability

In this case devices from same vendor are used. In the above figure WX shows this type of portability. These devices belong to two different families of Xilinx. We chose Spartan-6 and Zynq devices for this experiment.

3) Inter-Vendor portability

In this case devices belong to two different vendors. As shown above YZ shows this type of portability. We chose Zynq devices from Xilinx and Cyclone devices from Altera for this experiment.

The goal of this study will be to determine the challenges faced to achieve the three types of portability.

3.5. Efficiency

Efficiency deals with the area consumption of the design, the total power consumed by it and finally the time to calculate the PUF response.

Actual Power is equal to the power consumed by the FPGA device during the generation of PUF response bits. Area is calculated by the tool and this information can be easily extracted from the mapping report generated by tools. It is determined in terms of slice counts in FPGAs. Characterization time for RO PUF is calculated from the following equation,

The total characterization time = (# of components · enable_time)/ (board_frequency)

Where enable_time is the duration in which each component is allowed to run freely.

Changing the efficiency metrics severely affects the quality of PUF. For example, in order to save the PUF area, if two rings oscillators are implemented inside a single CLB (Configurable Logic Block). Then due to the proximity of a neighboring ring, the frequency of two rings might lock with each other. It means that two rings will oscillate with same frequency. Similarly, in order to generate the PUF response quicker, if the characterization time of a ring is reduced, then the frequency of rings is severely affected. It will result into a very unreliable PUF response bit.

In case of SR-latch PUF when external logic is allowed to configure the latch CLB then the number of counts are badly affected by the external logic. It implies that care need to be taken while reducing the area of a SR-latch PUF.

We developed SR-Latch PUF that is 2x smaller and is more reliable than the state of the art design. To conserve less power, we enable the PUF components only when the PUF-

ID is required. There are no free running, ROs or SR-Latches. Our design generates the PUF-ID in a reasonable time of ~5sec.

3.6.   List of Contribution

- Design of a novel memory based PUF for FPGAs.

    - SR-Latch PUF is 2x smaller in area than the state of art.

    - SR-Latch PUF is more reliable than the state of art.

    - SR-Latch uniqueness measure is comparable to the state of the art.

    - Validated on Spartan-6 and Zynq FPGAs.

- Design of a novel delay based PUF for FPGAs.

    - Due to the programmable nature of our RO-PUF, we can generate 2x more bits than the traditional RO-PUF. It implies our PUF requires less chip area to generate the same number of PUF response bits.

    - The uniqueness and uniformity measures of our RO-PUF responses are comparable to the ideal case.

    - The design has been validated on Spartan-3 FPGAs.

    - Frequency analysis of RO components.

    - Determination of systematic and manufacturing variations.

- Implementation and evaluation of PUF architectures on multiple FPGA and SoC platforms.

- Characterization of SR-Latch PUF over 10 Zynq devices and 25 Spartan-6 devices.

- Characterization of RO-PUF over 31 Spartan-3 devices.

- Comprehensive analysis of PUF response generation schemes.

- Development of post processing schemes and their software implementations for analysis and evaluation of various PUF designs.

- Evaluation of the PUF for 'Key generation' application.

- Selection and integration of the most suitable error correcting schemes.

- Development of a final product that generates the PUF response and does all calculations on-chip in real time.

4. Methodology

Our methodology is based on the following steps,

4.1.   Collection of Raw Results

The unprocessed data we collect from FPGA device is called a raw result. This result consists of frequencies of ring oscillators or counts of metastable latches. These components (latches and oscillators) constitute the PUF design. We implemented RO PUF on Spartan-3 devices. Similarly, SR-latch based PUFs have been implemented on Spartan-6 and Zynq [71] devices. We collected data from these devices at room temperature and at nominal voltage. For reliability purposes the data is also collected at varied temperature and voltage. The nominal voltage of Spartan-6 device is 1.2V and for Zynq devices, it is equal to 1V. We varied the voltage by ±5%. Similarly, the temperature was varied from -5ºC to 85ºC. We tested our devices at -5ºC, 0ºC, 20ºC, 45ºC, 65ºC and 85ºC. It was made sure that devices are heated or cooled down for at least one hour in the heating chamber before the data is collected from it. Furthermore, when results were analyzed, the corner cases were also tested to see the result of worse case operating conditions on the PUF quality. It must be noted that the Spartan-6 boards were tested, while they were new. This was done to prevent the negative effect of ageing on the PUF results.

In the next stage we plan to implement PUF designs on Artix-7 and collect raw PUF data. This data collection will be done at rated operational condition of temperature and voltage allowed by the manufacturer of these devices.

4.2. Converting Raw Results into PUF IDs

In this process, software scripts are employed to convert the raw data collected from the devices into a binary PUF response. The frequency of two components (latches or oscillators) is compared with each other and a response bit is generated. Different schemes can be used to select these two components. We describe these schemes below. It must be mentioned that python script is used for PUF response generation in each scheme.

4.3. Mathematical description of PUF properties

We briefly explained the PUF properties in section 2.2 . Here they are explained with mathematical equations and illustrations.

- Inter-chip Hamming Distance

It is equivalent to the hamming distance between the responses of any two FPGA devices at the room temperature and nominal voltage. It shows the difference between the responses of any two devices. It is defined as ,

$$\text{Inter} - \text{chip Hamming distance} \ = \ \frac{HD(R_i, R_j)}{L} \times 100\% \qquad (1)$$

Where L is the length of response bits. Ri and Rj are the responses of two FPGA devices. The following figure explains it,
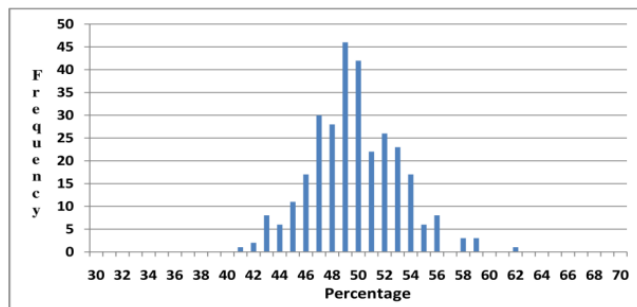
Figure 13: Normalized Inter-chip Hamming Distance

In the above figure, the x-axis shows the normalized Hamming Distance. While the y-axis (denoted frequency) shows the total number of times a given normalized inter-chip Hamming distance was obtained. In the above figure 25 devices have been used, the total number of combinations (i.e., the total number of board pairs {i,j}) is $\binom{25}{2}$ = 300. In ideal case, the normalized inter-chip HD should be 50% and will follow the binomial distribution. It means that 50% PUF output bits are different between PUF A and PUF B? If the PUF produces uniformly distributed independent random bits, the inter-chip variation should be 50% on average. Below in Fig. 14 it is shown how an actual PUF data compares with ideal case.
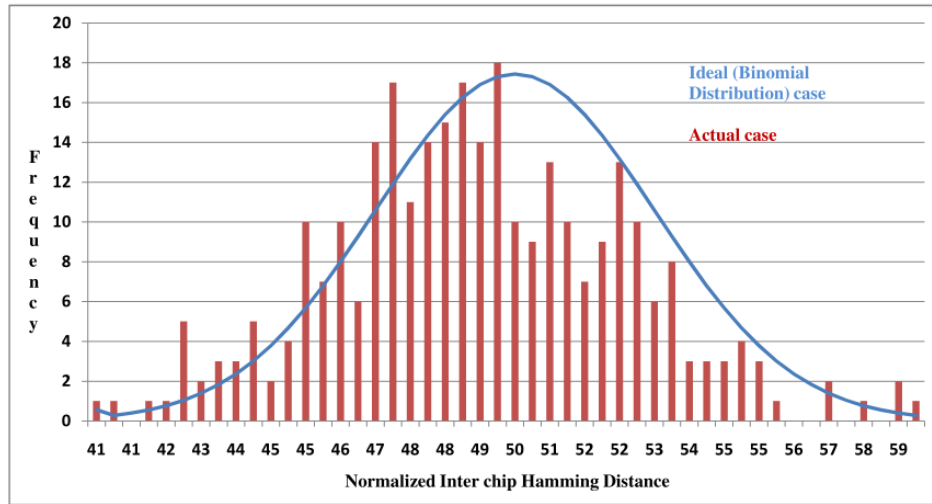
Figure 14: Normalized Inter chip HD (Average = 49%)

- Intra-chip Hamming Distance

The Normalized intra-chip Hamming distance is defined as $(HD\,(R_i, R'_{i,t})/L) \cdot 100\%$. Where R_i is the response of a device at room temperature and $R'_{i,t}$ is the response of the same device when either temperature or voltage is changed. Ideally the response should not change and the device output should be similar to what we get at a room temperature. However, in actual case, both the voltage and temperature affects the response.
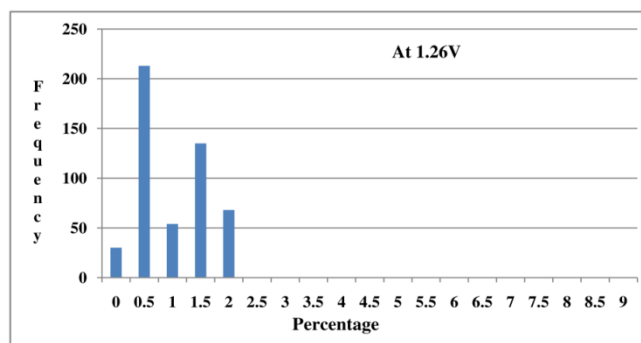
Figure 15: Normalized Intra-chip Hamming Distance

The histogram of normalized intra-chip Hamming Distance at 1.26V is shown above in Fig. 19. In this figure the x-axis shows the Hamming Distance in terms of percentage. While the y-axis (denoted frequency) shows the total number of times a given normalized intra-chip Hamming distance was obtained. In the above figure PUF response is generated 100 times for five boards at 25°C. Therefore, the total number of combinations is 500. In ideal case all 500 readings at 1.26V should have 0% HD. However, from the figure it is evident that a 2% change occurs in the HD at 1.26V. It must be mentioned that the nominal voltage for the devices used in this experiment was 1.2V.

In literature, intra-chip HD is sometimes referred to reliability.

- Uniqueness

Uniqueness represents the ability of a PUF to uniquely distinguish a particular chip among a set of devices of the same type. Hamming distance (HD) between a pair of PUF identifiers

is used to evaluate uniqueness. If two devices, i and j ($i \neq j$), have n-bit responses, Ri and Rj, respectively, for the same challenge, the average inter-chip HD among N devices is defined as,

$$\text{Uniqueness} = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{HD(R_i, R_j)}{L} \times 100\% \quad (2)$$

It is an estimate of the inter-chip variation in terms of the PUF responses and not the actual probability of the inter-chip process variation.

- Worse case Uniqueness

    It is equal to the minimum HD between any two chips.

$$\text{Worse case Uniqueness} = \text{Min}_{i=1,j=i+1}^{i=N-1,j=N} \left(\frac{\text{Min}(HD(Ri,Rj), L - HD(Ri,Rj))}{L} * 100\% \right) \quad (3)$$

    In ideal case, it should be 50%.

- Uniformity

Uniformity of a PUF estimates how uniform the proportion of '0's and '1's are in the PUF response.

$$\text{Uniformity (i)} = \frac{1}{L} \sum_{l=1}^{L} r_{i,l} \times 100\% \quad (4)$$

Where $r_{i,l}$ is the lth binary bit in the response of a chip i.

- Reliability

Reliability measures how accurately the PUF responses are reproduced under different operating conditions. These conditions include temperature, voltage or radiation. Intra-chip HD among several samples of PUF response bits are used to evaluate it. To estimate the

intra-chip HD, L-bit reference response (Ri) from the chip i at normal operating condition

(at room temperature using the normal supply voltage) is extracted. The same L-bit

response is extracted at a different operating condition

(different ambient temperature or different supply voltage) with a value R_i. T samples of

Ri are collected. For the chip i, the average intra-chip HD is estimated as follows:

$$\text{HD Intra (i)} = \frac{1}{T} \sum_{t=1}^{T} \frac{HD(R_i, R\prime_{i,t})}{L} \times 100\% \qquad (5)$$

where $R_{i,t}$ is the tth sample of a device i . HD_{INTRA} indicates the average number of noisy

PUF response bits. In other words, the reliability of a PUF can be defined as

$$\text{Reliability} = 100\% - \text{HD Intra (i)} \qquad (6)$$

Fig. 8 shows how the reliability of a PUF is evaluated using the time dimension of PUF

measurement.

- Worse Case Reliability

It is based on the maximum number of noisy bits under any set of conditions. Normally the

corner cases of temperature and voltage are evaluated to see the number of noisy bits at the

extreme level. We believe that for error correction purposes we need to consider the worse

case reliability. Worse case reliability for chip i can be calculate as,

$$\text{Worse case Reliability}(i) = \text{Min}_{i=1}^{N}\left(1 - \frac{\overset{C}{\underset{c=1}{\text{Max HD}(R_i, R_{i,c})}}}{L}\right) \times 100\% \quad (7)$$

Where C shows all possible conditions.

- Bit-aliasing

If bit-aliasing happens, different devices may produce nearly identical PUF responses, which is an undesirable effect. It gives us the information about the presence of systematic and spatial effect across devices. We estimate bit-aliasing of the lth bit in the PUF identifier as the percentage Hamming weight (HW) of the lth bit of the identifier across N devices:

$$\text{Bit aliasing }(l) = \frac{1}{N}\sum_{i=1}^{N} r_{i,l} \times 100\% \quad (8)$$

Where $r_{i,l}$ is the response of chip i at lth location.
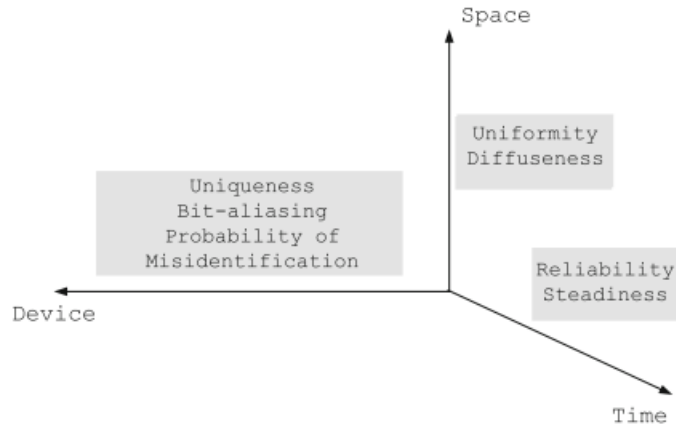
Figure 16: Parameters mapped on the PUF measurement dimension [58]

The properties of PUF are distributed along the device, space and time dimensions as shown above in the figure.

Notations used in the properties are:

Table 2: Notations used in equations

N = Number of FPGA devices
n = index of an ID in a chip ($1 \leq n \leq N$)
M = Number of ROs or Latches
L = Size of ID in bits
l = index of a response bit ($1 \leq l \leq L$)
R_i = Response of chip i, at normal operating
R_j = Response of chip j, at normal operating
HD= Hamming distance
T = total number of samples measured per ID
t = index of a sample ($1 \leq t \leq T$)
$R_{i,C}$ = Response of chip i, at condition C.

4.4.   Entropy

Actual entropy of a PUF is a function of complex physical processes. Therefore, it is close to impossible to calculate the actual entropy of a PUF response. Normally, only the estimated upper bounds on the underlying entropy can be calculated. These bounds can be calculated using at least the following two methods.

1) Based on the analysis of single bits

2) Based on the analysis of pairs of bits

Both these methods have been explained in [48]. The first method assumes that an adversary knows a bias for each position of a PUF response. In this method every bit position in a PUF response vector will have its own bias. An adversary knowing these individual bit-dependent biases can make a more accurate prediction by guessing individual bits in favor of these biases. This upper bound, called the bit-dependent bias entropy bound, can be calculated by using

$$H(Y^n) = \sum_{i=1}^{n} h(p_i) \tag{9}$$

In the above equation, h(pi) is the binary entropy function, it is calculated for n bit positions. The second method is based on the analysis of a pair of PUF response bits. This method assumes that an adversary knows pairwise joint distributions for pairs of consecutive bits i and i+1.This bound is tighter than the bound given by (9). It is called pairwise joint distribution entropy bound. It can be calculated using equations (10) and (11).

$$H(Y^n) = \sum_{i=1}^{n} h(p_i) - \sum_{i=1}^{n-1} I(Y_i, Y_{i+1}) \tag{10}$$

Where,

$$I((Y_1, Y_2) = \sum_{y1 \in y1} \sum_{y2 \in y2} p(y1, y2). \log_2 \frac{p(y1,y2)}{p(y1)p(y2)} \tag{11}$$

In equation (10), h(pi) is the binary entropy function, while I(Yi,Yi+1) is the mutual information between two random variables Yi and Yi+1. The mutual information between two random variables is a measure for the amount of information which is shared by both variables. We estimate the pairwise joint distributions of all possible pairs of the considered response bits, by counting the occurrences of each of the four possible pairs ('00', '01', '10', and '11') in the n bit response of all devices.

5. Efficient SR-PUF design

In this chapter we present a reliable and efficient SR-Latch based PUF design, with two times improvement in area over the state of the art, thus making it very attractive for low-area designs. This PUF is able to reliably generate a 128-bit cryptographic key. The PUF response is generated by quantifying the number of oscillations during the metastability state for preselected latches. The derived design has been verified on 25 Xilinx Spartan-6 FPGAs (XC6SLX16) and 10 Xilinx Zynq SoC (XC7Z010) devices. The design exhibited ~49% uniqueness figures when tested on both types of FPGAs. The reliability figures were > 94% for temperature variation (0-85°C) and ± 5% of core voltage variation. We also propose two error correcting schemes that assure that a key generated in the field is similar to the one generated under nominal conditions. This chapter is based on the work presented at [70, 75].

Contents

5.1. Introduction

In the last couple of decades there has been an exponential increase in the digital information processing and communication systems. With this phenomenal increase, security challenges are becoming significant. Recent research has led to an increased interest in security measures, especially in solutions that are physically unique and unclonable. In this regard, different structures of Physical Unclonable Functions (PUFs) have been developed and investigated to efficiently meet the requirements of these solutions. PUFs are physical primitives which produce unclonable and device-specific measurements of silicon Integrated Circuits (ICs). These measurements are then processed to generate either responses in challenge-response schemes or secure keys for cryptographic functions. Manufacturing process variations give physical uniqueness, but many physical unclonable functions (PUFs) are noisy and exhibit low circuit efficiency. Therefore, while designing new PUF structures, we need to focus on the efficiency besides the unclonability, uniqueness, and reliability, because expensive designs cannot fulfill the requirements of low-area applications.

PUF structure is incorporated in the silicon devices for targeting two major applications. First one is the identification of silicon devices and another one is secure key generation for cryptographic functions. In the case of identification, the silicon devices go through the

process of enrollment. In this step, challenge response pairs (CRP) are generated and then stored in the database. This step is carried out at room temperature and nominal voltage. Once these devices reach the users in the field, the device PUF response is again generated. It is like a fingerprint used for human identification. If this PUF response is equivalent to the response stored in the database, the devices are identical, otherwise they are different. Since we do not know the operational conditions in the field, therefore a PUF is said to be reliable if it can generate the same response even if the operating conditions, namely voltage and temperature, are changed. Similarly, the response of PUF should be unique, so that the responses of any two devices are substantially different. Another major application of PUF is the generation of keys for cryptographic functions. The minimal requirements for a secure key generation and storage are: A) a source of randomness to ensure that generated keys are unique and unpredictable, and B) Keys are secure and shielded from unauthorized access. A PUF-based key generator tries to take care of both requirements at the same time by using a PUF to harvest static but device-unique randomness, and by processing it into a cryptographic key. It also avoids the need for a protected non-volatile memory to store the key. Since we know that key must be reproducible, therefore error correction schemes are employed to regenerate the key reliably in the field. In this paper we are targeting the secure key generation application using SR-latch PUF.

Part of this paper has been published in [70], we extended that work by implementing the design on Zynq devices. We tested Zynq devices at ± 5% of core voltage and (0-85°C) temperature. In Section 5.2, we explain the motivation for our PUF design. In Section 5.3 we describe the related work for a better understanding of our study on PUF. Section 5.4

explains the design methodology. In Section 5.5, we explain the bit-string generation. Results and analysis are covered in Section 5.6. Implementation on Zynq based SoC devices is described in Section 5.7. Conclusions are given in Section 5.8.

5.2. Motivation

State of the art SR-latch PUF designs are very expensive in terms of chip area. These designs are not suitable for area constrained applications. We need to develop PUF designs that have small area footprint. Additionally these designs have to be reliable at different environmental conditions, i.e., the response of PUF at different voltage and temperature should match closely the response generated at the nominal voltage and room temperature. Furthermore, the PUF responses should be unique so that one PUF-ID can be easily differentiated from another ID. Apart from the reliability and uniqueness measure, the design should not be affected by the neighboring logic. These motivations lead us to propose a PUF design that is very efficient from the area consumption point of view and highly reliable at different voltages and temperatures. The effect of nearby logic inside the same CLB is negligible. The uniqueness measure is close to the ideal case.

5.3. Related Work

Since 2002, silicon based PUFs have been extensively investigated. There are two categories of silicon based PUF circuits: Delay based PUFs and Memory based PUFs. In delay based PUF, the propagation delay of a signal is used to generate a PUF response. Delay based PUFs include Arbiter PUF, Ring-Oscillator (RO) PUF and S-ArbRO PUF. In [30] and [39], the concept of programmable delay lines is presented, in which the LUT delays are used to create a metastable condition, which is further used to develop a PUF

49

and TRNG, respectively. Similarly, in [56], a PUF is developed using the programmable delay lines of LUT.

Another category of silicon PUF is based on memory. It includes SRAM PUF, Butterfly PUF, and Latch PUF. In [10], the first SRAM-PUF is presented, in which the start-up values of uninitialized Embedded RAMs are used as a PUF response. However, in the current state of the art Xilinx and Altera FPGAs, the start-up values of memory locations are controlled by the chip manufacturer, which renders SRAM PUF useless for FPGAs. In [16], Butterfly-PUF is presented. Latch PUF (LPUF) was introduced in [13]. In [35], the concept of transient effect ring oscillator (TERO) is presented. A true random number generator (TRNG) is developed by counting the oscillations of elements during metastability. The least significant bit of that count is selected as a random bit. The same approach is further developed, where an element with PUF capability is presented. In [54], PUF design is developed which is based on (TERO) cells. The randomness is harvested by measuring the metastability counts. Final bits are generated by averaging the metastability counts and then reading the most significant two (or four) bits of an eight bit counter for each latch. Furthermore, each latch count is measured 2^{18} times.

In [53], SR-latch based PUF is developed; It has been implemented on Spartan-3 and Spartan-6 devices. In Spartan-6 based design, 128 SR-latches have been implemented. Each latch is configured by using two neighboring CLBs in each column. Inside each CLB a Look up Table (LUT) and a Flip Flop (FF) are used. PUF response is determined by the final state of the latch. All the latches are excited by a 2.5MHz clock signal. For each latch two bits are contributed towards the PUF response. If the final state of the latch is logic '0'

in 1000 repetitions of the experiment, the corresponding response bits are '00'. If the final state is logic '1' in 1000 repetitions of the experiment, the corresponding response bits are '11'. If the final state changes at least once during 1000 repetitions of the experiment, this state is declared random, and encoded as '10'. A detection circuit determines all three cases and generates the response bits in each case. In our proposed design, we tried to improve the circuit efficiency of this design. In addition, our source of entropy is based on the exact number of oscillations at the output of an SR-latch during the metastable state, rather than a final state of each latch, as in [53]. Because of the encoding method, in [53] the goal is to increase the number of random latches, while in our work, we decrease the number of random latches. We include only the most *stable latches*, i.e., latches generating consistently the same number of oscillations in the metastable state, in the bit generation step during enrollment. On top of that, the method of [53] does not differentiate between the behavior of latches that generate no oscillations at the output from the behavior of latches that generate an even number of oscillations. This distinction is clearly made in our scheme. Furthermore, the method of [53] is prone to the influence of neighboring logic. We diminished this influence by prohibiting the tool from assigning any resources of the latch CLB to any external logic.

5.4. Design Methodology

In our design, an SR-latch is made from two LUTs configured as a NAND gate each. Additionally, two flip-flops are used in this latch to reduce the clock skew. Initially, a latch is forced into a metastable state by applying a rising-edge at a 'ctrl' signal, as shown in Fig. 17. During this state, the SR-latch oscillates. An eight-bit counter is used to count these

oscillations. Once the metastable state is over, the latch stops oscillating, and the counter value is stored into the block RAM (BRAM). Before applying the 'ctrl' signal, the two flip-flops (FF) are reset. It is done to ensure that latches always start oscillations with the same initial state.
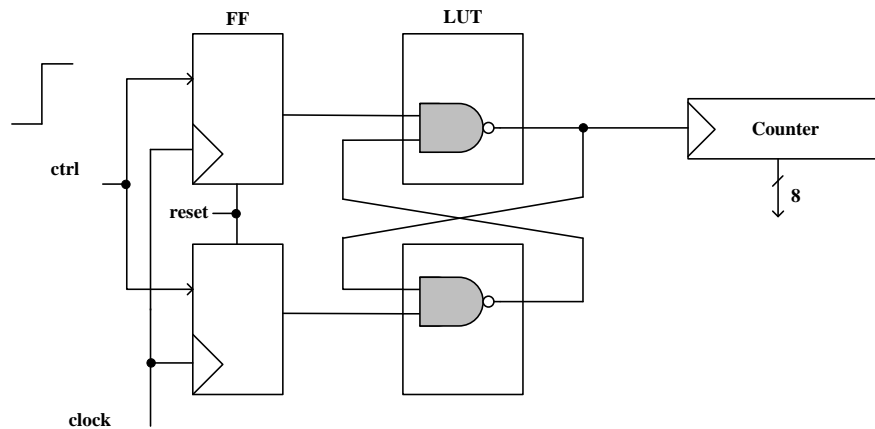
Figure 17: A single SR-Latch design

Once the process of the characterization is over for all the latches, the data from the BRAM is read-out to the PC via Enhanced Parallel Port (EPP) protocol as shown in Fig. 2. EPP protocol has been used for data retrieval because it has a very small area imprint. On the PC side, Digilent Port Communications (DPC) utilities are used, which are provided with Digilent Adept software [30]. In our design, a 9-bit multiplexer address line selects a

particular latch. This latch is then excited by applying a low to high transition at the 'ctrl' signal, as shown in Fig. 18. The eight-bit counter, available at the output of the multiplexer, counts the number of oscillations during the metastable state. These values are then stored in the neighboring BRAM. The bit generation and analysis are done during post-processing. It must be mentioned that only one latch is characterized at any given time. This is done to prevent any correlation between the neighboring latches and also to save the FPGA logic resources. Therefore, only one counter is used to measure the latch-counts during the metastable state. In addition, all the control signals are provided by the FSM. 512 latches are implemented, which requires 128 CLBs. The prototype design requires BRAM and EPP, because we want to analyze all the latch counts. However, in the final product, EPP can be replaced by a different interface and the size of the required BRAM will be reduced.
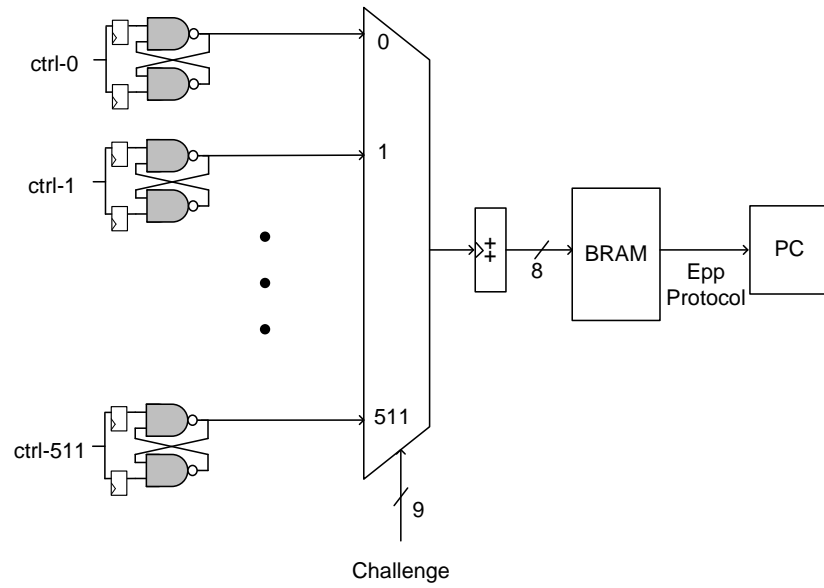
Figure 18: SR-latch PUF design

A. FPGA Layout

The placement of latches is constrained by the slice location attribute. All the latches are placed in a rectangular matrix of CLBs. The dimensions of this matrix are 16x8 as shown in Fig. 19.
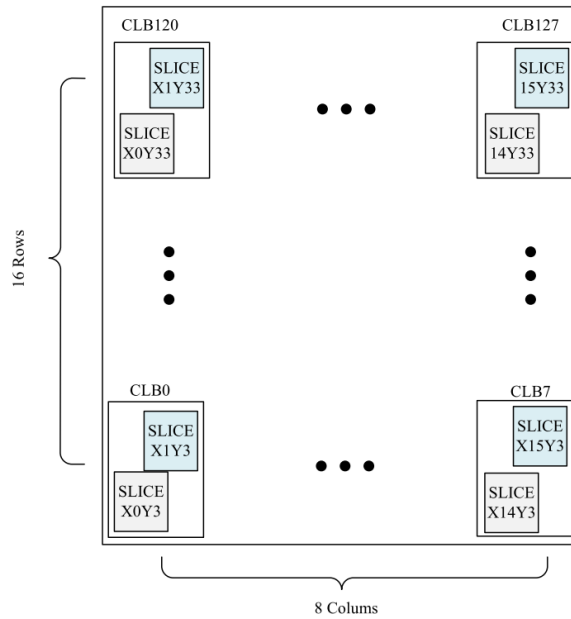
Figure 19: Layout on FPGA

We implemented our design on Spartan-6 (XC6SLX16) device. In Spartan-6 devices, each CLB has two slices. Inside each slice there are four 6-input LUTs. Four latches are implemented inside a single CLB in our design. These four latches (L1, L2, L3 and L4) are shown in Fig. 20, each with a different color scheme. This design is developed to eliminate the inter-CLB routing. We believe that due to the capacitance of long wires, the variation due to routing delays become significant, as explained in [27].
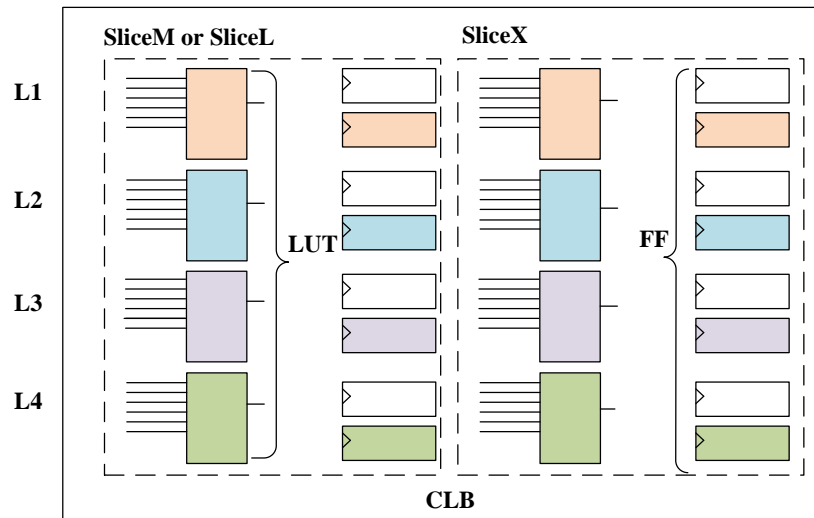
Figure 20: Our proposed design: Implementation of 4 SR-latches per CLB

As evident from the above figure, each latch consists of two LUTs and two FF. All the LUTs inside the CLB are utilized in the implementation of latches, thus, achieving a hardware efficiency of 100% in terms of LUTs. By comparison, the design proposed in [53], shown in Fig. 21, has 12.5% LUT utilization because, only two LUTs are utilized from the two adjacent CLBs. In addition, the circuit efficiency for FF is 50%, while it is only 6.25% in [53].

We also implemented the design proposed in [53], shown in Fig. 21. We added a counter to count the latch oscillations. We found that on the average 46% latches did not oscillate, i.e., the latch count remained zero.
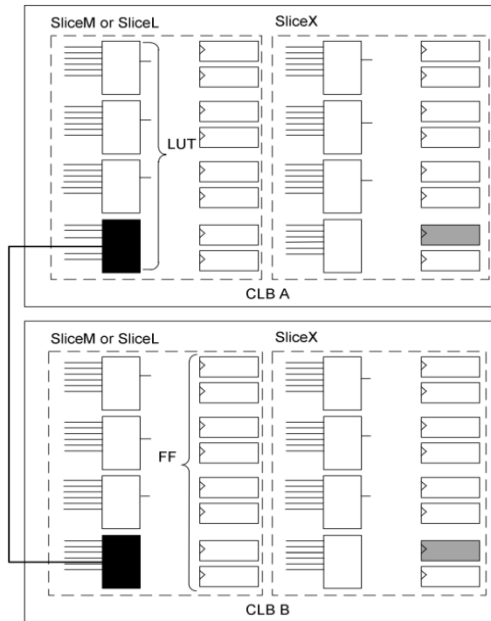
Figure 21: Design proposed in [53] for Spartan-6 FPGAs

We also implemented another design proposed in [53], shown in Fig. 22. In this case only Slice-X has been used. For consistency with [53], we implemented 128 latches. We define a latch to be 'random' if across the hundred samples the standard deviation is not zero. As evident from Fig. 23, the average number of latches outputting random number is 64 out of 128. In addition, the tool has used the remaining LUTs for extra logic.
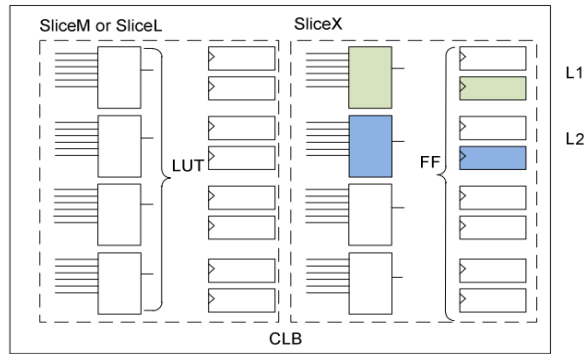
Figure 22: Single latch implemented per slice. 128 such latches were configured
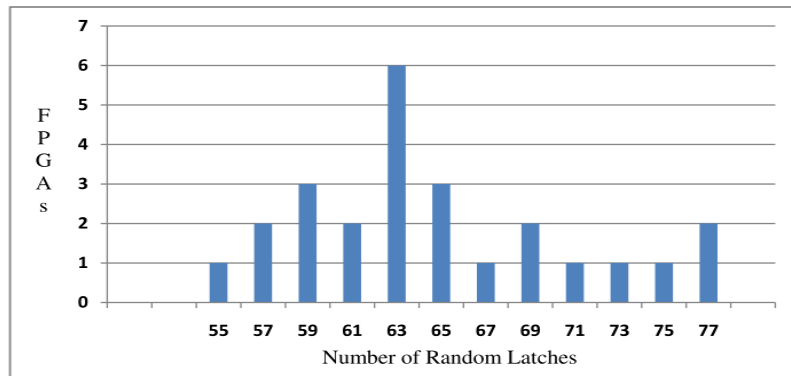
Figure 23: Histogram for the number of random latches when a single latch per slice is

implemented. This result corresponds to the design from Figure 22
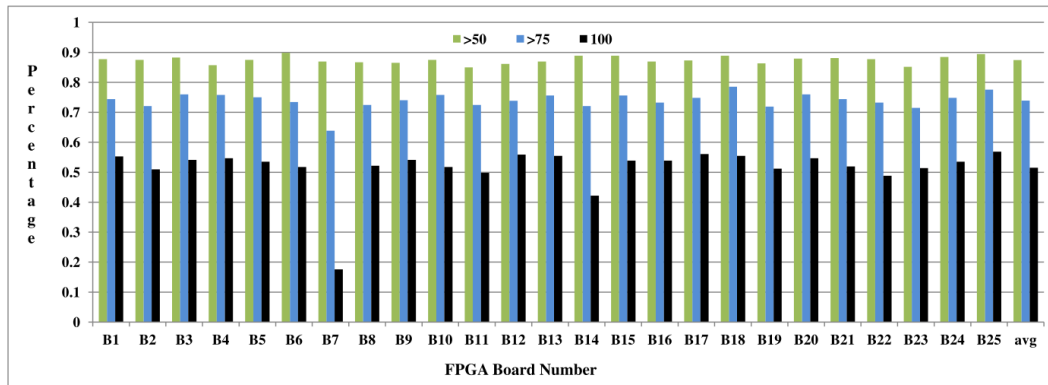
Figure 24: Proposed design: Percentage of stable latches per board at 1.2V and 25ºC.

This result corresponds to the design from Figure 20
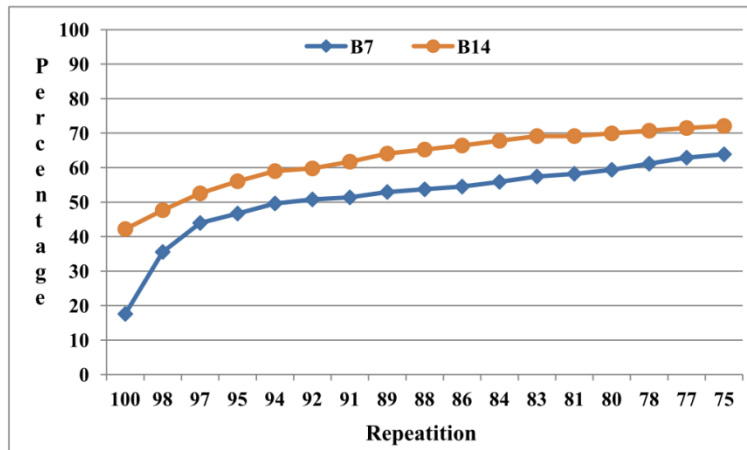
Figure 25: Stability of two boards
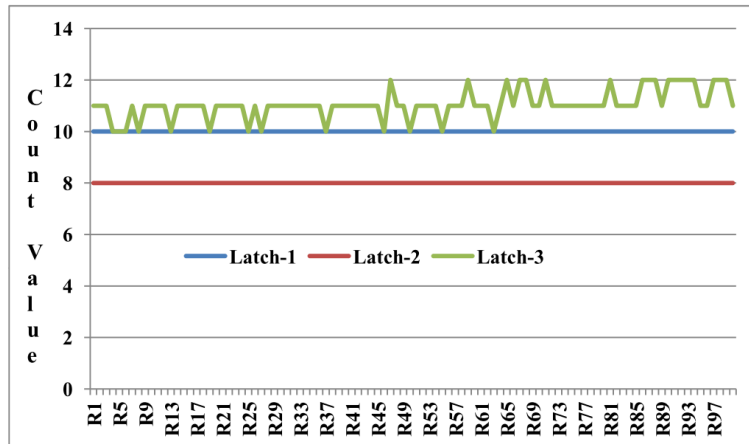
Figure 26: Count of three latches shown
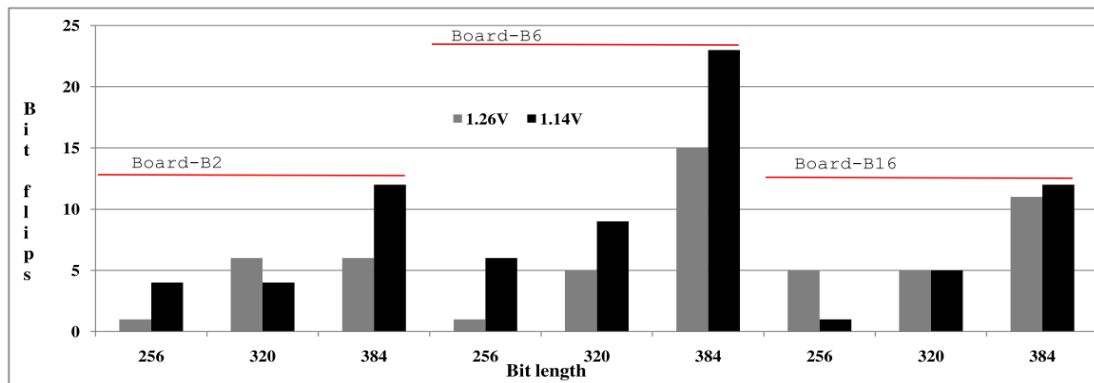
Figure 27: The number of bit flips vs. the bit length of PUF

5.5. Bit Generation

Our design consists of 512 SR-latches. Each latch is sampled 100 times and the corresponding latch count values are stored in the BRAM. Once all the latches are characterized, we select highly stable latches and ignore the remaining ones. In our method, a latch is defined to be stable if the latch count value remains the same for all 100 samples.

During enrollment, we store for each latch, a latch number, a corresponding bit showing whether this latch is stable, and the latch count value (i.e., we store: {Latch #, *stable*, count}). For bit generation during *enrollment*, only the stable latch count values are considered. In this method the count values for L+1 latches are used. These values are used to generate the L bit PUF response. We number stable latches in a snake-like fashion: L_1, L_2, L_3, ... L_{L+1}, and then we do the comparisons of neighboring latches [L_1, L_2], [L_2, L_3], [L_3, L_4], ...,[L_L,L_{L+1}]. A binary response bit '1' is generated if the count value of latch L_i is greater than the count value of latch L_{i+1};otherwise it is '0'. In the *field*, the count value of *stable* latches is recalculated by sampling each latch 100 times. We get 100 count values between 0 and 255, e.g.,{127, 127, 128, 127, 127, 127... 128, 127}. For further calculations, we use the number that appears the largest number of times, e.g., 127 in the example above. This number can be determined in hardware by storing the number of repetitions of each possible count value (between 0 and 255) in the memory location equal to the count value. A 256-byte memory is sufficient for these calculations.

This method of bit generation is flexible and can be generalized to obtain more PUF response bits per device. However, any attempt at generalization may affect the reliability of response bits. This dependence is explained in Figs 24 and 27.

61

In Fig. 24, the percentage of stable latches per device is shown. On the average, 87% of latches ($0.87 \cdot 512 = 445$) in all devices repeat the latch count at least 50 times out of 100 samples during enrollment (i.e., have stability mode "> 50"). This percentage drops to 51.4% when the latch counts are identical 100 times out of 100 samples (i.e., have stability mode "100"). As evident from Fig. 8, the FPGA boards B7, B14 and B22 have the smallest number of stable latches. In such cases, we consider as stable even latches that repeat the count value at least 90 times. Now the question arises, how many latches we should include in the bit generation process. It depends on the reliability of PUF response. In this work, we tested our boards at +5% of nominal voltage and -5 % of nominal voltage. It needs to be mentioned that on our Nexys-3 FPGA boards, the nominal voltage is equal to 1.2V, and it is supplied by LTC3633 regulator. We change this voltage by changing the resistor at the output of a voltage regulator, as recommended in [31].

In Fig. 25, the stability of two devices has been shown. These two devices are the least stable ones in our set. The horizontal axis shows the repetition of latches. The vertical axis shows the percentage of latches. For instance device B7 has 18% latches that repeat the latch count value 100 times. Therefore to select 256 strong latches from 512 available, we need all latches that repeat the count value at least 94 times out of 100. Similarly, in B14 we need to include all those latches that repeat at least 97 times out of 100. In Fig. 26, the count values of three latches are shown. Each latch is sampled 100 times. The horizontal axis shows the 100 runs and the vertical axis shows the respective count value for each run. From the figure it is clear that latch#1 and latch#2 are the stable ones, with count values

equal to 10 and 8 respectively. Latch#3 fluctuates between 10 , 11 and 12. Therefore we regard it as an unstable latch.

In Fig. 27, the horizontal axis shows the length of PUF response, while vertical axis shows the number of bits that change at the respective voltages. We generated 320 bits from 321 latches by treating as stable latches for which the stability mode was in the range from 75 to 100. The same approach was used in the case of 384 bit responses. This graph shows the results for three specific boards randomly selected from our set of 25 FPGA boards. From this figure, we deduce that the number of bit flips increases with the increase in the bit length of PUF response. This diagram also shows that the worst case is 23 bit flips for Board-B6 for the PUF response size of 384 bits. Even in that case the reliability is 94%, which is better than for the design from [53] in the same voltage range. However, we recommend the 256-bit version, which consists of highly stable latches (with the stability mode "100"), and results in reliable bits.

To prove that a latch is not biased in our design, we analyzed the count value of all the latches and found that 50.12% of them are odd, while the remaining ones are even. It proves that the symmetry of latch is not affecting the count value. We need to mention here that our final bit response from each pair of latches is dependent on the latch count values and not on the final state of any particular latch. Therefore, the bit response from a pair of latches will always be the same as long as the differences of the two count values do not change the sign.

5.6. Results

We compare our results with [53]. This comparison is summarized in Tables 3 and 4.

Table 3: Details of dataset for Spartan-6

	This work	[53]
No. of Chips	25	20
PUF per Chip	1	2
Samples (T)	100	100
ID size (L)	256	256
SR-Latches	512	128
FPGA family (Device used)	Spartan-6 (XC6SLX16-3CSG324)	Spartan-6 (XC6SLX16-2CSG324)

The uniqueness metric is used in [58]. We calculated uniqueness using the following equation:

$$\text{Uniqueness} = \frac{2}{N(N-1)} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \frac{HD(R_i, R_j)}{L} \times 100\% \qquad (12)$$

Ri = Response of chip i
Rj = Response of chip j
HD= Hamming distance
T = total number of samples measured
t = index of a sample ($1 \leq t \leq T$)

Similarly, reliability is calculated according to the following equations:

$$HD_{INTRA i} = \frac{1}{T} \sum_{t=1}^{T} \frac{HD(R_i, R'_{i,t})}{L} \times 100\% \tag{13}$$

$$Reliability_i = 100\% - HD_{INTRA i} \tag{14}$$

$$Average\ Reliability = \frac{1}{P} \sum_{j=1}^{P} Reliability_i \tag{15}$$

Reliability shown in Table 4 is the average reliability of seven random boards from a set of 25 boards. Average Reliability of P chips can be calculated using equation (15).

Table 4: Comparison of results with Yamamoto et al [53].

	This work	[53]	Ideal
Uniqueness	49.24%	49%	50%
Reliability @ 1.20V	**99.50%**	99.14%	100%
Reliability @ 1.14V	**97.54**%	94.70%	100%
Reliability @ 1.26V	**98.67**%	95.20%	100%

To carry out a more comprehensive comparison of reliability. The encoding method of [53] is used to generate 1024 bits using 512 latches in our design. We assume that the contribution of latches whose final value is '0' for 100 repetitions of the experiment are encoded as "00". Similarly latches whose final value is '1' and this value repeats 100 times are encoded "11". Finally, all the latches which have a variable final state in 100 repetitions

of the experiment are regarded as random latches and are encoded as "10". Then the average reliability at -5% voltage is 86.74%, similarly the average reliability at +5% voltage is 88.9%. We used the same seven boards as those used to obtain results reported in Table 4. From this experiment, it is clear that [53] generates four times more bits , however the voltage reliability drops by 8% at -5% voltage and by 7% at +5% voltage.

Uniqueness

In Fig. 28, the normalized inter-chip Hamming distance, $(HD(R_i, R_j)/L) \cdot 100\%$ is shown. The mean is 49.24%, while the standard deviation is 3.36%. This data is generated from 25 FPGA boards at 25°C and 1.2V supplied as the core voltage. The total number of combinations (i.e., the total number of board pairs $\{i, j\}$) is $\binom{25}{2} = 300$. The y-axis (denoted frequency) shows the total number of times a given normalized inter-chip Hamming distance was obtained.

Reliability

In Table 5, the information on bit flips for seven boards is shown. This set includes B7 and B14 as shown in Fig. 24. It is clear from this table that the worst case is 13 bit flips. The Normalized intra-chip Hamming distance is defined as $(HD(R_i, R'_{i,t})/L) \cdot 100\%$.
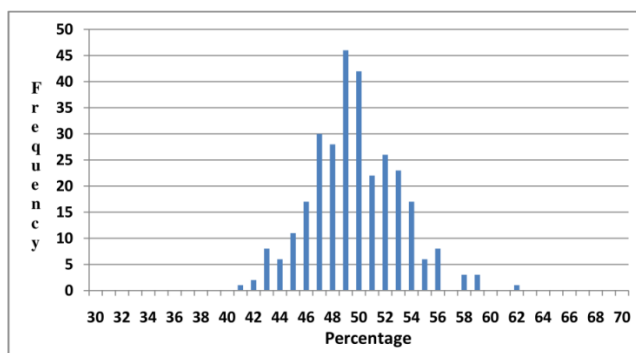
Figure 28: Normalized inter-chip Hamming distance for Spartan-6 (Mean=49.24%)

Table 5: Voltage vs. Intra-chip Hamming Distance for Spartan-6 devices.

Board No.	No. of stable latches at 1.2V	Bit flips at 1.26V	Bit flips at 1.14V	Maximum HD(R_i, $R'_{i,t}$)	Worst case HD
B2	260	1	**4**	4	
B4	280	1	**13**	**13**	
B6	264	1	**6**	6	
B7	261	**8**	2	8	**13**
B14	268	6	**11**	11	
B16	275	**5**	1	5	
B23	262	3	**9**	9	
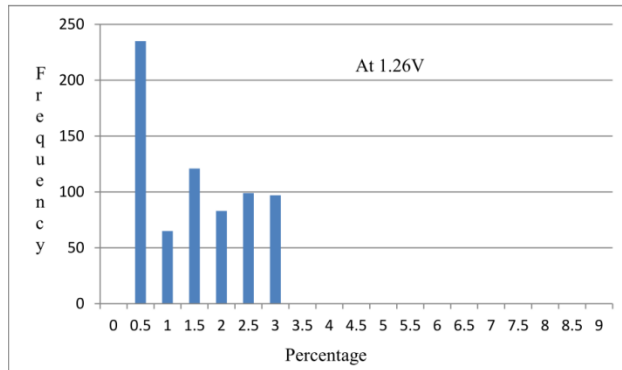
Figure 29: Normalized intra-chip Hamming Distance at 1.26V for Spartan-6 devices.

(Mean = 1.33%)
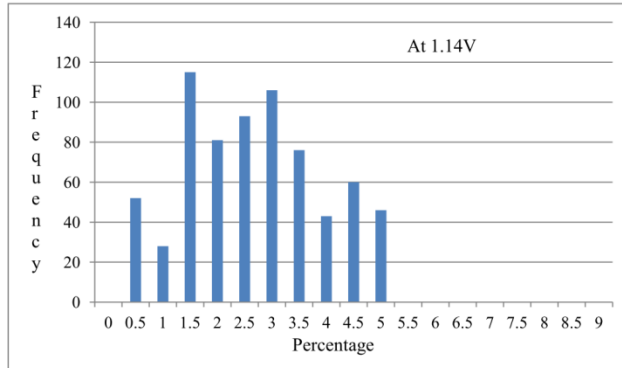
Figure 30: Normalized intra-chip Hamming Distance at 1.14V for Spartan-6 devices

(Mean = 2.46%)

Figs 29 and 30 show the histogram of normalized intra-chip Hamming Distance at 1.26V and 1.14V, respectively. In both cases, a PUF response is generated 100 times for seven boards at 25°C. Therefore, the total number of combinations is 700. From figures 29 and 30 it is evident that more bits flip occur at 1.14V than at 1.26V.

Temperature Resistance

We tested the boards at 0°C and 85°C. Table 6 shows the results for 10 boards. From Table 6, it is evident that for reliability the effect of 85°C is always worse than the effect of 0°C. Overall the worst case Hamming distance is 16. We also tested the five FPGA boards at the four corner cases of voltage and temperatures, as shown in Fig. 31. In each case 256 bits were generated.

Table 6: Temperature vs. Hamming Distance for Spartan-6 devices.

Board	Bit	Bit	Bit	Maximum	Worst
B1	256	4	**11**	11	
B2	256	7	**12**	12	
B3	256	2	**12**	12	
B4	256	0	**7**	7	
B5	256	3	**13**	13	**16**
B6	256	2	**7**	7	
B7	256	3	**10**	10	
B8	256	7	**16**	**16**	
B9	256	8	**8**	8	
B10	256	5	**8**	8	
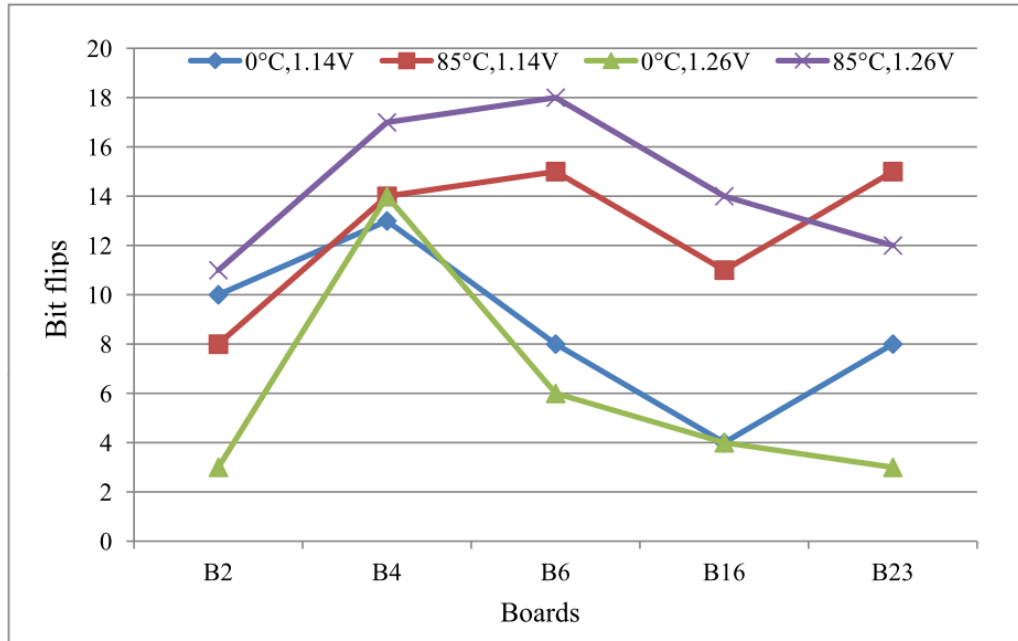
Figure 31: Bit flips at all corners for Spartan-6 devices

Reliability vs. Error Correction

For error correction, we propose two methods. Both methods are described below.

*Method-A***:** This method is inspired by the constructions described in [76] and practical designs described in [61]. It is based on the use of BCH code. It does the error correction by removing the noise from PUF response in the field. This method consists of two procedures: Generation and Reproduction. Generation is carried out at the room temperature and nominal voltage, while Reproduction is carried out in the field. During the

generation process, Secure Sketch (SS) is applied to PUF output w, as shown in Fig. 32. The second input to SS is the key K, generated using a True Random Number Generator, RNG1. The output of SS, denoted by s, is stored as helper data in the database. During the reproduction process, the helper data is used to regenerate the key K from a noisy PUF response w'. BCH decoder is used to regenerate the 131 bit key as shown in the figure. We propose to use BCH with the following parameters: (n=255, k=131, t=18) code. The meaning of these parameters is as follows: n=255 is the output block size, k=131 is the input block size (in our case, the size of the key to be encoded), and t=18 is the number of errors that can be corrected by this code. We chose these parameters because the code with these parameters can easily correct the worst case errors shown in Tables 3 and 4, and Fig. 31. Please note that in our scheme, the key K is not a function of the PUF response during the Generation process, but becomes a function of the PUF response during the Reproduction process in the field. This feature differentiates our scheme from two methods presented in [61]. It is important to note that this method is very suitable for low area designs.
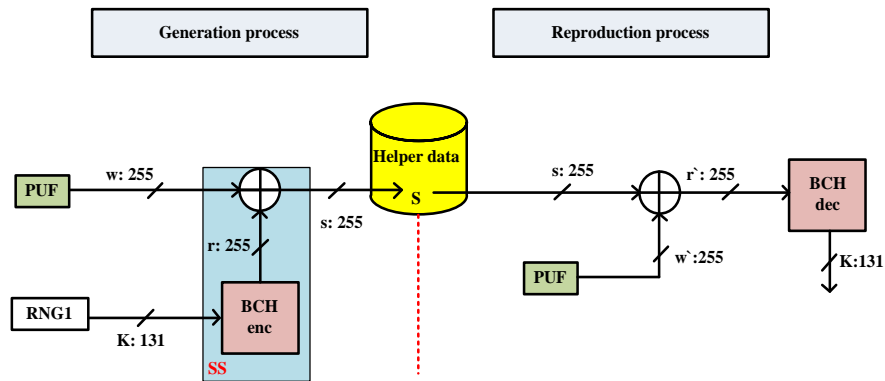
Figure 32: Error correction scheme A

Method-B: In Fig. 33, another method of applying error correction scheme is shown. This method has two Random Number Generators, RNG1 and RNG2. During the generation process, Secure Sketch (SS) is applied to PUF output W, as shown in the figure below. The second input to SS is K1, generated using RNG1. The output of SS, denoted by H1, is stored as helper data in the database. Similarly, the output of RNG2 is denoted by K2. Hash function (H) is applied to the output of K1 XOR K2. The output of this hash fuction is the key for our scheme. During the reproduction process, the helper data is used to regenerate the key from a noisy PUF response W'. BCH decoder is used to regenerate the 131-bit string K1 as shown in the figure below. It must be noted that the helper data consists of both H1 and H2. H2 is equal to the 131-bit string K2. We propose to use BCH with the following parameters: (n=255, k=131, t=18) code in Method-B as well. A nice property of this method is that if an intruder knows the key, he is not able to calculate either

W or K' corresponding to the new helper data (H1', H2'). We also want to add that the logic resources of programmable logic can be saved in area constrained designs by moving the error correction part to the processing system.
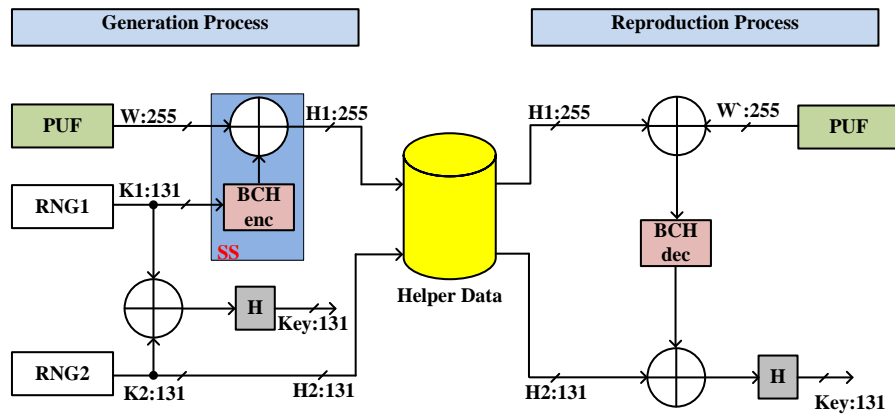
Figure 33: Error correction scheme B

Entropy Analysis

Actual entropy of a PUF is a function of complex physical processes. Therefore, it is close to impossible to calculate the actual entropy of a PUF response. Normally, only the estimated upper bounds on the underlying entropy can be calculated. These bounds can be calculated using at least the following two methods.

    1. Based on the analysis of single bits

2. Based on the analysis of pairs of bits

Both these methods have been explained in [48]. The first method assumes that an adversary knows a bias for each position of a PUF response. In this method every bit position in a PUF response vector will have its own bias. An adversary knowing these individual bit-dependent biases can make a more accurate prediction by guessing individual bits in favor of these biases. This upper bound, called the *bit-dependent bias entropy bound*, can be calculated by using

$$H(Y^n) = \sum_{i=1}^{n} h(p_i) \qquad (16)$$

In equation (16), $h(p_i)$ is the binary entropy function, it has been calculated for n = 256 bit positions. We calculated *bit-dependent bias entropy bound* based on PUF responses of 25 FPGAs. This entropy bound appeared to be equal to 0.959 or 95.9%. This result implies that the 256-bit response contains a maximum of 245 bits of entropy.

The second method is based on the analysis of a pair of PUF response bits. This method assumes that an adversary knows pair wise joint distributions for pairs of consecutive bits i and i+1.This bound is tighter than the bound given by (16). It is called pair wise *joint distribution entropy bound*. It can be calculated using equations (17) and (18).

$$H(Y^n) = \sum_{i=1}^{n} h(p_i) - \sum_{i=1}^{n-1} I(Y_i, Y_{i+1}) \qquad (17)$$

Where,

$$I((Y_1, Y_2) = \sum_{y1 \in y1} \sum_{y2 \in y2} p(y1, y2) \cdot log_2 \frac{p(y1,y2)}{p(y1)p(y2)} \qquad (18)$$

In equation (17), $h(p_i)$ is the binary entropy function, while $I(Y_i, Y_{i+1})$ is the mutual information between two random variables Y_i and Y_{i+1}. The mutual information between two random variables is a measure for the amount of information which is shared by both variables. We estimate the pairwise joint distributions of all possible pairs of the considered response bits, by counting the occurrences of each of the four possible pairs ('00', '01', '10', and '11') in the 256 bit response of all 25 devices. We found that for n = 256; the pairwise joint distribution entropy bound is equal to 0.866 or 86.6%. Therefore, a 256-bit response contains approximately 221 bits of pairwise entropy. Finally, we want to add that the entropy of the key is equal to $\rho \cdot n - (n-k) = 0.866 \cdot 255 - (255-131) = 96.83$ bits in our top-level key generation scheme with error-correction code, shown in Fig. 32.

Cost

It has been stated in [53] that the unused LUTs in each CLB can be used by the tool for other purposes. Therefore it is claimed that the circuit efficiency is still higher. However, we believe that neighboring logic, routed through the CLB, adversely affects the PUF response bits. To prove this claim we implemented ring-oscillators inside the latch-CLB for the design proposed in [53] and shown in Fig. 21. The result is listed in Table 7. We believe this change can become significant at different voltage. Therefore it is highly recommended to prohibit the external signals from being routed through the latch-CLBs.

Table 7: Effect of external signals on SR-latch for Spartan-6 devices.

	Without Ring	With Ring
Reliability @ 1.2V	99%	92%

In our design, the external signals cannot be routed through any CLBs designated to be used for SR-latches. We also prohibit the tool from using any resources inside the latch-CLBs for external logic. Therefore, the effect of external signal on the latch performance is eliminated. This goal is achieved by using *LOCK_PINS* attribute provided by Xilinx. By doing so we ensured that all interconnects leading to LUT inputs are fixed and cannot be arbitrarily changed by the routing tool. As shown in Fig. 17, the latch requires only two LUT input pins. Thus we connect the remaining four input pins of each LUT to logic '0', and lock them.

Table 8: Comparison with [53] for Spartan-6 devices.

	This work	[53]
Total latches configured	512	128
Total CLBs used for PUF	128	256
Response bit length	256	256
Latch/#CLB	4	0.5
Response bits/#CLB	2	1
Response bits entropy	221	167.9
Response bits entropy/#CLB	1.72	0.65

It must be mentioned that we implement our PUF only on even rows of CLBs. This is done to leave one set of rows for additional logic to be implemented by the tool. This logic includes multiplexers, decoders, registers, and a counter. We would like to emphasize that the latch in our design is compact and does not share a switch-box with any other external signal (each CLB is associated with a single switch-box). Based on this discussion, the design proposed in [53] for Spartan-6 FPGAs is two times more expensive than the one we are proposing in this work. Table 8 lists the FPGA resources used by both designs.

Characterization Time

Since we implemented 512 latches, we record 100 samples of data from each latch. The delay between any two samples is 10,000 clock cycles. This figure comes from the fact that some latches oscillate longer during metastable state. The on-board clock frequency is

100MHz. As a result, the frequency of the ctrl signal is set to 10kHz. For comparison, in [57], the ctrl signal's frequency is equal to 75kHz.

The total characterization time for each FPGA is equal to (512 x 100 x 10,000)/ (100MHz) = 5.12 sec.

Since each latch is sampled 100 times, therefore the total number of bytes saved into BRAM is equal to 512 x 100 = 51.2Kbytes. In the final product, the requirement for BRAM will be reduced. Since in the field, we will sample only the stable latches, therefore 257 bytes of BRAM will be used to store final latch count values. Each final count value will be calculated using 100 count samples. The algorithm performing these calculations requires maximum 256 bytes of memory, as explained in Section IV. This memory can be reused for all latches, independently of their number. Therefore, the total BRAM requirement becomes (256 + 257) = 513 bytes. The execution time of BCH algorithm is listed below,

	MicroBlaze Softcore (cycles)	ARM Hard-core Zynq (cycles)
BCH Encoding	608649	41340
BCH Decoding	3288283	84498

5.7. Implementation on Zynq SoC

For a wide range of applications, heterogeneous chips (such as Xilinx All Programmable Systems on Chip, Zynq-7000) bring superior performance and flexibility, while consuming a small fraction of the power required by traditional FPGAs. The FPGA fabric (Programmable Logic) of Zynq devices is equivalent to 28nm Artix technology. Spartan-6 FPGAs, are based on 45nm technology. Additionally, Zynq devices integrate dual-core ARM Cortex A9 processor in the Processing System (PS). Therefore it is very important to validate the strength of our proposed design on inherently different technologies and architectures. The CLB of Zynq device consists of two slices, that offer four 6-input LUTs and eight FFs each.

We implemented a PUF having 512 SR-latches on a Zynq (XC7Z010) device. For this purpose 128 CLBs are used in the Programmable Logic (PL) of Zynq device. Four latches per CLB are implemented in this design. For consistency of design, we selected those CLBs which have slice-L only. It must be mentioned that there are no slice-X in Zynq devices. For interfacing with the Processing System we used AXI-lite protocol as shown below in the Fig. 34.
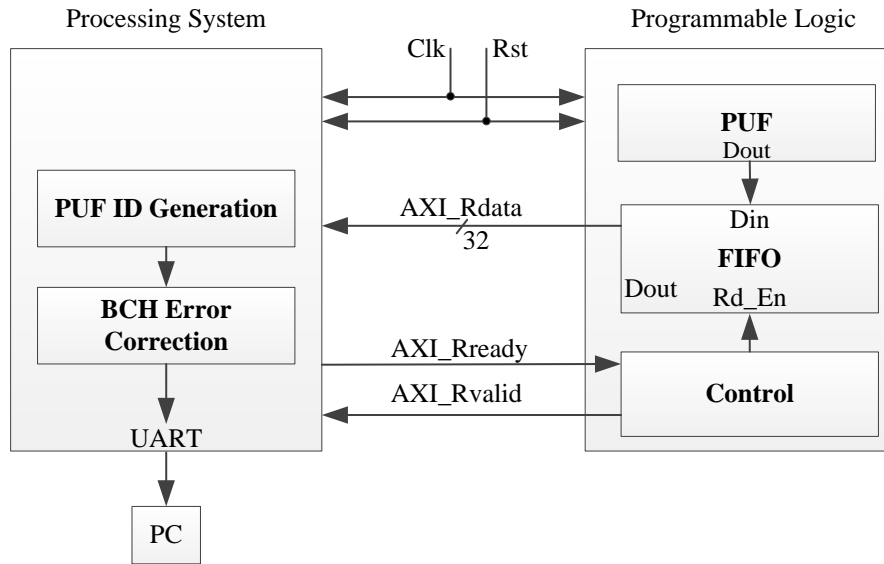
Figure 34: Interfacing PS of Zynq to PL using AXI-lite protocol implemented on Zynq-7010

The PUF data is processed at the PS side. PUF response bits are generated from the latch counts. The final response is then transmitted to the PC via UART. It must be mentioned that the recommended internal voltage of FPGA fabric in Zynq is 1V, while the minimum and maximum operating voltage is 0.95V and 1.05V respectively [77].

Fig. 35 shows the normalized inter-chip Hamming Distance with a mean of 49.87%. Similarly, Figs 36 and 37 show the histogram of normalized intra-chip Hamming Distance at 1.05V and 0.95V, respectively. In both cases, a PUF response is generated 100 times for five boards at 25°C. Therefore, the total number of combinations is 500. From figures 36

and 37 it is evident that on the average there are more bits flip at 0.95V than at 1.05V. On the average 5.32% bits flip at 0.95V, while 4.61% bits flip at 1.05V. Tables 10 shows the average reliability and Table 11 shows the worst case reliability for voltage variation.
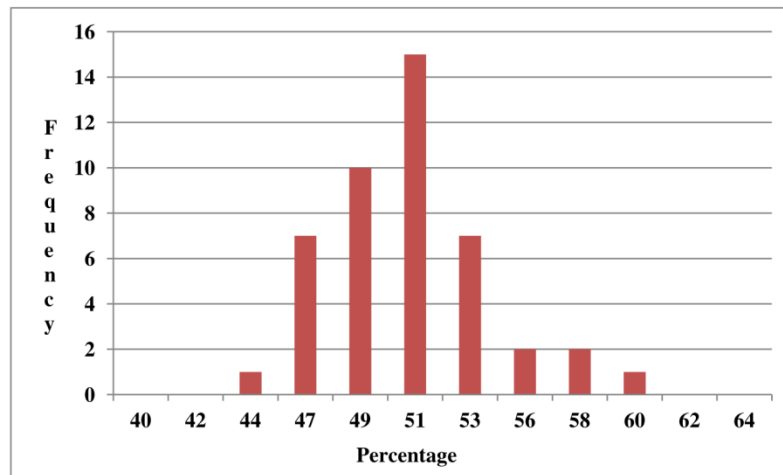
Figure 35: Normalized inter-chip Hamming distance for Zynq-7010 devices

(Mean=49.87%)

Table 9: Data set for Zynq-7010.

	This work
No. of Chips	10
PUF per Chip	1
Samples (T)	100
ID size (L)	256
SR-Latches	512
SoC Family (Device used)	Zynq (XC7Z010CLG400)

Table 10: Results of Zynq-7010.

	This	Ideal
Uniqueness	49.87%	50%
Reliability @ 1.0V	99.10%	100%
Reliability @ 0.95V	94.68%	100%
Reliability @ 1.05V	95.39%	100%

Table 11: Voltage vs. Intra-chip Hamming Distance for Zynq-7010 devices.

Board No.	No. of stable latches at 1.0V	Bit flips at 1.05V per 256 bits	Bit flips at 0.95V per 256 bits	Maximum HD(R_i, $R'_{i,t}$)	Worst case HD
B1	260	13	**18**	**18**	
B2	258	8	**12**	12	
B3	261	9	**15**	15	**18**
B4	262	**15**	8	15	
B5	258	14	**14**	14	
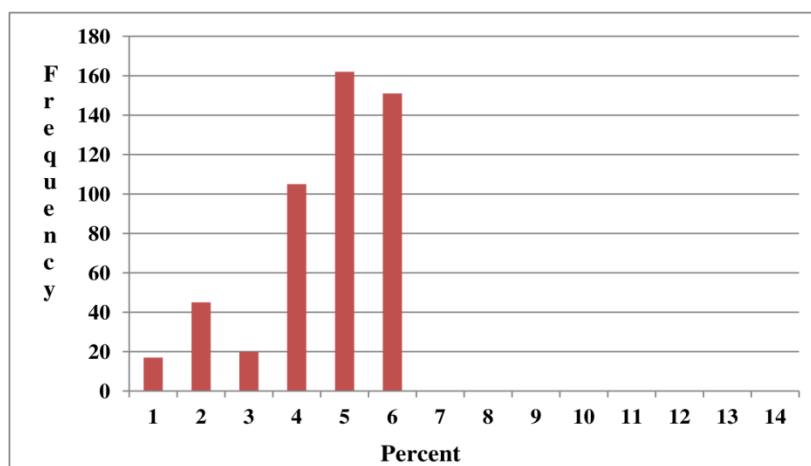
Figure 36: Normalized intra-chip Hamming Distance at 1.05V for Zynq-7010 (Mean

=4.61%)

83

We observerd that the number of counts of a latch during metastable state is affected by the operating voltage. Increasing the voltage of FPGA fabric by +5% increases the count value by 20.5% on the average. Similarly, reducing the voltage by 5% decreases the count value by 13.71% on the average for five boards. Fig. 38 shows this phenomenon. It must be added that only strong latches were considered in this experiment. Moreover, the temperature to conduct this experiment was fixed at 25°C.
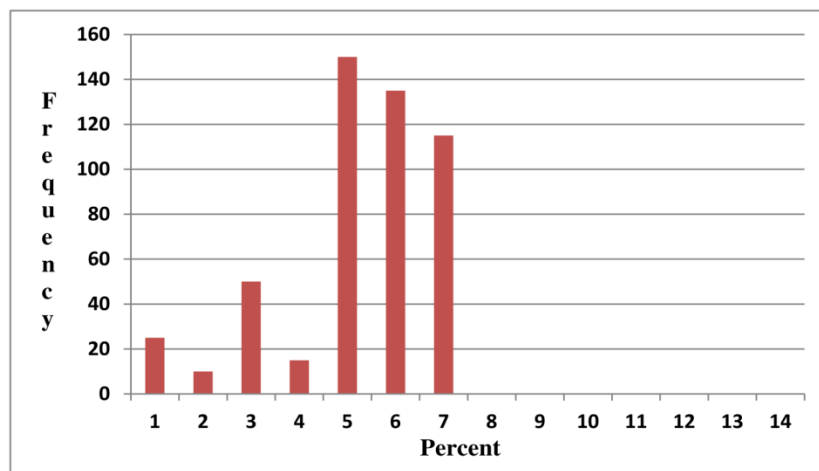
Figure 37: Normalized intra-chip Hamming Distance at 0.95V Zynq-7010 (Mean = 5.32%)
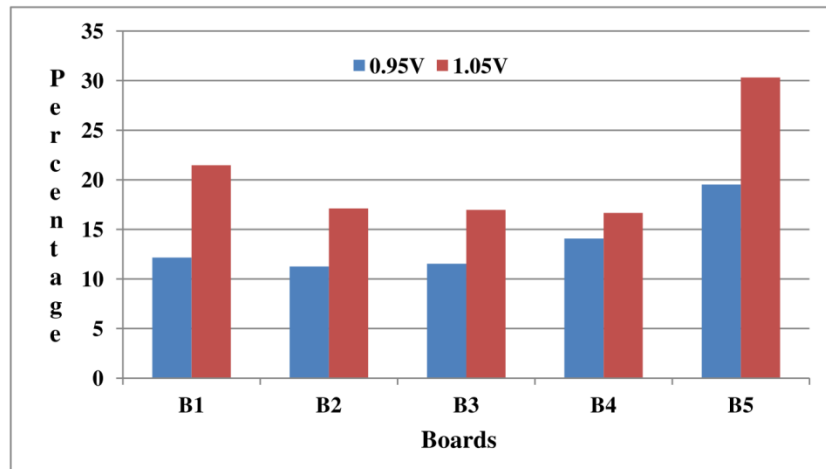
Figure 38: Absolute values of average relative changes in the metastability counts of

latches for the boundary voltage values of 0.95V (negative changes in count values)

and 1.05 (positive changes in count values)

Temperature Resistance

The operating temperature for the device ranges from 0°C to 85°C [77]. We tested the boards at these temperatures. Table 12 shows the results for 10 boards. From Table 12, it is evident that for reliability the effect of 85°C is worse than the effect of 0°C. Overall the worst case Hamming distance is 15 bits.

Table 12: Temperature vs. Hamming Distance for Zynq-7010.

Board	Bit	Bit	Bit	Maximum	Worst
B1	256	7	**8**	8	
B2	256	7	**12**	12	
B3	256	9	**10**	10	
B4	256	10	**11**	11	
B5	256	6	**13**	13	**15**
B6	256	10	**12**	12	
B7	256	7	**12**	12	
B8	256	12	**15**	**15**	
B9	256	10	**15**	15	
B10	256	8	**13**	13	

Comparison with [54] and [57]

Our implementation of SR-latch is inherently different from TERO based PUF designs. These differences range from the differences at the design of basic building block level to the final bit generation process. However, our design does the counting of metastable oscillations like [54]; therefore we deem it necessary to compare our design to the designs presented in these two papers. The design described in [57] consists of both PUF and TRNG, while the one described in [54] is a pure PUF design, like ours. Table 13 shows the implementation details of all three designs.

In Table 14, the inter-device variation (uniqueness) and intra-device variation (reliability) are shown for the three designs. This table also shows Reproducibility, which means, how many times the result is repeated for the same challenge at the room temperature and nominal voltage. It needs to be mentioned that in case of Reproducibility, [57] used eight different placements on FPGA fabric to measure results. The reliability for different voltages and temperatures is not reported by [54]. The nominal voltage (V_{nom}) for our FPGA is 1.2V, and 1.5V for the FPGA used in [57].

Table 13: Implementation details of PUF.

	This work	**[57]**	**[54]**
Design type	SR-Latch	TERO	TERO
Target FPGA family	Xilinx Spartan-6	Actel Fusion	Altera Cyclone-II
Devices used	25	10	9
Basic cells per device	512 Latches	216 TEROs	1172 TEROs
PUFs per chip	1	1	4
Size of basic building block	4 latches/CLB	8 LEs/TERO	16 LEs/TERO
Bits generated	256	216	252
Samples	100	128	128

In addition, we tested five boards at extreme voltage ($\pm 5\%$ of V_{nom}) and ten boards at extreme temperatures ($T_{min} = 0°C$, $T_{max} = 85°C$), while [57] tested two boards at extreme voltages ($V_{max}=+6.6\%$ of V_{nom}, $V_{min} = -13.3\%$ of V_{nom}) and extreme temperatures ($T_{min}=-13°C$, $T_{max}=100°C$). V_{core} is the voltage applied to the core of an FPGA device.

Table 14: Results of SR-Latch PUF.

	This work	[57]	[54]
Inter-device variation (%)	49.24	49.48	49.27
Reproducibility*	99.5	97.75	97.25
Reliability at $V_{core}=V_{min}$, T_R	94.92%	96.6%	N/A
Reliability at $V_{core}=V_{max}$, T_R	98.04%	97.0%	N/A
Reliability at T_{min}, $V_{core}=V_{nom}$	96.87%	95.7%	N/A
Reliability at T_{max}, $V_{core}=V_{nom}$	93.75%	97.8%	N/A

*At nominal voltage and room temperature.

The reliability shown in Table 14 is for the worst case scenario. Furthermore, room temperature is defined as $T_R=25°C$ in our case, while it is defined as $T_R=24°C$ in [57].

5.8. Conclusions

We presented a reliable and efficient SR-latch PUF in this work and compared the results to the state of the art implementation. The latch is designed to keep the effect of routing at minimum and extract the randomness at the same time. Strong and stable latches are selected in this method during enrolment. A novel method of mode calculation is used to determine strong latches. From the circuit efficiency point of view, the proposed design is two times more efficient than the state of the art. The derived design has been verified on 25 Xilinx Spartan-6 FPGAs (XC6SLX16) and 10 Xilinx Zynq SoC (XC7Z010) devices. The uniqueness is close to the ideal value of 50%. In case of Spartan-6 devices it is 49.24%. Similarly, the uniqueness measure for Zynq devices is 49.87%. The PUF responses exhibit

high resistance to temperature and voltage variations. For this purpose the design has been tested at ± 5% of core voltage and also over the commercial temperature range [0-85°C]. For Spartan-6 devices the reliability at +5% of nominal voltage is 98.67%, while at -5% of nominal voltage it is 97.54%. At both voltages it is more reliable than the start of the art. For Zynq devices the reliability is 94.68% at -5% of nominal voltage and 95.39% at +5% of nominal voltage. We also did the entropy analysis. We calculated *bit-dependent bias entropy bound* based on PUF responses of 25 Spartan-6 FPGAs. This entropy bound appeared to be equal to 0.959 or 95.9%. Similarly, the *pairwise joint distribution entropy bound* is equal to 0.866 or 86.6%. We proposed two error correcting schemes for our PUF design. It was shown that the bit flips at extreme voltage and temperature were in the range of our proposed error correction schemes.

6. PUF design using Programmable Delay Lines

In this chapter a novel design of an FPGA friendly Ring Oscillator (RO) based Physical Unclonable Function (PUF) is presented. In this design the internal variations of FPGA Look-Up Tables are exploited to generate a PUF response. Statistical tests were performed to study the strength of this PUF. Furthermore, we also analyzed the frequencies to extract the random variation offered by our design.

Contents

6.1. Introduction

With an increasing number of communication and computing devices, security challenges are becoming significant. An on-chip PUF (Physical Unclonable Function) can solve these challenges effectively and efficiently. A PUF is a chip-dependent unclonable challenge-response function that can be used to uniquely identify a specific integrated circuit. Furthermore, the PUF itself is tamper resistant against physically invasive attacks. Due to

these attributes, a PUF offers security against intellectual property (IP) theft and counterfeiting, and solves issues such as chip authentication, reverse engineering, trusted computing, and secure key generation. The idea of PUFs was first presented in [1]. Since then, the scientific community has profoundly investigated it. Silicon based PUFs use the idea of extracting the maximum variability of the chip manufacturing process. This variation is inherent and results in a unique signature for each chip similar to a biometric thumb impression of humans. Even for the same manufacturing process each chip carries a different signature due to process variations. A strong PUF is classified to be the one that extracts the maximum process variation and is reliable to exhibit this variation under different conditions.

There are strong reasons to design PUFs for FPGAs. Cryptographic functions are implemented using FPGAs for faster execution compared to software execution, and PUFs can provide random keys for these functions. Furthermore, FPGAs are reconfigurable in nature and IP protection during configuration of an FPGA is an important issue from a security point of view.

In Section 6.2, we describe the previous work for a better understanding of our study on PUF. We present and discuss implementation details, as well as highlight results in Section 6.3. In 6.4 we highlight the bit-string generation and in 6.5 we do the frequency analysis. The conclusion and future study are described in Sections 6.6.

6.2. Previous Work

Since 2002, silicon based PUFs have been extensively investigated. The initial proposal of a delay based arbiter PUF was made in [2]. Arbiter PUF was further explored in [3] and [4] to investigate reliability and security features. Although the Arbiter-PUF offers strong PUF properties, it is prone to machine-learning attacks [49]. In [11], the idea of a Ring-Oscillator (RO) based PUF is presented. In this PUF the challenge is the selection of a pair of ROs. The response is the one-bit comparison result of the frequencies of those ROs. RO PUF has some weaknesses associated with it, like dependence on supply voltage and temperature. In addition, special care must be taken to reduce the effect of systematic variations [34]. More recently S-ArbRO PUF, with the increased number of challenge response pairs (CRP) was presented [51].

A large-scale characterization of RO based PUF has been done in [23]. In [16], Butterfly-PUF is presented, which requires symmetric paths between registers for causing metastability. FPGA tools do not offer complete access to symmetric design at the wire level, therefore, routing schemes make it hard to achieve symmetric butterfly design on FPGAs. This fact has been verified by [27] for both Arbiter and Butterfly PUF. In [30] and [39], the concept of programmable delay lines is presented, in which the LUT delays are used to create a metastable condition which is further used to develop a PUF and TRNG respectively. Our approach is to employ the concept of programmable delay lines in ring oscillators based PUF for improving the number of independent response bits. We did not create metastability for randomness. In [34], Maiti et al. presented an RO PUF, in which multiplexers were introduced in the ring to select different paths inside the ring. In this

93

design, only two rings with identical paths were compared at a time because the authors wanted to determine a configuration which resulted in the maximum frequency difference between two ROs. However, in our case, the path outside of the LUTs stays constant. Therefore, we minimize the impact of routing or wire delays on the frequency of ring oscillator. Hence, the randomness is purely due to internal LUT variation. This delay is the significant and deciding factor in the comparison of two ring oscillators. In [37], the number of configurations of ring oscillator has been improved by introducing a latch in the path of a ring, making it impossible to compare a latch-path with no-latch-path. Our approach is completely different because we do not introduce anything in the path of a ring. We extract only the randomness inherent in LUTs, where we have the luxury of comparing any configuration of a LUT with any other configuration i.e., any configuration from '000' to '111' with any other configuration from '000' to '111'. All configurations are applied to LUT input lines. We will show in the next section that the programmable delay offers to generate random and independent bits with a very strong capability to repeat them.

This PUF is designed for Spartan xc3s100e devices, where each CLB has four slices, each with two 4-input LUTs. Our PUF design is based on a RO, which has one AND gate and three inverters. This design uses one LUT for an AND gate and three LUTs for inverters. We used one LUT-input to connect in a ring, while the remaining LUT-inputs are varied in order to generate programmable delays, as shown in Fig. 39.

In Fig. 39, three inverter LUTs in a ring are connected to three delay lines each, while the remaining LUT is tied to two lines. Because we wanted to analyze the maximum effect of

delay lines on the ring oscillator frequency, therefore we connected all free LUT inputs to our programmable delay controller. All the LUT inputs are locked by using a LOCK_PINS attribute provided by Xilinx tools. By doing so we ensured that all interconnects leading to LUT inputs are fixed and cannot be arbitrarily changed by routing tools. Additionally, we ensure that all the rings are identical and are subject only to CLB internal routing delays. In order to achieve this, we defined each ring oscillator as a macro and placed them at selected locations. The output of this ring oscillator is fed into a 32-bit counter, which determines the frequency of this oscillator. It is important to mention that the response from our design is solely dependent on the internal variation of FPGA LUTs, and variations due to routing delays are minimized. In particular, our method eliminates any differences in routing paths (and thus routing delays) caused by the tools. Figure 40 shows the placement of 130 ROs that make up our PUF. Each ring oscillator is constrained in a single CLB. In this design, 130 rings are configured at the center of a chip in a 13x10 matrix, The chip under test does not allow us to place 130 rings in a square format placed at the center of a chip.
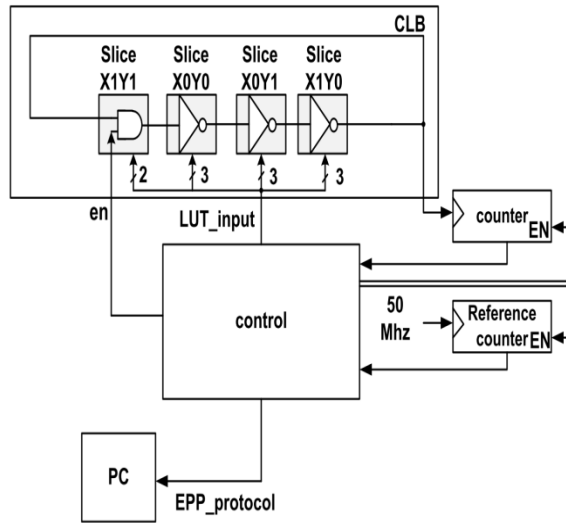
Figure 39: Single Ring-Oscillator with programmable delay lines

In Figure 40, rings start from the bottom left (R0) corner and the last one is shown at the top-right corner numbered R129. We selected 130 as a number of rings, because the FPGA devices available to us had 240 CLBs in total. We cannot use all CLBs for rings, because we need some logic resources to use for control purposes as shown in the Table 15.We could have decreased the slice count by forcing two rings per CLB (as each CLB slice contains two neighboring LUTs), but we intentionally rejected that approach, because the two rings might lock with each other, and hence their frequency affect each other. This phenomenon is also reported in [33].

For data retrieval we used Enhanced Parallel Port (EPP protocol), which has a very small area imprint. On the PC side, Digilent Port Communications (DPC) utilities were used, which are provided with Digilent Adept software.

Table 15. Area requirements of our design

	Number of slices occupied	Percentage
Slices for rings	4·130 =520	54.2%
Slices for other logic	27	23.6%
Total slices	47	77.8%
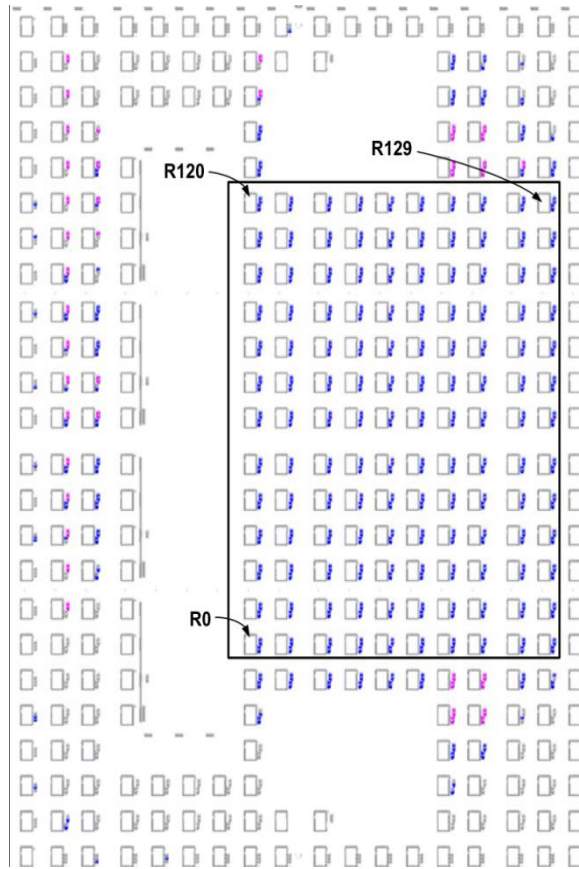
Figure 40: PUF array configuration of 130 RO on Spartan xc3s100e device

6.3. Implementation Details

LUT_input bits can be varied from '000' to '111', resulting in eight different frequencies

of a ring oscillator. Additionally, these frequencies are highly repeatable for any particular

ring as shown in Fig. 41. From Fig. 41, it is evident that the frequency varies significantly

depending on the LUT_input sequence. In [30] and [39], it is stated that maximum

frequency is achieved with '000' and minimum with '111' sequence. However, based on our experiments, this is not always the case.
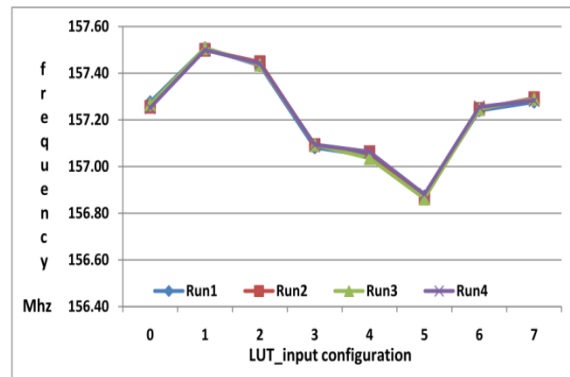
Figure 41: Frequency distribution due to LUT_input bits variation

The pattern presented in Fig. 41 changes completely when we select another ring oscillator. Even a neighboring CLB exhibits a different pattern. One reason might be that we use Spartan-3e devices which are based on 90nm technology while in [30, 39], Virtex-5 devices are used which are 65nm technology. We also observed that the standard deviation among 20 samples never exceeds 0.018 % of ring oscillator frequency. By using longer characterization time the noise further decreases. It is important to mention here that we did not test repeatability under different voltage and temperature conditions.

We tested our PUF for different characterization times; which is the time required to allow the ring to oscillate freely. Each RO should be enabled for enough time, such that the delay pattern associated with each LUT input value is sufficiently repeatable as shown in Fig. 42. Preliminary investigation revealed that a small characterization time causes huge differences in the pattern. In Fig. 43, the characterization time is reduced from 1sec to 1msec and standard deviation increases among four runs from 0.018% to 0.15%.
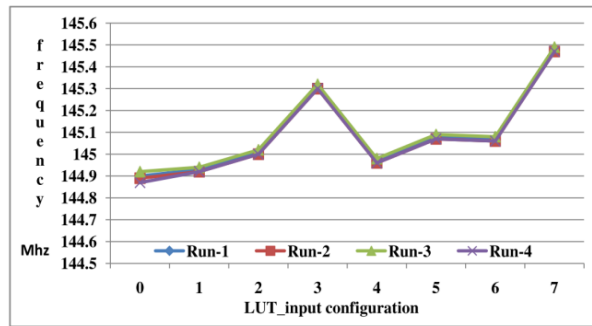
Figure 42: Characterization time equal to 1 sec
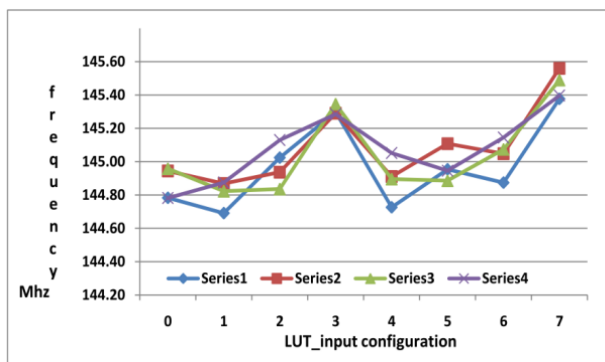
Figure 43: Characterization time equal to 1 msec

We used a characterization time of 1 sec when we extracted data from PUF, implemented on 31 Digilent boards. We did the frequency analysis of each ring and devised two schemes to analyze the improvement in the number of response bits.

Scheme # P1

In this scheme, we compared eight frequencies resulting from eight different values (000 to 111) applied to LUT input of one ring oscillator with eight frequencies of the neighboring oscillator i.e., only ROs under the same configurations are compared. In post-processing, if $fr0 > fr1$, the response is '1' otherwise it is '0'. To remove systematic variation (an unwanted correlated variation due to spatial location of a ring-oscillator on a chip), under each configuration, only the comparisons shown in Fig. 44 are made. Therefore, the PUF will output $8 \cdot (r - 1)$ bits response for each FPGA, and chip ID is extracted from these 1032 bits.
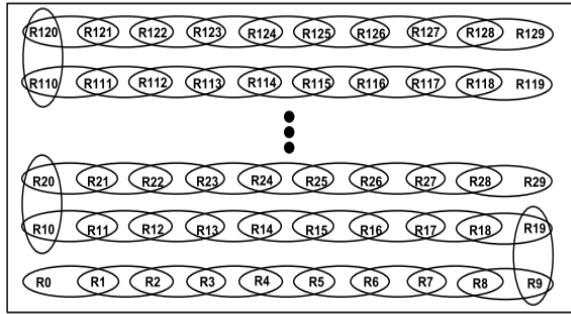
Figure 44: Comparison of rings along the rows

Scheme # P2

In this scheme we compared the frequencies along the CLB columns as shown in Fig. 45, where each circle shows a single comparison, otherwise it is similar to scheme P1. From this scheme, we were able to extract the same 1032 bit response. However, the inter-chip variation decreases to 95.34% from 96.6% as shown in Table 17. One reason is that the first row of the devices (R0-R9) were having the smallest frequencies and resulted in a similar response when compared to the frequencies of the 2nd row (R10-R19). This behavior reduced the HD and subsequently the Inter-chip variation.
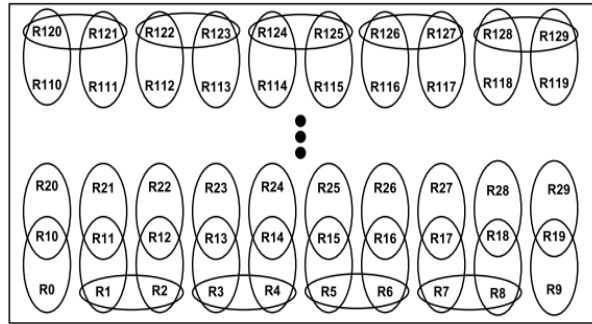
Figure 45: The comparison of rings along the columns

6.4. Bit-string generation

For 130 ROs, each ring pair is able to generate 8 bits due to 8 LUT configurations, which yields bitstring of length $129 \cdot 8 = 1032$ bits. In this work we only compare the neighboring rings to cancel out the effect of systematic variation. During comparison, if the frequencies of two rings cross, then we record 8 bits, otherwise a single bit is contributed toward the PUF ID. Therefore, during enrollment, we store for each ring number a corresponding crossover bit (i.e., we store pairs: (Ring #, c)). In Fig. 46, a meaning of the crossover is demonstrated.
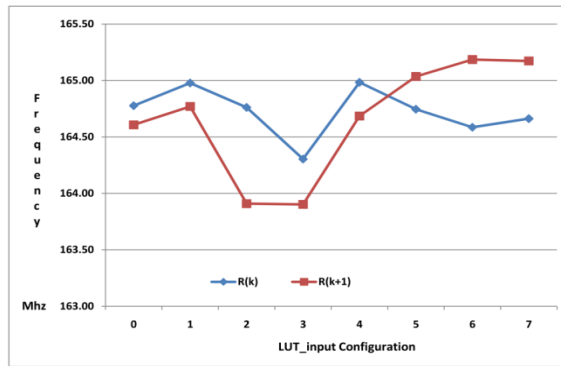
Figure 46: Crossover of two rings

A thresholding technique as explained below is employed to discard those comparisons which are vulnerable to producing "bit-flips".

The normalized inter-chip Hamming distance, $(HD(R_i, R_j)/L) \cdot 100\%$ is shown below,
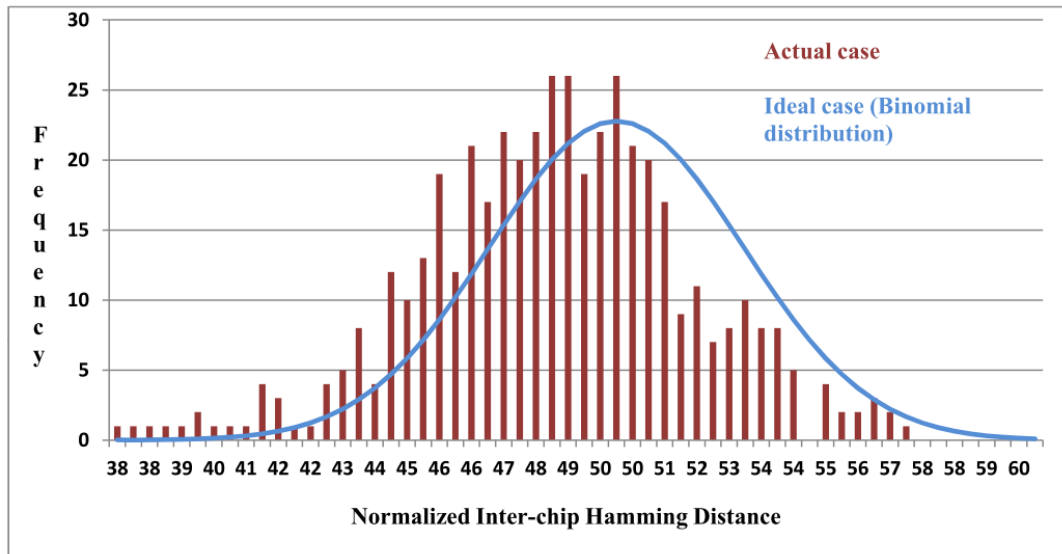
Figure 47: Normalized inter-chip Hamming distance

The mean is 48.48%. This data is generated from 31 FPGA boards at 25°C and nominal voltage supplied at the core. The total number of combinations (i.e., the total number of board pairs {i,j}) is $\binom{31}{2}$= 465. The y-axis (denoted frequency) shows the total number of times a given normalized inter-chip Hamming distance was obtained. In this data, 248 bit response from each FPGA board is used. The reason is that in our data set, the minimum number of bits generated by any FPGA device using the crossover method was 248 bits.

Thresholding Technique:

If an ID bit is flipped during regeneration then it reduces the reliability. In order to avoid this condition, a thresholding technique is employed which accepts a bit if the difference in frequency for any comparison is greater than 150kHz ($\Delta f \geq 150$ kHz). Otherwise we discard that bit, because it can be flipped by noise during regeneration. But if the

frequencies of two rings fail the thresholding condition ($\Delta f \geq 150$ kHz), then only one bit

is contributed towards the chip ID, and this bit is generated by the majority vote of 8

comparisons. Furthermore, Δf is fixed at 150 kHz, because we observed that the average

standard deviation among 20 samples of each frequency is around 30 kHz for all the rings.

Therefore, we keep it at least 5 times the standard deviation. After applying this condition,

the number of strong bits per chip is 283 as shown in Table 16. We call this sequence of

bits the Chip-ID. It is important to mention here that 283 bits is the minimum length of bit

string generated by a particular FPGA, all other devices generated more than 283 bits.

The more different a pattern is from another ring oscillator in absolute frequency terms, the

stronger 8-bits we will get from their comparison.

Table 16. Details of dataset

	Maiti et al. [34]	This work
No. of Devices (N)	193	31
Samples (T)	100	20
No. of ID's (K)	1	1
ID size (L) bits	511	283
Ring oscillators (M)	512	130
FPGA family (Device used)	Spartan-3E (XC3S500E)	Spartan-3E (XC3S100E)

In Tables 16, 17 and 18 we compared our results with the results shown by Maiti et al. in [34].

Table 17. Comparison with Maiti et al. [34]

	P1	P2	Maiti	Ideal
Uniformity	50.13	50.75	50.56	50%
Uniqueness	96.6	95.34	93.98	100%
Bit-aliasing	51.8	50.75	50.56	50%
Reliability	97.88	98.1	99.13	100%

We measured our PUF responses at room temperature and then generated results shown in Table 17, by running a script available at [78] on our PUF data.

From Table 17 and 18, it is evident that our result set is comparable to Maiti et al. However, we believe that with four times smaller PUF size (in-terms of the number of CLB slices for Ring Oscillators) we were able to generate more than twice as many bits per ring oscillator. Furthermore, our PUF-IDs are more biased towards '1', resulting in Uniformity greater than ideal by 0.13%. All the five PUF properties are thoroughly explained in [58].

Table 18. Properties of independent strong bits

	This work (P1)	This work (P2)	Maiti et al.[34]
Number of ring oscillators [A]	130	130	512
Average Independent, strong response bits[*] [B]	283	318	511
Bits per Ring [B/A]	2.17	2.44	~1

*strong bit = When the $\Delta f \geq 150kHz$, [Average of 31 Devices Frequency $\approx$ 165MHz]

6.5. Frequency Analysis

To show the extent of randomness offered by the chip under test and extracted by our design, we analyzed the frequencies of all rings under all configurations. The frequencies of all 31 devices are shown in Fig. 48. Each frequency is the average of all 1040 frequencies generated by ROs of a given chip. Each ring generates 8 different frequencies and there are 130 rings in total. The standard deviation in the frequency of 1040 points per chip is shown in Fig. 49.
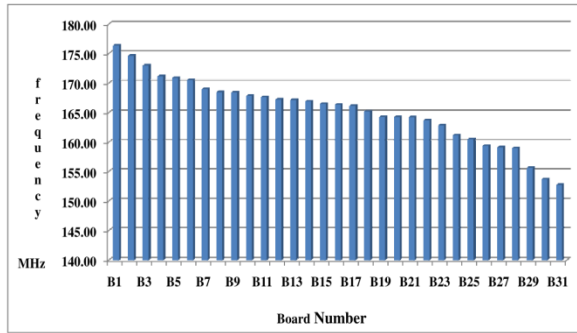
Figure 48: Average frequency of all ring oscillators located on each board

The average frequency of each ring oscillator averaged over all boards is shown in Fig. 50. Each point in this figure is the average of (8 frequencies · 31 boards) = 248 frequencies. From Fig. 50, it is evident that the highest peaks in the frequency occur at the central part of the chip. While the lowest frequencies occur at the edges. This behavior has also been reported in [8]. Our PUF layout is not placed exactly at the center of the chip as explained in section IV, therefore in Fig. 50 the highest peaks seem off the center.
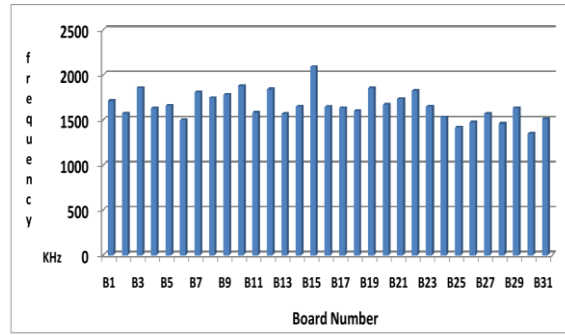
Figure 49: Average standard deviation in frequency of 1040 points per board
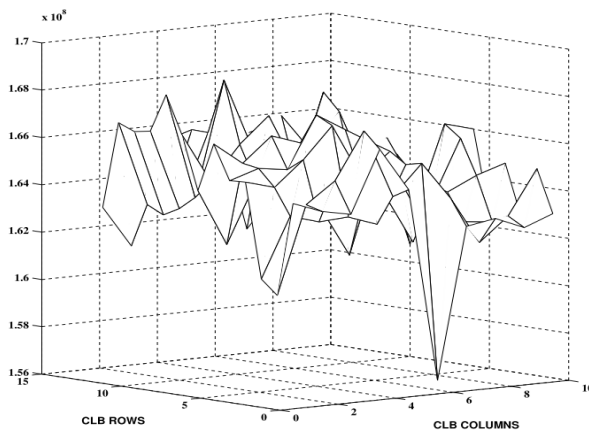
Figure 50: The average frequency of all boards for 130 Ring oscillators depending on

CLB locations

Systematic and Manufacturing Variation

The frequency of any ring oscillator consists of both systematic variation and manufacturing process variation. For PUF statistical and qualitative analysis we need to decompose these variations into within-die and die-to-die variations.

Decomposition Methodology

With the introduction of programmable delay lines, we were able to generate eight different frequencies from a single ring oscillator. Therefore we had 1040 total frequencies for 130 ring oscillators. We analyzed 130 rings that make the rectangle shown in Fig. 40 at the central part of the chip. This rectangle is a matrix of 13x10 ring oscillators. We laid these 1040 frequencies in the form of a matrix with 13 rows and 80 columns. Any point in this matrix is denoted by F(x,y), where x ranges from 0 to 79 and y ranges from 0 to 12, i.e., each row contains 80 frequencies from 10 ring oscillators. Other dimensions could also be employed for this experiment. F(0,0) is the frequency of ring oscillator 0, with LUT_input configuration equal to '000'. We call F(0,0) as the nominal frequency in our calculations.

We decomposed the frequency of rings into random and systematic variation components. Our decomposition method follows the method explained in [42], as shown in equation .

$$F_{(x,y)} = F_{(0,0)} + R_{WID} + S_{WID(x,y)} + S_{D2D(x,y)} \quad (19)$$

Here, RWID is the random within die variation component, SWID is the systematic within die variation component while SD2D is the systematic die to die variation.

We used Down Sampled Moving Average (DSMA) to extract the random and systematic variation from equation 19. In DSMA we moved the window over 1040 points and we got the average frequency of all the points in a window.

$$DSMA_{(x,y)} = (2z+1)^{-2} \sum_{i=x-z,\, j=y-z}^{x+z,y+z} F_{(i,j)} \quad (20)$$

The window size is 9 with a 3x3 dimension, by setting z=1 in equation (20). We keep z = 1, because with a big window size we will be averaging too many points, which will suppress the randomness due to programmable delay lines.

$$DSMA_{(x,y)} = F_{(x,y)} - R_{WID} \quad (21)$$

From equation 21, we got

$$R_{WID} = F_{(x,y)} - DSMA_{(x,y)} \quad (22)$$

For 1040 points in total and z=1, we get 858 random values. We normalized it over $F_{(0,0)}$ to get the R_{WID} variation. It has been shown in Fig 51.
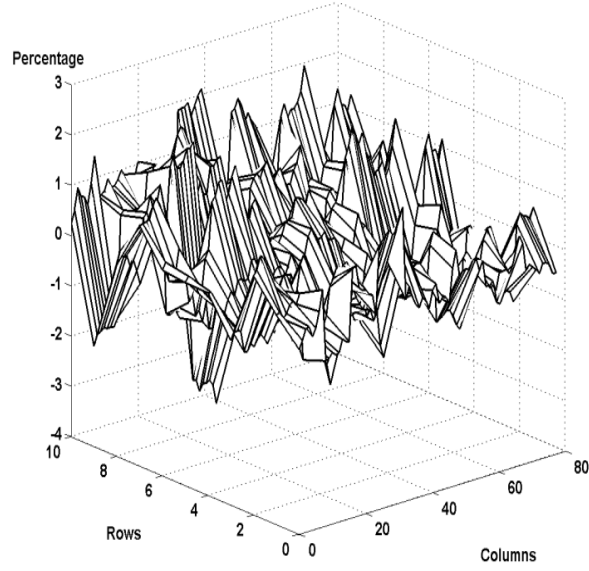
Figure 51: Random within die variation (normalized over F(0,0), shown as a

percentage)

The properties of random within die variation has been listed in Table 19.

The distribution of this randomness is shown in Fig. 52. The distribution of Random

within Die variation is plotted using a histogram. It is evident that the plot is centered at

0.0.

Table 19. Properties of Random with-in die variation normalized over F(0,0)

Mean	0.0
Min	-3.23 %
Max	2.27 %
Peak to Peak	5.5%
Standard Deviation	0.84%

6.6. Conclusion

In this work we presented a novel PUF design, based on ring oscillators, where the programmable delays of FPGA LUTs were used to generate additional bits of an ID. The strength of this design is its ability to generate more than one random frequency per ring oscillator without changing the path of the ring outside LUTs. This solution offers the option to reduce the area requirements of ring oscillator PUFs. To demonstrate the strength of this PUF, it was shown that our design generated more bits per ring oscillator, and these bits are as strong as the ones reported in literature for Configurable Ring Oscillators. The statistical and PUF properties were analyzed and were shown to be very strong from a security point of view.
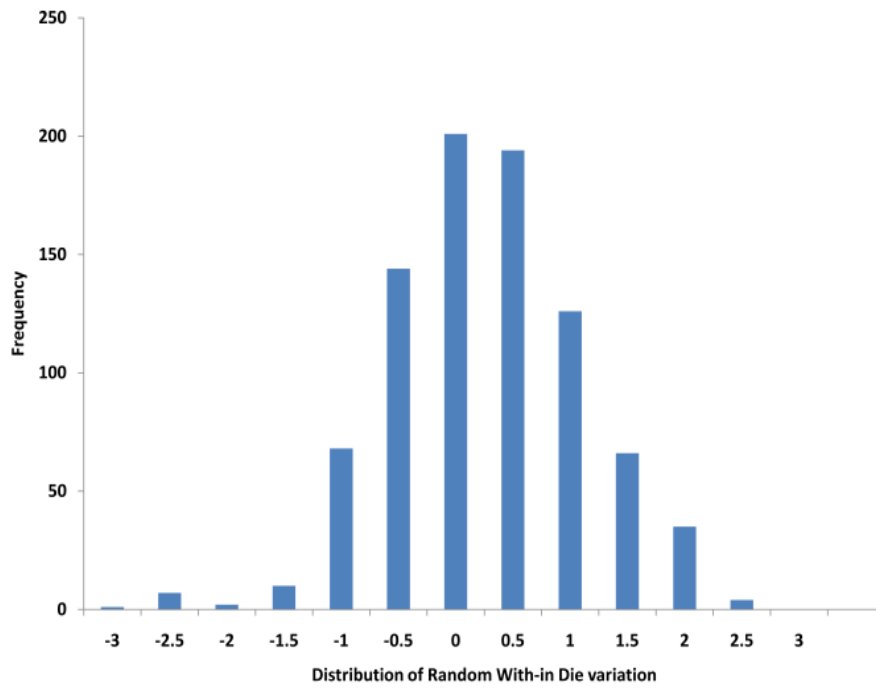
Figure 52: Distribution of the random within die variation

7. A Comprehensive Set of Schemes for PUF Response Generation

In this chapter a comprehensive set of schemes to generate Physical Unclonable Functions (PUF) responses, is presented. Software scripts have been developed to generate the PUF IDs using five different schemes. We also propose a new set of PUF metrics that are based on the Worst Case (WC) PUF performance. Values of these metrics, for the Ring Oscillator (RO) PUF and SR-Latch PUF, have been presented and analyzed in the chapter.

Contents

7.1. Introduction

Extensive research has been going on to develop new hardware primitives for security. PUF is one of such primitives that can efficiently solve several problems in hardware security. These problems range from reverse engineering, counterfeiting, to detection of pirated devices. For this purpose the two important applications where PUF can be used

are: key generation and device authentication. The input to PUF is called a challenge and the output is called a response. Therefore it works in the Challenge Response Pairs (CRPs). In the past researchers used only one scheme to generate the PUF responses. For more in-depth analysis we need to generate responses using more than one scheme. It gives us the flexibility to control the response bit size. We present five most popular schemes in this work. Additionally we also explain the effect on PUF metrics when different schemes are used. In Section 8.2, we describe the previous work for a better understanding of our study on PUF. Section 8.3 explains the PUF schemes; methodology is covered in section 8.4. In Section 8.5, we discuss the results. The conclusion and future study are described in Section 8.6.

7.2. **Previous Work**

The previously reported schemes for PUF response generation from raw data included: comparing the counts of neighboring ROs [34], Lehmer-Gray encoding method [46], Identity-mapping [50] and S-ArbRO method [51]. Quantitative and statistical performance evaluations of Arbiter PUF and RO-PUF were presented in [26] and [58], respectively. In this work we analyze PUF responses generated using five different schemes. Each scheme is described in detail using a pseudocode. For evaluation of responses, PUF metrics have been developed.

7.3. **PUF Schemes for ID generation**

PUF responses are generated from the raw PUF data as shown below in Fig. 53,
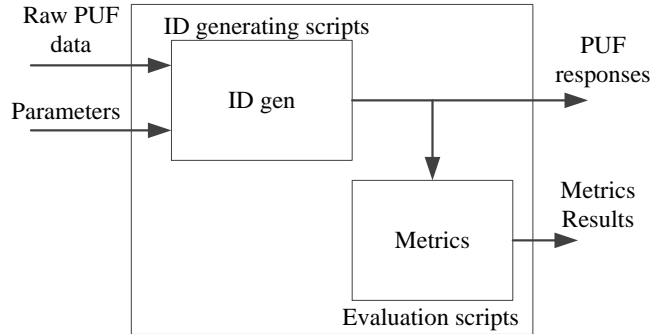
Figure 53: PUF ID generation and evaluation

As shown above in Fig. 53, the raw PUF data is the input to our PUF generation module. In case of RO-PUF raw input consists of the number of ring-oscillator periods within a predefined fixed time interval. Similarly in case of SR-Latch PUF, raw input consists of the number of oscillations of a latch during metastable state. In this work we used three data sets; Spartan-3 data-set for ROs [78], Spartan-6 data-set for SR-Latches [73] and Zynq data-set for SR-Latches. Devices used for Zynq data belongs to (XC7Z010) family. Five schemes are explained below,

i) **Comparing the neighboring components (CNC)**

In this scheme raw data from the neighboring components of FPGA are compared. Fig. 54 shows this scheme. In this figure $C_0, C_1…C_{M-1}$ shows the physical location of components

on the FPGA fabric. Raw data from the neighboring components are compared to mitigate

the effect of systematic variation.

Algorithm 1 Comparing the Neighboring Components

```
CNC( F[], M):            //F[] is the input array of M
components.
for(i = 0; i< M-1; i++)
        if(F[i] > F[i+1])
               Response[i] = 1
        else
               Response[i] = 0
        end if
end for
```

Where `Response` array stores the PUF response of M components. In this scheme each

component is compared with both neighbors except the first and last component. In case

of first and last component, comparison is done only with a single neighbor. The

comparison of (C_0,C_1), (C_1, C_2), (C_2, C_3)... (C_{M-2},C_{M-1}) is carried out. Therefore for M

components the total number of PUF response bits will be equal to M-1.
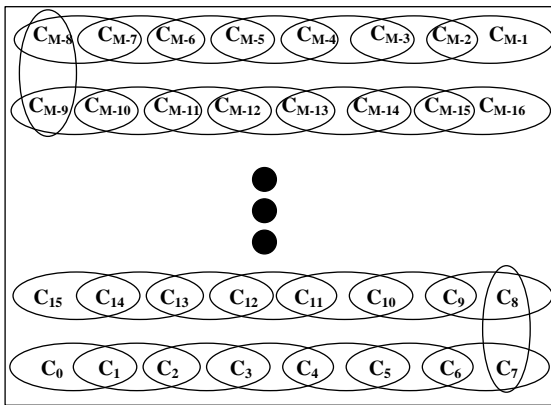
Fig. 54. Comparison of neighboring Components   Fig. 55. Pairwise comparison with neighbors
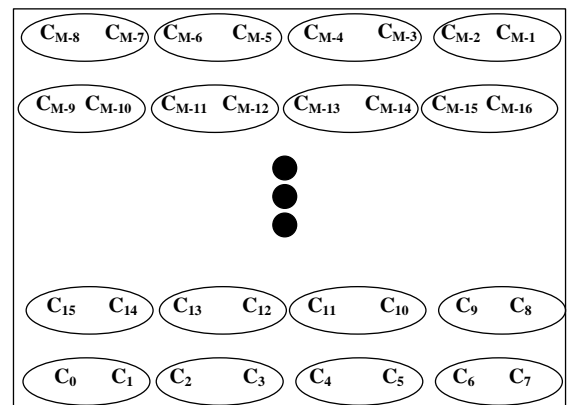
ii) **Pairwise Comparison (PC)**

In this comparison each component is compared only once with its neighbor. The comparison of (C_0, C_1), (C_2, C_3), (C_4, C_5)…. (C_{M-2}, C_{M-1}) is carried out. Therefore for M components the total number of PUF response bits will be equal to $\lfloor M/2 \rfloor$. Fig. 55 shows this scheme.

Algorithm 2 Pairwise Comparison
| ```
PC(F[], M):              //F[] is the input array of M
components
for(i=0; I < 2 · ⌊M/2⌋; i+=2)
       if(F[i] > F[i+1])
              Response[i] = 1
       else
              Response[i] = 0
       end if
end for
``` |

Where `Response` array stores the PUF response of N components.

iii) **Binary Lehmar-Gray (BLG) encoding**

In LG encoding, all components are divided into sets of size S. Encoding the ordering of S component measurements $F^s = (F_0, \ldots, F_{s-1})$ results into an L-bit response. A Lehmer code is a unique numerical representation of an ordering which is moreover efficient to obtain since it does not require explicit value sorting. It represents the sorted ordering of F components as a coefficient vector $L^{s-1} = (L_1, \ldots, L_{s-1})$ with $L_i \in \{0, 1, \ldots, i\}$. It is clear

120

that L^{s-1} can take $2 \times 3 \times \ldots \times S = S!$ possible values which is exactly the number of possible orderings. The Lehmer coefficients are calculated from F as $L_j = \sum_{i=0}^{j-1} gt \ (F_j, F_i)$ with $gt \ (x, y) = 1$ if $x > y$ and 0 otherwise. The Lehmer encoding has the nice property that a minimal change in the sorted ordering caused by two neighboring values swapping places only changes a single Lehmer coefficient by ± 1.

The total number of bits generated for each set is:

$$\text{Bits generated per set (L)} = \sum_{i=2}^{S} \lceil \log_2 i \rceil \tag{22}$$

Below is a pseudo code for converting counts of M components into an $(M/S) \cdot L$ -bit response denoted by Response.

Algorithm 3 Binary Lehmer Gray Encoding

```
BLG (F[], M, S):   //F[] is the input array of components, S=Set size.
for(t=0; t < ⌊M/S⌋ ; t++)

    Response= Response || Lehmer(F[t·S:(t+1)·S-1], S)
end for

Lehmer(array[], S):
for(j=1; j< S ; j++)
     sum = 0
     for(i=0; i<j ; i++)
        if (array [j] > array [i])
          sum= sum + 1
        end if
     end for
     L_Response = L_Response ||(Gray(bin(sum, ceil(log₂(j+1)))))
end for
return L_Response

Gray(bin_bits):   //array of binary bits is passed to Gray().
Len_bits=len(bin_bits)            //Length of an array
G[0] = bin_bits[0]
for(i=1; i< Len_bits; i++)
        G[i] = XOR(bin_bits [i], bin_bits [i-1])
end for
```

Above a function named `bin(p,t)`, converts a decimal number `p` to `t` binary bits. For M components , the total response is equal to $(M/S) \cdot L$ bits. In [46], set size S, is 16. Similarly,

a function named `Gray()` is used for encoding the Lehmer co-efficients. Gray encoding make it sure that each subsequent number is only a single bit different than a previous one. Where `bin_bits` is an array of binary bits. `G` is an array of corresponding Gray encoded bits.

iv) **S-ArbRO-2**

S-ArbRO-2 is described in [51]. In this design the number of CRPs have been improved. Components are divided into elements. Each element has a pair of components associated with it as shown below in Fig. 56,

Fig. 56. Element contains a pair of components

The difference between the counts of components in each element is the respective count associated with that element (r_1-r_2 or r_2-r_1). This difference in count value may be positive or negative. The next step is to select a group size for elements. This is done by selecting a value for parameter K. Inside this group, elements are added with each other. The range of K is $2 \leq K \leq N$. Here, N is the total number of elements and K is the number of elements selected in each group. Challenge is the selection of group of elements, while response (Rc) is the one bit result as shown in Fig. 57,

Algorithm 4 S-ArbRO-2

```
S-ArbRO-2(E[], K):      //K is the parameter passed.
combo=[]    //It will hold all the possible combinations //of groups of
                             Elements.
sums=[]   //It will hold the result of all the additions.
Kp= 2 ^ (K - 1)             // power(2, K-1).
i = 0
for x in combinations(E,K): // Generate all combinations
                            // of K out of N elements.
           combo [i]= x
           i = i + 1
end for
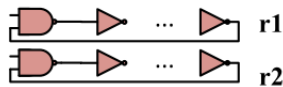
for(i=0; i< len(combo); i++):
    for(j = 0; j< Kp; j++):
     temp=[]
     p=bin(j,K)                //convert j into K binary bits.
     for(s=0; s<len(p); s++):
       if(p[s] =='0')
         temp = temp || (combo[i][s][0]- combo[i][s][1]) //r1-r2
       else
          temp = temp || (combo[i][s][1]- combo[i][s][0]) //r2-r1
       end if
      end for
     sums= sums || (sum(temp))
   end for
end for


for(i=0; i< len(sums); i++):
        if(sums[i]> 0)
                 Response[i]=1
         else
                 Response[i]=0
        end if
end for
```

If the result of adding elements is positive, the response is '1', otherwise it is '0'. The total

Challenge Response (CR) space is,

$$Total\ CR\ space = \frac{N!}{K! \times (N - K)!} \times 2^{K-1} \quad (23)$$

Fig. 57. S-ArbRO-2 showing the relationship between Challenge Response Pairs (CRPs)

As evident from the Fig. 57, the total number of elements is N. For example, 64 components will result in N = 32 elements. Suppose the parameter K is equal to 2. Then the total number of possible combinations are 992. In the pseudo code, E[] is an input array of N elements.

Each Element has 2 components. K is the subset size. Combo holds all the possible combinations of $\binom{N}{K}$ elements. Sums will hold all the sums for K elements. Combination(N,K) calculates the $\binom{N}{K}$, sum(array) adds all the elements of an array. The response of S-ArbRO-2 is returned by an array Response[]. The pseudo code will generate all the possible CRPs. Assume the list of components is [10,5,6,4,17,11]. Therefore the three elements formed are E1=[10,5], E2=[6,4] and E3=[17,11]. The possible number of combinations for K = 2 is $\binom{3}{2}$ = 3. Hence three groups of elements formed will be {E1, E2}, {E1, E3} and {E2, E3}. Combo will contain [{E1, E2}, {E1, E3},{E2, E3}] or [{[10,5], [6,4]}, {[10,5], [17,11]},{[6,4], [17,11]}]. If the group challenge is (01), it will select group {E1,E3}. Similarly inside each group if the challenge is (00). It will result in

125

0=>E1[r1-r2]= 10-5= 5 and 0=>E3[r1-r2]= 17-11= 6. Sums will hold 5+6=11. Since 11>0, therefore the final PUF response will be '1'.

v) Identity Mapping (Id-Map)

This scheme is described in [50]. In identity mapping m components can generate $2^M - M - 1$ response bits. In this method, t component counts are selected from m component counts where $2 \leq t \leq M$. Initially all pairs of component counts are determined S2,

$$|S_2| = \binom{M}{2}$$

Similarly, S3 contains all possible triplets of component counts.

$$|S_3| = \binom{M}{3}$$

Likewise,

$$|S_t| = \binom{M}{t}$$

Then a random variable Q_t is defined that assigns a real number X to each outcome of S_t

$$Q_t: S_t \rightarrow X \; such \; that$$

$$Q_t(f_{x1}, f_{x2}, f_{x3}, \dots. f_{xt}) = \sum_{u=1}^{t-1} \sum_{v=u+1}^{t} w_{(xu)(xv)} \cdot \left\| f_{(xu)} - f_{(xu)} \right\|^e \quad (24)$$

Where $1 \leq x1, x2, x3, \dots, xt \leq M$

And, $x1 \neq x2 \neq x3 \neq .. \neq xt$ and $2 \leq t \leq M$

The weight factor $w_{(xu)(xv)}$ can depend on a particular design. However, in our script it is equivalent to 1. We chose 1 because we believe that systematic variations come into the effect, when far away components are compared. Therefore we keep the weight factor constant for all Q values. Response R, from Q is generated by using the following equation,

$$R = mod(Q[i]/q, 2)) \ for \ i = 0,1,2,..$$

Where q is the bucket size. The size of array Q depends on the value of t selected. For instance if t =2, the total elements of Q are $\binom{M}{2}$. If t=3, then total length of Q will be $\binom{M}{2}$ + $\binom{M}{3}$, and so on.

In addition to the response bits, a set of helper data is also generated. This helper data is used to reduce the effect of noise in the field. For example, with noise the count value for a component is different from the one calculated during enrolment. Therefore, a helper data is used to mitigate the effect of noise.

Algorithm 5 Identity Mapping

```
Identity_map(components[], q, t, e):
               //q, t and e are parameters.
S=[]           //It will hold the (M)
                                  (t) component counts.
Q=[]           //It will hold the Qs.
i = 0
for x in combinations(components,t)
        S[i]= x
        i = i + 1
end for

j = 0
for a,b in combinations(S,2)
        Q[j]= (|a-b|)ᵉ
        j = j + 1
end for

for(i=0; i< len(Q); i++)
        Response_enrollment [i]= (Q[i]/q) mod 2
        Wt[i] = (0.5 · q) - (Q[i]- ⌊ (Q[i]/q) ⌋ · q)
end for
for (i=0; i< len(Q'); i++)      //Q' is generated in the
field.
        temp[i]= ( Wt[i] + Q'[i])       //error correction
        Response_field[i]= (temp[i]/q) mod 2
end for
```

Helper data W_t is calculated using the following equation,

$$W_t = \binom{q}{2} - (Q[i] - q \cdot \lfloor \tfrac{Q[i]}{q} \rfloor) \ for \ i=0, 1, 2, 3...$$

In the above equation, q is the bucket size. It must be appropriately chosen. If q is chosen very big, then too many bits will be encoded into the same bucket. Therefore, it will reduce the uniqueness significantly. Similarly, if q is chosen very small, then it will affect the reliability property. A small change by the noise in the field will move the response bit to another bucket. It must be noted that for each element of Q, only one value of W_t is calculated.

In the pseudo code, components[] is an input array that contains the counts of components. q is the bucket size , parameter e is any real number except 1 and t is the parameter , such that $2 \leq t \leq m$. PUF Response during enrolment is stored in an array Response_enrollment. While Wt is the array that contains the helper data. In the field, each response bit is recalculated using W_t and noisy Q' values. Response_field[] contains the PUF response generated at the field.

7.4. **Methodology**

We use three properties of PUF to rank the schemes. These properties are Average uniformity, WC uniqueness and WC reliability. We chose the WC because in key generation and device authentication we are more interested in the worst case uniqueness and reliability. These properties are explained here,

Uniformity of a PUF estimates how uniform the proportion of '0's and '1's are in the PUF response of a device. It is calculated using equation 25,

$$\text{Uniformity (i)} = \frac{1}{L} \sum_{l=1}^{L} r_{i,l} \times 100\% \qquad (25)$$

128

Where $r_{i,l}$ is the lth binary bit in the response of a chip i. Response of each device is L bits. Average uniformity of N devices is shown in equation 26. The optimal value is 50% for a set of N devices.

$$\text{Avg Uniformity} = \frac{1}{N} \sum_{i=1}^{N} \text{Uniformity (i)} \qquad (26)$$

The best among five schemes in terms of Average Uniformity is based on equation 27.

$$\text{Best scheme} = \text{Min}_{s=1}^{5} |(|50\% - Avg\ Uniformity(s)|) \quad (27)$$

Worst case uniqueness is equal to minimum relative Hamming Distance between the response R(i) and response R(j), as well as between R(i) and complement of R(j). It is determined by looking at all pairs of responses from devices i and j. Its optimal value is equal to 50%. It is calculated at the room temperature and nominal voltage. It is defined as,

$$\text{WC Uniqueness} = \text{Min}_{i=1,j=i+1}^{i=N-1,j=N} \left(\frac{\text{Min}(\text{HD}(\text{Ri}, \text{Rj}), L - \text{HD}(\text{Ri}, \text{Rj}))}{L} \right) * 100\% \ (28)$$

Where L is the length of response bits. Ri and Rj are the responses of two FPGA devices. To determine the best scheme we chose the one that has the highest WC uniqueness. The worst case reliability is calculated using the following equation,

$$\text{WC Reliability} = \text{Min}_{i=1}^{N} \left(1 - \frac{\overset{C}{\underset{c=0}{\text{Max HD}(\text{R}_i, \text{R}_{i,C})}}}{L} \right) * 100\% \qquad (29)$$

It describes how close to the 100% reproduction of the PUF bits a given scheme is getting in the worst case. Optimal value is equal to 100%. In equation 29, HD is the hamming distance. Ri is the response of device i at the nominal condition. Ri,c is the response of device i in the field. L is response size in bits and C is the total number of conditions applied

129

in the field. The best scheme in terms of WC reliability is the one that has the highest value

for equation 29. Overall, we used the following equation to determine the optimum

parameters:

$$Z = Max\{50\% - Avg\ uniformity, 50\% - WC\ uniqueness, 100\% - WC\ Reliability\} \quad (30)$$

We choose the parameter for any dataset which gives the smallest value for Z according to

equation 30. These parameters are shown in Tables 21 to 23.

The input to PUF-ID generating scripts is in the chip-row format. In this format the rows

contain the data for a particular device while the columns contain the M components. This

format is shown in Fig. 58,

Fig. 58. Data input format for PUF-ID generating scripts

'Comma separated values' (CSV) format has been used to store the input data. Only one

value is stored in the input file per component. It is done by taking the median of all the

samples per component. The output of PUF-ID generating scripts is stored in the text (.txt) format. It consists of 1's and 0's. The same text files are then input to the evaluation scripts. The result of evaluation script is then stored in a separate text file.

7.5. **Results**

In order to generate same number of PUF response bits using different schemes. We chose the PUF response size of 128 bits. It is due to the fact that Pairwise Comparison (PC) generates only 128 bits using all the 256 components of SR_latch data.

Table 20**.** Details of dataset.

	CNC	PC	BLG	S-ArbRO-2	Id-Map
Parameters			Set size = 16	K= 2	t=2
PUF Response length (L)	128	128	128	128	128
Min Components Required (M)	129	256	48	24	17
PUF Response (L)	M-1	$\lfloor M/2 \rfloor$	$(M/S)*L_s$†	$\binom{M}{K} * 2^{k-1}$	$\binom{M}{t}$

† For S = 16, L_s=49

a. **Zynq Data set**

Total devices = 10, Total Components per device = 256

Table 21. Zynq data.

	CNC	PC	BLG	S-ArbRO-2	Id-Map
Parameters			Set size =16	K = 2	e=0.5, q = 2, t=2
Inter-chip HD Mean	49.02%	49.87%	46.64%	50.46%	50.14%
Inter-chip HD Min	38.28%	42.19%	36.72%	29.69%	39.84%
Inter-chip HD Max	57.03%	58.59%	53.91%	71.88%	58.59%
Inter-chip HD Std Dev	4.48%	3.36%	3.8%	10.28%	4.37%
WC Uniqueness	38.28%	**41.41%**	36.72%	28.12%	39.84%
Avg Uniformity	46.87%	46.79%	**47.89%**	54.14%	47.34%
Std Dev Uniformity	1.39%	4.49%	4.58%	10.19%	12.44%

131

b. Spartan-6 Data set

Total devices = 25, Components per device = 256

Table 22. Spartan-6 Data.

	CNC	PC	BLG	S-ArbRO-2	Id-Map
Parameters			Set size =16	K = 2	e=0.5, q = 1, t=2
Inter-chip HD Mean	49.36%	49.25%	46.12%	49.77%	46.37%
Inter-chip HD Min	35.16%	33.59%	32.81%	25.78%	34.38%
Inter-chip HD Max	63.28%	60.16%	59.38%	83.59%	62.5%
Inter-chip HD Std Dev	5.2%	4.45%	4.34%	10.41%	4.82%
WC Uniqueness	**35.16%**	33.59%	32.81%	16.41%	34.38%
Avg Uniformity	44.03%	44.71%	44.65%	**52.96%**	63.68%
Std Dev Uniformity	1.95%	4.52%	4.24%	8.63%	6.41%

c. Spartan-3 Data set

Total devices = 193, Components per device = 512

Table 23. Spartan-3 data set.

	CNC	PC	BLG	S-ArbRO-2	Id-Map
Parameters			Set size =16	K=2	e=0.5, q = 30, t=2
Inter-chip HD Mean	46.83%	46.61%	45.85%	46.13%	48.76%
Inter-chip HD Min	28.91%	30.47%	26.56%	13.28%	24.22%
Inter-chip HD Max	65.63%	62.5%	66.41%	83.59%	64.84%
Inter-chip HD Std Dev	4.86%	4.45%	4.76%	10.2%	4.47%
WC Uniqueness	28.91%	**30.47%**	26.56%	13.28%	24.22%
Avg Uniformity	49.86%	52.25%	47.70%	**50.24%**	57.80%
Std Dev Uniformity	2.53%	4.58%	6.70%	9.39%	4.96%

From Table 21, 22 and 23, it is evident that the best scheme in terms of WC uniqueness is

PC for both Zynq and Spartan-3 datasets. However for Spartan-6 data it is the CNC scheme

that generates the best results. Similarly for Average uniformity the best results are offered

by BLG in Zynq dataset and S-ArbRO-2 in both Spartan-6 and Spartan-3 datasets.

Tables 24-28 shows the worse case reliability as defined in equation (29). Spartan-3 data

set contains voltage and temperature data for only five devices. Similarly Zynq data set

contains voltage and temperature data for ten devices. Additionally the Spartan-6 data set

contains voltage and temperature data for fifteen devices. The nominal voltage of Zynq

devices is 1V on the other hand it is 1.2V for both Spartan-6 and Spartan-3 devices.

Table 24. Comparing the neighboring components (CNC).

	Zynq	**Spartan-6**	**Spartan-3**
PUF Type	SR-Latch	SR-Latch	RO-PUF
No. of devices tested	10	15	5
Rel @ +5% V	93.75%	99.21%	**87.50%** (+10%V)
Rel @ -5% V	93.75%	95.31%	91.40% (-10%V)
Rel @ 85°C	**91.40%**	**94.53%**	95.31% (+65°C)
Rel @ 0°C	94.53%	96.87%	N/A

Table 25. Pairwise Comparison (PC).

	Zynq	**Spartan-6**	**Spartan-3**
Rel @ +5%V	93.75%	97.65%	**91.40%** (+10%V)
Rel @ -5%V	93.75%	**96.09%**	92.96% (-10%V)
Rel @ 85°C	**89.84%**	96.09%	92.96%(+65°C)
Rel @ 0°C	94.53%	96.09%	N/A

BLG encoding: Set size (S) = 16, M=48, PUF response size = $(M/S) \cdot 49$

Table 26. Binary Lehmar-Gray (BLG) encoding.

	Zynq	**Spartan-6**	**Spartan-3**
Rel @ +5% V	86.71%	94.57%	**83.59%** (+10%V)
Rel @ -5% V	85.93%	90.62%	84.37% (-10%V)
Rel @ 85°C	**78.9%**	**85.93%**	90.62% (+65°C)
Rel @ 0°C	89.84%	92.18%	N/A

S-ArbRO-2: Parameter K = 2 , Elements = M/2

Table 27. S-ArbRO-2 scheme.

	Zynq	**Spartan-6**	**Spartan-3**
Rel @ +5% V	92.18%	96.87%	48.14%(+10%V)
Rel @ -5% V	92.96%	96.09%	**38.2%** (-10%V)
Rel @ 85°C	**83.59%**	**89.06%**	39.06%(+65°C)
Rel @ 0 °C	92.96%	93.75%	N/A

Identity Mapping: Parameters, e=0.5, t = 2

From Table 24-28, the best scheme in terms of worst case reliability is CNC for Zynq, PC for Spartan-6 and Id-Map for Spartan-3 datasets. The bold values shown in each column shows the minimum value. For any dataset the best scheme is chosen that results in the highest bold value

Table 28. Identity Mapping scheme.

	Zynq	**Spartan-6**	**Spartan-3**
Parameter chosen	q=2	q=1	q=30
Rel @ +5% V	**75.81%**	89.84%	94.53% (+10%V)
Rel @ -5% V	91.4%	**85.93%**	**92.96%** (-10%V)
Rel @ 85 °C	64.84%	89.84%	97.65%(+65°C)
Rel @ 0 °C	92.18%	90.62%	N/A

The worst result is offered by S-ArbRO-2 for Spartan-3 dataset. It might be due to the fact that certain ROs are affected more by voltage and temperature variation than others. In S-ArbRO-2 if the sign of the elements change in the field. Then it results in a bit flip, hence low reliability.

7.6. **Conclusion and Future work**

From this work we conclude that PUF responses should be generated using multiple schemes to determine the uniformity and worst cases of uniqueness and reliability. S-ArbRO-2, Lehmer-Gray and Identity mapping offers the ability to use less number of components for PUF design; however the CRPs are no longer independent. Additionally the effect of systematic variation is not taken into account when Identity mapping is used, especially for RO-PUF. BLG scheme offers the best results in terms of uniformity in Zynq, however it is the S-ArbRO-2 scheme that generates the best uniformity results for both Spartan-3 and Spartan-6 datasets. In case of uniqueness PC scheme offers best results for both Zynq and Spartan-3 datasets, however it is the most expensive scheme in terms of area. In case of Reliability we have no winner as three different schemes offer the best results for three data sets. In the future we intend to enhance the scope of our method by including the Arbiter PUF data set [79] in our analysis.

Appendix

Configurations based comparison

In [34], a scheme based on the configurations of RO is developed. In this scheme eight configurations are used for each RO as shown in Fig. 59. Each configuration results into a different RO frequency. The difference between frequencies (Δf_1, Δf_2, Δf_3....Δf_8) of two neighboring ROs is chosen based on the highest absolute difference in frequency among a set of eight values. This scheme is intended for improving the reliability of PUF. If the difference between two ROs is very high then the chance of a bit flip reduces due to noise in the field.

Figure 59: Selection of the RO-pair with the maximum frequency difference between two configurable ROs.

```
Pseudo code:
i = 0
While (i < N-1)
        j =0
        Max = |f[i][j] - f[i+1][j]|
        While (j<8)
                If(|f[i][j] - f[i+1][j]| > Max)
                        Max = |f[i][j] - f[i+1][j]|
                        Index_Max = j
                j = j + 1
        If(f[i][ Index_Max] > f[i+1][ Index_Max])
                Response[i] = 1
        Else
                Response[i] = 0
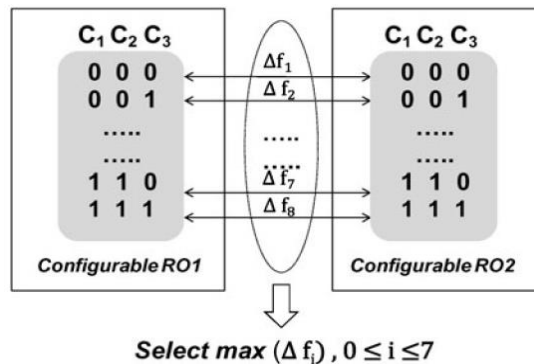
        i = i + 1
```

In the above pseudo code, `f[i][j]` represents a frequency of ring oscillator `i` with configuration `j` selected. For N, ROs the total number of bits generated are N-1.

8.   Research Contributions

My research primarily focused on three areas of PUF. These areas are shown below in Fig. 60.

Figure 60: Dimensions of my proposed research

As evident from Fig. 60, the three main areas are: Robustness, Efficiency and Portability.

All three are critical for the quality of PUF. For instance, if a PUF is very robust from

reliability point of view and portable to a large number of devices but very inefficient from cost point of view, then it will not be regarded as an attractive design. We have to integrate all three dimensions while developing a PUF based system. Ignoring any one of them will seriously affect the quality of PUF. To comprehensively cover the PUF space we developed two new types of PUFs. One is memory based and the other is delay based PUF. In the memory based PUF we developed a new SR-Latch design. Similarly, in the delay based PUF we developed a new RO-PUF design. Both designs are summarized in section 8.1 and 8.2 respectively. The dimensions of PUF space namely Robustness, Efficiency and Portability are covered in section 8.3 to 8.5.

8.1.   **Development of new SR-Latch PUF**

The state of the art SR-Latch PUF [44, 53] is very expensive from area point of view. It requires 2 CLBs of an FPGA to generate a single bit of PUF response. It is prone to the affect of nearby logic. Thus the reliability of PUF bit is severely affected if the tool configures the same CLB with external logic. Our design approach is based on determining the length of metastable state. Our design of SR-Latch is shown below in Fig. 61,

Figure 61: Single SR-Latch design

In the above figure SR-Latch is made from two NAND gates. The rising edge of a 'ctrl' signal creates a metastable state. During the metastable state, oscillations are generated by the latch and the counter counts it. The duration of metastable state of a latch is based on the inherent manufacturing variation. Thus entropy harvested is based on the manufacturing variation. The assertion of a 'reset' signal before the rising 'ctrl' signal ensures that the latch is always initialized from the same initial condition. Strong latches are selected for bit generation, the remaining latches are discarded. A latch is regarded as a strong one, if it repeats for the same duration of oscillation during metastable state. Once strong latches are determined, then PUF response bits are generated by comparing the number of oscillations of strong latches.

To achieve area efficiency we configured four latches per CLB of an FPGA as shown below,

Figure 62: Placement of four SR-Latches in a Xilinx CLB

With this design we utilized all the LUTs available inside a CLB, thus achieving a 100% LUT efficiency. We configured 512 latches, which required 128 CLBs. Below are the advantages of our design, we compare it with the state of the art design [53].

Table 29: Advantages of our SR-Latch design.

	Our SR-Latch Design	SR-Latch [53]
Reliability @ 1.20V	99.50%	99.14%
Reliability @ 1.14V	97.54%	94.70%
Reliability @ 1.26V	98.67%	95.20%
Uniqueness	49.24%	49%
Total latches configured	512	128
Total CLBs used for PUF	128	256
Response bit length	256	256
Latch/#CLB	4	0.5
Response bits/#CLB	2	1
Response bits entropy	221	167.9
Response bits entropy/#CLB	1.72	0.65

Below the major differences have been listed between our design and the designs presented for SR-latch PUF in literature.

Table 30: Major difference between our design and other designs

	Our design	Other designs
Source of Entropy	The duration of metastable state is used.	In [53], the final state of the latch after the is used as a source of entropy. In [54], the randomness is harvested by measuring the

		metastability counts. Final bits are generated by averaging the metastability counts and then reading the most significant two (or four) bits. In order to assure the reproducibility of these bits, each latch count is measured 2^{18} times.
Configuration Area	4 latches configured per CLB	In [53], a single latch is configured per pair of CLB.
Encoding Method	PUF response is '1' if the counts of a strong latch L1 is greater than another strong Latch L2 counts. Otherwise it will be '0'.	If the final state of the latch is logic '0', the corresponding response bits are '00'. If the final state is logic '1', the corresponding response bits are '11'. Otherwise it is "10".
Selection of Latches for PUF response generation	Only strong latches are used in PUF response generation. Strong latches are highly repeatable.	In [53], all latches are used. If latch is not repeatable then it is regarded as random and encoded as "10".
Influence of neighboring logic.	No external logic can influence the counts of a latch during metastablitiy.	In [53], the external logic can be configured inside the latch CLB. Therefore, it can affect the counts or the final state of the latch.

In depth details of SR-Latch PUF has been covered in chapter 5.

8.2. Development of new RO-PUF

We developed a novel FPGA friendly Ring Oscillator (RO) based Physical Unclonable Function (PUF). In this design the internal variations of FPGA Look-Up Tables are exploited to generate a PUF response. Statistical tests were performed to study the strength

143

of this PUF. Moreover, stability is compared with the state of the art reported in literature to date. Our design has been tested on 31 Spartan-3e devices and the results are promising with the uniqueness measure of 48.3%, Uniformity 50.13%, Bit-aliasing 51.8% and Steadiness 99.5%. Furthermore, we also analyzed the frequencies to extract the random variation offered by our design.

Below is our design,

Figure 63: Programmable Ring Oscillator (RO) along with the control

In our design each RO is made from a single AND gate and three inverters. It uses one LUT for an AND gate and three LUTs for inverters. We used one LUT-input to connect in a ring, while the remaining LUT-inputs are varied in order to generate programmable delays, as shown in Fig. 63. In Fig. 63, three inverter LUTs in a ring are connected to three delay lines each, while the remaining LUT is tied to two lines. Because we wanted to analyze the maximum effect of delay lines on the ring oscillator frequency, therefore we connected all free LUT inputs to our programmable delay controller. Below are the advantages of our design, we compare it with the state of the art design [34].

Table 31: Major difference between our RO-PUF design and other designs

	Our RO-PUF design	Other designs
Source of Entropy	The frequency of RO varies due to the programmable delay lines. This variation can be harvested to generate more than one bit per RO.	In [34], multiplexers are used to determine the most stable configuration. Each RO has different frequency due to manufacturing variation.
Bits generated per RO	2.17 bits/RO	1 bit/RO in [34].

	Our path outside the LUT is constant for all the configurations.	In [34], different path is chosen for each configuration.
Effect of routing delays		
Bit encoding	Since there are 8 configurations possible for each RO. If two neighboring ROs crosses, then 8 bits are encoded otherwise only 1 bit is added to the PUF response. All the configurations where the $\Delta f<150KHz$, are discarded.	The frequency of each RO is compared with its neighbor. Hence, a single bit is generated per pair of ROs. For a pair of ROs, the frequency chosen is based on the largest Δf between two ROs.

Regarding our new PUF designs we describe the dimensions of PUF space covered in my research as follows,

8.3. **Robustness**

Robustness deals with the ability of the PUF to exhibit 'reliable', 'unique' and 'uniform' responses. In our experiments, we compared our results with the state of the art.

We developed a reliable and efficient SR-latch PUF in this work and compared the results to the state of the art implementation. A novel method of mode calculation is used to determine strong latches. The derived design has been verified on 25 Xilinx Spartan-6 FPGAs (XC6SLX16) and 10 Xilinx Zynq SoC (XC7Z010) devices. The uniqueness is close to the ideal value of 50%. In case of Spartan-6 devices it is 49.24%. Similarly, the uniqueness measure for Zynq devices is 49.87%. The PUF responses exhibit high resistance to temperature and voltage variations. For this purpose, the design has been tested at ± 5% of core voltage and also over the commercial temperature range [0-85°C]. For Spartan-6 devices the reliability at +5% of nominal voltage is 98.67%, while at -5% of nominal voltage it is 97.54%. At both voltages it is more reliable than the start of the art. For Zynq devices the reliability is 94.68% at -5% of nominal voltage and 95.39% at +5% of nominal voltage. We also did the entropy analysis. We calculated bit-dependent bias entropy bound based on PUF responses of 25 Spartan-6 FPGAs. This entropy bound appeared to be equal to 0.959 or 95.9%. Similarly, the pairwise joint distribution entropy bound is equal to 0.866 or 86.6%. We proposed two error correcting schemes for our PUF design. It was shown that the bit flips at extreme voltage and temperature were in the range of our proposed error correction schemes.

8.4.   **Portability**

Portability of PUF design means that different FPGA devices can be configured with the same design. These devices may belong to different FPGA vendors. Similarly devices may belong to same vendor but different families. Lastly devices may belong to the same family

of a particular vendor. These three types of portability can be explained with the following figure,

Figure 64: Three types of portability

As shown above in Fig. 64, the three types of portability are,

1) Intra-Family portability

In this case devices belong to the same family of a particular vendor. As shown above XY shows this type of portability.

2) Inter-Family portability

In this case devices belong to two different families of a particular vendor. As shown above WX shows this type of portability. Two families of Xilinx were chosen for this experiment.

The device specific information was generated using python scripts. Below a VHDL code is shown,

```
Latches_gen: FOR i in 0 to N-1 GENERATE
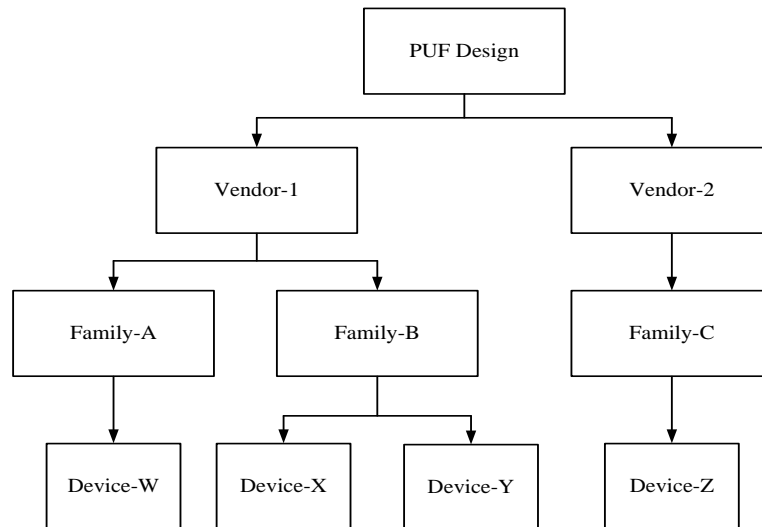
SR_latch_gen: SR_latch generic map(top_slice=>slice_array_top(i), bottom_slice=>
slice_array_bottom(i))

port map(clk=>clk, ff_in=> ff_in(i•4), CE=>CE, CLR=> CLR , Q_out=> ring_osc(i•4)
);

END GENERATE;
```

In the above code SR-latches are instantiated for SR-Latch PUF. The placement of latches is based on the `top_slice` and `bottom_slice` parameters. To make the code portable, the slice locations are generated using python script as shown below in a VHDL code,

```
constant N : integer := 128;

type slice_locations is array(0 to N-1) of string(1 to 12);

constant slice_array_top    : slice_locations := ("SLICE_X13Y0 ",… "SLICE_X23Y63");

constant slice_array_bottom : slice_locations := ("SLICE_X12Y0 ",… "SLICE_X22Y63");
```

All the elements of `slice_array_top` and `slice_array_bottom` are generated using python scripts.

3) Inter-Vendor portability

In this case devices belong to two different vendors namely Xilinx and Altera. According to Fig. 64, YZ shows this type of portability. Zynq devices from Xilinx and Cyclone

devices from Altera were selected for this experiment. Both families are based on 28nm technology. Additionally both have dual core ARM hard cores.

Ideally we planned to develop a vendor independent IP and then add the placement constraints as shown below in Fig. 65,

Figure 65: Ideal portability directed towards FPGA devices

In the above figure a Fitter and Placement constraints are different for each vendor. However, the vendor independent IP cannot be developed for SR-Latch PUF due to the following reasons,

A. Devices from both vendors are significantly different at the lower logic level. Below is an explanation of both devices,

Figure 66: High level block diagram of Adaptive Logic Module (ALM) in Cyclone V

[81]

As shown above in Fig. 66, Cyclone V devices have 8-input LUT and 4 FFs inside each ALM. There are ten ALMs inside each LAB (Logic Array Block). Detailed diagram of an ALM is shown below Fig. 67,

Figure 67: Detailed diagram of an ALM in Cyclone V devices

As shown above in Fig. 67, F0,F1..F3 constitute the Adaptive LUT. Inside each ALM there is a pair of 4-input LUTs and two pairs of 3-input LUTs. Altera calls it a fracturable LUT. It can implement a single 6-input function, a combo of [5-input, 3-input] functions and finally a pair of 4-input functions. The vendor claims that it is more efficient than a single 6-input LUT offered by Xilinx. Altera ALMs are described in [80].

On the other hand the slice of Zynq devices has four 6-input LUTs as shown below in Fig. 68,

Figure 68: High level block diagram of Zynq slice

Each CLB (Configurable Logic Block) has two such slices. In case of SR-Latch we need two LUTs

and two flip flops. These components are highlighted in red color as shown in Fig. 69,

Figure 69: Detailed functional diagram of sliceL in Zynq devices

As evident from the above figures the LUTs are very different for both vendors. Additionally, we need different approaches to configure the same type of design in each vendor.

   B. Below in table 32, we show the primitives required for each vendor to configure an SR-Latch.

Table 32: Primitives used for SR-Latch design

Altera	Xilinx
DFF primitive is used for instantiation of a flip flop.	**FDCE** primitive is used for instantiation of a flip flop.
LCELL primitive is used for LUT instantiation.	**LUT6** is used to instantiate a 6-input LUT.

Table 33: Functionality of Altera DFF primitive

PRn	CLRn	Clk	Din	Qout	Comment
1	1	⌐↑	D	D	Normal Operation
X	0	X	X	0	Clear state
0	1	X	X	1	Pre-Set

Table 34: Functionality of Xilinx FDCE primitive

CE	CLR	Clk	Din	Qout	Comment
1	0	⤒	D	D	Normal Operation
X	1	X	X	0	Clear state
0	0	X	X	No Change	

CE = Clock Enable

Since we use the flip-flop components only for normal operation and clear states, therefore we can connect the CE input pin to PRn in Altera. Similarly CLR is inverted and tied to CLRn. Thus we can achieve a limited portability of flip-flop primitives in both vendors.

As shown in the table above, the FF primitives can be made portable with an inverted 'clock enable' input. However, the LUT primitive are inherently different for both vendors.

Additionally, the placement of primitives in Altera is achieved by modifying the Quartus Setting File (.qsf), as shown below

```
set_location_assignment FF_X4_Y1_N13 -to Latch_0|FF_top

set_location_assignment LABCELL_X4_Y1_N12 -to Latch_0|q

set_location_assignment LABCELL_X4_Y1_N18 -to Latch_0|q_not
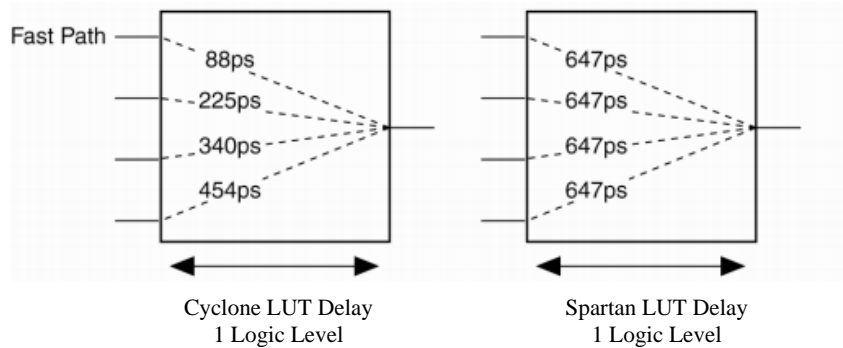
set_location_assignment FF_X4_Y1_N19 -to Latch_0|FF_bottom
```

The placement of components is different from Xilinx.

We implemented a 512 component SR-Latch PUF on cyclone V. Only 8.2% latches showed any sign of metastability. The reason for this low percentage is due to the very small propagation delay of cyclone LUT [82] as compared to Xilinx LUT. As shown below in Fig 70,

Figure 70: Propagation delay of Cyclone vs Spartan LUTs

Due to the very small propagation delay of LUTs, the SR-Latch does not oscillate as explained in [83].

In case of intra-vendor portability the only difference is the placement of components on different FPGAs. We proved that by generating the placement parameters and using a generic VHDL code, we can easily achieve the said types of portability. However, in case of inter-vendor portability, the devices were very different at the lower logic level. We

highlighted the difference and described the challenges we face in this experiment. The goal of this study was to determine the challenges faced to achieve the three types of portability. As we explained the Vendor-Independent IP cannot be developed for SR-Latch. However, we were able to demonstrate the inter-family and intra-family portability.

8.5. **Efficiency**

Efficiency deals with the area consumption of the design, the total power consumed by it and finally the time to calculate the PUF response.

Actual Power is equal to the power consumed by the FPGA device during the generation of PUF response bits. Area is calculated by the tool and this information can be easily extracted from the mapping report generated by tools. It is determined in terms of slice counts in FPGAs. Characterization time for RO PUF is calculated from the following equation,

The total characterization time = (# of components $\cdot$ enable_time)/ (board_frequency)

Where enable_time is the duration in which each component is allowed to run freely.

Changing the efficiency metrics severely affects the quality of PUF. For example, in order to save the PUF area, if two rings oscillators are implemented inside a single CLB (Configurable Logic Block). Then due to the proximity of a neighboring ring, the frequency of two rings might lock with each other. It means that two rings will oscillate with same frequency. Similarly, in order to generate the PUF response quicker, if the characterization

time of a ring is reduced, then the frequency of rings is severely affected. It will result into a very unreliable PUF response bit.

In case of SR-latch PUF when external logic is allowed to configure the latch CLB then the number of counts are badly affected by the external logic. It implies that care need to be taken while reducing the area of a SR-latch PUF.

We developed SR-Latch PUF that is 2x smaller and is more reliable than the state of the art design. To conserve less power, we enable the PUF components only when the PUF-ID is required. There are no free running, ROs or SR-Latches. Our design generates the PUF-ID in a reasonable time of ~5sec.

Table 35: Power consumption of PUF (mWatt)

	Spartan-6	Zynq	Cyclone V	Spartan-3e
SR-Latch PUF	20	1669 1569(PS7) 100(PL)	423.6(PL)	X
RO-PUF	X	X	X	34

PL=Programmable Logic, PS7= Processing System

Table 36: Total Area consumption of PUF (slices)

	Spartan-6	Zynq	Cyclone V	Spartan-3e
SR-Latch PUF	601	674	2785 (ALMs)	X
RO-PUF	X	X	X	727

The figures in table 36, shows the total area consumed. It includes the logic area required

for control unit and all the components in a PUF. It is surprising that the Xilinx tool reports

higher slice count for Zynq device as compared to Spartan-6. In both cases, it is the same design for SR-Latch PUF.

8.6. **List of Contribution**

- Design of a novel SR-Latch PUF for FPGAs.

  - SR-Latch PUF is 2x smaller in area than the state of art.

  - SR-Latch PUF is more reliable than the state of art.

  - SR-Latch uniqueness measure is comparable to the state of the art.

  - Validated on Spartan-6 and Zynq FPGAs.

- Design of a novel RO-PUF for FPGAs.

  - Due to the programmable nature of our RO-PUF, we can generate 2x more bits than the traditional RO-PUF. It implies our PUF requires less chip area to generate the same number of PUF response bits.

  - The uniqueness and uniformity measures of our RO-PUF responses are comparable to the ideal case.

  - The design has been validated on Spartan-3 FPGAs.

  - Frequency analysis of RO components.

  - Determination of systematic and manufacturing variations.

- Implementation and evaluation of PUF architectures on multiple FPGA and SoC platforms.

- Characterization of SR-Latch PUF over 10 Zynq devices and 25 Spartan-6 devices.

- Characterization of RO-PUF over 31 Spartan-3 devices.

- Comprehensive analysis of PUF response generation schemes.

- Development of post processing schemes and their software implementations for analysis and evaluation of various PUF designs.

- Evaluation of the PUF for 'Key generation' application.

- Selection and integration of the most suitable error correcting schemes.

- Development of a final product that generates the PUF response and does all calculations on-chip in real time.

9. Conclusions and future work

We developed a novel SR-latch PUF in this work and compared the results to the state of the art implementation. The latch is designed to keep the effect of routing at minimum and extract the randomness at the same time. Strong and stable latches are selected in this method during enrolment. A novel method of mode calculation is used to determine strong latches. From the circuit efficiency point of view, the proposed design is two times more efficient than the state of the art. The derived design has been verified on 25 Xilinx Spartan-6 FPGAs (XC6SLX16) and 10 Xilinx Zynq SoC (XC7Z010) devices. The uniqueness is close to the ideal value of 50%. In case of Spartan-6 devices it is 49.24%. Similarly, the uniqueness measure for Zynq devices is 49.87%. The PUF responses exhibit high resistance to temperature and voltage variations. For this purpose the design has been tested at $\pm$ 5% of core voltage and also over the commercial temperature range [0-85°C]. For Spartan-6 devices the reliability at +5% of nominal voltage is 98.67%, while at -5% of nominal voltage it is 97.54%. At both voltages it is more reliable than the start of the art. For Zynq devices the reliability is 94.68% at -5% of nominal voltage and 95.39% at +5% of nominal voltage. We also did the entropy analysis. We calculated *bit-*

dependent bias entropy bound based on PUF responses of 25 Spartan-6 FPGAs. This entropy bound appeared to be equal to 0.959 or 95.9%. Similarly, the *pairwise joint distribution entropy bound* is equal to 0.866 or 86.6%. We proposed two error correcting schemes for our PUF design. It was shown that the bit flips at extreme voltage and temperature were in the range of our proposed error correction schemes.

Additionally we developed a new RO based PUF. In this PUF, the programmable delays of FPGA LUTs were used to generate additional bits of an ID. The strength of this design is its ability to generate more than one random frequency per ring oscillator without changing the path of the ring outside LUTs. This solution offers the option to reduce the area requirements of ring oscillator PUFs. To demonstrate the strength of this PUF, it was shown that our design generated more bits per ring oscillator, and these bits are as strong as the ones reported in literature for Configurable Ring Oscillators. The statistical and PUF properties were analyzed and were shown to be very strong from a security point of view.

Apart from developing new and efficient PUF designs, we developed a coherent method to generate PUF responses using five different schemes. We developed a software tool for this purpose. Moreover, the tool is capable to analyze the PUF responses and determine the quality of PUF responses using different metrics. From this work we concluded that PUF responses should be generated using multiple schemes to determine the uniformity and worst cases of

uniqueness and reliability. S-ArbRO-2, Lehmer-Gray and Identity mapping offers the ability to use less number of components for PUF design; however the CRPs are no longer independent. Additionally the effect of systematic variation is not taken into account when Identity mapping is used, especially for RO-PUF. BLG scheme offers the best results in terms of uniformity in Zynq, however it is the S-ArbRO-2 scheme that generates the best uniformity results for both Spartan-3 and Spartan-6 datasets. In case of uniqueness PC scheme offers best results for both Zynq and Spartan-3 datasets, however it is the most expensive scheme in terms of area. In case of Reliability we have no winner as three different schemes offer the best results for three data sets. Lastly, we developed new metrics to determine the quality of PUF responses. These metrics were based on the worst case Uniqueness and worst case Reliability.

In the future we intend to enhance the scope of our method by including the Arbiter PUF data set [79] in our analysis. Similarly we also intend to carryout the entropy analysis of PUF responses. It will include calculating the average binary entropy, worse case Entropy and Joint Pairwise entropy.

A. Publications to date

- B. Habib, J.-P. Kaps and K. Gaj. "**Implementation of Efficient SR-Latch PUF on FPGA and SoC devices**". Journal of Computers & Electrical Engineering, 2015. (Under review).

- B. Habib and Kris Gaj . "**A Comprehensive Set of Schemes for PUF Response Generation**," Proc. 12th International Symposium on Applied Reconfigurable Computing, Rio de Janeiro, Brazil, 22-24 March, 2016.

- B. Habib, J.-P. Kaps and K. Gaj. "**Efficient SR-Latch PUF**", Proc, 11th International Symposium on Applied Reconfigurable Computing, 2015, Bochum, Germany, April 15-17. 2015.

- B. Habib, K. Gaj and J.-P. Kaps. "**FPGA PUF Based on Programmable LUT Delays**", Proc. 16th EUROMICRO Conference on Digital System Design, 2013, Santander, Spain, Sep. 2013.

- J.-P. Kaps, P. Yalla, K.K. Surapathi, B. Habib, S. Vadlamudi, and S. Gurung, "**Lightweight Implementations of SHA-3 Finalists on FPGAs**", Proc. 3rd SHA-3 Candidate Conf., Washington, D.C., Mar. 2012.

- J.-P. Kaps, P. Yalla, K.K. Surapathi, B. Habib, S. Vadlamudi, S. Gurung, and J. Pham. "**Lightweight implementations of SHA-3 candidates on FPGAs**", Progress in Cryptology - INDOCRYPT 2011, Lecture Notes in Computer Science (LNCS), volume 7107, Springer Berlin / Heidelberg, pages 270-289, Dec, 2011.

B.   Course Work in PhD program

- ECE 545 - Digital System Design with VHDL - Fall 2009
- ECE 645 - Computer Arithmetic - Spring 2010
- ECE 681 - VLSI Design for ASICs - Fall 2010
- ECE 511 - Microprocessors - Fall 2010
- ECE 646 - Cryptography and Computer Network Security - Fall 2010
- ECE 682 - VLSI Test Concepts - Spring 2011
- ECE 746 - Advanced Applied Cryptography - Spring 2011
- ECE 899 - Cryptographic Engineering - Spring 2012
- CS 659 - Theory and Applications of Data Mining - Spring 2013
- CEIE 894 - Design and Inventive Engineering - Fall 2013

Bibliography

1. R. Pappu. "Physical One-Way Functions," PhD Thesis, Massachusetts Institute of Technology, 2001.

2. B. Gassend, D. Clarke, M. van Dijk and S. Devadas. "Controlled Physical Random Functions". *In Proc*. 18th Annual Computer Security Applications Conference (IEEE Computer Society, Washington, DC, 2002), p. 149-159.

3. J.W. Lee, D. Lim, B. Gassend, G.E. Suh, M. van Dijk and S. Devadas. "A Technique to Build a Secret Key in Integrated Circuits for Identification and Authentication Application" *In Proc*. Symposium on VLSI Circuits, 2004, pp. 176–159.

4. D. Lim. "Extracting Secret Keys from Integrated Circuits". Master's thesis, MIT, MA, USA, 2004.

5. B. Škorić, P. Tuyls, W. Ophey. "Robust key extraction from Physical Unclonable Functions," *In Proc*. Applied Cryptography and Network Security (ACNS). LNCS, vol. 3531, pp. 99–135. Springer Berlin/Heidelberg (2005)

6. G. Suh, C. O'Donnell, I. Sachdev, and S. Devadas. "Design and Implementation of the AEGIS Single-Chip Secure Processor Using Physical Random Functions". *In Proc*. ISCA, 2005.

7. D. Lim, J.W. Lee, B. Gassend, G.E. Suh, M.van Dijk and S. Devadas. "Extracting secret keys from integrated circuits". *In Proc*. Very Large Scale Integration (VLSI) Systems, IEEE Transactions on 13(10), 1200–1205 (Oct 2005).

8. P. Sedcole and P.Y.K. Cheung. "Within-die Delay Variability in 90-nm FPGAs and beyond", *In Proc*. FPT 2006.

9. P. Tuyls and L. Batina. "RFID-tags for anti-counterfeiting," Topics in Cryptology (CT-RSA). LNCS, vol. 3860, pp. 115–131. Springer Berlin/Heidelberg (Feb 2006).

10. J. Guajardo, S. S. Kumar, G. Schrijen, and P.Tuyls. "FPGA intrinsic PUFs and their use for IP protection," *In Proc*. Workshop on Cryptographic Hardware and Embedded Systems - CHES 2007, Lecture Notes in Computer Science (LNCS), volume 4727, Springer, pp 63–80, 2007.

11. G.E. Suh and S. Devadas. "Physical unclonable functions for device authentication and secret key generation," *In Proc*. 44th Annual Design Automation Conference, New York, 2007.

12. J. Guajardo, S.S. Kumar, G. Schrijen, and P. Tuyls. "FPGA intrinsic PUFs and their use for IP protection", *In Proc*. Workshop on Cryptographic Hardware and Embedded Systems - CHES 2007, Lecture Notes in Computer Science (LNCS), volume 4727, Springer, pages 63–80, 2007.

13. Y Su, J Holleman and B Otis "A digital 1.6 pj/bit chip identification circuit using process Variations". (2008) IEEE J Solid-State Circ 43(1):69–77.

14. C. Bosch, J. Guajardo, A.-R. Sadeghi, J. Shokrollahi, and P. Tuyls. "Efficient helper data key extractor on FPGAs". *In Proc*. 10th International Workshop on Cryptographic Hardware and Embedded Systems, pages 181-197, Berlin, Heidelberg, 2008. Springer-Verlag.

15. M. Majzoobi, F. Koushanfar and M. Potkonjak. "Testing techniques for hardware security," *In Proc*. IEEE international test conference, ITC 2008, pp 1–10.

16. S. S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen and P. Tuyls. "Extended abstract: The butterfly PUF protecting IP on every FPGA", *In Proc*. Hardware-Oriented Security and Trust – HOST 2008, pp 67–70, 2008.

17. Y. Dodis, R. Ostrovsky, L. Reyzin, A. Smith. "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data". SIAM Journal of Computing. 2008, 38(1): 97-139.

18. Maes, R., Tuyls, P., and Verbauwhede, I. "Intrinsic PUFs from Flip-flops on Reconfigurable Devices". *In Proc*. Workshop on Information and System Security – WISSec 2008.

19. J. Bringer, H. Chabanne, T. Icart, "On physical Obfuscation of Cryptographic Algorithms". *In Proc*. 10th International Conference on Cryptology in India, New Delhi, India (Springer, Berlin, Heidelberg, 2009), pp. 88–103.

20. G. Qu ,C. Yin. "Temperature-Aware Cooperative Ring Oscillator PUF". *In Proc*. IEEE International Workshop on Hardware-Oriented Security and Trust, 2009. HOST '09.

21. M. Gora, A. Maiti and P. Schaumont. "A Flexible Design Flow for Software IP Binding in Commodity FPGA". *In Proc*. IEEE Symposium on Industrial Embedded Systems (SIES 2009), July 2009.

22. F. Armknecht, R. Maes, A.R Sadeghi, B. Sunar and P. Tuyls. "Memory Leakage-Resilient encryption based on physically unclonable functions". *In Proc*. Advances in Cryptology (ASIACRYPT). LNCS, vol. 5912, pp. 685–702. Springer Berlin/Heidelberg (2009).

23. A. Maiti, J Casarona, L McHale and P Schaumont. "A large scale characterization of RO-PUF". *In Proc*. IEEE international symposium on hardware-oriented security and trust (HOST) 2010, pp 94–99.

24. V. Vivekraja, L. Nazhandali. "Circuit-Level Techniques for Reliable Physically Unclonable Functions". *In Proc*. IEEE International Workshop on Hardware-Oriented Security and Trust, San Francisco, CA, USA, 27 July 2009, pp. 30–35.

25. B. Škori´c, M.X. Makkes. "Flowchart Description of Security Primitives for Controlled Physical Unclonable Functions". Cryptology ePrint Archive, Report 2009/328, 2009.

26. Y. Hori, T. Yoshida, T. Katashita and A. Satoh. "Quantitative and statistical performance evaluation of arbiter physical unclonable functions on FPGAs". *In Proc*. International conference on Reconfigurable computing and FPGAs (ReConFig) 2010, pp 298–303, Dec 2010.

27. S. Morozov, A. Maiti, and P. Schaumont. "An analysis of delay based PUF implementations on FPGA". *In Proc*. Reconfigurable Computing: Architectures,

Tools and Applications –ARC 2010. LNCS, vol. 5992, Springer, pp. 382–387, 2010.

28. R. Maes and I. Verbauwhede. "Physically unclonable functions: A study on the state of the art and future research directions," In A.R. Sadeghi and D. Naccache (Eds.), Towards Hardware-Intrinsic Security, Information Security and Cryptography, Springer, Heidelberg, 2010, pp. 3–37.

29. A. R Sadeghi, I. Visconti and C. Wachsmann. "Enhancing RFID security and privacy by physically unclonable functions," *In Proc*. Towards Hardware-Intrinsic Security. pp. 281–305. Information Security and Cryptography, Springer Berlin/Heidelberg (Nov 2010).

30. M. Majzoobi, F. Koushanfar and S Devadas. "FPGA PUF using Programmable Delay Lines". *In Proc*. WIFS, 2010.

31. J. H. Anderson. "A PUF design for secure FPGA-based embedded systems". *In Proc*. Design Automation Conference (ASP-DAC), 2010 15th Asia and South Pacific, pages 1–6, jan. 2010.

32. D. Merli, F. Stumpf, C. Eckert. "Improving the Quality of Ring Oscillator PUFs on FPGAs," *In Proc*. Workshop on Embedded Systems Security (2010).

33. C. Costea, F. Bernard, V. Fischer and R. Fouquet. "Analysis and Enhancement of Ring Oscillators Based Physical Unclonable Functions in FPGAs" *In Proc*. 2010 International Conference on Reconfigurable Computing and FPGAs, pp 262–267, IEEE, 2010.

34. A. Maiti and P. Schaumont. "Improved Ring oscillator PUF: An FPGA Friendly Secure Primitive," Journal of Cryptology, volume 24, number 2, pp. 375-397, Oct. 2010.

35. M. Varchola, and M. Drutarovsky. "New High Entropy Element for FPGA Based True Random Number Generators," *In Proc*. Cryptographic Hardware and Embedded Systems, CHES 2010, Lecture Notes in Computer Science Volume 6225, 2010, pp. 351-365.

36. A. Maiti, L. McDougall and P. Schaumont. "The Impact of Aging on an FPGA-Based Physical Unclonable Function". *In Proc*. 21st International Conference on Field Programmable Logic and Applications (FPL 2011), September 2011.

37. X. Xin, J. Kaps, and K. Gaj. "A Configurable Ring-Oscillator-Based PUF for Xilinx FPGAs," *In Proc*. 14th EUROMICRO Conference on Digital System Design – DSD'11, pp 651–657, IEEE, Sep. 2011.

38. M. Yu, D. M'Raihi, R. Sowell, S. Devadas. "Lightweight and Secure PUF Key Storage Using Limits of Machine Learning". *In Proc*. CHES'11 Proceedings of the 13th international conference on Cryptographic hardware and embedded systems.

39. M. Majzoobi, F. Koushanfar, and S. Devadas "FPGA-Based True Random Number Generation Using Circuit Metastability with Adaptive Feedback Control". *In Proc*. CHES 2011.

40. M. Majzoobi, G Ghiaasi, F. Koushanfar and S. Nassif. "Ultra-low Power Current-Based PUF". *In Proc*. International Symposium on Circuits and Systems (ISCAS), 2011.

41. I. Eichhorn, P. Koeberl and V. Leest. "Logically reconfigurable PUFs: Memory-based secure key storage". *In Proc*. ACM Workshop on Scalable Trusted Computing (ACM STC). pp. 59–64. ACM, New York, NY, USA (2011)

42. T. Tuan, A. Lesea , C. Kingsley and S. Trimberger. "Analysis of Within-Die Process Variation in 65nm FPGAs". *In Proc*. 12th International Symposium on Quality Electronic Design (ISQED), March, 2011.

43. S. Schulz, A.R Sadeghi, C. Wachsmann. "Short paper: Lightweight remote attestation using physical functions". *In Proc*. Fourth ACM Conference on Wireless Network Security (ACM WiSec). pp. 109–114. ACM, New York, NY, USA (2011).

44. D. Yamamoto, K. Sakiyama, M. Iwamoto, K. Ohta, T. Ochiai, M Takenaka and K Itoh K "Uniqueness enhancement of PUF responses based on the locations of random outputting RS latches". *In Proc*. 13th international conference on

Cryptographic hardware and embedded systems, CHES 2011. Springer, Berlin, Heidelberg, pp 390–406.

45. A. R Krishna, S. Narasimhan, X. Wang, and S. Bhunia. "MECCA: A Robust Low-Overhead PUF Using Embedded Memory Array". *In Proc*. CHES 2011.

46. R. Maes, A. Herrewege and I. Verbauwhede, "PUFKY: A Fully Functional PUF-based Cryptographic Key Generator," *In Proc*. Workshop on Cryptographic Hardware and Embedded Systems (CHES), 2012, LNCS vol. 7428, pp. 302-319

47. K. Stefan, et. al., "PUFs: Myth, factor busted? A security evaluation of physically unclonable functions (PUFs) cast in silicon," *In Proc*. CHES 2012, LNCS vol. 7428, Springer Berlin/Heidelberg, 2012, pp. 283–301.

48. R. Maes "Physically Unclonable Functions: Constructions, Properties and Applications". PhD Thesis, katholiek universiteit Leuven (2012).

49. G. Hospodar, R. Maes and I. Verbauwhede. "Machine learning attacks on 65nm Arbiter PUFs: Accurate modeling poses strict bounds on usability". *In Proc*. WIFS 2012: pp. 37-42.

50. A. Maiti, I. Kim, and P. Schaumont "A Robust Physical Unclonable Function With Enhanced Challenge-Response Set". IEEE Transactions on Information Forensics and Security, Vol. 7, No. 1, February 2012.

51. D. Ganta and L.Nazhandali. "Easy-to-Build Arbiter Physical Unclonable Function with Enhanced Challenge/Response Set". *In Proc*. Quality Electronic Design (ISQED), 2013 14th International Symposium on Quality Electronic Design.

52. C. Helfmeier, C. Boit, D. Nedospasov, J. Seifert. "Cloning physically unclonable functions". *In Proc*. IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), Jun. 2013, pp.1-6.

53. D. Yamamoto, K. Sakiyama , M. Iwamoto ,K. Ohta , M. Takenaka and K. Itoh. "Variety enhancement of PUF responses using the locations of random outputting RS latches". Journal of Cryptographic Engineering- November 2013, Volume 3, Issue 4, pp 197-211.

54. L. Bossuet, X. Ngo, Z. Cherif and V. Fischer "A PUF Based on a Transient Effect Ring Oscillator and Insensitive to Locking Phenomenon". *In Proc*. IEEE Transactions On Emerging Topics In Computing 2013.

55. P.Z. Wieczorek and K. Golofit. "Metastability occurrence based physical unclonable functions for FPGAs". 2014 Electronics Letters Volume:50 Issue:4.

56. B. Habib, K. Gaj and J. Kaps. "FPGA PUF Based on Programmable LUT Delays," *In Proc*. 16th EUROMICRO Conference on Digital System Design, 2013, Santander, Spain, Sep. 2013.

57. M. Varchola, M. Drutarovsky and V. Fischer. "New Universal Element with Integrated PUF and TRNG Capability" *In Proc*. International Conference on Reconfigurable Computing and FPGAs (ReConFig), 2013. IEEE

58. 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 2013, pp 245-267.

59. T. Rahman, D. Forte, J. Fahrny and M. Tehranipoor. "ARO-PUF: An Aging-Resistant Ring Oscillator PUF Design". *In Proc*. Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014.

60. S. Wei, J. Wendt, A. Nahapetian and M. Potkonjak . "Reverse Engineering and Prevention Techniques for Physical Unclonable Functions Using Side Channels". *In Proc*. Design Automation Conference (DAC), 2014.

61. H. Kang, Y. Hori, T. Katashita, M. Hagiwara and K. Iwamura, "Cryptographic Key Generation from PUF Data Using Efficient Fuzzy Extractors". *In Proc*. 16th International Conference on Advanced Communication Technology (ICACT 2014). IEEE, pp.3–26.

62. D. E. Holcomb and K. Fu. "Bitline PUF: Building Native Challenge-Response PUF Capability into Any SRAM". *In Proc*. Workshop on Cryptographic Hardware and Embedded Systems (CHES), 2014, LNCS vol. 8731, pp. 510–526.

63. Verayo, Inc.: Technology webpage. http://verayo.com/tech.php

64. Intrinsic ID: Product webpage. http://www.intrinsic-id.com/products.htm

65. D. P. Sahoo, S. Saha, D. Mukhopadhyay, R. Chakraborty, and H. Kapoor. "Composite PUF: A New Design Paradigm for Physically Unclonable Functions on FPGA". *In Proc*. IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), 2014, pp.50-55.

66. A. Wild and T. Guneysu. "Enabling SRAM-PUFs on Xilinx FPGAs", *In Proc*. 24th International Conference On Field Programmable Logic And Applications , (FPL) , 2014.

67. M. Dijk and U. Ruhrmair . "PUF Interfaces and their Security", *In Proc*. IEEE Symposium on VLSI (ISVLSI). 2014, pp. 25-28.

68. http://www.xilinx.com/support/documentation/data_sheets/ds162.pdf

69. H. Kang, Y. Hori, T. Katashita, M. Hagiwara and K. Iwamura, "Performance Analysis for PUF Data Using Fuzzy Extractor," *In Proc*. Ubiquitous Information Technologies and Applications, Lecture Notes in Electrical Engineering, Springer-Verlag, Vol.280, pp.277–284, 2014.

70. B. Habib, J. Kaps and K. Gaj. "Efficient SR-Latch PUF", *In Proc*. 11th International Symposium on Applied Reconfigurable Computing, 2015, Bochum, Germany, April 15-17. 2015.

71. http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

72. https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/wp/wp-01252-secure-device-manager-for-fpga-soc-security.pdf .

73. B. Habib and K. Gaj. "A Comprehensive Set of Schemes for PUF Response Generation," *In Proc*. 12th International Symposium on Applied Reconfigurable Computing, Rio de Janeiro, Brazil, 22-24 March, 2016..

74. https://cryptography.gmu.edu/puf/

75. B. Habib, J.-P. Kaps and K. Gaj. "Implementation of Efficient SR-Latch PUF on FPGA and SoC devices". Journal of Computers & Electrical Engineering, 2015. (Under review).

76. Dodis, Y., Reyzin, L., Smith, A.: "Fuzzy extractors: How to generate strong keys from biometrics and other noisy data". In: Proc. EUROCRYPT 2004.

77. http://www.xilinx.com/support/documentation/data_sheets/ds187-XC7Z010-XC7Z020-Data-Sheet.pdf.

78. http://rijndael.ece.vt.edu/puf/script_download.html.

79. https://staff.aist.go.jp/hori.y/en/puf/index.html

80. https://www.altera.com/en_US/pdfs/literature/wp/wp-01003.pdf

81. https://www.altera.com/en_US/pdfs/literature/hb/cyclone-v/cv_51001.pdf

82. https://www.altera.com/ja_JP/pdfs/literature/wp/wpcycsptn3pa.pdf

83. Kacprzak, T.: "Analysis of Oscillatory Metastable Operation of an R-S Flip-Flop". IEEE Journal of Solid-State Circuits 23(1), 260–266 (1988).

Biography

Bilal Habib received his Bachelor of Science in Computer Engineering from University of Engineering and Technology, Pakistan in 2005. He worked for two years at And-Or Logic, Islamabad, Pakistan as a design engineer. He received his Master of Science in Computer Engineering from George Washington University in 2010. He joined the Cryptographic Engineering Research Group at George Mason University as a PhD student in 2010. He completed his PhD in 2016.